IEEE Access
Multidisciplinary | Rapid Review | Open Access Journal

Received November 4, 2020, accepted November 15, 2020, date of publication November 19, 2020,
date of current version December 9, 2020.

Digital Object Identifier 10.1109/ACCESS.2020.3039388

An Enhanced Multi-Modal Recommendation Based on Alternate Training With Knowledge Graph Representation

YUEQUN WANG[1,2], LIYAN DONG[1,2], HAO ZHANG[1,2], XINTAO MA[1,2], YONGLI LI[3], AND MINGHUI SUN[1,2]

[1]College of Computer Science and Technology, Jilin University, Changchun 130012, China
[2]Key Laboratory of Symbolic Computation and Knowledge Engineering of Ministry of Education, Jilin University, Changchun 130012, China
[3]School of Information Science and Technology, Northeast Normal University, Changchun 130117, China

Corresponding author: Minghui Sun (smh@jlu.edu.cn)

This work was supported in part by the National Natural Science Foundation of China under Grant 61872164, and in part by the Program of Science and Technology Development Plan for Jilin Province, China, under Grant 20190302032GX.

ABSTRACT Deep network recommendation is a cutting-edge topic in current recommendation system research, which as a combination of recommendation systems and deep learning theory can effectively improve recommendation accuracy. In a real recommendation scenario, all the effective information in a data set should be extracted, both explicit and implicit, because the comprehensive degree of information is proportional to the recommendation performance. This article proposes an enhanced multi-modal recommendation based on alternate training with knowledge graph representation (SI-MKR) based on the MKR deep learning recommendation model. Our framework is an enhanced recommendation system based on knowledge graph representation, using valuable external knowledge as multi-modal information. The SI-MKR model solves the problem of ignoring the diversity of data types in the multi-modal knowledge-based recommendation system, which adds user and item attribute information from a knowledge graph as an enhancement recommendation multi-tasking training. By analysing the content of the item and user attributes, the SI-MKR model classifies the attributes of the items and users, processes the text type attributes and multi-value type attributes separately for feature extraction, and other types of attributes are used as inputs to the knowledge graph embedding unit. In addition, the knowledge graph data form a triplet unit, thus continuing the knowledge graph data training process. The feature extraction unit of the knowledge graph and the recommended unit are connected through the cross-compression unit for alternate training. During the deep learning framework training process, the recommendation system's item has a potential correlation with the head entity in the knowledge graph which embodies the idea of multi-tasking. Through extensive experiments on real-world datasets, we demonstrate that SI-MKR achieves substantial gains in movie recommendation over advanced model baselines. Even user-item interactions are sparse, SI-MKR maintains better performance than the MKR model.

INDEX TERMS Feature learning, graph representations, knowledge graph, recommendation system.

I. INTRODUCTION

With the development of the digital age, the amount of data has grown explosively. Extracting useful information from massive amount of data has become a popular research topic. The first recommendation system was proposed by Resnick in 1997 [1] to recommend corresponding items or web pages to users according to their behavioural data. A recommendation system can recommend suitable items for users by evaluating user preferences through item or user portraits. The recommendation algorithm is the core element of recommender systems, which is mainly categorized into collaborative filtering (CF)-based recommender systems, content-based recommender systems, and hybrid recommender systems [2].

However, with the improvements in data attributes, the content within data has become increasingly detailed, so traditional recommendation systems are inept at

The associate editor coordinating the review of this manuscript and approving it for publication was Chang Choi.

unearthing deeper links among data, such as in click-through rate (CTR) models for news sites. A traditional recommendation system can only conduct collaborative filtering recommendations based on whether users click on the news when the news are published, or the news collection the user clicks on [3]. However, this method cannot deeply ascertain the potential content from the news site. Many researchers have integrated other technologies into recommendation systems to deeply extract the features of users and items. To date, representation learning methods and deep learning methods have grown relatively mature in recommendation systems.

Representation learning refers to the vectorization of the data in a recommendation system. The data mainly includes sequences [4], graphs [5] or other construction methods. Sequence construction builds a user's historical behavior data (such as click data, subscription data, etc.) into a sequence and then learns through a skip model and negative sampling technology where the RNN model is a typical representative in this area.

Graph construction methods form user behaviors into a graph structure. Utilizing the mutual combination of graph topology and walking algorithms, a sequence is obtained by using a random walk algorithm. Standard walking algorithms include the Deepwalk [6] algorithm, node2vec algorithm [7], and Line algorithm [8], and then representation learning can be performed through the Skip-gram [9] algorithm and negative sampling algorithm.

In recent years, knowledge graphs (KGs) are widely used in recommender systems (i.e., KG-based recommendation) due to their comprehensive auxiliary data for effective recommendation. A KG is a heterogeneous graph where nodes function as entities and edges represent relations between entities. Items and their attributes can be mapped into the KG to understand the mutual relations between items [10]. Moreover, users and user attributes can also be integrated into the KG, which allows relations between users and items, as well as user preferences, to be captured more accurately.

Knowledge graphs contain rich semantic associations between entities and provide a potential source of multi-modal information for recommendation systems. The introduction of a knowledge graph into a recommendation system can bring the following characteristics to the recommendation system:

(1) The knowledge graph introduces more semantic relations for items and can find user interests deeper.

(2) A knowledge graph can connect user history and recommendation results to improve user satisfaction and acceptance of recommendation results and enhance user trust in the recommendation system.

(3) A knowledge graph is conducive to the divergence of recommendation results

A. MULTI-MODAL INFORMATION

Multi-modal information is often integrated into recommendation systems to improve recommendation accuracy or solve cold start problems. Multiple side information here can be understood as multi-modal information, such as the brand of an item, the name of the store, the category, and so on. This learning method is called graph embedding with side information (GES). Common multi-modal information recommendation systems include the factorization machine (FM) model [11], the FFM model [12], and the logistic regression (LR) model. For example, in the FM model, features are combined to judge the probability of two features appearing at the same time and are calculated as the weights. PNN [13], a fully scaled product-based neural network, asserts that the cross-feature expression learned after embedding into MLP is insufficient. Some scholars proposed a product layer concept, which is based on a multiplication operation to reflect a DNN network with cross-features. The FM model finds the combinatorial relationship between two features by means of hidden variables; however, this is limited to the combinatorial relationship between two features. Later, a deep neural network was developed to mine the combinatorial relationship of features at a higher level. Before the use of neural networks, gradient boosting decision trees (GBDTs) was also an effective way to find feature combinations.

Multi-modal representation learning refers to the presentation of various data in the form of data sequences, which then allocates the weight through an attention mechanism to obtain the final embedding sequence. For example, information such as knowledge graphs, user portraits, content understanding, posters and even voices are taken as model inputs to generate the final vector representation jointly. Multi-modal learning can be applied in the fields of speech [14], image [15], variable decoding [16] and multimode automatic coding [17] recommendation. Various recommendation techniques, including deep learning, natural language processing, and image processing, are cleverly integrated to improve the prediction accuracy and find the relation of intrinsic attributes. Facebook's 2014 article solved the LR feature combination problem through a GBDT [18]. In recent years, Ali has published many recommendation algorithms in traditional fields, such as the MLR algorithm [19], as well as deep-learning fields, such as the entire space multi-task model and deep interest network. At the same time, Ali cooperated with Tsinghua University to explore the field of reinforcement learning and proposed the MARDPG algorithm. Moreover, another important recommendation system model was proposed by Ali: the Deep Interest Network (DIN) [20]. This method consists of accurate directional retrieval and basic algorithms, which makes full use of the information in the historical behaviour data of users to improve CTR estimation performance.

It can be seen that during the recommendation process, the main concept and research content of deep recommendation is to conduct relationship mining on existing attributes to obtain hidden attributes. During the item recommendation process, the difficulty degree of the recommendation process is related to the feature extraction of the items and users. The more accurate the feature extraction is, the more complex the recommendation process will be. In this article, we use

a knowledge graph (KG), user attributes and item attributes as multi-modal information to make deep multi-modal recommendations.

The KG is a practical approach to represent large-scale information from multiple domains. A common way to describe a KG is to follow the resource description framework (RDF) standard [21], in which nodes represent entities, while edges in the graph function represent relations between entities. Each edge is represented in the form of a triple (head entity, relation, tail entity), also known as a fact in the graph, implying the specific relationship between the head entity and tail entity. Multimodal knowledge graph introduces other modal information into traditional knowledge graph, which enriches the types of knowledge. Entity description can provide important textual information for knowledge representation learning. Most traditional methods only learn knowledge representation from structured triples and ignore the various data types (such as text) that are often used in the knowledge base. It is common to use knowledge graphs as multi-modal information in recommendation systems. Combining feature learning of knowledge graphs and recommendation systems typically follows sequential training, joint training or alternate training. Sequential training methods mainly include DKN [22]; the Ripple Network is the primary method for joint training; alternative training mainly adopts a multi-task concept, and the main methods include MKR [23].

There are two network inputs for a DKN: candidate news set and the news title sequence clicked by the user. The input data are extracted through the KCNN. Besides, an attention layer is used to calculate the candidate news vector's attention weight and the user click history vector. After joining the two parts of the vectors on the top layer, a DNN is used to calculate the user's probability of clicking the news. Embedding each word in the title and the entity corresponding to each word in the title are realized in three aspects. Thus, the embedded context of each word is obtained. The model for each word can be realized through the pretraining of word2vec. However, the entity embeddings are required in advance of using the DKN, causing the DKN to lack an end-to-end method of training. Another concern about the DKN is that it has difficulty incorporating side information other than texts. To improve the accuracy of the recommendation, these ignored attributes should be included in the model. Therefore, different project attributes should be addressed separately.

In Hongwei Wang's article on multi-task feature learning for knowledge graph enhanced recommendation (MKR), he noted that DKN could not be trained end-to-end, and a RippleNet [24] representation of the relation vector is not sufficient. He thus put forward the MKR model.

MKR is a general, end-to-end deep recommendation framework that aims to assist recommendation tasks with knowledge graph embedding (KGE). The two tasks are independent of each other, but they are highly related due to the interrelationship between the items in an RS and the entities in a KG. The whole framework is trained by alternately optimizing two tasks, which endows MKR with high flexibility and adaptability in a real recommended scene.In addition, Hongwei Wang proposed a KGCN [25] combined with a graph convolution neural network.

RippleNet focuses on the expansion of user history, whereas KGCN focuses on the expansion of item entities. In the same year, Hongwei Wang proposed KGNN-LS [26], which is an improvement of the KGCN that adds label smoothness (LS) after the GCN to improve the robustness of the model, preventing overfitting problems; the effect is ideal. Xiang Wang proposed the knowledge graph attention network [27] (KGAT) in 2019, which uses recursive neighbour propagation to learn node embedding and an attention mechanism to distinguish the importance of neighbour embedding.

B. CONTRIBUTION

Three methods of knowledge graph feature learning in a recommendation system have been introduced above. However, existing KG-based recommendation methods largely ignore the multimodal information, such as images and text descriptions of items. Those visual or textual features may play a significant role in recommendation systems. For instance, before watching a movie, the user first needs to see the type of movie and the information in the movie title. The SI-MKR model proposed in this article is a model improvement based on MKR, which retains the multi-task training idea of the MKR model, and combines the knowledge graph training unit and the recommendation system training unit through cross-compression unit. Knowledge graph data is an additional data source, however, in the training process, due to the highly structural similarity between the form of the knowledge graph triplet and the user-item score information in the recommendation system, the data of the items and users can correspond to the head vector in the knowledge graph triplet. But not all kinds of attributes are suitable to be represented by knowledge graphs because some attributes have text information. If we embed the text content attribute of an item directly into an ID, the text content's deep information will be vacant. In addition, multivalued attributes contain multiple types of attribute values, which are also not suitable to be represented as knowledge graphs. For example, a movie has multiple types, and the relationship between a movie and its type is one to many. One to many or text-rich content attributes' potential content is lost during the knowledge graph embedding process. Therefore, it is necessary to classify the attributes in the training process. At the training level, this means a combination of sequential training and alternate training. In this section, the MKR model and DKN model will be discussed at a deeper level, and based on these two models, a fusion of training methods will be carried out to maximize the information function of the knowledge graph.

The MKR model is divided into three units: a recommendation unit, a knowledge graph unit and a cross-compression unit. The recommendation unit uses the rating data as a recommendation, and the knowledge representation model uses the form of a triple to represent the item attribute. Taking movie recommendation as an example, the knowledge graph

is expressed in the form of a triple (movie ID, director, director name). In fact, the item vector and entity vector are two descriptions of the same object. The sharing of information between them reveals extra information from each other through the cross-compression unit, which remedies the data sparsity problem.

Items and users attribute values are added as side information in the recommendation system, the (item,attribute type,attribute value) need to be constructed in the form of knowledge graph triples. However, MKR treats text properties (such as movie titles) as a common attribute, wasting the intrinsic meaning of the attribute value itself. In the MKR model, only item attributes are considered, while user-related attributes are ignored.

In this article, we propose an SI-MKR model based on the MKR model to compensate for the limitations of the above works. Similar to the MKR model, our model also has multiple units: a recommendation module, a knowledge graph module and an intermediate cross-compression unit, besides, our model also adds a feature extraction unit.

In the feature extraction unit, classification discussion should be conducted between users and items. In the MKR model, there is no corresponding processing of user attributes. The cross-training method is completely adopted for the extraction of project attribute features, ignoring the intrinsic relationship between project text attributes and multi-value attributes. For the SI-MKR model, all user attributes need to be MLP processed, and all attributes such as user ID, gender and occupation need to be trained, studied and integrated to finally output the user feature vector. For projects, the attributes need to be learned alternately with the KGE module. The attributes of items only use text type attributes and multi-value type attributes as inputs for feature extraction, then output corresponding item feature vectors through MLP and carry out cross-training with the KGE module.

In the recommendation unit, we use the user, the item and their attributes as inputs. We obtain the end user embedding using a multilayer perceptron (MLP) to extract the user characteristics. Similarly, for end item embedding, we first use a multilayer perceptron (MLP) to extract item attribute features and then extract the item characteristics using the cross-compression unit. Next, we put the features through another MLP, and finally, we obtain the probability prediction results.

The knowledge graph unit takes a head and relation as inputs, uses a MLP to extract relation features, uses the cross-compression unit to extract head features, uses the head and relation to calculate the representation of the prediction tail, and then uses a function f to calculate the similarity between prediction tail and the actual tail as the performance score for knowledge graph embedding (KGE) link prediction.

The intermediate cross-compression unit is the key to connecting the recommendation module with the KGE module. This unit can automatically learn the high-order interaction characteristics of RS items and entities in a KG.

The theoretical analysis shows that the cross-compression unit can represent higher-order characteristic interactions between the item and the entity.

SI-MKR is a generalized framework that covers several typical recommendation systems and multi-task learning methods, including factorization machines, text convolution networks, cross-stitch networks, and deep recommendation frameworks. The system is verified by the MovieLens dataset, and the experimental results show that the average AUC as well as the accuracy and recall rate are improved.

II. OUR APPROACH

In this section, we introduce several concepts and models related to this work, reasonably allocate item attributes, select appropriate attributes for knowledge representation, and use other item attributes and user attributes to perform representation learning. We also show how to improve the corresponding cross-compression unit.

A. PROBLEM FORMULATION

During the actual item recommendation process, the data can be divided into item datasets, user datasets and user-item interaction datasets. The interactive dataset can be represented as triples (i, u, r), and $i \in I\,(i_1, i_2, i_3 \cdots i_n)$, $u \in U\,(u_1, u_2, u_3 \cdots u_m)$. I represents the set of items, U represents the set of users, and R is the $n \times m$ rating matrix. Each item i_n has several attributes, each of which corresponds to its feature value. These feature values can be understood as the context of the item, that is, an item can be represented by these feature values. Similarly, each user also has n feature values, which also represent the context of the user and thus represent the user. For item $\mathrm{i} \in \mathrm{I}\,(\mathrm{i}_1, \mathrm{i}_2, \mathrm{i}_3 \cdots \mathrm{i}_n)$, each item has the same number of attribute characteristics. Therefore, the combination of items can also be expressed as $I = [A1, A2, A3 \ldots Ax]$. This represents that item I contains x attributes. Then, the attribute matrix of the item has the form of $n \times x$. As shown in Figure 1, each item has five groups of attribute values.

B. ATTRIBUTE CLASSIFICATION

In a traditional MKR model, these attributes and items need to be converted to triples to create a knowledge graph. For example, the A_1 attribute of item i_1 has an attribute value of a_{11}. In the triplet of the knowledge graph, this relation is expressed as (i_1, A_1, a_{11}). However, as discussed above, not all item attributes are suitable to be converted into knowledge graph triples. Therefore, we divide item attribute set A into three types according to the form of item attribute A, namely, text type attribute A^T, multi-value composite attribute A^M and other type attribute A^E, i.e., $\mathrm{A} = A^T \cup A^M \cup A^E$.

For an attribute of type A^{T}, its value is generally composed of sentences containing multiple words. Notably, text attributes have practical meaning. The text attribute values of different items also have some relevance through the actual meaning of the text. If the attributes are directly expressed in the form of a triple, the text attribute value will be IDs in the

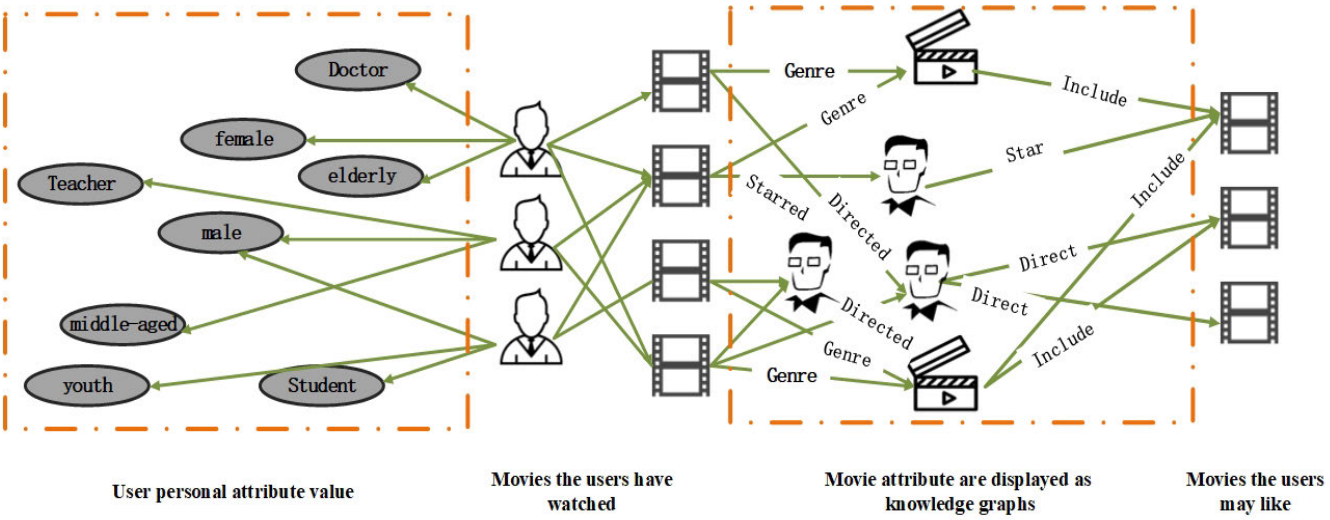

FIGURE 1. User attributes and item attributes.

training process that only consider the structural relationship between attributes but without the internal relationship of attributes. The translation of text attributes into triples will lose some of the most important meanings themselves. Therefore, training with triple knowledge graphs is not an ideal option. Instead, we can train the attribute values with the CNN model, vectorize the text attribute, and obtain the embedding. Finally, the value of embedding is passed as an input to the recommendation module of the model.

On the other hand, A^M is a multivalued type of attribute, that is, a combination of subattributes of various types. Taking the movie as an example, a movie can be a combination type of comedy, romance and horror. If this attribute is represented as a triple as an entity in the knowledge graph, the whole attribute will lose its meaning. For example, two movies have three subtypes, two of which are of the same type. If we train according to the knowledge graph triplet, the relation between the subattributes is vacant. Therefore, similar to text attributes, deep training should be conducted in the form of knowledge representation to obtain embeddings, as inputs fed into the recommendation module. Other item attributes can be directly converted to knowledge graph triples, which will eventually be sent to the knowledge graph KGE unit of the SI-MKR model for training or converted to embeddings as input for the recommendation system unit.

The MKR model only focuses on expanding item entities, which are expressed in the form of knowledge graphs. However, no corresponding expansion of the user entity is implemented. For our SI-MKR model, the extension of user entity attributes is also added. There is no need to classify the user entity attributes because they do not need to interact with the knowledge graph unit. For all the attributes of the user, methods of representation learning are adopted. All user attributes are encoded in the form of one-hot coding, and the final user embedding is obtained through deep learning framework training. User features are extracted by a multilayer perceptron (MLP) to obtain the users' embedding and finally interact with the item embedding.

C. FRAMEWORK

The traditional MKR model consists of three main components: a recommendation module, a KGE module, and a cross-compression unit. The SI-MKR model proposed in this article mainly consists of four parts. In addition to the recommendation module, the KGE module, and the cross-compression unit proposed by the MKR model, the SI-MKR model also adds a feature extraction unit. Feature extraction units function on a recommendation module for the feature representation of the users and the items. SI-MKR architecture is shown in Figure 2. Our SI-MKR improves the recommendation module of the system. The item and user attributes are represented and learned. Then, we obtain the vector representation of the user and the item by using the vectorization of attribute values.

1) FEATURE EXTRACTION UNITS

The input of the recommendation module in MKR includes two raw feature vectors u and v that describe user u and item v. For the SI-MKR model proposed in this article, user u and item v are learned for knowledge representation. For this purpose, the feature presentation layer is added on top of MKR. We adopt the techniques of text convolution and feature vectorization to calculate the feature data embeddings of the knowledge graph.

For this module, classification discussion should be conducted between users and items, but the feature extraction units of users and items can be represented in Figure 3. In the MKR model, there is no corresponding processing of user attributes, and the method of cross-training is completely adopted for the extraction of project attribute features,

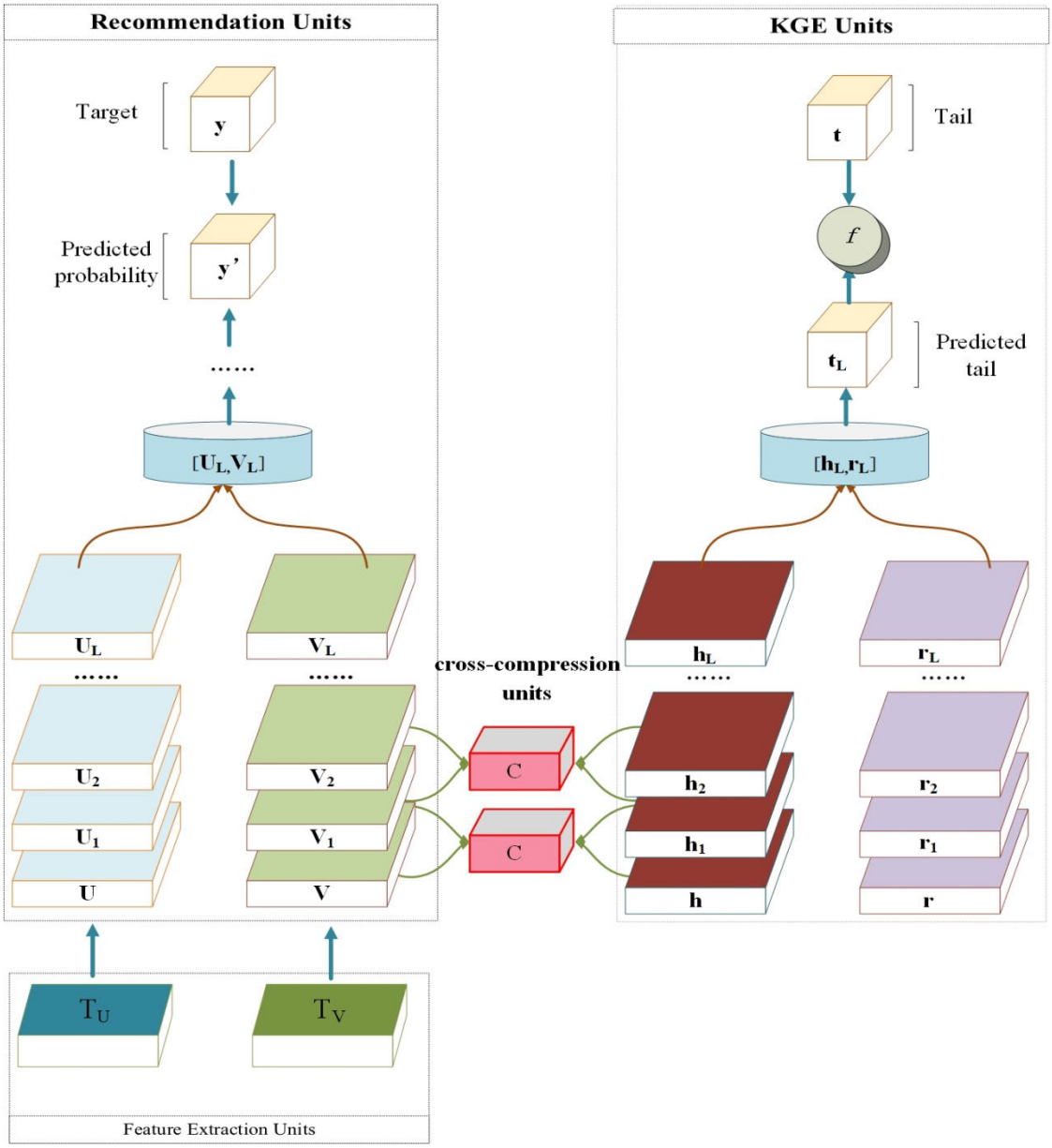

FIGURE 2. SI-MKR network structure.

ignoring the intrinsic relationship between project text attributes and multi-value attributes. For the SI-MKR model, all user attributes need to be MLP processed. All attributes such as user ID, gender and occupation need to be trained, studied and integrated to output the user feature vector finally. For projects, the attributes need to be learned alternately with the KGE module. The attributes of items only use text type attributes and multi-value type attributes as inputs for feature extraction, then output corresponding project feature vectors through MLP and carry out cross-training with the KGE module.

The ID, text type attribute, and multi-value type attribute of each user or item are further processed to obtain the corresponding user u and item v characteristics. In this article, we use the embedded layer and the full connection layer to combine the information of various users u and v together to form items or users' characteristics.

First, the text CNN model for text extraction is introduced. The structure of the model is shown in Figure 4. The methods used to solve text feature extraction include natural language processing (NLP), Markov networks, maximum entropy models, conditional random fields, and cyclic neural networks, which are often used to extract relevant text features in traditional NLP algorithms. Because the target data are a text sentence with a single structure, a convolutional network has the advantage of fewer parameters to

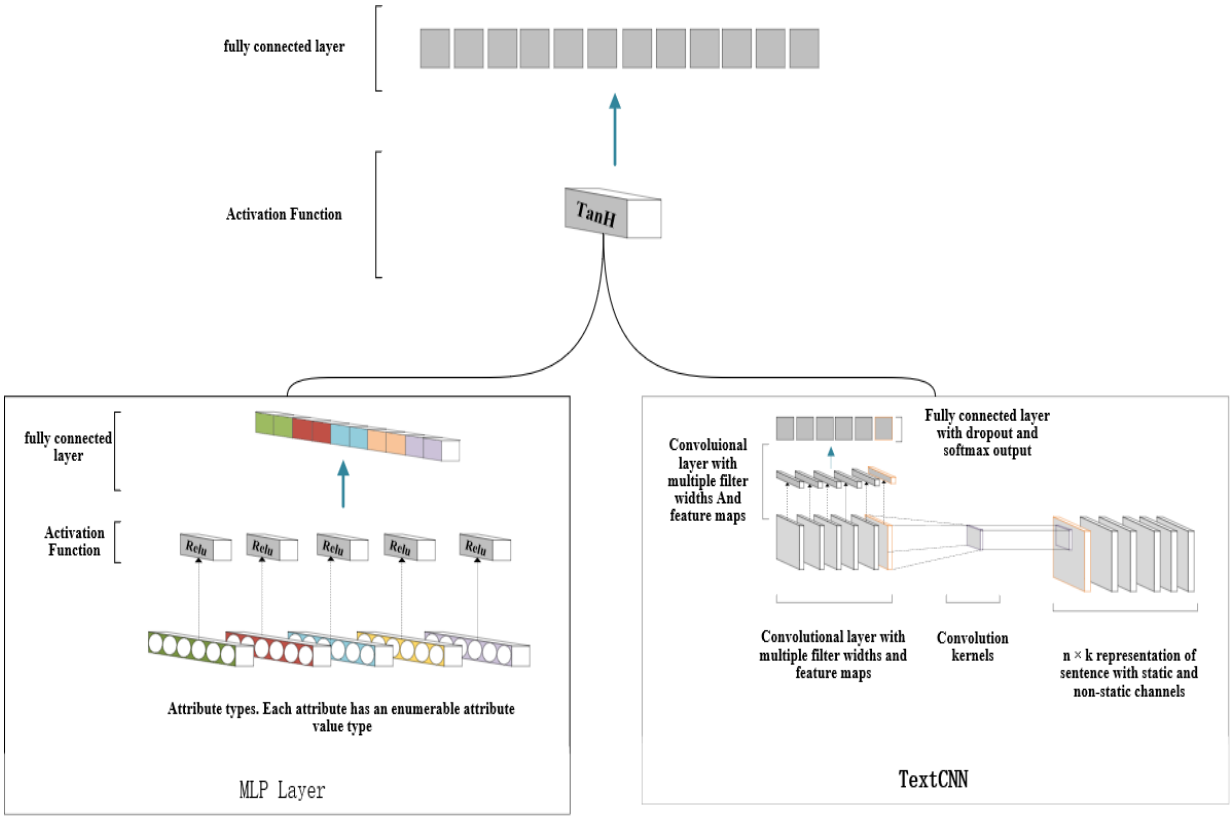

FIGURE 3. SI-MKR network feature extraction units. The left side is general feature extraction, and the right side is text feature extraction.

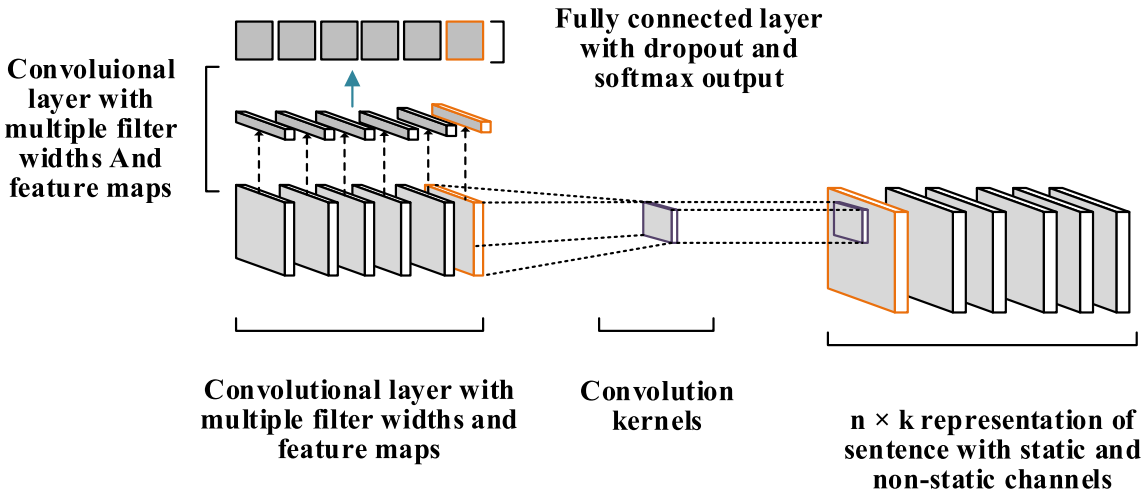

FIGURE 4. Text CNN Model.

be updated at each iteration, thus increasing the computing speed compared with other method blocks, which saves considerable computing resources when processing text data. In this article, the text data characteristics are extracted by using the concepts of natural language processing and text convolutional networks.

A convolutional neural network (CNN) uses layers with convolutional filters to extract items' local features. The CNN model was initially been invented for computer vision and later proved to be useful for NLP and achieved good results in semantic analysis, search and query retrieval, sentence modelling, and other traditional natural language processing tasks.

The greatest difference between a text convolutional network and an image convolutional network lies in the design of the convolution kernel. The convolution kernel of an

image is usually $2 \times 2, 3 \times 3, 5 \times 5$ [28]. It is applied in the characteristic graph to slide and convolute with a certain number of pixels. The text convolution needs to cover the embedding vector of the whole word, so the convolution kernel (filter) of the whole line matrix (word) is typically used. The width of the input matrix is generally the same as that of the convolution kernel. We convert words into word vectors, where the length is the width of the matrix [29]. The width of the matrix is actually converted from one word by cross-multiplying word vectors with the convolution kernel. Various convolution kernels are available, whose sizes are generally expressed as (number of words, vector dimension), generally sliding from a two to five word window at a time.

In essence, text feature extraction is also a natural language processing task, so we design a simple text convolutional network. This model architecture is an improvement of the text convolutional neural network architecture proposed by Collabert *et al* [30]. Figure 4 shows a CNN structure for natural language processing.

For an $n \times K$ size text feature, each line is the feature vector of a word. We use two convolution kernels of different sizes: sliding and fixed. In this structure, the convolution kernel size is set to 2, 3, 4, and 5, i.e., $2 \times k$, $3 \times k$, $4 \times k$, $5 \times k$, where k is the length of the embedding. It is necessary to count the maximum value of each characteristic graph by maximum pooling. Every feature vector is concatenated to become a feature vector. Finally, dropout is used in the full connection layer for regularization. Then the softmax layer is used to receive this feature vector as input, classifying sentences and describing possible output states. Therefore, we describe the calculation process according to the above model as follows:

First, we embed text information in a matrix, and the matrix of each behaviour is a word element. Assume that a total of seven words, where each word is a five-dimensional vector, results in a $7 * 5$ matrix. This matrix is equivalent to an "image" and used for a convolution operation in the convolutional layer. Assuming that there are m words in total, and each word can be converted into a k-dimensional vector, the word list can be expressed as $m \times k$.

$$\begin{bmatrix} x_{11} & \cdots & x_{1i} & \cdots & x_{1k} \\ x_{21} & \cdots & x_{2i} & \cdots & x_{2k} \\ \vdots & \cdots & \vdots & \cdots & \vdots \\ x_{m1} & \cdots & x_{mi} & \cdots & x_{mk} \end{bmatrix}$$

Let $x_i \in R^k$ be the k-dimensional word vector corresponding to the ith word in the sentence. Then, a sentence of length n can be expressed as:

$$x_{1:n} = x_1 \oplus x_2 \oplus \ldots \oplus x_n \tag{1}$$

where "$\oplus$" is the join operator. In general, $x_{i:i+j}$ is used to represent the connected words $x_i, x_{i+1} \ldots x_{i+j}$. The convolution operation of characteristic c_i is obtained from the $x_{i:i+h-1}$ word window as follows:

$$c_i = f(w \cdot x_{i:i+h-1} + b) \tag{2}$$

where w $\in R^{hk}$ is the convolution kernel, $b \in R$ is the bias term, and $f(\cdot)$ is the nonlinear function (activation function). The convolution kernel is applied to every possible word window in the sentence to obtain the feature figure c $\in R^{n-h+1}$ of this layer, as shown below:

$$c = [c_1, c_2, \ldots, c_{n-h+1}] \tag{3}$$

Then, we use the maximum pool operation to capture the most reflective characteristics of the value of c^=max{c}. With the help of regularization dropout, we obtain the characteristics of the movie name. The process described above aims to extract a feature from a filter (convolution kernel) [31]. We use multiple convolution kernels (with different window sizes) to obtain multiple features. These features are formed at the second layer from the bottom and passed to a dropout and a full connection layer, yielding a text-type attribute feature vector. The characteristics are then merged with the first fully connected layers of the recommendation algorithm.

Assuming that there are m neurons in the full connection layer, after the ReLu activation function, we obtain a vector of fixed size, that is, a text feature vector for learning. The calculation formula is as follows:

$$t_j = cnn(W, Y_j) \tag{4}$$

After describing the processing of text attributes, we introduce the processing method of multi-value type attribute A^M, which is similar to that of text type attribute A^T, on the sense of the preprocessing of attributes. For multivalued attributes, we need to enumerate all the attribute values and assign the attribute with one-hot encoding. The architecture diagram for this process is shown in Figure 3 to the left.

Suppose the attribute value of the item is $x : \{x_1, x_2 \ldots x_m\}$, where x_i represents one of the item attributes. As described earlier, it can be a multi-value type attribute or any other type. In the case of movies, the genres of the movie can be a multivalued attribute, and the movie's ID is an attribute of other types. The attributes of the user can be expressed as $y : \{y_1, y_2 \ldots y_n\}$, where y_i represents one of the user attributes. Then, we input the attributes of the user and the item into the embedding layer, obtaining the feature vectors $\bar{x}$ and $\bar{y}$ of the user and the item attributes:

$$\bar{x} = f(w_1 x + b_1) \tag{5}$$

$$\bar{y} = f(w_2 y + b_2) \tag{6}$$

where w_1 and w_2 represent the weight, b_1 and b_2 represent the bias, and $f(\cdot)$ represents the activation function. The various types of attribute processing are described below.

The full connection layer is used to vectorize the characteristics of attributes, through which the final embedding of the user and the item are expressed as:

$$u_i = concateate(\bar{x}) \tag{7}$$

$$p_i = concateate(\bar{y}) \tag{8}$$

For these two kinds of data for multi-value type attributes and IDs, the traditional method converts them into a one-hot

encoding form; however, this can introduce data sparsity. Therefore, to avoid the increase in computational effort and the difficulty of feature extraction caused by data sparsity, we adopt an index matrix and embedded layer to transform these data.

Suppose attribute Y is a multivalued type attribute that has m attribute values. We argue that a piece of data has multiple values of this attribute. For example, a piece of movie data can have multiple type labels. Attribute Y of item I is represented by [Y_1, Y_2, Y_n. . .]; for example, the genres attribute of film I is composed of three attributes: comedy, thriller and love. We index these m attributes in the form of continuous numbers. Thus, the embedding matrix uses sequences of 1∼m as the index. For a certain item, the attribute can be expressed as a d-dimensional vector.

The attribute Y of item I can be expressed as the following formula:

$$Y_i = Y_1 \oplus Y_2 \oplus \ldots \oplus Y_n \tag{9}$$

where $Y_i \in R^{1\times d}$, and we use an initial vector representation of a d dimension for all item Y attributes.

For the processing of item attributes, we only need to process the text type and multi-value type attributes. For other types of attributes, we can directly represent them in the form of triples and train them through the SI-MKR model's knowledge graph unit. However, for user attributes, all attribute values need to be processed in the form of a multilayer perceptron (MLP) because there is no knowledge graph unit for user attribute processing in the SI-MKR model.

W represents all the weights and bias variables, Y_j represents the original text information of learning resource j, and t_j represents the text feature vector of learning resource j.

Formula (10) shows that the attribute feature of learning resource j is p_j, and the text feature of learning resource j is t_j. According to formula (4), the feature vj of learning resource j can be expressed as:

$$v_j = p_j + t_j \tag{10}$$

2) RECOMMENDATION MODULE

After the analysis of attribute feature extraction, we describe the method of attribute feature acquisition integrated into the system's structure.

The input of the recommendation module in SI-MKR consists of two raw feature vectors u and v that describe user u and item v. In the original MKR model, there was no processing of the attribute, only the encoding process of the ID. For the SI-MKR model, user attributes are perfected and incorporated into multi-modal information.

We obtain the user's feature vector according to formula (7) and the feature vector of the item according to formula (10). The feature vector of the item takes the text attribute and the attribute of multi-value type as multi-modal information, yielding the item's feature representation. Then, we transmit the feature of the item to the entity set of the crossover unit and knowledge graph unit for cross-learning.

The input of the recommendation module in SI-MKR consists of two raw feature vectors u and v that describe user u and item v. Given user u' 's raw feature vector u, we use an L-layer MLP to extract this latent condensed feature:

$$u_L = M(M(\cdots M(u))) = M_L(u) \tag{11}$$

where $M(x) = \sigma(Wx + b)$ is a fully connected neural network layer with weight W, bias b, and nonlinear activation function $\sigma(\cdot)$. By combining formula 11 with formula 4, the following can be obtained:

$$u_L = M_L(concateate\,(\bar{x})) \tag{12}$$

For item v, we use a cross-compression unit to extract its feature:

$$vL = E_{e\sim S(v)}\left[C^L\,(v, e)\,[v]\right] \tag{13}$$

where S(v) is the set of associated entities of item v. For item v and one of its associated entities e, we first construct a d ×d cross-compression unit that has two parts: cross and compression. More details will follow.

After acquiring the latent feature of user u and project v, the final predicted probability of user u engaging item v can be obtained through the prediction function:

$$\hat{y}_{uv} = \sigma f_{RS}(M_L(concateate\,(\bar{x})), vL) \tag{14}$$

3) CROSS-COMPRESSION UNIT

The cross-compression unit is divided into two parts, namely, a crossing part and a compression part.

Cross-compression units are represented by a red rectangle in Figure 2, which will be described in this section.

A cross-compression unit is shown in Figure 5. This unit is the link module between item v and entity e. For the latent feature of latent feature $v_l \in R^d$ and the latent feature of latent feature $e_l \in R^d$, we construct C_l, representing the cross feature matrix of layer L.

$$C_l = v_l e_l^T = \begin{bmatrix} v_l^{(1)} e_l^{(1)} & \cdots & v_l^{(1)} e_l^{(d)} \\ \vdots & & \vdots \\ v_l^{(d)} e_l^{(1)} & \cdots & v_l^{(d)} e_l^{(d)} \end{bmatrix} \tag{15}$$

Formula 15 describes the crossing operation in a cross-compression unit. In cross-compression units, there are compression operations in addition to crossing operations. We output the feature vectors of items and entities for the next layer by projecting the cross feature matrix into the latent representation spaces.

$$\begin{aligned} v_{l+1} &= C_l w_l^{VV} + C_l^T w_l^{EV} + b_l^V \\ e_{l+1} &= C_l w_l^{EV} + C_l^T w_l^{EE} + b_l^E \end{aligned} \tag{16}$$

where $w_l \in R^d$ and $b_l \in R^d$ are trainable weight and bias vectors. The cross-compression unit can be denoted as:

$$[v_{l+1}, e_{l+1}] = C\,(v_l, e_l) \tag{17}$$

Through cross-compression units, SI-MKR can adaptively adjust the weights of knowledge transfer and learn the relevance between the two tasks.

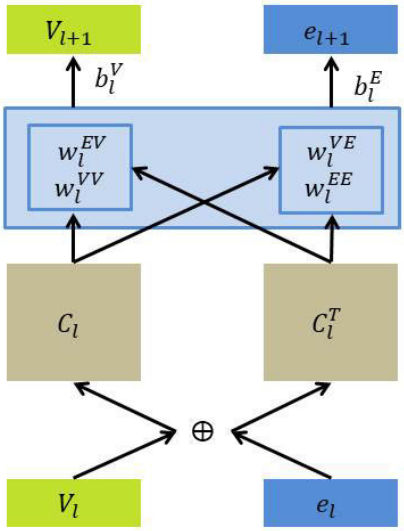

FIGURE 5. A framework for cross-compression units.

4) KGE MODULE

We now introduce the knowledge graph unit of the system. For the SI-MKR model, the structure of its knowledge graph embedding unit is the same as that of MKR and triples (h, r, t) are mainly used as the training input.

The features are extracted from the original head h and relation r by using the cross-compression unit and the multilayer perceptron, respectively. The corresponding vectors of the head and relation are spliced, and through the multilayer neural network, the estimated $\hat{t}$ of the corresponding vector of the tail is obtained.

$$hL = E_{e\sim S(h)}\left[C^L(v,h)[e]\right] \tag{18}$$

$$r_L = M^L(r) \tag{19}$$

$$\hat{t} = G(hL, rL) \tag{20}$$

where hL is the head feature representation corresponding to vL obtained through the cross-compression unit. rL is the initial representation of a relational feature, usually in the form of one-hot encoding, where S(h) stands for the association set of h in the knowledge graph, v stands for the item ID corresponding to h in the data of the recommendation system, and C^L is the cross-compression unit, which will be introduced later. $M(x)= \sigma(Wx + b)$ is a fully connected neural network layer with weight W, bias b, and nonlinear activation function $\sigma(\cdot)$, and $\hat{t}$ is the predicted vector of tail t. Finally, the score of the triple (h, r, t) is calculated using a score (similarity) function f_{KG}.

$$score(h, r, t) = \boldsymbol{f_{KG}}(t, \hat{t}) \tag{21}$$

In this model, f_{KG} is defined in the form of the inner product using the same treatment as in the MKR model:

$$\boldsymbol{f_{KG}}(t, \hat{t}) = \boldsymbol{\sigma}(\mathbf{t}^\top \hat{\mathbf{t}}) \tag{22}$$

D. LEARNING ALGORITHM

After introducing the main architecture of the model, we now design the loss function of the model. The system can be divided into three units, namely, the RS unit, the KGE unit, and the cross-compression units between them. The loss function for the whole system is designed to consist of three parts: the loss function of the recommended unit, the loss function of the KGE unit, and the regularization term.

$$L_{RS} = \mathcal{G}(concateate\left(v_j, u_j\right), R_{v_j,u_j}) \tag{23}$$

$$L_{KGE} = \mathcal{J}\left(\sigma\left(t^\top, G\left(hL, rL\right)\right), \sigma\left(t'^\top, G\left(hL', rL\right)\right)\right) \tag{24}$$

$\sigma(\cdot)$ is the normalized inner product, $\mathcal{G}(\cdot)$ is the cross-entropy function, and $\mathcal{J}(\cdot)$ aims to increase the score of all true triples while simultaneously decreasing the score of all false triples. The final loss function can be expressed as

$$L = \sum\nolimits_{v_j\in V, u_j\in U} L_{RS} + \lambda_1 L_{KGE} + \lambda_2 \|W\|_2^2 \tag{25}$$

$\lambda 1$ and $\lambda 2$ are the balancing parameters.

III. EXPERIMENTS

In this section, we evaluate the performance of SI-MKR using the MovieLens dataset.

A. DATASETS

MovieLens-1M [32] is a widely used benchmark dataset in movie recommendations, which has approximately 1 million explicit ratings (ranging from 1 to 5) collected on the MovieLens website.

This dataset contains 6036 users, 2347 items, 753772 ratings, and 20195 knowledge graph triples, which are composed of items, attribute names, and attribute contents.

For the user, attributes such as ID, gender, job, and age are included. There are user ID, gender, age, occupation ID and other fields. The format in user data is listed as:

$$UserID :: Gender :: Age :: Occupation$$

Gender is denoted by ““M”” for males and ““F”” for females. Age is chosen from the following ranges, and we encode occupation in a similar manner, which is divided into 21 categories by type.

For movies, there are attributes such as MovieID, titles, and genres. The format of movies is listed as:

$$MovieID ::Title :: Genres$$

Titles are identical to titles provided by the IMDB (including year of release). Movie genres consist of 18 basic types, whereby a movie can contain more than one type.

In a large movie dataset, if different words occupy one bit of the feature vector, the feature vector corresponding to the movie name with thousands of dimensions will be generated. To solve such problems, based on the statistics of the movie dataset, we establish a data dictionary so that every movie ID, title, and genre of each individual item in the data are converted to an integer number. Thus, we obtain the

TABLE 1. SI-MKR model.

Algorithm SI-MKR

Input: knowledge graph **G**, user attribute sets **U**, item attribute sets **I**, recommendation triple **(U,I,R)**
Output: the embedding of item and user and Prediction function
1: Classify attributes of **U** and **I**, Initialize all data
2: **for** number of training iteration **do:**
3: **for** t steps **do**:
4: **for** number of attributes of **U do**:
5: Vectorization of user attributes on Eq. (5) and Eq. (12)
6: **If** have Text type attribute **do**:
7: Eqs. (1-4)
8: **If** have multi-value type attribute **do**:
9: Eq. (9)
10: **return u**
11: **end for**
12: **for** number of attributes of **I do**:
13: Vectorization of item attributes on Eq. (6) and Eq. (9)
14: **If** have Text type attribute **do**:
15: Eqs. (1-4)
16: **If** have multi-value type attribute **do**:
17: Eq. (9)
18: **return i**
19: **end for**
20: Sample mini batch of positive and negative interactions from recommendation data;
21: Sample for each item **v** in the mini batch on Eq. (13)
22: Update parameters by gradient descent on Eqs. (23-25)
23: **end for**
24: Sample mini batch of true and false triples from G;
25: Sample for each head h in the mini batch on Eq. (18)
26: Update parameters by gradient descent on Eqs. (23-25)
27: **end for**

TABLE 2. Age classification.

1	Under 18
18	18-24
25	25-34
35	35-44
45	45-49
50	50-55
56	56+

corresponding embedding matrix with a fixed length index by preliminarily querying the data dictionary. According to the characteristics of each type of data connection after conversion, we generate the embedding layer.

To standardize the data format, the size of the embedded matrix is set as (N, 32), where the corresponding feature vectors are stored. Since movie ID is an integer and each movie has a unique ID, the value of N is the total number of movies plus one (the extra bit serves as a placeholder), and the same for the user ID, the user gender, the user age, and the type of job.

For multivalued type attributes, such as the movie type that is different from the movie ID, the type of each sample is not completely different, and a movie often has multiple types. For example, in the dataset, movies are divided into 18 types, such as love, comedy, war, etc., which means that there are theoretically no more than 18 types of a movie in the dataset. Therefore, we set the length of the index vector corresponding to the movie type to 19 bits, and the numbers 1-19 are used to represent each type. When the feature of the sample type is generated in the embedding layer, since multiple features can be found in a movie, expressed as (n, 32), it is necessary to add these features into the embedding layer so that the corresponding format is (1, 32).

According to statistics, there are no more than 5215 words in the movie name in this dataset, so the format of the embedding matrix corresponding to the movie name is (5216, 32). However, the difference is that, first, although the total number of words involved in the movie titles is very large, the actual length of the movie name is up to 14 words by statistics. Therefore, during data conversion, we only need to convert the movie titles into a 15(14+1)-bit index vector with numbers. Second, we obtain the textual characteristics of movie titles by using the relevant methods of natural language processing, so the corresponding embedding layer is not generated for movie titles.

TABLE 3. Original movie data.

ID	Titles	Categories
1	Toy Story	Animation|Children| Comendy
2	Jumanji	Adventure|Children| Comendy
3	Grumpier Old Men	Comedy|Romance
4	Wating to Exhale	Comedy|Drama
5	Father of the Bride part 2	Comedy

After processing the data, this movie data can be transformed into the following table:

TABLE 4. Processing movie data.

ID	Titles	Categories
1	[3258,3347,0...	[16,11,5,12,12,12...
2	[3419,2746,0...	[7,11,6,12,12,12...
3	[2286,3451,1086...	[5,2, 12,12,12...
4	[221,4209,4786...	[5,3,12,12,12...
5	[3118,3723,4355...	[5,12,12,12,12...

The SI-MKR model uses the embedded matrix in the first layer of the recommendation unit and takes the above

sequence of numbers as the index of the embedded matrix. The dimension of the embedded matrix is (N, 32). Since the title of the movie will be further processed, the movieID and the movie genres are joined to form the first full connection layer, with dimension (1,64).

In addition, the KGE unit mainly contains information such as movie, attribute and attribute value. The unit takes movie and attribute value as entity and attribute name as relation. Because MovieLens-1M are explicit feedback data, we convert them to implicit feedback. Each item is marked as 1, indicating that the user has rated the item (the threshold of rating is 4 for MovieLens-1M,), and the sample looked at for each user is set to 0, which is the rating of the same size. An unwatched set is sampled for each user, marked as 0 and of the same size as the watched set. We use Microsoft Satori to construct the knowledge graph for MovieLens-1M. We first select a subset of triples from the whole KG whose relation name contains a ''movie'', and the confidence level is greater than 0.9. Given the sub-KG, we collect all valid movies' IDs by matching their names with the tail of triples (head, film.film.name, tail). For simplicity, items with no matched or multiple matched entities are excluded. We then match the IDs with all KG triples' head and tail and select all well-matched triples from the sub-KG.

B. BASELINE

To demonstrate our algorithms' reliability, we use other models that incorporated knowledge graph techniques as a baseline. These models and the SI-MKR model proposed in this article jointly use the same dataset to conduct experimental verification in the sense of AUC value and ACC value of the model.

1) MKR

This model is the basis of the SI-MKR model proposed in this article. We set the number of high-level layers $K = 1$, f_{RS} as the inner product, and $\lambda_2 = 10^{-6}$,*L=1, d=8,* t=3 and $\lambda_1 = 0.5$. In this model, all attributes are added to the KGE unit, training is conducted in the knowledge graph unit, and only project-user-rating is used as the RS unit's training input.

2) PER

PER treats the KG as a heterogeneous information network and extracts meta-path-based features to represent the connectivity between users and items. In this article, we use manually designed user-item-attribute-item paths as features.

3) DKN

DKN uses entity embedding and word embedding as multiple channels and combines them together in CNN for CTR prediction. In this article, we use movie names as textual input for a DKN. The dimension of word embedding and entity embedding is 64, and the number of filters is 128 for window sizes 1, 2, and 3.

4) WIDE & DEEP [33]

It is a deep recommendation model combining a (wide) linear channel with a (deep) nonlinear channel. We concatenate the raw features of users and items, as well as the corresponding, averaged entity embeddings learned from TransR as input. The dimensions of the user, item, and entity are 64, and we use a two-layer deep channel with dimensions of 100 and 50 as well as a wide channel.

C. EXPERIMENT SETUP

In SI-MKR, we set the ratio of training, validation, and test set as 6:2:2. Each experiment is repeated 3 times, and the average performance is calculated. The number of epochs is set as *n_epochs=20*,$\lambda_2 = 10^{-8}$. The learning rate of the RS task is $lr\_rs=2e-4$, and the learning rate of the KGE task is $lr\_kge=2e-5$. For text CNN training, *filter_num=2 dropout=0.5.* We evaluate our method in two experimental scenarios: (1) In click-through rate (CTR) prediction, we apply the trained model to each piece of interaction in the test set and output the predicted click probability. We use AUC and accuracy to evaluate the performance of CTR prediction. (2) In the top-K recommendation, we use the trained model to select K items with the highest predicted click probability for each user in the test set and choose Precision@K and Recall@K to evaluate the recommended sets.

D. RESULTS

Figure 5.6 and Table 5 show the experimental comparison results of the SI-MKR model and other baseline models on MovieLens data set. Evaluation indexes such as AUC, ACC, Precision@K, and Recall@K were demonstrated. Table 5 shows the results of AUC and accuracy in CTR prediction.

TABLE 5. Results of AUC and accuracy in CTR prediction.

Model	MovieLens	
	AUC	ACC
PER	0.710	0.664
DKN	0.655	0.589
Wide&Deep	0.898	0.820
MKR	0.917	0.843
SI-MKR	0.921	0.845

- **PER** PER performs poorly on movie recommendations because user-defined meta-paths can hardly be optimal in reality.
- **DKN** Because the text length in the data set is relatively short, the results of the DKN model are also unsatisfactory in the data sets.
- **Wide&Deep** The performance of the Wide&Deep model is not as good as that of the MKR model because this model only splices attributes and does not integrate semantic analysis into training as side information like the MKR models.
- **MKR** For the MKR model, the results are excellent. However, compared with the SI-MKR model, there are still some deficiencies. Because the attributes are not

TABLE 6. Results of AUC on MovieLens-1M in CTR prediction with different ratios of training set r.

MODEL	20%	40%	60%	80%	100%
PER	0.607	0.638	0.663	0.688	0.710
DKN	0.582	0.601	0.620	0.638	0.655
Wide&Deep	0.802	0.815	0.840	0.876	0.898
MKR	0.874	0.882	0.897	0.908	0.917
SI-MKR	0.876	0.884	0.898	0.900	0.921

processed, the inherent information of the text attribute, user attribute and multi-value attribute is lost.

Therefore, it is effective to classify item attributes and then extract and represent attribute features with a variety of models. The performance can be improved by diversifying user attributes and importing user attributes into the MLP model to represent user vectors. In general, our SI-MKR performs best among all methods on the dataset.

SI-MKR also achieves outstanding performance in top-K recommendation, as shown in Figures 6 and 7.

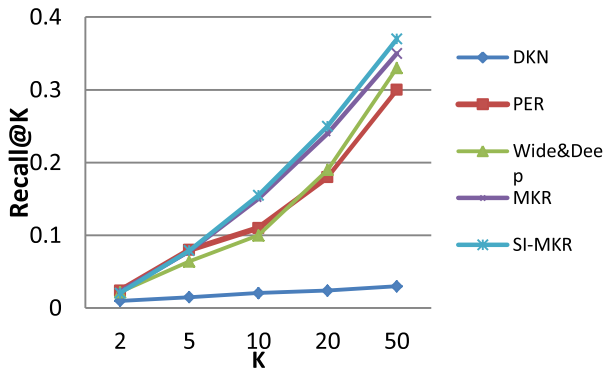

FIGURE 6. The results of Recall@K in top-K recommendation.

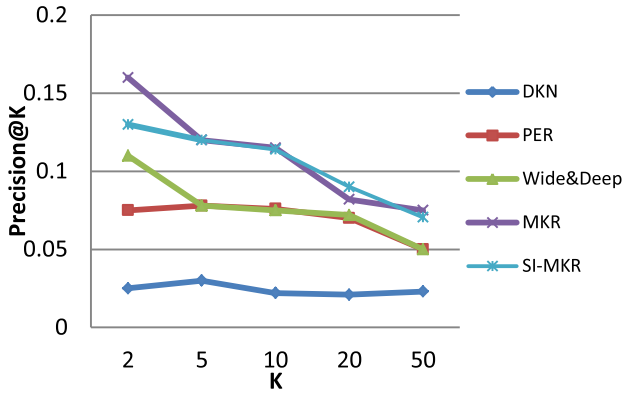

FIGURE 7. The results of Precision@K in top-K recommendation.

One major goal of using a knowledge graph in MKR is to alleviate the sparsity and the cold start problem of recommender systems. However based on MKR, the SI-MKR model proposed in this article can further alleviate the sparsity and the cold start problem of recommender systems. To investigate the efficacy of the KGE module in sparse scenarios, we vary the ratio of the training set of MovieLens-1M from 100% to 20% (while the validation and test set are kept fixed) and report the results of AUC in CTR prediction for all methods. The results are shown in Table 6.

IV. CONCLUSION

On the basis of MKR, this article uses knowledge graph and adds multi-modal information to enhance the performance of the recommendation system. We also use a deep neural network to process the original features of users and items. We propose the SI-MKR model, where the attributes of users and items are classified into text type attributes, multi-value type attributes and common type attributes. For the items, we input attributes of common types into the KGE units in the SI-MKR model. However, other attributes cannot acquire deep information through knowledge graph learning because of the limitation of knowledge graph triples. Therefore, for the attribute of text type, SI-MKR uses a CNN model to mine the internal information of text, such as the information of movie title. On the other hand, for attributes of multivalued types, the types need to be enumerated and then represented by adding the final vectors with the initial vector values. In all, attribute vectors that cannot be processed by the knowledge graph unit are integrated through a MLP and conducted by the RS unit. For user attributes, training is carried out through the MLP model, and then the trained vector is also processed by the RS unit. Afterwards, these two modules learn from each other in the cross-compression unit. Finally, we conduct experiments on the MovieLens dataset, and the results show that the SI-MKR model has significant advantages over other baseline models.

For future work, we plan to (1) add the user history behaviour as a related attribute and (2) design a model to better explore potential user interests and optimize cross-training units.

REFERENCES

[1] P. Resnick and H. R. Varian, "Recommender systems," *Commun. ACM*, vol. 40, no. 3, pp. 56–58, 1997.

[2] S. Ji, S. Pan, E. Cambria, P. Marttinen, and P. S. Yu, "A survey on knowledge graphs: Representation, acquisition and applications," 2020, *arXiv:2002.00388*. [Online]. Available: http://arxiv.org/abs/2002.00388

[3] J. Gao, X. Xin, J. Liu, R. Wang, J. Lu, B. Li, X. Fan, and P. Guo, "Fine-grained deep knowledge-aware network for news recommendation with self-attention," in *Proc. IEEE/WIC/ACM Int. Conf. Web Intell. (WI)*, Santiago, Chile, Dec. 2018, pp. 81–88.

[4] Q. Liu, F. Yu, S. Wu, and L. Wang, "A convolutional click prediction model," in *Proc. 24th ACM Int. Conf. Inf. Knowl. Manage. (CIKM)*, Melbourne, VIC, Australia, 2015, pp. 1743–1746.

[5] M. Jamali and M. Ester, "TrustWalker: A random walk model for combining trust-based and item-based recommendation," in *Proc. 15th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, 2009, pp. 397–406.

[6] B. Perozzi, R. Al-Rfou, and S. Skiena, "DeepWalk: Online learning of social representations," in *Proc. 20th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining (KDD)*, New York, NY, USA, 2014, pp. 701–710.

[7] A. Grover and J. Leskovec, "Node2vec: Scalable feature learning for networks," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, San Francisco, CA, USA, Aug. 2016, pp. 855–864.

[8] J. Tang, M. Qu, M. Wang, M. Zhang, J. Yan, and Q. Mei, "LINE: Large-scale information network embedding," in *Proc. 24th Int. Conf. World Wide Web (WWW)*, Florence Italy, 2015, pp. 1067–1077.

[9] P. Liu, X. Qiu, and X. Huang, "Learning context-sensitive word embeddings with neural tensor skip-gram model," in *Proc. IJCAI Int. Joint Conf. Artif. Intell.*, Buenos Aires, Argentina, 2015, pp. 1284–1290.

[10] F. Zhang, N. J. Yuan, D. Lian, X. Xie, and W.-Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Aug. 2016, pp. 353–362.

[11] S. Rendle, "Factorization machines," in *Proc. IEEE Int. Conf. Data Mining*, Sydney, NSW, Australia, Dec. 2010, pp. 995–1000.

[12] Y. Juan, Y. Zhuang, W.-S. Chin, and C.-J. Lin, "Field-aware factorization machines for CTR prediction," in *Proc. 10th ACM Conf. Recommender Syst.*, New York, NY, USA, Sep. 2016, pp. 43–50.

[13] Y. Qu, H. Cai, K. Ren, W. Zhang, Y. Yu, Y. Wen, and J. Wang, "Product-based neural networks for user response prediction," in *Proc. IEEE 16th Int. Conf. Data Mining (ICDM)*, Barcelona, Spain, Dec. 2016, pp. 1149–1154.

[14] D. Harwath, A. Torralba, and J. R. Glass, "Unsupervised learning of spoken language with visual context," in *Proc. Adv. Neural Inf. Process. Syst.*, Barcelona, Spain, 2016, pp. 1866–1874.

[15] X. Chen, Y. Duan, R. Houthooft, J. Schulman, I. Sutskever, and P. Abbeel, "InfoGAN: Interpretable representation learning by information maximizing generative adversarial nets," in *Proc. 30th Int. Conf. Neural Inf. Process. Syst. (NIPS)*. Red Hook, NY, USA: Curran, 2016, pp. 2180–2188.

[16] H. Edwards and A. Storkey, "Towards a neural statistician," 2016, *arXiv:1606.02185*. [Online]. Available: http://arxiv.org/abs/1606.02185

[17] W.-N. Hsu, Y. Zhang, and J. Glass, "Unsupervised learning of disentangled and interpretable representations from sequential data," in *Proc. Adv. Neural Inf. Process. Syst.*, 2017, pp. 1876–1887.

[18] X. He, J. Pan, O. Jin, T. Xu, B. Liu, T. Xu, Y. Shi, A. Atallah, R. Herbrich, S. Bowers, and J. Q. Candela, "Practical lessons from predicting clicks on ads at Facebook," in *Proc. 8th Int. Workshop Data Mining Online Advertising*, New York, NY, USA, 2014, pp. 1–9.

[19] K. Gai, X. Zhu, H. Li, K. Liu, and Z. Wang, "Learning piece-wise linear models from large scale data for ad click prediction," 2017, *arXiv:1704.05194*. [Online]. Available: http://arxiv.org/abs/1704.05194

[20] G. Zhou, X. Zhu, C. Song, Y. Fan, H. Zhu, X. Ma, Y. Yan, J. Jin, H. Li, and K. Gai, "Deep interest network for click-through rate prediction," in *Proc. 24th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, London, U.K., Jul. 2018, pp. 1059–1068.

[21] J. M. Gomez-Perez, J. Z. Pan, G. Vetere, and H. Wu, "Enterprise knowledge graph: An introduction," in *Exploiting Linked Data and Knowledge Graphs in Large Organisations*. Cham, Switzerland: Springer, 2017, pp. 1–14.

[22] H. Wang, F. Zhang, X. Xie, and M. Guo, "DKN: Deep knowledge-aware network for news recommendation," in *Proc. World Wide Web Conf. World Wide Web (WWW)*. Geneva, Switzerland: International World Wide Web Conferences Steering Committee, 2018, pp. 1835–1844.

[23] H. Wang, F. Zhang, M. Zhao, W. Li, X. Xie, and M. Guo, "Multi-task feature learning for knowledge graph enhanced recommendation," in *Proc. World Wide Web Conf. (WWW)*, San Francisco, CA, USA, 2019, pp. 2000–2010.

[24] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li, X. Xie, and M. Guo, "RippleNet: Propagating user preferences on the knowledge graph for recommender systems," in *Proc. 27th ACM Int. Conf. Inf. Knowl. Manage.*, New York, NY, USA, Oct. 2018, pp. 417–426.

[25] H. Wang, M. Zhao, X. Xie, W. Li, and M. Guo, "Knowledge graph convolutional networks for recommender systems," in *Proc. World Wide Web Conf. (WWW)*, San Francisco, CA, USA, 2019, pp. 3307–3313.

[26] H. Wang, F. Zhang, M. Zhang, J. Leskovec, M. Zhao, W. Li, and Z. Wang, "Knowledge-aware graph neural networks with label smoothness regularization for recommender systems," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Jul. 2019, pp. 968–977.

[27] X. Wang, X. He, Y. Cao, M. Liu, and T.-S. Chua, "KGAT: Knowledge graph attention network for recommendation," in *Proc. 25th ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*. New York, NY, USA: ACM, Jul. 2019, pp. 950–958.

[28] W.-T. Yih, X. He, and C. Meek, "Semantic parsing for single-relation question answering," in *Proc. 52nd Annu. Meeting Assoc. Comput.*, Baltimore, MD, USA, vol. 2, 2014, pp. 643–648.

[29] N. Kalchbrenner, E. Grefenstette, and P. Blunsom, "A convolutional neural network for modelling sentences," in *Proc. 52nd Annu. Meeting Assoc. Comput. Linguistics*, Baltimore, MD, USA, vol. 1, 2014, pp. 655–665.

[30] R. Collobert, J. Weston, L. Bottou, M. Karlen, K. Kavukcuglu, and P. Kuksa, "Natural language processing (almost) from scratch," *J. Mach. Learn. Res.*, vol. 13, no. 1, pp. 2493–2537, 2011.

[31] Y. Kim, "Convolutional neural networks for sentence classification," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, Doha, Qatar, 2014, pp. 1746–1751.

[32] (1997). *Movielens Dataset*. [Online]. Available: https://grouplens.org/datasets/movielens/

[33] H. Cheng, L. Koc, J. Harmsen, T. Shaked, T. Chandra, H. Aradhye, G. Anderson, G. Corrado, W. Chai, M. Ispir, and R. Anil, "Wide & deep learning for recommender systems," in *Proc. Workshop Deep Learn. Recommender Syst.*, Boston, MA, USA, 2016, pp. 7–10.

YUEQUN WANG received the bachelor's degree from the School of Software, Jilin University, in 2014, where he is currently pursuing the Ph.D. degree with the School of Computer Science and Technology. His research interests include social networking, deep learning, and artificial intelligence. He wants to combine the field of natural language processing with the field of social networks to solve problems related to social networks.

LIYAN DONG received the Ph.D. degree from Jilin University in 2007. He is currently a Professor with the College of Computer Science and Technology, Jilin University. His research interests include database theory, data mining, distributed database and application, embedded database, and information retrieval.

HAO ZHANG is currently a Professor with the College of Computer Science and Technology, Jilin University, China. He is also a Visiting Scholar with University of Missouri, CO, USA, and an outstanding National Visiting Scholar with Zhejiang University, China. His research interests include machine learning algorithm research in bioinformatics, biometric identification and simulation, pattern recognition and image processing, biological big data analytics, intelligent transportation system design, and applications of artificial intelligence.

XINTAO MA received the degree (exchange program) from the East China University of Science and Technology, the bachelor's degree from the Luebeck University of Applied Science, and the master's degree from the Hamburg University of Technology, Germany. She is currently pursuing the Ph.D. degree with the Department of Computer Science and Technology, Jilin University, China. Her major research fields are social network, artificial intelligence, and deep learning.

YONGLI LI was born in Harbin, China, in 1965. She received the Ph.D. degree from Jilin University 2010. She is currently an Associate Professor with the School of Information Science and Technology, Northeast Normal University. Her research interests include information security and data processing.

MINGHUI SUN received the Ph.D. degree in computer science from the Kochi University of Technology, Japan, in 2011. He is currently an Assistant Professor with the College of Computer Science and Technology, Jilin University, China. He is interested in using HCI methods to solve challenging real-world computing problems in many areas, including tactile interfaces, pen-based interfaces, and tangible interfaces.