applied sciences

MDPI

Article

Enhancing Recommender Systems with Semantic User Profiling through Frequent Subgraph Mining on Knowledge Graphs

Haemin Jung [1], Heesung Park [1] and Kwangyon Lee [2,*]

[1] Department of Industrial Engineering, Yonsei University, Seodaemun-gu, Seoul 03722, Republic of Korea; hmjung@yonsei.ac.kr (H.J.); gregmark@naver.com (H.P.)
[2] School of Electronic Engineering, Soongsil University, Dongjak-gu, Seoul 06978, Republic of Korea
* Correspondence: kylee@ssu.ac.kr

Abstract: Recommender systems play a crucial role in personalizing online user experiences by creating user profiles based on user–item interactions and preferences. Knowledge graphs (KGs) are intricate data structures that encapsulate semantic information, expressing users and items in a meaningful way. Although recent deep learning-based recommendation algorithms that embed KGs have demonstrated impressive performance, the richness of semantics and explainability embedded in the KGs are often lost due to the opaque nature of vector representations in deep neural networks. To address this issue, we propose a novel user profiling method for recommender systems that can encapsulate user preferences while preserving the original semantics of the KGs, using frequent subgraph mining. Our approach involves creating user profile vectors from a set of frequent subgraphs that contain information about user preferences and the strength of those preferences, measured by frequency. Subsequently, we trained a deep neural network model to learn the relationship between users and items, thereby facilitating effective recommendations using the neural network's approximation ability. We evaluated our user profiling methodology on movie data and found that it demonstrated competitive performance, indicating that our approach can accurately represent user preferences while maintaining the semantics of the KGs. This work, therefore, presents a significant step towards creating more transparent and effective recommender systems that can be beneficial for a wide range of applications and readers interested in this field.

Keywords: frequent subgraph mining; knowledge graph; recommender system; user profiling

Citation: Jung, H.; Park, H.; Lee, K. Enhancing Recommender Systems with Semantic User Profiling through Frequent Subgraph Mining on Knowledge Graphs. *Appl. Sci.* **2023**, *13*, 10041. https://doi.org/10.3390/app131810041

Academic Editor: José Machado

Received: 30 July 2023
Revised: 30 August 2023
Accepted: 3 September 2023
Published: 6 September 2023

1. Introduction

Recommender systems have long attracted attention and have become increasingly essential in many aspects of our lives. An effective recommendation algorithm benefits both the user and service (or content) provider. If the recommendation system can reasonably match the product, service, or content with what a user wants, high user satisfaction and sales growth for the provider can be guaranteed.

A knowledge graph (KG) is a database with a graph structure that can express various relationships [1]. In many machine learning algorithms, KGs have been used as an effective tool to provide explainability to their results [2]. Owing to their reasoning capability and human-readable characteristic, recommendation is also one of the areas where the use of KGs has received significant attention [3]. However, devising a method to process KGs for real-world recommendation applications is challenging because of their complex graph structure. With the recent progress in deep learning technology, recommendation methodologies using KG-embedding techniques have been proposed [4,5]. Despite their high performance, the explainability of KGs is erased in such methodologies, making it difficult to provide meaningful rationale for recommendations. Graph-learning-based algorithms [6,7] have also been proposed, but they require relatively large computing resources for an iterative propagation process.

In this study, we propose a user profiling method that represents user preferences while maintaining the semantic information inside the KG using frequent subgraph mining (FSM). Instead of directly embedding the KG, we identify the user-preferred attributes from frequent subgraphs and then use them to create vectors representing users and items. This user profiling method is rather simple but powerful; by training user–item representation pairs in a neural network, we obtained decent recommendation performance while still being able to describe user preferences with the user profile.

First, we combined a KG representing metadata about items and user–item interaction data to create an expanded KG. Considering the range of information to be used for the recommendations, this KG was divided into graphs for each user. Subsequently, we extracted frequent subgraphs, which explicitly represented the preference of each user.

The set of subgraphs was used to create user profiles, which are vectors containing both the semantic information of KG and frequency from the user–item interaction data. In other words, the user profile represented both the common attributes of the items interacted with by each user and the intensity (i.e., frequency) of the preference of each user for these attributes. By determining the dimension of the user profiles as the number of attributes appearing in the frequent subgraphs of all users, we could explain the preference of each user with respect to the preference space of the entire user group.

After obtaining the user profiles, we sampled positive and negative pairs of user–item interactions. We then fed the user profile and item vector pairs into a neural network classification model and trained the model to learn their relationship to recommend a specific item to the target user. Owing to the powerful approximation ability of the neural network, our methodology demonstrated good recommendation performance while providing concrete reasons for the recommendation based on the information in the user profiles.

To verify the performance of the proposed profiling method, we conducted experiments using movie data. The movie data have been used in previous studies [8–10] owing to having many attributes in a single item; relatively more people are involved in making a film than writing a book or piece of music. As filtering out only the meaningful attributes is essential for recommendation in the movie domain, it is an appropriate domain to examine the effectiveness of our profiling method, wherein user-preferred attributes are determined through FSM.

The characteristics of this study are as follows:

- We propose a novel user profiling method that generates user profiles, which clearly represent user preference through the use of FSM on a KG.
- We present a recommendation method that uses the information in the frequent subgraphs extracted from a KG to train a neural network model.
- Our method can overcome the disadvantages of a few existing opaque recommendation algorithms as it can provide recommendations using user profiles while maintaining the original semantics.
- We demonstrate the effective performance of our method using movie domain data, while factoring the characteristics of the domain.

The current section is followed by the Materials and Methods section, which reviews other existing recommendation algorithms before our user profiling methodology is explained in detail. The experimental setup and test results are then discussed in the Results section. Finally, the study and future research directions are discussed in the last section.

2. Materials and Methods

This section outlines the types and characteristics of recommendation algorithms. Subsequently, we introduce the recommender systems based on KG and explain the position and significance of our methodology in this field of research. We then describe our methodology. Figure 1 shows the entire process, which comprises two main steps. The first step is user profiling using frequent subgraphs, wherein data on user–item interactions are combined with a KG, followed by the creation of an individual graph for each user. Subsequently, frequent subgraphs are extracted from each user graph and converted into

vectors representing user preferences through user profiling. In the second step, pairs of user profile and item vectors are inputted to the deep neural network model to learn the relationship between them, and the outputted value is used for recommendation.

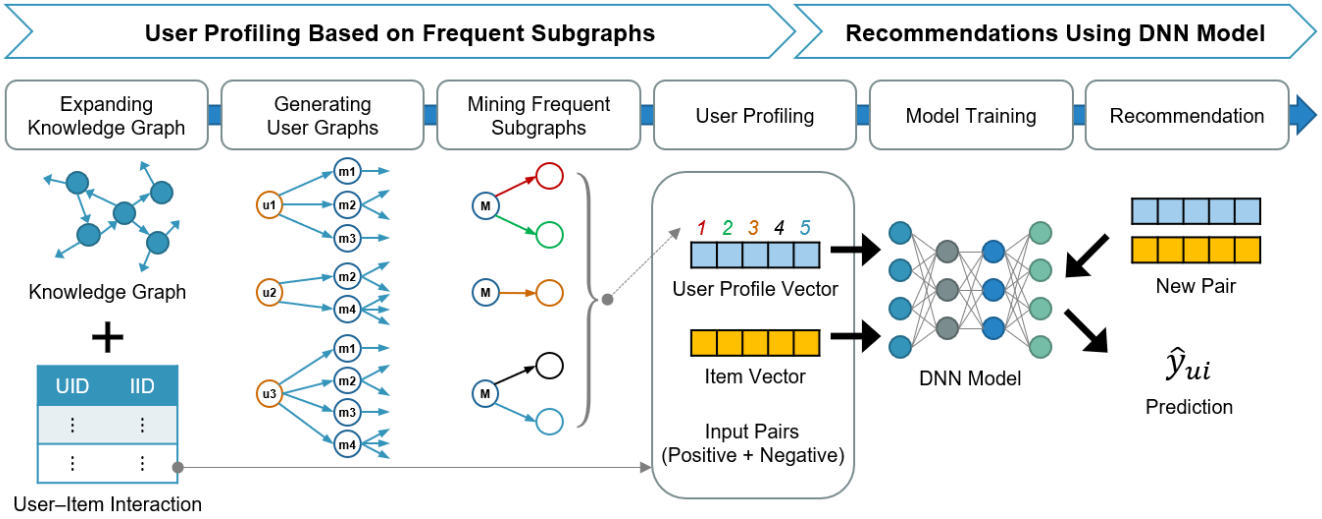

Figure 1. Overview of proposed method. Note: DNN, deep neural network.

2.1. Recommendation Algorithms

The purpose of a recommender system is to match users to items they may like based on their preference inferred from behavioral data (i.e., user–item interactions) [11]. Recommendation problems can be described using the following equation:

$$\hat{y}_{ui} = f(v_u, v_i). \tag{1}$$

The predicted preference of user u for item i, denoted as $\hat{y}_{ui}$, can be calculated using the function f of user representation v_u and item representation v_i [12]. Finding both the function f and a convenient way of representing users and items to make $\hat{y}_{ui}$ close to the actual preference y_{ui} is at the core of item recommendation.

Depending on the type of value y_{ui}, recommendation algorithms can be divided into two categories: implicit and explicit feedback-based [13]. Implicit feedback is based on whether a user has interacted with an item. Therefore, implicit feedback-based algorithms often attempt to solve binary classification problems and use accuracy and F1-scores as their performance metrics. By contrast, explicit feedback-based algorithms use more direct responses given by the user, such as user ratings. They view the recommendation task as a multi-class classification or regression problem to predict ratings and make recommendations if the predicted rating is high. For regression problems, metrics such as root mean square error are used. In this study, we proposed an algorithm based on implicit feedback; predicted preference y_{ui} is the probability that user u interacts with item i.

In addition, recommendation algorithms are divided into three categories based on their filtering types: collaborative, content-based, and hybrid filtering [13,14]. Collaborative filtering primarily considers the similarities between users or items in user–item interaction data. Since the underlying assumption is that users with similar preferences select similar items, this technique is domain independent. Various methods, such as clustering techniques [15] or matrix factorization [16], have been proposed. Content-based filtering uses domain-specific information to solve data sparsity and cold-start problems [10]. This additional domain-specific information varies from text or images about items to the demographic information of users, through which users and items are represented for recommendation. Finally, the hybrid method attempts to achieve higher performance by combining multiple approaches [17].

Our proposed method involves hybrid filtering; it benefits from the characteristics of content-based filtering in that it creates user profiles using a KG as external information but also resembles collaborative filtering in that information from multiple users is reflected together in user profiling.

KG-Based Recommendation Algorithms

In recent years, many researchers have introduced the KG as additional information to recommender systems [6]. The term "KG" has never been clearly defined with a single meaning; however, it can generally be understood from the perspective of Semantic Web technologies or Linked Open Data [1]. Based on this perspective, we used common elements and concepts such as TBox and ABox levels, schemas, classes, object properties, datatype properties, and instances, which are defined in W3C standards such as the Resource Description Framework (RDF), RDF Schema, and Web Ontology Language. Generally, a KG can be defined as follows:

$$KG = (V, E) = \{(v_1, e, v_2) | v_1, v_2 \in V, e \in E\} \quad (2)$$

where V is the set of vertices constituting the graph, E denotes the set of edges constituting the graph, and (v_1, e, v_2) refers to a unit that forms a graph called a triple.

One of the key benefits of KG-based recommendation is the explainability when recommending items [18]. Many attempts have been made to use the semantic information and expressiveness of KGs [3–5]. Existing KG-based recommender systems can be generally categorized into three classes: embedding-based, path-based, and unified (or graph neural network-based) methods [6].

Embedding-based methods use various types of information from the KG to model the representations of items or users. Representative methods [4,5] characterize items through graph embedding algorithms such as TransE [19] or TransR [20] and apply them for recommendation. Although these methodologies show a good performance, because of the characteristics of the embedding process, they suffer from the shortcoming of losing the semantic information in the original KG. In other words, as entities are converted into numerical vectors, the semantic information originally represented as paths in the KG is lost.

Path-based methods consider the pre-defined connecting patterns or meta-paths of users and items, leveraging the similarity of paths between entities to enhance the performance of the recommendation. Yu et al. [21] suggested HeteMF, which focused on the relations of items connected by meta-paths to calculate item–item similarity. Drawing from the collaborative filtering approach, Luo et al. [22] proposed the HeteCF algorithm using item–item similarity in addition to user–item and user–user similarities. Although the meta-paths can introduce explainability in recommendations, there is a downside; the effectiveness of these methods depends heavily on the quality and amount of manually created meta-paths, which requires expertise in the target domain. This reduces their applicability to other domains.

Graph neural network (GNN)-based methods attempt to obtain better representations by integrating semantic information with connecting patterns through embedding propagation [6]. Wang et al. [7] proposed a KG convolutional network, which iteratively updates the representation of each entity through an inward propagation process from its neighbors. Another method is KGAT [23], which initializes the representation of entities with TransR [20] and propagates information outwardly. Recent GNN-based algorithms are not necessarily used only for KG-based recommendations, but they show high performance with KGs owing to their graph modeling ability. However, they require a relatively large amount of computation compared to other methods. Our proposed method does not clearly fall into any of these three main categories.

Methodologies based on KGs that can use additional information about items with stronger explanatory power compared to methods solely based on user–item interactions and are relatively immune to cold start problems. Recent KG-based recommendation

methodologies, such as KGAT [23], have shown high performance on data in the entertainment domain (books, music) and business service domain (restaurants). Moreover, they exhibit strengths in domains where there is a high necessity to clearly explain the recommendations, such as in the education domain for recommending educational content and in the medical domain for clinical treatment recommendation scenarios.

However, KG-based methodologies tend to focus on utilizing semantic information and often overlook frequency information, which is a crucial element representing user preferences in user–item interactions. Our approach differs from existing methodologies by addressing this issue; we create user profiles by incorporating frequency information.

In this paper, we have devised a novel user profiling method that creates user profiles using frequent subgraphs extracted from the graph of each user. A frequent subgraph, consisting of common attributes of user-interacted items with their frequencies, represents user preference well without losing the semantic information of the original KG. Subsequently, we converted the subgraphs into user profile vectors and inputted them into a neural network along with item vectors with the same vector space. The neural network, acting as the function f in (1), effectively modeled the relationship between users and items to predict preference. Most state-of-the-art algorithms are based on GNNs, but our algorithm is relatively simple and much easier to understand; it does not need to consider the entire graph or require a complex neural net structure for embedding propagation. In other words, we paid attention to the fact that competitive recommendation performance could be obtained by using user preferences extracted through FSM on a KG.

2.2. User Profiling Based on Frequent Subgraphs

We wanted to aggregate information from KGs from the perspective of each user. Therefore, we created a small graph called a user graph that contained only the metadata of the items that a user interacted with. Then, FSM was applied to the user graph to find attributes commonly included in the items, i.e., each user preference.

This process could be viewed as a type of feature extraction that selected only what the user considered important when selecting an item among the numerous attributes that described it.

2.2.1. Expansion of the KG

Two types of inputted data were required to implement our methodology. First, we needed user–item interaction data, which are the most common and fundamental pieces of information used for traditional collaborative filtering tasks. They comprised sequential or nonsequential records of items that each user interacted with. The other inputted data were KG data, which contained information about all the items. In the TBox level or schema, the classes and properties related to the items were defined and organized, and in the ABox level, each item that was an instance of those classes was connected to another instance or value based on object or datatype properties.

We combined these two pieces of data to form a larger KG. Figure 2 illustrates this combination process and the resulting KG. The orange and purple circles represent classes and instances, respectively; the orange dotted lines connecting the classes and instances represent “instanceOf” relationships; the blue straight lines represent object properties; and the green straight lines represent datatype properties. First, we created a class named “USER” in the KG and added users appearing in the user–item interaction as an instance of the class. Then, we connected them with the items they had interacted with through an object property named “interactedWith”. The instances of the “USER” class could be dynamically created according to the set of users for whom we wanted to find patterns, while the original part of the KG was relatively static (i.e., new items and their related triples could be added, but existing items and their triples were less likely to be changed or removed). In this combination process, duplicate cases wherein a user interacted with an item multiple times were not considered.

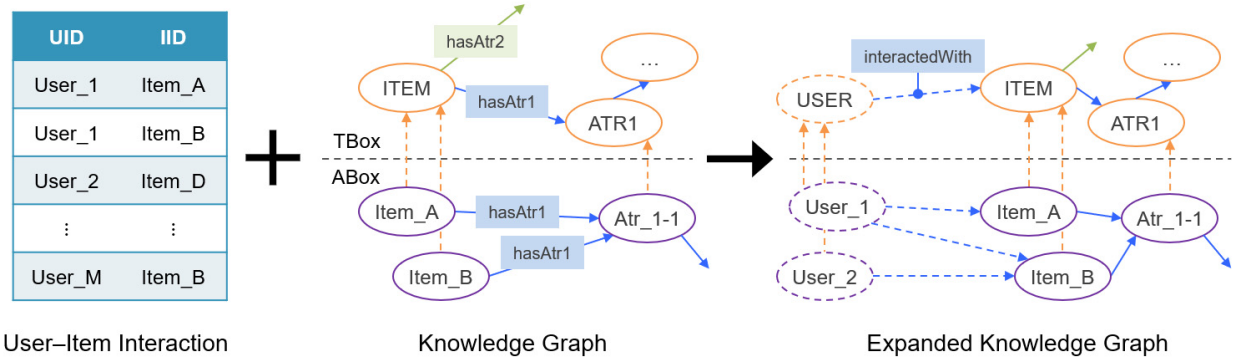

Figure 2. Generation of expanded knowledge graph.

2.2.2. Generation of User Graphs

To understand the preference of each user, we divided the expanded KG into multiple user graphs. A user graph is a part of a KG wherein only the items that each user has interacted with and their related triples are connected; each user has only one user graph. Figure 3 conceptually illustrates an example of a user graph; User_1 interacts with two items, and User_M interacts with three items. It should be noted that the user graph contained not only instance-level triple information but also information on the class to which each instance belonged.

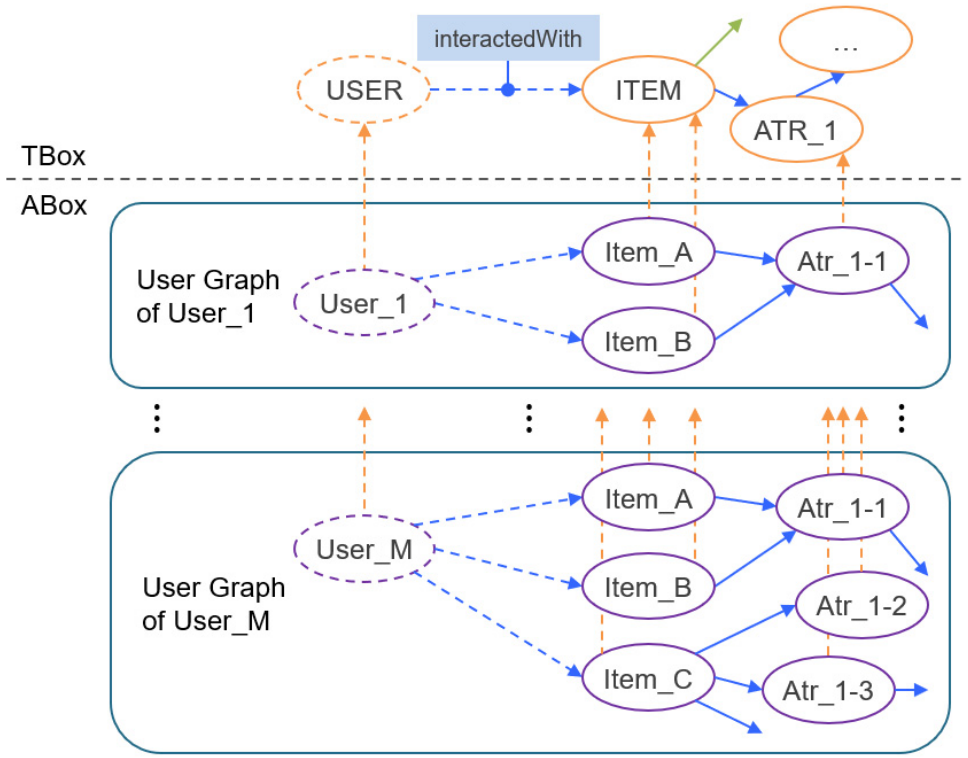

Figure 3. User graphs.

Due to the nature of the graph structure, instances of the expanded KG were connected to other instances at very remote ranges. However, not all the data included in the KG were required for recommendation. Therefore, we set path conditions to create user graphs that included only useful information.

There were two factors to be determined. One was the set of classes and properties to be excluded from the path search. This was information that could not be used or needed to be excluded for recommendation. For example, serial numbers, which are used as the

IDs for items, can be considered meaningless; when recommending a movie, the running time may be a relatively less important attribute than other attributes.

The other factor was the depth, which is related to the length of a path. The minimum depth removes short paths that may be included unnecessarily according to the structure of the KG. The maximum depth limits the path length that can in principle be infinite due to the nature of the graph. The depth is the number of properties connected from the "USER" class. For instance, the depth of class "ATR1" in Figure 3 is two, as it is connected to the "USER" class through the "ITEM" class.

These two factors should be carefully decided by domain experts who are aware of the KG structure, considering the attributes of the items they want to include as the reason for recommendation. By applying the factors differently according to the target users or purpose of recommendation, a single KG can be used in various ways.

Once these factors were set, a schema-level path that satisfied them needed to be found. The paths began from the "USER" class and consisted of continuous classes and properties. Subsequently, all the instance-level paths matching the schema-level path were searched and, finally, these paths were combined to form the user graph.

2.2.3. Mining Frequent Subgraphs

An FSM algorithm [24] was applied to each user graph, chunking the overlapping parts until there were no more candidate patterns with a frequency over the minimum support. An isomorphic part to count the frequency was required, as the item instances included in the user graph were essentially all different. Therefore, we applied abstraction to them. Abstraction is similar to conceptualization, wherein a specific instance is replaced with its class. Then, the algorithm finds common triples, counts their frequency, and selects the triples that exceed the minimum support as candidates. The most dominant candidate is replaced with a new node (or chunk), and the graph updates to reflect this replacement. This process is repeated until no more candidates appear, and a frequent subgraph is obtained by reconstructing the original shape of the last chunk.

Algorithm 1 is the pseudo code of the FSM algorithm.

Algorithm 1. FSM algorithm.

```
user graph U
candidates ← {}
chunks ← {}
min_support s
abstract(item instances in U)
count initial frequencies of triples in U
for each triple t in U
  if frequency(t) > s then
    candidates ← candidates + t
while |candidates| > 0 do
  target ← choose_from(candidates)
  chunks ← chunks + target
  candidates ← {}
  U ← update(U) # replace triples around target
  count triple frequencies in U
  for each triple t' in UG
    if frequency(t') > s then
      candidates ← candidates + t'
if |chunks| > 0 then
  subgraph ← restructure(chunks)
return subgraph
```

The main parameter to be set for this process was the minimum support for mining patterns. The minimum support allowed decisions to be made differentially by considering

the distribution of the number of items each user interacts with. For example, in the movie domain, if the minimum support was set to two for those who watched 10 movies and those who watched 100 movies, the degree of preference might be reflected differently. Therefore, it was appropriate to differentiate the minimum support.

The resulting set of subgraphs comprised common triples belonging to the items that a user commonly interacted with. Figure 4 shows a simple example of an extracted subgraph applied to a movie domain. As a movie is an item, it is abstracted. The numbers next to the properties indicate how many times there have been interactions with the movie with the corresponding triple.

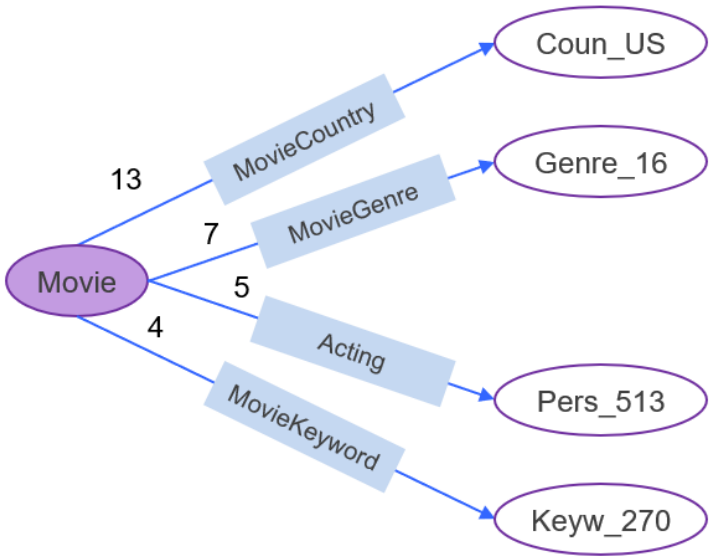

Figure 4. Example of frequent subgraph.

2.2.4. User Profiling

The frequent subgraph of a user expresses their preference. In other words, it represents the major attributes that the user considers when selecting an item and the strength of the preference the user has for that attribute. Each user has different frequent subgraphs; our proposed user profiling method makes these subgraphs into vectors of the same dimension.

Table 1 describes the notations of the main sets used in this section.

Table 1. Notation of Sets.

Notation	Definition
U	Users that exist in the user–item interaction data
G	Frequent subgraphs of U
T	Instance triples in KG
T_G	Instance triples in G
I	Items that exist in the user–item interaction data
I_u	Items that user u has interacted with
S	Samples in the training set

First, we collected a set of frequent subgraphs extracted for all the users of the user–item interaction. If the subgraph of each user u is denoted by g_u, and the triples included in g_u are denoted by $t(g_u)$, then the frequent subgraph set G of the users can be denoted as follows:

$$G = \{g_u | \exists g_u : |t(g_u)| > 0, u \in U\}. \quad (3)$$

That is, if the frequent subgraph obtained for each user includes one or more triples, these subgraphs are collected for set G. Additionally, the triple set T_G can be defined as follows:

$$T_G = \bigcup_{g_u \in G} t(g_u). \tag{4}$$

T_G can be viewed as a set of item attributes that all users consider important when selecting an item. Not all the elements of T_G are included in T due to abstraction, but they at least have a form like that of the triples included in T.

Finally, we define the user profile vector. The user profile vector v_u of user u can be expressed as follows:

$$v_u = \left(e_1, e_2, \cdots, e_i \cdots, e_{|T_G|}\right),$$

$$e_i = \begin{cases} 0 & (if t_i \notin t(g_u)) \\ frequency\ of\ t_i & (if t_i \in t(g_u)) \end{cases}. \tag{5}$$

If user u's subgraph contains the i-th triple t_i, which belongs to T_G, the value of element e_i is the frequency of the t_i that appears in user u's frequent subgraph. The user profile vector represents the subgraph extracted from the user graph in the dimensional space of $|T_G|$. In other words, it is the preference of a specific user u in the space of attributes that the user set U considers important in item selection.

2.3. Recommendations Using the Deep Neural Network Model

In this step, the deep neural network learns the relationship between the user profile vector and the item vector obtained in the previous step. After training, the model can predict whether a user with a certain preference will choose an item with a certain attribute.

2.3.1. Model Training

The structure of the proposed neural network is shown in Figure 5. A pair of vectors at the bottom represents a user profile vector and an item triple vector. The user profile vector in the example can be interpreted by itself; the frequent subgraph of the user contains two triples wherein PERS_0006908 ("Marc Macaulay") participates as an actor and three triples wherein GENR_00018 ("Drama") is the genre.

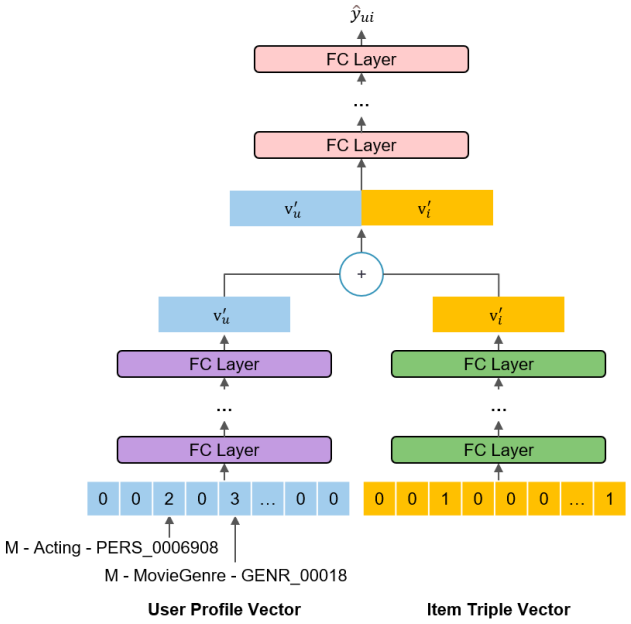

Figure 5. Structure of proposed prediction model.

There are various methods to embed items, with the simplest method setting the dimension of the item vector to that of the user profile vector. Here, the value of each element of the item vector is binary and is determined according to whether the corresponding item is connected to each triple in the KG.

The user profile and item vectors, which are sparse vectors, were transformed into dense representations v'_u and v'_i through each fully connected layer. Then, the two vectors were concatenated and passed through a layer that interpreted the relationship between them. Finally, a probability value $\hat{y}_{ui}$ that predicted whether the user would interact with the item was outputted using the sigmoid function. A brief formula for the model is shown as follows:

$$v'_u = f_m(W_u v_u + b_u), \tag{6}$$

$$v'_i = g_n(W_i v_i + b_i), \tag{7}$$

$$\hat{y}_{ui} = \sigma\left(h_o\left(W\left(v'_u \bigoplus v'_i\right) + b_{ui}\right)\right), \tag{8}$$

where the functions f_m, g_n, and h_o represent the fully connected layers of the user profile vectors, item vectors, and their concatenated embedding vectors, respectively, while m, n, and o represent the number of hidden layers.

Each input vector does not represent the ID of a specific user or item. As different users can have the same user profile vector, the proposed model calculates the probability that a user with a specific preference would watch a movie with a specific characteristic, rather than the probability of them watching a specific movie.

The input of the training data consisted of a pair of user profile and item vectors from the sample set S, and the label was either zero or one. For a set of positive samples S_{pos}, the label was one. The samples in S_{pos} were the samples included in the user–item interaction data. Label 0 was assigned to the samples in S_{neg}, which were obtained through negative sampling among the items that each user did not interact with.

$$S_{pos} = \left\{(v_u, v_i)\middle| v_u, v_i \in \mathbb{R}^k, i \in I_u\right\}, \tag{9}$$

$$S_{neg} = \left\{(v_u, v_i)\middle| v_u, v_i \in \mathbb{R}^k i \in (I - I_u)\right\}, \tag{10}$$

$$S = S_{pos} + S_{neg}. \tag{11}$$

As a binary classification task, it aimed to minimize the following loss:

$$\mathcal{L} = -\frac{1}{|S|}\sum_{(u,i)\in S}[(y_{ui} log S(\hat{y}_{ui}) + (1 - y_{ui}) log S(1 - \hat{y}_{ui})]. \tag{12}$$

2.3.2. Recommendation

After training, we can obtain a probability $\hat{y}_{ui}$ for a newly inputted user–item pair. Using this probability, we can select the most recommended item among several items that the user had never interacted with. Additionally, in real-world applications, a frequent subgraph can be used as a basis for recommendation. Each user's subgraph shows the attributes common to the items they have interacted with, acting as a visual representation that can attract their attention.

3. Results

In this section, we evaluate our methodology and assess the impact of the characteristics of data on its performance.

3.1. Experimental Design

3.1.1. Dataset

We used the MovieLens dataset, which is widely used for the performance evaluation of recommendation algorithms. The source of this dataset was GroupLens (https://grouplens.org/datasets/movielens/, accessed on 30 April 2023), and it provided user–item interaction data in various sizes. In our experiment, ml-Small and ml-1m were used to observe the changes in performance according to dataset size. Table 2 lists the number of users, items, and interactions in each dataset.

Table 2. Dataset Information.

	ml-Small	**ml-1m**
Users	630 (671)	6040 (6070)
Items	9048 (9066)	3666 (3706)
Interactions	99,898 (100,004)	997,407 (1,000,209)

The numbers in parentheses indicate the number of items in the original data. Users were excluded when frequent subgraphs were not extracted, and items were excluded when their metadata could not be found. Additionally, when either a user or an item was excluded, the corresponding interactions were also excluded. The data consisted of four columns: userId, movieId, rating, and timestamp. As this study targeted implicit feedback data, we did not use the rating column.

For a background KG, we used the data uploaded on Kaggle (https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset, accessed on 5 October 2022), which contained metadata on approximately 45,000 movies in The Movie Database (TMDB), along with a mapping between the TMDB movie ID and movieId column in the GroupLens data. After constructing a schema for this metadata, we built the KG by generating triples for each movie according to the schema. Finally, we created "USER" class instances and connected them with the item instances through the "interactedWith" property to prepare the KG for a state suitable for application in our algorithm.

3.1.2. Experimental Setup

We experimented with leave-one-out cross-validation. Specifically, after sorting using the timestamp column, only the movie that each user watched last was collected and saved as a test set. The hit ratio (HR) and normalized discounted cumulative gain (NDCG), which are frequently used terms in recommendation, were used as evaluation metrics. The HR was used to check whether an item was included in the top ranking, and the NDCG was affected by the position in the top ranking. To calculate these two metrics, a list of 100 movies that comprised the one left out at the end for testing and 99 randomly sampled movies unseen by the user were employed.

We first constructed a user graph for each user and extracted the frequent subgraph. The function of minimum support value was set as follows:

$$f(x) = \begin{cases} 2 & (3 \leq x < 8) \\ [lnx] & (x \geq 8) \end{cases} \tag{13}$$

where the number of movies the user has watched is x.

Most users only watched a few movies, but some watched hundreds or thousands. Thus, applying the same value to all the users would not be fair. However, if an excessively high threshold is applied to those who watch many movies, the criterion for judging the user preference exhibits an excessively large difference. Thus, considering both the absolute and relative values, we used $ln\ x$ which gradually increases; the value was approximated to an integer. Regarding the case wherein eight or less movies were watched, the value of $f(x)$ was one; hence, we set the value to two because a triple that appears once should not be considered frequent.

After the frequent subgraphs were selected, they were vectorized through user profiling. The movie pool for negative sampling was a set of all the items I that at least one user had interacted with, and the number of negative samples for each positive sample was four. Finally, 100 movies were ranked based on the probability value $\hat{y}_{ui}$ derived from the neural network.

The neural network model was implemented in TensorFlow, and the tested learning rates were 0.0005, 0.001, 0.002, and 0.005; the batch sizes were 256, 512, and 1024, and the dimensions of the embedding vector were 32, 64, 128, and 256.

The main baseline models were as follows: itemPop (item popularity) simply recommends items based on their popularity [25]; itemKNN [26] uses the similarities between items to identify the set of items to be recommended; Bayesian personalized ranking (BPR) employs the optimization criterion BPR-Opt and solves the recommendation problem from a Bayesian point of view [27]; the factored item similarity model (FISM) uses an item-item similarity matrix as the product of two latent factor matrices [28]; neural attentive item similarity (NAIS) uses a neural network model involving the application of the attention mechanism [29]; hamming spatial graph convolutional networks (HS-GCN) [30] capture hamming similarity between users and items through graph convolution operation; and recurrent KG embedding (RKGE) [31] obtains graph embeddings for recommendation using recurrent neural networks.

3.2. Results

The size of the user profile vector was 534 for ml-Small and 678 for ml-1m. Table 3 presents the number of properties included in the user profile and representative examples for the ml-1m data.

Table 3. Properties in User Profile Vector (ml-1m).

Property	Number of Properties	Examples
Production	69	George Lucas, Mike Fenton, Albert R. Broccoli
CostumeMakeup	17	John Mollo, Eugene Joseff, Joanna Johnston
MovieKeyword	115	Dystopia, independent film, android
MovieCompany	41	Paramount Pictures, Warner Bros., Twentieth Century Fox Film Corporation
Acting	159	Harrison Ford, Desmond Lleweyn, Michael Palin
BelongsToCollection	16	James Bond Collection, Planet of the Apes Original Collection, Puppet Master Collection
MovieRevenue	7	Revenue_Under_norm_100%, Revenue_Under_norm_40%, Revenue_Under_norm_50%
MovieVote Average	10	Vote_avg_Under_norm_90%, Vote_avg_Under_norm_100%, Vote_avg_Under_norm_80%
Art	20	Leslie Dilley, Herman F. Zimmerman, Norman Reynolds
Editing	19	Michael Khan, John Glen, Akira Kurosawa
MovieGenre	18	Drama, Comedy, Action
Camera	20	Takao Saito, Roland Totheroh, Robert Burks
Sound	36	John Williams, Scott Martin Gershin, Christopher Assells
VisualEffects	13	John Lounsbery, Milt Kahl, Ollie Johnston
Directing	26	Nick Park, Woody Allen,

Table 3. *Cont.*

Property	Number of Properties	Examples
Writing	39	George Lucas, Ian Fleming, Terry Gilliam
MovieBudget	8	Budget_Under_norm_30%, Budget_Under_norm_100%, Budget_Under_norm_40%
MoviePopularity	7	Popularity_Under_norm_100%, Popularity_Under_norm_90%, Popularity_Under_norm_80%
MovieCountry	11	United States, United Kingdom, France
MovieVote_Count	6	Vote_count_Under_norm_100%, Vote_count_Under_norm_50%, Vote_count_Under_norm_60%
Crew	12	Jessica Drake, Lisa Dennis Kennedy, Brian Johnson
MovieRuntime	9	Runtime_Under_norm_50%, Runtime_Under_norm_60%, Runtime_Under_norm_100%

3.2.1. Overall Performance

Table 4 shows the performance of our model and those of other methodologies. HR@k and NDCG@k were calculated for k = 5, 10, and 20. Our model had an embedding size of 128, a batch size of 1024, and a running rate of 0.001. The bold numbers in the table represent the best performances among the methodologies.

Table 4. Overall Performance.

	ml-Small						**ml-1m**					
	HR			**NDCG**			**HR**			**NDCG**		
	@5	**@10**	**@20**	**@5**	**@10**	**@20**	**@5**	**@10**	**@20**	**@5**	**@10**	**@20**
itemPOP	0.188	0.292	0.408	0.122	0.158	0.186	0.311	0.455	0.638	0.208	0.255	0.301
itemKNN	0.304	0.405	0.518	0.230	0.263	0.290	0.368	0.521	0.707	0.255	0.306	0.351
BPR	0.341	0.450	0.563	0.229	0.262	0.284	**0.510**	**0.687**	**0.843**	0.349	**0.405**	**0.443**
FISM	0.327	0.415	0.527	0.223	0.252	0.276	0.482	0.658	0.815	0.331	0.388	0.425
NAIS	0.336	0.424	0.531	0.234	0.250	0.276	0.497	0.671	0.826	**0.397**	0.397	0.435
HS-GCN	**0.380**	**0.477**	**0.675**	**0.257**	**0.280**	**0.347**	0.452	0.616	0.774	0.323	0.380	0.418
RKGE	0.365	0.461	0.630	0.248	0.275	0.345	0.435	0.623	0.784	0.318	0.377	0.408
Ours	**0.335**	**0.473**	**0.618**	**0.226**	**0.270**	**0.307**	**0.438**	**0.628**	**0.796**	**0.288**	**0.350**	**0.392**

We inferred the following facts:

- Our model showed sufficient performance on the ml-Small data. In HR@5, it showed similar performance to BPR, FISM, and NAIS; however, it showed better performance in HR@10 and HR@20. The algorithms utilizing graph structure showed good performance.
- The performance of our model on the ml-1m dataset was also sufficient. It showed a relatively low performance in metrics at rank 5, but for those at ranks 10 and 20, the difference was low. The performance of the graph-based algorithms is relatively poor.
- The performance of the algorithms showed significantly different patterns on the two datasets. Traditional methodologies exhibited relatively better performance on the ml-1m dataset, whereas the performance of HS-GCN, RKGE, and our methodology, which utilize graph structural information, actually decreased. We can infer that this is due to the characteristics of the data.

3.2.2. Influence of Data Characteristics

For the two datasets, the relative performance of our algorithm was different: for ml-Small, it showed a higher performance than the baseline algorithms; however, it did not show the same for ml-1m. Therefore, we analyzed the characteristics of the two datasets.

Table 5 shows the distribution of the number of users who watched a common movie.

Table 5. User per Movie.

Metrics	ml-Small	ml-1m
Mean	11.03	272.07
Standard deviation (σ)	24.05	385.07
Min	1	1
25%	1	34
50%	3	127
75%	9	353
Max	341	3428

The number of users for ml-1m was 10 times higher than that for ml-Small, although the total number of movies was 3666, which was relatively small compared to the 9066 for ml-Small (Table 2). Additionally, the average number of users who watched the same movie from the 9066 movies was 11.03, while the average number of users who watched the same movie from the 3666 movies was 272.07. In other words, the movies that users watched were much more concentrated in ml-1m. These results showed that the characteristics of the two datasets were completely different.

We calculated the sparsity according to the following formula:

$$sparsity = 1 - \frac{|interactions|}{|all\ possible\ interactions|}. \tag{14}$$

The sparsity values of the ml-Small and ml-1m datasets were 98.4 and 95.5%, respectively. Accordingly, we made the following assumptions. Owing to the nature of our method, which only included the preference—and not the ID—of a user in the input vectors, a good performance could only be achieved when the preference of each user was firmly distinguished. We inferred that the relative performance was low for ml-1m because the watching histories of the users had been duplicated to the extent that the extracted user profiles did not show a significant difference.

To prove this, we tried to adjust the ml-1m dataset. Figure 6 shows the number of users who watched each movie in the ml-1m dataset. The most watched movie was watched by 3428 of the 6040 users. We assumed that such popular movies were so popular that they would result in more indistinguishable user profiles. We created a dataset named ml-1m-3sig-out by excluding the 78 most watched movies that were outside 3σ from the mean (movies watched by more than 1427 users); this operation removed 152,841 interactions from the dataset.

Table 6 shows the comparison of performances for the new dataset when the hyperparameters are same. Our method showed a higher performance when extremely popular movies were excluded. This implied that the user profiling in this study would be more effective in situations wherein the item selection of each user was very different. Additionally, for practical applications, it would be helpful to exclude popular items in the dataset to extract distinctive user preferences.

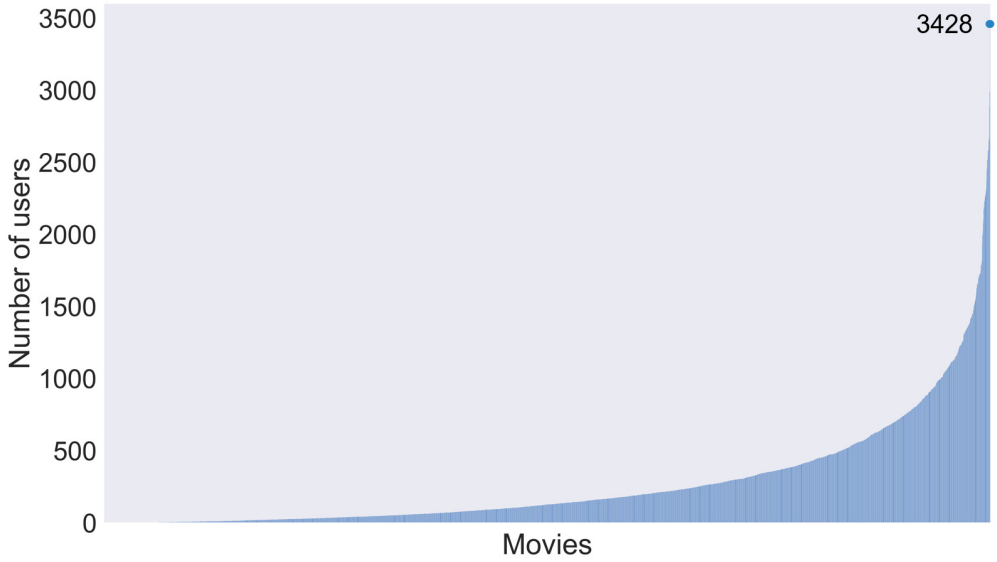

Figure 6. Number of users who have watched movies (sorted in ascending order).

Table 6. Performance Comparison Between Different Datasets.

Dataset	Users	Items	Interactions	Sparsity	HR			NDCG		
					@5	@10	@20	@5	@10	@20
ml-1m	6040	3666	997,407	95.5%	0.438	0.628	0.796	0.288	0.350	0.392
ml-1m-3sig	6040	3588	844,566	96.1%	0.440	0.635	0.805	0.291	0.357	0.399

4. Discussion

We conducted a comparative experiment between seven baseline algorithms and our methodology using two datasets from the movie domain, which vary in size and characteristics. As shown in Table 4, our model does not always outperform the others, but it provides greater explainability through the use of user profiles.

Table 3 presents a list of properties found in the frequent subgraphs, helping to understand the properties of movies that people are most interested in. These properties are the constituent elements of user profiles, and each user profile contributes to enhancing the explainability of each recommendation. Additionally, the fitting capability of the deep neural network model enables our method to demonstrate satisfactory performance.

Rather than adapting and testing various neural network structures, we concentrated on demonstrating that user profiling through frequent subgraphs provides explanatory recommendations with an adequate performance.

To better understand the cause of performance decline on the ml-1m dataset, we attempted to discern how the characteristics of the data affect our algorithm. Through experimentation, we concluded that sparsity plays a significant role; when there is a high overlap in the items selected by users, the differences between user profiles diminish, which, in turn, leads to lower performance. In other words, the algorithm performs well when users' choices or preferences are relatively diverse.

5. Conclusions

In this study, we propose a new user profiling recommendation method, employing FSM for KGs. Through this, we effectively represent user preferences in a matter that simultaneously reflects the semantic information contained in each subgraph and frequency information that indicates the strength of these preferences.

Regarding the application domain, movie recommendations are the most important concern for service providers, such as Netflix, Disney+, and Apple TV, helping them acquire

and retain subscribers. Therefore, our recommendation methodology focused on various characteristics of movie domain data.

In future research, we hope to address various issues, such as determining the optimal minimum support, improving the user profiling method to represent information more accurately in frequent subgraphs, and creating a neural network that yields better results. Additionally, we hope to explore how to use our user profiling in combination with the graph-learning-based algorithms; our user profiling can be used as initial graph embeddings. Furthermore, this research demonstrated the recommendation performance only using user-related data, while there is an abundance of other information that can be employed. For instance, we can include user and item IDs in the input vectors or use similarities between users and items.

Finally, the diversification of test datasets is required. We tested on the MovieLens dataset because it was a benchmark dataset that had been tested extensively in previous studies and the number of triples linked to each item was very large compared to those found in books and music datasets. In the future, we plan to test and expand the KG of MovieLens by using additional external metadata and experiment with KG datasets for other domains.

Author Contributions: Conceptualization, H.J., H.P. and K.L.; Methodology, H.J. and K.L.; Writing—original draft, H.J. and K.L.; Writing—review and editing, H.J., H.P. and K.L. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by Korea Institute for Advancement of Technology (KIAT): P0017123, The Competency Development Program for Industry Specialist.

Data Availability Statement: Publicly available datasets were analyzed in this study. This data can be found here: https://grouplens.org/datasets/movielens/ and https://www.kaggle.com/datasets/rounakbanik/the-movies-dataset (accessed on 30 April 2023).

Acknowledgments: Special thanks to researchers in Smart Systems Lab who shared their resources for the experiment.

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

References

1. Ehrlinger, L.; Wöß, W. Towards a definition of knowledge graphs. In Proceedings of the Posters and Demos Track of 12th International Conference on Semantic Systems, Leipzig, Germany, 12–15 September 2016.
2. Tiddi, I.; Schlobach, S. Knowledge graphs as tools for explainable machine learning: A survey. *Artif. Intell.* **2022**, *302*, 103627. [CrossRef]
3. Wang, X.; Wang, D.; Xu, C.; He, X.; Cao, Y.; Chua, T. Explainable reasoning over knowledge graphs for recommendation. In Proceedings of the AAAI Conference on Artificial Intelligence, Honolulu, HI, USA, 27 January–1 February 2019; Volume 33, pp. 5329–5336. [CrossRef]
4. Zhang, F.; Yuan, N.J.; Lian, D.; Xie, X.; Ma, W. 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. [CrossRef]
5. Huang, J.; Zhao, W.X.; Dou, H.; Wen, J.; Chang, E.Y. Improving sequential recommendation with knowledge-enhanced memory networks. In Proceedings of the SIGIR '18: 41st International ACM SIGIR Conference on Research & Development in Information Retrieval, Ann Arbor, MI, USA, 8–12 July 2018; pp. 505–514. [CrossRef]
6. Guo, Q.; Zhuang, F.; Qin, C.; Zhu, H.; Xie, X.; Xiong, H.; He, Q. A survey on knowledge graph-based recommender systems. *IEEE Trans. Knowl. Data Eng.* **2020**, *34*, 3549–3568. [CrossRef]
7. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge graph convolutional networks for recommender systems. In Proceedings of the WWW'19: The World Wide Web Conference, San Francisco CA USA, 13–17 May 2019; Association for Computing Machinery: New York, NY, USA, 2019; pp. 3307–3313. [CrossRef]
8. Choi, S.-M.; Ko, S.-K.; Han, Y.-S. A movie recommendation algorithm based on genre correlations. *Expert Syst. Appl.* **2012**, *39*, 8079–8085. [CrossRef]
9. Zamanzadeh Darban, Z.Z.; Valipour, M.H. GHRS: Graph-based hybrid recommendation system with application to movie recommendation. *Expert Syst. Appl.* **2022**, *200*, 116850. [CrossRef]
10. Chen, Y.-L.; Yeh, Y.-H.; Ma, M.-R. A movie recommendation method based on users' positive and negative profiles. *Inf. Process. Manag.* **2021**, *58*, 102531. [CrossRef]

11. Adomavicius, G.; Tuzhilin, A. Toward the next generation of recommender systems: A survey of the state-of-the-art and possible extensions. *IEEE Trans. Knowl. Data Eng.* **2005**, *17*, 734–749. [CrossRef]
12. Wu, S.; Sun, F.; Zhang, W.; Xie, X.; Cui, B. Graph Neural Networks in Recommender Systems: A Survey. *ACM Comput. Surv.* **2023**, *55*, 1–37. [CrossRef]
13. Isinkaye, F.O.; Folajimi, Y.O.; Ojokoh, B.A. Recommendation systems: Principles, methods and evaluation. *Egypt. Inform. J.* **2015**, *16*, 261–273. [CrossRef]
14. He, C.; Parra, D.; Verbert, K. Interactive recommender systems: A survey of the state of the art and future research challenges and opportunities. *Expert Syst. Appl.* **2016**, *56*, 9–27. [CrossRef]
15. Salah, A.; Rogovschi, N.; Nadif, M. A dynamic collaborative filtering system via a weighted clustering approach. *Neurocomputing* **2016**, *175*, 206–215. [CrossRef]
16. Koren, Y.; Bell, R.; Volinsky, C. Matrix factorization techniques for recommender systems. *Computer* **2009**, *42*, 30–37. [CrossRef]
17. Burke, R. Hybrid recommender systems: Survey and experiments. *User Model. User Adapt. Interact.* **2002**, *12*, 331–370. [CrossRef]
18. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Ripplenet: Propagating user preferences on the knowledge graph for recommender systems. In Proceedings of the 27th ACM International Conference on Information and Knowledge Management, Torino, Italy, 22–26 October 2018; pp. 417–426. [CrossRef]
19. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-Relational Data. Available online: https://proceedings.neurips.cc/paper/2013/file/1cecc7a77928ca8133fa24680a88d2f9-Paper.pdf (accessed on 30 April 2023).
20. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the AAAI Conference on Artificial Intelligence, Austin, TX, USA, 25–30 January 2015; Volume 29. [CrossRef]
21. Yu, X.; Ren, X.; Gu, Q.; Sun, Y.; Han, J. Collaborative Filtering with Entity Similarity Regularization in Heterogeneous Information Networks. 2013. Available online: https://www.google.com.hk/url?sa=t&rct=j&q=&esrc=s&source=web&cd=&ved=2ahUKEwi82I2j9ZSBAxXGjqQKHUGWDTwQFnoECBsQAQ&url=http%3A%2F%2Fink-ron.usc.edu%2Fxiangren%2Fijcai13_HINA.pdf&usg=AOvVaw1DW-Uz7L_ZxvDWQB6gSc2G&opi=89978449 (accessed on 30 April 2023).
22. Luo, C.; Pang, W.; Wang, Z.; Lin, C. Hete-CF: Social-based collaborative filtering recommendation using heterogeneous relations. In Proceedings of the IEEE International Conference on Data Mining, Shenzen, China, 14–17 December 2014; pp. 917–922. [CrossRef]
23. Wang, X.; He, X.; Cao, Y.; Liu, M.; Chua, T.-S. Kgat: Knowledge graph attention network for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Ser. KDD'19, New York, NY, USA, 4–8 August 2019; pp. 950–958. [CrossRef]
24. Lee, K.; Jung, H.; Hong, J.S.; Kim, W. Learning knowledge using frequent subgraph mining from ontology graph data. *Appl. Sci.* **2021**, *11*, 932. [CrossRef]
25. Steck, H. Item popularity and recommendation accuracy. In Proceedings of the 5th ACM Conference on Recommender Systems, Chicago, IL, USA, 23–27 October 2011; pp. 125–132. [CrossRef]
26. Deshpande, M.; Karypis, G. Item-based top- N recommendation algorithms. *ACM Trans. Inf. Syst.* **2004**, *22*, 143–177. [CrossRef]
27. Rendle, S.; Freudenthaler, C.; Gantner, Z.; Schmidt-Thieme, L. BPR: Bayesian personalized ranking from implicit feedback. In Proceedings of the 25th Conference on Uncertainty in Artificial Intelligence, Montreal, QC, Canada, 18–21 June 2009; pp. 452–461.
28. Kabbur, S.; Ning, X.; Karypis, G. Fism: Factored item similarity models for top-n recommender systems. In Proceedings of the 19th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Chicago, IL, USA, 11–14 August 2013; pp. 659–667. [CrossRef]
29. He, X.; He, Z.; Song, J.; Liu, Z.; Jiang, Y.; Chua, T. Nais: Neural attentive item similarity model for recommendation. *IEEE Trans. Knowl. Data Eng.* **2018**, *30*, 2354–2366. [CrossRef]
30. Liu, H.; Wei, Y.; Yin, J.; Nie, L. HS-GCN: Hamming spatial graph convolutional networks for recommendation. *IEEE Trans. Knowl. Data Eng.* **2022**, *35*, 5977–5990. [CrossRef]
31. Sun, Z.; Yang, J.; Zhang, J.; Bozzon, A.; Huang, L.K.; Xu, C. Recurrent knowledge graph embedding for effective recommendation. In Proceedings of the 12th ACM Conference on Recommender Systems, Vancouver, BC, Canada, 2–7 October 2018; pp. 297–305. [CrossRef]