applied sciences

MDPI

Article

Research on Hybrid Recommendation Model for Personalized Recommendation Scenarios

Wenkai Ni, Yanhui Du *, Xingbang Ma and Haibin Lv

College of Information and Cyber Security, People's Public Security University of China, Beijing 100038, China; 2021211449@stu.ppsuc.edu.cn (W.N.); 2021212361@stu.ppsuc.edu.cn (X.M.); 18259568358@163.com (H.L.)
* Correspondence: duyanhui@ppsuc.edu.cn

Abstract: One of the five types of Internet information service recommendation technologies is the personalized recommendation algorithm, and knowledge graphs are frequently used in these algorithms. RippleNet is a personalized recommendation model based on knowledge graphs, but it is susceptible to localization issues in user portrait updating. In this study, we propose NRH (Node2vec-side and RippleNet Hybrid Model), a hybrid recommendation model based on RippleNet that uses Node2vec-side for item portrait modeling and explores potential association relationships of items; the user portrait is split into two parts, namely, a static history portrait and a dynamic preference portrait; the NRH model adopts a hybrid recommendation approach based on collaborative filtering and a knowledge graph to obtain the user's preferences on three publicly accessible datasets; and comparison experiments with the mainstream model are lastly carried out. The AUC and ACC increased, respectively, by 0.9% to 29.5% and 1.6% to 31.4% in the MovieLens-1M dataset, by 1.5% to 17.1% and 4.4% to 18.7% in the Book-Crossing dataset, and by 0.8% to 27.9% and 2.9% to 24.1% in the Last.FM dataset. The RippleNet model was used for comparison experiments comparing suggestion diversity. According to the experimental findings, the NRH model performs better in accuracy and variety than the popular customized knowledge graph recommendation algorithms now in use.

Keywords: RippleNet; preference propagation; hybrid recommendation; user portrait

check for updates

Citation: Ni, W.; Du, Y.; Ma, X.; Lv, H. Research on Hybrid Recommendation Model for Personalized Recommendation Scenarios. *Appl. Sci.* **2023**, *13*, 7903. https://doi.org/10.3390/app13137903

Academic Editors: Konstantinos Pliakos and Alireza Gharahighehi

Received: 4 May 2023
Revised: 26 June 2023
Accepted: 4 July 2023
Published: 5 July 2023

Copyright: © 2023 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

The rapid development of the Internet has brought us a tremendous amount of information but also the problem of information overload [1]. Personal recommendation systems have emerged to alleviate the contradiction between information overload and users' personalized needs. The personalized recommendation aims to recommend information that matches users' interests according to their customized needs and behaviors and improves users' satisfaction and experience. At present, personalized recommendation algorithms can be divided into [2] association rule-based recommendation algorithms [3], content-based recommendation algorithms [4], user (or item)-based collaborative filtering recommendation algorithms [5], and hybrid recommendation algorithms, among which the most commonly used is the collaborative filtering-based recommendation algorithm, but it relies too much on the user's historical behavior. The sparsity of user–item interaction data will directly affect the accuracy of recommendation results. In recent years, the rapid development of knowledge graphs [6] has played an enormous role in integrating massive information and data mining. The recommendation algorithm based on a knowledge graph [7] can not only effectively incorporate the information of different structural levels but also explore the potential interest preferences of users to avoid a single recommendation result and promote the development of personalized recommendation algorithms. In addition, the knowledge graph has better interpretability, which can effectively improve the user's experience and increase the user's trust in the recommendation system.

In the conventional knowledge graph-based recommendation system, user portraits are modeled based on views and purchases made by users in the past. Utilizing the knowledge graph for preference propagation, the recommendation system generates suggestions based on the past behaviors of users and the entity relationships in the knowledge graph. Due to the potential limitations of knowledge representation in the knowledge graph, the recommendation system may be unable to exploit the global connections and characteristics among items. The interests and preferences of users are subject to change over time. When user interests change, if the recommendation system fails to update the user portrait promptly, localization of preference propagation will become a problem. The knowledge graph-based recommendation system tends to classify User A into the interest group of comedy movies based on their past behaviors and the association in the knowledge graph. Furthermore, the system is more likely to associate them with comedy movies. However, if User A's interest gradually shifts to suspense films, since the knowledge graph fails to discover the association between comedy movies and suspense movies promptly, resulting in the recommendation system failing to update the user portrait, the recommendation system continues to recommend comedy movies to User A despite User A's current interest. Therefore, the recommendation result is biased toward comedic films and needs more diversity and customization.

This paper introduces a hybrid recommendation model NRH (Node2vec-side and Ripplenet Hybrid Model) for personalized recommendation scenarios. The optimized model can overcome the problem of localization of preference propagation in the process of user image updating by the RippleNet [8] model and improve the "information cocoon". To build item portraits and fully utilize the association relationship between items, the NRH model uses Node2vec-side knowledge representation. This approach is suitable for large-scale personalized recommendation scenarios. The NRH model also creates a static historical image of users based on information about user–item history interaction before optimizing the RippleNet model for preference propagation. We adopt a hybrid prediction method based on collaborative filtering and a knowledge graph to make recommendations. We first mine the association relationships between knowledge graph nodes to find the items users may be interested in. Then, we combine the user portrait and item portrait to determine the users' preferences for strange things. Finally, we recommend the items that users like. In this study, we make experimental comparisons on the real datasets MovieLens-1M, Book-Crossing, and Last.FM. The findings demonstrate that the NRH model outperforms existing benchmark models regarding accuracy, variety, and recommendation power.

2. Background and Related Work

2.1. Node2vec-Side Fusion Knowledge Representation

Node2vec-side fusion knowledge representation is a knowledge representation proposed based on Node2vec [9], the basic idea of which is to reconstruct and homogenize the knowledge graph of the recommended domain kept in triples into a weighted directed graph and then use multivariate random wandering strategies to obtain the node representations. The relationships between the nodes are aggregated based on the generated wandering sequences. Finally, the knowledge representations of the target knowledge graph are obtained by combining the outcomes of each wandering strategy's representations.

Where assuming $f(x)$ is the knowledge representation obtained by node x under a particular preference strategy wandering, then for the set of relations $\boldsymbol{R} = \{\boldsymbol{r_1}, \boldsymbol{r_2}, \boldsymbol{r_3}, \ldots, \boldsymbol{r_n}\}$ in the knowledge graph; assuming any link $\boldsymbol{r_i} \in \boldsymbol{R}$, the vector representation form of relation $\boldsymbol{r_i}$ can be obtained by the following equation:

$$f(r_i) = \frac{\sum\limits_{(h,r_i,t)\in \mathrm{G}} (f(t) - f(h))}{min(\mathrm{count}(r_i), \mathrm{Z})}, \tag{1}$$

where $Z = \lceil \alpha \cdot \mathrm{sum}(G) \rceil$, and $\alpha \in (0,1)$, where G denotes the set of knowledge graph triples, i.e., $G = \{(h, r, t)\}$; sum (G) represents the total number of triples in the knowledge graph; S denotes the set of wandering sequences; i.e., $\mathcal{S} = \{S_v \mid v \in V\}$, where V is the set of nodes; N_{r_i} denotes the value of relation $\boldsymbol{r_i}$ weights; $f(t)$ and $f(h)$ represent, respectively, the knowledge representation of the head and tail nodes based on the Node2vec representation; and count $(\boldsymbol{r_i})$ represents the number of occurrences of the relation $\boldsymbol{r_i}$ in the knowledge graph. Based on the above representation, the score function of the Node2vec-side is determined as follows:

$$g_r(h,t) = \mathrm{sim}(h+r,t) = \cos(h+r,t) = \frac{(h+r)\cdot t}{\|h+r\| \times \|t\|}, \tag{2}$$

where *sim* denotes the similarity function, and this paper uses cosine similarity as the similarity calculation function Finally, the knowledge representation $f_i(h,r,t)$ under a specific preference random walk is formed: $f_i(h)$, $f_i(t)$, $f_i(r)$.

The knowledge representation can be divided into two parts to obtain a more comprehensive knowledge representation. After receiving the knowledge representation $f_1(h, r, t)$, $f_2(h, r, t)$ by depth-first wandering (DFS) and breadth-first wandering (BFS) strategies, the knowledge representation $f(h, r, t)$ under this knowledge graph is then obtained by fusion, and $\cup$ in Equation (6) denotes the splicing operation.

$$f(h,r,t) = f_1(h,r,t) \cup f_2(h,r,t), \tag{3}$$

The Node2vec-side fusion knowledge representation score function $G(h,r,t)$ is as follows:

$$G(h,r,t) = h(g_{1r}(h,r,t), g_{2r}(h,r,t)), \tag{4}$$

of which

$$g_{ir}(h,t) = \mathrm{sim}(h+r,t) = \cos(h+r,t) = \frac{(h+r)\cdot t}{\|h+r\| \times \|t\|}, \tag{5}$$

$h(x, y)$ denotes the fusion function, which in this paper is taken as the mean function $h(X_1, X_2) = \frac{X_1+X_2}{2}$.

2.2. *Knowledge Graph-Based Recommendation Algorithm*

The current research direction of knowledge graph-based recommendation algorithms is mainly to enhance the accuracy of recommendations by knowledge embedding [10–16], path [17,18] or higher-order information aggregation [19–24], etc. Yang [10] et al. used the Metapath2Vec [11] model in the knowledge graph to introduce entity nodes to improve the recommendation result accuracy; in 2016, Zhang FZ et al. proposed the CKE [13] model by fusing collaborative filtering and the knowledge graph feature learning (TransR [12] model). Huang [14] used TransE [15] as a knowledge embedding model to describe user–item preferences; in 2018, Wang [16] et al., based on KCNN-embedded text word, proposed the deep knowledge-aware network (DKN). Similar recommendation algorithms based on knowledge embedding require preprocessing operations on the knowledge graph, using knowledge embedding to input relevant characteristics of items into the neural network. The recommendation model eventually learns and understands the knowledge structure inherent in the knowledge graph. Although the knowledge embedding-based approach is flexible and facilitates the use of models for migration in multiple scenarios, the lack of utilization of higher-order information leads to a limited representation of entities.

The path-based knowledge graph recommendation algorithm started earlier. Yu et al. proposed a heterogeneous information network method in 2014 [17], which uses matrix decomposition to obtain implicit features of users and items of different meta-paths for recommendation; based on this, Zhao et al. proposed a meta-graph-based recommendation method in 2017 [18], and the main idea is to use meta-graphs to represent the higher-order recommendation semantics and fully exploit its structural features by constructing meta-graph to make up for the shortage of the meta-path. Both approaches represent the relationship between users and items by extracting meta-paths or meta-graphs. The path-based recommendation approach aims to establish the path connection between user-items, which enhances the interpretability of recommendation results. Still, the quality of the mined path instances seriously affects the final recommendation results.

Knowledge graph recommendation algorithms based on higher-order information aggregation (e.g., KGAT [19]) aim to combine semantic information with paths to enrich the representation of users and items by surrounding neighborhood nodes. Similar recommendation algorithms only characterize the knowledge graph. Still, the recommendation results need to be more comprehensive regarding interpretability and logic. Hence, Wang et al. first proposed the concept of "preference diffusion" and thus offered the RippleNet [8,20] model. The core idea of the RippleNet model is to propagate user preferences on the knowledge graph and then continuously portray user profiles through the semantic relationships between entity nodes and nodes in the propagation process. It has good recommendation performance and is widely used.

2.3. RippleNet

RippleNet [8,20] is an end-to-end recommendation model framework based on knowledge graphs. Ripple contains two layers: (1) similar to the formation of ripples by water waves in reality, in terms of user portrait construction, the user's historical preferences are used as the center to spread outward layer by layer to form the distribution of the user's potential preferences for items; (2) similar to the gradual weakening of ripple amplitude in reality, the user's preference degree for an entity decreases with the increase of the propagation hops.

Therefore, RippleNet is analogous to how ripples propagate on the water surface, propagating user preferences on the set of knowledge graph entities and discovering potential user interests along the paths in the knowledge graph to achieve autonomous iterative propagation. It is characterized by inputting user–item pairs, obtaining deep-level possible connections between nodes by mining the correlations in the knowledge graph, and superimposing multiple "ripples" obtained from the propagation of users' historical click nodes to form a preference distribution of the user's preference for candidate item nodes, and finally using this preference distribution to calculate the probability of user interaction (e.g., clicking, browsing). In short, the user–item interaction matrix Y and the knowledge graph G are used to obtain the predicted click scores of User u for item v to be selected.

Knowing the interaction matrix Y and the knowledge graph G, the kth associated entity of User u is defined as

$$\varepsilon_u^k = \left\{ t \mid (h,r,t) \in G, h \in \varepsilon_u^{k-1} \right\} k = 1, 2, \cdots, H, \tag{6}$$

where $\varepsilon_u^0 = \{v \mid y_{uv} = 1\}$ indicates that the user has clicked on the item v. Based on (6), the definition of the Ripple set is given:

$$S_u^k = \left\{ (h,r,t) \mid (h,r,t) \in G, h \in \varepsilon_u^{k-1} \right\} k = 1, 2, \cdots, H. \tag{7}$$

However, the traditional RippleNet recommendation model has the following shortcomings: (1) The knowledge embedding is simple. In the embedding part of the knowledge graph, a single embedding method is used. The nodes are updated each time, and the outer nodes of potential preferences are prone to low semantic logic, etc. (2) There are problems such as user cold start and information cocoon. The AKTUP [21] model follows the node aggregation mechanism of the RippleNet model to mine user preferences. Still, the method needs to include the path connection between users and target items and complexity and can handle large-scale personalized recommendations. Huanqing Cui [22] et al. proposed a knowledge water wave graph convolutional network (KRGCN) that uses an end-to-end approach to obtain higher-order semantic information of the knowledge graph, improves the item feature learning part of the RippleNet model, and uses the convolutional graph network to aggregate higher-order domain information of items. Still, the recommendation results are less interpretable and have the user cold start problem. Moreover, the above improvements for the RippleNet model mainly consider the accuracy aspects of recommendations. Still, with the increase in user demand, users also pay more and more attention to recommendation diversity and put forward higher requirements for mining the relationship between items in personalized recommendations.

3. NRH Recommendation Model

The overall framework of the NRH recommendation model (Node2vec-side and RippleNet Hybrid Model) is shown in Figure 1, which makes the following innovations compared with the traditional RippleNet model: (1) Improving user portrait construction. Node2vec-side knowledge representation is introduced by embedding the Ripple set, which combines the static history portrait and the dynamic preference portrait to form a new user portrait. (2) Adopting a hybrid prediction approach. The probability values are calculated separately for the two parts of the user profile, and, finally, the two parts are combined to predict the user's preference for an item.

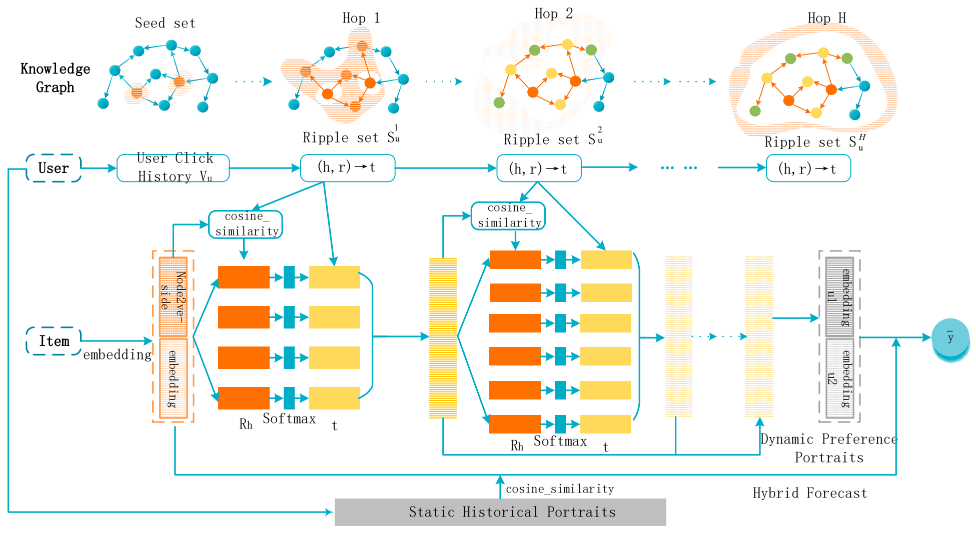


Figure 1. NRH recommendation model framework.

The following essential elements make up the basic framework of NRH:

Input: The knowledge network of suggested items and past user behavior data is input to the NRH model. The user's previous interactions, including purchases, browsing, ratings, etc., can be included in the historical behavior data. The items' properties, tags, and other features are included in the recommended item knowledge graph.

Node and relationship embedding generation: The NRH model uses the Node2vec-side technique to learn the embedding representation of the knowledge graph's nodes and relationships. In the Node2vec algorithm, random wandering on the knowledge graph produces wandering sequences. There are correlations and similarities between the objects in these node sequences. The embedded representations of nodes and edges are then learned after the nodes and edges are processed using relational aggregation, a variety of wandering algorithms, etc.

User interest representation: The NRH model builds the user interest representation using the RippleNet algorithm. For feature and representation learning of entities and association information in the knowledge graph, RippleNet employs attention methods. The association relationship between users and items is captured by creating a comprehensive view of user–item interactions. The static history portrait and dynamic preference portrait of users and the association between things are combined. In contrast, the interest representation of users is fused with the node-embedding representation.

Preference prediction part: The user's preference for a particular item is forecast by combining the user's static history portrait and item portrait. The prediction probability is calculated, and the score is adjusted to do this. By combining this data, the NRH model may produce recommendation results that are more precise and individualized.

3.1. Object Portrait Construction

The personalized recommendation system's term "item portrait" refers to the comprehensive description and feature extraction. It can help the recommendation system better grasp the similarity and correlation among items and realize more thorough and varied recommendations. On the other hand, it can help the recommendation system better hold the users' needs and improve the accuracy and effect of recommendations.

The association between items is typically sparse. Hence the item knowledge graph in the personalized recommendation method has less network density and less node aggregation than the classic knowledge graph. On the other hand, attributes function more autonomously as nodes. The knowledge graph frequently has a graph structure with items as the core and item attributes as the nodes to grow. Items are the basis of personalized recommendations, and item attributes are crucial nodes used to explain the qualities and properties of objects.

This part uses the Node2vec-side knowledge representation to generate item portraits, which increases their expressiveness while addressing the peculiarities of knowledge graphs in the customized recommendation domain. Items and attributes can be used as nodes in the item–attribute knowledge network employed in the personalized recommendation method. Edges can be used to describe the relationship between items and attributes. The precise procedure is as follows: First, the item–attribute knowledge graph is reconfigured into a weighted directed graph following the attributes, and two wandering strategies are used to separately capture the node information in the knowledge graph thoroughly following the knowledge graph in the personalized recommendation domain. It is possible to retrieve the item and attribute vector representation. The subgraphs with the same relationship are obtained, while the subgraphs with different relationships are segregated according to the random wandering sequence of nodes. These subgraphs' head and tail nodes aggregate the associations to produce new feature vectors. These are merged with the vector representations under the two wandering techniques to create a high-dimensional vector representation form. The core of this model is based on using the relational information in the knowledge graph, aggregating the relational subgraphs through the sequences produced by the wandering process, and combining entity nodes with relational information to create richer feature vectors. It also uses two different wandering strategies that consider global and local information, respectively, to further capture the node information in the knowledge graph.

3.2. User Profile Construction

The user profile is a user model formed by extracting user characteristics from different dimensions based on actual historical user behavior data, which has features of uniqueness, representativeness, and dynamism. In constructing user portraits based on knowledge graphs for recommender systems, the problems of "information cocoon" and the novelty of recommended items can be well solved by fusing the contextual information of entity nodes and the network space structure and comprehensively improve the performance of recommendation models. In this section, we propose a user profile representation method based on historical preferences, which takes user history as the core and constructs two dimensions from the user dynamic preference profile and user static history profile (as shown in Figure 2), where the dynamic user preference profile is mainly based on the RippleNet preference propagation process. The user static history profile is constructed by the item profile based on the user's past behavior data. The final portrait of User u is obtained by combining the two parts.

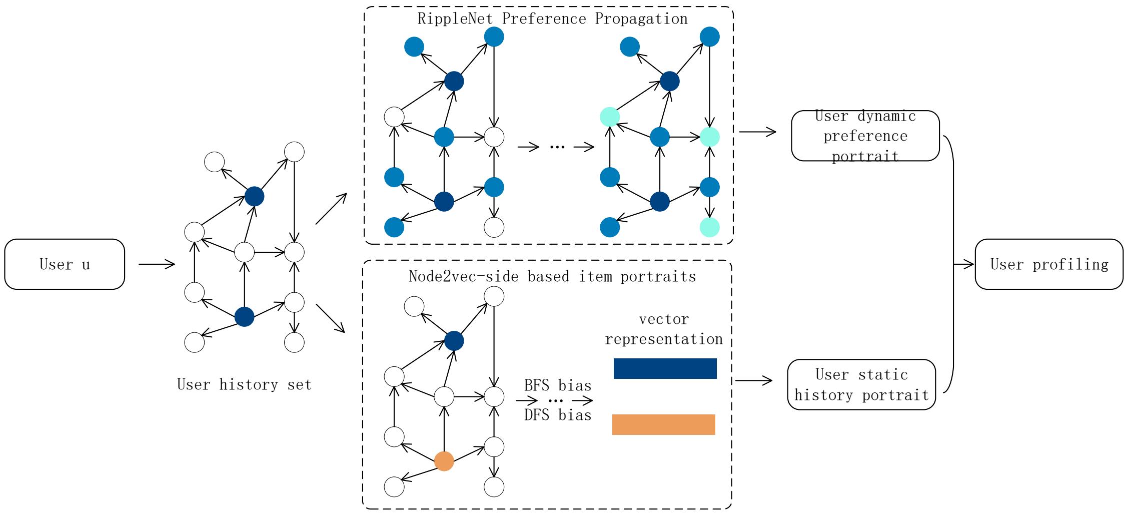

RippleNet Preference Propagation
User dynamic preference portrait
User u
User history set
Node2vec-side based item portraits
vector representation
BFS bias
DFS bias
User static history portrait
User profiling

Figure 2. User profile modeling based on historical preferences.

3.2.1. User Static History Portrait

To fully exploit users' historical behavioral data information, this part adopts the method of portraying users from the feature dimension of items themselves to form static historical portraits of users. We start the item preference set based on the historical preferences of User u and take the item portraits in this preference set as the static historical portraits of User u. These portraits are usually relatively stable. Specifically, we first obtain the item vector v' after item portrait modeling of the item–attribute knowledge graph in the recommender system using Node2vec-side and then obtain the historical preference items v'_u in the set $\varepsilon_u^0 = \{v \mid y_{uv} = 1\}$ of User u's ever-interaction objects according to Equation (6), and we use the vector sum of historical click items v'_u as the static history portrait of User u:

$$u = \Sigma v'_u. \tag{8}$$

Another study [23] shows that new users are more likely to choose items that are recommended as "popular"; therefore, for new users who do not have a history of clicks, we can balance the weights between items with a high number of clicks (popular items) and items with a low number of clicks (cold items) in the recommendation system (as shown in Equation (9)). An initial static portrait of the new user is generated. The popular and cold items in the system are used to make recommendations to predict new users' interest preferences as accurately as possible. This method can effectively solve the cold start problem of new users and improve the accuracy and user experience of the recommender system. Then,

$$u_{\text{new}} = \sum_{i=1}^{n} v'_{\text{hoti}} - \sum_{i=1}^{n} v'_{\text{coldi}} / \alpha, \tag{9}$$

where v'_{hoti} is the vector of items ranked i in descending order of click-through popularity, v'_{coldi} is the vector of items ranked i in descending order of click-through popularity, and α represents the weight assignment value.

3.2.2. User Dynamic Preference Portraits

The dynamic user preference portrait part is mainly portrayed from the perspective of neighborhood features of interactive items. The Ripple set (water wave set) S_u^k is constructed based on the click history vu of a specific User u. Vu is used as the starting point of preference propagation, propagated outward layer by layer on the item–attribute knowledge graph to form Ripple set $S_u^k (k = 1, 2, \cdots, H)$ for each layer of Hop, where

Ripple set S_u^k is the user click history set. The Ripple set is the set of triples after preference propagation by $\boldsymbol{v_u}$. Based on the original RippleNet item vector $\boldsymbol{v_1}$, the item–image v' based on Node2vec-side knowledge representation is introduced in the item–image representation part. The two are aggregated to obtain the item–vector representation $\boldsymbol{v}$ of the dynamic preference propagation part. The User u and item $\boldsymbol{v}$ information are fused to form the dynamic user preference image $\boldsymbol{o}$. The specific process is as follows:

The historical clicked items of User u are recorded as $\boldsymbol{v_u}$ in the item–attribute knowledge graph, and the preference-related entity set of historical items vu is thus generated, i.e., the preference-related entity set is created for User u in RippleNet by a circular recursive method, as shown in (10).

$$\varepsilon_u^k = \left\{ t \mid (h,r,t) \in G, h \in \varepsilon_u^{k-1} \right\},\ k = 1,2,\cdots,H, \tag{10}$$

where $\varepsilon_u^0 = V_u = \{v \mid y_{uv} = 1\}$ indicates the set of user history click items.

These entities can be considered preference-related entities of User u in the knowledge graph based on historical clicks on $\boldsymbol{v_u}$. After defining preference-related entities, all the K-hop Ripple sets of User u are defined as follows:

$$\mathcal{S}_u^k = \left\{ (h,r,t) \mid (h,r,t) \in \mathcal{G}\ and\ h \in \mathcal{E}_u^{k-1} \right\}, k = 1,2,\ldots,H, \tag{11}$$

In each Hop, an aggregated Embedding is used for item portrait representation. The original Ripple model computes the correlation probability p_{1i} for each triplet ($\boldsymbol{h_i}$, $\boldsymbol{r_i}$, $\boldsymbol{t_i}$) for item $\boldsymbol{v_1}$ with head $\boldsymbol{h_i}$ and relationship matrix $\boldsymbol{R_i}$ in the original Ripple model, respectively.

$$p_{1i} = softmax\left(v_1^T R_i H_i\right) = \frac{exp\left(v_1{}^{\mathrm{T}} R_i h_i\right)}{\sum (h,r,t) \in S_u^1 exp\left(v_1{}^{\mathrm{T}} Rh\right)}, \tag{12}$$

Additionally, in the item vector $\boldsymbol{v_2}$ modeled using Node2vec-side with the head hi and relationship $\boldsymbol{r_i}$, the correlation probability p_{2i} for each triplet ($\boldsymbol{h_i}$, $\boldsymbol{r_i}$, $\boldsymbol{t_i}$) is calculated.

$$p_{2i} = softmax\left(cos\left(v_2^T, (h_i + r_i)\right)\right) = \frac{exp\left(cos\left(v_2^T, (h_i + r_i)\right)\right)}{\sum (h,r,t) \in S_u^1 exp\left(cos\left(v_2^T, (h_i + r_i)\right)\right)}, \tag{13}$$

After obtaining the correlation probabilities, all the tail entities t_i in $\mathcal{S}_u^1$ are multiplied by the corresponding correlation probabilities p_{1i} and p_{2i}, respectively, returning vectors o_{1u}^1 o_{2u}^1, and where o_{ju}^1 (j = 1,2) can be regarded as the first response of User u to the historical click record of item $\boldsymbol{v}$, which is used to form the first propagated dynamic preference portrait of User u.

$$o_{1u}^1 = \sum_{(h_i, r_i, t_i) \in S_u^1} p_{1i} t_i, \tag{14}$$

$$o_{2u}^1 = \sum_{(h_i, r_i, t_i) \in S_u^1} p_{2i} t_i, \tag{15}$$

$$o_u^1 = f\left(o_{1u}^1, o_{2u}^1\right), \tag{16}$$

Here the fusion function $f(x)$ is chosen as the linear regression function $y = wx + b$.

The user dynamic preference portrait vector $\boldsymbol{U}$ can be obtained according to Equation (16) as follows:

$$U = o_u^1 + o_u^2 + \cdots o_u^H, \tag{17}$$

The final image v of the item is obtained by fusing the two item representations $\boldsymbol{v_1}$ and $\boldsymbol{v_2}$, i.e.,

$$v = g(v_1, v_2), \tag{18}$$

Here, the fusion function $g(x)$ is chosen as the linear regression function $g = wx + b$, where the loss function min$\mathcal{L}$ of this modular model is

$$\min\mathcal{L} = \mathcal{L}_{RS} + \mathcal{L}_{KG} + \mathcal{L}_{REG} = \sum_{(u,v)\in Y} F(\hat{y}_{uv}, y_{uv}) + \frac{\lambda_2}{2}\sum_{r\in\mathcal{R}_1}\left\|I_r - E_1{}^{\mathrm{T}}\mathrm{R}_1\mathrm{E}_1\right\|_2^2 + \\ \frac{\lambda_2'}{2}\sum_{(h_i,r_i,t_i)\in S_u^k}(1-\cos(t_i,(r_i+h_i))) + \frac{\lambda_1}{2}\left(\|V_1\|_2^2 + \|\mathrm{E}_1\|_2^2 + \sum_{r\in\mathcal{R}_1}\|R_1\|_2^2 + \|V_2\|_2^2 + \|\mathrm{E}_2\|_2^2 + \sum_{r\in\mathcal{R}_2}\|r_i\|_2^2\right) \tag{19}$$

where $F(\hat{y}_{uv}, y_{uv})$ denotes the cross-entropy function; and $V_{1,2}$, $E_{1,2}$, and $\boldsymbol{R_{1,2}}$ denote the item and knowledge graph entities and relationships under the two item representations, respectively. The loss function consists of three parts, where $\mathcal{L}_{RS}$ denotes the value of the cross-entropy loss function between the predicted probability and the true value, $\mathcal{L}_{KG}$ denotes the item error in the knowledge graph, and $\mathcal{L}_{REG}$ is the L2 regularization to prevent overfitting of the model.

3.3. Hybrid Forecast

In this section, as shown in Figure 3, the probability values are calculated separately for the constructed user profiles, and the final recommendation prediction is made by combining the two probability values. In the Embedding layer, the input data include item portraits and user portraits; in the preference prediction part, a hybrid recommendation approach is adopted, i.e., on the one hand, the prediction score $\hat{y}_{uv}$ is calculated based on the static history portraits of users and item portraits, and on the other hand, the prediction score $\hat{y}'_{uv}$ is calculated based on the dynamic preference portraits of users and item portraits; the target score y_{uv} is continuously optimized, and, finally, the preference score of a user for a specific item is output in the output layer by combining the two prediction scores.

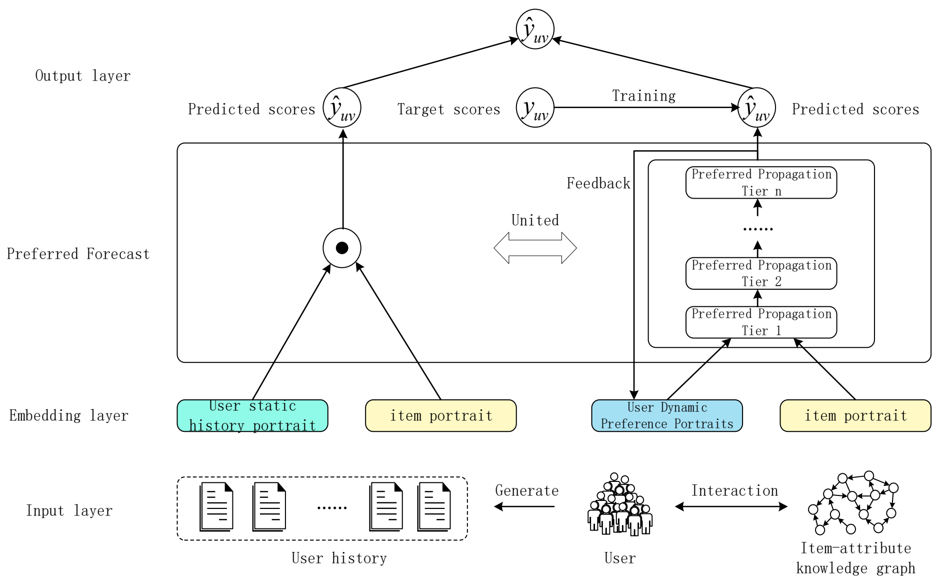


Figure 3. Framework diagram of NRH recommendation model.

The core part of the NRH recommendation model is preference prediction. First, the Node2vec-side knowledge representation is used to construct the item–attribute knowledge graph for the item portrait. Then the static history portrait of the user is built based on the user's historical click data, and the prediction probability $\hat{y}'_{uv}$ is obtained by calculating the cosine similarity between the static history portrait vector of User u and item vector v. Moreover, the parameter β is set to adjust the score. The prediction score $scores_1$ is obtained as follows:

$$scores_1 = \hat{y}'_{uv} = cos(u, v') - \beta, \tag{20}$$

where β represents the score adjustment parameter, which can be adjusted according to different recommended data.

Secondly, the single knowledge representation model is changed in the dynamic preference prediction part. The item portrait based on Node2vec-side knowledge representation is added. The knowledge representation of each node and relationship is obtained based on the knowledge graph, and the tail node is predicted in the Ripple set using cosine similarity. After that, the dynamic preference portrait of User u is then obtained by preference propagation iterations based on the Hop count and feedback to update the portrait of item v. The predicted click probability $\hat{y}_{uv}$ and score $scores_2$ are output by combining User u and item v as follows:

$$\text{scores}_2 = \hat{y}_{uv} = \sigma\left(U^T v\right), \tag{21}$$

where

$$\sigma(x) = \frac{1}{1 + \exp(-x)}, \tag{22}$$

The NRH recommendation model takes User u and item v as initial data input, respectively, and finally obtains the click probability (preference score) of User u on item v. Combining (20) and (21) yields the combined prediction score scores of User u for item v:

$$scores = (1 - \alpha)scores_2 + \alpha scores_1, \tag{23}$$

where if scores $\geq$ 0.5, the model can recommend item v to User u. If scores < 0.5, the model will not recommend item v to User u. Specifically, the algorithm for NRH is described as Algorithm 1 follows:

Algorithm 1 NRH algorithm

```
Input: recommendation item knowledge graph triad set N, user history data click_history.
Output: recommendation_list
Initialization: set random walk parameters: p, q, implicit feedback parameters: $\alpha$; set model
hyperparameters: embedding_dimension, walk_length, num_walks, window_size, epochs
1. Build item–attribute knowledge graph.
   (a) Initialize the graph: graph = EmptyGraph()
   (b) Initialize item attribute data: item_list = get_item_list(N); item_attributes =
   get_item_attributes(item_list, N)
   (c) add item nodes: for item in item_attributes: graph.add_node(item, 'item')
   (d) add attribute node: for attribute in attributes: graph.add_node(attribute, 'attribute')
   (e) associate item and attribute: for item, attribute in item_attributes:
    graph.add_edge(item, attribute)
2. Item portrait construction:
   (a) Construct item vector by Node2vec-side algorithm: $v_2$ = Node2vec_side(graph, p, q,
    walk_length, num_walks)
3. Construct a static history portrait of the user:
   (a) initialize user static history vector: user_static_profile =
   initialize_embeddings(embedding_dimension)
   (b) for user in click_history.
         for item in click_history [user].
              item_embedding = $v_2$ [item]
              user_static_profile [user] + = item_embedding
         user_static_profile [user] = user_embeddings[user] − cold_item_embedding
4. User preference propagation:
   (a) Initialize the user dynamic preference profile: user_dynamic_profile = {}
   (b) Update the profile according to user preference propagation: user_dynamic_profile,
    $v_1$ = RippleNet(user_static_profile, $v_2$, click_history)
   (c) User profile construction: user_embeddings = User_profile_construction(user_static_profile,
    user_dynamic_profile)
   (d) Item portrait update: item_embeddings = Item_portrait_fusion($v_1$,$v_2$)
```

Algorithm 1 *Cont.*

```
5. Predict preference scores.
   (a) initialize recommendation score: recommendation_scores = {}
   (b) for user in user_embeddings.
         for item in item_list.
            if item not in click_history[user].
               user_embedding = user_embeddings[user]
               item_embedding = item_embeddings[item]
               score = compute_score(user_embedding, item_embedding)
               recommendation_scores[(user, item)] = score
6. Generate recommendation results.
   (a) Sort by recommendation score: recommendation_list =
    sort_recommendations(recommendation_scores)
7. Return the recommendation results recommendation_list.
```

4. Experiments

4.1. Dataset

In the experiments of this paper, as shown in Table 1, the following three public datasets, namely, MovieLens-1M, Book-Crossing, and Last.FM, were utilized, and the item knowledge graph constructed in the datasets was used.

Table 1. Basic statistical information of the dataset.

Dataset	#Users	#Items	#Interaction	#Triples	#Entities	#Relationship
MovieLens-1M	6036	2445	753,772	1,241,995	182,011	12
Book-Crossing	17,860	14,967	139,746	151,500	77,903	25
Last.FM	1872	3846	42,346	15,518	9366	60

MovieLens-1M is a public dataset on movie recommendations, which includes 1,000,209 rating data from 6040 users and 3706 movies. The corresponding knowledge graph contains 1,241,995 triples with 182,011 entities and 12 relationships.

Book-Crossing is a public dataset on book recommendations, including 1,149,780 ratings data from 105,283 users and 340,555 books. The corresponding knowledge graph contains 151,500 triples with 77,903 entities and 25 relationships.

Last.FM is a public dataset on music recommendations, including 92,834 rating data from 1892 users and 17,632 artists, and the corresponding knowledge graph contains 15,518 triples with 9366 entities with 60 relations.

4.2. Experimental Environment and Evaluation Index

The operating system used for the experiments was ubuntu, version Ubuntu 22.04.2 LTS, and the experimental environment used included cuda10.2, python3.8.13, torch1.10.1 + cu102, numpy1.23.5, etc. The values of the NRH model parameters are given in Table 2, where d denotes the vector dimension, H represents the Hop count, lr indicates the learning rate, λ_1 denotes the weight of the l2 regularization term, λ_2 denotes the weight of the KGE term, λ_2' denotes the weight of the Node2vec-side term, and α denotes the weight of the mixed recommendation mode module. In this paper, each dataset was processed. Each dataset was divided into three parts, namely, training set, validation set, and test set, and the optimal results were selected after conducting five experiments.

Table 2. Hyperparameters of the NRH recommended model.

Dataset	d	H	λ_1	λ_2	λ_2'	α	lr
MovieLens-1M	32	2	1×10^{-7}	1×10^{-2}	1×10^{-5}	0.4	0.02
Book-Crossing	16	3	1×10^{-7}	1×10^{-2}	1×10^{-5}	0.6	0.025
Last.FM	32	2	1×10^{-7}	1×10^{-2}	1×10^{-4}	0.3	0.01

The recommendation accuracy comparison experiment used ACC (accuracy) and AUC (area under curve) as evaluation metrics. Among them, ACC denotes the accuracy rate of recommendation, and AUC indicates the area under the ROC curve with the coordinate axes, taking values between 0.5 and 1. The ROC graph is generally used to judge the merits of binary classifiers, and a graph with FPR (false positive rate) as the horizontal coordinate and TPR (actual positive rate) as the vertical coordinate is obtained, and the larger the AUC value, the better the overall recommendation ability of the algorithm, where the ACC values are calculated as follows:

$$ACC = \frac{TP + TN}{N}, \tag{24}$$

where *TP* (true positive) indicates that the positive cases are correctly classified as positive cases, *TN* (true negative) indicates that the negative cases are correctly classified as negative cases, *TP* + *TN* indicates the total number of correctly predicted cases in the test sample, and *N* represents the number of samples tested.

Recommendation diversity is used to measure the variation of items in the user recommendation list. In this paper, the type (genre) of items was used as the basis for classification, and the diversity value was used as the evaluation index for the experiment. Among them, the formula of diversity is shown as follows:

$$diversity = \frac{series\_count(N)}{N}, \tag{25}$$

Information on the dataset used for the recommendation diversity experiment is given in Table 3. *N* denotes the first *N* items in the user recommendation list, and series_count (*N*) represents the number of item types (genres) in *N* items. In the movie, book, and music recommendation datasets, different numbers of objects were selected according to the item types (genres) and targeted to the user groups in the system. Suppose *N* = 10 and User u has 10 items in their recommendation list, which covers 6 different item types (genres); then the following calculation is conducted according to the above formula: diversity = 6/10 = 0.6. This means that the recommendation system has a recommendation diversity value of 0.6 for this user, which indicates that the percentage of item types in the recommendation results is 60% with some diversity. Using this item type-based classification basis, the diversity of item types in the recommendation results can be measured. By calculating the diversity value, it is possible to evaluate whether the system can provide diverse item types during the recommendation process and thus judge the performance of the model in terms of recommendation diversity.

Table 3. Basic statistical information about the dataset.

Dataset	#Series	#Items	#Top_N
MovieLens-1M	30	47,137	10
Book-Crossing	56	2568	10
Last.FM	34	66	20

4.3. Experimental Results and Analysis

4.3.1. Recommended Accuracy

In this section, we constructed recommendation experiments using NRH recommendation models based on the knowledge graphs provided by movie, book, and music recommendation datasets using current mainstream recommendation algorithms as comparisons, including the knowledge-embedding independent learning-based recommendation algorithms CKE and DKN; meta-path-based recommendation algorithms such as PER [17]; classical recommendation algorithms such as LibFM [24] and Wide&Deep [25]; and the knowledge graph-based recommendation algorithms RippleNet model, MKR [26], KGAT [19], KGNN-LS [27], KGCN [28], and CKAN [29]. Knowledge ripple graph convolutional network (KRGCN) was used as a control group. KRGCN introduced graph convolutional aggregation based on the RippleNet model based on the introduction of graph convolutional aggregation information; a partial explanation of the comparison model is shown below, and the experimental results are shown in Table 4:

1. LibFM: a feature-based factorization model for CTR scenarios that take as inputs to the model the user ID and item ID and the corresponding entity embedding connections learned through the TransR algorithm.
2. DKN: a framework for news recommendations using knowledge graphs. It treats entity embeddings and word embeddings as multiple channels and combines them in a convolutional neural network for click-through rate prediction.
3. KGNN-LS (knowledge-aware graph neural networks with label smoothness regularization for recommender systems): a graph neural network model that uses label smoothness loss as a regularization term to prevent the model from overfitting. The model's parameters are as follows: epoch is 15, the number of adjacent samples considered is 16, dim is 32, the regularization term coefficient is 0.0001, and lr is 0.024.
4. KGCN: a model for mining the importance of items in the knowledge graph using graph neural networks. The parameters of the model in the experiment are as follows: epoch is 10, the number of adjacent samples considered is 4, dim is 32, the regularization term coefficient is 0.0001, and lr is 0.02.
5. MKR: a deep end-to-end framework that uses alternate learning to assist recommendation tasks using knowledge graph embedding tasks. The experiment model parameters are as follows: epoch is 20, dim is 8, regularization term coefficient is 0.00001, recommendation task lr is 0.02, and knowledge graph learning task lr is 0.01.
6. CKAN (collaborative knowledge-aware attentive network for recommender systems): a collaborative knowledge-aware attentive network model that uses a heterogeneous propagation strategy to encode knowledge attribute associations and user–item synergistic signals.

Table 4. Comparison of AUC and ACC of different models under each dataset.

Model	**MovieLens-1M**		**Book-Crossing**		**Last.FM**	
	AUC	**ACC**	**AUC**	**ACC**	**AUC**	**ACC**
CKE	0.796 (−14.1%)	0.739 (−13.9%)	0.674 (−10.1%)	0.635 (−10.9%)	0.744 (−10.9%)	0.673 (−12.0%)
DKN	0.655 (−29.5%)	0.589 (−31.4%)	0.621 (−17.1%)	0.580 (−18.7%)	0.602 (−27.9%)	0.581 (−24.1%)
PER	0.712 (−23.4%)	0.667 (−22.3%)	0.623 (−16.8%)	0.588 (−17.5%)	0.633 (−24.2%)	0.596 (−22.1%)
LibFM	0.892 (−4.0%)	0.812 (−5.4%)	0.685 (−8.5%)	0.639 (−10.4%)	0.777 (−6.9%)	0.709 (−7.3%)
Wide&Deep	0.903 (−2.8%)	0.822 (−4.2%)	0.711 (−5.1%)	0.623 (−12.6%)	0.756 (−9.5%)	0.688 (−10.1%)
MKR	0.917 (−1.3%)	0.843 (−1.8%)	0.734 (−2.0%)	0.682 (−4.4%)	0.795 (−4.8%)	0.730 (−4.6%)
RippleNet	0.921 (−0.9%)	0.844 (−1.6%)	0.729 (−2.7%)	0.662 (−7.2%)	0.768 (−8.0%)	0.691 (−9.7%)
KGAT	0.872 (−6.2%)	0.802 (−6.5%)	0.651 (−13.1%)	0.624 (−12.5%)	0.716 (−14.3%)	0.647 (−15.4%)
KGNN-LS	0.913 (−1.8%)	0.840 (−2.1%)	0.689 (−8.1%)	0.635 (−10.9%)	0.795 (−4.8%)	0.726 (−5.1%)
KGCN	0.907 (−2.4%)	0.831 (−3.2%)	0.694 (−7.3%)	0.635 (−10.9%)	0.794 (−4.9%)	0.723 (−5.5%)
CKAN	0.915 (−1.5%)	0.840 (−2.1%)	0.738 (−1.5%)	0.655 (−8.1%)	0.831 (−0.8%)	0.744 (−2.9%)
KRGCN	0.903 (−2.8%)	0.841 (−2.0%)	0.730 (−2.5%)	0.666 (−6.6%)	0.805 (−3.6%)	0.738 (−3.5%)
NRH	**0.929**	**0.858**	**0.743**	**0.713**	**0.837**	**0.765**

In this paper, we tested the performance of different recommendation algorithms under three datasets, as shown in Table 4. Compared with other mainstream recommendation algorithms, the AUC and ACC of the NRH recommendation model were better under the three datasets. The main reason is that the model adopts the Node2vec-side algorithm for item portrait modeling, profoundly explores the association relationship between items, divides the user portrait into two parts (static history portrait and dynamic preference portrait), and uses hybrid recommendations to enhance the accuracy of recommendations.

By comparing the experimental results, it was found that compared to the traditional recommendation algorithms CKE and DKN that adopt the translation model (Trans series) as the representative knowledge-based embedding, MovieLens-1M and NRH improved the AUC and ACC metrics by 13 percentage points and 12 percentage points, respectively, compared with CKE, and they improved the AUC and ACC metrics by 27 percentage points, respectively, compared with DKN. This may be because CKE adopts a translation-based model as its knowledge representation, which uses independent learning to represent knowledge graph entities and relationships, thus ignoring the network structure characteristics of the knowledge graph itself and making it challenging to discover the correlation relationships among entities. At the same time, DKN was initially applied in the field of long-text news recommendation and tends to analyze text content only when the text content contained in the entities is long enough to ensure its recommendation accuracy. In the experimental dataset involving entities, relationships, and attributes with short ranges, DKN cannot obtain valuable information, which also causes its results to be biased.

Compared with the MKR model, the model in this paper effectively exploits the neighborhood relationships among entities in the knowledge graph and can extract user features effectively compared with the KGCN model. In addition, although KGAT overlays multiple attentional embedding propagation layers to convey embedding information, KGAT cannot capture the complex semantics of user–item connections, so its performance is inferior to that of KGCN, RippleNet, KRGCN, and NRH. The NRH recommendation model is more effective than the original model regarding recommendation results, as it improves the item feature learning part of the RippleNet model by using a graph convolutional network to aggregate higher-order items. The NRH uses Node2vec-side knowledge representation to construct user portraits and fully explores the association relationship between entities through little random wandering, taking into account the homogeneity and isomorphism of nodes in the knowledge graph, reducing the impact of localization of user portrait update in the original model, and making the recommendation results more accurate. CKAN generates recommendations using collaborative filtering techniques. However, standard collaborative filtering techniques may not effectively capture the user's preferences and interests when the user–item interaction data are sparse. As a result, in sparse data, the recommendation effectiveness of CKAN may be constrained. In this study, NRH employed a hybrid recommendation model that could effectively capture users' preferences and interests and produce more individualized recommendations to perform better in the face of scant user activity data.

4.3.2. Recommended Diversity

The current research on personalized recommendation algorithms mainly focused on improving recommendation accuracy, the lack of which leads to the frequent appearance of similar types of items and quickly causes the "information cocoon" problem. In this section, according to the experimental environment and metrics in Section 4.2, an experimental study on the diversity of recommended items was conducted. The comparison model was the original RippleNet model, and the experimental results under three datasets are shown in Figure 4.

The experimental results demonstrated that the NRH recommendation model provides more diverse recommendations than the conventional RippleNet. Typically, users are distributed across both recommendation models. Nevertheless, the overall standard distribution curve of users under the NRH recommendation model was displaced to the right relative to RippleNet, indicating that when making recommendations to users, the NRH recommendation model tends to be more item-centric. Diversity, precisely the static history portrait, can more accurately characterize users' long-term interests and preferences. In contrast, the dynamic preference portrait can more accurately reflect the users' short-term preferences and changing trends, and the combination of the two can more accurately describe the users' interests and requirements. In addition, the separation processing of user portraits enables the NRH model to manage users' historical and current behaviors better, avoiding some unnecessary interference and increasing the diversity of recommendation outputs.

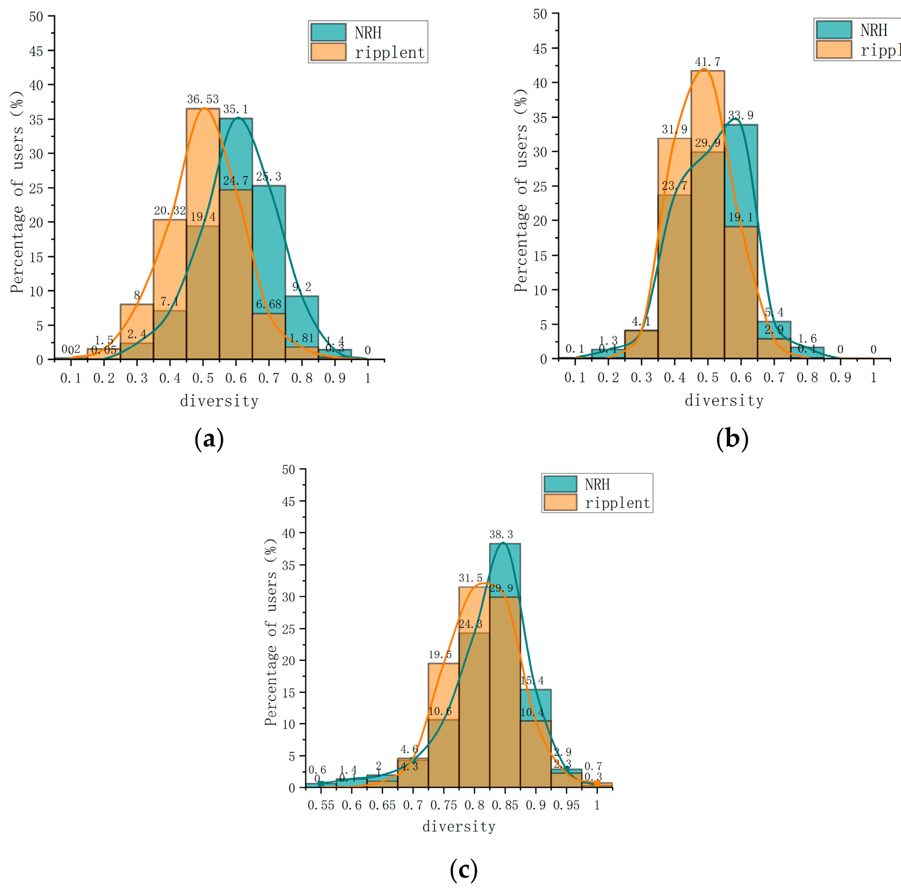


Figure 4. Experimental results of diversity under each dataset: (**a**) experimental results of MovieLens-1M; (**b**) experimental results of Book-Crossing; (**c**) experimental results of Last.FM.

According to the box plots in Figure 5, NRH outperformed RippleNet in terms of diversity of recommendation results, as shown by the fact that the median line and box height of NRH boxes were much more significant than RippleNet in the three datasets. The Book-Crossing dataset showed that NRH had a wider variety of box top and bottom edges, indicating a more substantial diversity difference. The recommendation results also spanned a more comprehensive range of genres. The box plot shows that the NRH box length in the MovieLens-1M dataset was comparable to RippleNet. However, the NRH box's upper and lower edges are more significant than RippleNet's, suggesting that NRH's recommendation results contained more high-scoring data points than RippleNet's. This indicates that NRH has an advantage over RippleNet regarding suggestion variety. In comparison, RippleNet's lower upper and lower box margins may suggest that some of the recommendation results contained data points with lower scores or some outliers, leading to a significantly lower level of suggestion quality. NRH includes a disproportionate number of outlier points in the Last.FM dataset, which suggests that the recommendation results have more rare or uncommon types. The Last.FM dataset, which deals with music recommendations, includes people with vastly different musical tastes, which could be a contributing factor. Due to this variation, it may be challenging for the NRH model to faithfully represent all users' tastes, resulting in some uncommon or odd suggestions. To improve the diversity of recommendation results, NRH uses Node2vec-side for item portrait representation and network structure information to learn node embeddings, keeping the distance between neighboring nodes in the embedding space as close as possible.

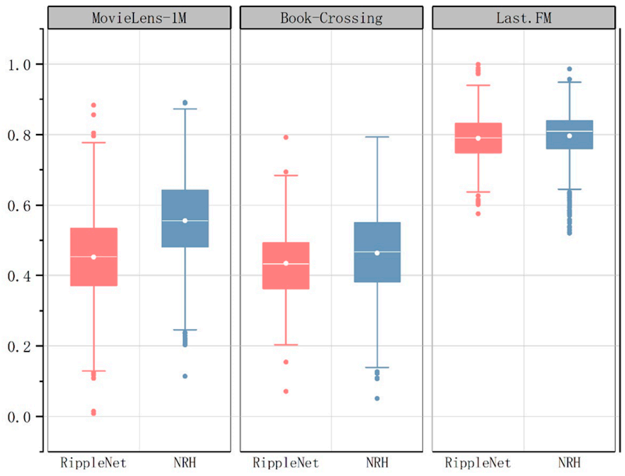


Figure 5. Comparison of RippleNet and NRH recommended diversity.

4.4. Ablation Experiments

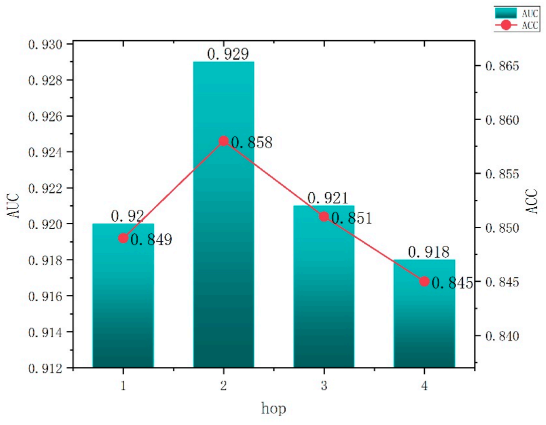
Table 5 illustrates the impact of various components on the model's recommendation performance. The Node2vec-side knowledge representation, static history portrait of users, and hybrid push mode affected the model's recommendation performance. In the case of RippleNet-based recommendations, the introduction of Node2vec-side knowledge representation improved the average accuracy and AUC by 0.89 and 0.72 percentage points, respectively, on the three datasets. This is likely because the model could better capture the relationship and contextual information between nodes after introducing Node2vec-side knowledge representation. The average precision and AUC of the model improved by 2.94 and 1.24 percentage points, respectively, after introducing both Node2vec-side knowledge representation and a static history portrait of users. This is likely because the introduction of a static history portrait of users could fully account for the historical behaviors and preferences of users. After incorporating Node2vec-side knowledge representation and the hybrid recommendation model, the average accuracy and AUC of the model increased by 4.64 and 2.24 percentage points, respectively. The hybrid recommendation model may provide more complete and accurate results. The diversity and scope of recommendations can be expanded by combining multiple methods.

Table 5. Effect of different components on model recommendation results.

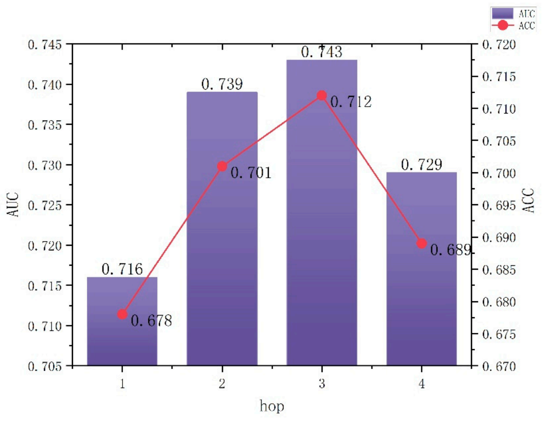
Modules			MovieLens-1M		Book-Crossing		Last.FM	
Node2vec-Side	Static Portrait	Hybrid Forecast	AUC	ACC	AUC	ACC	AUC	ACC
√			0.923	0.845	0.734	0.670	0.778	0.701
√	√		0.925	0.850	0.737	0.686	0.786	0.722
√		√	0.927	0.851	0.735	0.696	0.809	0.746
√	√	√	0.930	0.856	0.742	0.711	0.835	0.753

4.5. Hyperparametric Analysis

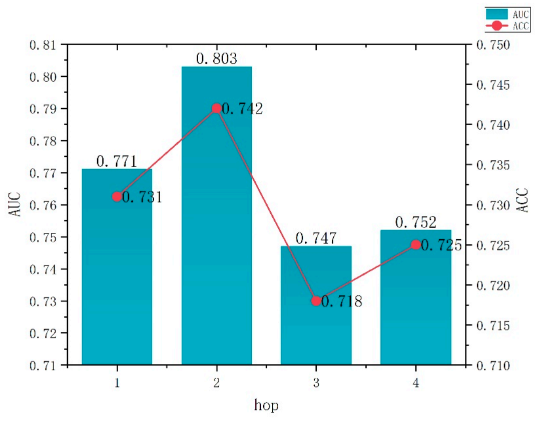
In this section, we focus on the effects of the parameters Hop and α on the model in NRH. As can be seen in Figure 6, the higher the Hop value, the better. For the MovieLens-1M dataset and the Last.FM dataset, the model performed best when Hop was at 2 (i.e., when the number of jumps was at two hops), while the Book-Crossing dataset was at a Hop value of 3. The best results are recommended. It can be seen from Figure 7 that the model performed best under different datasets with different α, indicating that the weighting of the two recommendation methods had additional adaptability and did not work best under the average distribution. Additionally, according to the experiments, α had different effects on the recommendation performance of other datasets. FM was more influenced by α.



(**a**)

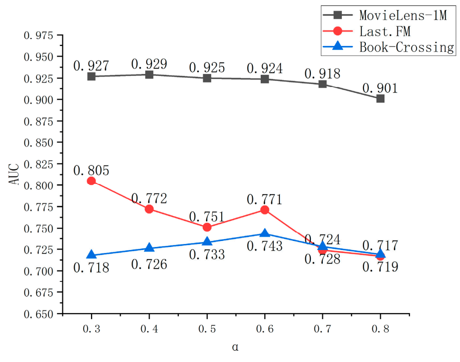


(**b**)

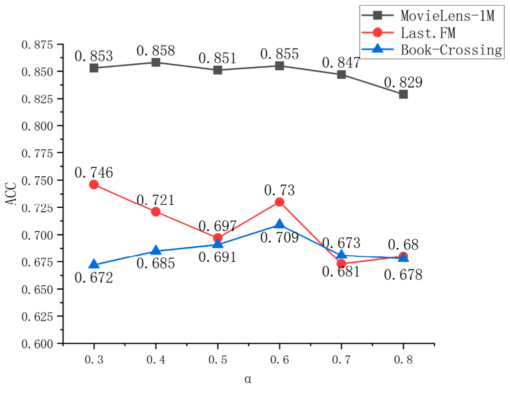


(**c**)

Figure 6. AUC and ACC for different Hop: (**a**) experimental results of MovieLens-1M; (**b**) experimental results of Book-Crossing; (**c**) experimental results of Last.FM.



(**a**)



(**b**)

Figure 7. AUC and ACC of different α: (**a**) AUC for different α under each dataset; (**b**) ACC for different α under each dataset.

4.6. Time Complexity Analysis

The time cost of NRH is mainly composed of two parts, namely, preprocessing and model training, where the preprocessing part primarily uses Node2vec-side for the construction of item portraits, and node2vec-side fused knowledge representation adds relational representation and diversified wandering strategies based on Node2vec. The average time complexity of node2vec random sampling for each sampling point is $O\left(\frac{l}{k(l-k)}\right)$, the time complexity of the skip-gram part using hierarchical softmax optimization is $O(\log_2 N_v)$, and the time complexity of the relational aggregation part is $O(rlN_v)$. The time complexity of the Node2vec-side preprocessing stage is $O\left(\frac{2lN_v}{k(l-k)} + 2\log_2 N_v + 2rlN_v\right)$ under the case of using two wandering strategies: l denotes the length of random wandering, k means the number of domain nodes, N_v represents the number of nodes, and r represents the random wandering number of times. The model training part was mainly combined with RippleNet for update iterations, so the time complexity was similar to RippleNet, and the average running time comparison between NRH and RippleNet is given in Table 6.

Table 6. Comparison of time consumption of NRH and RippleNet models.

Dataset	Model	Preprocessing (s)	Training Time(s)			Total Time (s)
			Average Per Epoch	Epochs	Training Cost	
MovieLens-1M	RippleNet	5.81	11.27	60	676.2	682.01
	NRH	496.83	11.85	20	237	733.83
Book-Crossing	RippleNet	2.48	2.06	50	103	105.48
	NRH	796.41	2.21	20	44.2	840.61
Last.FM	RippleNet	0.21	0.54	30	16.2	16.41
	NRH	14.40	0.54	10	5.4	19.8

Compared to the RippleNet model, NRH has some drawbacks regarding temporal complexity. The NRH model introduces a Node2vec-side item painting representation stage, which increases the preprocessing part of the model time cost but in the model training stage, which is the main reason why the preprocessing time of the NRH model is substantially longer than that of the RippleNet model. However, adding Node2vec-side item pictures does not increase the processing time NRH. However, it can lower the number of training epochs, which can be considered in the future to enhance the NRH model's preprocessing procedure and decrease the preprocessing time overhead.

5. Conclusions

In user portrait localization, the conventional RippleNet recommendation model changes the user preference set by randomly selecting favored items. In contrast, the suggested NRH recommendation model deeply examines the association relationship between items and leverages Node2vec-side for item portrait modeling. In contrast, the static history portrait and dynamic preference portrait based on user history information are separated during the user portrait construction stage, which is beneficial in removing the issue with the localization of user portrait updates. A hybrid recommendation method based on collaborative item filtering and knowledge graph can solve the localization problem of traditional user profile updates and improve the accuracy and variety of personalized recommendations. The user profile construction stage is divided into two parts based on user history information, static history profiles, and dynamic preference profiles. The deep association relationships between nodes can be better explored using the NRH model proposed in this research. This model is primarily applicable to personalized recommendation situations that need to balance diversity and accuracy, such as fields for music and movie recommendations.

Nevertheless, this study also identifies some NRH flaws. Due to the complex structure of the model and the increase in the number of parameters, NRH models have a higher computational cost and complexity than traditional models. In addition, NRH models have unique requirements for feature engineering and pre-processing steps, which may significantly increase computational resources and time overhead when applying NRH models to large datasets. The NRH model's performance could suffer if the suggestion data are erratic or not closely aligned with user preferences. To ensure the accuracy and reliability of the recommendation data, thorough data cleaning and feature selection must be made before NRH models are used. Additionally, NRH models are challenging to interpret and must inform users how to evaluate recommendations. Further, many domain and dataset features may impact NRH performance.

We can think about further mining the knowledge graph node information, mining the association relationship between nodes by adding more node features, optimizing and accelerating the training and recommendation process of the model to reduce the computational cost and time overhead, allowing users to understand the reason for and basis of recommendation results using visualization and user interface design, etc., as well as performing cross-domain. Additionally, we take into account enhancing user involvement and feedback methods as well as modifying models through active learning to match users' unique demands. Last, we concentrate on algorithm fairness, create suitable assessment measures and fairness constraints, and minimize bias against particular users or items.

Author Contributions: W.N. designed the proposed method and wrote the paper; W.N. and H.L. wrote the code and performed the experiments; W.N. and X.M. analyzed the data; Y.D. modified the paper and offered support. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Double-class Special Project on Cyberspace Security and Law Enforcement Technology of the Chinese People's Public Security University (2023SYL07).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Not applicable.

Acknowledgments: The authors thank the Major Project of Basic Research Business Fund of People's Public Security University of China in 2021 (2021JKF105) and the Project of Building a High-level Non-Staff Institution of People's Public Security University of China (2021FZB13) for covering the costs to publish open access and the costs incurred when writing this study. In addition, the authors thank the anonymous reviewers for their insightful comments that helped improve the quality of this study.

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

References

1. Warren, J.; Marz, N. *Big Data: Principles and Best Practices of Scalable Realtime Data Systems*; Simon and Schuster: New York, NY, USA, 2015.
2. 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]
3. Huang, L.; Chen, H.; Wang, X.; Chen, G. A fast algorithm for mining association rules. *J. Comput. Sci. Technol.* **2000**, *15*, 619–624. [CrossRef]
4. Tewari, A.S. Generating Items Recommendations by Fusing Content and User-Item based Collaborative Filtering. *Procedia Comput. Sci.* **2020**, *167*, 1934–1940. [CrossRef]
5. Zeng, L.; Xie, X. Collaborative Filtering Recommendation Based On CS-Kmeans Optimization Clustering. In Proceedings of the ICIIP 2019: 2019 4th International Conference on Intelligent Information Processing, Xi'an, China, 16–17 November 2019.
6. Amit, S. *Introducing the Knowledge Graph*; Official Blog of Google: California, USA, 2012.
7. Chicaiza, J.; Valdiviezo-Diaz, P. A comprehensive survey of knowledge graph-based recommender systems: Technologies, development, and contributions. *Information* **2021**, *12*, 232. [CrossRef]
8. Wang, H.; Zhang, F.; Wang, J.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Exploring High-Order User Preference on the Knowledge Graph for Recommender Systems. *ACM Trans. Inf. Syst.* **2019**, *37*, 32. [CrossRef]

9. Grover, A.; Leskovec, J. node2vec: Scalable Feature Learning for Networks. In Proceedings of the Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016.
10. Yang, D.; Guo, Z.; Wang, Z.; Jiang, J.; Xiao, Y.; Wang, W. A knowledge-enhanced deep recommendation framework incorporating GAN-based models. In Proceedings of the IEEE International Conference on Data Mining (ICDM), Singapore, 17–20 November 2018; pp. 1368–1373.
11. Dong, Y.; Chawla, N.V.; Swami, A. metapath2vec: Scalable Representation Learning for Heterogeneous Networks. In Proceedings of the Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017.
12. Lin, Y.; Liu, Z.; Sun, M.; Liu, Y.; Zhu, X. Learning entity and relation embeddings for knowledge graph completion. In Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, Austin, TX, USA, 19 February 2015.
13. Zhang, F.Z.; Nicholas, J.Y.; Lian, D.F.; Xie, X.; Ma, W.-Y. Collaborative Knowledge Base Embedding for Recommender Systems. In Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, San Francisco, CA, USA, 13–17 August 2016; pp. 353–362.
14. 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 October 2018; pp. 297–305.
15. Bordes, A.; Usunier, N.; Garcia-Duran, A.; Weston, J.; Yakhnenko, O. Translating Embeddings for Modeling Multi-relational Data. In *Proceedings of Neural Information Processing Systems 2013*; MIT Press: Cambridge, MA, USA, 2013; pp. 2787–2795.
16. Wang, H.; Zhang, F.; Xie, X.; Guo, M. DKN: Deep Knowledge-Aware Network for News Recommendation. In Proceedings of the WWW '18: Proceedings of the 2018 World Wide Web Conference, Lyon, France, 23–27 April 2018.
17. Yu, X.; Ren, X.; Sun, Y. Personalized entity recommendation: A heterogeneous information network approach. In Proceedings of the 7th ACM International Conference on Web Search and Data Mining, New York, NY, USA, 24–28 February 2014; pp. 283–292.
18. Zhao, H.; Yao, Q.; Li, J.; Song, Y.; Lee, D.L. Meta-Graph Based Recommendation Fusion over Heterogeneous Information Networks. In Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Halifax, NS, Canada, 13–17 August 2017; pp. 635–644.
19. 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, Anchorage, AK, USA, 4–8 August 2019; pp. 950–958.
20. 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.
21. Tang, X.; Wang, T.; Yang, H.; Song, H. AKUPM: Attention-enhanced knowledge-aware user preference model for recommendation. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019; pp. 1891–1899.
22. Cui, H.; Song, W.; Yang, J. Knowledge Ripple Graph Convolutional Network for Recommendation. *Comput. Sci. Explor.* **2022**, 1–10. [CrossRef]
23. Zhang, C.-J.; Zeng, A. Behavior patterns of online users and the effect on information filtering. *Phys. A Stat. Mech. Its Appl.* **2012**, *391*, 1822–1830. [CrossRef]
24. Rendle, S. Factorization machines with libfm. *ACM Trans. Intell. Syst. Technol. TIST* **2012**, *3*, 57. [CrossRef]
25. Cheng, H.T.; Koc, L.; Harmsen, J.; Shaked, T.; Chandra, T.; Aradhye, H.; Anderson, G.; Corrado, G.; Chai, W.; Ispir, M.; et al. Wide & deep learning for recommender systems. In Proceedings of the 1st Workshop on Deep Learning for Recommender Systems, Boston, MA, USA, 15 September 2016; pp. 7–10.
26. Wang, H.; Zhang, F.; Zhao, M.; Li, W.; Xie, X.; Guo, M. Multi-Task Feature Learning for Knowledge Graph Enhanced Recommendation. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019.
27. Wang, H.; Zhang, F.; Zhang, M.; Leskovec, J.; Zhao, M.; Li, W.; Wang, Z. Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems. In Proceedings of the 25th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining, Anchorage, AK, USA, 4–8 August 2019.
28. Wang, H.; Zhao, M.; Xie, X.; Li, W.; Guo, M. Knowledge Graph Convolutional Networks for Recommender Systems. In Proceedings of the World Wide Web Conference, San Francisco, CA, USA, 13–17 May 2019. [CrossRef]
29. Wang, Z.; Lin, G.; Tan, H.; Chen, Q.; Liu, X. CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems. In Proceedings of the SIGIR '20: The 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, China, 25–30 July 2020. [CrossRef]

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.