Knowledge Recommendation Based on Dual Channel Hypergraph Convolution

Yue Li[1*]
[1] Guangdong Communication Polytechnic, School of Information, Guangzhou, 510650, China
[e-mail: liyue@gdcp.edu.cn]
*Corresponding Author: Yue Li

Received December 6, 2022; revised July 27, 2023; accepted November 3, 2023; published November 30, 2023

Abstract

Knowledge recommendation is a type of recommendation system that recommends knowledge content to users in order to satisfy their needs. Although using graph neural networks to extract data features is an effective method for solving the recommendation problem, there is information loss when modeling real-world problems because an edge in a graph structure can only be associated with two nodes. Because one super-edge in the hypergraph structure can be connected with several nodes and the effectiveness of knowledge graph for knowledge expression, a dual-channel hypergraph convolutional neural network model (DCHC) based on hypergraph structure and knowledge graph is proposed. The model divides user data and knowledge data into user subhypergraph and knowledge subhypergraph, respectively, and extracts user data features by dual-channel hypergraph convolution and knowledge data features by combining with knowledge graph technology, and finally generates recommendation results based on the obtained user embedding and knowledge embedding. The performance of DCHC model is higher than the comparative model under AUC and F1 evaluation indicators, comparative experiments with the baseline also demonstrate the validity of DCHC model.

Keywords: Graph convolution, Super-edge construction, Hypergraph neural network, knowledge graph, Recommender system.

1. Introduction

People have gained convenience as a result of the advancement of information technology, but they have also faced the challenge of information overload. How to help users get more valuable data from the huge amount of data quickly and efficiently is an important issue that needs to be solved in the era of information explosion. The essential cause of the information overload is the mismatch between users and data. From the user's point of view, the user does not know the characteristics of the data, the relevance of the data to the user's current needs, or how much value and help the data will have for him or her. The substance and worth of the information are decided by the data itself, but the data itself cannot be perceived by the users to whom the data will be valuable. At this point, we can see that the essence of dealing with information overload is to create a matching channel between the user and the data.

Search and recommendation are now the most popular ways to deal with the information overload [1]. Search-based approaches are limited by their inability to perceive user scenarios and produce results that are tailored to the user's individual needs. Recommender systems are a class of systems that effectively filter data through algorithms to recommend valuable results to users, which is of great value in alleviating information overload [2]. Recommender systems have been widely used in many e-commerce systems, social media and community sharing platforms, and have become an important kernel of these systems.

Traditional recommendation algorithms and deep learning-based recommendation algorithms are the two primary categories of recommendation algorithms [3]. Traditional recommendation algorithms include collaborative filtering models, latent factor models, matrix decomposition models, logistic regression models and hybrid models. The traditional recommendation algorithm model mainly has problems such as poor algorithm generalization ability, feature selection relying on manual design, model prone to overfitting, and recommendation results lacking surprise. With the extensive development of artificial intelligence technology, numerous recommendation algorithms based on deep learning technology have emerged. Compared with traditional algorithms, deep learning-based recommendation algorithms may extract richer data features, have more generalization and expressiveness, and make better use of data and the complex relationship between data. Deep learning-based recommendation methods include multilayer perceptron, convolutional neural networks, recurrent neural networks, gating networks, attention mechanisms, graph neural networks and many other methods. Among them, recommendation algorithms based on graph neural networks are of great significance for the development of recommendation systems. Graph neural networks can directly model graph structure data in the real world and have stronger characterization ability for expressing structural data in non-Euclidean space, which is a new type of neural network incorporating convolutional neural networks and graph embedding methods [4]. However, the graph neural networks currently applied to recommendation systems construct graph structures between users and items, but the structures that exist in the real world between users and users, between users and items, and between items and items are often very complex, and real-world problems often cannot be accurately modelled based on simple graph structures.

Knowledge recommendation is a special kind of recommendation system, which is different from general commodity recommendation. Its recommendation purpose and recommendation content have specific attributes, so its data processing, recommendation process and recommendation content selection are all different from traditional recommendation. With the development and application of artificial intelligence technology, knowledge graph has been widely used in the field of personalized recommendation. Triples are commonly used to store

entities and their relationships in knowledge graphs. Each triple is made up of three parts: a head entity, a relationship, and a tail entity. Triples can not only help us understand the relationship between knowledge entities, but they can also be used to store knowledge entity attributes [5]. Incorporating a knowledge graph into a recommendation system not only facilitates information mining.

The hypergraph is a type of graph structure that can be thought of as a generalized representation of the graph structure. A hypergraph, as opposed to the traditional graph structure in which an edge is associated with two nodes, is a graph in which an edge can contain any number of nodes [6]. This paper proposes a recommendation model with dual-channel hypergraph convolution (DCHC), which is a recommendation model for knowledge data based on hypergraph structure and incorporating knowledge graph technology, based on the superior performance of hypergraph structure in modeling data and considering the value of knowledge graph to realize knowledge recommendation. The model uses hypergraph convolution to achieve user feature extraction and knowledge feature extraction by decomposing the original data into a user subhypergraph and a knowledge subhypergraph, respectively, and experimental comparisons are conducted on two public datasets. The experimental results show that the model we proposed can produce better knowledge recommendation results.

The rest of this paper is as follows: section2 introduces the relevant background knowledge, section3 describes the model structure proposed in this paper in detail, section4 shows the performance of the model through ablation experiments and baseline comparison experiments, and finally, section5 summarizes this paper.

2. Related Work

2.1 Knowledge Recommendation

Knowledge recommendation is a kind of recommendation system. Burke. R [7] proposes that knowledge recommendation is a system that uses user and product knowledge to generate recommendations and infers which knowledge may meet the needs of users. Knowledge content-oriented recommendation differs from general recommendation systems in terms of recommendation content and purpose [8]. Firstly, the recommendation content generated by knowledge-based recommendation is domain knowledge. There is a strong logical relationship between this domain knowledge, and the information behind the knowledge is also very rich. These implicit contents are also related to the knowledge recommendation results. At the same time, the goal of knowledge-oriented recommendation differs from that of general recommendation. The general goal of common e-commerce recommendations is to increase user click-through rates and product exposure while also increasing product sales. The goal of social media recommendations is to target users' interests in order to improve their experience and loyalty to the platform. Knowledge-oriented recommendations, on the other hand, are not intended to force users to buy knowledge or to cater to their preferences, but rather to recommend knowledge content that is truly useful to them in their current context by sensing their situation. Because of these distinctions, the knowledge recommendation process and outcomes differ from those of general recommendations. The current methods of knowledge recommendation primarily include methods based on user contextual information, methods based on association rules, methods based on knowledge graphs, and methods based on social network information.

2.2 Hypergraph Neural Network

Graph convolutional neural network is a deep learning-based generalization of convolutional neural network in graph structure. It can perform end-to-end learning on node feature and structure information at the same time [9-10]. The graph convolutional neural network is useful for nodes and graphs of any topology. The effect on public data sets is far superior to other methods in tasks such as node classification and edge prediction. As a result, applying graph convolutional neural network models to recommendation systems can result in better recommendation results.

C.Berge proposed hypergraph theory in 1970, and it was expanded on more systematically in the subsequent [11]. The hypergraph structure is a more general relational model than traditional graphs with pairwise node relationships. Many real-world problems have been solved using hypergraph theory over the last few decades. Because of hypergraphs' powerful representational capabilities, they can efficiently model networks, data structures, process scheduling, and systems with complex object relationships [12]. Theoretically, hypergraphs can generalize certain theorems on ordinary graphs and even replace several traditional graph theorems with a single hypergraph theorem. In practice, hypergraph structures are becoming increasingly popular over traditional graph structures.

The academic community has always conducted research on hypergraphs, but early research focused primarily on traditional graph theory problems. With the rapid development of hypergraph theory, some applied problems, such as graph node classification problems, graph node importance ranking problems, and some image processing tasks, have been targeted and studied on hypergraphs. The rapid development of neural network research in recent years has resulted in new research directions in hypergraph learning. Researchers are gradually investigating the problems of higher-order interaction relations of hypergraphs, dynamic hypergraphs, and indecomposable hypergraphs using neural networks' superior feature extraction capability and model flexibility. Hypergraph learning has gradually been applied to image processing tasks, biological reaction analysis, recommender systems, and other fields as a means of effectively mining hypergraph information. Typical algorithms for recommendation based on hypergraph structure include DHCF [13], KHNN [14], MHCN [15], etc. Several experimental results show that hypergraph structures outperform traditional graph structures when it comes to representing higher-order data interactions.

2.3 Knowledge Graph

There are already study findings on knowledge graph-based knowledge recommendation. Many modern knowledge-driven recommendation systems make use of knowledge by first modeling entities and relationships as low-dimensional dense vectors, also known as knowledge representation vectors [16]. After that, the knowledge representation vector is input into a deep learning model, which is used to calculate the final recommendation score. The knowledge representation vector's learning process is focused at fitting the topological aspects of the knowledge graph, which reflect the numerous links between things, and this is the key to how the knowledge graph may benefit recommender systems. As a result, creating the knowledge graph's topology is a crucial step in many knowledge-driven recommendation algorithms. The vector distance translation-based model is one of the research approaches for the topology of knowledge networks that are now available. These models' early forerunners include TransE [17], TransD [18], TransH [19], TransR [20], etc.

With the advancement of graph neural network (GNN) research in recent years, an increasing number of knowledge-based recommendation models have adopted the GNN architecture to train entity and relationship representations in order to produce better

recommendation outcomes. Message propagation is accomplished by GNN using embedded representations based on neighbor information aggregation. Typical algorithms include RippleNet [21], KGCN [22] and KGAT [23] models, etc.

In summary, the typical algorithms based on graph neural networks, knowledge graphs, and hypergraph neural networks for recommendation and the advantages and disadvantages of each type of technology are summarized in the following **Table 1**.

Table 1. Summary of technology

	Typical Model	Advantages	Disadvantages
Graph Neural Network	RippleNet, KGCN, KGAT	Can model non Euclidean spatial data for end-to-end training	Cannot express complex unpaired relations
Knowledge Graph	TransE, TransD, TransR, TransH	Knowledge graphs embedded in recommender systems can improve the interpretability and accuracy of the system	The timeliness and convergence of knowledge graphs and domain knowledge need to be considered
Hypergraph Neural Network	DHCF, KHNN, MHCN	Strong generalization capabilities for better modeling of complex relationships in the real world	At a new stage, hypergraph theory and modeling methods need to be further studied in depth.

3. Methodology

This paper proposes a hypergraph convolutional neural network based on dual channels to realize knowledge recommendation by combining the advantages of hypergraph structure in mode18ling practical problems and considering the advantages of knowledge graph technology for knowledge extraction and expression. The model structure proposed in this paper is as shown in **Fig. 1**.

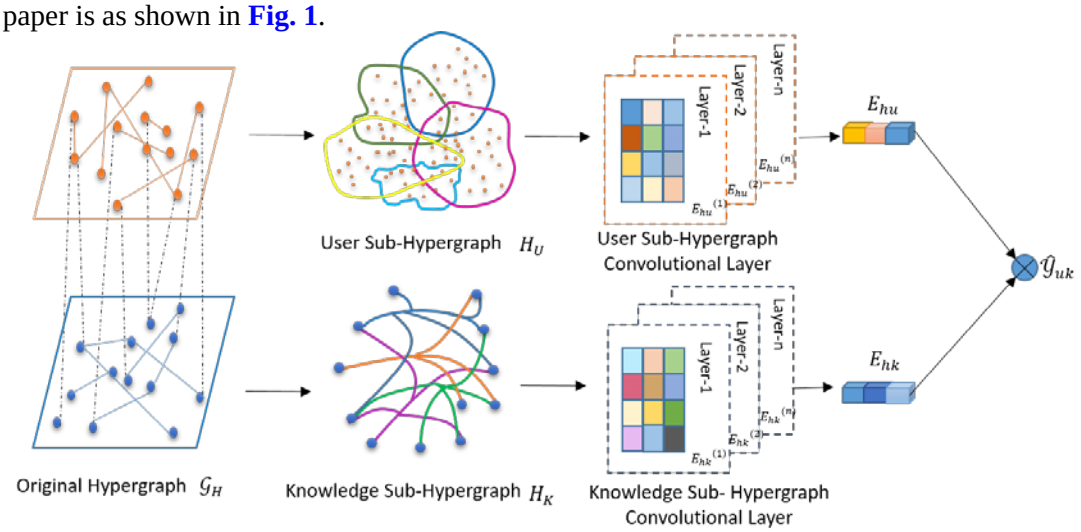


Fig. 1. DCHC model

For the original data, which is made up of users and knowledge, the model recovers the structural relationship between users and data to the greatest extent possible using a hypergraph structure represented by $\mathcal{G}_H$, providing the groundwork for good feature extraction.

Then, for the user data, a user subhypergraph structure denoted as H_U with the user as the node is formed, and for the knowledge data, a knowledge subhypergraph structure indicated as H_K with the knowledge as the node is constructed. For the subhypergraphs under the two channels, the domain information is superimposed layer by layer using hypergraph convolution respectively, so as to obtain the user's embedding expression E_{hu} and the knowledge's embedding expression vector E_{hk}, and finally the obtained embedding expression vector is used to form the prediction recommendation. The details of the implementation are described in the following paper.

3.1 Hypergraph Basic Structure Definition

For a general hypergraph structure, a hyperedge can be associated with 2 or more nodes. The hypergraph structure is generally defined as Equation (1).

$$\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathrm{W}) \tag{1}$$

where $\mathcal{V}$ is the set of all vertices, $\mathcal{E}$ denotes the set of all hyperedges, and W is the weight factor of the hyperedges. $\mathrm{h}(v, e)$ denotes the vertex v is contained by the hyperedge e and exists the following definition as Equation (2).

$$\mathrm{h}(v, e) = \begin{cases} 1 & if\ v \in e \\ 0 & if\ v \notin e \end{cases} \tag{2}$$

In addition, the degree of the vertices and the degree of the hyperedges in the hypergraph can be defined for the hypergraph structure. Where the degree of a vertex v is defined as Equation (3):

$$d(v) = \sum_{e \in \mathcal{E}} \omega(e)\mathrm{h}(v, e) \tag{3}$$

In (3), e is the hyperedge in the set of hyperedges, $\omega(e)$ denotes the weight of that hyperedge in the hyperedge, and the degree of the hyperedge can be defined as Equation (4):

$$\delta(e) = \sum_{v \in \mathcal{V}} \omega(v)\mathrm{h}(v, e) \tag{4}$$

Finally, the degree matrix is represented by D_v and D_e as Equation (5).

$$\begin{aligned} D_v &\in \mathbb{N}^{|\mathcal{V}| \times |\mathcal{V}|} \\ D_e &\in \mathbb{N}^{|\mathcal{E}| \times |\mathcal{E}|} \end{aligned} \tag{5}$$

For the graph structure in the hypergraph, we can define the method of convolution on the graph as Equation (6).

$$\mathrm{E}^{(l+1)} = \sigma(\mathrm{D}_v^{-1/2}\mathrm{HWD}_e^{-1/2}\mathrm{H}^T\mathrm{D}_v^{-1/2}\mathrm{E}^{(l)}\Theta^{(l)}) \tag{6}$$

Among them, $E^{(l)} \in \mathbb{R}^{N \times C}$ and denotes n nodes in layer l with node feature dimension C. When $l = 0$ denotes the original input to the graph, σ can be any kind of nonlinear activation function.

3.2 User Subhypergraph

The first stage in creating a user-centric network is to build a user social hypergraph network. The goal of creating a user social hypergraph is to get a better depiction of user attributes using the hypergraph structure. The user sub hypergraph uses the user as a node in the hypergraph, and the most significant content in creating the user hypergraph is determining the hyperedge. There are numerous methods for dividing the hypergraph edges.

Some researchers have advocated using meta-path to build graph representation models, learning the latent feature representation of nodes, and others extract meta-path using random walk methods, such as metapath2vec. However, the goal of these methods is to extract meta-paths, and the development of meta-paths is mostly dependent on expert views. At the same time, in the case of complicated hypergraph data, the length of fake settings influences meta-path truncation. Because of these arbitrary choices, the path chosen may not be the ideal path.

As a result, building hyperedges on pathways will result in an over-reliance on expert opinion for model performance, and it will be impossible to fully exploit the data features of graph structure.

Another approach is to build hyperedges based on user modality. The original input user data is separated into modalities, each modality into a hyper-edge, and the users with that modality into the nodes associated with that hyper-edge. This sort of hyper-edge partitioning clusters users with the same modality based on hyper-edges, allowing for richer modality-based user attributes, but it also relies on empirical modality data selection. Simultaneously, in the case of large-scale graph data, there may be a huge number of missing user modal data. Because of the limited data, the division of hyperedges may become unduly reliant on the arbitrary selection of experts, affecting the model's effect on feature extraction.

Knowledge-based social networks connect users through knowledge, and the connection between these users differs from that of e-commerce networks and social networks. There are apparent knowledge-authority user nodes in knowledge-based social networks. These authoritative users wield far more power over other users than ordinary users. In a knowledge-based social network, for example, if a user wishes to learn about "graph neural networks," he will believe the comments expressed by users "T. N. Kipf," "Hamilton," "Gori. M," and others more. These authoritative users are more visible and have a bigger influence in the realm of professional knowledge. Therefore, the hyperedges in the user sub-hypergraph in the knowledge social network may be split according to the user clustering, and the user clustering can be automatically clustered according to the user's knowledge influence.

Knowledge authority users have a greater influence on the users around them and attract other users to congregate around them to form denser clusters of users. At the same time, when compared to ordinary users, authoritative users will have a greater impact on users in a wider range, forming a larger user influence range than ordinary users. Based on this premise, users with greater knowledge influence can be identified first to form a central node candidate set of clustering hyperedges. To begin, N users can be chosen to form the central node candidate data based on the value of the user influence feature, which is an implicit feature derived from the following.

- The user's basic influence

The user's basic attributes in the recommendation system primarily include some explicit eigenvalues such as account number, gender, age, education level, and domain of interest. Furthermore, some systems provide users with explicit levels; for example, on Weibo, users can be ordinary users without any authentication, ordinary users with identity authentication, officially certified V users, self-publishing certified users, and so on. The influence of a user's basic attributes is determined by the completeness of the user's basic attributes and the level of trust provided by the platform. The influence of a user's basic attributes is recorded as A_u.

- The user’s traffic influence

Different users receive different levels of attention in any social network, and we define this measure of attention received by a user as user's traffic. There is a link between user influence and the amount of attention they receive in knowledge social networks. User influence is generally proportional to traffic. For example, content posted by authoritative knowledge users is always more likely to be forwarded or read, whereas knowledge shared by ordinary system users may receive less attention. Therefore, to identify potential core users in knowledge social networks, it is only necessary to calculate their comprehensive influence and sort them.

Interaction between users happens in the recommender system, and the influence of this interaction is greater than the impact of the user's own attribute values. Visits, likes, followers, comments, messages, retweets, and pulling or blocking others are all examples of social

behavior. A user's active conduct in the system might raise the user's impact on other users, hence increasing the user's influence value. Active behavior refers to behavior that the user initiates actively and does not prioritize social connection with others, such as login, posting posts, reviewing items, and so on. The social behavior of the user is directional. User A, for example, pays attention to user B, whereas user B does not always pay attention to A. Similarly, user A retweets user B's post, although B does not necessarily like A's message. When a user's influence is considered, the impact reflected in social actions initiated by the user is smaller than the influence represented in social actions received by the user. Posts by extremely important people with a large number of followers among Weibo users, for example, will always earn more likes, although such users may not regularly like other users' stuff. When calculating the user's traffic influence, we take into account the social behavior influence data brought by the user as the receiver, i.e., the in-degree social behavior in the user's social relationship network.

The following processes are used for user feature extraction based on the theoretical basis of the preceding analysis.

3.2.1 Determining User Influence

The implicit traffic influence of users must be calculated by turning their social behavior into traffic influence. The information uploaded by users, the amount of attention users receives, the citation rate, reading quantity, recommendation degree, and assessment degree of information provided by users, and so on comprise the behavior in the knowledge social network. After converting the above data, the comprehensive influence S_u of a single published content and the overall influence h_u of all published material are computed, and the user's traffic influence is obtained as Equation (7).

$$B_u = W_1 S_u + W_2 h_u \tag{7}$$

In (7), W_1 and W_2 are weighting factors. Combining these two aspects of influence, the total influence of any user u in the system is calculated by the following formula Equation (8).

$$\begin{gathered} E_u = \alpha A_u + \beta B_u \\ \alpha + \beta = 1 \end{gathered} \tag{8}$$

Among them, A_u denotes the attribute influence of user u, $B_u$denotes the traffic influence, α and β are the weighting coefficients of different influences. It is important to note that in knowledge-based social networks, the influence brought by user traffic is decisive and much higher than the base influence of users.

3.2.2 Determine the Number of Hyperedges

For a system with N users, select the top $\sqrt{N}$ user with the highest influence ranking to form the super edge center candidate set denoted as SEC-set. For the user data in the SEC-set, the ultimate goal is to construct hyper-edges by automatic clustering. Meanwhile, unlike general user clustering, which generally classifies a user into one class cluster, but in practice, a user can often belong to multiple clusters due to different feature selection. For example, in Zhihu website, a user A can belong to both the "Artificial Intelligence" interest group and the "Suspense Fiction Lovers" interest group, and the user can be associated with the knowledge authority users in different groups. Therefore, this paper makes an improvement on the traditional density clustering algorithm to select the super edges.

The density $\boldsymbol{\rho_i}$ is used to denote the density around user i. The distance between user u_i and u_j is denoted as $dist(u_i, u_j)$, and the distance between two user vectors can be measured using the Euclidean metric as Equation (9).

$$dist(u_i, u_j) = \sqrt{\sum_{k=1}^{n}(u_{ik} - u_{jk})^2} \quad (9)$$

Where K denotes the dimension of user features, and the Euclidean distance between the user and the user itself is 0. The greater the distance between users indicates the lower the similarity between users, so the greater the distance the lower the contribution of the peripheral users to the density calculation of user. In order to prevent outlier points from interfering with the density data when calculating the distance, the calculated user distance should be normalized and calculated, and we use the Z-score normalization method to normalize the data of the distance, and the calculation formula is as follows Equation (10).

$$dist(u_i, u_j)^* = (dist(u_i, u_j) - \mu)/\sigma \quad (10)$$

Where μ denotes the mean of the user sample and σ denotes the standard deviation of the user sample data. The density of user i is calculated by the formula as Equation (11).

$$\rho_i = \sum_{u_i \in U} D((\sqrt{\sum_{k=1}^{n}(u_{ik} - u_{jk})^2} - \mu)/\sigma - d_c) \quad (11)$$

Where the density contribution of user i to itself is 1 and the density contribution of the remaining users to user i is represented by the Gaussian function as Equation (12).

$$D(i,j) = \begin{cases} 1 & if\ i = j \\ \exp\left(-(\frac{\sqrt{\sum_{k=1}^{n}(u_{ik} - u_{jk})^2} - \mu}{\sigma} - d_c)^2\right) & if\ i \neq j \end{cases} \quad (12)$$

Where d_c is the node distance cut-off value, which can also be viewed as a super parameter, i.e., it is considered that user j beyond the d_c distance does not have an impact on the density of the currently calculated user i.

Rodriguez et al. [24] pointed out that when the size of the data set (i.e., the number of samples included) is big, the density clustering algorithm's clustering results are less affected by the truncation distance, and vice versa. In the complex hypergraph structure composed of users and knowledge, which commonly contains a large amount of data, the truncation distance should not be too small, because the purpose of the density calculation is to select the most representative hyperedges and divide users into hyperedges, otherwise a large amount of data valuable to the density of users in the calculation center will be lost. The amount of data in the model described in this work has been reduced exponentially once the super edge candidate cluster center set is produced by estimating the user influence. As a result, the truncation distance used in this research is the intermediate value of the distance between the cluster center candidate samples set's candidate cluster centers. The formula for calculation is as Equation (13).

$$d_c = \frac{1}{2}(MAX(dist(u_i, u_j)) + MIN(dist(u_i, u_j))) \quad (13)$$

The user data in the hyper-edge center candidate set SEC-set are sorted in order to calculate the user density, and the data with the TOP-N density are selected according to the ranking size to form the initial center value data set for the next hyper-edge clustering, which is denoted as EC-set. The obtained data of TOP-N is the number of super edges. In practical applications, the number of TOP-N can be selected according to the actual situation, so as to decide the size of the number of super edges.

3.2.3 Creating Hyperedges for User Subgraphs

The next process is to classify the system's users into the corresponding hyper-edges. In contrast to traditional clustering, where a user may only belong to one cluster, the user nodes in the hyperedge can belong to numerous hyperedges. As a result, there is node overlap between hyperedges in the split of user nodes linked with hyperedges, implying that a user can

be divided into multiple different hyperedges. This corresponds to the affiliation in the real-world knowledge social network. A user in the 'knowledge social network may be interested in a variety of topics. For example, user A may be interested in machine learning knowledge and pay attention to authoritative people in the field, and it may be subordinate to the user clustering hyperedge in the field of machine learning or the user clustering hyperedge in the field of network security if it has an interest in network security expertise. As the initial hyperedge cluster centers, the N elements in the hyperedge initial center value data set EC-set are employed. One by one, the distance from all user data u to the hyperedge cluster center C_i is determined as follow Equation (14).

$$d(u, C_i) = \sqrt{\sum_{j=1}^{n}(u_j - C_{ij})^2} \tag{14}$$

We must divide a super boundary coefficient for the calculated distance. Because a user can belong to multiple hyperedges, unlike the traditional clustering algorithm, which assigns a user to the nearest cluster, we can assign a user to multiple hyperedges within the set hyperedge boundary by setting the hyperedge boundary threshold. We can sort the calculated distances in descending order and choose the TOP-EK among them to classify a user into K different hypergraphs. The hyperedge boundary threshold divides users into different hyperedges, forming the hyperedge construction of the user sub-hypergraph. So far, we have obtained the user nodes and hyperedges of a user sub-hypergraph and built a complete user sub-hypergraph.

3.2.4 Convolution of User Subhypergraphs

For the user sub-hypergraph H_U, the hypergraph convolution method is used to extract user features. $\mathrm{h}_u(v_u, e_u)$ means that the user vertex v_u is contained by the hyperedge e_u, and has the following definition as Equation (15).

$$\mathrm{h}_u(v_u, e_u) = \begin{cases} 1 & if\ v_u \in e_u \\ 0 & if\ v_u \notin e_u \end{cases} \tag{15}$$

Finally, D_{v_u} and D_{e_u} are used to denote the degree matrices of the hyper edge and the user as Equation (16).

$$\begin{aligned} D_{v_u} &\in \mathbb{N}^{|\mathcal{V}|\times|\mathcal{V}|} \\ D_{e_u} &\in \mathbb{N}^{|\mathcal{E}|\times|\mathcal{E}|} \end{aligned} \tag{16}$$

For the graph structure in the hypergraph, we can define the following Equation (17) convolution method on the graph.

$$E_{hu}{}^{(l+1)} = LeakyReLU((\mathrm{D}_{v_u}^{-1/2} H_U \mathrm{W} \mathrm{D}_{e_u}^{-1/2} \mathrm{H}^T \mathrm{D}_{v_u}^{-1/2} E_{hu}{}^{(l)} \Theta^{(l)}) \tag{17}$$

Where $E_{hu}{}^{(l)} \in \mathbb{R}^{N\times C}$ indicates that there are n nodes in layer l and the node feature dimension is C. $E_{hu}{}^{(l+1)}$ indicates the embedding expression of the user in the user subhypergraph at layer $l+1$, and when $l=0$ indicates the original input of the user subhypergraph.

3.3 Knowledge Subhypergraph

With the development of artificial intelligence technology, knowledge graph has been widely used in intelligent search, personalized recommendation and other fields. Knowledge graph usually stores entities and their relationships in the form of triples. It models knowledge in the real world into the form of (k_h, r, k_t) triples, where k_h and k_t represent head entities and tail entities respectively, and r represents the relationship between entities. Triples can not only help us understand the relationship between knowledge entities, but also store the

attributes of knowledge entities [25]. The rich semantic associations between items in the knowledge graph can be used to explore the potential connections between them and improve the accuracy of recommendation results. The introduction of knowledge graph into the recommendation system not only facilitates the information mining and recommendation result dispersion, but also enhances the interpretability of recommendations [26].

Knowledge graph representation learning should consider not only the structural features of graphs, but also the semantic type information of nodes and edges. Although knowledge graph models such as TransE and DIstmult can also capture the structural information of graphs, graph neural networks are more adequate for considering the signals of graph structural features. Therefore, we can use graph neural networks in knowledge graph representation learning algorithms to capture the structural information in the graphs [27].

A knowledge hypergraph is an extension of the knowledge graph structure, and the construction of a knowledge hypergraph focuses on determining the relevant classes that can be classified as hyper-edges and determining the weight of each hyper-edge. Firstly, we construct a knowledge hypergraph H_K, which contains knowledge nodes and hyperedges. $\mathrm{h}_k(v_k, e_k)$ denotes that the knowledge vertex v_k is contained by the hyperedge e_k , and has the following definition as Equation (18).

$$\mathrm{h}_k(v_k, e_k) = \begin{cases} 1 & if\ v_k \in e_k \\ 0 & if\ v_k \notin e_k \end{cases} \tag{18}$$

The construction and feature extraction of knowledge sub-hypergraph can adopt the following steps.

3.3.1 Hyperedge Construction and Representation of Knowledge Nodes

To begin, the relation of knowledge graph can be employed to extract the hyperedges in the knowledge hypergraph. In a traditional knowledge graph, a head node might connect numerous tail nodes based on various relationships. Similarly, the same relationship can link various head and tail nodes. This complicated relationship coverage and multi-point linkage are typical of hypergraphs. The relation r in the knowledge triple (k_h, r, k_t) in the knowledge graph can be utilized to extract these relations, which form the knowledge hypergraph's hyperedge denoted as e_k. It should be noticed that the nodes contained in the hyperedge do not directly incorporate the head and tail nodes in the knowledge graph's triple form. The node selection and representation methods contained in the specific hyperedge are shown in the next step.

The most crucial step after constructing the hyperedge is to choose the nodes that will be covered by it. Although the knowledge nodes k_h and k_t are already present in the data linked with the connection r in the triple (k_h, r, k_t), this relationship is directional. The information difference between the head node k_h and the tail node k_t is significant. If all of the associated nodes of the relationship r are included in the range of nodes covered by the super edge, a high number of noise features are introduced, affecting knowledge feature extraction.

To overcome this problem, we use the homogeneous graph convolution approach. Extract all the triples in the knowledge graph with k as the head node and construct a homogeneous first-order graph structure with k as the central node and all the tail nodes as the first-order neighbors for the knowledge node k that wishes to retrieve the feature expression. The information of its adjacent nodes is then aggregated to the center node via a layer of graph convolution. The aggregation calculation formula of the first-order neighbor is as Equation (19).

$$e_k^{(1)} = LeakyRelu(W_1 e_k^{(0)} + \sum_{i \in N_k} \frac{1}{\sqrt{|N_k||N_i|}} (W_1 e_i^{(0)} + W_2 (e_i^{(0)} \odot e_k^{(0)}))) \tag{19}$$

Where N_k represents the set of all neighbor nodes of node K, W_1 and W_2 is the weight matrix. $e_k^{(1)}$ is the updated expression formed by attaching the tail node information in the knowledge graph triple to the head node, denoted as v_k.

3.3.2 Knowledge Subhypergraph Convolution

For the knowledge subhypergraph H_K, knowledge vertex v_k and knowledge hyperedge e_k constructed above, the degree of knowledge vertex v_k is defined as Equation (20).

$$d(v_k) = \sum_{e_k \in \mathcal{E}} \omega(e_k) \mathrm{h}_k(v_k, e_k) \tag{20}$$

Where e_k is the hyperedge in the set of hyperedges in the knowledge hypergraph, $\omega(e_k)$ represents the weight of the hyperedge in the hyperedge. In the hypergraph, the weight of the hyperedge can reflect the internal relevance of the knowledge vertices associated with the hyperedge. Similarly, the degree of the hyperedge can be defined as Equation (21).

$$\delta(e_k) = \sum_{v_k \in \mathcal{V}} \omega(v_k) \mathrm{h}_k(v_k, e_k) \tag{21}$$

Finally, D_{v_k} and D_{e_k} are used to represent the degree matrix of hyperedge and knowledge vertex as follows Equation (22).

$$\begin{aligned} D_{v_k} &\in \mathbb{N}^{|\mathcal{V}| \times |\mathcal{V}|} \\ D_{e_k} &\in \mathbb{N}^{|\mathcal{E}| \times |\mathcal{E}|} \end{aligned} \tag{22}$$

For the knowledge sub-hypergraph structure in the hypergraph, we can also define the convolution method as Equation (23) on the graph.

$$E_{hk}{}^{(l+1)} = LeakyReLU(\mathrm{D}_{v_u}^{-1/2} H_K \mathrm{W} \mathrm{D}_{e_k}^{-1/2} \mathrm{H}^T \mathrm{D}_{v_k}^{-1/2} E_{hk}{}^{(l)} \Theta^{(l)}) \tag{23}$$

Where $E_{hk}{}^{(l)} \in \mathbb{R}^{N \times C}$ denotes n nodes in layer l and node feature dimension is C. $E_{hk}{}^{(l+1)}$ denotes the embedding expression of knowledge in the knowledge subhypergraph at layer $l+1$, and when $l = 0$ denotes the original input of the knowledge subhypergraph.

3.4 Knowledge recommendation

For the user embedding expression E_{hu} obtained in Section 3.2 and the knowledge embedding expression E_{hk} obtained in Section 3.3, the following Equation (24) are used for prediction and recommendation.

$$\hat{\mathcal{Y}}_{uk} = (E_{hu})^T E_{hk} \tag{24}$$

The loss function of the model in the parameter solution uses the BPR loss function, which is defined by the Equation (25).

$$\mathrm{Loss} = \sum_{(u,k_i,k_j) \in D_s} \ln \sigma \left(\hat{y}_{uk_i} - \hat{y}_{uk_j} \right) - \lambda_\Theta \|\Theta\|_2^2 \tag{25}$$

D_s is defined as Equation (26).

$$D_s = \left\{ (u, k_i, k_j) \middle| (u, k_i) \in \mathcal{R}^+, (u, k_j) \in \mathcal{R}^- \right\} \tag{26}$$

Among them, $\mathcal{R}^+$ denotes the data where user u evaluates knowledge k_i higher than knowledge k_j and $\mathcal{R}^-$ denotes the data where user u evaluates knowledge k_i lower than knowledge k_j. $\hat{y}_{uk}$ denotes the user's prediction score of knowledge, λ_Θ denotes the parameter that controls the strength of L_2 regularization to prevent overfitting, and Θ denotes all trainable parameters in the model.

After the above steps, the prediction evaluation of user u on knowledge k can be obtained, and the list of knowledge recommendations for user u can be generated by selecting the top N according to the actual recommendation requirements.

4. Experiment

4.1 Baseline

The following algorithm was selected as the comparison baseline in the experiments.

KGCN [22] is a propagation-based model which extends non-spectral GCN approaches to the knowledge graph by aggregating neighborhood information selectively and biasedly, which is able to learn both structure information and semantic information of the KG as well as users personalized and potential interests.

KGNN-LS [28] is a propagation-based model which transforms heterogeneous KG into a user-specific weighted graph and computes personalized item embedding in graph neural network with label smoothness regularization.

KGAT [23] is also a propagation-based model which combines the UIG and KG together as a homogeneous unified graph called CKG. Compared with KGCN, KGAT utilizes an attention mechanism to discriminate the importance of neighbors in CKG during propagation.

CKAN [29] employs a heterogeneous propagation strategy to explicitly encode both kinds of information, and applies a knowledge-aware attention mechanism to discriminate the contribution of different knowledge-based neighbors.

CKE [30] is a model designed with three components to extract items' semantic representations from structural content, textual content and visual content. It adopt a heterogeneous network embedding method, termed as TransR, to extract items' structural representations by considering the heterogeneity of both nodes and relationships.

The datasets and evaluation metrics used in comparative baseline model are shown in the **Table 2**.

Table 2. Summary of baseline models

Model	Dataset	Evaluation metrics
KGCN	MovieLens-20M, Book-Crossing, Last.FM	AUC, F1
KGNN-LS	MovieLens-20M, Book-Crossing, Last.FM, Dianping-Food	Recall, AUC
KGAT	Amazon-book, Last-FM, Yelp2018	Recall, NDCG
CKAN	Last.FM, Book-Crossing, MovieLens-20M, Dianping-Food	AUC, F1
CKE	MovieLens-1M, IntentBooks	Recall, MAP

4.2 Dataset

In recommendation systems, commonly used experimental datasets include Yelp, MovieLens, Douban, LastFM, Book Crossing, etc. Because the model we proposed uses hypergraph neural network and Knowledge graph to recommend knowledge, the experiment should give priority to the data set that is convenient to use Knowledge graph to embed and express the recommended content, so the Last.FM and Book Crossing data sets are selected as the data basis for experimental comparison. Last.FM is a dataset commonly used in recommendation systems, mainly providing music recommendations. For each user in the dataset, the dataset contains a list of their most popular artists and the number of plays. It also includes user application tags that can be used to build content vectors, and the dataset also contains information about the user's social networks. Book-Crossing is collected from the book-crossing community, which consists of trenchant ratings (ranging from 0 to 10) from different readers about various books.

Data set information is shown in **Table 3**.

Table 3. Data set

	Last.FM	Book.Crossing
Users	1,872	17,860
Items	2,445	14,967
Interaction	753,772	139,746
Entities	182,011	77,903
Relations	60	25
KG Triples	15,518	151,500

4.3 Experiments and Discussions

In this section, we aim to answer the following research questions:
RQ1: in hypergraph convolutional neural networks, the relationship between the number of convolutional layers and model performance.
RQ2: whether the size of the number of hyper edges in the hypergraph structure affects the model performance.
RQ3: whether the two-channel convolution we proposed is more effective than the single-channel convolution.

4.3.1 Impact of Convolution Layers

For hypergraph neural networks, there is a relationship between the number of layers of the graph convolution and the model performance. Existing studies have shown that one of the drawbacks of graph neural networks is that as the number of convolution layers deepens, the model produces over smoothing [31]. The hypergraph neural network is an extension of the graph neural network. For the model proposed in this paper, in order to obtain better performance and to investigate the relationship between the number of hypergraph convolutional layers and performance, we study the performance influence of the number of hypergraph convolution layers through experiments. The experimental results of different convolution layers on different data sets are shown in **Fig. 2**.

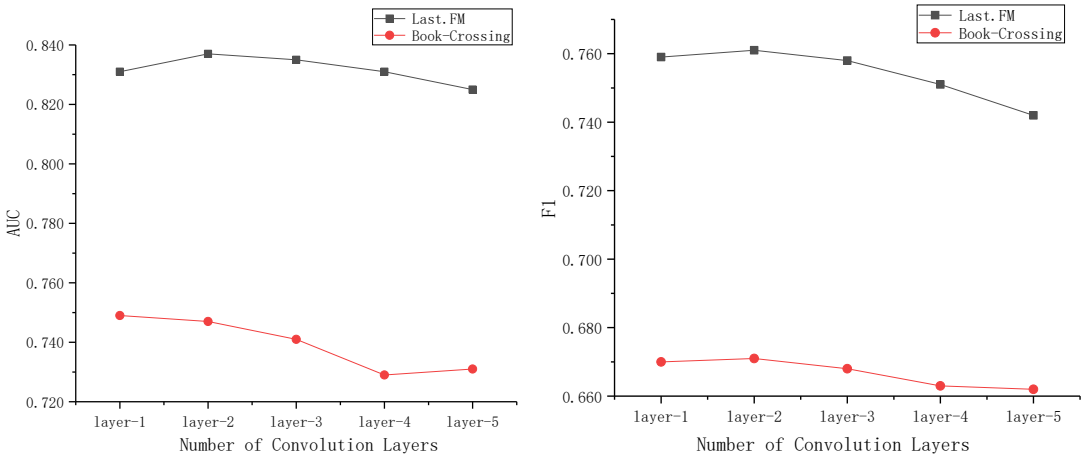


Fig. 2. The impact of different convolution layers on different datasets

Through the experiments we can see that the model performance decreases with the increase in the number of hypergraph convolutional layers on both datasets, indicating that the extraction of various feature information encounters obstacles instead with the increase in the

number of convolutional layers. This phenomenon has also been mentioned in papers related to graph neural networks, indicating that not only the deepening of layers in graph convolutional neural networks impairs the model performance, but the same property also exists in hypergraph convolutional neural networks. Through experimental comparison, and considering the computational overhead of many nodes in hypergraph convolution, we set the convolution of hypergraph neural network to 2 layers.

4.3.2 Impact of Embedding Dimension

The performance of the model depends in part on the model's capacity to represent data features, which is affected by the size of the spatial dimension of the vector. Different dimensional embedding vectors use different precise methods to describe the characteristics of the data. We typically assume that the more spatial dimensions there are, the more data characteristics can be defined, and the more knowledge and user data features there are, the more positive an impact they may have on the recommendation. The performance metrics for the two datasets that we used in our experiments, where we set the vector dimension h to 32, 64, 128, 256, 512, and 1024, respectively, are as follows **Fig. 3**.

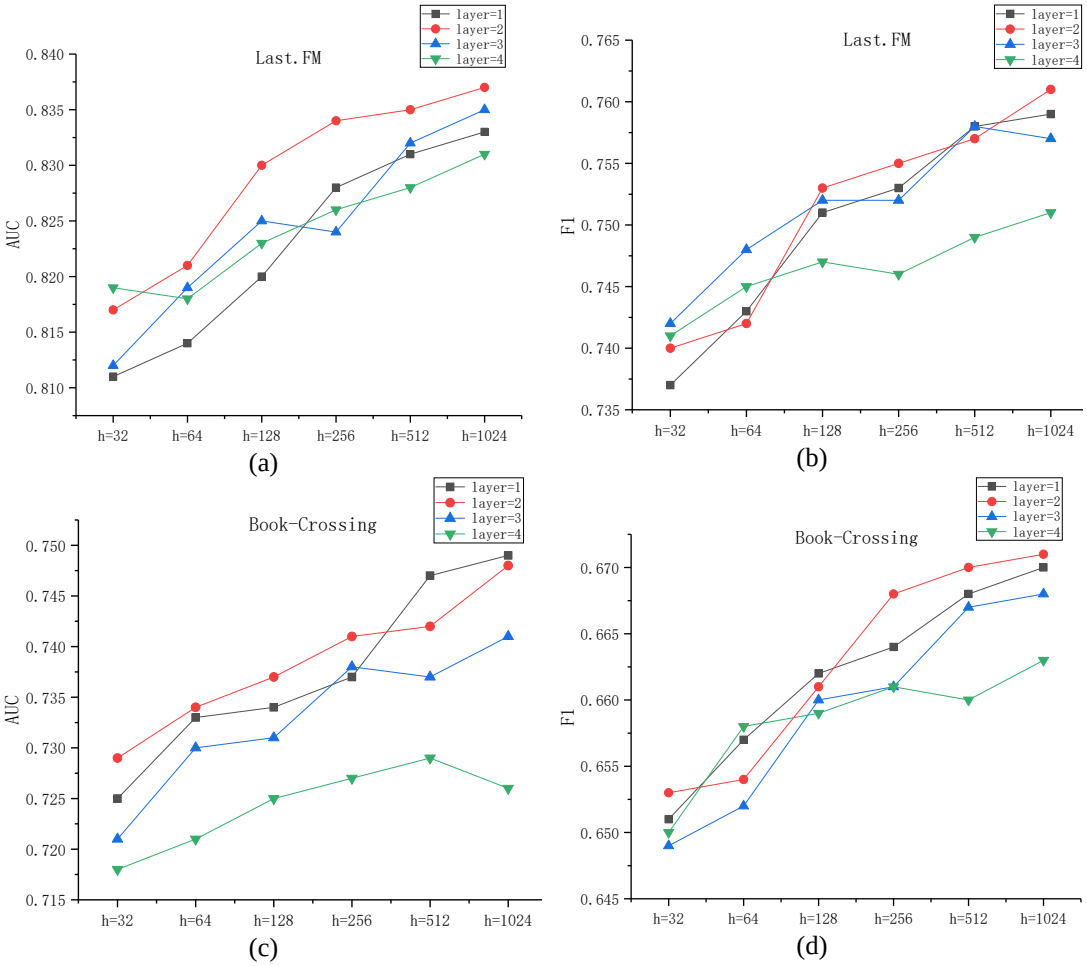


Fig. 3. The impact of embedding dimension on different datasets

The experimental results show that the algorithm's performance on both datasets dramatically increases as h rises, proving that the performance of the model is affected by the size of the vector space dimension. The performance improvement is most obvious in the interval from 32 to 256 dimensions. But we also see that when h increases, the pace of this curve's rise reduces. When the vector space dimension h is increased from 512 to 1024, the F1 metric even exhibits a decline in performance under three-layer hypergraph convolution.

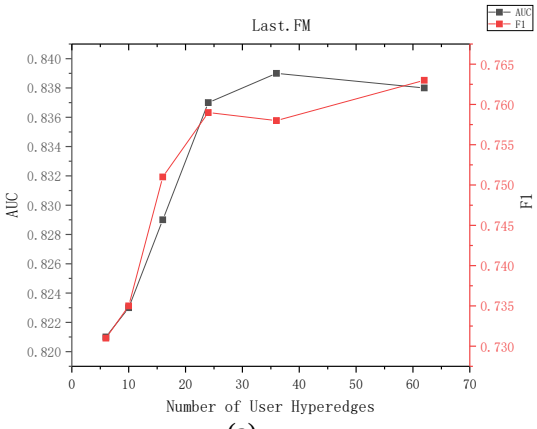
On the book-crossing dataset, the AUC metric also showed a decrease when the vector space dimension h was extended from 512 to 1024 in the 4-layer hypergraph convolution. This shows that although the model's performance is h-dependent, it does not always get better as h increases. When h reaches a particular level, the model's performance not only stops improving after a given point, but it may also require extra training time due to the high value of h. Therefore, the optimal size of h is set to 256 or 512 in this paper.

4.3.3 Effect of the Hyperedges Number

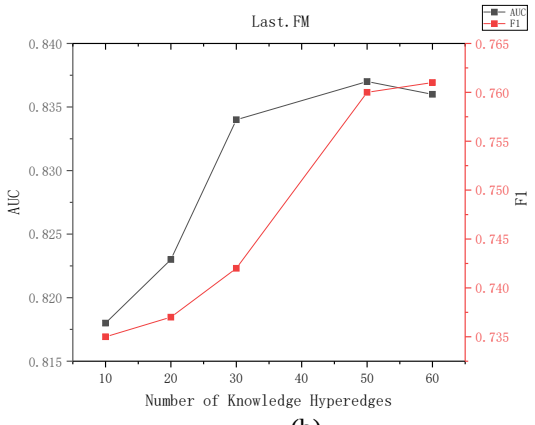
The quantity of hyperedges in the hypergraph structure has an impact on both the hypergraph's shape and construction. The quantity of hyperedges in the hypergraph structure has an impact on both the hypergraph's shape and construction. According to the proposed model, choosing the number of TOP-N in the user density calculation is largely responsible for the construction of hyperedges in the user subhypergraph. The value of N determines the number of hyperedges, which in turn affects the distribution of user nodes and, in turn, the creation of the entire hypergraph.

Similar to this, the number of relationships in the knowledge graph influences the number of relationships in the knowledge subhypergraph, which in turn dictates the number of hyperedges and the morphological structure of the knowledge subhypergraph. We define the number of users hyperedges in the user subhypergraph as NEU, and the number of hyperedges in the knowledge subhypergraph as NEK, in order to study the relationship between the choice of the number of hyperedges and the effectiveness of knowledge suggestion.

We define the number of users hyperedges in the user subhypergraph as NEU, and the number of hyperedges in the knowledge subhypergraph as NEK, in order to study the link between the choice of the number of hyperedges and the effectiveness of knowledge suggestion. For the two datasets, the NEU is set to 6, 10, 16, 24, and 36, respectively. For the last.fm dataset, NEK is set to 10, 20, 30, 50, and 60, whereas NEK is set to 5, 10, 15, and 25 for the Book.Crossing dataset. The experimental results are shown in **Fig. 4**.



(a)



(b)

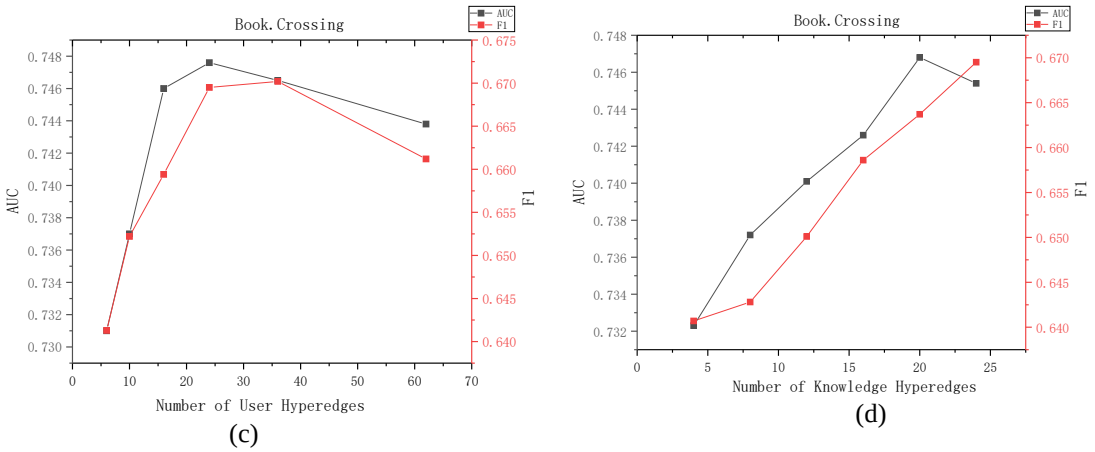


Fig. 4. The impact of the hyperedges number on different datasets

According to the experimental data, performance on the Last.FM dataset grows as the number of user super-edges increases, and the model's performance similarly rises when the number of knowledge super-edges increases. However, it's important to keep in mind that as the number of hyper-edges rises, this rising trend begins to reduce. This implies that while adding more hyper-edges might improve recommendation performance while the number of original hyper-edges is low, the effect does not continue to grow indefinitely. It can be seen that when the number of hyper-edges rises in the Book.Crossing dataset, the model's performance first improves and then degrades. The model's performance increases, declines, reaches its peak when there are 24 user hyperedges, and then declines again. The number of knowledge super-edges has a similar performance inflection point as it rises.

The outcomes of this experiment demonstrate that in the initial scenario, increasing the number of user hyper-edges or the number of knowledge hyper-edges may successfully extract the features of users and the features of knowledge through the hyper-edge structure.

However, as the number of hyper-edges rises, there will be fewer nodes embedded in each hyper-edge and a greater division of a node belonging to many hyper-edges, all of which will add undesirable data to the calculation of the model and hence degrade its performance. In order to provide knowledge recommendations based on hypergraph structure, knowledge hyper-edges and user hyper-edges must both select the right number.

4.3.4 Effect of Channel Remove

In order to verify whether this dual-channel convolutional model design is more valuable than single-channel, we compare the full model DCHC proposed in this paper with a simplification. Firstly, the complete model is simplified into a model with user subhypergraph channels and pure knowledge feature expression subgraphs, and the model is simplified as DCHC-UH. The knowledge hypergraph sub-channel design based on knowledge triple representation and the model that only retains user embedding are simplified as DCHC-KH. The experimental results of the complete model and the simplified model on the two datasets are shown in **Fig. 5**.

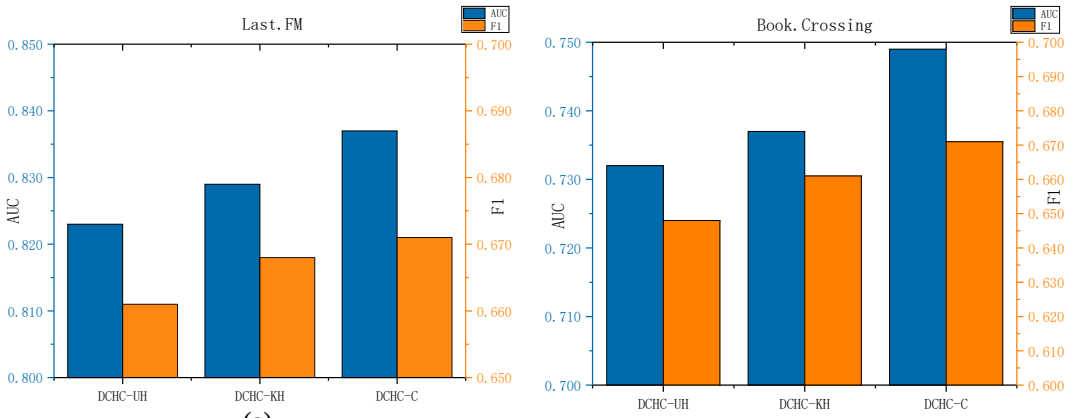


Fig. 5. The effect of channel remove on different datasets

The experimental results show that the performance of the simplified model with the user subhypergraph removed and the simplified model with the knowledge subhypergraph removed decreases on both datasets, which indicates that the design of feature extraction of users and knowledge through the user subhypergraph and knowledge subhypergraph is effective.

4.3.5 Comparison with the Baseline

As can be seen by comparing the data in the following **Fig. 6**, when comparing the DCHC model proposed in this paper with the baseline model, the model proposed in this paper achieves better results on both experimental datasets, indicating that the model proposed in this paper is effective in enhancing the recommendation performance.

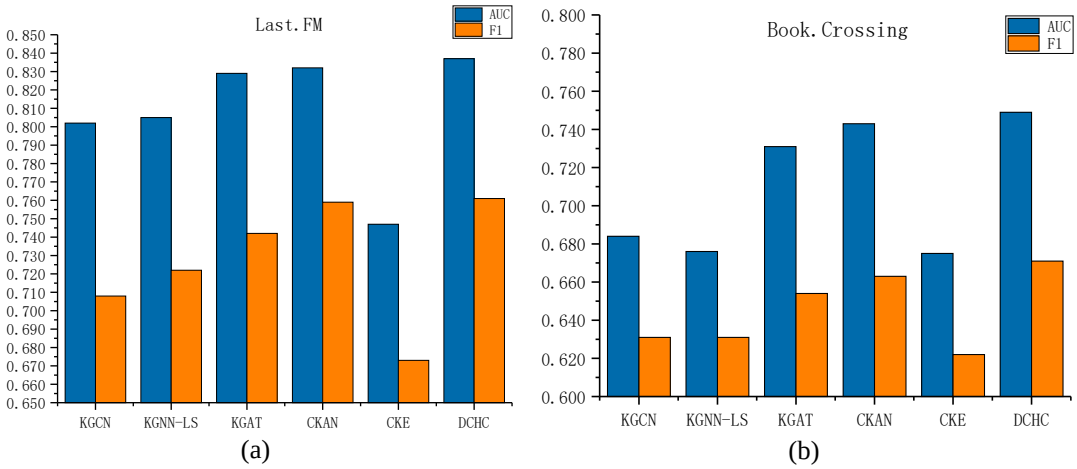


Fig. 6. Comparison with the baseline on different datasets

The experiments in 4.3.1 demonstrate that the graph convolutional layers in hypergraph neural networks do not need to be over stacked. The experiment in 4.3.2 proves that it is not necessary to use oversized space for embedding representation, and that too high spatial dimensions do not improve the model performance infinitely. The experiment in 4.3.3 proves that increasing the number of hyper edges helps to improve the model performance, but the improvement will not go up indefinitely, so we need to choose an optimal value of the number of hyper edges in practical applications. Through the channel masking method of dual-channel in 4.3.4, it is

proved that the performance of dual-channel convolution is higher than that of single-channel, and it also proves the effectiveness of our proposed dual-channel approach.

5. Conclusion

Using graph neural networks to tackle recommendation problems is a common study in the field of deep learning, and when compared to graph structures, hypergraph structures can better reflect real-world data relationships. By decomposing the traditional user-project bipartite graph into user sub-hypergraphs and knowledge sub-hypergraphs, and replacing graph convolution with hypergraph convolution, higher quality user embedding and knowledge embedding expressions are obtained, thereby improving the performance of knowledge recommendation. In comparison with the baseline model, the model we proposed also achieved optimal performance under both AUC and F1 metrics, which also proves that the dual channel hypergraph convolution model is effective for achieving better knowledge recommendation.

We will also continue to investigate methods of hyper-edge partitioning based on users' multimodal behaviors and constructing hyper-edges based on knowledge density clustering in the following work, and investigate whether these different methods of hyper-edge construction can produce more effective recommendations.

Acknowledgement

This work was supported by Department of Education of Guangdong Province under Grant No. GDGJ2021111 and the Chinese Society for Technical and Vocational Education under Grant No. ZJ2022B23.

References

[1] M. Mameli, M. Paolanti, R. Pietrini, G. Pazzaglia, E. Frontoni et al., “Deep learning approaches for fashion knowledge extraction from social media: A review,” *IEEE Access*, vol. 10, pp. 1545–1576, Dec. 2021. Article (CrossRef Link)

[2] Q. Guo, F. Zhuang, C. Qin and H. Zhu, “A survey on knowledge graph-based recommender systems,” *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 8, pp. 3549-3568, 2022. Article (CrossRef Link)

[3] Z. Batmaz, A. Yurekli, A. Bilge and C. Kaleli, “A review on deep learning for recommender systems: challenges and remedies,” *Artificial Intelligence Review*, vol. 52, no. 1, pp.1–37, Aug. 2018. Article (CrossRef Link)

[4] Stefanie. J, “Theory of Graph Neural Networks: Representation and Learning,” *arXiv preprint arXiv: 2204.07697v1*, Apr. 2022. Article (CrossRef Link)

[5] J. Liu and L. Duan, “A survey on knowledge graph-based recommender systems," in *Proc. of IEEE 5th Adv. Inf. Technol., Electron. Automat. Control Conf. (IAEAC)*, vol. 5, pp. 2450-2453, Apr. 2021. Article (CrossRef Link)

[6] Zhang. L, Guo. J, Wang. J, Li. S and Zhang. C, “Hypergraph and Uncertain Hypergraph Representation Learning Theory and Methods,” *Mathematics*, vol. 10, no.11, 1921, Jun. 2022. Article (CrossRef Link)

[7] Burke. R, “Hybrid Recommender Systems: Survey and Experiments,” *User Model User-Adap Inter*, vol. 12, pp. 331–370, Nov. 2002. Article (CrossRef Link)

[8] Chicaiza. J and Priscila.V, “A Comprehensive Survey of Knowledge Graph-Based Recommender Systems: Technologies, Development, and Contributions,” *Information*, vol. 12, no. 6, 232, May. 2021. Article (CrossRef Link)
[9] S. Wu, F. Sun, W. Zhang, X. Xie and B. Cui, “Graph neural networks in recommender systems: a survey,” A*CM Computing Surveys*, vol.55, No.97, pp. 1–37, Dec. 2022. Article (CrossRef Link)
[10] Z. Zhang, P. Cui and W. Zhu, "Deep Learning on Graphs: A Survey," *IEEE Transactions on Knowledge and Data Engineering*, vol. 34, no. 1, pp. 249-270, 2022. Article (CrossRef Link)
[11] C. Berge, *The theory of graphs and its applications*, Methuen & Co. Ltd., London, John Wiley & Sons Inc., New York, 1962.
[12] J. Zhang, F. Li, X. Xiao, T. Xu, Y. Rong et al., “Hypergraph Convolutional Networks via Equivalency between Hypergraphs and Undirected Graphs,” *arXiv preprint arXiv: 2203.16939*, Jul. 2022. Article (CrossRef Link)
[13] S. Ji, Y. Feng, R. Ji, X. Zhao, W. Tang and Y. Gao, “Dual Channel Hypergraph Collaborative Filtering,” in *Proc. of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*, pp. 2020–2029, Aug. 2020. Article (CrossRef Link)
[14] B. Liu, P. Zhao, F. Zhuang, X. Xian, Y. Liu and V.S. Sheng, “Knowledge-Aware Hypergraph Neural Network for Recommender Systems,” in *Proc. of DASFAA 2021: Database Systems for Advanced Applications*, pp 132–147, Apr. 2021. Article (CrossRef Link)
[15] J. Yu, H. Yin, J. Li, Q. Wang, N. Q. V. Hung and X. Zhang, “Self-Supervised Multi-Channel Hypergraph Convolutional Network for Social Recommendation,” in *Proc. of the Web Conference 2021 (WWW '21). Association for Computing Machinery*, pp. 413–424, Apr. 2021. Article (CrossRef Link)
[16] D. Yang, X. Xia, L. Ye, L. Xue and Y. Xiao, “Knowledge-enhanced Recommender Systems: A Survey and Prospect,” *Journal of Cyber Security*, vol. 6, no. 5, pp.35-51, Oct. 2021. Article (CrossRef Link)
[17] Bordes. A, Usunier. N, Garcia-Duran. A, Weston. J and Yakhnenko. O, “Translating embeddings for modeling multi-relational data,” in *Proc. of Neural Information Processing Systems Conference*, pp. 2787–2795, Dec. 2013. Article (CrossRef Link)
[18] G. Ji, S. He, L. Xu, K. Liu and J. Zhao, “Knowledge graph embedding via dynamic mapping matrix,” in *Proc. of the 53rd annual meeting of the association for computational linguistics and the 7th international joint conference on natural language processing*, vol. 1, pp. 687-696, Beijing, China, Jul. 2015. Article (CrossRef Link)
[19] Z. Wang, J. Zhang, J. Feng and Z. Chen, “Knowledge Graph Embedding by Translating on Hyperplanes,” in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 28, no. 1, Jun. 2014. Article (CrossRef Link)
[20] F. Che, D. Zhang, J. Tao, M. Niu and B. Zhao, “Parame: Regarding neural network parameters as relation embeddings for knowledge graph completion,” in *Proc. of the AAAI Conference on Artificial Intelligence*, vol. 34, no. 03, pp. 2774-2781, Apr. 2020. Article (CrossRef Link)
[21] H. Wang, F. Zhang, J. Wang, M. Zhao, W. Li et al., “RippleNet: Propagating user preferences on the knowledge graph for recommender systems,” in *Proc. of the 27th ACM international conference on information and knowledge management*, Torino, Italy, pp. 417-426, Oct. 2018. Article (CrossRef Link)
[22] H. Wang, M. Zhao, X. Xie, W. Li and M. Guo, “Knowledge graph convolutional networks for recommender systems,” in *Proc. of The world wide web conference*, San Francisco, CA USA, pp. 3307-3313, May. 2019. Article (CrossRef Link)
[23] X. Wang, X. He, Y. Cao, M. Liu and T. S. Chua, “KGAT: Knowledge graph attention network for recommendation,” in *Proc. of the 25th ACM SIGKDD international conference on knowledge discovery & data mining*, pp. 950-958, Jul. 2019. Article (CrossRef Link)
[24] A. Rodriguez and A. Laio, “Clustering by fast search and find of density peaks,” *Science*, vol. 344, pp. 1492-1496, Jun. Jun. 2014. Article (CrossRef Link)
[25] I. Tiddi and S. Schlobach, “Knowledge graphs as tools for explainable machine learning: A survey,” *Artificial Intelligence*, vol. 302, 103627, Jan. 2022. Article (CrossRef Link)

[26] D. Lu, D. Zhu, H. Du, Y. Sun, Y. Wang et al., "Fusion recommendation system based on collaborative filtering and knowledge graph," *Computer Systems Science and Engineering*, vol. 42, no.3, pp. 1133–1146, 2022. Article (CrossRef Link)
[27] X. Liu, H. Tan, Q. Chen and Lin, G., "RAGAT: Relation Aware Graph Attention Network for Knowledge Graph Completion," *IEEE Access*, vol. 9, pp. 20840–20849, Jan. 2021. Article (CrossRef Link)
[28] H. Wang, J. Leskovec, F. Zhang, M. Zhao, W. Li et al., "Knowledge-aware Graph Neural Networks with Label Smoothness Regularization for Recommender Systems," in *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, Anchorage, AK, USA, pp. 968–977, Jul. 2019. Article (CrossRef Link)
[29] Z. Wang, G. Lin, H. Tan, Q. Chen and X. Liu, "CKAN: Collaborative Knowledge-aware Attentive Network for Recommender Systems," in *Proc. of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, Virtual Event, China, pp. 219-228, Jul. 2020. Article (CrossRef Link)
[30] F. Zhang, N. J. Yuan, D. Lian, X. Xie and W. Y. Ma, "Collaborative knowledge base embedding for recommender systems," in *Proc. of the ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, California, USA, pp.353-362, Aug. 2016. Article (CrossRef Link)
[31] Bingbing. X, Keting. C, Junjie. H, Huawei. S and Xueqi. C, "A survey on graph convolutional neural network," *Chinese Journal of Computers*, vol. 43, no. 5, pp. 755-780, May. 2020. Article (CrossRef Link)

Li Yue received her bachelor's degree in computer science from Wuhan University, China (2002). She Received her master's degree in software engineering from Sun Yat-sen University in 2005, China. She is working in Guangdong Communication Polytechnic, School of Information. Her research direction is recommendation system, graph neural network, etc.