PLOS ONE

RESEARCH ARTICLE

MIRN: A multi-interest retrieval network with sequence-to-interest EM routing

Xiliang Zhang[1], **Jin Liu**[1]*, **Siwei Chang**[1], **Peizhu Gong**[1], **Zhongdai Wu**[2,3], **Bing Han**[2]

1 College of Information Engineering, Shanghai Maritime University, Shanghai, China, **2** Shanghai Ship and Shipping Research Institute, Shanghai, China, **3** COSCO Shipping Technology Co., LTD, Shanghai, China

* jinliu@shmtu.edu.cn

Citation: Zhang X, Liu J, Chang S, Gong P, Wu Z, Han B (2023) MIRN: A multi-interest retrieval network with sequence-to-interest EM routing. PLoS ONE 18(2): e0281275. https://doi.org/10.1371/journal.pone.0281275

Editor: Xiangjie Kong, Zhejiang University of Technology, CHINA

Received: November 8, 2022

Accepted: January 19, 2023

Published: February 2, 2023

Data Availability Statement: All relevant data are within the paper.

Funding: This work was supported by the National Key Technologies Research and Development Program of China under Grant 2021YFC2801001, in part by the National Social Science Foundation of China under Grant 20&ZD130, in part by the National Natural Science Foundation of China under Grant 61872231.

Competing interests: The authors have declared that no competing interests exist.

Abstract

Vector-based retrieval have been widely adopted to process online users' diverse interests for recommendations. However, most of them utilize a single vector to represent user multiple interests (UMI), inevitably impairing the accuracy and diversity of item retrieval. In addition, existing work often does not take into account the scale and speed of the model, and high-dimensional user representation vectors need high computation cost, leading to inefficient item retrieval. In this paper, we propose a novel lightweight multi-interest retrieval network (MIRN) by incorporating sequence-to-interest Expectation Maximization (EM) routing to deal with users' multiple interests. By leveraging representation ability of the Capsule network, we design a multi-interest representation learning module that clusters multiple Capsule vectors from the user's behavior sequence to represent each of their interests respectively. In addition, we introduce a composite capsule clustering strategy for the Capsule network framework to reduce the scale of the network model. Furthermore, a Capsule-aware module incorporating an attention mechanism has been developed to guide model training by adaptively learning multiple Capsule vectors of user representations. The experimental results demonstrate MIRN outperforms the state-of-the-art approaches for item retrieval and gains significant improvements in terms of metric evaluations.

1 Introduction

Recommendation systems are an effective way to alleviate information overload and suggest relevant options to end users to satisfy their individual needs and interests, especially in many user-oriented online services, such as e-commerce (Amazon, Taobao) and social media (Facebook, Instagram) sites [1]. Recommendation systems usually consist of two stages, namely the matching stage and the ranking stage. Fig 1 shows an illustrative diagram of this process. Here, the purpose of matching stage is to efficiently retrieve a subset of items from the entire corpus that are relevant to user's interests. That is, the massive multi-category rating dataset stored in the corpus is retrieved by our model to generate thousands of candidate items related to users' multiple interests during the matching stage. In the ranking stage, the retrieved items not only need to be refined by accurate recommendation algorithms and scoring algorithms but also need to be adjusted in conjunction with specific business rules to produce the final items to be

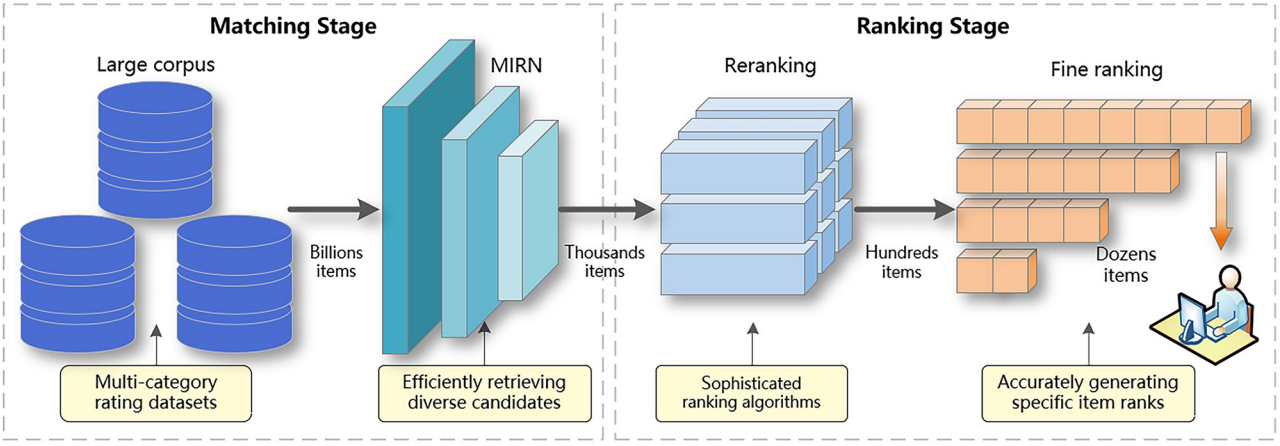

Fig 1. An illustrative diagram shows how to use MIRN in an industrial recommendation system. MIRN focuses on the matching stage, which aims to retrieve user-interested items efficiently. Note that the matching stage is concerned with retrieving good items, rather than specific item ranks.

https://doi.org/10.1371/journal.pone.0281275.g001

recommended. As the initial phase of the entire recommendation process, the performance of the retrieval model determines the upper limit of the final recommendation. Therefore, it is critical to model user interests and find user representations for matching stage in order to retrieve items that satisfy user interests.

Recently, vectorization-based retrieval methods have been widely adopted, such as DSSM [2], Deep FM [3], YouTube DNN [4], etc. The basic idea is to embed users and items into a potential vector space and utilize the inner product of vectors to represent the preferences between users and items. Although these methods have made some progress in the matching stage, existing recommendation models still have bottlenecks in learning multiple interest embeddings from user behavior sequences. Besides, knowledge graph-based approaches, such as Deep Walk [5], EGES [6], and Graph SAGE [7], are used to investigate recommendation diversity. The fundamental idea is to construct a knowledge graph utilizing users' behavioral data and then develop a function that performs an aggregated representation of surrounding vertices to provide an embedding vector of target items. While the knowledge graph-based approach makes a small contribution to solving the UMI problem, requirements for huge computational cost and storage space make it difficult to deploy on devices with limited resources. On the one hand, the size of the model and the algorithm's inference speed are the pressing issues that we need to address in our deployment. On the other hand, the amount of user-item interactions amassed in online service platforms grows over time, and user behavior sequences keep becoming larger, so extracting merely a single embedding vector from them is no longer feasible. In fact, the matching stage should take more responsibility for diversity since it is more concerned with the coverage of user-interest items, which directly determines the upper limit of recommendation effectiveness. The diversity of recommenders needs to be ensured in the matching stage first. Otherwise, the homogeneity of candidate items generated by the matching stage will inevitably lead to a lack of diversity in the final recommendations.

In this paper, to overcome above limitations, we focus on the problem of modeling different interests of users, which is crucial in real-world recommender systems. We propose a light-weight Multi-Interest Retrieval Network (MIRN) with Sequence-to-Interest Expectation Maximization routing (EM) to reflect user representations with different interests. Specifically, we design a multi-interest representation learning module that uses Sequence-to-Interest Expectation Maximization routing (S2I-EM) to adaptively cluster users' historical behaviors into user representations. For a particular user, MIRN outputs multiple representation vectors, and

each of them represents one interest tendency of the user, respectively. Ultimately these representation vectors can be used in the matching stage to retrieve user-interest related items from a large corpus. We conduct extensive experiments on two public datasets. The experimental results confirm that MIRN exhibits excellent performance compared to current state-of-the-art models. To summarize, the main contributions of this work are as follows:

- To obtain UMI from behavior sequences, we designed a multi-interest representation learning module incorporating S2I-EM routing, which adaptively clusters users' behavior sequences into multiple representation vectors. This work lays a good foundation for the successful application of EM routing algorithms in the matching stage.
- In order to reduce the size of our model and to be able to deploy it smoothly in real scenarios, we designed a composite capsule clustering strategy and applied multiple ways to speed up the computation of algorithm iterations. This is the first study to apply a model that integrates three characteristics (accuracy, diversity, and complexity) for item retrieval.
- Extensive experiments conducted on two real-world datasets demonstrate that MIRN achieves significant better performance consistently.

2 Related work

Early work on item retrieval typically used collaborative filtering (CF) to model user preferences based on their interaction history [8, 9]. Model-based collaborative filtering approaches [10–15] are an increasingly common work line. These approaches project users and items into a shared vector space and estimate a user's preference for an item by measuring the similarity between the user and the items in her/his interaction history using a pre-computed item-to-item similarity matrix. In practice, however, the number of users is frequently much greater than the number of items, resulting in a significant overhead in maintaining and storing these similarity matrices. Furthermore, a user's historical data is frequently minimal, and the accuracy of finding similar users who have only made a few transactions or clicks is low.

In the last few years, tree-based approaches have been extensively studied in the item matching stage [16–18], especially in capturing the diversity of user interests. Tree-based retrieval approaches [19–23] use a tree structure as an index and attach each item in the corpus to a leaf node by clustering [23] or joint learning [22]. Specifically, a node in the tree represents an embedding vector, and the model outputs a probability value corresponding to that vector by computing it. The probability value is used to describe the user's interest in a particular node, and the whole tree represents the hierarchical information about the user's interest (diversity of user interests). In addition, although the tree-based retrieval algorithm allows the model to learn more information about the interaction between users and candidate items (improve the accuracy of item retrieval), the complexity of the structure and algorithm limits it from wide application in industry. Therefore, tree-based retrieval algorithms have greater potential for improvement both theoretically and engineering-wise.

Since then, a number of vector-based approaches for diversifying retrieval results have been proposed. Embedding approaches have had a major impact in both academia and industry. Covington et al. [4] mapped users and items into a low-dimensional dense vector in a YouTube video recommendation system, and then performed item retrieval by an efficient retrieval method. In the deep-attention network (DAN) [24], the self-attention mechanism guides the model to appropriately filter feature vectors to reduce the interference of redundant information. Huang et al. [25] developed a vector embedding-based retrieval framework and applied it to an inverted index-based search system (Facebook). This work is a successful practice of vector embedding-based retrieval algorithms deployed to large search systems. Nie et al.

[26] proposed a novel cross-domain learning network (CLN) for 2D image-based 3D shape retrieval task. Xinyang, Yi et al. [3] proposed a neural deep retrieval (NDR) algorithm based on a two-tower network architecture for retrieving personalized suggestions from a corpus of tens of millions of videos (YouTube). The core idea is to encode a wide variety of item feature vectors using a network called item tower. An advantage of these vector embedding-based retrieval methods is that quantization-based indexing [27] techniques and hierarchical graph indexing [28] techniques can be employed to speed up their retrieval efficiency.

Recently, several researchers have attempted to use graph neural networks [29–33] for item retrieval [34, 35]. Hamilton et al. presented GraphSAGE [7], a new graph learning model that learns the embedding representation of the target node by aggregating the feature information of the node's neighbors, i.e., to improve the user-side representation and accuracy in the matching stage. Ying et al. [36] created PinSAGE, an efficient graph convolution approach that enriches item-side representation by generating item-based knowledge graphs. Besides, to improve the diversity and robustness of item retrieval, the model is trained with increasingly difficult samples. While graph-based retrieval approaches can take into consideration the diversity of user interests, the complexity of knowledge graph construction, as well as the higher storage capacity and processing capability requirements, are the most prominent challenges for deployment in real industry applications.

3 Methodology

3.1 Problem formalization

In this article's recommendation scenario, we let $\mathcal{U} = \{u_1, u_2, \ldots, u_{|\mathcal{U}|}\}$ be the user set and $\mathcal{I}$ be the complete items pool, where $|u|$ is the total number of unique users. Each user $u \in \mathcal{U}$ can be associated with an item set $\mathcal{I}_u \in \mathcal{I}$, i.e., $\mathcal{I}_u$ stores all items interacted by user u (also called user behavior). The goal of our framework is to retrieve a subset of items from a million-scale items pool $\mathcal{I} - \mathcal{I}_u$ for each user $u \in \mathcal{U}$ such that the subset contains only a few thousand items and each item is relevant to the user's interests. In addition, the basic information (like user gender and age) of user u can be represented by $\mathcal{B}_u$, and $\mathcal{F}_i$ is used to represent the characteristics of the target item, such as item id and category id.

The core task of MIRN is to learn a function that maps raw features into user representations, which can be expressed as

$$\mathcal{M}_u = f_{user}(\mathcal{B}_u, \mathcal{I}_u),$$

where the $\mathcal{M}_u = \{\vec{m}_u^1, \vec{m}_u^2, \ldots \vec{m}_u^{\mathcal{K}}\} \in \mathbb{R}^{d \times \mathcal{K}}$ denotes the representation vectors of user u. d is the dimensionality, and $\mathcal{K}$ denotes the number of representation vectors. In this paper, we use a dynamic strategy to replace the fixed parameter $\mathcal{K}$. The details are described in Eq (16). Besides, the representation vector of target item i is obtained by an embedding function as

$$\vec{v}_i = f_{item}(\mathcal{F}_i),$$

where $\vec{v}_i \in \mathbb{R}^{d \times 1}$ denotes the representation vectors of item i. Now a similarity score can be computed based on the user representation vector and the item representation vector to filter out the top N items related to the interests of each user u from the item candidate pool. The scoring function is as follows:

$$f_{score}(\mathcal{M}_u, \vec{v}_i) = \max\{\vec{v}_i^{\top} \vec{m}_u^k\}, 1 \leq k \leq \mathcal{K},$$

where N is the final number of candidates, which is set in advance based on the whole system. The main notations we used are listed in Table 1.

Table 1. Glossary.

Symbol	Description
$\mathcal{U}$	the set of users
$\mathcal{I}$	the items pool
$\mathcal{I}_u$	the item set interacted by user u
$\mathcal{B}_u$	basic information of user u
$\mathcal{F}_i$	the characteristics of target item i
$\mathcal{M}_u$	the representation of user u
$\vec{v}_i$	the representation of item i
$\mathcal{P}$	the lower layer behavioral Capsules
$\mathcal{Q}$	the upper layer interest Capsules
$\mathcal{K}$	the expected number of interests
$\mathcal{R}_{ij}$	the connection probability
$\mathcal{V}_{ij}$	the vote matrix
N	the number of candidates

https://doi.org/10.1371/journal.pone.0281275.t001

3.2 Embedding layer

As shown in Fig 2, the input of MIRN contains three parts, user basic information $\mathcal{B}_u$, user behavior sequence $\mathcal{I}_u$, and label item $\mathcal{F}_i$. The label item is split from user behaviors and belongs to a recent real purchase made by a user. Due to the large size of the dataset, the dimensionality of each feature is high, especially some categorical id features, so it is difficult to operate with one-hot encoding and the similarity between features is difficult to capture. To reduce the number of parameters in the computation process, we adopt the word2vec [37] embedding technique to embed high-dimensional features into low-dimensional dense vectors, and each feature of this dense vector can be considered to have practical value. Specifically, for users containing features such as age and gender from user basic information $\mathcal{B}_u$ are encoded into user feature embeddings $\vec{b}_u$, while item ids along with other categorical ids from $\mathcal{F}_i$ are encoded to form the label item embedding $\vec{v}_i$ after averaging the pooling layer. And the items interacted by users from $\mathcal{I}_u$, corresponding item embeddings are obtained to form the user behavior embedding.

3.3 Multi-interest representation learning

The multi-interest representation learning module is developed to learn multiple representation vectors from the user's behavior sequence, and finally to form the corresponding interest Capsule of the user.

3.3.1 Expectation maximization routing. We begin with a brief review of the important basics about Expectation Maximization (EM) routing, an iterative algorithm that can also be considered as a new form of neural units represented by matrices. The original EM routing was designed for image processing by computing pixel values to update background images, cannot being applied for recommendation task. However, the powerful feature capture and attribute processing capabilities exhibited by EM routing at the fine-grained level provide us an important insight that we may use it to handle category properties with strong entity representation in recommendation tasks.

Another key basis is the Gaussian Mixture Model (GMM), a clustering method utilized in EM routing. The advantages of the GMM can be summarized as follows: on the one hand, it can handle nonlinear decision boundaries and produce good clustering results. On the other hand, it can cluster data based on its distribution and interpret the clustering results more

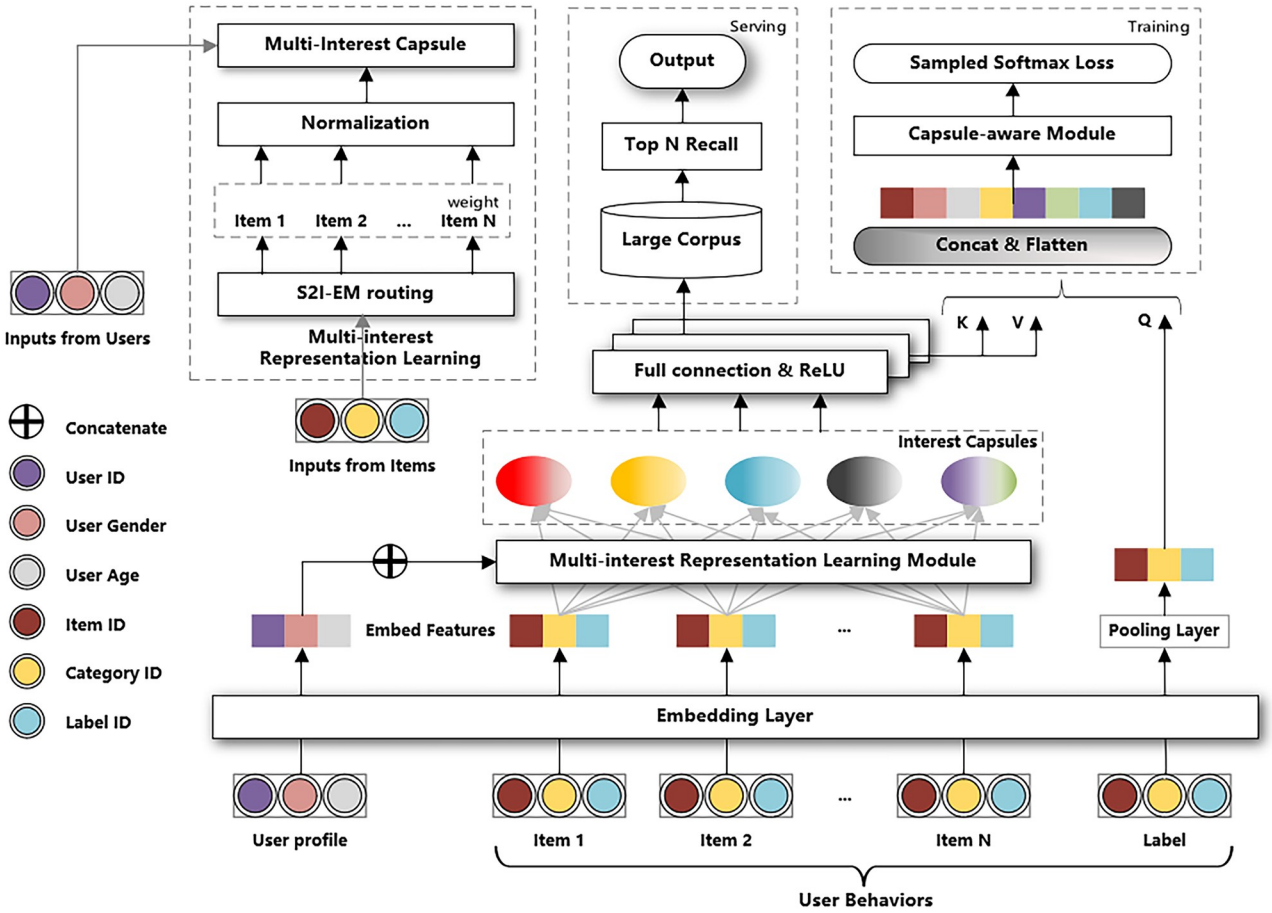

Fig 2. Network structure of MIRN. MIRN takes a sequence of user profile features and user behaviors as input and outputs multiple user representation vectors for item retrieval. User behaviors and other user features (gender, age, etc.) are fed to the multi-interest representation learning module through the corresponding operations (feature embedding and average pooling) of the embedding module. Then, the interest Capsules generated by the multi-interest representation learning module are stitched with the user's other feature embeddings separately, and multiple user representation vectors are obtained through several Full connection & ReLU layers. Finally, multiple user representation vectors are used in parallel for Top N item retrieval. The Capsule-aware module is mainly used to guide the training of the model.

https://doi.org/10.1371/journal.pone.0281275.g002

accurately. In this paper, we suppose that the user historical behavior data can be separated into $\mathcal{K}$ categories and that they all follow a Gaussian distribution, the probability density function of GMM may be stated as:

$$\begin{aligned} \mathcal{P}(x) &= \sum_{j=1}^{\mathcal{K}} \mathcal{P}(j)\mathcal{P}(x|j) \\ &= \sum_{j=1}^{\mathcal{K}} \pi_j \mathcal{N}(x; \mu_j; \Sigma_j) \end{aligned} \tag{1}$$

where j represents the category, $\mathcal{P}(j)$ denotes the probability of each category being chosen, usually written as π_j. The probability distribution $\mathcal{P}(x|j)$ for each category/interest can be expressed as:

$$\mathcal{N}(x; \mu_j; \Sigma_j) = \frac{1}{\sqrt{(2\pi)^d |\Sigma_j|}} exp^{(-\frac{1}{2}(x-\mu_j)^\top \Sigma_j^{-1}(x-\mu_j))} \tag{2}$$

where d represents the dimensionality of input vector/matrix x. μ_j, Σ_j are the mean (vector) and covariance (matrix) of the GMM distribution, respectively.

Intuitively, the values of three parameters π_j, μ_j and Σ_j need to be determined during the UMI modeling. There is a deduction based on the Wiener-khinchin law of large numbers, for a Gaussian distribution, the results of default maximum likelihood estimation are equal to the second-order moment estimation in a large enough amount of data. We arrive at a simplified version of the derivation process based on Bayesian definition and the formula is as follows:

$$\begin{aligned} \mathcal{P}(j|x) &= \frac{\mathcal{P}(x|j)\mathcal{P}(j)}{\mathcal{P}(x)} \\ &= \frac{\pi_j \mathcal{N}(x;\mu_j;\Sigma_j)}{\sum_{j=1}^{\mathcal{K}} \pi_j \mathcal{N}(x;\mu_j;\Sigma_j)} \end{aligned} \tag{3}$$

The mean μ can be calculated as:

$$\begin{aligned} \mu_j &= \int \mathcal{P}(x|j) = E\left[\frac{\mathcal{P}(j|x)}{\mathcal{P}(j)}x\right] \\ &= \frac{1}{n}\sum_{i=1}^{n}\frac{\mathcal{P}(j|x_i)}{\mathcal{P}(j)}x_i \\ &= \frac{1}{\pi_j n}\sum_{i=1}^{n}\mathcal{P}(j|x_i)x_i c \end{aligned} \tag{4}$$

Similarly, the covariance matrix Σ and the parameter π_j are calculated as follows:

$$\Sigma_j = \frac{1}{\pi_j n}\sum_{i=1}^{n}\mathcal{P}(j|x_i)x_i(x_i-\mu_j)^{\top}(x_i-\mu_j) \tag{5}$$

$$\pi_j = \frac{1}{n}\sum_{i=1}^{n}\mathcal{P}(j|x_i) \tag{6}$$

The above parameters vary in the S2I-EM routing algorithm inexorably, regulating the quality of user interest diversity. Obtaining their maximum likelihood estimates would be ideal, however they do not have a close form analytical solution. Therefore, we propose the S2I-EM routing algorithm for solving the above problem.

3.3.2 Sequence to interest expectation maximization routing. There are two layers in our routing framework, the lower layer is the sequence of user's behaviors (behavioral Capsules) and the upper layer is the Capsules obtained by algorithmic clustering, which are user's interest Capsules. In this paper, $\mathcal{P}_i, (1 \leq i \leq n)$ is used to denote lower layer behavioral Capsules, and n is the number of users' behavioral Capsules. We let $\mathcal{Q}_j, (1 \leq j \leq \mathcal{K})$ denote the upper Capsules, where $\mathcal{K}$ is the number of interest Capsules/categories. The goal of S2I-EM routing is to compute the values of upper Capsules in an iterative manner once the values of lower Capsules are known. The specific details will be discussed in the following four ways.

1) ***Covariance matrix replacement with a diagonal matrix***. Without loss of generality, the covariance matrix Σ_j is defined as a diagonal matrix, i.e., $\Sigma_j = diag\sigma_j^2$, where σ_j^2 is the variance vector of category j. Therefore, the variance vector is used to describe the covariance matrix mainly for the following considerations. On the one hand, we can avoid solving the inverse of a matrix as well as the determinant, which can greatly reduce memory consumption and computation, improving efficiency of algorithm. On the other hand, σ_j is a standard deviation vector, and σ_j^l denotes the *l-th* component of this standard deviation vector, which makes each

component of input vector/matrix x decoupled and remains independent. Based on the above analysis, the simplification of Eq (2) can be obtained as:

$$\mathcal{N}(x;\mu_j;\Sigma_j) = \Pi_{l=1}^{d}\frac{1}{\sqrt{2\pi}\sigma_j^l} exp^{\left(-\frac{1}{2(\sigma_j^l)^2}(x_i^l-\mu_j^l)^2\right)} \tag{7}$$

where μ_j^l denotes the *l-th* component of mean vector of interest *j*. Similarly, the solution expression of Σ_j is converted to the expression of σ_j^2 by the following:

$$\sigma_j^2 = \frac{1}{\Sigma_{i=1}^{n}\mathcal{P}(j|x_i)}\sum_{i=1}^{n}\mathcal{P}(j|x_i)(x_i-\mu_j)^2 \tag{8}$$

As a result, we may construct a local representation of the S2I-EM routing, as shown in Algorithm 1.

Algorithm 1: Local version of *S2I* EM Routing

Procedure S2I EM ROUTING
$\forall_i \in \Omega_i$, $j \in \Omega_j$: **Initialize** $\mathcal{R}_{ij} = \frac{1}{|\Omega_j|}$
for iter=1,...,r **do**
  for all behavior capsule i: $p_{ij} \leftarrow \mathcal{N}(\mathcal{P}_i;\mu_j;\sigma_j^2)$
  for all behavior capsule i: $\mathcal{R}_{ij} \leftarrow \frac{\pi_j p_{ij}}{\sum_{j=1}^{k}\pi_j p_{ij}}$, $r_{ij} \leftarrow \frac{\mathcal{R}_{ij}}{\sum_{i=1}^{n}\mathcal{R}_{ij}}$
  for all interest capsule j: $\mathcal{Q}_j \leftarrow \sum_{i=1}^{n} r_{ij}\mathcal{P}_i$
  for all interest capsule j: $\sigma_j^2 \leftarrow \sum_{i=1}^{n} r_{ij}(\mathcal{P}_i - \mathcal{Q}_j)^2$
  for all interest capsule j: $\pi_j \leftarrow \frac{1}{n}\sum_{i=1}^{n}\mathcal{R}_{ij}$
end
return $\{\mathcal{Q}_j, \pi_j\}$

$$p_{ij} = \mathcal{N}(\mathcal{P}_i;\mu_j;\sigma_j^2) \tag{9}$$

$$\mathcal{R}_{ij} = p(j|x_i) \tag{10}$$

In this paper, p_{ij} and $\mathcal{R}_{ij}$ are two simple notations for Eqs (9) and (10), respectively. And $\mathcal{R}_{ij}$ is the connection probability between behavioral and interest Capsules, which we initialize to a uniform distribution. The calculation of upper interest Capsule $\mathcal{Q}_j$ will be discussed in Eq (15). The connection probability $\mathcal{R}_{ij}$ is used to quantify the correlation between the behavior Capsules and the interest Capsules. For example, if cluster B (lower-level behavioral Capsule) and cluster A (upper-level interest Capsule) are independent of each other, then the connection probability between them is 0. The high correlation between them corresponds to a high connection probability value. Furthermore, the connection probability of a Capsule is governed by the activation probability, i.e., the Capsule with a zero activation probability value also has a zero connection probability.

2) ***A composite capsule clustering strategy***. It is important to select $\mathcal{K}$ behavioral Capsules at random to represent the mean value (centroid) of each large interest cluster in the GMM clustering procedure, and then update this mean value in iterations. However, the cost of iterative updates to the GMM is enormous, which not only increases the computing overhead of our model but also severely affects the convergence speed of the algorithm. In addition, large-size models are difficult to deploy in real scenarios where storage space is limited. To ligthen the architecture of our model and retain the advantages of GMM clustering, we designed a composite capsule clustering strategy. Specifically, we first apply a lightweight clustering

algorithm, namely *K-means*, to accomplish coarse-grained clustering of user behavioral Capsules and generate $\mathcal{K}$ data clusters. The GMM algorithm is then utilized to complete a fine-grained partitioning of the $\mathcal{K}$ data clusters, with the cluster results/centroids acquired from *K-means* serving as one of the inputs to the GMM algorithm. The following are some of the benefits of this strategy: 1) Since the cluster centroids supplied by *K-means* are identical to the mean μ discovered by GMM, the computation time of mean μ in GMM can be considerably reduced. 2) The composite clustering technique minimizes the number of GMM iterations, reduces the size of the model, and speeds up the algorithm's convergence. Note that we also pre-set a value for the stop iteration operation, which can be used to force termination if the algorithm does not converge for a long time.

3) ***Information entropy describes activation value***. Similar to traditional neural networks, we use a scalar a_j to measure the salience of Capsule features in the information propagation process of S2I-EM routing, where we call a_j the activation value of a Capsule. Then we ask: *how to obtain the activation value?* A promising solution is as follows:

We can cluster users' interests by using GMM and obtain a probability distribution $p(x|j)$, which can describe a user-specific interest Capsule/category. The salience of Capsules can be indirectly measured by the degree of solidarity, so we introduce information entropy in S2I-EM routing to quantify the activation value. Specifically, the salience of a Capsule can be represented by the information entropy, and the higher information entropy (close to uniform distribution), the smaller activation value. On the contrary, the lower information entropy (the closer to the Gaussian distribution), the higher activation value of the category. The information entropy $\mathcal{S}_j$ in our S2I-EM routing is defined in the following way:

$$\mathcal{S}_j = -\sum_{i=1}^{n} r_{ij} \ln p_{ij} = \frac{d}{2} + \left(\sum_{l=1}^{d} \ln \sigma_j^l + \frac{d}{2} \ln (2\pi)\right) \sum_{i=1}^{n} r_{ij} \tag{11}$$

We believe that the high information entropy of a category group indicates that the current category group is still chaotic. Take for example, there is still interest belonging to category B clustered in category A, making the Capsules of category A clusters hold less weight in the voting process. Conversely, when the information entropy is small, it means that the cluster is close to convergence and can contribute to the upper-level Capsule, hence the corresponding activation value should be larger.

Here we have supplemented the derivation of Eq (11). Information entropy can be used to measure the degree of confusion of a category, which in turn can describe the activation value of the category. Firstly, we give the formula for calculating information entropy as:

$$\begin{aligned}
\mathcal{S}_j &= -\int p(x|j) \ln p(x|j) \mathrm{d}x \\
&= -\frac{1}{p(j)} \int p(j|x)p(x) \ln p(x|j) \mathrm{d}x \\
&= -\frac{1}{p(j)} E[p(j|x) \ln p(x|j)] \\
&= -\frac{1}{n\pi_j} \sum_{i=1}^{n} \mathcal{R}_{ij} \ln p_{ij} \\
&= -\frac{1}{\sum_{i=1}^{n} \mathcal{R}_{ij}} \sum_{i=1}^{n} \mathcal{R}_{ij} \ln p_{ij} \\
&= -\sum_{i=1}^{n} \mathcal{R}_{ij} \ln p_{ij}
\end{aligned} \tag{12}$$

Note that, one of the essential reasons why we use the above procedure to calculate the information entropy instead of integrating it directly is that the latter calculates the theoretical result and the former obtains the actual result for this batch of data.

Next, we work on the derivation of the information entropy in S2I-EM routing. Here, it is assumed that p_{ij} corresponds to an independent normal distribution of d elements with the following equation:

$$\begin{aligned}\mathcal{S}_j^l &= -\sum_{i=1}^{n} r_{ij}\ \ln\ p_{ij}^l \\ &= -\sum_{i=1}^{n} r_{ij}\ \ln\ \left[\frac{1}{\sqrt{2\pi}\sigma_j^l} exp^{\left(-\frac{(x_i^l-\mu_j^l)^2}{2(\sigma_j^l)^2}\right)}\right] \\ &= \left(\frac{1}{2}\ \ln\ 2\pi +\ \ln\ \sigma_j^l\right)\sum_{i=1}^{n} r_{ij} + \frac{\sum_{i=1}^{n} r_{ij}(x_i^l-\mu_j^l)^2}{2(\sigma_j^l)^2} \\ &= \left(\frac{1}{2}\ \ln\ 2\pi +\ \ln\ \sigma_j^l\right)\sum_{i=1}^{n} r_{ij} + \frac{1}{2}\end{aligned} \tag{13}$$

This concludes the derivation of the information entropy in S2I-EM routing.

Since the value of information entropy is inversely proportional to the significance of Capsule features, it is a more reasonable choice to use $-\mathcal{S}_j$ to measure the activation value of features. Ultimately, activation value a_j is compressed (interval 0-1) by the sigmoid function to more intuitively reflect the activation level of Capsule features.

$$a_j = sigmoid(\gamma_a - (\gamma_b + \sum_{l=1}^{d}\ \ln\ \sigma_j^l)\sum_{i=1}^{n} r_{ij}) \tag{14}$$

where γ_a, γ_b is a set of training parameters assigned to each upper-level Capsule, optimized by backpropagation.

4) ***Number of dynamic interests***. Let's explore more details of the upper-level interest Capsule $\mathcal{Q}_j$ calculation. The probability that behavior Capsule i is clustered into the same interest Capsule j is influenced by the similarity between voting matrices $\mathcal{V}_{ij}$. Specifically, the voting matrix $\mathcal{V}_{ij}$ is obtained by the transformation of behavioral Capsule i, i.e., behavior Capsule i and weight matrix $\mathcal{W}_{ij}$ are multiplied by the product operation, where the weight matrix is learned by cost function and backpropagation.

$$\mathcal{V}_{ij} = \mathcal{P}_i \mathcal{W}_{ij} \tag{15}$$

Note that the voting matrix $\mathcal{V}_{ij}$ of each behavior Capsule i is associated with the predicted value of an element in interest Capsule j. That is, the mean value μ_j^h of the Gaussian distribution of elements in voting matrix indicates the h-th element in interest Capsule j. We essentially explain the process of S2I-EM routing execution, i.e., extracting multiple interest Capsules from the lower-level behavior Capsules to represent the UMI.

In addition, since user behaviors with implied interest diversity exhibit a distinct multi-peaked distribution, where the number of peaks corresponds to the number of interest Capsules. The value of $\mathcal{K}$ must be adjusted for different users. We employ heuristic rules to adjust how many interest Capsules will be given in the end. Specifically, the number of dynamic interests is determined by a combination of the Gaussian distribution of user behaviors and a

predetermined $\mathcal{K}$. The number of dynamic interests $\mathcal{K}_u$ for user u is calculated as:

$$\mathcal{K}_u = max(1, min(\mathcal{K}, \log_2|\mathcal{I}_u|)) \tag{16}$$

We do not propose that all users employ the same value of $\mathcal{K}$, because this dynamic technique of altering the number of user interests is primarily designed to provide better assistance for those users with fewer interests, in the meaning while, saving computing and memory resources. Algorithm 2 lists the whole S2I EM routing process.

3.4 Capsule-aware module

With the multi-interest representation learning module, we obtain multiple interest Capsules from user's behavioral embeddings to represent different interests. Then we ask: *How to measure the importance of these interest Capsules?* We attempt to answer this question in the following perspective:

A user's final behavior will be influenced by a certain interest, i.e., there is a clear correlation between the user's actual behavior and a specific interest Capsule. Therefore, we designed a module based on a dot product attention mechanism during the training process, called Capsule-aware, for easier selection of interest Capsules to guide the model training. In Capsule-aware module, the label item is the query and the interest Capsules are both keys and values, as shown in Fig 2. Specifically, we first calculate the compatibility between each interest Capsule and the label item embedding. Second, we selected an interest Capsule corresponding to the maximum compatibility and calculated its weighted values, where the weight of the interest Capsule is determined by the corresponding compatibility. Finally, the user representation vector of the label item is represented by the weighted values of the selected interest Capsule and used to guide the training of our model. It's worth noting that one of the primary reasons we didn't utilize the weighted sum of all interest Capsules as the user representation vector for the label item throughout the training phase was to avoid confounding UMI and compromising the user's maximum expectation. The target interest Capsule is obtained by the following equation:

$$\vec{q}_{usim} = f_{att}(\vec{q}_j^{\top}\vec{v}_i) \tag{17}$$

where $\vec{q}_j \in \mathcal{Q}$ denotes representation vector of user's $j-th$ interest Capsule, $j \in \{1, \ldots, \mathcal{K}_u\}$. f_{att} is used to compute the similarity between labeled items and the interest Capsule. Thus, the final expression $\vec{m}_u$ for user u is computed as follows:

$$\vec{m}_u = f_{concat}(\vec{q}_{usim}, \vec{m}_u) \tag{18}$$

where f_{concat} represents the splicing function of user vector and the interest Capsule.

Algorithm 2: *S2I*-EM Routing

Input: user behavior embeddings $\{\vec{v}_i, i \in \mathcal{I}_u\}$, iteration times r, number of interest Capsules $\mathcal{K}$

Output: Return **Activations** a_j and **Representation** $\mathcal{Q}_j, j = 1, \ldots, \mathcal{K}_u$ of interest Capsules

Pre-processing

Initialize activations $a_i = 0.65$ of behavior Capsules

Calculate adaptive number of interest Capsules $\mathcal{K}_u$ by (16)

Calculate votes matrix $\mathcal{V}_{ij}$: $\mathcal{V}_{ij} = \vec{v}_i \mathcal{W}_{ij}$

Procedure *K-means*

Arbitrarily choose $\mathcal{K}_u$ objects from $\mathcal{I}_u$ as the initial centroid, i.e., $\mathcal{C} \leftarrow$ *RandomSimple*$(\mathcal{I}_u, \mathcal{K}_u)$

repeat

Assign each object $\vec{v}_i$ to its closest centroid
Update the cluster means, i.e., compute the new centroid (mean) of each cluster
until *no change;*
return $\mathcal{C} = \{c_1, c_2, \ldots, c_j\}$ is a set of k *cluster centroids*
Procedure *S2I-EM* **ROUTING**
$\forall i \in \mathcal{I}_u, j \in \mathcal{K}_u$: **Initialize**$\mathcal{R}_{ij} = 1/|\mathcal{K}_u|$
for *iter = 1,...,r* **do**
  $\forall j \in \mathcal{K}_u : \mathbf{M} - \mathbf{STEP}(\mathbf{a}, \mathcal{R}, \mathcal{Q}, \mathcal{S})$
  $\forall i \in \mathcal{I}_u : \mathbf{E} - \mathbf{STEP}(\mathbf{a}, \mathcal{V}, \mathcal{R})$
end
return $a, \mathcal{Q}$
Procedure $\mathbf{M} - \mathbf{STEP}(\mathbf{a}, \mathcal{Q}, \mathcal{S})$
for all interest capsule j: $\mathcal{Q}_j \leftarrow \sum_{i=1} r_{ij} \mathcal{V}_{ij}$
for all interest capsule j: $\sigma_j^2 \leftarrow \sum_{i=1} r_{ij} (\mathcal{V}_{ij} - \mathcal{Q}_j)^2$
for all interest capsule j: $\mathcal{S}_j \leftarrow (\gamma_b + \sum_{l=1}^{d} \ln \sigma_j^l) \sum_i r_{ij}$
for all interest capsule j: $a_j \leftarrow sigmoid(\gamma_a - \mathcal{S}_j)$
Procedure $\mathbf{E} - \mathbf{STEP}(\mathbf{a}, \mathcal{V}, \mathcal{R})$
for all behavior capsule i: $p_{ij} \leftarrow \mathcal{N}(\mathcal{V}_{ij}; c_j; \sigma_j^2)$
for all behavior capsule i: $\mathcal{R}_{ij} \leftarrow \frac{a_j p_{ij}}{\sum_{j=1}^{\mathcal{K}_u} a_j p_{ij}}, r_{ij} \leftarrow \frac{a_i \mathcal{R}_{ij}}{\sum_i a_i \mathcal{R}_{ij}}$
EndProcedure

3.5 Multi-interest representation training objective

After the final user representation $\vec{m}_u$ and the label item embedding $\vec{v}_i$ are learned, we calculate the likelihood score $p(i|u)$ with respective to candidate item i as:

$$p(i|u) = p(\vec{v}_i|\vec{m}_u) = \frac{exp(\vec{m}_u^\top \vec{v}_i)}{\sum_{j=1}^{|\mathcal{I}_u|} exp(\vec{m}_u^\top \vec{v}_j)} \tag{19}$$

After that, we train MIRN by minimizing the probability of user u interacting with the target item i. In this sense, we can give the model a high level of flexibility and nonlinear modeling capabilities. The complete loss function of our MIRN is as follows:

$$min\mathcal{L} = -\sum_{u \in \mathcal{U}, i \in (\mathcal{I} - \mathcal{I}_u)} \log p\,(i|u) \tag{20}$$

The sum operator of Eq (19) is computationally expensive; thus, we use the sampled softmax technique [4] to make the objective function trackable and optimize the network by applying Adam [38], which is a variant of Stochastic Gradient Descent (SGD) to adapt the learning rate automatically. After training, the MIRN network except for the Capsule-aware module can be used as a user representation mapping function f_{user}. And in the matching stage, the user's behavior sequence and user base information are fed into the f_{user} function, generating multiple representation vectors for each user. These representative vectors are then used to retrieve the top N items through some approximate nearest neighbor methods, such as the Elasticsearch search engine [39]. Note that once the training is completed, we fix the model and construct approximate preference embeddings by fusing other attribute features of users to solve the cold start problem for new users.

4 Experiments

In experiments, we conduct extensive offline evaluations to verify that MIRN can improve both accuracy and diversity. In this section, we attempt to answer the following three research questions:

- RQ1: How does the proposed MIRN perform against different types of competitive models on recommendation accuracy in the matching stage?
- RQ2: What effect do different essential parameters have on MIRN's accuracy and diversity?
- RQ3: How do the efficiency and complexity of MIRN compared to other models?

4.1 Offline evaluation

In this section, we present a comparison of the performance of MIRN and existing methods in the offline setting.

4.1.1 Datasets and experimental setup. We evaluate recommendation performance on two publicly available datasets from different domains. One of them is the popular movie rating dataset MovieLens (https://grouplens.org/datasets/movielens/) provided by [40], which contains about 10 million user ratings from January 1996 to July 2014. The other is Amazon Product Data (http://jmcauley.ucsd.edu/data/amazon/) provided by [41], including user reviews (ratings, text, useful/unused votes, etc.) and product metadata (category, price, brand, etc.) on Amazon from May 1996 to July 2014. For MovieLens-Latest, due to the short user behavior sequence and simple data labels, we only consider the factors such as age, gender, occupation, and movie category. For Amazon Product Data, we deleted users whose operation records were less than 10 and items whose interaction records were less than 8. In addition, we consider the evaluation of fewer than three stars and above as positive, recorded as 1, and the evaluation of fewer than three stars as bad, recorded as 0. The statistics of the two datasets are shown in Table 2.

4.1.2 Compared methods.

- WALS [42] WALS, abbreviated as Weighted Alternating Least Squares, is a classical matrix decomposition algorithm. It performs the corresponding recommendation by confidence weights, giving a larger weight to items with high user preference and a smaller weight to items with no feedback.
- YouTube DNN [4]. In YouTube's recommendation process, the division of the recommendation process into two phases, retrieval and sorting, is the most successful example in the field of recommendation systems.

Table 2. Statistics of the two datasets for offline evaluation.

Dataset	Usrs	Categories	Items	Total
MovieLens-Latest	283,228	20	58,034	27,753,444
Amazon Electron	22,040	671	22,012	561,100
Amazon Movies-TV	35,832	15	28,545	752,676
Amazon Video-Games	5,459	59	4,258	83,748
Amazon Beauty	3,731	172	2,612	854,225
Amazon Office-Products	1,792	171	893	29,387
Amazon Home-Kitchen	11,511	67	7,727	143,088
Amazon Digit-Music	1,645	59	1,539	28,852

https://doi.org/10.1371/journal.pone.0281275.t002

- MaxMF [43]. It is a non-linear model. It uses multiple vectors to represent users, and then a vector to represent products. Finally, the largest score in the user vector is taken as the matching score.
- MIND [44]. The vectorized retrieval approach, represented by MIND, is one of the mainstream recommendation algorithms today. It generates multiple embeddings characterizing users' interests to enhance the matching stage and avoids the problem of causing head effects.

4.1.3 Evaluation metrics. We use the idea of sequential recommendation for prediction, i.e., we predict the next interaction between a user and an item, and thus evaluate the performance of the above method accordingly. We randomly divided the user-item interaction data into training set and test set by a ratio of 17:3. For each user, we first use the last interaction data as the target item and then use the remaining interaction data for that user as the user behavior.

- **Hit rate**. Hit rate (HR) is a commonly used metric for measuring recall in current top N recommendation studies. It calculates the percentage of underlying true items in the top N positions of the correctly ranked list, and a larger metric indicates better recommendation performance. The HR@N score is defined as:

$$HitRate@N = \frac{\sum_{(u,i)\in\mathcal{D}_{test}} f_i(i_{target} \in \mathcal{T}_n)}{|\mathcal{D}_{test}|} \tag{21}$$

  where $\mathcal{D}_{test}$ denotes the test set consisting a large number of users and target items (u, i). $\mathcal{T}_n$ is a set of the top N items in item ranking list and $f_i(\cdot)$ is an indicator function.

- **Normalized discounted cumulative gain**. NDCG is a ranking performance evaluation metric widely used in recommender systems, which indirectly evaluates the performance of model by measuring the relevance of retrieved Top N item list to user u. The higher the relevance, the better the performance. So, it is defined as:

$$NDCG@N = \frac{1}{|\mathcal{U}|}\sum_{u\in\mathcal{D}_{test}} \frac{Rank(sim) : \sum_{i=1}^{N} sim_u^i/\log\ 2(i+1)}{Rank(label) : \sum_{i=1}^{N} sim_u^i/\log\ 2(i+1)} \tag{22}$$

  where $Rank(\cdot)$ represents a ranking function on input sequence or values. sim_u^i denotes the similarity score between the representation vector of user u and the embedding vector of candidate item i, and label is the true rating value of the item.

4.1.4 Implementation details. In our experiments, we set the size of the embedding vector to 1024. We set the mini-batch size to 128 and use the Adam optimizer [38] with a 0.001 learning rate to minimize the total loss. In addition, we dynamically adjust the dropout rate in the range of {0.2, 0.4, 0.7} to prevent the overfitting of our models. And for other parameters, we use a grid search strategy to find the optimal settings. Finally, we adjust the $l2$ regularization term λ from 0.0001, increasing by a factor of 10 each time, and the learning rate from {0.001, 0.005, 0.01}. The implementation of the experiments is based on the server with the open-source framework Tensorflow-gpu2.4.0 [45], keras2.4.3 and python3.7, CUDA11.2. The GPU used for the experiments is NVIDIA GeForce RTX 3090 with 128GB memory.

4.1.5 Model complexity analysis. In this subsection, we investigate the complexity of the proposed MIRN. It is obvious that the calculation of dynamic interest capsules and S2I-EM

Table 3. Performance comparison of different models on publicly available datasets.

	MovieLens Latest		Amazon Electron		Amazon Video-Games		Amazon Movies-TV	
Metrics@50	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**	**HR**	**NDCG**
WALS	8.18	2.16	5.53	1.44	4.71	1.38	4.36	1.18
YouTube DNN	8.23	2.36	7.46	2.31	7.41	2.26	7.47	2.14
MaxMF	9.03	2.44	8.62	2.85	9.78	2.93	8.15	2.61
MIND	10.32	3.07	10.41	3.09	**10.22**	**3.11**	10.72	3.19
MIRN	**10.48**	**3.11**	**10.91**	**3.19**	10.19	3.01	**10.94**	**3.25**
Imp.	+1.55%	+1.30%	+4.80%	+3.23%	-0.29%	-3.32%	+2.05%	+1.88%

The best results are highlighted in bold. Numbers in the table indicate percentages, omitting the '%'.

https://doi.org/10.1371/journal.pone.0281275.t003

routing are the main operations. Given $|u|$ users and $|\mathcal{I}|$ items, suppose that each user directly connects with $|\mathcal{I}_u|$ items. In the process of calculating dynamic interest capsules, the computational complexity of procedure K-means is $\mathcal{O}(\mathcal{K}_u)$, where $\mathcal{K}_u$ denotes the number of interest capsules. In the process of S2I-EM routing, the time cost of the E-Step is $\mathcal{O}(r|\mathcal{I}_u|)$, and the M-Step requires time complexity $\mathcal{O}(r|\mathcal{K}_u|)$, where $|r|$ represents the iteration times. Since the number of interest capsules $|\mathcal{K}_u|$ is smaller than the number of items $|\mathcal{I}_u|$ with which users interacted, we can regard it as a constant. The total time complexity is approximately $\mathcal{O}(r|\mathcal{I}_u|)$. Under the same experimental settings (i.e., representation sizes and initialized clusters), MIRN has a smaller complexity than MIND. Therefore, the total time complexity is acceptable in practice.

4.1.6 Results analysis (RQ1). We begin with the comparison w.r.t. HR@50 and NDCG@50. The empirical results are reported in Table 3, where %Imp. denotes the relative improvements of the best performing method (starred) over the strongest baselines (underlined). We find that:

- MIRN consistently outperforms most baselines across four datasets in terms of all measures. Specifically, it achieves significant improvements over the strongest baselines w.r.t. HR@50 by 4.80%, 2.05%, 1.55% in Amazon-Electron, Amazon Movie-TV, and MovieLens-Latest, respectively. This demonstrates the rationality and effectiveness of MIRN. We attribute these improvements to the multi-interest extraction capability of MIRN: (1) By mining users' different interests, MIRN is able to better describe the relationships between users and items, and produce more robust representations of users and items. In contrast, other baselines ignore the importance of multiple interest representations, and use a single vector to model UMI; (2) Benefiting from our S2I-EM routing algorithm, MIRN can preserve the overall semantics of interest propensity and collect more information from behavioral Capsules, than USI-based baselines (i.e., WALS, YouTube DNN).
- Combining four datasets for MIRN analysis, we find that the improvement is more significant on Amazon-Electron than on Amazon-Video-Games. This is reasonable since (1) the user-item interaction data on Amazon-Electron provides denser and richer information than the data on Amazon-Video-Games; and (2) sparse interest categories dominate on Amazon-Video-Games. This suggests that MIRN is good at exploiting the potential of multi-interest extraction.
- Our MIRN achieves the best performance on several datasets, including Amazon Electron and Amazon Movies-TV. Compared to MIND the improvement is around 1% to 4%. For Amazon datasets, the improvements introduced by mining UMI are more significant,

suggesting that users in Amazon tend to have more diverse interests in filtering items than in MovieLens-Latest.

- The matrix decomposition method, WALS, is defeated by other methods, indicating that deep learning is effective in improving the retrieval performance of the model. However, MaxMF performs much better than WALS without deep learning, which can be explained by the fact that MaxMF is a nonlinear model that uses multiple user representation vectors for retrieval. And we can also see from the results that the method using multiple user representation vectors (MaxMF, MIND) shows better performance than other methods (WALS, YouTube DNN). Thus, using multiple user representation vectors can effectively model UMI and boost the accuracy of recommendations.

In addition, we analyzed the experiments in which MIRN performed poorly by evaluating HR@10 and HR@50 in Table 4. An obvious finding is that these datasets (i.e., Amazon Office-Products, Amazon Beauty, and Amazon Digit-Music) are generally small (see Table 2 for details), making it difficult for the model to fit the data completely. While MaxFM performs better than MIRN on these datasets, we conjecture that (1) MaxFM benefits from the advantages of matrix factorization and can produce appropriate recommendations even with sparse data.; (2) It alleviates the high-dimensional disaster by introducing hidden vectors to decompose the high-dimensional behavioral sequences into multiple different representation vectors. This also demonstrates the effectiveness of using multiple Capsule vectors to represent UMI.

The visualization results of the performance comparison of different models on different datasets are shown in Fig 3. From the experimental process of Amazon Electron with the Amazon Movies-TV dataset, it can be found that although MIRN requires longer iteration rounds compared to MIND, the performance achieved by MIRN is more advantageous. Besides, we found that there may be oscillations in our model-fitting curve when the data is small, which indicates that our model needs further optimization to learn enough relevant features when dealing with small datasets. However, the information collected by commercial platforms servicing millions of users is generally in the tens of thousands, so MIRN's recommendation is more in accordance with the needs in real scenarios. Finally, we fine-tune our experiments on the Amazon Home-Kitchen and Amazon Video-Games datasets directly using the model already trained on Amazon Movies-TV. The model fitting curves show that our model learns the common behavioral patterns of users with good portability.

4.2 Discussion of hyperparameters (RQ2)

In this section, we conduct experiments on the Amazon Electron and Movielens-Latest datasets to investigate the effect of hyperparameters within the multi-interest representation learning module and Capsule-aware module, respectively.

Table 4. Performance comparison of different models on other Amazon datasets.

	Amazon Office-Products		Amazon Home-Kitchen		Amazon Beauty		Amazon Digit-Music	
Metrics@10/50	**HR@10**	**HR@50**	**HR@10**	**HR@50**	**HR@10**	**HR@50**	**HR@10**	**HR@50**
WALS	0.0166	0.0635	0.0133	0.0482	0.0163	0.0617	0.0156	0.0621
YouTube DNN	0.0285	0.0762	0.0223	0.0750	0.0263	0.0758	0.0268	0.0757
MaxMF	**0.0291**	**0.0932**	0.0274	0.0861	**0.0284**	0.0934	0.0274	**0.0948**
MIND	0.0288	0.0926	0.0312	0.1045	0.0271	**0.0942**	**0.0278**	0.0922
Ours	0.0282	0.0921	**0.0314**	**0.1048**	0.0268	0.0916	0.0271	0.0931

The best results are highlighted in bold.

https://doi.org/10.1371/journal.pone.0281275.t004

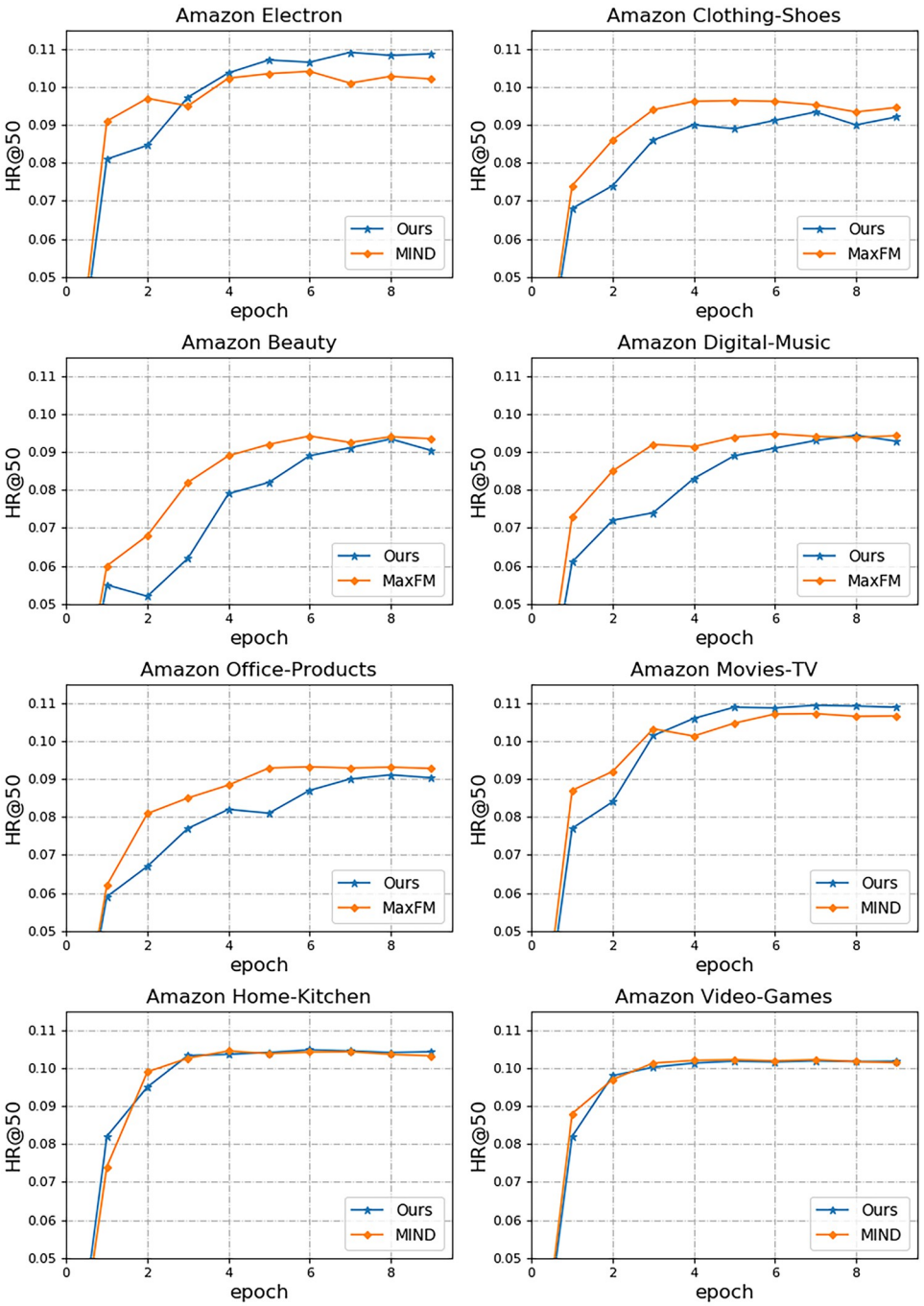

Fig 3. Comparison of HR@50 performance between our model and other models on different datasets.

https://doi.org/10.1371/journal.pone.0281275.g003

1) ***Parameter K sensitivity***. We performed a corresponding sensitivity study on the hyper-parameter $\mathcal{K}$ in the model architecture. $\mathcal{K}$ controls the overall capacity of the model. If $\mathcal{K}$ is small, the number of clusters is small for all the items; if $\mathcal{K}$ is big, the time complexity of the training and matching stage grows linearly with $\mathcal{K}$. Moreover, a large $\mathcal{K}$ may increase the possibility of overfitting. Table 5 summarizes the performance of the model when the hyperparameter $\mathcal{K}$ is changed. And we can understand that the performance of our model steadily

Table 5. Model performance for different parameters $\mathcal{K}$.

	Movielens-Latest				Amazon Electron			
	Metrics@10		Metrics@50		Metrics@10		Metrics@50	
Metrics	HR	NDCG	HR	NDCG	HR	NDCG	HR	NDCG
MIND($\mathcal{K}=3$)	31.17	6.83	**48.46**	**15.34**	3.09	1.18	10.41	1.47
MIRN($\mathcal{K}$=1)	30.24	5.60	48.31	10.66	2.21	0.82	9.57	1.03
MIRN($\mathcal{K}$=3)	31.77	6.25	48.34	12.28	**3.19**	**1.01**	10.02	1.36
MIRN($\mathcal{K}$=5)	**32.71**	**7.47**	48.40	13.53	3.16	0.97	**10.91**	**1.88**
MIRN($\mathcal{K}$=7)	31.15	7.06	48.32	13.01	3.14	0.81	10.71	1.56

All figures are presented as percentages and "%" is omitted.

https://doi.org/10.1371/journal.pone.0281275.t005

increases as $\mathcal{K}$ increases, peaking at $\mathcal{K}=5$. For the Movielens-Latest dataset, the best performance of MIRN is obtained when $\mathcal{K}=5$. For the Amazon Electron dataset, strong performance is still achieved when $\mathcal{K}=5$.

Fig 4 shows the iterative fitting speed and accuracy of MIRN on the Amazon Electron dataset. We notice that the performance is worst when $\mathcal{K}=1$. This is because users show the same intention for all items, making the accuracy of the model suboptimal due to the lack of interest diversity. However, as $\mathcal{K}$ rises, user interest diversity gradually surfaces, user-side representations get richer, and the model is able to capture more accurate information to improve the ability of item retrieval. Compared to MIND, our model exhibits higher performance with different $\mathcal{K}$ values, which demonstrates the efficacy of our modeling strategy for extracting multiple interests.

2) ***Connection probability***. The connection probability R_{ij} is used to quantify the correlation between the behavior Capsules and the interest Capsules. The value of the connection probability fluctuates continually over the algorithm's iterations, and in this study, we visualize its value to show that the multi-interest extraction process is interpretable.

Fig 5 illustrates the connection probabilities associated with a user, where each row represents one of the user's interest Capsules, and each column corresponds to one of the user's behaviors. The intensity of the color in heat map indicates the user's preference for the item. A

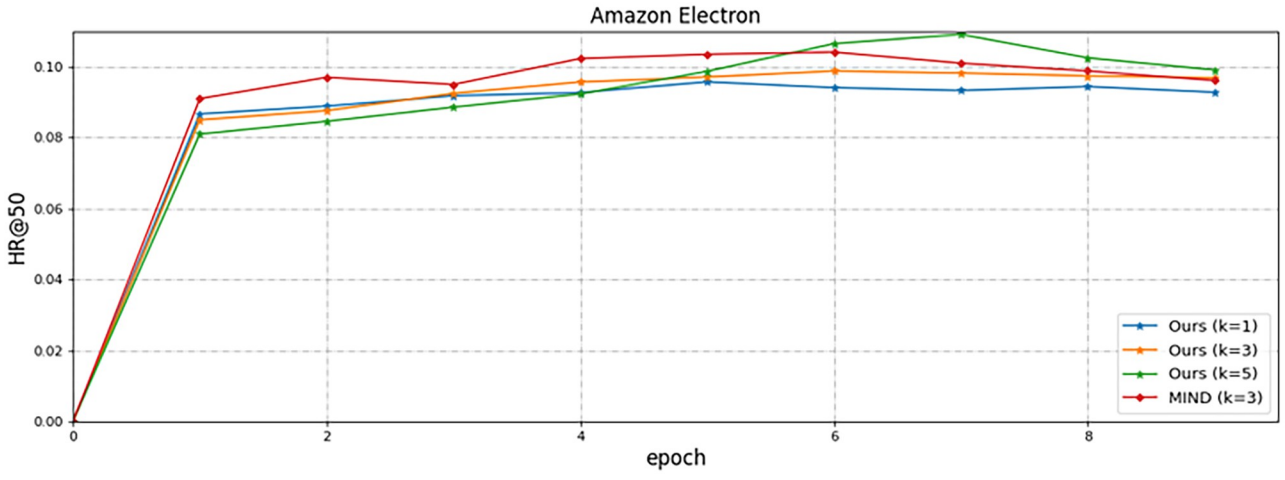

Fig 4. Performance comparison of different numbers of interest $\mathcal{K}$ in MIRN. The y-axis represents the value of HR@50 and the x-axis gives the number of iterations. MIRN performs better with bigger $\mathcal{K}$.

https://doi.org/10.1371/journal.pone.0281275.g004

Fig 5. Heatmap of coupling coefficients for user behaviors. Each behavior has a corresponding coupling coefficient in the corresponding capsule of interest.

https://doi.org/10.1371/journal.pone.0281275.g005

dark color indicates a strong preference for the item and, conversely, a light color implies a lack of interest. From the results, we can see that the user interacts most frequently with three categories of goods (cell phones, computers, and headphones), and their respective connection probabilities are the largest in the interest Capsule, i.e., these three categories of goods form the user's corresponding interest. Regarding this result, we believe that the work of extracting multiple interests of the user has been achieved, i.e., the UMI are divided into different interest Capsules.

4.3 Model efficiency and complexity (RQ3)

We analyze the efficiency and complexity of retrieval models in various terms. In Table 6, we report the number of parameters, the number of floating-point operations (FLOPs), and the inference latency of item retrieval in the matching stage. MaxFM learns nonlinear latent factors to represent multiple user vectors, leading to a disadvantage in parametric number and inference time. YouTube DNN represents a user with a long single vector, which has fewer parameters and FLOPs than MaxFM, but its accuracy is likewise low. This demonstrates that the use of multiple vectors to represent the user can make a greater contribution to improving models' accuracy. In addition, some of the more advanced models, such as MIND and MIRN, take the form of multi-vector representations, as indicated in Table 6. It can be observed that MIRN with a composite clustering strategy not only outperforms its much larger counterpart, MIND, in terms of performance; its reduced model size also significantly reduces inference costs and greatly facilitates the use of smaller hardware in downstream tasks. Second, in our analysis, the number of initialized interest categories $\mathcal{K}$ and the length of user behavior sequences are the key factors affecting FLOPs. Increasing the value of $\mathcal{K}$ increases the number

Table 6. Comparison of the speed and accuracy of different dense retrieval models on Amazon Electron.

Models	Params. (M)	FLOPs (G)	Latency (ms)	HR@50 (%)
Youtube DNN	93	16.4	61.2	7.46
MaxFM	147	19.6	73.7	8.62
MIND($\mathcal{K} = 3$)	67	14.1	47.9	10.41
MIRN($\mathcal{K} = 3$)	54	13.3	46.4	10.02
MIRN($\mathcal{K} = 5$)	61	13.8	47.3	10.91

The latency in this table is measured without post-processing. Latency is counted on a workstation with Intel E5-2650 v2 CPU and NVIDIA GeForce RTX 3090 GPU.

https://doi.org/10.1371/journal.pone.0281275.t006

of parameters in our model marginally while also improving its performance. In conclusion, MIRN outperforms other methods in terms of model size and multi-interest extraction.

5 Conclusion and future works

In this paper, we propose a new model named MIRN, which can generate multiple Capsule vectors of users' different interest representations for item retrieval in the matching stage. Specifically, we design an S2I EM routing algorithm to dynamically extract the different interests of users, and then these interests are trained with the Capsule-aware module. Besides, a composite clustering strategy is applied in this paper to make our model more lightweight. To investigate the performance of MIRN, we conducted extensive offline evaluations, results analysis. Experiments have shown that our model achieves significant improvements compared to the baseline approaches on two large public datasets, Movielens-Latest and Amazon Product. In the future, we will work in two directions. The first is to build a multi-channel network to capture the temporary interests generated by users in a short time. The second direction is to investigate the embedding of user preference with the GNN and metric-based learning techniques in order to enhance the performance of cold-start recommendation.

Author Contributions

Conceptualization: Xiliang Zhang, Jin Liu.

Data curation: Jin Liu, Siwei Chang.

Formal analysis: Jin Liu, Peizhu Gong.

Funding acquisition: Zhongdai Wu.

Investigation: Peizhu Gong, Bing Han.

Methodology: Xiliang Zhang.

Project administration: Jin Liu.

Resources: Zhongdai Wu, Bing Han.

Software: Xiliang Zhang, Siwei Chang.

Visualization: Siwei Chang.

Writing – original draft: Xiliang Zhang.

Writing – review & editing: Jin Liu.

References

1. Stai E, Kafetzoglou S, Tsiropoulou EE, Papavassiliou S. A holistic approach for personalization, relevance feedback & recommendation in enriched multimedia content. Multimedia Tools and Applications. 2018; 77(1):283–326. https://doi.org/10.1007/s11042-016-4209-1
2. Huang PS, He X, Gao J, Deng L, Acero A, Heck L. Learning deep structured semantic models for web search using clickthrough data. In: Proceedings of the 22nd ACM international conference on Information & Knowledge Management; 2013. p. 2333–2338.
3. Yi X, Yang J, Hong L, Cheng DZ, Heldt L, Kumthekar A, et al. Sampling-bias-corrected neural modeling for large corpus item recommendations. In: Proceedings of the 13th ACM Conference on Recommender Systems; 2019. p. 269–277.
4. Covington P, Adams J, Sargin E. Deep neural networks for youtube recommendations. In: Proceedings of the 10th ACM conference on recommender systems; 2016. p. 191–198.
5. Perozzi B, Al-Rfou R, Skiena S. Deepwalk: Online learning of social representations. In: Proceedings of the 20th ACM SIGKDD international conference on Knowledge discovery and data mining; 2014. p. 701–710.

6. Wang J, Huang P, Zhao H, Zhang Z, Zhao B, Lee DL. Billion-scale commodity embedding for e-commerce recommendation in alibaba. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. p. 839–848.
7. Hamilton W, Ying Z, Leskovec J. Inductive representation learning on large graphs. Advances in neural information processing systems. 2017; 30.
8. Sarwar B, Karypis G, Konstan J, Riedl J. Item-based collaborative filtering recommendation algorithms. In: Proceedings of the 10th international conference on World Wide Web; 2001. p. 285–295.
9. Koren Y, Rendle S, Bell R. Advances in collaborative filtering. Recommender systems handbook. 2022; p. 91–142. https://doi.org/10.1007/978-1-0716-2197-4_3
10. Paterek A. Improving regularized singular value decomposition for collaborative filtering. In: Proceedings of KDD cup and workshop. vol. 2007; 2007. p. 5–8.
11. Rendle S. Factorization machines with libfm. ACM Transactions on Intelligent Systems and Technology (TIST). 2012; 3(3):1–22. https://doi.org/10.1145/2168752.2168771
12. Rendle S, Freudenthaler C, Gantner Z, Schmidt-Thieme L. BPR: Bayesian Personalized Ranking from Implicit Feedback. In: Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence. Arlington, Virginia, USA: AUAI Press; 2009. p. 452–461.
13. Koren Y. Factorization meets the neighborhood: a multifaceted collaborative filtering model. In: Proceedings of the 14th ACM SIGKDD international conference on Knowledge discovery and data mining; 2008. p. 426–434.
14. Kabbur S, Ning X, Karypis G. Fism: factored item similarity models for top-n recommender systems. In: Proceedings of the 19th ACM SIGKDD international conference on Knowledge discovery and data mining; 2013. p. 659–667.
15. Koren Y, Bell R, Volinsky C. Matrix factorization techniques for recommender systems. Computer. 2009; 42(8):30–37. https://doi.org/10.1109/MC.2009.263
16. Prabhu Y, Kag A, Harsola S, Agrawal R, Varma M. Parabel: Partitioned label trees for extreme classification with application to dynamic search advertising. In: Proceedings of the 2018 World Wide Web Conference; 2018. p. 993–1002.
17. Tanno R, Arulkumaran K, Alexander D, Criminisi A, Nori A. Adaptive neural trees. In: International Conference on Machine Learning. PMLR; 2019. p. 6166–6175.
18. You R, Zhang Z, Wang Z, Dai S, Mamitsuka H, Zhu S. Attentionxml: Label tree-based attention-aware deep model for high-performance extreme multi-label text classification. Advances in Neural Information Processing Systems. 2019; 32.
19. Liu J, Zhang X, Li Y, Wang J, Kim HJ. Deep learning-based reasoning with multi-ontology for IoT applications. IEEE Access. 2019; 7:124688–124701. https://doi.org/10.1109/ACCESS.2019.2937353
20. Zhu H, Zhang P, Li G, He J, Li H, Gai K. Learning Tree-based Deep Model for Recommender Systems. CoRR. 2018;abs/1801.02294.
21. Zhuo J, Xu Z, Dai W, Zhu H, Li H, Xu J, et al. Learning optimal tree models under beam search. In: International Conference on Machine Learning. PMLR; 2020. p. 11650–11659.
22. Zhu H, Chang D, Xu Z, Zhang P, Li X, He J, et al. Joint optimization of tree-based index and deep model for recommender systems. Advances in Neural Information Processing Systems. 2019; 32.
23. Zhu H, Li X, Zhang P, Li G, He J, Li H, et al. Learning tree-based deep model for recommender systems. In: Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2018. p. 1079–1088.
24. Nie W, Zhao Y, Song D, Gao Y. Dan: deep-attention network for 3d shape recognition. IEEE Transactions on Image Processing. 2021; 30:4371–4383. https://doi.org/10.1109/TIP.2021.3071687 PMID: 33848247
25. Huang JT, Sharma A, Sun S, Xia L, Zhang D, Pronin P, et al. Embedding-based retrieval in facebook search. In: Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining; 2020. p. 2553–2561.
26. Nie W, Zhao Y, Nie J, Liu AA, Zhao S. CLN: Cross-Domain Learning Network for 2D Image-Based 3D Shape Retrieval. IEEE Transactions on Circuits and Systems for Video Technology. 2021; 32(3):992–1005. https://doi.org/10.1109/TCSVT.2021.3070969
27. Johnson J, Douze M, Jégou H. Billion-scale similarity search with gpus. IEEE Transactions on Big Data. 2019; 7(3):535–547. https://doi.org/10.1109/TBDATA.2019.2921572
28. Malkov YA, Yashunin DA. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs. IEEE transactions on pattern analysis and machine intelligence. 2018; 42 (4):824–836. https://doi.org/10.1109/TPAMI.2018.2889473 PMID: 30602420

29. Gong P, Liu J, Yang Y, He H. Towards knowledge enhanced language model for machine reading comprehension. IEEE Access. 2020; 8:224837–224851. https://doi.org/10.1109/ACCESS.2020.3044308
30. Fan W, Ma Y, Li Q, He Y, Zhao E, Tang J, et al. Graph neural networks for social recommendation. In: The world wide web conference; 2019. p. 417–426.
31. Jin B, Gao C, He X, Jin D, Li Y. Multi-behavior recommendation with graph convolutional networks. In: Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval; 2020. p. 659–668.
32. Tan Q, Liu N, Hu X. Deep representation learning for social network analysis. Frontiers in big Data. 2019; 2:2. https://doi.org/10.3389/fdata.2019.00002 PMID: 33693325
33. Yu W, Qin Z. Graph convolutional network for recommendation with low-pass collaborative filters. In: International Conference on Machine Learning. PMLR; 2020. p. 10936–10945.
34. He X, Deng K, Wang X, Li Y, Zhang Y, Wang M. Lightgcn: Simplifying and powering graph convolution network for recommendation. In: Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval; 2020. p. 639–648.
35. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining; 2018. p. 974–983.
36. Ying R, He R, Chen K, Eksombatchai P, Hamilton WL, Leskovec J. Graph convolutional neural networks for web-scale recommender systems. In: Proceedings of the 24th ACM SIGKDD international conference on knowledge discovery & data mining; 2018. p. 974–983.
37. Mikolov T, Sutskever I, Chen K, Corrado GS, Dean J. Distributed representations of words and phrases and their compositionality. Advances in neural information processing systems. 2013; 26.
38. Kingma DP, Ba J. Adam: A Method for Stochastic Optimization. In: 3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings; 2015. Available from: http://arxiv.org/abs/1412.6980.
39. Banon S. The Future of Compass & Elasticsearch. Retrieved Feb. 2010; 12:2017.
40. Yang D, Zhang D, Qu B. Participatory cultural mapping based on collective behavior data in location-based social networks. ACM Transactions on Intelligent Systems and Technology (TIST). 2016; 7(3):1–23. https://doi.org/10.1145/2814575
41. He R, McAuley J. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In: proceedings of the 25th international conference on world wide web; 2016. p. 507–517.
42. Aberger CR. Recommender: An analysis of collaborative filtering techniques. Personal and Ubiquitous Computing Journal. 2014;.
43. Weston J, Weiss RJ, Yee H. Nonlinear latent factorization by embedding multiple user interests. In: Proceedings of the 7th ACM conference on Recommender systems; 2013. p. 65–68.
44. Li C, Liu Z, Wu M, Xu Y, Zhao H, Huang P, et al. Multi-interest network with dynamic routing for recommendation at Tmall. In: Proceedings of the 28th ACM International Conference on Information and Knowledge Management; 2019. p. 2615–2623.
45. Abadi M, Barham P, Chen J, Chen Z, Davis A, Dean J, et al. TensorFlow: A System for Large-Scale Machine Learning. In: Proceedings of the 12th USENIX Conference on Operating Systems Design and Implementation. USA: USENIX Association; 2016. p. 265–283.