A Category-aware Multi-interest Model for Personalized Product Search

Jiongnan Liu[1], Zhicheng Dou[1], Qiannan Zhu[1], and Ji-Rong Wen[2,3]
[1]Gaoling School of Artificial Intelligence, Renmin University of China, China
[2]Beijing Key Laboratory of Big Data Management and Analysis Methods, China
[3]Key Laboratory of Data Engineering and Knowledge Engineering, MOE, China
liujn@ruc.edu.cn,dou@ruc.edu.cn,zhuqiannan@ruc.edu.cn,jirong.wen@gmail.com

ABSTRACT

Product search has been an important way for people to find products on online shopping platforms. Existing approaches in personalized product search mainly embed user preferences into one single vector. However, this simple strategy easily results in sub-optimal representations, failing to model and disentangle user's multiple preferences. To overcome this problem, we proposed a category-aware multi-interest model to encode users as multiple preference embeddings to represent user-specific interests. Specifically, we also capture the category indications for each preference to indicate the distribution of categories it focuses on, which is derived from rich relations between users, products, and attributes. Based on these category indications, we develop a category attention mechanism to aggregate these various preference embeddings considering current queries and items as the user's comprehensive representation. By this means, we can use this representation to calculate matching scores of retrieved items to determine whether they meet the user's search intent. Besides, we introduce a homogenization regularization term to avoid the redundancy between user interests. Experimental results show that the proposed method significantly outperforms existing approaches.

CCS CONCEPTS

• **Information systems → Personalization**.

KEYWORDS

personalized product search, multi-interest, knowledge graph

ACM Reference Format:
Jiongnan Liu[1], Zhicheng Dou[1], Qiannan Zhu[1], and Ji-Rong Wen[2,3]. 2022. A Category-aware Multi-interest Model for Personalized Product Search. In *Proceedings of the ACM Web Conference 2022 (WWW '22), April 25–29, 2022, Virtual Event, Lyon, France.* ACM, New York, NY, USA, 9 pages. https://doi.org/10.1145/3485447.3511964

WWW '22, April 25–29, 2022, Virtual Event, Lyon, France

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511964

1 INTRODUCTION

With the rapid development of e-commerce platforms such as Amazon and Taobao, online shopping is becoming more and more popular in people's daily life. To buy certain items, the most common paradigm is that users issue queries to describe their demands, then the platform provides item lists related to the queries for users to purchase. Previous studies [2, 11, 29–32] have shown that user history is helpful to obtain the explicit user intent and further improve the quality of search results. For example, if a user used to buy apple's products and now issues a query "computer", what she wants to buy may be "Mac". If the search engine could rank "Mac" higher, the user's satisfaction will be improved. Recently, to encode users' history and model their preferences, many personalized product search approaches [2, 3, 5, 6, 12, 13, 21, 28] have been proposed.

Existing personalized product search approaches can be roughly classified into query-independent and query-dependent ones. The query-independent approaches [3, 4, 21, 28] embed users into one general profile vector without considering the current query to represent their interests. These methods can be conveniently applied to real systems as the user representations can be calculated and stored in the offline training stage. However, they will fail to model the user's dynamic search intent without considering the issued query. To obtain user dynamic search intent, query-dependent approaches [2, 5, 6, 12] build user representation according to the current query or candidate items in the running time. Unfortunately, these approaches bring additional online computation costs because they cannot build user embeddings in advance and need to access the whole user history in the running time.

However, no matter which kind of profiles they use, most existing methods assume user interests can be represented as a single vector, which we think has the following limitations. First, a single vector in query-independent approaches cannot reflect and express users' multiple preferences in different categories. However, users may have different preferences (in brand, price, features, etc.) when they buy different kinds of products. For example, a user may prefer "iPhone" when she wants to buy a cellphone but prefer "Surface" when she needs a computer for work. Under this situation, if we embed this user into one vector, the preferences of different categories will be mixed together and recommendations based on this single preference vector will be inaccurate sometimes. Second, if we construct a single user vector considering the current query as query-dependent approaches, the query latency will be high due to the online processing time of history items of users. Therefore, we argue that we need multiple offline preference vectors to describe a user's diverse needs across different types of products. By this means, we can selectively aggregate these vectors to rank products

when the user searches for different categories of products instead of using a single vector or integrating all of the user histories.

To achieve this goal, in this paper, **we propose to use multiple vectors reflecting diverse preferences of users.** Moreover, we focus on the offline stage and build query-independent user vectors. **Firstly**, we build and store user diverse preferences separately, so that interests in different categories will not affect each other in a disorganized manner. A certain purchased item will only have effects on the interests related to it. **Secondly**, in the online service or testing phase, the systems only need to aggregate a limited number of preference vectors with regard to the query and candidate items. By getting rid of processing the whole history in this way, the systems will reduce a lot of computing consumption and query latency, compared with the query-dependent models. Furthermore, **we preliminarily leverage the category information to construct and aggregate multiple interests of users**. Existing approaches modelling multiple interests [9, 18] in recommendations mainly use the representations of products themselves to cluster user historical interests into several vectors, without any external attribute information. Different from them, in our model, we capture the category embeddings for queries and items and use them to help separate user interests. Besides, we associate each preference vector with a category indication representation to indicate the distribution of categories this preference focuses on. For example, a user's first preference may focus on cellphones and tablets while the second one aims at computers. Embracing these category information, the aggregation of multiple interests will have external messages as references. We can also determine which interests stimulate a certain purchase more accurately through these category indications.

More specifically, we propose the **C**ategory-**A**ware **M**ulti-**I**nterest model (**CAMI**) to construct multiple user profiles. The model is derived from the knowledge graph embedding method as it is shown to be effective under personalized product search situations in [4] and can automatically learn the representations for users and products. To disentangle user diverse interests in different categories, we substitute the single embedding vector for a user with K preference vectors and build corresponding category indication embeddings for item, query, and each user preference. We need to confirm that both preference and indication embeddings for a certain user are created based on her own histories and different users can have different category distributions. Furthermore, to avoid the homogenization between user interests, we add a regularization term measuring the distances between the category indication embeddings for user interests into the final loss function. This redundancy regularization term forces the model to separate interests and avoids it degrading to the single embedding method. In the running time, our model scores items through the matching scores between the product embeddings and the comprehensive user embedding based on multiple preference vectors. The combining weights for multiple preferences are distributed through the attention mechanism based on the category embeddings of users, queries and items. Experimental results show that the proposed methods can significantly outperform existing approaches and can process queries more efficiently compared with query-dependent approaches. The main contribution of our work is three-fold:

(1) We learn multiple interest representations to encode diverse preferences of users in the offline stage for personalized product search. By this means, our method can reflect user's various interests in different categories.

(2) We leverage the category information to aggregate the multiple interests for users. Instead of using product representations themselves to integrate user's multiple interests, our method has category indications as references and is more accurate.

(3) We introduce homogenization regularization into the final optimization to avoid the redundancy between diverse interests. The regularization term aims to maximize the margin between the category indications, forcing the model to separate user interests.

2 RELATED WORK

2.1 Personalized Product Search

As we discussed in Section 1, we can divide approaches in personalized product search into two categories according to whether the profiles are independent on dependent to queries. Query-independent approaches build user representations in the offline training stage, and these vectors keep static regardless of the current query. Instead, query-dependent approaches construct dynamic profiles according to search context in running time.

Query-independent approaches. Existing approaches in personalized product search mainly fall into this category. Ai [3] proposed HEM model constructing user and product representations by their related words using the latent space model. DREM [4] applied knowledge graph embedding method leveraging meta information about products including brands, categories into the model. Liu [21] proposed GraphSRRL to excavate "conjunctive graph pattern" in the user-item graphs to learn user embeddings. There also exist probabilistic models [28] in the product search area. All of these models can calculate user embeddings and store them in advance, so it is convenient and efficient to apply in real systems. However, as we discussed before, these approaches embedded users into single vectors, which will messily integrate different interests.

Query-dependent approaches. To capture user interests dependent on the current query or candidate item, there are some approaches building user profiles in the running time. ZAM and AEM [2] model adopted query attention to extract user interests from their historical interacted items. TEM [5] further improved them by replacing query attention with Transformer structure [26]. RTM [6] also used Transformer to integrate the words in reviews related to the user and the candidate product and the current query. ALSTP [12] used hierarchical RNN to model user's long-term preference and short-term preference and calculate their matching scores with products respectively. These models construct user profiles in the running time so that they can capture dynamic user interests. However, they cannot store user embedding as query-independent approaches and need to dynamically build them, which may bring additional computing costs (For example, these approaches need to access the whole history of the current user). As a result, query-dependent approaches are too inefficient to apply in real systems.

2.2 Multi-Interest Networks

How to model user multiple interests has been a challenge in recommendation systems for years. Li [17] adopted capsule network and regarded the high-level capsules as users' interests. Cen applied self-attention network [9] to disentangle their different preference.

Liu [22] proposed a novel network "archive network" which initialized different heads in attention with the orthogonal basis of the embedding space. However, these approaches mainly aggregate user different profiles without additional knowledge, which leads to inaccuracy in integrating preferences. There exist a few approaches [8] trying to leverage category information into models but it explicitly builds one interest vector for one category, which cannot apply if the amounts of categories are large.

2.3 Knowledge Embedding

Knowledge embedding is proposed to model multi-relational data by capturing latent embedding for entities and relations. There are several ways to conduct knowledge embedding including matrix factorization [15, 19, 24, 25] and bayesian framework [23, 33]. Recently, some approaches [7, 20, 27] leveraging graph structure have been proposed. These approaches model the relations as the edges and entities as the nodes to construct a knowledge graph. In our model, we adopt the translation-based method in knowledge graph approaches and generative method to capture embedding and maximize the probability of existing relations.

3 THE CAMI MODEL

3.1 Overview

In personalized product search, there are sufficient and complicated relationships between users, products, and product attributes which are used for modeling user and product information. A user can interact with many products and a product can have several reviews. To model these relationships, following DREM [4], we construct a knowledge graph (KG) (shown in the left part in Figure 1) because it is shown to be effective for personalized product search. In this graph, each node represents an entity whose type is user, item (product), or attribute, and each edge represents an observed relationship between entities. Each edge is also labelled with a particular symbol representing its type. For example, as shown in Figure 1, product "Surface Pro 7" has a relation "*Is_brand*" to the "Microsoft". In this way, the personalized product search task can be treated as a relation prediction task in KG. That is, given a query q issued by a user u, we need to predict which product i can be the tail of the edge with the label "search&purchase" of context q starting from u.

To solve this problem, we apply typical translation-based embedding methods to embed both entity nodes and relations into continuous latent space, and regard all relations as the translations from one entity to another. For a triplet $\langle x, r, y\rangle$ in KG, translation methods assume that y should be equal to the translation entity $T(x, r)$, where T is the designed translation operator. Based on this assumption, we can optimize the embedding $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y}$ for x, r, y by maximizing the similarity $S(y|x, r)$ between y and $T(x, r)$. Specifically, our task is to grade $S(i|u, q)$ to rank items, so $S(i|u, q)$ can also be regarded as the score function for item i under the query q issued by user u under personalized product search situation.

However, the vanilla translation method, which embeds each entity and relation into one vector, is ineffective for some relations. In the KG built for personalized product search, we can categorize relations into static ones and dynamic ones. Static relations refer to relations irrelevant to search context such as an item belonging to a brand. These static relations are similar to the relations in other KG and can be directly optimized in the original single embedding way. Dynamic ones refer to the search&purchase relations between users and items whose content should be determined by the issued query. We argue that existing translation methods cannot fit in this kind of relations. Firstly, the relation embedding should be context-aware, which means it should be a dynamic embedding $\boldsymbol{r}(\boldsymbol{q})$ according to the search context instead of a static relation representation $\boldsymbol{r}$. Secondly, as we discussed in Section 1, a user u can have multiple interests in different categories, and the purchase of item i through query q may be only affected by some of them. If we still use the vanilla way to directly maximize the single vector similarity between i and $T(u, q)$, all user interests will be forced to surround i, where the irrelevant preferences are not helpful and may introduce noise in the optimization process.

To overcome the above problem, we embed user profile into K preference vectors $[\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots, \boldsymbol{u_K}]$ instead of a single vector. To capture which interests cause a certain search, we also construct the corresponding category representation for K user profiles ($[\boldsymbol{c_{u_1}}, \boldsymbol{c_{u_2}}, \cdots, \boldsymbol{c_{u_K}}]$), items ($\boldsymbol{c_i}$) and queries ($\boldsymbol{c_q}$). We need to argue that these user-specific category representations $[\boldsymbol{c_{u_1}}, \boldsymbol{c_{u_2}}, \cdots, \boldsymbol{c_{u_K}}]$ indicating category distributions of corresponding interests are virtual and automatically learnt from scratch using relations in KG. They may not direct to an exact category in dataset but are mixture of them. Furthermore, different users can have their different distributions of interests as the categories of products they purchased are different. For the score or similarity function $S(i|u, q)$ of dynamic relations, we aggregate the multiple preferences of users by the category attention with query and item to calculate it. Furthermore, we also introduce homogenization regularization to avoid the redundancy between different interest vectors. The overall structure of the CAMI is shown in Figure 1.

3.2 Static Relation

Static relations refer to the relations which are independent to the search context. The representation of relation in this categorization holds static in spite of the head and tail entity and the search context. As it is consistent with the original KG, we use the vanilla translation method to optimize these relations. For a certain static relation $\langle x, r, y\rangle$, we use dot product to measure the similarity $S(y|x, r)$ between the tail entity y and translation entity $T(x, r) = x + r$.

$$S(y|x, r) = \boldsymbol{y}^\top \cdot (\boldsymbol{x} + \boldsymbol{r}), \tag{1}$$

where $\boldsymbol{x}, \boldsymbol{r}, \boldsymbol{y} \in \mathbb{R}^d$ denote the embedding for x, r, y with d dimensions. Following embedding-based generative model such as [1, 3, 4], we can estimate the probability of a relation and maximize them of observed ones in KG to optimize the embedding representations. In our model, the probability of relation $\langle x, r, y\rangle$ is calculated by softmax function:

$$P(y|x, r) = \frac{\exp(S(y|x, r))}{\sum_{y' \in Y} \exp(S(y'|x, r))},$$

where Y denotes the set of all entities with the same type as y. However, as the size of Y can be very large, the computation cost is too huge to practice. To estimate it more effectively, following [2–4],

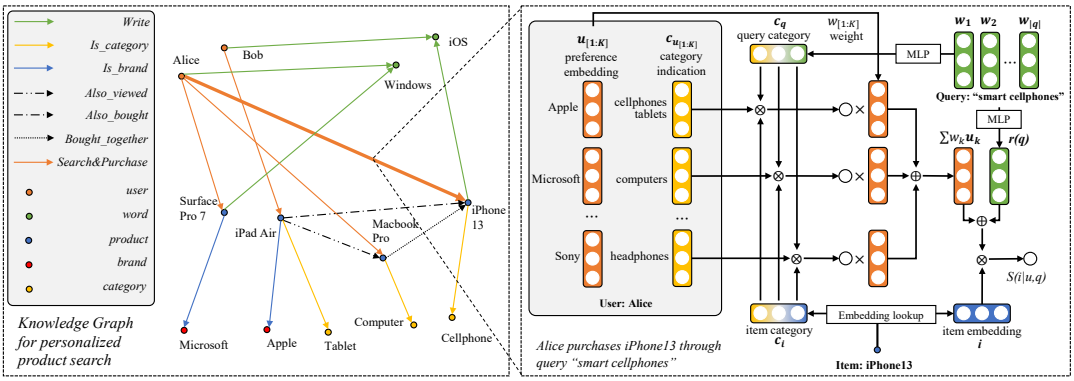


Figure 1: The main structure of Category-Aware Multi-Interest model CAMI for personalized product search. We use brand to represent user preference in Alice's user representation for convenience.

we adopt negative sampling to approximate the log probability:

$$\log P(y|x,r) = \log \sigma(S(y|x,r)) + n \cdot \mathbb{E}_{y' \sim Y}[\log \sigma(-S(y'|x,r))], \quad (2)$$

where n is the size of negative samples.

Finally, we optimize our model through maximizing the log likelihood of all the ground-truth static triplets $\langle x, r, y \rangle$ in KG:

$$\begin{aligned} \mathcal{L}_S &= -\sum_{\langle x,r,y \rangle \in \mathcal{S}_{\langle x,r,y \rangle}} \log P(y|x,r) \\ &= -\sum_{\langle x,r,y \rangle \in \mathcal{S}_{\langle x,r,y \rangle}} \log \sigma(S(y|x,r)) + n \cdot \mathbb{E}_{y' \sim Y}[\log \sigma(-S(y'|x,r))]. \end{aligned} \quad (3)$$

3.3 Dynamic Relation

As we discussed before, the original knowledge embedding strategy cannot fit the dynamic relations in the personalized product search KG. To adapt and improve the simple translation method, we will introduce how to leverage multiple vectors encoding user interests into the knowledge graph, exactly, the *search&purchase* relations in the following part.

First, different from the static relation such as *bought_together*, the relation vector in dynamic relation should be determined by the search context. For example, the relation vector for query "cellphones" should be different from the query "computers". In order to measure the meaning of a certain relation, we need to calculate the relation vector $\boldsymbol{r(q)}$ in the online stage according to the content of the query q. Following previous work [2–4], we use the non-linear projection of the average word embedding to represent it:

$$\boldsymbol{r(q)} = f(q) = \tanh(W \cdot \frac{\sum_{w \in q} \boldsymbol{w}}{|q|} + b), \quad (4)$$

where $W \in \mathbb{R}^{d \times d}, b \in \mathbb{R}^d$ are trainable parameters, $\boldsymbol{w}$ is the corresponding embedding of word w in KG.

Second, as we introduced in Section 1, we enhance the category information from the metadata to help capture user interests. To aggregate and separate the multiple preferences of users, we need to specify the category representation of current query (i.e. $\boldsymbol{c_q}$) and scored item (i.e. $\boldsymbol{c_i}$). Only after capturing them, we can determine which categories this purchase behaviour focuses on. For $\boldsymbol{c_q}$, we directly use the same representation $\boldsymbol{r(q)}$ in Eq. (4). For $\boldsymbol{c_i}$, we can use the entity embeddings of corresponding categories in KG, but these embeddings are automatically learnt from the observed relations without any semantic information. By this means, there will exist representation mismatches between the word-based category embedding for q and the relation-based category embedding for i. To solve this mismatch problem, we choose to use the words in item i's category label c_i to calculate $\boldsymbol{c_i}$ in the same way in Eq. (4):

$$\boldsymbol{c_q} = \boldsymbol{r(q)} \qquad \boldsymbol{c_i} = f(c_i) = \tanh(W \cdot \frac{\sum_{w \in c_i} \boldsymbol{w}}{|c_i|} + b), \quad (5)$$

Please note that after the training process, we can store $\boldsymbol{c_i}$ for each item and don't need to calculate repeatedly while testing.

So far, we have already captured the content and category information of query and item. As we introduced before, we propose to model a user u's interests in K vectors $([\boldsymbol{u_1}, \boldsymbol{u_2}, \cdots, \boldsymbol{u_K}]$, where each interest has a corresponding category indication vector $([\boldsymbol{c_{u_1}}, \boldsymbol{c_{u_2}}, \cdots, \boldsymbol{c_{u_K}}])$. We will introduce the score function $S(i|u,q)$ for triplet $\langle u, q, i \rangle$ based on the multiple preference vectors in the following part.

For a certain triplet $\langle u, q, i \rangle$, we can score the item i, or in other words, measure the similarity between i and the translation entity, using following equation:

$$\begin{aligned} S(i|u,q) &= \lambda_u \cdot (\boldsymbol{i}^\top \cdot (\sum_k w_k \cdot \boldsymbol{u_k} + \boldsymbol{r(q)})) + (1 - \lambda_u) \cdot p_i \\ &= \lambda_u \cdot \sum_k w_k \cdot (\boldsymbol{i}^\top \cdot (\boldsymbol{u_k} + \boldsymbol{r(q)})) + (1 - \lambda_u) \cdot p_i, \end{aligned} \quad (6)$$

where $\boldsymbol{u_k}, \boldsymbol{i}$ is the embedding for user u's k-th preference and item, $\boldsymbol{r(q)}$ is the relation vector calculated in Eq. (4). w_k denotes the combining weight for the preference u_k. The combining weight can also infer which interests affect this purchase. $\boldsymbol{u} = \sum_k w_k \boldsymbol{u_k}$ is the weighted combination interest of K preferences and we use it to calculate the matching score with item i. p_i denotes the trainable popularity of product i, which can be considered as global interests. λ_u denotes the trainable user-specific combining weight between personal and global interests. We introduce these two variables into model to imitate what users consider before purchases. For example, some people tend to purchase the products with high sales while others mainly consider their own interests. Furthermore, this

equation can also be regarded as a combination of K matching scores between the product and preference vectors of user as the second line. From this perspective, our model actually represents user as K profiles instead of one single vector as it in existing methods.

The key point of the CAMI is the combining weight w_k. Intuitively, it should be determined by the query category embedding c_q and the indication embedding c_{u_k} of preference u_k. The closer these two representations are, the chances that this query is related to this interest are bigger. So we need to measure the similarity between c_{u_k} and c_q. Furthermore, the model should also take the category information of products into consideration. The category of the purchased product i should align with the categories of the interests related to the query q. Following the above discussions, we implement w_k as follows:

$$w_k = \frac{\exp((\boldsymbol{c}_{\boldsymbol{u_k}}^\top \cdot \boldsymbol{c}_{\boldsymbol{i}}) \cdot (\boldsymbol{c}_{\boldsymbol{u_k}}^\top \cdot \boldsymbol{c}_{\boldsymbol{q}})/\tau)}{\sum_{j=1}^{K} \exp((\boldsymbol{c}_{\boldsymbol{u_j}}^\top \cdot \boldsymbol{c}_{\boldsymbol{i}}) \cdot (\boldsymbol{c}_{\boldsymbol{u_j}}^\top \cdot \boldsymbol{c}_{\boldsymbol{q}})/\tau)}, \tag{7}$$

where τ infers the temperature for softmax function. For τ, we explore two different implementations in our experiments. In the first manner, we consider τ as a hyper-parameter dynamically changing in the training process. Following [10], τ should gradually decrease and make the distribution after softmax become sharper and sharper. In the second way, we regard τ as trainable parameters and should be various among users (so it should be written as $\tau(u)$ in this way). The reason is that different users can have different purchasing strategies. Some users may buy products just because the products match one of their interests, while others make a purchase only after deep consideration and comparison.

After scoring the tail entity of each *search&purchase* relation, we use the softmax function to estimate the probability of the observed triplet $\langle u, q, i\rangle$:

$$P(i|u,q) = \frac{\exp(S(i|u,q))}{\sum_{i' \in I} \exp(S(i'|u,q))}, \tag{8}$$

where I is the set of all items. Then, similar to the static relations and existing methods [2–4], we adopt the negative sampling strategy to approximate the probability:

$$\log P(i|u,q) = \log \sigma(S(i|u,q)) + n \cdot \mathbb{E}_{i' \sim I}[\log \sigma(-S(i'|u,q))]. \tag{9}$$

Finally, similar to the static part again, the loss function for the dynamic part is to maximize the log likelihood of the observed triplets in the dynamic relation set $\mathcal{D}_{\langle u,q,i\rangle}$:

$$\begin{aligned}\mathcal{L}_{\mathcal{D}} &= -\sum_{\langle u,q,i\rangle \in \mathcal{D}_{\langle u,q,i\rangle}} \log P(i|u,q) \\ &= -\sum_{\langle u,q,i\rangle \in \mathcal{D}_{\langle u,q,i\rangle}} \log \sigma(S(i|u,q)) + n \cdot \mathbb{E}_{i' \sim I}[\log \sigma(-S(i'|u,q))].\end{aligned} \tag{10}$$

Discussion. In our model, we do not force the whole of user interests needs to align with the product i ($u + q \sim i$), which may introduce noises to irrelevant preferences. Otherwise, we only require some of her preferences u_k whose weight w_k is high need to be near to i ($u_k + q \sim i$). The choices of these interests are decided by the similarities of category indication representations among the user interests, the current query, and the scored item.

3.4 Homogenization and Redundancy

In CAMI, we construct different representations for different preferences of users. However, these representations can be very similar in the training process because we do not explicitly measure their diversity in our model. In this way, though we build multiple vectors and indicators for users, the diverse preferences may still integrate together in each vector. Thus the model will fail to separate multiple interests in different categories and degrade to the original single interest model. To prevent this homogenization phenomenon, we add a disagreement loss into our model following [9]. Existing approaches mainly directly maximize the distances between the preference vectors. This may introduce some bias into the model because some users can be passionate fans of a company and only buy the products belonging to it. If we still require the model to distinguish between these similar preferences, the model will become inaccurate. Thus, we maximize the margins between the indication representations of interests instead, because most users will buy products belonging to different categories. By this means, we can explicitly measure the diversity between user interests in our model. The loss for homogenization is as follows:

$$\mathcal{L}_{\mathcal{H}} = \sum_{u} \sum_{1 \le i < j \le K} \frac{|\boldsymbol{c}_{\boldsymbol{u_i}}^\top \cdot \boldsymbol{c}_{\boldsymbol{u_j}}|}{\left\|\boldsymbol{c}_{\boldsymbol{u_i}}\right\| \cdot \left\|\boldsymbol{c}_{\boldsymbol{u_j}}\right\|}, \tag{11}$$

where we use the cosine similarity to represent the distance between category indication embeddings because it fits the dot product method in our model.

Finally, we integrate all the losses in Eq. (3), Eq. (10) and Eq. (11) into a unified one via linear combination:

$$\mathcal{L} = (1-\lambda)\mathcal{L}_{\mathcal{S}} + \lambda \mathcal{L}_{\mathcal{D}} + \mu \mathcal{L}_{\mathcal{H}}, \tag{12}$$

where λ and μ are hyper-parameters.

3.5 Online Service

In the online running time, when a user u issues a query q, the model calculates the score $S(i|u,q)$ for each candidate item i and ranks them in decreasing order. To optimize the computation cost, we implement $S(i|u,q)$ as the second line in Eq. (6), which first calculates K matching scores and then aggregates them. As a result, this method is only constant (K) times slower than the query-independent approaches. This coefficient can be accelerated by GPU parallel computing. Furthermore, as we can store K user preferences and indication embeddings contiguously, we can capture them through one reading operation. However, in the query-dependent approaches, user historical interacted items are stored separately and the model needs to obtain them through multiple reading or selection operations, which takes lots of I/O time. As a result, our method is more efficient than approaches building dynamic profiles while online serving. Besides, as it can calculate and store K user profiles in advance, CAMI model can also perform the first-stage retrieval task through approximate nearest neighbour technique [16]. We can regard the union set of nearest neighbourhoods of each preference as the retrieved candidates. In contrast, query-dependent approaches need to access each interacted item to aggregate user interest, which makes them hard to apply in the first-stage retrieval.

4 EXPERIMENTAL SETUP

4.1 Datasets

As there are no large-scale datasets in the personalized product search area, we use Amazon dataset[1] as our experiment corpus, consistent with existing approaches [2–4]. To construct reliable user profiles, we use the 5-core data, the dense sub-dataset of the whole corpus, where each user and each item has at least 5 associated reviews. The experiments are conducted on four categories, *Clothing, Shoes & Jewelry, Toys & Games, CDs & Vinyl, Electronics*. These datasets both contain several categories so that users may have different interests. The first two categories are in a small scale, we use them to verify the effectiveness of the proposed CAMI model in limited data.

4.2 Query Construction

As the Amazon dataset is collected for the recommendation originally, it only consists of information about users and items. To construct queries from the dataset, we apply a two-step strategy following existing approaches [3, 14]. This strategy utilizes the hierarchical category marks of products to label queries. First, we extract the multi-level class information from the metadata to form category lists for products. Then, we remove the duplicated words and stopwords in one category list and concatenate terms in it to a string. This string is considered as a query submitted by users which leads to the purchase of the corresponding product.

4.3 Evaluation

To evaluate the performance of models, we split each dataset into a training set and a test set. For each user, we select 70% purchase behaviour (each purchase corresponds to a review) to form the training set. The rest purchase can be regarded as the judgements in the testing phase. After constructing the labels in the test set, we need to build the query set. First, we randomly select 30% queries in the query corpus as the initial test query set. Then, if all queries of an item are in the test query set, we put a random one of them back to the training query set, which guarantees that each item has at least one query in the training data. In the evaluation process, the personalized product search is a ranking problem basically, so we use the metrics of ranking problems to evaluate the models, including MRR, NDCG, MAP. We also conduct two-tailed t-test to measure the significant differences between results.

4.4 Entities and Relationships

In this work, we introduce five entities and seven relationships into the knowledge graph to calculate the representations for users and items following [4]. The entities include users, items, words, brands, and categories. As the category information for products is hierarchical, we split the m-level hierarchy into m independent entities. The relationships we used in our model include:

Write: Word w is occurred under the review for item i ($i \to w$) or is written by user u ($u \to w$). For $u \to w$ relations, we average K interests to represent the head entity in the score function in Eq. (1).

Is_brand: Item i belongs to brand b ($i \to b$).

Is_category: Item i belongs to category c ($i \to c$).

Also_bought: Item i_1 and i_2 have been purchased by the same user(s) ($i_1 \to i_2$).

Bought_together: Item i_1 and i_2 have been purchased under the single transaction ($i_1 \to i_2$).

Also_viewed: Item i_1 and i_2 have been viewed by the same user(s) ($i_1 \to i_2$).

Search&Purchase: User u issues a query q and then purchases an item i ($u + q \to i$).

4.5 Baselines

LSE: LSE [14] is a non-personalized product search model. It learns the vectors of words and items via generative model and calculates the scores using the similarity between queries and items.

HEM: HEM [3] is a personalized product search model. It applies latent space model and argues that each word in reviews is generated by the latent vector of the corresponding user and item.

AEM: AEM [2] is an attention-based personalized product search model and constructs dynamic user profiles by aggregating users' historical interacted items by the attention weight with query.

ZAM: ZAM [2] is the extension version of AEM. It concatenates the purchased item list with a zero vector. The model can adjust the personalization extent by attending the query to the zero vector.

TEM: TEM [5] replaces the attention network in ZAM with transformer [26] structure to construct query-dependent user representations.

DREM, **DREM-m**: DREM [4] is a KG-based personalized product search model. It embeds each entity into one single vector and uses generative model to calculate the embeddings. Furthermore, to verify the improvement of the proposed model is not caused by the enlargement of embedding size d, we multiple the embedding size with K in DREM and denote this model as DREM-m.

CAMI-p, **CAMI-h**, **CAMI-r**: The proposed category-aware multi-interest model. CAMI-p refers to the implementation regarding τ as parameters while CAMI-h regards τ as a hyper-parameter. For CAMI-r model, we remove relationships except *Write* and *Search&Purchase* from the KG in CAMI-h. We design this model to align with the configuration of query-dependent baselines, which only use the review information in the dataset.

4.6 Parameter Settings

For CAMI, we tune the hyper-parameters including λ and μ from 0.01 to 0.1 and 0.1 to 0.9 respectively to obtain the best performance. The embedding size d for all models is 200. We set the multiple interest's size $K = 4$ as it gets the best performance in our experiments [2] , which is consistent with [9, 17]. we use the last and smallest caetgory in the category hierarchy as the category c_i for item i. The negative samples for all the relations is set to 5. For the temperature τ in the score function Eq. (7) in CAMI-h and CAMI-r, we use linear annealing where $\tau = \tau_{\max} - \frac{t}{T} * (\tau_{\max} - \tau_{\min})$. We set $\tau_{\max} = 3.0$ and $\tau_{\min} = 0.05$. For CAMI-p, we set the initial value $\tau_{\text{init}} = 1.0$ for each user. To avoid overfitting, we add L2 regularizations to the representation of all entities and set the coefficient γ as 0.005. Our codes can be found at **https://github.com/rucliujn/CAMI**.

[1] http://jmcauley.ucsd.edu/data/amazon/

[2] In our experiments, the performance maintain stable regardless of K while $K \geq 4$.

Table 1: Overall performances of models. "†" indicates the model outperforms all baselines significantly with paired t-test at $p < 0.05$ level. "‡" indicates the model outperforms all baselines except DREM and DREM-m significantly with paired t-test at $p < 0.05$ level. The best results are shown in bold.

Dataset		*Clothing, Shoes & Jewelry*			*Toys & Games*			*CDs & Vinyl*			*Electronics*		
Model		MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP
Non-personalized	LSE	0.023	0.024	0.022	0.079	0.080	0.079	0.003	0.003	0.003	0.181	0.234	0.180
personalized	*Query-dependent methods*												
	AEM	0.022	0.026	0.022	0.063	0.069	0.062	0.038	0.037	0.032	0.265	0.290	0.265
	ZAM	0.022	0.024	0.022	0.088	0.128	0.087	0.035	0.035	0.030	0.287	0.314	0.286
	TEM	0.023	0.027	0.023	0.129	0.186	0.127	0.036	0.038	0.033	0.196	0.222	0.196
	Query-independent methods												
	HEM	0.021	0.022	0.021	0.088	0.098	0.085	0.019	0.018	0.015	0.263	0.283	0.262
	DREM	0.078	0.093	0.077	0.210	0.274	0.206	0.078	0.079	0.067	0.330	0.358	0.329
	DREM-m	0.081	0.094	0.081	0.249	0.298	0.246	0.083	0.086	0.073	0.352	0.376	0.351
	Our methods												
	CAMI-r	$0.037^{\ddagger}$	$0.045^{\ddagger}$	$0.037^{\ddagger}$	$0.164^{\ddagger}$	0.195	$0.159^{\ddagger}$	0.037	0.036	0.032	$0.310^{\ddagger}$	$0.334^{\ddagger}$	$0.310^{\ddagger}$
	CAMI-p	$0.089^{\dagger}$	$0.104^{\dagger}$	$0.087^{\dagger}$	$0.239^{\ddagger}$	$0.276^{\ddagger}$	$0.236^{\ddagger}$	$0.087^{\ddagger}$	$0.092^{\ddagger}$	$0.077^{\ddagger}$	$0.374^{\dagger}$	$0.405^{\dagger}$	$0.374^{\dagger}$
	CAMI-h	$\mathbf{0.093}^{\dagger}$	$\mathbf{0.109}^{\dagger}$	$\mathbf{0.092}^{\dagger}$	$\mathbf{0.280}^{\dagger}$	$\mathbf{0.316}^{\dagger}$	$\mathbf{0.277}^{\dagger}$	$\mathbf{0.090}^{\dagger}$	$\mathbf{0.095}^{\dagger}$	$\mathbf{0.080}^{\dagger}$	$\mathbf{0.383}^{\dagger}$	$\mathbf{0.416}^{\dagger}$	$\mathbf{0.382}^{\dagger}$

5 EXPERIMENTAL RESULTS

5.1 Overall Results

The whole results are shown in Table 1 and we can find that:

(1) Query-dependent methods outperform query-independent methods under the same settings. Besides the DREM model utilizing additional meta information in the dataset, approaches building dynamic profiles including AEM, ZAM, TEM outperform the query-independent approach HEM in most datasets. This result infers that it is necessary and effective to capture interests relevant to the current query rather than interests containing all histories.

(2) Compared with query-independent methods, our model achieves significant improvements as our model can disentangle diverse interests for users. The relative improvement in terms of NDCG@10 over the DREM model is 15.9%, 14.5%, 20.2%, 16.2% respectively in four datasets. We can observe that the improvements in larger datasets (*CDs & Vinyl* and *Electronics*) are bigger. The reason may be that our model can disentangle user interests more precisely when the data is rich. Furthermore, we can find that DREM-m only obtains a marginal improvement compared with DREM, which infers that simply enlarging embedding size does little to the improvement of performance and our model gets improvement mainly through separating multiple preferences into different vectors for users.

(3) Our model outperforms all query-dependent methods as we leverage the meta information in the knowledge graph, especially the category for interests. The reason why our model is better than these methods may be the additional relations we modeled in KG. This may inspire us to try to integrate extra messages into query-dependent approaches to improve performances. Furthermore, though only using the review data, the CAMI-r model can still achieve comparable or better performance over TEM. The reason may be that we explicitly capture the category information of query and item to help aggregate user interests.

(4) CAMI-h outperforms CAMI-p in all datasets in our experiments. The reason may be tuning τ for each user is difficult for the KG embedding model. To verify this hypothesis, we analyze the values of τ after training for CAMI-h and find that most of them gather into the initial value, which is unexplainable. To overcome this problem, we may explore a more explicit way to model user's different strategies in combining interests in future work.

Table 2: Performance of ablation models in *Electronics*

Model	MRR	NDCG	MAP
CAMI-h	**0.383**	**0.416**	**0.382**
w/o hom	0.368 (-3.92%)	0.400 (-3.85%)	0.367 (-3.93%)
w/o qattn	0.353 (-7.83%)	0.380 (-8.65%)	0.353 (-7.59%)
w/o cate	0.347 (-9.40%)	0.376 (-9.62%)	0.346 (-9.42%)
w/o cate + hom	0.363 (-5.22%)	0.392 (-5.88%)	0.362 (-5.24%)
hard selection	0.361 (-5.74%)	0.390 (-6.25%)	0.360 (-5.76%)
cate entity emb	0.365 (-4.70%)	0.394 (-5.29%)	0.364 (-4.71%)

5.2 Ablation Study

In this section, we conduct an ablation study of the main parts in CAMI. These models are shown as follows:

w/o. hom. We remove the homogenization regularization $\mathcal{L}_{\mathcal{H}}$ from the unified loss function in Eq. (12).

w/o. qattn. We remove the query attention term $\boldsymbol{c}_{\boldsymbol{u_k}}^{\top} \cdot \boldsymbol{c_q}$ from the weight calculation in Eq. (7).

w/o. cate. We remove the category indication embeddings for users, items, and queries from our model. The combining weight in this model is calculated by the dot similarity between $\boldsymbol{u_k}$ and $\boldsymbol{i}$.

w/o. cate + hom. We add the redundancy regularization modelling the distances between user preference embeddings $\boldsymbol{u_k}$ into the **w/o. cate.** model.

hard selection. We use the max selection method to aggregate user's multiple interests instead of the softmax function in 6. The score function for dynamic relation in this model is as follows:

$$S(i|u,q) = \lambda_u \cdot \boldsymbol{i}^{\top} \cdot (\boldsymbol{u}_{\boldsymbol{k}_{\text{opt}}} + \boldsymbol{r}(\boldsymbol{q})) + (1-\lambda_u) \cdot p_i$$

$$k_{\text{opt}} = \arg\max_k (\boldsymbol{c}_{\boldsymbol{u_k}}^{\top} \cdot \boldsymbol{c_i}) \cdot (\boldsymbol{c}_{\boldsymbol{u_k}}^{\top} \cdot \boldsymbol{c_q}).$$

cate entity emb. We replace the $\boldsymbol{c_i} = f(c_i)$ with the entity embedding of the smallest category of item i.

The results of the ablation study are shown in Table 2. We can find all these ablation models underperform CAMI. The decrease of removing the homogenization regularization infers the redundancy between interests does harm to the performance. However, as we also explicitly measure the category indication for each preference to help separate user interests, the decline is subtle. The models **w/o. cate.** and **w/o. qattn.** without modelling the category information both cause the decline of performance. This result reveals that leveraging additional attributes such as categories can help the disentanglement of user diverse interests. The low result of **w/o. cate.** infers that directly using the embedding representations of products and users leads to inaccuracy. We need to capture other information to aggregate interests. However, the improvement of **w/o. cate + hom.** over **w/o. cate.** shows that it is useful to add disagreement loss to help separate user interests if we have no external information such as categories. The result of **w/o. qattn.** shows that it is beneficial to enhance query information into the score function. The **hard selection.** method only calculates the item's matching score with the closest user preference. The performance decrease compared with CAMI may be due to that some purchases are not caused by only one interest but some of them. However, It still outperforms the DREM because the model can separate interests and keep them independent of each other while training and test. The result of **cate entity emb.** method illustrates that the mismatch between relation-based and word-based representations does cause the decline of performances.

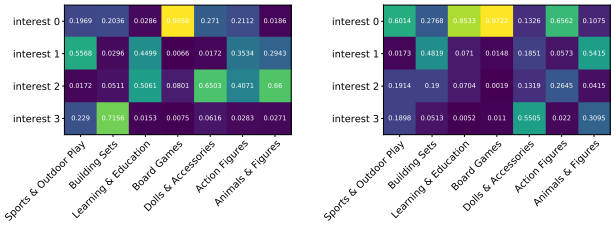
5.3 Case Study

Figure 2 shows the coefficients between the category indication representations of two user interests and several category embeddings for items in dataset *Toys & Games*. The coefficients are calculated in the same way as the weight in Eq. (7). The lighter a grid is, the corresponding row interest and the corresponding column category are more related. From the figure, we can infer that these diverse preference vectors model preferences in different categories. For example, user A's first interest mainly focuses on the "Board Games" category and her fourth interest mainly aims at the "Building Sets". Our model separates them into different vectors to avoid disorganized integration in previous models.

Besides, we can also observe that the disentangled interests are different between users. The fourth interest of user A focuses on the "Building Sets" and that of user B embeds the preferences on "Dolls & Accessories". We can also find that the interest of "Building Sets" doesn't fall into any of user B's preferences exactly, which may indicate that neither of these multiple vectors models it. It is reasonable because the interacted items are diverse between users and user B may never interact with items belonging to the "Building Sets" category, so the constructed user profiles should be different.

5.4 Query Latency Analysis

In this section, we compare the average running time on each query to represent query latency between DREM, TEM, CAMI. DREM and TEM are representatives of query-independent and query-dependent approaches respectively. The results are shown in Table. 3. We can observe that TEM is much slower than DREM since it needs to process the whole user history. However, the query latency of CAMI is very close to the single embedding model DREM because it only needs to aggregate K user preferences and this can be accelerated by the GPU parallel computing. The results accord with the qualitative analysis in section 3.5 and show that our model is more efficient than query-dependent approaches.



(a) User A (b) User B

Figure 2: The coefficients between the category indications of two users and several categories in *Toys & Games*. We conduct softmax function on the row direction.

Table 3: Average test time per query in millisecond(ms)

Model	*Clothing, Shoes & Jewelry*	*Toys & Games*	*CDs & Vinyl*	*Electronics*
DREM	19.7ms	14.1ms	59.8ms	59.7ms
TEM	53.5ms	31.9ms	239.4ms	183.2ms
CAMI	20.6ms	18.6ms	60.9ms	63.9ms

6 CONCLUSION

In this work, we propose a category-aware multi-interest model CAMI for personalized product search to represent user diverse preferences in different categories by learning entity and relation embedding in KG. In our model, we replace the single vector for users in original KG with K multiple vectors with their corresponding category indication vectors. To score items, we aggregate user multiple preferences through category-attention considering query and items into a combined representation. Furthermore, we introduce a homogenization regularization term to avoid redundancy between the multiple interests. Experimental results show that our model can significantly outperform all existing models and is more efficient than query-dependent approaches. In future work, we plan to further explore the implementation of different user strategies and to utilize the relations in datasets more explicitly.

ACKNOWLEDGMENTS

Zhicheng Dou is the corresponding author. This work was supported by National Natural Science Foundation of China No. 61872370 and No. 61832017, Beijing Outstanding Young Scientist Program NO. BJJWZYJH012019100020098, China Unicom Innovation Ecological Cooperation Plan, and Intelligent Social Governance Platform, Major Innovation & Planning Interdisciplinary Platform for the "Double-First Class" Initiative, Renmin University of China. We also acknowledge the support provided and contribution made by Public Policy and Decision-making Research Lab of RUC.

REFERENCES

[1] Qingyao Ai, Vahid Azizi, Xu Chen, and Yongfeng Zhang. 2018. Learning Heterogeneous Knowledge Base Embeddings for Explainable Recommendation. *Algorithms* 11, 9 (2018), 137. https://doi.org/10.3390/a11090137

[2] Qingyao Ai, Daniel N. Hill, S. V. N. Vishwanathan, and W. Bruce Croft. 2019. A Zero Attention Model for Personalized Product Search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 379–388. https://doi.org/10.1145/3357384.3357980

[3] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W. Bruce Croft. 2017. Learning a Hierarchical Embedding Model for Personalized Product Search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval, Shinjuku, Tokyo, Japan, August 7-11, 2017*, Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.). ACM, 645–654. https://doi.org/10.1145/3077136.3080813

[4] Qingyao Ai, Yongfeng Zhang, Keping Bi, and W. Bruce Croft. 2020. Explainable Product Search with a Dynamic Relation Embedding Model. *ACM Trans. Inf. Syst.* 38, 1 (2020), 4:1–4:29. https://doi.org/10.1145/3361738

[5] Keping Bi, Qingyao Ai, and W. Bruce Croft. 2020. A Transformer-based Embedding Model for Personalized Product Search. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval, SIGIR 2020, Virtual Event, China, July 25-30, 2020*, Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1521–1524. https://doi.org/10.1145/3397271.3401192

[6] Keping Bi, Qingyao Ai, and W. Bruce Croft. 2021. Learning a Fine-Grained Review-based Transformer Model for Personalized Product Search. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 123–132. https://doi.org/10.1145/3404835.3462911

[7] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating Embeddings for Modeling Multi-relational Data. In *Advances in Neural Information Processing Systems 26: 27th Annual Conference on Neural Information Processing Systems 2013. Proceedings of a meeting held December 5-8, 2013, Lake Tahoe, Nevada, United States*, Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.). 2787–2795.

[8] Renqin Cai, Jibang Wu, Aidan San, Chong Wang, and Hongning Wang. 2021. Category-aware Collaborative Sequential Recommendation. In *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.). ACM, 388–397.

[9] Yukuo Cen, Jianwei Zhang, Xu Zou, Chang Zhou, Hongxia Yang, and Jie Tang. 2020. Controllable Multi-Interest Framework for Recommendation. In *KDD '20: The 26th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, Virtual Event, CA, USA, August 23-27, 2020*, Rajesh Gupta, Yan Liu, Jiliang Tang, and B. Aditya Prakash (Eds.). ACM, 2942–2951.

[10] Junsu Cho, SeongKu Kang, Dongmin Hyun, and Hwanjo Yu. 2021. Unsupervised Proxy Selection for Session-based Recommender Systems. *CoRR* abs/2107.03564 (2021). arXiv:2107.03564 https://arxiv.org/abs/2107.03564

[11] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing Search Results Using Hierarchical RNN with Query-Aware Attention *(CIKM '18)*. Association for Computing Machinery, New York, NY, USA, 347–356.

[12] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan S. Kankanhalli. 2019. Attentive Long Short-Term Preference Modeling for Personalized Product Search. *ACM Trans. Inf. Syst.* 37, 2 (2019), 19:1–19:27.

[13] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Xin-Shun Xu, and Mohan S. Kankanhalli. 2018. Multi-modal Preference Modeling for Product Search. In *2018 ACM Multimedia Conference on Multimedia Conference, MM 2018, Seoul, Republic of Korea, October 22-26, 2018*, Susanne Boll, Kyoung Mu Lee, Jiebo Luo, Wenwu Zhu, Hyeran Byun, Chang Wen Chen, Rainer Lienhart, and Tao Mei (Eds.). ACM, 1865–1873. https://doi.org/10.1145/3240508.3240541

[14] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning Latent Vector Spaces for Product Search. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management, CIKM 2016, Indianapolis, IN, USA, October 24-28, 2016*, Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, and Parikshit Sondhi (Eds.). ACM, 165–174.

[15] Richard A. Harshman and Margaret E. Lundy. 1994. PARAFAC: Parallel factor analysis. *Computational Statistics & Data Analysis* 18, 1 (1994), 39–72.

[16] Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2021. Billion-Scale Similarity Search with GPUs. *IEEE Trans. Big Data* 7, 3 (2021), 535–547.

[17] Chao Li, Zhiyuan Liu, Mengmeng Wu, Yuchi Xu, Huan Zhao, Pipei Huang, Guoliang Kang, Qiwei Chen, Wei Li, and Dik Lun Lee. 2019. Multi-Interest Network with Dynamic Routing for Recommendation at Tmall. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 2615–2623. https://doi.org/10.1145/3357384.3357814

[18] Jianxun Lian, Iyad Batal, Zheng Liu, Akshay Soni, Eun Yong Kang, Yajun Wang, and Xing Xie. 2021. Multi-Interest-Aware User Modeling for Large-Scale Sequential Recommendations. *CoRR* abs/2102.09211 (2021). arXiv:2102.09211 https://arxiv.org/abs/2102.09211

[19] Dawen Liang, Jaan Altosaar, Laurent Charlin, and David M. Blei. 2016. Factorization Meets the Item Embedding: Regularizing Matrix Factorization with Item Cooccurrence. In *Proceedings of the 10th ACM Conference on Recommender Systems, Boston, MA, USA, September 15-19, 2016*, Shilad Sen, Werner Geyer, Jill Freyne, and Pablo Castells (Eds.). ACM, 59–66. https://doi.org/10.1145/2959100.2959182

[20] Yankai Lin, Zhiyuan Liu, Maosong Sun, Yang Liu, and Xuan Zhu. 2015. Learning Entity and Relation Embeddings for Knowledge Graph Completion. In *Proceedings of the Twenty-Ninth AAAI Conference on Artificial Intelligence, January 25-30, 2015, Austin, Texas, USA*, Blai Bonet and Sven Koenig (Eds.). AAAI Press, 2181–2187.

[21] Shang Liu, Wanli Gu, Gao Cong, and Fuzheng Zhang. 2020. Structural Relationship Representation Learning with Graph Embedding for Personalized Product Search. In *CIKM '20: The 29th ACM International Conference on Information and Knowledge Management, Virtual Event, Ireland, October 19-23, 2020*, Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.). ACM, 915–924. https://doi.org/10.1145/3340531.3411936

[22] Zheng Liu, Yu Xing, Fangzhao Wu, Mingxiao An, and Xing Xie. 2019. Hi-Fi Ark: Deep User Representation via High-Fidelity Archive Network. In *Proceedings of the Twenty-Eighth International Joint Conference on Artificial Intelligence, IJCAI 2019, Macao, China, August 10-16, 2019*, Sarit Kraus (Ed.). ijcai.org, 3059–3065. https://doi.org/10.24963/ijcai.2019/424

[23] Kurt T. Miller, Thomas L. Griffiths, and Michael I. Jordan. 2009. Nonparametric Latent Feature Models for Link Prediction. In *Advances in Neural Information Processing Systems 22: 23rd Annual Conference on Neural Information Processing Systems 2009. Proceedings of a meeting held 7-10 December 2009, Vancouver, British Columbia, Canada*, Yoshua Bengio, Dale Schuurmans, John D. Lafferty, Christopher K. I. Williams, and Aron Culotta (Eds.). Curran Associates, Inc., 1276–1284. https://proceedings.neurips.cc/paper/2009/hash/437d7d1d97917cd627a34a6a0fb41136-Abstract.html

[24] Maximilian Nickel, Volker Tresp, and Hans-Peter Kriegel. 2011. A Three-Way Model for Collective Learning on Multi-Relational Data. In *Proceedings of the 28th International Conference on Machine Learning, ICML 2011, Bellevue, Washington, USA, June 28 - July 2, 2011*, Lise Getoor and Tobias Scheffer (Eds.). Omnipress, 809–816. https://icml.cc/2011/papers/438_icmlpaper.pdf

[25] Ajit Paul Singh and Geoffrey J. Gordon. 2008. Relational learning via collective matrix factorization. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining, Las Vegas, Nevada, USA, August 24-27, 2008*, Ying Li, Bing Liu, and Sunita Sarawagi (Eds.). ACM, 650–658. https://doi.org/10.1145/1401890.1401969

[26] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Ł ukasz Kaiser, and Illia Polosukhin. 2017. Attention is All you Need. In *Advances in Neural Information Processing Systems*, I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett (Eds.), Vol. 30. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2017/file/3f5ee243547dee91fbd053c1c4a845aa-Paper.pdf

[27] Zhen Wang, Jianwen Zhang, Jianlin Feng, and Zheng Chen. 2014. Knowledge Graph Embedding by Translating on Hyperplanes. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence, July 27 -31, 2014, Québec City, Québec, Canada*, Carla E. Brodley and Peter Stone (Eds.). AAAI Press, 1112–1119.

[28] Teng Xiao, Jiaxin Ren, Zaiqiao Meng, Huan Sun, and Shangsong Liang. 2019. Dynamic Bayesian Metric Learning for Personalized Product Search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management, CIKM 2019, Beijing, China, November 3-7, 2019*, Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.). ACM, 1693–1702. https://doi.org/10.1145/3357384.3358057

[29] Yujia Zhou, Zhicheng Dou, Bingzheng Wei, Ruobing Xie, and Ji-Rong Wen. 2021. Group based Personalized Search by Integrating Search Behaviour and Friend Network. In *SIGIR*. ACM, 92–101.

[30] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Enhancing Re-finding Behavior with External Memories for Personalized Search. In *WSDM*. ACM, 789–797.

[31] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Encoding History with Context-Aware Representation Learning for Personalized Search. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*. Association for Computing Machinery, New York, NY, USA, 1111–1120.

[32] Yujia Zhou, Zhicheng Dou, Yutao Zhu, and Ji-Rong Wen. 2021. PSSL: Self-supervised Learning for Personalized Search with Contrastive Sampling. In *CIKM*. ACM, 2749–2758.

[33] Jun Zhu. 2012. Max-Margin Nonparametric Latent Feature Models for Link Prediction. In *Proceedings of the 29th International Conference on Machine Learning, ICML 2012, Edinburgh, Scotland, UK, June 26 - July 1, 2012*. icml.cc / Omnipress. http://icml.cc/2012/papers/374.pdf