

Enhancing Sequential Personalized Product Search with External Out-of-sequence Knowledge

JIONGNAN LIU and ZHICHENG DOU, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China

JIAN-YUN NIE, DIRO, University de Montreal, Montreal, Quebec, Canada

ZHENLIN CHEN, GUOYU TANG, and SULONG XU, JD.com Inc, Beijing, China

Ji-RONG WEN, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, Beijing, China, and Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China

A key challenge in personalized product search is to capture user's preferences. Recent work attempted to model sequences of user historical behaviors, i.e., product purchase histories, to build user profiles and to personalize results accordingly. Although these approaches have demonstrated promising retrieval performances, we notice that most of them focus solely on the intra-sequence interactions between items. However, as there is usually a small amount of historical behavior data, the user profiles learned by these approaches could be very sensitive to the noise included in it. To tackle this problem, we propose incorporating out-of-sequence external information to enhance user modeling. More specifically, we inject the external item-item relations (e.g., belonging to the same brand), and query-query relations (e.g., the semantic similarities between them), into the intra-sequence interaction to learn better user profiles. In addition, we devise two auxiliary decoders, with the historical item sequence reconstruction task and the global item similarity prediction task, to further improve the reliability of user modeling. Experimental results on two datasets from simulated and real user search logs respectively show that the proposed personalized product search method outperforms existing approaches.

CCS Concepts: • **Information systems** → **Personalization**;

Additional Key Words and Phrases: personalized product search, sequential model, graph

This work was supported in part by the Natural Science Foundation of China No. 62272467, and in part by the Outstanding Innovative Talents Cultivation Funded Programs 2024 of Renmin University of China.

Authors' Contact Information: Jiongnan Liu, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; e-mail: liujn@ruc.edu.cn; Zhicheng Dou (corresponding author), Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China; e-mail: dou@ruc.edu.cn; Jian-Yun Nie, DIRO, University de Montreal, Montreal, Quebec, Canada; e-mail: nie@iro.umontreal.ca; Zhenlin Chen, JD.com Inc, Beijing, China; e-mail: chenchenlin6@jd.com; Guoyu Tang, JD.com Inc, Beijing, China; e-mail: tangguoyu@jd.com; Sulong Xu, JD.com Inc, Beijing, China; e-mail: xusulong@jd.com; Ji-Rong Wen, Gaoling School of Artificial Intelligence, Renmin University of China, Beijing, China, Engineering Research Center of Next-Generation Intelligent Search and Recommendation, Ministry of Education, Beijing, China, and Beijing Key Laboratory of Big Data Management and Analysis Methods, Beijing, China; e-mail: jrwen@ruc.edu.cn.

Permission to make digital or hard copies of all or part of this work for personal or classroom use is granted without fee provided that copies are not made or distributed for profit or commercial advantage and that copies bear this notice and the full citation on the first page. Copyrights for components of this work owned by others than the author(s) must be honored. Abstracting with credit is permitted. To copy otherwise, or republish, to post on servers or to redistribute to lists, requires prior specific permission and/or a fee. Request permissions from permissions@acm.org.

© 2025 Copyright held by the owner/author(s). Publication rights licensed to ACM.

ACM 1558-2868/2025/5-ART84

<https://doi.org/10.1145/3726864>

ACM Reference format:

Jiongnan Liu, Zhicheng Dou, Jian-Yun Nie, Zhenlin Chen, Guoyu Tang, Sulong Xu, and Ji-Rong Wen. 2025. Enhancing Sequential Personalized Product Search with External Out-of-sequence Knowledge. *ACM Trans. Inf. Syst.* 43, 4, Article 84 (May 2025), 25 pages. <https://doi.org/10.1145/3726864>

1 Introduction

In recent years, more and more people have been buying products on online platforms such as Amazon.com and JD.com. They issue queries on these platforms and then are provided with a list of products by the web sites' ranking systems, on which they can click and make purchases. Similar to the web search scenario [29, 30, 35, 36], it is shown that leveraging user history to provide personalized product search results can improve user satisfaction [1, 3]. For example, if a user used to buy Logitech's computer accessories, it is more reasonable that the search engine ranks Logitech's products higher when the user searches for "keyboard." Many personalized product search models [1–4, 7, 8, 11, 13, 16–18, 20, 24, 25, 27, 31, 32] have been proposed to accomplish this task.

Recently, several models [1, 7, 16, 17] are proposed to build query-dependent user profiles over historical user behavior sequences and calculate products' personalized ranking scores by comparing them with the profiles. For example, AEM [1] applied the query attention mechanism to generate query-dependent user profiles. Compared with the query-independent user representations captured by latent space models [3] or knowledge-centric model [4] without considering current queries, these query-dependent profiles can pay more attention to the historical items more related to the current intent, hence can bring more satisfying results to users.

Although the aforementioned sequential personalized product search models have achieved encouraging performances through attention mechanisms to build dynamic user profiles, they suffer from limited and noisy user histories. There are usually a small number of products purchased by a user, and the profiles learned based on them might be inaccurate and unstable since the calculated attention can only refer to limited information within user sequences. In this manner, the products randomly purchased by the user (they are irrelevant to the user's main interests) can strongly affect the built user profiles. In this article, we claim that some *external* out-of-sequence knowledge can supplement the original sequences and help learn more informative and stable user profiles. Indeed, different from documents in web search, products typically have richer global relations with each other. For instance, two products may belong to the same brand/category or have been bought in the same transaction. We argue that these global *item–item relations* can help us build better user profiles. For example, if a user kept buying Apple products in her history, we can strengthen the interactions between these products during user profiling, thereby profiling her as a passionate Apple fan and recommending relevant Apple products when she issues new related queries. The weights of other products in her histories for capturing user interests will also decrease correspondingly, leading to a more accurate user profile. Furthermore, additional *query–query relations* are also helpful to user interest modeling: the items purchased under similar queries should be highly correlated and have similar contextualized representations while constructing the user profiles. In contrast, if the meanings of two historical queries are different from each other, their corresponding items should also be irrelevant. In summary, by introducing this out-of-sequence knowledge and information, the user profiling process can focus more on the primary preferences and ignore other secondary and noisy ones.

As discussed above, we argue that the aforementioned external item–item and query–query relationships should be incorporated into the modeling of user behavior sequences, yet they remain

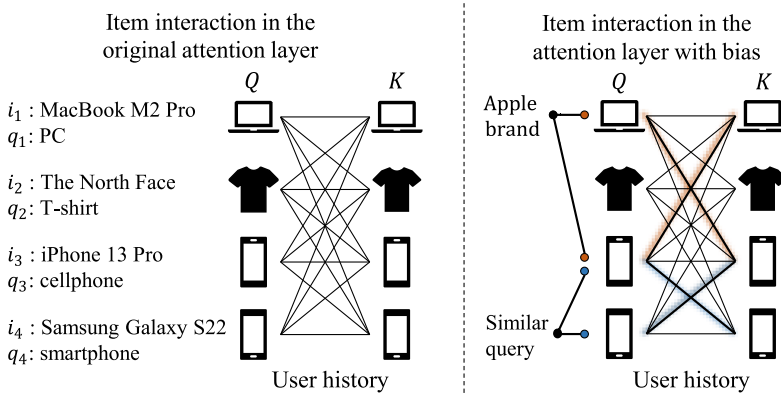


Fig. 1. Comparison between the existing sequential user interest modeling in personalized product search and our proposed model using additional attention bias. Because i_1 and i_3 have the same brand and q_3 and q_4 are similar, the interactions between (i_1, i_3) and (i_3, i_4) are strengthened.

overlooked by existing approaches. The most natural strategy for encoding these relations is to build a global **Knowledge Graph (KG)** including these relations and construct pre-trained knowledge embeddings to initialize sequential models. However, as the KG is built from all existing relations, the knowledge embeddings cannot focus on the current user sequences. Such overall knowledge tends to be too general to enrich the current user profile. Besides, it is difficult to incorporate the query relations into the knowledge embeddings. In this article, to supplement the KG initialization, we propose to extract the corresponding external relations relevant to the current user sequence and incorporate them with the original intra-sequence interaction. By this means, only the item–item and query–query relations contained in the current sequences are considered. These relations should be more specific and effective to pilot the intra-sequence profile building and understand the current user.

To achieve this goal and build better user profiles, we propose a **Biased Attention-based Transformer AutoEncoder (BATA)** model. Inspired by the existing works [33, 34], we propose adjusting the transformer-based user encoder, which is widely used in previous models [7, 8, 20], by modeling external item–item relations and query–query dependencies as attention biases in the attention network. More specifically, for items, we build an item–attribute graph based on the metadata of products, measure the distances between items, and use them as external *global item relation bias*. For queries, we calculate similarities between their semantic representations as additional *historical query dependency bias*. These attention biases can serve as prior knowledge about the item proximity in the current sequence and provide signals to guide the intra-sequence item interactions, which is shown in Figure 1. We hope this prior knowledge will help filter the noisy behaviors in user history and yield better sequence representations.

However, this external knowledge, together with the input features in user sequences, may vanish during the propagation in the stacked attention layers of the transformer encoder without additional supervision. To further enhance these external relations and force the encoder to capture the important characteristics in the input data, we adopt the AutoEncoder [5] paradigm and enrich the sequence encoder by devising auxiliary decoders which try to reconstruct the global item relations and the original sequence. With the adjusted attention and the global item relation recovery task, we expect to improve the robustness and reliability of the generated user representations.

Experimental results on four sub-categories of simulated Amazon datasets and JDsearch dataset [23] based on real user search logs show that the proposed method outperforms existing approaches.

The main contributions of our article are three-fold:

- (1) We propose incorporating supplementary out-of-sequence item–item and query–query relations into user behavior sequence modeling for personalized product search.
- (2) We modify the attention mechanism in the transformer encoder by adding a bias term to incorporate the above external relationships to guide user interest modeling.
- (3) We devise two auxiliary decoders to ensure the information contained in the input sequences and global relations is well captured in user encoders.

The rest of the article is organized as follows. We introduce related works in Section 2. Following this, we introduce the model framework in Section 3. We describe experimental settings in Section 4, and analyze experimental results in Section 5. We conclude the article in Section 6.

2 Related Work

2.1 Personalized Product Search

Personalized product search usually either builds *general user profiles* or utilizes sequence-based mechanisms to construct *sequential user profiles* from their histories.

2.1.1 General Personalized Product Search Approaches. Most early approaches for personalized product search belong to this category. HEM [3] followed the non-personalized product search model LSE [19] to adapt the latent space model to calculate user and item representations through user reviews. DREM [4] built a KG containing users, items, and attributes based on the adequate relations in the metadata of items. CAMI [24] improved it by replacing the single user profile in KG with multiple preference vectors and their category indications. GraphSRRL [25] tried to find the designed patterns in the user–item graph to optimize the model. Recently, hypergraph [11] was introduced into personalized product search, allowing to inclusion of queries in the graph. However, in these general approaches, a unique model is built for a user or an item, regardless of the user’s query being considered. Such approaches run the risk that the general user profiles may be useless or even harmful to the current user’s query.

2.1.2 Sequential Personalized Product Search Approaches. To capture query-specific user interests, some approaches try to consider current queries to help aggregate user histories from a sequential perspective. ALSTP [17] introduced hierarchical **Recurrent Neural Network (RNN)** to encode user behaviors at both long-term and short-term levels. AEM and ZAM [1] applied query attention to build query-specific user profiles. Several approaches [7, 8, 20] further improved them by replacing the single attention layer with the transformer encoder structure and bringing the sequential or temporal information into modeling. RTM [8] substituted the product embeddings in TEM with embeddings of words in their reviews to capture fine-grained interactions. TEM [7] improved ZAM by using the transformer encoder structure and adding position embedding to bring the order information into modeling. RTM [8] substituted the product embeddings in TEM with embeddings of words in their reviews to capture fine-grained interactions. CoPPS [13] further improved it by applying contrastive learning methods. HIFN [27] disentangles the short-term and long-term interests and captures them, respectively. MRA [20] focused on the time interval between two purchases to enhance temporal signals instead of order signals. Nevertheless, as previously stated, the majority of these approaches concentrated solely on item interactions in the current user’s sequence, ignoring the external item–item relations and dependencies among historical queries. SBG [16] and HGN [2] tried to alleviate the problem by boosting item and user representations with the graph embedding leveraging other users’ histories. However, they still neglected historical queries, and no improvement in profile building is observed with limited user

histories. Besides, the graph embeddings introduced are global embeddings and do not focus on the current histories.

2.2 Attention Bias in Transformer

The transformer [28] structure has been successfully used in many areas such as natural language processing [14], **Information Retrieval (IR)** [7, 35], and CV [15]. Based on a multihead self-attention network, the transformer encoder can perform deep interactions between inputs to obtain contextualized output embeddings and sequence representations. Recently, some researchers [33, 34] find that adding a bias term to the attention matrix can better leverage out-of-sequence features. For example, Graphormer [34] introduced the distance matrix and edge features between nodes, and this improved the graph node classification performance. This additional bias term can be considered as prior knowledge [22] injected into the interaction process. In our model, we adopt the same way to capture the external item–item and query–query relations and incorporate them as biases.

2.3 AutoEncoder in Information Retrieval

AutoEncoders have been widely adopted to learn representations of data such as images and texts. They compress the high-dimension inputs into low-dimension dense embeddings by encoders and design decoders to perform the reconstruction. Recently, this idea has been applied to IR [21, 26] to learn better representations of documents. Studies have shown that a shallow decoder can improve the performance of the encoder by verifying whether the dense encoded representations contain enough information and the limited parameters in the decoder can prevent the overfitting problem. For example, SEED-Encoder [26] designed an auxiliary non-fully connected transformer decoder with three layers to support the full-size 12-layer transformer encoder. In the search scenario, we can consider the user behavior histories as sequences of web sites or items visited, which is similar to considering document contents as sequences of terms. So, it becomes natural to exploit AutoEncoder for personalized product search. In our model, we use two weak and shallow decoders based on RNN and **Multi-Layer Perceptron (MLP)** to decode the user histories and global relations between items to improve the performance of the user encoder.

3 The BATA Model

As mentioned earlier, most of the approaches in sequential personalized product search focus on current sequences and ignore the out-of-sequence relationships between items (e.g., items belonging to the same brand). The dependencies between historical queries are also overlooked in most models. Besides, the design based on encoder-only architecture makes it difficult to guarantee that the important input information is correctly encoded in the encoding process. To solve these problems, we propose the *BATA* model, which incorporates these external item–item and query–query relationships as attention biases into a transformer-based user encoder. We also design two decoders to provide auxiliary signals for user encoders and augment user profiles.

3.1 Overview

We start with the problem formulation. Given the user u , the current query q_c , and the candidate item i , we need to grade the item i with score $S(i|u, q_c)$ to rank the item for user u . We assume that there is a sequence of historical queries $Q = [q_1, \dots, q_m]$ issued by user u and a sequence of historical items $I = [i_1, \dots, i_m]$ purchased by user u . Notice that we assume the lengths of historical queries and historical items are the same, which means users interacted with one product under each query. However, if users interact with multiple ones under one certain query, we can repeat this query to align the number of products and to make these two sequences consistent.

For simplicity, we assume that the length of all users' histories equals m . Shorter sequences are padded, while longer sequences are truncated, to fit them into the required length.

First, we project the item and word¹ IDs into dense representations. To initially encode the item-item relations in our model, we first utilize the relationships among users and items in the datasets to build a global KG and employ a KG model to initialize item and word embedding tables. Then, we apply the generative language model to further optimize them through the text information of reviews following existing approaches for query-item matching. It aims to use the items' embeddings to generate each word's embedding in their descriptions and reviews. For user representations, attention-based models have shown effectiveness in capturing sequential features in many areas [7, 10, 35]; therefore, we apply the transformer model to build user interests based on their histories. Furthermore, as we mentioned in Section 1, relations among purchased items and semantic similarities among historical queries can provide additional knowledge for constructing user profiles. Therefore, we design a transformer-based user encoder in which attention is biased by these item-item and query-query relations. However, while this external knowledge along with the input features is provided to the transformer encoder, they may not be well-captured by the module without extra guidance. To ensure that the important item features are well captured by the user encoder, following the principle of AutoEncoder, we further introduce auxiliary sequence and graph decoders to reconstruct the input data and item relations from the output user embedding and contextualized item representations. Finally, we can score the candidate items based on their matching scores with the query and the user and personalized features. To optimize our model, we design four losses to tune parameters:

- Generative language model loss L_{gl} : it aims to jointly tune product and word embeddings for query-item matching;
- Sequence and graph decoder losses L_{sd} and L_{gd} : they evaluate the informativeness of user encoders;
- Personalized product search loss L_{ps} : it maximizes the similarities between user profiles and ground truth items.

The overall structure of the proposed model is shown in Figure 2. In the following parts, we will describe the details of different modules in BATA.

3.2 Product and Query Representation

First, we create two global embedding tables for items and words following the most personalized product search approaches. As we mentioned before, we first construct a KG to initialize the embedding tables to incorporate global item relations. Following previous KG-centric methods [2, 4] which suggests that including rich relations can yield better performances, we include five types of entities in our KG: users, items, words, brands, and categories. Seven types of relationships are included:

- *Write*: Word w is occurred under the review for item i ($i \rightarrow w$) or is written by user u ($u \rightarrow w$).
- *Is_brand*: Item i belongs to brand b ($i \rightarrow b$).
- *Is_category*: Item i belongs to category c ($i \rightarrow c$).
- *Also_bought*: Items i_1 and i_2 have been purchased by the same user(s) ($i_1 \rightarrow i_2$).
- *Bought_together*: Items i_1 and i_2 have been purchased under the single transaction ($i_1 \rightarrow i_2$).
- *Also_viewed*: Items i_1 and i_2 have been viewed by the same user(s) ($i_1 \rightarrow i_2$).
- *Search&Purchase*: User u issues a query q and then purchases an item i ($u + q \rightarrow i$).

¹The word means the terms in user queries, product descriptions, and product reviews.

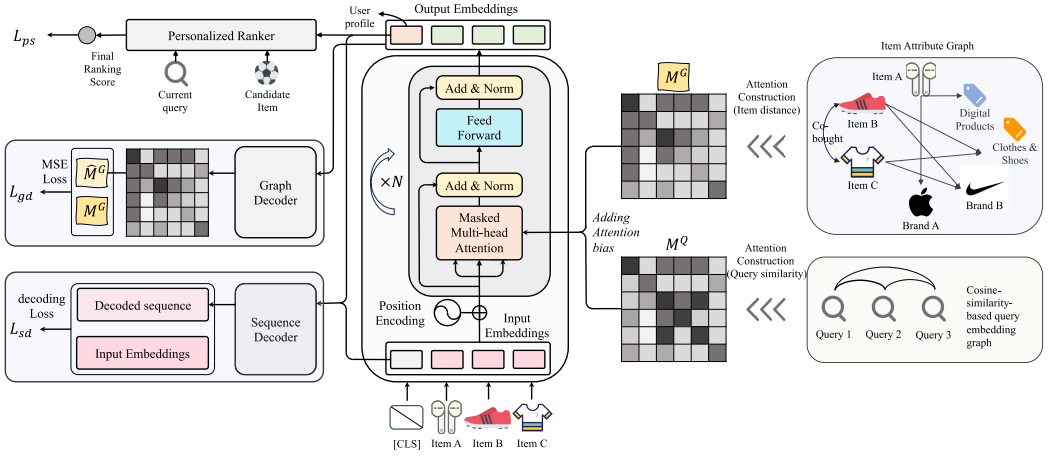


Fig. 2. The main structure of BATA model for personalized product search.

All these relations are originally included in the metadata of the experiment datasets. For the construction process, we first add all entities in datasets into the graph. Then, we add the corresponding relations between entities if they exist in the datasets. Specifically, the first six types excluding *Search&Purchase* are static relations, which means they can be modeled into a certain embedding in spite of the head and tail entity in the KG. However, the *Search&Purchase* is included to model the purchase behaviors between a user u and an item i with a dynamic query q , thus the relation embedding should be dynamic according to the contents of the query q . Specifically, it adopts the query embedding in Equation (2) as the relation representation. To build embeddings for these relations and entities, we use TransE [9]-based model DREM [4] to acquire the knowledge embeddings of words and products.

To align items with users' queries, we need to link these two embeddings together. Similar to existing approaches [1, 3, 7], we assume that the embeddings of items should be able to generate the words related to them, e.g., the reviews. Therefore, we can maximize the likelihood of generating review words of all purchased items including historical ones to bridge the gap between embeddings of words and items. The generative language modeling loss is defined as follows:

$$L_{gl} = - \sum_{\text{purchased item } i} \log \sum_{w \in R_i} \frac{\exp(\mathbf{w}^\top \cdot \mathbf{i})}{\sum_{w' \in \mathcal{V}} \exp(\mathbf{w}'^\top \cdot \mathbf{i})}, \quad (1)$$

where \mathbf{w} and \mathbf{i} are the embeddings of word w and item i by embedding look-up, \mathcal{V} is the whole vocabulary in corpus, and R_i is the reviews of product i . Notice that due to the large vocabulary, we adopt a negative sampling technique to accelerate the computation.

For query embedding, following existing approaches [1, 4, 7], we adopt a learnable non-linear function to project the average query word embedding. For example, for the current query q_c , its representation is calculated by:

$$\mathbf{q}_c = \tanh\left(W \cdot \frac{\sum_{w \in q_c} \mathbf{w}}{|q_c|} + b\right). \quad (2)$$

In this manner, we can build the sequences of historical item and query embeddings for the subsequent user encoder module as follows:

$$\mathbf{Q} = [\mathbf{q}_c, \mathbf{q}_m, \dots, \mathbf{q}_j, \dots, \mathbf{q}_1], \quad (3)$$

$$\mathbf{I} = [\mathbf{i}_{\text{cls}}, \mathbf{i}_m, \dots, \mathbf{i}_j, \dots, \mathbf{i}_1], \quad (4)$$

where $\mathbf{q}_j, \mathbf{i}_j$ is the corresponding embedding of query q_j and item i_j . Notice that the above sequences are in reverse order. We insert the current query embedding \mathbf{q}_c and a special embedding \mathbf{i}_{cls} ² at the beginning of the sequences. For convenience of description, we may use \mathbf{Q}, \mathbf{I} instead of \mathbf{q}, \mathbf{i} in the following parts, where $\mathbf{Q}_0 = \mathbf{q}_c, \mathbf{Q}_k = \mathbf{q}_{m-k+1} (1 \leq k \leq m)$ and $\mathbf{I}_0 = \mathbf{i}_{\text{cls}}, \mathbf{I}_k = \mathbf{i}_{m-k+1} (1 \leq k \leq m)$.

3.3 User Encoder Module

To model user interests, we choose to build user profiles from her historical behaviors following sequential approaches. We assume that a user's profile is reflected by the items she purchased in the past. We leverage the transformer encoder and feed historical item embedding sequence \mathbf{I} into it to model interactions and build the final user profile. We have:

$$\tilde{\mathbf{I}} = \text{Trm}^L(\mathbf{I}) = \underbrace{\text{Trm}(\text{Trm}(\dots(\mathbf{I})))}_L, \quad (5)$$

where Trm^L is a L -layer transformer encoder and $\tilde{\mathbf{I}}$ denotes the contextualized output embeddings. The transformer encoder layer Trm is defined as follows:

$$\text{Trm}(\mathbf{I}) = \text{LN}(\mathbf{X} + \text{FFN}(\mathbf{X})), \quad (6)$$

$$\mathbf{X} = \text{LN}(\mathbf{I} + \text{SA}(\mathbf{I})), \quad (7)$$

$$\text{SA}(\mathbf{I}) = \text{Attn}(W^Q \mathbf{I}, W^K \mathbf{I}, W^V \mathbf{I}), \quad (8)$$

$$\text{Attn}(Q, K, V) = \text{softmax}\left(\frac{QK^\top}{\sqrt{d}}\right)V, \quad (9)$$

where SA, LN, and FFN denote the self-attention network, layer normalization, and feed-forward network, respectively. $\text{Attn}(Q, K, V)$ is the attention, we use the multihead technique but omit it here to simplify the description.

The special token \mathbf{i}_{cls} we concatenated at the beginning of item sequence \mathbf{I} is intended to capture the next item the user would buy. Its corresponding contextualized output reflects both the current intent and the historical behaviors. We use this representation as the user profile embedding, i.e.,

$$\mathbf{u} = \tilde{\mathbf{I}}_0. \quad (10)$$

The above user profile does not reflect only the user's long-term interests, as in most previous work. Instead, it reflects the user's interest at a specific moment. We can rely more on it to determine the next item.

However, the above user profile is only built on interactions with historical items, which is prone to noisy purchase behaviors. Although item-item relationships have been encoded into the initial knowledge embeddings, these representations remain too generic to effectively capture specific user interests, as the KG is constructed based on global relations across all entities. A more promising way is to localize these global item-item relations to guide profile construction. In addition, KG is often unable to capture query-query relations. To address this issue, we further introduce an attention bias term into the transformer to leverage these external references for better sequential modeling. Finally, the bias is composed of global item relations and history query

²Implemented as zero vector in our model.

dependencies related to the current sequence. The modification is made to the attention mechanism in the transformer encoder. Specifically, we change the original $\text{Attn}()$ equation to:

$$\text{BiasAttn}(Q, K, V, M) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}} + M\right)V, \quad (11)$$

where M is the bias matrix built from the external information. In this manner, we expect that the biased attention transformer can absorb the prior knowledge contained in M [22]. We will discuss how we build this bias term for personalized product search in the following sections.

3.3.1 Global Item Relation Bias. As we discussed before, the relations among items can pilot and benefit the user profile building. These out-of-sequence item–item relations have been encoded into item embeddings through KG embedding initially. However, as the built KG is composed of relations among all items, the proximity between items in the current user’s history has not been emphasized. Thus, we need to zoom into the current sequences to effectively capture the potential of global item relationships. For instance, if two products with historical interaction records have been co-purchased previously or share the same brand affiliation, they should exhibit higher correlation strength and more significant interaction weights compared to other product pairs. To better embrace these global relations for guiding the user profile building, we choose to embed the proximity between current items into attention bias to provide prior knowledge for integration in the transformer encoder. We build an item–attribute graph to measure the proximity between current items. In the graph, there are three kinds of nodes: item, brand, and category, and three kinds of edges:

- Is_brand : One item belongs to one brand.
- $Is_category$: One item belongs to one category.
- $Is_related_product$: Two items are related products. Their relations could be “also bought,” “also viewed,” or “bought together.”

An example of an item–attribute graph is shown in the right-middle part of Figure 2. The distances between items in such a graph can represent the out-of-sequence proximity among products. Since the item relation graph only includes limited attributes and items as nodes and limited relations as edges, the distances between items can be meaningful. The distance 1 between item pairs infers they may be bought or viewed together, which implies a strong relation. The distance 2 between item pairs infers they may belong to the same categories or brands, which may suggest they have some common features but are not as strong as the previous one. Item pairs with distance 3 are less co-related.

The following distance matrix bias is defined to represent the item–item relations:

$$M_{jk}^G = \begin{cases} \text{distance}(i_{m-j+1}, i_{m-k+1}) & (j, k \neq 0) \\ 0 & (j = 0 \text{ or } k = 0) \end{cases}, \quad (12)$$

Notice that because we reverse the history list and attach the current search behavior to the beginning of it in Equation (3), we reverse the indexes and fill the first row and column with zero in the distance matrix.

Due to the large number of nodes in the graph, it is too expensive to compute the exact shortest distance between every item pair. The time complexity of the Floyd algorithm is $O(|V|^3)$, which is slow and unacceptable in larger datasets. However, since we only want to leverage closer item pairs to better guide user interest profiling, we can ignore item pairs with far distances. Specifically, we accelerate the computation of M_{jk}^G by only calculating the distance between items less than a

threshold τ_{val} and approximately set the distances of all the rest item pairs with a larger value τ_{pad} . The details of distance calculation can be found in Algorithm 1 in Section 4.2.3.

3.3.2 Historical Query Dependency Bias. In addition to the item–item relations, the historical query sequence can also help infer user historical intents. For example, if the current query is “computer,” items under query “cellphone” should receive more attention than items under query “cloth.” Besides, items with similar queries should be grouped to strengthen user interest in the specific intent behind these similar queries.

To accomplish this, some existing approaches [35] add the query representations to the item embeddings as the position and segment embeddings added in BERT [14]. However, different from the web search scenario, in personalized product search, there are acute language mismatch issues [3, 19] between queries and product reviews. For example, a user may search with the word “cellphone” but write reviews about the screen or battery of the cellphone she purchased. Therefore, the semantic meanings behind the embeddings of products derived from reviews can be different from those of queries. It is thus sub-optimal to directly fuse them. This is confirmed in our preliminary experiments (in our ablation study in Section 5.2), even though we tried to project the query representations into product spaces via non-linear functions. Thus, we propose to disentangle the representation between products and queries instead of mixing them up. However, in datasets with lower query-product semantic discrepancy, directly adding them together can yield achieve competitive performance.

Thus, we build a query similarity matrix to model their relations and regard it as a part of the attention bias term to guide the interactions between item embeddings. In this way, we maintain the query dependencies as references for item aggregations without explicitly mixing representations of queries and items together. To obtain this query dependency matrix M^Q , we adopt the cosine similarity distance and leave more complex ways for future work:

$$M_{jk}^Q = \mathbf{Q}_j^\top \cdot \mathbf{Q}_k \cdot (\|\mathbf{Q}_j\| \cdot \|\mathbf{Q}_k\|)^{-1}. \quad (13)$$

These two relationship matrices in Equations (12) and (13) can be regarded as external references to provide prior knowledge of item interactions and profile building. We add them together as attention bias terms. Notice that the monotonicities of these two matrices and scales of values are different: M^Q is a similarity matrix with values ranging in $[-1, 1]$, while M^G is a distance or dissimilarity matrix with non-negative values. We learn a negative coefficient a for M^G during training to make them consistent:

$$M = M^Q + aM^G + b \quad (a < 0). \quad (14)$$

Finally, we feed the history item sequence into the biased attention transformer encoder to obtain contextualized item embedding sequence \tilde{I} and user profile \mathbf{u} as follows:

$$\tilde{I} = \text{BiasTrm}^L(\mathbf{I}, M), \quad \mathbf{u} = \tilde{I}_0, \quad (15)$$

$$\text{BiasTrm}(\mathbf{I}, M) = \text{LN}(X + \text{FFN}(X)), \quad (16)$$

$$X = \text{LN}(\mathbf{I} + \text{BiasSA}(\mathbf{I}, M)), \quad (17)$$

$$\text{BiasSA}(\mathbf{I}, M) = \text{BiasAttn}(W^Q \mathbf{I}, W^K \mathbf{I}, W^V \mathbf{I}, M). \quad (18)$$

So far, we have built the sequential user profile by leveraging external item–item and query–query relations. However, as the transformer encoder is stacked with L attention layers, the knowledge in original input sequences and attention bias introduced may vanish during propagation. The extent to which the information input to the encoder is encoded into the user embedding remains

quantitatively uncharacterized. To force the model to sufficiently encode the input information, we design two decoders to decode sequence-based and graph-based encoder input, respectively.

3.4 Sequence Decoder Module

As we discussed before, the absence of decoders in the current approaches to personalized search does not guarantee that important information in the input has been encoded. The sequence decoder aims to reconstruct the original item sequence from the built user profile to ensure the encoder captures important features in the input sequence. Specifically, we leverage the built user profile embedding in Equation (10) as input to decode the item embedding of user histories in a step-by-step manner. We adopt an auto-regressive loss for decoding the original user sequence:

$$L_{sd} = - \sum_{k=m}^1 \log P(i_k | i_{k+1:m}; \mathbf{u}). \quad (19)$$

Notice that as we reversed the original purchasing item list, the decoding process is also reversed (from the last purchased product to the first one). The probability of decoding the ground-truth item i_k is estimated as:

$$P(i_k | i_{k+1:m}; \mathbf{u}) = \frac{\exp(\mathbf{i}_k^\top \cdot \hat{\mathbf{i}}_k)}{\sum_{i' \in \mathcal{I}} \exp(\mathbf{i}'^\top \cdot \hat{\mathbf{i}}_k)}, \quad (20)$$

where $\hat{\mathbf{i}}_k$ is the predicted item embedding by the sequence decoder and \mathcal{I} is the item corpus. Similarly, we adopt negative sampling for higher efficiency due to a large number of products. To generate more difficult negative samples for the decoder, we also add items at other positions in user sequences as hard negatives. This is used to supplement the randomly sampled negatives.

In this paradigm, we need to devise a decoder to rebuild the historical item list step by step. It has been shown [26] that a shallow decoder can benefit the encoder. Therefore, we design a simple RNN-based decoder to improve the transformer-based encoder. The predicted item embedding is obtained through GRU [12] cell fed with the hidden vector \mathbf{h}_{k+1} and the last predicted item embedding $\hat{\mathbf{i}}_{k+1}$:

$$\mathbf{h}_k = \text{GRUcell}(\mathbf{h}_{k+1}, \hat{\mathbf{i}}_{k+1}), \quad \hat{\mathbf{i}}_k = W^{\text{sd}} \mathbf{h}_k + b^{\text{sd}}, \quad (21)$$

where $W^{\text{sd}}, b^{\text{sd}}$ are trainable parameters. We set $\mathbf{h}_{m+1} = \mathbf{u}$ and $\hat{\mathbf{i}}_{m+1} = \mathbf{i}_{\text{cls}}$ to initialize the decoding process. With the requirement of rebuilding the historical item sequences from the user profile, it is expected that the former is encoded into the latter.

3.5 Graph Decoder Module

The sequential decoder incorporated in our model enables the injection of sequential signals, including historical items, into the encoding process. We also maintain an item-attribute graph that encodes global proximity metrics between items through structural distance measures. To enforce the encoder to incorporate such global information, we employ a graph decoder to predict the pairwise item distances, i.e. M^G in Equation (12), based on the contextualized item embeddings after interactions in the transformer. We concatenate the first-order and second-order interaction features to predict the distance for item pairs, i.e.,:

$$\hat{M}_{jk}^G = \text{MLP}([\tilde{\mathbf{I}}_j; \tilde{\mathbf{I}}_k; \tilde{\mathbf{I}}_j \cdot \tilde{\mathbf{I}}_k]), \quad (22)$$

where \hat{M}_{jk}^G is the predicted distances in the item-attribute graph between historical item i_{m-j+1} and i_{m-k+1} . Accordingly, the graph decoder loss is defined as the MSE loss between the ground-truth

item distance matrix and the predicted one:

$$L_{\text{gd}} = \sum_{1 \leq j \leq k \leq m} (\hat{M}_{jk}^G - M_{jk}^G)^2. \quad (23)$$

Designing this item distance prediction task, together with the introduced attention bias, the model can better leverage the global item relations related to current sequences into the generated user representations.

We need to further state that both the sequence decoder module and the graph decoder module will not be applied during inference time and there is no information leakage problem. These modules serve as the autoencoder structure in our BATA model which aims to guarantee the user encoder module indeed memorizes and utilizes the given information.

3.6 Personalized Ranker Module

Having built the reliable and informative user profile vector \mathbf{u} , we can calculate the matching score between a user and an item i . This would lead to a personalized product search. However, this personalization is query-independent, i.e., it is the same for any submitted query. To make a query-specific search, we also need to measure the similarity between query q_c and item i as an *ad hoc* matching score. So, the matching score combines both user–item and query–item scores as follows:

$$S^{\text{mat}}(i|u, q_c) = \lambda \times \mathbf{u}^\top \mathbf{i} + (1 - \lambda) \times \mathbf{q}_c^\top \mathbf{i}, \quad (24)$$

where λ is initialized as 0.5 and tuned in the training process.

Besides the matching scores, we can also assess the personalization of scoring item i through its distances to historical items I . We denote this distance vector as \mathbf{D} in which $D_j = \text{distance}(i, i_j)$. This distance vector \mathbf{D} can also be regarded as a simple version of **Short- and Long-Term Behavior (SLTB)** features [6] under the personalized web search scenario, which aims to represent the similarities between current candidates and historical clicked/purchased ones. Thus, the feature score in the personalized product search situation can also be calculated through an MLP layer in the same way as SLTB:

$$S^{\text{fea}}(i|u, q_c) = \text{MLP}(\mathbf{D}). \quad (25)$$

Finally, the final ranking score $S(i|u, q_c)$ for item i is determined by combining the two scores in Equations (24) and (25):

$$S(i|u, q_c) = \text{MLP}([S^{\text{mat}}(i|u, q_c); S^{\text{fea}}(i|u, q_c)]). \quad (26)$$

Following previous approaches, we optimize the personalized product search task by maximizing the probability of the ground-truth purchase triplet $\langle i, u, q \rangle$, which means that item i is purchased under the query q issued by user u . The personalized product search loss is calculated as:

$$L_{\text{ps}} = - \sum_{\langle i, u, q \rangle} \log \frac{\exp(S(i|u, q_c))}{\sum_{i' \in \mathcal{I}} \exp(S(i'|u, q_c))}. \quad (27)$$

Similar to the generative language modeling loss L_{gl} in Equation (1) and sequence decoder loss L_{sd} in Equation (19), we use negative sampling. Finally, we add all four losses together:

$$L = L_{\text{ps}} + L_{\text{gl}} + \alpha L_{\text{sd}} + \beta L_{\text{gd}}, \quad (28)$$

where α and β are hyper-parameters to balance the weight of two decoding losses.

Table 1. Statistics of the Datasets

| Dataset | #Users | #Items | #Categories | #Brands | #Interactions |
|---------------------------------------|---------|------------|-------------|---------|---------------|
| <i>Cell Phones and Accessories</i> | 27,879 | 10,429 | 206 | 955 | 194,439 |
| <i>Clothing, Shoes, and Jewellery</i> | 39,387 | 23,033 | 1,193 | 1,182 | 278,677 |
| <i>Sports and Outdoors</i> | 35,598 | 18,357 | 1,443 | 2,412 | 63,001 |
| <i>Electronics</i> | 192,403 | 63,001 | 983 | 3,525 | 1,689,188 |
| JDsearch Dataset | 173,831 | 12,872,636 | 76 | 182,585 | 26,667,260 |

4 Experimental Setup

4.1 Datasets

In our experiments, we collect two types of datasets to evaluate the performance of models: the simulated *Amazon datasets*³ and the *JDsearch dataset*⁴ [23] collecting real user search logs. The statistics of datasets are shown in Table 1. We will introduce the details of the datasets in the following part.

4.2 Amazon Datasets

4.2.1 Dataset Collection. Following previous studies [1–4, 7], we first use the public Amazon 5-core datasets to conduct experiments, which filter long-tail products and users with less than five interactions. As the dataset is initially designed as a recommendation dataset without queries, we need to create queries for products to simulate the search scenario. Following previous work [1–4, 7], we concatenate the words in the category list of products and remove the duplicated words and stopwords to construct the pseudo queries for products. For example, if the category list of one product is [Electronics, Camera, Camera Lenses], the constructed query would be “Electronics Camera Lenses.” We conduct our experiments on four categories among all Amazon datasets: *Cell Phones and Accessories*, *Clothing, Shoes and Jewellery*, *Sports and Outdoors*, and *Electronics*.

4.2.2 Evaluation. To evaluate the performance of models, we split the Amazon dataset into the training set, validation set, and test set according to the ratio 0.8:0.1:0.1 in chronological order to ensure there is no data leakage in training. To construct the queries of current purchasing behaviors, we distribute the pseudo queries of each item to the training set and the test set according to the ratio 0.7:0.3. This guarantees that no valid or test query has been seen and memorized by the model during the training process. The test set and the validation set share the same query corpus. If all queries about one item are in the test set, we will move one of the random queries back to the training set so that all products have at least one query seen in the training process. For the queries of historical items, we randomly choose one of the pseudo queries to put into the historical sequence. We use **Mean Reciprocal Rank (MRR)**, **Normalized Discounted Cumulative Gain (NDCG)@10**, and **Precision@1** as evaluation metrics.

4.2.3 Implementation Details. We set the length m of all user histories as 20. We set the embedding dimension of all models as 128 and train them for 20 epochs. For the knowledge embedding initialization in Section 3.2, we directly use the DREM’s embeddings of items and words as DREM can be regarded as a modified translation-based knowledge embedding method. For our BATA model, we choose the coefficient α , β of the sequence decoder and the graph decoder in {0.1, 0.3, 0.5}. We choose the learning rate from {0.0005, 0.001, 0.003}. The number of transformer layers is chosen from {1, 2}, and the number of attention heads is set to 8. We use the original settings for

³<http://jmcauley.ucsd.edu/data/amazon/>.

⁴<https://github.com/rucliujn/JDsearch>.

Algorithm 1: Distance Calculation in Item–Attribute Graph

```

1: input: Item-attribute Graph:  $\mathcal{G} = \{\mathcal{V}, \mathcal{E}\}$ , item node set:  $\mathcal{V}_I$ , distance threshold  $\tau_{\text{pad}}$ .
2: output: distance projection: distance:  $\mathcal{V}_I \times \mathcal{V}_I \mapsto N$ .
3: //initialization
4: distance( $\cdot, \cdot$ ) =  $\tau_{\text{pad}}$ .
5: for node  $v \in \mathcal{V}_I$  do
6:   distance( $v, v$ ) = 0.
7: end for
8: for edge  $e = \langle v_i, v_j \rangle \in \mathcal{E}$  do
9:   //distance = 1
10:  if  $v_i \in \mathcal{V}_I$  and  $v_j \in \mathcal{V}_I$  then
11:    distance( $v_i, v_j$ ) = 1.
12:    distance( $v_j, v_i$ ) = 1.
13:  end if
14:  //distance = 2
15:  for  $v_k \in \text{Neighbour}(v_i)$  do
16:    if  $v_k \in \mathcal{V}_I$  and distance( $v_k, v_j$ ) > 1 then
17:      distance( $v_k, v_j$ ) = 2.
18:      distance( $v_j, v_k$ ) = 2.
19:    end if
20:  end for
21:  for  $v_k \in \text{Neighbour}(v_j)$  do
22:    if  $v_k \in \mathcal{V}_I$  and distance( $v_i, v_k$ ) > 1 then
23:      distance( $v_i, v_k$ ) = 2.
24:      distance( $v_k, v_i$ ) = 2.
25:    end if
26:  end for
27:  //distance = 3
28:  for  $v_k \in \text{Neighbour}(v_i)$  do
29:    for  $v_p \in \text{Neighbour}(v_j)$  do
30:      if  $v_k \in \mathcal{V}_I$  and  $v_p \in \mathcal{V}_I$  and distance( $v_k, v_p$ ) > 2 then
31:        distance( $v_k, v_p$ ) = 3.
32:        distance( $v_p, v_k$ ) = 3.
33:      end if
34:    end for
35:  end for
36: end for
37: return distance

```

position embedding in the transformer to encode order information. For all negative sampling in our model, we randomly choose five negative samples in the uniform distribution over the corpus. To construct the item–attribute graph, we utilize the metadata of products to build edges between nodes. The distance threshold τ_{val} is set to 3 and τ_{pad} is chosen from $\{4, 5\}$.

To measure the distances in the graph, we visit the edges and the neighbors of their corresponding nodes to avoid loops on nodes in algorithms such as Floyd. The algorithm is shown in Algorithm 1 and its time complexity is $O(|\mathcal{E}|d^2)$ compared with the Floyd’s $O(|\mathcal{V}|^3)$, where d is the average degree of nodes.

4.3 JDsearch Dataset

4.3.1 Dataset Collection. Different from the stimulated scenario, in real situations, users may issue a variety of queries based on the different attributes such as titles, brands, and categories to

search products. The simple pseudo query generation process in the Amazon datasets cannot mimic the real search behaviors well. Besides, the Amazon sub-category dataset has also been cleaned by excluding long-tail products from user history and keeping only the products that belong to one single category. This is different from the real situation where diverse and sparse user behaviors are common. Therefore, we use the JDsearch dataset [23], which collects real user queries and has no restrictions on products or users, to conduct our experiments to test the approaches in a more realistic situation. Furthermore, from the statistics shown in Table 1, we can find that the JDsearch dataset is much sparser than the Amazon dataset, which is more challenging for models.

4.3.2 Evaluation. We use the test queries and their candidate products in the JDsearch dataset as the test set and utilize their behavior histories as the training set. Specifically, the queries and items in the test set are exactly the last issued and interacted items of each user. Different from the full ranking strategy in the Amazon datasets consistent with previous approaches [3], we only rank the candidate products of the test queries in the JDsearch dataset. We also use MRR, NDCG@10, and Precision@1 as metrics to evaluate the ranking results.

4.3.3 Implementation Details. For privacy reasons, this dataset lacks the review information of users and products. The textual elements such as queries and product titles in the dataset are also anonymized and only word IDs are provided. It is challenging to optimize the word embeddings from the beginning because this dataset is much more sparse than the Amazon datasets. Therefore, word2vec is used to initialize the word embeddings in models. Additionally, unlike the conditions in the Amazon datasets, the products in the test set are not guaranteed to be observed during training, so simply optimizing the product embedding table as existing approaches may cause severe overfitting. As a result, we derive the product representation by calculating their semantic-based vectors. Specifically, we compute the product vectors using the similar non-linear function in Equation (2) based on their titles, brands, and categories. Given the absence of reviews, we remove the generative language modeling loss for items and users from all models. Thus, in HEM [3], DREM [4], we directly apply the non-linear function to create user embeddings from all the terms in their historical interacted items. For KG-based DREM, HGN [2], and our BATA model, as the JDsearch dataset doesn't provide us with the relationships between products (such as "bought together" and "also bought"), so we only take the category and brand information to construct the item-attribute graph. For the hyper-parameter settings, the settings of models are the same as the configurations in the Amazon datasets, except that we train the models for 30 epochs and the length of user histories is set to 50.

4.4 Baselines

We compare several baselines in *ad hoc* and personalized product search:

(1) *Ad Hoc Product Search Method*

LSE: LSE [19] is a non-personalized model. It adopts the latent space model to learn the representation of items.

(2) *General Personalized Product Search Methods*

HEM: HEM [3] is a latent vector-based personalized model. It builds users' and items' vectors by generative language models on reviews.

DREM: DREM [4] is a KG-based personalized model. It utilizes the metadata of items to establish a KG between users, items, and several attributes including categories and brands.

(3) *Sequential Personalized Product Search Methods*

AEM, ZAM: AEM [1] is an attention-based personalized model. It aggregates the user's historical interacted items with the current query to construct a query-specific user profile.

Table 2. Overall Performances on the Amazon Datasets

| Model | Amazon Dataset | | | | | | | | | | | |
|---|------------------------------------|--------------------------|--------------------------|------------------------------------|--------------------------|--------------------------|----------------------------|--------------------------|--------------------------|--------------------------|--------------------------|--------------------------|
| | <i>Cell Phones and Accessories</i> | | | <i>Clothing, Shoes and Jewelry</i> | | | <i>Sports and Outdoors</i> | | | <i>Electronics</i> | | |
| | MRR | NDCG | Prec | MRR | NDCG | Prec | MRR | NDCG | Prec | MRR | NDCG | Prec |
| <i>Non-Personalized Product Search Method</i> | | | | | | | | | | | | |
| LSE | 0.013 | 0.011 | 0.002 | 0.007 | 0.008 | 0.003 | 0.007 | 0.006 | 0.002 | 0.030 | 0.041 | 0.001 |
| <i>General Personalized Product Search Methods</i> | | | | | | | | | | | | |
| HEM | 0.027 | 0.027 | 0.014 | 0.009 | 0.010 | 0.005 | 0.002 | 0.001 | 0.001 | 0.104 | 0.116 | 0.047 |
| DREM | <u>0.100</u> | 0.106 | <u>0.049</u> | 0.021 | 0.023 | 0.006 | 0.043 | 0.043 | 0.021 | 0.149 | 0.155 | 0.078 |
| <i>Sequential Personalized Product Search Methods</i> | | | | | | | | | | | | |
| AEM | 0.044 | 0.050 | 0.021 | 0.006 | 0.007 | 0.002 | 0.003 | 0.001 | 0.001 | 0.121 | 0.136 | 0.078 |
| ZAM | 0.049 | 0.056 | 0.021 | 0.005 | 0.005 | 0.001 | 0.008 | 0.008 | 0.004 | 0.135 | 0.143 | 0.098 |
| TEM | 0.061 | 0.052 | 0.038 | 0.022 | 0.024 | 0.008 | 0.071 | <u>0.079</u> | 0.032 | 0.102 | 0.118 | 0.056 |
| CoPPS | 0.094 | 0.097 | 0.042 | 0.027 | 0.030 | 0.012 | 0.073 | 0.084 | 0.034 | 0.128 | 0.137 | 0.066 |
| HGN | 0.092 | 0.108 | 0.033 | 0.030 | 0.033 | <u>0.014</u> | 0.073 | 0.078 | 0.028 | 0.140 | 0.152 | 0.073 |
| <i>Our Methods</i> | | | | | | | | | | | | |
| BATA-v | <u>0.100</u> | 0.116 | 0.049 | 0.033 [†] | 0.038 [†] | 0.013 | 0.075 [†] | 0.089[†] | 0.038 [†] | 0.160 [†] | 0.166 [†] | 0.117 [†] |
| BATA | 0.137[†] | 0.164[†] | 0.061[†] | 0.038[†] | 0.043[†] | 0.016[†] | 0.081[†] | 0.089[†] | 0.042[†] | 0.171[†] | 0.189[†] | 0.120[†] |

The best and the second results are denoted in bold and underlined fonts, respectively.

[†]Indicates the model outperforms all baselines significantly with paired t -test at $p < 0.05$ level.

ZAM improves AEM by concatenating a zero vector to the item list to adjust the extent of conducting personalization.

TEM: TEM [7] is a transformer-based personalized model. It directly concatenates the current query at the end of purchased item lists of users and feeds them into the transformer to capture user profiles.

CoPPS: CoPPS [13] is a transformer-based personalized model. It proposes several contrastive learning tasks to improve the transformer-based model's performance. The original CoPPS utilizes fine-grained product term representations to build user profiles, which is slow and inefficient. For a fair comparison, we use the TEM model as the backbone model in CoPPS in our experiments.

HGN: HGN [2] integrates *DREM* and *ZAM* models. It uses the relations in KG to boost the representations of products and users.

BATA-v, *BATA*: These are our models. In *BATA-v* (*BATA-vanilla*), we initialize the embeddings with random values, while in *BATA*, we initialize them by the KG representations in Section 3.2.

5 Experimental Results

We present the overall results in Section 5. To verify the effectiveness of each module in the *BATA* model, we perform an ablation study in Section 5.2 and a knowledge initialization study in Section 5.3. We analyze the decoders in the *BATA* model in Section 5.4. Finally, we conduct a case study of the biased attention in Section 5.6.

5.1 Overall Results

The overall results are shown in Tables 2 and 3. We can see that the performances on the Amazon datasets and commercial dataset show different patterns. So we will analyze them separately.

Table 3. Overall Performances on the JDsearch Dataset

| Model | JDsearch Dataset | | |
|---|--------------------------|--------------|--------------------------|
| | MRR | NDCG | Prec |
| <i>Non-Personalized Product Search Method</i> | | | |
| LSE | 0.177 | 0.172 | 0.072 |
| <i>General Personalized Product Search Methods</i> | | | |
| HEM | 0.196 | 0.191 | 0.085 |
| DREM | 0.165 | 0.158 | 0.063 |
| <i>Sequential Personalized Product Search Methods</i> | | | |
| AEM | 0.197 | 0.192 | 0.085 |
| ZAM | 0.197 | 0.192 | 0.085 |
| TEM | 0.223 | <u>0.219</u> | 0.105 |
| CoPPS | <u>0.225</u> | 0.220 | <u>0.108</u> |
| HGN | 0.166 | 0.159 | 0.063 |
| <i>Our Methods</i> | | | |
| BATA-v | 0.230[†] | 0.220 | 0.120[†] |
| BATA | 0.215 | 0.206 | <u>0.108</u> |

The best and the second results are denoted in bold and underlined fonts, respectively. [†]Indicates the model outperforms all baselines significantly with paired t-test at $p < 0.05$ level.

(1) On the Amazon dataset, compared with both the general and the sequential personalized product search models, our BATA model achieves significant improvements. BATA-v model has absolute improvements of 1.0%/1.9%, 1.5%/0.8%, 4.6%/0.5%, and 1.1%/2.9%, respectively, on four datasets in terms of NDCG over DREM/CoPPS model. The improvements over general models indicate that our approach is able to focus on the historical items related to current queries, resulting in more accurate user profiles than the general profiles for specific user intents. The comparisons between BATA-v and CoPPS models demonstrate that the prior bias term introduced in the attention provides useful guidance for item interactions and profile building. Combining these extra signals, our model can incorporate the out-of-item-sequence dependencies with original sequences. Besides, the designed decoders can ensure that the input information is incorporated into the user encoding.

The absolute improvements of our BATA model in terms of NDCG over the HGN model are 5.8%, 1.0%, 1.1%, and 3.7%, respectively, on four datasets. HGN model upgrades the ZAM model by enhancing the historical sequences with attributes of products such as brands and categories. These findings demonstrate the inferiority of the simple ensemble method for integrating general and sequential models in HGN. Besides utilizing the global KG to improve the representations of users and products, we also need to leverage external information such as relations in KGs to guide the combination of user history to better represent user preferences.

By initializing the input with KG embeddings, the overall performances of BATA are improved compared with BATA-v. It indicates that the pre-trained knowledge embeddings considering the rich relations between products in Amazon datasets can provide useful prior information for model training. These results show a promising knowledge initialization pipeline for training personalized product search models.

(2) *On the JDsearch dataset, our BATA-v model brings improvements over existing models.* We can observe that the BATA-v model outperforms all baselines, which verifies the effectiveness of our model under real scenarios. Different from Amazon datasets, the real queries in the JDsearch dataset can better represent user intents. As previous models ignore these queries to directly aggregate the user histories and conduct personalization, they may introduce noisy and irrelevant history behaviors into user profiles. However, our model can explicitly capture the query dependencies through the query dependency bias. Therefore, it can pay more attention to the historical products more related to current user intents and construct a more accurate user profile.

Besides, different from that of Amazon datasets, the KG-based models including DREM and HGN achieve lower results. The inconsistency is primarily due to the different richness levels of item-item relationships in the two datasets. In Amazon datasets, there exist adequate item-item relationships such as “bought together” and “also view” and review information, the KG representations trained on these datasets are effective. Therefore, they can helpfully initialize the representation of product and word embeddings, which yields better results of BATA. However, due to the sparse item-item relationships and absence of review in the JDsearch dataset, the quality of the KG representations is low. Besides, KG models also ignore query-query similarities in user sequences and most relations are query-independent. This makes the DREM and HGN models perform badly. This also leads to the poor performance of the knowledge initialization BATA model. For the performances of the CoPPS model, since the training data in the JDsearch dataset is more sufficient, contrastive learning based pre-training doesn’t bring too much improvement.

5.2 Ablation Study

In this section, we conduct an ablation study to verify the impact of modules. We use BATA-v as the backbone to avoid the knowledge initialization impact. The ablation models include:

w/o. Global Relation (GR) and w/o. Query Dependency (QD). We remove the distance matrix M^G between items in the item-attribute graph and the query similarity matrix M^Q from the attention bias term respectively.

w/o. QD + Add Query Embedding (AQ). Based on the *w/o. QD* model, we add the historical query embedding Q to the historical item embedding I as the input embedding to the user encoder.

w/o. Sequence Decoder (SD) and w/o. Graph Decoder (GD). We remove the sequence decoder module and graph decoder module respectively from our model.

The results of the ablation study are shown in Table 4. We find that removing the sequence decoder (*w/o. SD*) causes the slightest (or even no) performance drop among all ablations. The reason may be that the sequential information is fed directly into the encoder and is easy to reconstruct. From the ablation results on the JDsearch dataset, we can find that the improvements of our model mainly originate from the query dependency *QD*. Compared with the Amazon datasets, the real user queries in the JDsearch dataset can provide more useful information for user modeling. The *QD* module can provide external information about which part of history is relevant to the current user query. The efficacy of the *QD* module can also confirm our hypothesis in the analysis of performances on the JDsearch dataset in Section 5. However, since the pseudo queries in Amazon datasets are constructed from the category lists of items and may be already modeled into item embeddings, the improvements caused by *QD* module are not as significant as those on the JDsearch dataset. Further, the graph decoder module *GD* offers additional self-supervision signals for product embedding fine-tuning on the JDsearch dataset. Different from the conclusion on the JDsearch dataset, the global relations *GR* among items in the encoder in the BATA-v model contribute the most improvement on Amazon datasets, as there are richer product-product relations compared with those of the JDsearch dataset. In model *w/o. QD + AQ*, we can observe that simply adding the item embeddings and query embeddings together is worse than the disentangling strategy adopted

Table 4. Performance of Ablation Models in NDCC

| Model | <i>Cellphones</i> | <i>Clothing</i> | <i>Sports</i> | <i>Electronics</i> | JDsearch |
|-----------|-------------------|-----------------|---------------|--------------------|--------------|
| BATA-v | 0.116 | 0.038 | 0.089 | <u>0.166</u> | <u>0.220</u> |
| w/o GR | 0.045 | 0.027 | 0.068 | 0.112 | 0.219 |
| w/o QD | 0.086 | 0.029 | 0.068 | 0.161 | 0.183 |
| w/o SD | <u>0.106</u> | <u>0.035</u> | <u>0.083</u> | 0.168 | 0.219 |
| w/o GD | 0.057 | 0.029 | 0.071 | 0.151 | 0.213 |
| w/o QD+AQ | 0.085 | 0.028 | 0.058 | 0.154 | 0.232 |

The best and the second results are denoted in bold and underlined fonts, respectively.

in our model on Amazon simulated datasets. These results confirm that directly mixing up two heterogeneous embeddings is a sub-optimal solution as we mentioned in Section 3. Specifically, item embedding is calculated through one embedding table while query embedding is built from word embeddings. However, because the query and product embeddings are both semantic-based embeddings derived from the word vectors in the implementation and are homogeneous on the JDsearch dataset, simply adding them together may better integrate information. Therefore, the results of this ablation model on the JDsearch dataset are higher than the original BATA-v model.

5.3 Ablation Study with Knowledge Initialization

From the results of overall experiments and ablation study, we can conclude the main improvements of the BATA model on Amazon datasets originate from the introduced global relations between items in the dense KG. We introduce them in three different ways:

- Initializing embeddings by the KG model.
- Measuring the distances in the item–attribute graph as attention bias.
- Devising the knowledge decoder to rebuild the distance matrix.

To further investigate how these global relations between items benefit our model, we conduct an additional ablation study of the knowledge initialization model BATA. These ablation models include:

BATA w/o. (GR) and *BATA w/o. (QD)*: We remove these two parts but retain the knowledge initialization.

KE + TEM: We initialize the embeddings of products and words with knowledge embeddings in TEM as the BATA model does in Section 3.2. This method can be regarded as a naive version of the BATA model without attention bias and devised two decoders.

The results are shown in Table 5. First, we can see that both the global relation matrix M^G in Equation (12) and the graph decoder benefit the performance even if the global item–item relations have been modeled through KG previously. Removing them, the information in pre-trained embeddings cannot focus on the current sequences and will soon vanish during the training process. We need to introduce them to guide the aggregation process to boost user profiling. Second, our BATA model achieves better or comparable performances to the naive KE+TEM model. These results further infer the introduced modules are useful and beneficial for performance as supplementary support to the ablation studies in Section 5.2. The comparable results on *Clothing*, *Shoes and Jewellery* may be due to both the small scale of the datasets compared with *Electronics* and the sparse item relations (only 28.1% item pairs’ distances are lower than τ_{val}) in it. The small

Table 5. Performance of the Knowledge Initialized Ablation Models in NDCG on the Amazon Datasets

| | <i>Cellphones</i> | <i>Clothing</i> | <i>Sports</i> | <i>Electronics</i> |
|-------------------|-------------------|-----------------|---------------|--------------------|
| BATA | 0.164 | 0.043 | 0.089 | 0.189 |
| w/o GR | <u>0.152</u> | 0.033 | 0.061 | 0.170 |
| w/o GD | 0.126 | <u>0.034</u> | 0.065 | <u>0.180</u> |
| KE+TEM | 0.131 | 0.043 | <u>0.082</u> | 0.148 |
| Relation Sparsity | 71.3% | 28.1% | 32.8% | 18.4% |

The relation sparsity means the percentage of item pairs whose distances are lower than τ_{val} . The best and the second results are denoted in bold and underlined fonts, respectively.

size and rates give the KG model a chance to memorize all relations exactly and can easily transfer to current sequences without information loss.

5.4 Analysis of Decoders

In this section, we analyze the two auxiliary decoders in our model. We show the learning curves of losses in the training process in Figure 3. We can see similar general trends of personalized product search loss L_{ps} and two decoding losses L_{gd} and L_{sd} . This indicates that the personalization product search task and reconstruction task are not incompatible with each other. Moreover, after the personalization loss L_{ps} converges, the two decoding losses keep going down. These results confirm that decoding tasks can provide additional self-supervised signals as a supplement to the sparse personalization label for better training. Further, we show a decoding case in the test set in Figure 4. We can see that the decoded distance matrix is very similar to the ground truth distance. This result shows that our model has the ability to retain the external prior knowledge about global relations.

5.5 Complexity Analysis

In this section, we discuss both the time and space complexity of the proposed BATA model. Compared with previous personalized product search models such as TEM, the main additional modules of the BATA model are *two auxiliary decoders* and *the attention bias term*.

First, the two auxiliary decoders only participate in the model training and don't bring any additional computational costs in the inference. These two modules also only utilize the information that is already encoded in the main biased attention-based user encoder module without extra processing during training.

Second, for the attention bias term part, since we can previously calculate the relationship between items (not requiring running-time calculation and update) during both the training and inference process, the global item relation bias term M^G also doesn't introduce additional training computations. The computation costs of historical query dependency bias M^Q are similar to those of original attention matrix computation and these two matrices can be calculated in parallel. Furthermore, during online serving, since the word embeddings are fixed after training, the historical query representations are also fixed and can be computed and stored in advance. We only need to leverage the representations for current queries into the computations of the historical query dependency bias term M^Q . Therefore, the inference time complexity of our model is comparable to the previous transformer-based personalized product search model TEM.

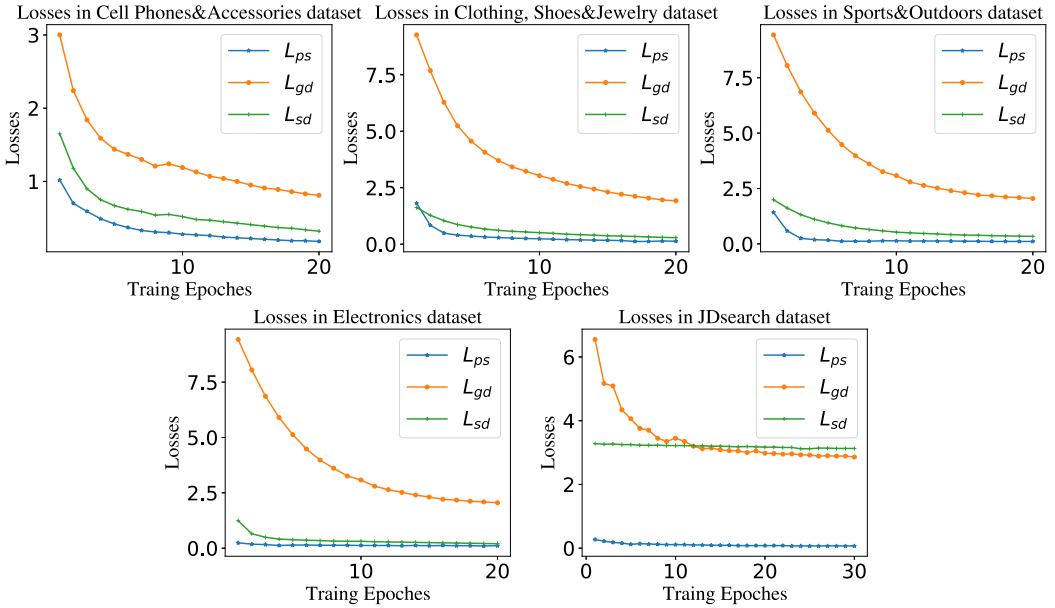


Fig. 3. The losses in all datasets.

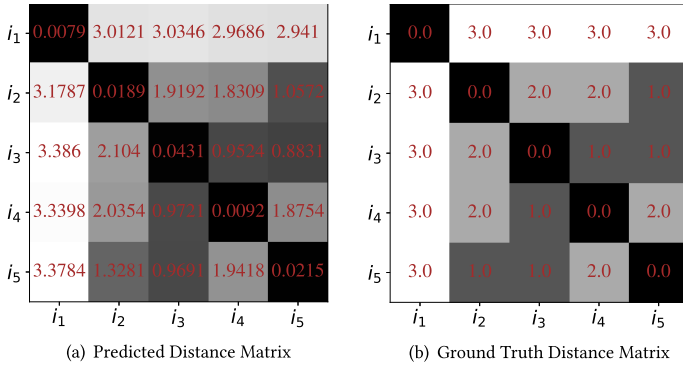


Fig. 4. Decoding case in *Cell Phones and Accessories* dataset.

For the training complexity, the overall training time on each dataset on 2 10G Titan V GPUs is less than 24 hours. We also compare the averaged training time on each step compared with the TEM model in Table 6. We can infer that the BATA model only has about a 1.2× increase in training time over the TEM model on Amazon datasets. The relative increase is slightly higher on the JDsearch dataset which may be caused by the larger size of this dataset as our model needs to process more information between items. In general, the additional training cost of our BATA model is applicable compared with previous sequential personalized product search models.

For the space complexity, compared with previous personalized product approaches, the main additional part is the precalculated item distance, which is relatively small considering the word and item embedding storage.

Table 6. Averaged Training Time on Each Step of TEM and BATA

| Model | Cellphones | Clothing | Sports | Electronics | JDsearch |
|-------------------|------------|----------|---------|-------------|----------|
| TEM | 29.61 s | 29.12 s | 23.42 s | 28.61 s | 19.20 s |
| BATA | 36.65 s | 34.88 s | 28.48 s | 32.27 s | 29.78 s |
| Relative Increase | 1.23× | 1.20× | 1.22× | 1.13× | 1.55× |

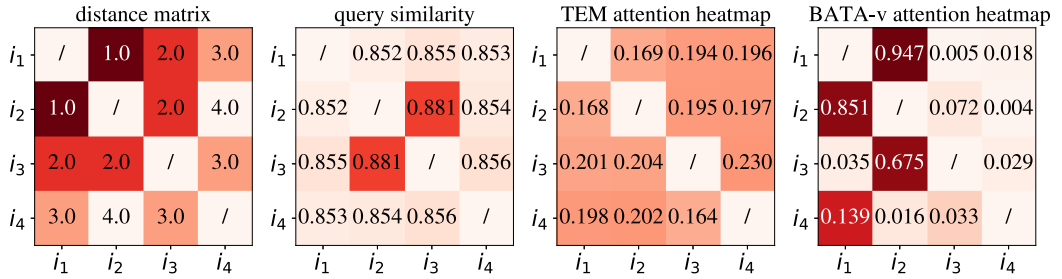


Fig. 5. An attention heatmap case in Amazon *Cell Phones and Accessories* dataset. The items include: i_1 : Samsung Galaxy S3 Replacement Battery; i_2 : Sports Gym Jogging Armband for Samsung Galaxy S3 III; i_3 : USB Charger for Samsung Galaxy Note; and i_4 : InvisibleShield for Motorola Droid RAZR.

5.6 Case Study of Attention

In this section, we show a case of attention weights in the TEM model and the BATA-v model on the same user historical item sequence. The product distance matrix M^G , query similarity matrix M^Q , and two attention heatmaps are shown in Figure 5. We can observe that the attention weight distribution in the TEM model is uniform, which implies similar interaction intensities among all items. However, our BATA-v model can perform stronger interactions among the first three items (they are all Samsung cellphone accessories), especially between i_1 and i_2 as they have been bought together before (their distance in the item–attribute graph is one). While two item pairs have the same distance in the graph, the BATA-v model can focus on the item pairs with larger query similarities (e.g., (i_3, i_2) has a larger attention weight than that of (i_3, i_1)). Therefore, our model can leverage external item–item and query–query relations to better guide item interaction and user profiling.

6 Conclusion

In this work, we proposed a sequential personalized product search model that utilizes additional item–item and query–query relations to guide the user profile modeling. We employ query similarity matrices and distances in the item–attribute graph based on metadata to represent these two types of external information. They are introduced as biases in the attention mechanism in the transformer encoder. Furthermore, we devise sequence decoders and graph decoders to reconstruct the sequence-based and graph-based input to force the user encoder to retain important information in user profile modeling. Experimental results confirm that our model can outperform existing approaches.

Acknowledgements

The work was partially done at the Engineering Research Center of Next-Generation Intelligent Search and Recommendation, MOE.

References

- [1] Qingyao Ai, Daniel N. Hill, S. V. N. Vishwanathan, and W. Bruce Croft. 2019. A zero attention model for personalized product search. In *28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.), ACM, 379–388. DOI : <https://doi.org/10.1145/3357384.3357980>
- [2] Qingyao Ai and Lakshmi Narayanan Ramasamy. 2021. Model-agnostic vs. model-intrinsic interpretability for explainable product search. In *30th ACM International Conference on Information and Knowledge Management (CIKM '21)*. Gianluca Demartini, Guido Zuccon, J. Shane Culpepper, Zi Huang, and Hanghang Tong (Eds.), ACM, 5–15. DOI : <https://doi.org/10.1145/3459637.3482276>
- [3] Qingyao Ai, Yongfeng Zhang, Keping Bi, Xu Chen, and W. Bruce Croft. 2017. Learning a hierarchical embedding model for personalized product search. In *40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. Noriko Kando, Tetsuya Sakai, Hideo Joho, Hang Li, Arjen P. de Vries, and Ryen W. White (Eds.), ACM, 645–654. DOI : <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), 1–29. DOI : <https://doi.org/10.1145/3361738>
- [5] Johannes Ballé, Valero Laparra, and Eero P. Simoncelli. 2017. End-to-end optimized image compression. In *5th International Conference on Learning Representations (ICLR '17)*. OpenReview.net. Retrieved from <https://openreview.net/forum?id=rjxdQ3jeg>
- [6] Paul N. Bennett, Ryen W. White, Wei Chu, Susan T. Dumais, Peter Bailey, Fedor Borisyyuk, and Xiaoyuan Cui. 2012. Modeling the impact of short- and long-term behavior on search personalization. In *35th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '12)*. William R. Hersh, Jamie Callan, Yoelle Maarek, and Mark Sanderson (Eds.), ACM, 185–194. DOI : <https://doi.org/10.1145/2348283.2348312>
- [7] Keping Bi, Qingyao Ai, and W. Bruce Croft. 2020. A transformer-based embedding model for personalized product search. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Jimmy Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.), ACM, 1521–1524. DOI : <https://doi.org/10.1145/3397271.3401192>
- [8] Keping Bi, Qingyao Ai, and W. Bruce Croft. 2021. Learning a fine-grained review-based transformer model for personalized product search. In *44th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '21)*. Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai (Eds.), ACM, 123–132. DOI : <https://doi.org/10.1145/3404835.3462911>
- [9] Antoine Bordes, Nicolas Usunier, Alberto García-Durán, Jason Weston, and Oksana Yakhnenko. 2013. Translating embeddings for modeling multi-relational data. In *27th Annual Conference on Advances in Neural Information Processing Systems 26*. Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger (Eds.), 2787–2795. Retrieved from <https://proceedings.neurips.cc/paper/2013/hash/1cecc7a77928ca8133fa24680a88d2f9-Abstract.html>
- [10] Haonan Chen, Zhicheng Dou, Yutao Zhu, Zhao Cao, Xiaohua Cheng, and Ji-Rong Wen. 2022. Enhancing user behavior sequence modeling by generative tasks for session search. In *31st ACM International Conference on Information & Knowledge Management*. Mohammad Al Hasan and Li Xiong (Eds.), ACM, 180–190. DOI : <https://doi.org/10.1145/3511808.3557310>
- [11] Dian Cheng, Jiawei Chen, Wenjun Peng, Wenqin Ye, Fuyu Lv, Tao Zhuang, Xiaoyi Zeng, and Xiangnan He. 2022. IHGNN: Interactive hypergraph neural network for personalized product search. In *ACM Web Conference 2022 (WWW '22)*. Frédérique Laforet, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.), ACM, 256–265. DOI : <https://doi.org/10.1145/3485447.3511954>
- [12] Kyunghyun Cho, Bart van Merriënboer, Çağlar Gülçehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. In *2014 Conference on Empirical Methods in Natural Language Processing (EMNLP '14)*. Alessandro Moschitti, Bo Pang, and Walter Daelemans (Eds.), ACL, 1724–1734. DOI : <https://doi.org/10.3115/v1/d14-1179>
- [13] Shitong Dai, Jiongnan Liu, Zhicheng Dou, Haonan Wang, Lin Liu, Bo Long, and Ji-Rong Wen. 2023. Contrastive learning for user sequence representation in personalized product search. In *29th ACM SIGKDD Conference on Knowledge Discovery and Data Mining (KDD '23)*. Ambuj K. Singh, Yizhou Sun, Leman Akoglu, Dimitrios Gunopulos, Xifeng Yan, Ravi Kumar, Fatma Ozcan, and Jieping Ye (Eds.), ACM, 380–389. DOI : <https://doi.org/10.1145/3580305.3599287>
- [14] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. 2019. BERT: Pre-training of deep bidirectional transformers for language understanding. In *2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies (NAACL-HLT '19)*. Jill Burstein, Christy Doran, and Tamar Solorio (Eds.), Association for Computational Linguistics, 4171–4186. DOI : <https://doi.org/10.18653/V1/N19-1423>
- [15] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. 2021. An image is worth 16x16 words:

- Transformers for image recognition at scale. In *9th International Conference on Learning Representations (ICLR '21)*. OpenReview.net. Retrieved from <https://openreview.net/forum?id=YicbFdNTTy>
- [16] Lu Fan, Qimai Li, Bo Liu, Xiao-Ming Wu, Xiaotong Zhang, Fuyu Lv, Guli Lin, Sen Li, Taiwei Jin, and Keping Yang. 2022. Modeling user behavior with graph convolution for personalized product search. In *ACM Web Conference 2022 (WWW '22)*. ACM, 203–212. DOI: <https://doi.org/10.1145/3485447.3511949>
- [17] 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), 1–27. DOI: <https://doi.org/10.1145/3295822>
- [18] 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 '18)*. Susanne Boll, Kyoung Mu Lee, Jiebo Luo, Wenwu Zhu, Hyeran Byun, Chang Wen Chen, Rainer Lienhart, and Tao Mei (Eds.), ACM, 1865–1873. DOI: <https://doi.org/10.1145/3240508.3240541>
- [19] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *25th ACM International Conference on Information and Knowledge Management (CIKM '16)*. Snehasis Mukhopadhyay, ChengXiang Zhai, Elisa Bertino, Fabio Crestani, Javed Mostafa, Jie Tang, Luo Si, Xiaofang Zhou, Yi Chang, Yunyao Li, et al. (Eds.), ACM, 165–174. DOI: <https://doi.org/10.1145/2983323.2983702>
- [20] Furkan Kocayusufoglu, Tao Wu, Anima Singh, Georgios Roumpos, Heng-Tze Cheng, Sagar Jain, Ed. H. Chi, and Ambuj K. Singh. 2022. Multi-resolution attention for personalized item search. In *15th ACM International Conference on Web Search and Data Mining (WSDM '22)*. K. Selcuk Candan, Huan Liu, Leman Akoglu, Xin Luna Dong, and Jiliang Tang (Eds.), ACM, 508–516. DOI: <https://doi.org/10.1145/3488560.3498426>
- [21] Chunyuan Li, Xiang Gao, Yuan Li, Baolin Peng, Xiujun Li, Yizhe Zhang, and Jianfeng Gao. 2020. Optimus: Organizing sentences via pre-trained modeling of a latent space. In *2020 Conference on Empirical Methods in Natural Language Processing (EMNLP '20)*. Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu (Eds.), Association for Computational Linguistics, 4678–4699. DOI: <https://doi.org/10.18653/v1/2020.emnlp-main.378>
- [22] Tianyang Lin, Yuxin Wang, Xiangyang Liu, and Xipeng Qiu. 2021. A survey of transformers. arXiv:2106.04554. Retrieved from <https://arxiv.org/abs/2106.04554>
- [23] Jiongnan Liu, Zhicheng Dou, Guoyu Tang, and Sulong Xu. 2023. JDsearch: A personalized product search dataset with real queries and full interactions. In *46th International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '23)*. Hsin-Hsi Chen, Wei-Jou (Edward) Duh, Hen-Hsen Huang, Makoto P. Kato, Josiane Mothe, and Barbara Poblete (Eds.), ACM, 2945–2952. DOI: <https://doi.org/10.1145/3539618.3591900>
- [24] Jiongnan Liu, Zhicheng Dou, Qiannan Zhu, and Ji-Rong Wen. 2022. A category-aware multi-interest model for personalized product search. In *ACM Web Conference 2022 (WWW '22)*. Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.), ACM, 360–368. DOI: <https://doi.org/10.1145/3485447.3511964>
- [25] Shang Liu, Wanli Gu, Gao Cong, and Fuzheng Zhang. 2020. Structural relationship representation learning with graph embedding for personalized product search. In *29th ACM International Conference on Information and Knowledge Management (CIKM '20)*. Mathieu d'Aquin, Stefan Dietze, Claudia Hauff, Edward Curry, and Philippe Cudré-Mauroux (Eds.), ACM, 915–924. DOI: <https://doi.org/10.1145/3340531.3411936>
- [26] Shuqi Lu, Di He, Chenyan Xiong, Guolin Ke, Waleed Malik, Zhicheng Dou, Paul Bennett, Tie-Yan Liu, and Arnold Overwijk. 2021. Less is more: Pretrain a strong siamese encoder for dense text retrieval using a weak decoder. In *2021 Conference on Empirical Methods in Natural Language Processing (EMNLP '21)*. Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih (Eds.), Association for Computational Linguistics, 2780–2791. DOI: <https://doi.org/10.18653/V1/2021.EMNLP-MAIN.220>
- [27] Qijie Shen, Hong Wen, Jing Zhang, and Qi Rao. 2022. Hierarchically fusing long and short-term user interests for click-through rate prediction in product search. In *31st ACM International Conference on Information & Knowledge Management*. Mohammad Al Hasan and Li Xiong (Eds.), ACM, 1767–1776. DOI: <https://doi.org/10.1145/3511808.3557351>
- [28] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. 2017. Attention is all you need. In *Annual Conference on Advances in Neural Information Processing Systems 30*. Isabelle Guyon, Ulrike von Luxburg, Samy Bengio, Hanna M. Wallach, Rob Fergus, S. V. N. Vishwanathan, and Roman Garnett (Eds.), 5998–6008. Retrieved from <https://proceedings.neurips.cc/paper/2017/hash/3f5ee243547dee91fbd053c1c4a845aa-Abstract.html>
- [29] Shuting Wang, Zhicheng Dou, Jing Yao, Yujia Zhou, and Ji-Rong Wen. 2023. Incorporating explicit subtopics in personalized search. In *ACM Web Conference 2023 (WWW '23)*. Ying Ding, Jie Tang, Juan F. Sequeda, Lora Aroyo, Carlos Castillo, and Geert-Jan Houben (Eds.), ACM, 3364–3374. DOI: <https://doi.org/10.1145/3543507.3583488>

- [30] Shuting Wang, Zhicheng Dou, and Yutao Zhu. 2023. Heterogeneous graph-based context-aware document ranking. In *16th ACM International Conference on Web Search and Data Mining (WSDM '23)*. Tat-Seng Chua, Hady W. Lauw, Luo Si, Evimaria Terzi, and Panayiotis Tsaparas (Eds.), ACM, 724–732. DOI : <https://doi.org/10.1145/3539597.3570390>
- [31] Bin Wu, Zaiqiao Meng, Qiang Zhang, and Shangsong Liang. 2022. Meta-learning helps personalized product search. In *ACM Web Conference 2022 (WWW '22)*. Frédérique Laforest, Raphaël Troncy, Elena Simperl, Deepak Agarwal, Aristides Gionis, Ivan Herman, and Lionel Médini (Eds.), ACM, 2277–2287. DOI : <https://doi.org/10.1145/3485447.3512036>
- [32] Teng Xiao, Jiaxin Ren, Zaiqiao Meng, Huan Sun, and Shangsong Liang. 2019. Dynamic Bayesian metric learning for personalized product search. In *28th ACM International Conference on Information and Knowledge Management (CIKM '19)*. Wenwu Zhu, Dacheng Tao, Xueqi Cheng, Peng Cui, Elke A. Rundensteiner, David Carmel, Qi He, and Jeffrey Xu Yu (Eds.), ACM, 1693–1702. DOI : <https://doi.org/10.1145/3357384.3358057>
- [33] Jingfeng Yang, Aditya Gupta, Shyam Upadhyay, Luheng He, Rahul Goel, and Shachi Paul. 2022. TableFormer: Robust transformer modeling for table-text encoding. In *60th Annual Meeting of the Association for Computational Linguistics (ACL '22)*. Smaranda Muresan, Preslav Nakov, and Aline Villavicencio (Eds.). Association for Computational Linguistics, 528–537. DOI : <https://doi.org/10.18653/V1/2022.ACL-LONG.40>
- [34] Chengxuan Ying, Tianle Cai, Shengjie Luo, Shuxin Zheng, Guolin Ke, Di He, Yanming Shen, and Tie-Yan Liu. 2021. Do transformers really perform badly for graph representation? In *Annual Conference on Advances in Neural Information Processing Systems 34 (NeurIPS '21)*. Marc'Aurelio Ranzato, Alina Beygelzimer, Yann N. Dauphin, Percy Liang, and Jennifer Wortman Vaughan (Eds.), 28877–28888. Retrieved from <https://Proceedings.neurips.cc/Paper/2021/Hash/f1c1592588411002af340cbaedd6fc33-Abstract.html>
- [35] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Encoding history with context-aware representation learning for personalized search. In *43rd International ACM SIGIR Conference on Research and Development in Information Retrieval (SIGIR '20)*. Jimmy X. Huang, Yi Chang, Xueqi Cheng, Jaap Kamps, Vanessa Murdock, Ji-Rong Wen, and Yiqun Liu (Eds.). ACM, 1111–1120. DOI : <https://doi.org/10.1145/3397271.3401175>
- [36] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Enhancing re-finding behavior with external memories for personalized search. In *13th ACM International Conference on Web Search and Data Mining (WSDM '20)*. James Caverlee, Xia (Ben) Hu, Mounia Lalmas, and Wei Wang (Eds.). ACM, 789–797. DOI : <https://doi.org/10.1145/3336191.3371794>

Received 1 July 2024; revised 4 March 2025; accepted 11 March 2025