IHGNN: Interactive Hypergraph Neural Network for Personalized Product Search

Dian Cheng[1*], Jiawei Chen[1*†], Wenjun Peng[1], Wenqin Ye[1],
Fuyu Lv[2], Tao Zhuang[2], Xiaoyi Zeng[2], Xiangnan He[1].
[1]University of Science and Technology of China, [2]Alibaba Group.
{cdboy@mail.,cjwustc@,pengwj@mail.}ustc.edu.cn,ywwwwq@pku.edu.cn,
{fuyu.lfy,zhuangtao.zt}@alibaba-inc.com,yuanhan@taobao.com,xiangnanhe@gmail.com.

ABSTRACT

A good personalized product search (PPS) system should not only focus on retrieving relevant products, but also consider user personalized preference. Recent work on PPS mainly adopts the representation learning paradigm, *e.g.,* learning representations for each entity (including user, product and query) from historical user behaviors (*aka.* user-product-query interactions). However, we argue that existing methods do not sufficiently exploit the crucial *collaborative signal*, which is latent in historical interactions to reveal the affinity between the entities. Collaborative signal is quite helpful for generating high-quality representation, exploiting which would benefit the learning of one representation from other related nodes.

To tackle this limitation, in this work, we propose a new model IHGNN for personalized product search. IHGNN resorts to a hypergraph constructed from the historical user-product-query interactions, which could completely preserve ternary relations and express collaborative signal based on the topological structure. On this basis, we develop a specific *interactive hypergraph neural network* to explicitly encode the structure information (*i.e.,* collaborative signal) into the embedding process. It collects the information from the hypergraph neighbors and explicitly models neighbor feature interaction to enhance the representation of the target entity. Extensive experiments on three real-world datasets validate the superiority of our proposal over the state-of-the-arts.

CCS CONCEPTS

• **Information systems → Personalization**.

KEYWORDS

Hypergraph; Personalized Product Search; Interaction

ACM Reference Format:
Dian Cheng, Jiawei Chen, Wenjun Peng, Wenqin Ye, Fuyu Lv, Tao Zhuang, Xiaoyi Zeng, Xiangnan He. 2022. IHGNN: Interactive Hypergraph Neural Network 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, 10 pages. https://doi.org/10.1145/3485447.3511954

∗ Dian Cheng and Jiawei Chen contribute equally to the work.
† Jiawei Chen is the corresponding author.

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

ACM ISBN 978-1-4503-9096-5/22/04…$15.00
https://doi.org/10.1145/3485447.3511954

1 INTRODUCTION

Online shopping pervades our daily lives. As the number of products in e-shopping platforms grows explosively, it is almost impossible for a user to discover desirable products without the help of product search engines. The search engine would retrieve a list of potential products for each user when he submits a query. The quality of the search results is crucial for both user satisfaction and retailer revenues.

Different from the traditional ad-hoc search task that focuses on finding the items matching the query, product search is more challenging as the target products are highly personalized [12, 29]. In a typical e-shopping scenario, it is common that users have quite different purchase intents even if they issue the same query. Take the query "Delicious Food" as an example, European users may expect some Pasta while Chinese users may be interested in dumplings. It is widely recognized that user purchases would be affected by their personalized preference [2, 38]. Therefore it is important for a product search engine to be personalized, with the goal to "understand exactly what the user wants and give him personalized suggestions" [13].

To achieve this goal, existing methods on personalized product search (PPS) [1, 2, 4, 12, 24, 42, 47, 48] mainly adopt the representation learning paradigm. They transform each entity (including user, product, and query) to a vectorized representation and then predict user purchase inclination based on the embeddings. Despite their decent performance, we argue that existing methods have not sufficiently exploited the *collaborative signal*. It is latent in historical user-product-query interactions to reveal the affinity among the entities, which is crucial for personalized search. For example, the users engaging with the same product may have similar preferences; the queries under which the same products are purchased by a user may have similar semantics. When equipped with such rich affinity information, the learning of one representation can benefit from other related ones, resulting in higher-quality representations. The work that is closest to ours is [22], however, it only exploits three manually-designed affinity patterns, which is far from sufficient. How to fully leverage the collaborative signal for PPS is still an open problem.

This work fills the research gap. Being aware of the effectiveness of graph neural network (GNN) for relational representation learning [15, 20], we wish to take its advantages for PPS. In traditional

search [30, 43] and recommendation [18, 37], the graph is bipartite that presents query-word matching and user-item matching. Performing graph convolution on these graphs can collect information from similar neighbors, which strengthens the representations with collaborative signal explicitly. However, the appealing idea is non-trivial to transfer to the PPS task, due to two main difficulties:

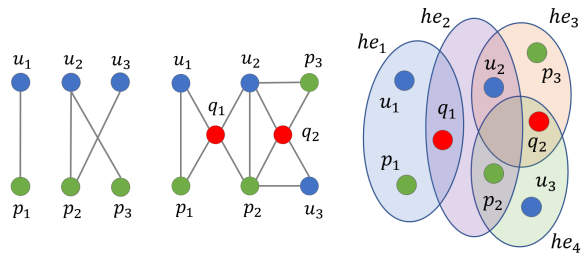
(P1) More complicated than traditional search and recommendation, the interactions in PPS are ternary rather than binary — each interaction involves three elements: user, product and query. It is intractable to construct a simple graph to preserve ternary relations. For example, if we forcibly split the ternary relations into three binary relations among users, products and queries [17], we will lose the information like under which query the user-product interaction happens. Figure 1(b) gives a toy example, we cannot determine under which query the interaction (u_2, p_2) happened: q_1, or q_2, or both of them. Since a simple graph cannot represent the ternary relations without loss, we need to resort to a more general topological structure to develop an effective personalized search method.

(P2) Existing GNNs mainly adopt a linear aggregation over the features of neighbors, ignoring the high-order feature interactions of neighbors. In fact, in PPS, the interactions between related entities could be a strong signal to indicate the characteristics of the target node. For example, when a user searches for "women's bag" and finally purchases a bag of the brand "Hermès", the interaction of the query and product would generate a quite useful semantic (*e.g.,* "women's luxury brands") for profiling the user's preference. We need to explicitly consider feature interaction to enhance the representation for PPS.

To tackle these problems, we propose to construct a *hypergraph* from the ternary user-product-query interactions. Compared with simple graph, hypergraph is a more suitable data structure for modeling ternary relations, because each hyperedge can connect any quantity of nodes. On this basis, we further propose a novel personalized product search model, named Interactive HyperGraph Neural Network (IHGNN), which recursively aggregates neighbor information along the aforementioned hypergraph. Distinct to the GNNs for recommendation or non-personalized search, our IHGNN makes two important improvements: (1) As each hyperedge in the hypergraph connects multiple nodes, IHGNN adopts a two-step information propagation scheme — node aggregation, which aggregates the information from the connected nodes to update the hyperedge representation; and hyperedge aggregation, which collects the information from the related hyperedges to update the target node representation. (2) As the neighbor interaction is important in PPS, we explicitly conduct high-order feature interaction of neighbors, and then aggregate the interacted results to enhance the target node representations.

In summary, this work makes the following contributions:

- We approach the PPS task with a user-product-query hypergraph and develop a hypergraph neural network to explicitly encode the collaborative signal into representation learning.
- We highlight the importance of exploiting feature interaction in representation learning, and propose to explicitly model high-order feature interactions of neighbors in hypergraph embedding aggregation.



(a) Bipartite graph (b) Collapsed Graph (c) Hypergraph

Figure 1: (a) gives a sample of user-product graph in recommendation, while (b)(c) give samples of collapsed graph and hypergraph in personalized product search. $u_i, q_i, p_i (i = 1, 2, 3)$ are nodes representing user, product or query respectively, $he_i (i = 1, 2, 3, 4)$ denotes a hyperedge shown as an ellipse in (c). Each user-query-product interaction corresponds to a hyperedge. There are four interactions: (u_1, q_1, p_1), (u_2, q_1, p_2), (u_2, q_2, p_3) and (u_3, q_2, p_2).

- We conduct extensive experiments on three real-world datasets to demonstrate the effectiveness and the rationality of each component design of IHGNN.

2 TASK FORMULATION AND PRELIMINARY

In this section, we first give the formal definition of personalized product search task, and then give some background knowledge related to hypergraph.

2.1 Personalized Product Search

Suppose we have a product search engine with a user set $\mathcal{U}$, an item set $\mathcal{P}$, a possible query set $\mathcal{Q}$ and a set of historical user-product-query interactions $\mathcal{L}$. Let u (p, or q) denote a user (an item, or a query) in $\mathcal{U}$ ($\mathcal{P}$, or $\mathcal{Q}$). $\mathcal{L}$ consists of a list of user-item-query triples (u, p, q), indicating user u has purchased[1] the product p under the query q. Also, we use $y_{upq} \in \{0, 1\}$ to indicate whether the interaction (u, p, q) happens, *i.e.,* $y_{upq} = 1$ for $(u, p, q) \in \mathcal{L}$ and $y_{upq} = 0$ for $(u, p, q) \notin L$. The goal of personalized product search is to learn a score function $f : \mathcal{U} \times \mathcal{P} \times \mathcal{Q} \rightarrow R$ to accurately predict the probability of a user u to purchase product p when searching query q.

2.2 Hypergraph

Different from simple graph, a hypergraph is a more general topological structure where a hyperedge could connect two or more nodes. Let $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathbf{H})$ be an instance of hypergraph, which includes a node set $\mathcal{V}$ and a hyperedge set $\mathcal{E}$. The $|\mathcal{V}| \times |\mathcal{E}|$ incidence matrix $\mathbf{H}$ describes the connectivity of the hypergraph, with entries defined as:

$$h(v, e) = \begin{cases} 1, & \text{if } e \text{ connects } v, \\ 0, & \text{if } e \text{ disconnects } v, \end{cases} \tag{1}$$

On this basis, we further give some notations in a hypergraph. For each node $v \in \mathcal{V}$, its degree is defined as $d(v) = \sum_{e \in \mathcal{E}} h(v, e)$; For

[1]We remark that here the "purchase" can be replaced by other type of implicit feedback such as " Click" or "Add-to-Cart". In this work we simply use the word "purchase" as a placehoder for better description.

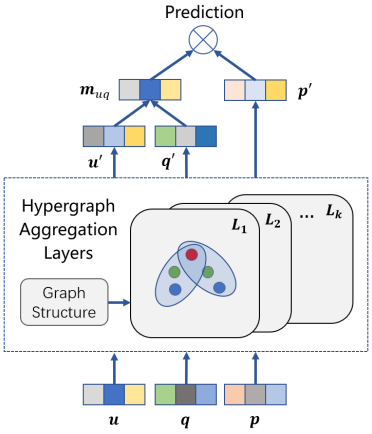


Figure 2: Illustration of the proposed IHGNN.

each edge $e \in \mathcal{E}$, its degree is $d(e) = \sum_{v \in \mathcal{E}} h(v, e)$. They can be further collected as two diagonal matrices D_v and D_e of node degrees and edge degrees, respectively. Let $\mathcal{E}_v$ denote a set of related hyperedges that connect to the node v (*i.e.,* $\mathcal{E}_v = \{e \in \mathcal{E} | h(v, e) = 1\}$), $\mathcal{V}_e$ denote a set of nodes to which the hyperedge e connects(*i.e.,* $\mathcal{V}_e = \{v \in \mathcal{V} | h(v, e) = 1\}$). Also, we can define the "neighbors" (N_v) of node v as a set of nodes that share at least one hyperedge with the node v (*i.e.,* $N_v = \{a \in \mathcal{V} | \exists e \in \mathcal{E}, h(v, e) = 1 \& h(a, e) = 1\}$).

3 PROPOSED METHOD: INTERACTIVE HYPERGRAPH NEURAL NETWORK (IHGNN)

In this section, we detail the proposed IHGNN for personalized product search. IHGNN aims at explicitly encoding collaborative signal into representation using user-product-query hypergraph. IHGNN contains four modules: (1) a hypergraph construction module that constructs hypergraph from the historical user-product-query interactions $\mathcal{L}$; (2) an embedding generation module that transforms the features of entities into their initial representations; (3) an aggregation module that refines the embeddings by collecting the information from neighbors and high-order neighbors; and (4) a prediction module that generates the prediction of user purchase inclination based on the refined embeddings. Finally, we discuss the properties of IHGNN and its connection with existing methods.

3.1 Hypergraph Construction

We first construct a hypergraph $\mathcal{G}$ based on historical ternary user-product-query interactions. Specifically, given historical interactions $\mathcal{L}$, we have a hypergraph $\mathcal{G} = (\mathcal{V}, \mathcal{E}, \boldsymbol{H})$ including: (1) a node set $\mathcal{V}$ whose element represents a user, product or query (*i.e.,* $\mathcal{V} = \mathcal{U} \cup \mathcal{Q} \cup \mathcal{P}$); (2) a hyperedge set $\mathcal{E}$, whose element represents a ternary relation depicting there exists an historical interaction among them (*i.e.,* $e \in \mathcal{E} \leftrightarrow (u, p, q) \in \mathcal{L}$, $h(u, e) = 1, h(p, e) = 1, h(q, e) = 1$).

The constructed hypergraph could preserve complete ternary relations and capture collaborative signal via topological structure. From which, we can easily deduce nodes' affinity based on their proximity in the hypergraph. Figure 1(c) gives a toy example. We can deduce user u_1 and u_2 may have similar preferences as they both connect to (issued) the query q_2. Such hypergraph provides an opportunity to fully exploit collaborative signal for PPS, *e.g.,* we can leverage hypergraph embedding to encode such important signal (*i.e.,* hypergraph structure) into representation. We will introduce details on how to utilize the hypergraph structure for enhancing PPS as follows.

3.2 Embedding Generation Module

This module aims at mapping entities into a common representation space. Here we follow previous work [34] to transform the features of queries to their representations. That is, for each query $q \in \mathcal{Q}$, we utilize the query content information and use a mean pooling over word embeddings z_w to generate the query embedding $z_q \in R^d$:

$$z_q = \frac{\sum_{w \in q} z_w}{|q|} \tag{2}$$

For each user $u \in \mathcal{U}$ (or product $p \in \mathcal{P}$), we directly generate its embedding $z_u \in R^d$ (or $z_p \in R^d$) from an embedding look-up table $\mathbf{E} \in R^{((|\mathcal{U}|+|\mathcal{P}|) \times d)}$ (*i.e.,* a parameter matrix).

3.3 Aggregation Module

In this module, we describe our embedding aggregation scheme, which iteratively aggregates the information from the neighbors to encode the collaborative signal into the embeddings. We first illustrate the design of one-layer aggregation and then generalize it to multiple successive layers.

3.3.1 First-order Aggregation. Intuitively, hypergraph neighbors provide useful signals to understand the target node's characteristics. For example, the interacted products or submitted queries provide direct evidences on a user's preferences; Analogously, relevant products are quite helpful to depict the semantics of the query. Thus, it is nature to collect the information from the neighbors to refine the representation of the target node. Distinct to the GNNs for recommendation [36] or non-personalized search [45], our IHGNN is conducted on hypergraph and thus adopts a two-step information propagation scheme — *i.e.,* hyperedges serve as mediums to process and transfer the information from the neighbors. To be more specific, for each node v, as shown in Figure 3(a), we perform the following two operations to update its representation:

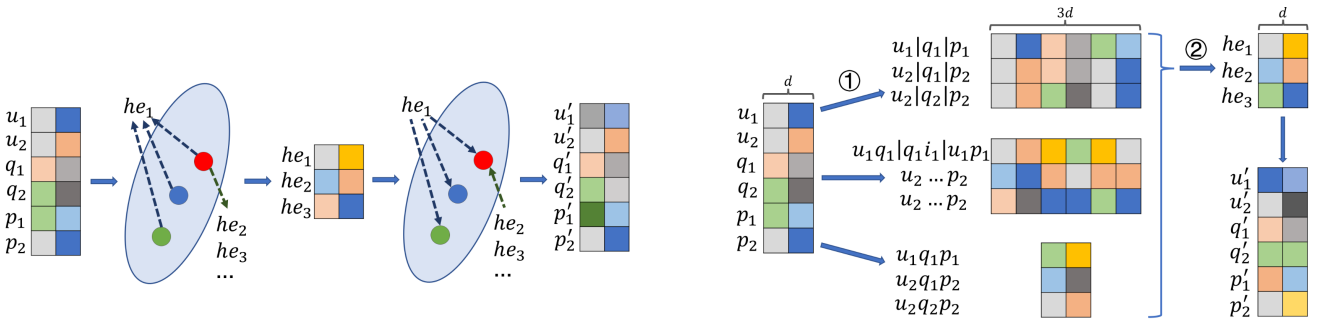
Node aggregation. In this stage, we aggregate the information propagated from v's neighbors to related hyperedge. Specifically, for each $e \in \mathcal{E}_v$, we define the aggregation function as follow:

$$m_e = [z_u \,||\, z_p \,||\, z_q]\mathbf{W} \tag{3}$$

where m_e denotes the information contained by the medium e, u, p, q denote the user, product, query node that e connects, $||$ denotes the concatenation operation, $\mathbf{W} \in \mathbb{R}^{3d \times d}$ is a trainable weight matrix to distill useful knowledge from the neighbor nodes. Here we choose linear transformation instead of the conventional simple average, as the hyperedge would collect different types of nodes and we believe they may make different contributions.

Hyperedge aggregation. In this stage, we aggregate the information from the related hyperedge to refine the representation of v as:

$$z_v^{(1)} = \frac{\sum_{e \in \mathcal{E}_v} m_e}{d(v)} \tag{4}$$



(a) vanilla hypergraph propagation

(b) Improved hypergraph propagation with higher-order interaction

Figure 3: Illustration of how our IHGNN aggregates the information from neighbors.

With such two operations, the knowledge of neighbor nodes has been explicitly injected into the target node's representation, reinforcing its quality.

Modeling neighbor feature interaction. However, we argue that such linear aggregation scheme is insufficient, ignoring the knowledge from the feature interaction among the neighbors. In fact, in PPS, the neighbor interaction could be a strong signal that indicates the characteristics of the target node. For example, provided a user searches for "women's bag" and finally purchases a bag of the brand "Hermès", the interaction between the query and the product would generate a quite useful semantic (*e.g.,* "women's luxury brands") for profiling the user's preference. Based on this point, we further develop an improved node aggregation operation as shown in Figure 3(b), which explicitly models the feature interaction among the nodes (*i.e.,* $u, p, q \in \mathcal{V}_e$) that the hyperedge e connects. We have:

$$\begin{aligned} f_e^{o1} &= z_u \,||\, z_q \,||\, z_p \\ f_e^{o2} &= (z_u \odot z_q) \,||\, (z_u \odot z_p) \,||\, (z_q \odot z_p) \\ f_e^{o3} &= z_u \odot z_q \odot z_p \end{aligned} \tag{5}$$

Where $\odot$ stands for element-wise product, $f_e^{o1}, f_e^{o2}, f_e^{o3}$ respectively capture 1,2,3-order feature interaction of neighbors. Analogously, we use a linear layer to aggregate the interaction information as:

$$m_e = [f_e^{o1} \,||\, f_e^{o2} \,||\, f_e^{o3}]\, \boldsymbol{W}_{o3} \tag{6}$$

This useful information will be further propagated into the target node representation via the hyperedge aggregation. We remark that here we just conduct feature interaction among the neighbors that share common hyperedge (*i.e.,* $u, p, q \in \mathcal{V}_e \& e \in \mathcal{E}_v$) rather than among all neighbors (*i.e.,* $u, p, q \in \mathcal{N}_v$). The reason is that these specific neighbors are highly related and their interactions potentially generate strong signal. Also, this treatment is much more efficient. Conducting feature interaction among all neighbors is usually computationally unavailable.

3.3.2 Higher-order Aggregation. To completely exploit the collaborative signal, we further consider to stack aforementioned aggregation module such that the target node's representation could benefit from the high-order neighbors. In fact, although these nodes are not directly connected with the target node, they indeed share some similarity and provide useful knowledge to learn the target representation. Concretely, As Figure 3(b) displays, in the l-th layer, the representation of the node v is recursively updated as follow:

$$z_v^{(l)} = \frac{\sum_{e \in \mathcal{E}_v} m_e^{(l)}}{d(v)} \tag{7}$$

where the hyperedge information $m_e^{(l)}$ can be calculated from $z_u^{(l-1)}$, $z_p^{(l-1)}$,$z_q^{(l-1)}$ with equation (3) or (6). Finally we concatenate the representations learned by different layers to generate the entity's final representation:

$$z_v^* = z_v^{(0)} || z_v^{(1)} || ... || z_v^{(L)} \tag{8}$$

By doing so, we enrich the initial embeddings with the information propagated from similar (high-order) neighbors. The collaborative signal has been explicitly injected in the representation.

3.4 Prediction Module

This model aims at generating prediction based on the learned embeddings. As this module is not our focus, we simply refer to [2] for implementation. It is relatively simple but proved effective. Specifically, we estimate the purchase probability of user u towards the product p when searching query q as follow:

$$\hat{y}_{uqp} = \text{sigmoid}\left((\lambda z_u^* + (1-\lambda) z_q^*)^T z_p^* \right) \tag{9}$$

where λ is a hyper-parameter controlling the contribution from the user preference and item relevance.

3.5 Model Optimization

For fairly comparison, we closely follow the related work [22] to learn the model parameters. Specifically, we optimize the following objective function:

$$L = \sum_{(u,q,p) \in \mathcal{L} \cup \mathcal{B}^-} -y_{uqp} \cdot \log \hat{y}_{uqp} - (1 - y_{uqp}) \cdot \log(1 - \hat{y}_{uqp})$$

where $\mathcal{B}^-$ denotes a negative sampled set that contains the triples with $y_{uqp} = 0$.

3.6 Discussion

3.6.1 Connection with existing PPS methods. First, we compare our IHGNN with GraphSRRL [22]. GraphSRRL explicitly considers three manually-designed affinity patterns — *e.g.,* Provided there

are two interactions (u_1, q, p), (u_2, q, p), u_1 and u_2 may have similar preference. In fact, these affinity patterns can adaptively be captured by our model. With embedding aggregation, the information of z_q, z_p as well as their interactions $z_q \cdot z_p$ can be propagated into the representation of u_1 and u_2, making u_1 and u_2 hold a certain similarity. Similar analysis can be conducted for other patterns. Besides such three patterns, our IHGNN captures more and thus yields better performance than GraphSRRL as reported in our experiments.

Also, it is worth to discuss the connections with some related work [7, 13] that models Long Short-Term Preference. These methods would deduce user long-term (or short-term) preference from his purchased products (or submitted queries). In fact, it can be considered as a special way of utilizing collaborative signal. Our IHGNN can also capture this pattern in the one-layer aggregation. We would collect the information from the neighbors to update the representation.

3.6.2 Connection with existing hypergraph neural network. Recent years also witnessed some work [11, 41] on hypergraph neural network. These methods derive HGNN from spectral convolution and it has similar propagation scheme as our IHGNN. But IHGNN differs from HGNN in the following two aspects: (1) when performing node aggregation, we give different weights for different types of nodes; (2) we explicitly model feature interaction of neighbors, which is of importance in PPS.

3.6.3 Connection with graph neural network. Here we mainly compare our IHGNN with the GNNs on the collapsed graph, where we split the ternary relations into three binary relations. The key difference is in that our IHGNN utilize a medium (*i.e.,* hyperedge) to process and transfer information, such that the ternary relation can be treated in a holistic view; while GNN can only handle the fragment information. It makes our IHGNN usually achieve better performance than GNNs on the collapsed graph.

4 EXPERIMENTS

In this section, we conduct experiments to validate the effectiveness of our IHGNN. Our experiments are intended to address the following research questions:

- **RQ1**: Does IHGNN outperform state-of-the-art PPS methods?
- **RQ2**: How do different components (*e.g.,* using hypergraph, weighted propagation, high-order interaction) contribute to the model performance?
- **RQ3**: How do different hyper-parameters (*e.g.,* depth of aggregation L, and embedding size d) affect the performance of IHGNN?

4.1 Datasets

We utilize three datasets in our experiments, including one real-world dataset that collected from real PPS scenario and two available conventional semi-synthetic dataset.

AlibabaAir dataset. The dataset is collected from taobao.com, one of the biggest e-shopping platforms in China. We randomly sampled 200,000 users and collect their search logs on the platform from 23-Dec-2020 to 29-Dec-2020. The dataset contains the display information of user-query-product, and labels denoting whether the user clicked the product. Approximately 930,000 clicks and 10,000 purchases are contained. This dataset also contains basic query word segmentation results. Users, query words/characters, products were all encrypted as id numbers for privacy protection. We conduct 5-core and 1-core filtering (*i.e.,* retaining entities with at least 5 or 1 interactions) to generate two datasets, marked as Ali-1Core and Ali-5Core, respectively. This treatment would like to show the robust of our model on cold start users (or items).

Table 1: Dataset Statistics.

	users	queries	products	interactions
Ali-1Core	200,000	102,816	220,779	940,946
Ali-5Core	39,976	46,098	39,326	403,982
CIKM	23,882	3,256	23,550	339,341
CDs_5	112,379	509	67,602	1,296,885

CIKMCup dataset. This is a product search dataset from CIKMCup2016 Track2. However, over half of the search logs are anonymous and do not have user ids. Another shortcoming is that most queries are manually composed according to category browsing. In our experiments, in order to use enough data, we preserve the category browsing results, click data, and view data after we filter out the anonymous search logs. Also, 5-core filtering has been conducted. Note that GraphSRRL [22] directly used the default recommendation results as ground truth, which may not be consistent with user true preference and thus is not adopted by us.

Amazon simulated dataset. This is a review dataset consisting of users' reviews on products. It is used by [2] in earlier research of PPS task. A user-product review pair is extended to a user-query-product triple by treating the product's category string as a query. Since Amazon datasets are semi-synthetic and are not as convincing as other datasets (AlibabaAir) adopted in this work, here we simply choose one typical sub-dataset (*i.e.,* CDs_5) for experiments.

The statistics of three datasets is shown in Table 1.

4.2 Compared methods

The following methods are tested in our experiments:

- **LSE** [30] is a classic latent vector space model for product search. LSE learns word and item representations in one latent vector space and directly models the match score between products and query words. It does not consider to model users and therefore is non-personalized.
- **HEM** [2] extends LSE [30] with personalization setting. HEM adopts representation learning paradigm. It learns user, product and query representation and then utilize an inner product function to make a prediction.
- **ZAM** [1] extends HEM to determine how much personalization is needed.
- **TEM** [4] is a transformer-based embedding model for personalized product search. It improves ZAM by adding the ability of dynamically controlling the influence of personalization. Its core idea is to encode the sequence of query and user's purchase history with a transformer architecture,

Table 2: Performance comparisons for personalized product search. The boldface font denotes the winner in that column. Also, the best baselines are marked with †. The row 'Gain' indicates the relative performance gain of our IHGNN compared to the best baseline. '*' and '' denote the improvement is significant with t-test with $p < 0.05$ and $p < 0.1$, respectively.**

	Ali-1Core			Ali-5Core			CIKM			CDs_5		
	HR@10	NDCG@10	MAP@10	HR@10	NDCG@10	MAP@10	HR@10	NDCG@10	MAP@10	HR@10	NDCG@10	MAP@10
LSE	0.1373	0.0848	0.0870	0.1979	0.1257	0.1343	0.3147	0.2215	0.2735	0.1300	0.0696	0.0515
HEM	0.1490	0.0957	0.0980	0.2238	0.1485	0.1610	0.4100	0.3541	0.4601	0.1409	0.0762	0.0567
ZAM	0.1412	0.0915	0.0978	0.2559	0.1749	0.1916	0.3767	0.3273	0.4491	0.0938	0.0482	0.0345
TEM	0.1320	0.0916	0.1020	0.2493	0.1758	0.1939	0.4471	0.3576	0.4509	0.1292	0.0666	0.0478
GraphSRRL	0.1306	0.0872	0.0952	0.2445	0.1659	0.1811	0.4729	0.3729	0.4700	0.1577	0.0846	0.0625
GCN	0.1773	0.1177	0.1221	0.2549	0.1743	0.1876	0.4424	0.3557	0.4394	0.1451	0.0785	0.0584
GAT	0.1908†	0.1264†	0.1329†	0.2715†	0.1858†	0.2018†	0.5238†	0.4211†	0.5217†	0.1652†	0.0873†	0.0638†
HyperGCN	0.1803	0.1180	0.1212	0.2655	0.1826	0.1986	0.4100	0.3474	0.4430	0.1396	0.0755	0.0561
IHGNN-O3	**0.2073***	**0.1465***	**0.1591***	**0.2894***	**0.2091***	**0.2307***	**0.5514***	**0.4855***	**0.6314***	**0.1672***	**0.0893****	**0.0657****
Gain	8.6%	15.9%	19.7%	6.6%	12.5%	14.3%	5.3%	15.3%	21.0%	1.2%	2.3%	3.0%

which gives different personalization weights to different history items.

- **GraphSRRL** [22] improves HEM with utilizing three manual-designed affinity patterns to enhance the representation learning. Because GraphSRRL demonstrate superiority over DREM [3], we do not include DREM as baselines.
- **GCN [20], GAT [33], HyperGCN [11]**: We also design three quite competitive baselines — utilizing graph-based methods for enhancing representation learning. GNN and GAT are conducted on collapsed graph, while HyperGCN is conducted on the same hypergraph as ours. We remark that here we do not compare some other graph-based methods [26], as they require other side information, which is unfair.
- **IHGNN**: the proposed method in this paper. We test different version of IHGNN, where 'IHGNN-O*' denotes the model considering different maximum orders of feature interaction.

4.3 Experimental Setup

4.3.1 Data split. We split the dataset into 3 parts according to the search time of user interaction: 70% for training, 10% for validation, and 20% for testing.

4.3.2 Evaluation. For each test data (user, query, products), we use the top-10 setting, where the top-10 ranking results are utilized to calculate the evaluation metrics. We use three conventional metrics: hit ratio (HR), normalized discounted cumulative gain (NDCG) and mean average precision (MAP). HR focuses on the retrieval performance by calculating the ratio of positive products appearing in search results. NDCG and MAP are common metrics for ranking algorithms.

4.3.3 Implementation Details. We implement our IHGNN [2] with PyTorch. The embedding size is set to 32. We randomly sample 10 negative products for one user-query-product interaction. We use Adam optimizer with learning rate 0.001 to optimize all models, where the batch size is 100. The embeddings are initialized by using the default Xavier uniform method and other parameters are initialized by the Kaiming uniform method. All graph-based models, including GraphSRRL, GCN, GAT, HyperGCN, and IHGNN, use 2 graph layers. Models are trained on i9-9900X CPU and 2080Ti GPU. In training process, we calculate the metrics on validation and test dataset. We choose the test metrics according to the best NDCG@10 result on validation set. We train all models until they converge for fair comparison. Hyper-parameter λ is set to 0.5.

[2]https://github.com/CDboyOne/IHGNN

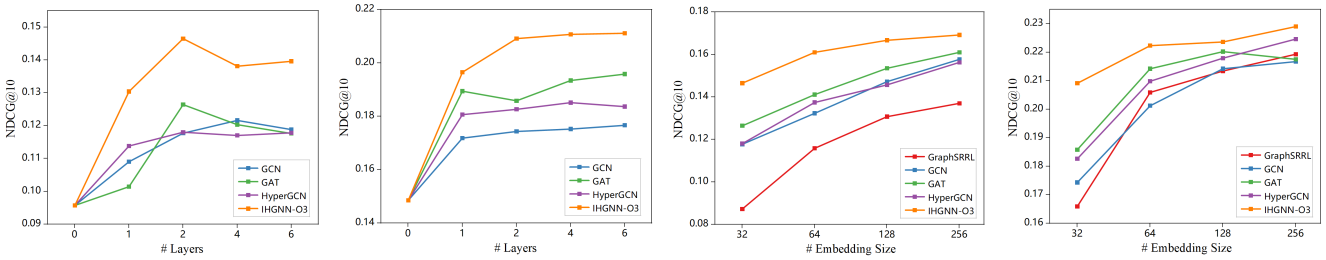
4.4 Results and Analysis (RQ1)

Table 2 presents the performance our IHGNN comparing with baselines. From the results, we have the following observations:

Personalized Vs. non-Personalized. LSE has the worst performance. HEM is a personalized extension of LSE and has much better performance, which demonstrates that personalization is effective. This result is easy to understand because product search is a very personal behavior.

Graph-based Vs. not Graph-based. Among all the baselines, methods based on graph usually have better performance. For example, on Ali-1Core, GCN performs 23.0% better than methods without graph and GAT performs 32.1% better in terms of NDCG@10. On other datasets, GCN and HyperGCN usually have good performance, and GAT always performs better than other baselines. This is because: 1) GNNs describe the relation among user, query and product very well, which is helpful for better personalization. 2) GNNs naturally model the interaction among users, queries, or products through multi-hop neighbors. HEM, ZAM and TEM does not exploit interactions in the user-query-product triples, and the meta-path-like method used by GraphSRRL does not directly utilize the interaction relation.

IHGNN Vs. Others. Our proposed IHGNN significantly outperforms the baselines on all datasets. This is because: 1) Hypergraphs completely preserve the information in triple interaction data. Although HyperGCN does not outperform GCN at a significant level on Ali-1Core, the improvement from HyperGCN to IHGNN is much more than improvement from GCN to GAT. This indicates that the structure of hypergraph provides better ability to mine the personalization information in data. 2) We introduce weights in node aggregation. As different type of nodes may carry different levels of information, using weights would improve the performance. Similar results can be seen from the better performance of GAT over



(a) Ali-1Core NDCG@10 (b) Ali-5Core NDCG@10 (c) Ali-1Core NDCG@10 (d) Ali-5Core NDCG@10

Figure 4: Performance with different numbers of graph layers and different embedding sizes on Ali-1/5Core dataset.

Table 3: Variants of IHGNN for ablations study. "Node" means node types in the graph, which contains options of user(u), query(q), and product(p). "Weighted prop." points out if the model has mechanism of weighted propagation. "Feature order" refers to the maximum feature interaction order of the model.

	Node	Weighted prop.	Feature order
IHGNN-up	u, p	×	1
IHGNN-qp	q, p	×	1
HyperGCN	u, q, p	×	1
IHGNN-O1	u, q, p	√	1
IHGNN-O2	u, q, p	√	2
IHGNN-O3	u, q, p	√	3

GCN. 3) Introduction of high-order feature interaction gives our model the ability to capture feature interactions that generally exist in the datasets.

HyperGCN Vs. GCN. To our surprise, we can find HyperGCN performs quite close to GCN, even performs worse in some datasets. This result validates that although the hypergraph carries rich collabarative signal, we need to carefully design the embedding model to enjoy the merit of it. Blindly use existing methods may not bring much performance gain. Our IHGNN is specifically designed for PPS and achieves better performance than HyperGCN and GCN.

Comparison in terms of datasets. As mentioned in section 4.1, real-world search data are usually very sparse. Therefore, we generate Ali-1Core to test models' performance on sparse data. From the table we can see that our IHGNN still yields a strong result. From Ali-5Core to Ali-1Core, baselines such as LSE and GraphSRRL decrease by 40.5% and 42.2% on average on HR@10 and NDCG@10. Baselines using GNN (GCN, GAT, HyperGCN) decrease by 30.8% and 33.3% on average. In contrast, our IHGNN only decrease by 28.4% and 29.9% on the two metrics. It shows that our model degrades less on sparser data.

4.5 Ablation Study (RQ2)

In Table 3, we listed different variants of IHGNN used for ablation study. The results of ablation study is shown in Table 4.

Contribution of weighted node aggregation. Note that the main difference between HyperGCN and IHGNN-O1 is whether to use weights in node aggregation. By comparing their results in Table 4, we observe that the weighted propagation brings big performance promotion. This is because weighted aggregation helps our model learn different importance of user, query, and product on different datasets. It also allows model to learn the importance of different features.

Table 4: Results of ablation study.

	Ali-1Core		Ali-5Core	
	HR@10	NDCG@10	HR@10	NDCG@10
IHGNN-up	0.1523	0.0997	0.2711	0.1893
IHGNN-qp	0.1750	0.1171	0.2410	0.1637
HyperGCN	0.1803	0.1180	0.2655	0.1826
IHGNN-O1	0.2038	0.1427	0.2812	0.2035
IHGNN-O2	0.2023	0.1420	0.2873	**0.2094**
IHGNN-O3	**0.2073**	**0.1465**	**0.2894**	0.2091

Contribution of modeling high-order interaction. By comparing the results of IHGNN-O1~3 in Table 4, we can find considering 2- and 3-order feature indeed boosts PPS performance. This result is coincident with our intuition. The interactions among neighbors indeed provide useful signal to enhance the representation learning of the target node.

Contribution of leveraging complete relations. Section 4.4 has discussed the effect of using hypergraph comparing with collapsed graph. In this section, we further explore the effect of using complete relations. We construct a model that only consider user and product nodes: IHGNN-up, and another model that only considers query and product nodes: IHGNN-qp. By comparing IHGNN-up/qp with IHGNN-O1~3 in Table 4, we see that modeling user-query-product triples using a hypergraph is much better than only modeling user-product or query-product tuples in product search scenario.

4.6 Sensitivity Analysis (RQ3)

Effect of GNN layer count. The number of GNN layers decides how many hops each node can visit in the process of message passing. In this experiment, we vary the number of GNN layers from 0 to 6 and test the performance of different models including GCN, GAT, HyperGCN and IHGNN-O3. The result is presented in Figure 4 (a,b). With the layer of the convolutions increasing,

the performance will become better at the beginning. This result validates the effectiveness of leveraging (hyper)-graph aggregation in the recommender system. But when the layer surpasses a threshold, the performance becomes unaffected or even experiences some degradation with the further increase. Too deep layer may not bring additional collaborative signal, and even may bring some noise in learning.

Effect of embedding size. We have also studied the effect of embedding size on models' performances on Ali-1Core and Ali-5Core data. We varied the embedding sizes from 32 to 256. The result is shown in Figure 4 (c, d). The results show that increasing embedding size benefits IHGNN model. And our IHGNN model outperforms all baselines with different embedding sizes. We also observe that the superiority of IHGNN is more obvious on the sparser Ali-1Core dataset, which demonstrates that our model has better ability to capture feature information from sparse data.

5 RELATED WORK

5.1 Product Search

Early product search is based on structural data retrieved from relational databases, such as brands and categories [21, 32]. Duan et al. [8, 9] proposed a probabilistic mixture model by analysing e-commerce search logs to effectively match query and title. But there are still semantic gaps between query words and product titles, which leads to the semantic mismatch problem. Some works [19, 28, 31] proposed to map query and product text into a hidden semantic space to learn the representations using deep neural networks.

Other than semantic match, personalization is playing a important role in improving user experience and increasing the retailer revenues in product search. Ai et al. [2] and Ge et al. [12] both proposed a hierarchical embedding model to jointly learn representations of users, queries, and products in a hidden semantic space. A zero attention model (ZAM) [1] is then proposed that automatically determines the level of personalization for a specific user-query pair. Bi et al. [4, 5] proposed a transformer-based embedding model (TEM) and a review-based transformer model (RTM) to dynamically control the influence of personalization. Zhou et al. [47] proposed a context-aware long-short term preference model to enhance the representation of the current query.

Some researchers focus on different scenarios in product search, such as streaming [40] and conversational systems [6, 44]. Some works leverages different information in product search to help ranking. Long et al. [23] used popularity combined with relevance to improve product ranking. Guo et al. [14] utilized multi-modal data in product search. Wu et al. [39] utilized click and purchase data to jointly optimize search results.

5.2 Graph Based Product Search

Graph embedding method has been proven effective in information retrieval task [16]. By using graph embedding based method, Ai et al. [3] proposed to construct explainable PPS model through a product-category-product knowledge graph. Similarly, Niu et al. [26] construct a dual heterogeneous graph attention network for the query-item graph in e-commerce search. Zhang et al. [45] used graph-based features to enhance learning-to-rank frameworks. Ren et al. [27] used heterogeneous click graph to learn generic search intents.

We argue that existing graph-based methods only include two types of entities ((query,product) or (user,product)) rather than all of them (user,product,query) in the constructed graph, failing to modeling the ternary relation among them. Our experiments also validate that modeling ternary relations is quite useful. To the best of our knowledge, only one work Liu et al. [22] considered all the entities in the graph and exploited there types of structural relationship in user-query-product interactions. But such three manually-designed patterns are far from sufficient.

5.3 Hypergraph Learning

Relations of three or more entities cannot be represented by a traditional graph. Hypergraph naturally models high-order relations. Zhou et al. [46] first introduced hypergraph learning into transductive classification task, generalizing the methodology of spectral clustering to hypergraphs, which originally operates on undirected graphs. Feng et al. [11] presented hypergraph neural networks (HGNNs) to consider the high-order data structure to learn representation better with multi-modal data. Mao et al. [25] applied hypergraph ranking to multi-objective recommendation task by establishing a user-product-attribute-context data model. Wang et al. [35] developed a next-item recommendation framework empowered by sequential hypergraphs to infer the dynamic user preferences with sequential user interactions.

However, to our best knowledge, few works used hypergraph in personalized product search. In this paper, we applied hypergraph neural networks to PPS task and improved our model's performance by optimizing model structure and exploiting high-order feature interactions. Also, we remark these methods may not be suitable to be directly applied in PPS. Our IHGNN is designed specifically for PPS with modeling weighs in node aggregation and modeling high-order feature interactions. Although there are some recent work [10, 49] modeling high-order feature interaction on GNN, the feature interaction on hypergraph has not been explored.

6 CONCLUSIONS AND FUTURE WORK

In this article, we proposed a new hypergraph-based method IHGNN to better model the crucial *collaborative signal* among users, queries and products for personalized product search. Also, an improved hypergraph neural network is developed and the neighbor feature interactions are explicitly introduced. Extensive experiments have been conducted to validate the superiority of IHGNN over baselines. We also analyze the contribution of each component in our model.

For future work, we consider using attention mechanisms in both hypergraph propagation phases and high-order feature calculation. Also, it would be interesting to explore disentanglement representation in PPS, as users may have diverse preference and queries may have diverse semantics.

7 ACKNOWLEDGEMENTS

This work is supported by the National Natural Science Foundation of China (62102382,U19A2079), the USTC Research Funds of the Double First-Class Initiative (WK2100000019) and the Alibaba Innovative Research project (ATT50DHZ420003).

REFERENCES

[1] Qingyao Ai, Daniel N Hill, SVN 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*. 379–388.

[2] 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*. 645–654.

[3] Qingyao Ai, Yongfeng Zhang, Keping Bi, and W Bruce Croft. 2019. Explainable product search with a dynamic relation embedding model. *ACM Transactions on Information Systems (TOIS)* 38, 1 (2019), 1–29.

[4] 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*. 1521–1524.

[5] Keping Bi, Qingyao Ai, and W Bruce Croft. 2021. Learning a Fine-Grained Review-based Transformer Model for Personalized Product Search. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 123–132.

[6] Keping Bi, Qingyao Ai, Yongfeng Zhang, and W Bruce Croft. 2019. Conversational product search based on negative feedback. In *Proceedings of the 28th acm international conference on information and knowledge management*. 359–368.

[7] Keping Bi, Choon Hui Teo, Yesh Dattatreya, Vijai Mohan, and W Bruce Croft. 2019. Leverage implicit feedback for context-aware product search. *arXiv preprint arXiv:1909.02065* (2019).

[8] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. A probabilistic mixture model for mining and analyzing product search log. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2179–2188.

[9] Huizhong Duan, ChengXiang Zhai, Jinxing Cheng, and Abhishek Gattani. 2013. Supporting keyword search in product database: a probabilistic approach. *Proceedings of the VLDB Endowment* 6, 14 (2013), 1786–1797.

[10] Fuli Feng, Xiangnan He, Hanwang Zhang, and Tat-Seng Chua. 2021. Cross-GCN: Enhancing graph convolutional network with k-Order feature interactions. *IEEE Transactions on Knowledge and Data Engineering* (2021).

[11] Yifan Feng, Haoxuan You, Zizhao Zhang, Rongrong Ji, and Yue Gao. 2019. Hypergraph neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, Vol. 33. 3558–3565.

[12] Songwei Ge, Zhicheng Dou, Zhengbao Jiang, Jian-Yun Nie, and Ji-Rong Wen. 2018. Personalizing search results using hierarchical RNN with query-aware attention. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 347–356.

[13] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Yinglong Wang, Jun Ma, and Mohan Kankanhalli. 2019. Attentive long short-term preference modeling for personalized product search. *ACM Transactions on Information Systems (TOIS)* 37, 2 (2019), 1–27.

[14] Yangyang Guo, Zhiyong Cheng, Liqiang Nie, Xin-Shun Xu, and Mohan Kankanhalli. 2018. Multi-modal preference modeling for product search. In *Proceedings of the 26th ACM international conference on Multimedia*. 1865–1873.

[15] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[16] William L Hamilton, Rex Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Proceedings of the 31st International Conference on Neural Information Processing Systems*. 1025–1035.

[17] Xiangnan He, Tao Chen, Min-Yen Kan, and Xiao Chen. 2015. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 1661–1670.

[18] Xiangnan He, Kuan Deng, Xiang Wang, Yan Li, Yongdong Zhang, and Meng Wang. 2020. Lightgcn: Simplifying and powering graph convolution network for recommendation. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*. 639–648.

[19] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. 2013. Learning deep structured semantic models for web search using click through data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. 2333–2338.

[20] Thomas N Kipf and Max Welling. 2016. Semi-supervised classification with graph convolutional networks. *arXiv preprint arXiv:1609.02907* (2016).

[21] Soon Chong Johnson Lim, Ying Liu, and Wing Bun Lee. 2010. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing & Management* 46, 4 (2010), 479–493.

[22] Shang Liu, Wanli Gu, Gao Cong, and Fuzheng Zhang. 2020. Structural Relationship Representation Learning with Graph Embedding for Personalized Product Search. In *Proceedings of the 29th ACM International Conference on Information & Knowledge Management*. 915–924.

[23] Bo Long, Jiang Bian, Anlei Dong, and Yi Chang. 2012. Enhancing product search by best-selling prediction in e-commerce. In *Proceedings of the 21st ACM international conference on Information and knowledge management*. 2479–2482.

[24] Shuqi Lu, Zhicheng Dou, Xu Jun, Jian-Yun Nie, and Ji-Rong Wen. 2019. Psgan: A minimax game for personalized search with limited and noisy click data. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 555–564.

[25] Mingsong Mao, Jie Lu, Jialin Han, and Guangquan Zhang. 2019. Multiobjective e-commerce recommendations based on hypergraph ranking. *Information Sciences* 471 (2019), 269–287.

[26] Xichuan Niu, Bofang Li, Chenliang Li, Rong Xiao, Haochuan Sun, Hongbo Deng, and Zhenzhong Chen. 2020. A dual heterogeneous graph attention network to improve long-tail performance for shop search in e-commerce. In *Proceedings of the 26th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 3405–3415.

[27] Xiang Ren, Yujing Wang, Xiao Yu, Jun Yan, Zheng Chen, and Jiawei Han. 2014. Heterogeneous graph-based intent learning with queries, web pages and wikipedia concepts. In *Proceedings of the 7th ACM international conference on Web search and data mining*. 23–32.

[28] Yelong Shen, Xiaodong He, Jianfeng Gao, Li Deng, and Grégoire Mesnil. 2014. A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Information & Knowledge Management*. 101–110.

[29] Jaime Teevan, Susan T Dumais, and Daniel J Liebling. 2008. To personalize or not to personalize: modeling queries with variation in user intent. In *Proceedings of the 31st annual international ACM SIGIR conference on Research and development in information retrieval*. 163–170.

[30] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 165–174.

[31] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM international on conference on information and knowledge management*. 165–174.

[32] Damir Vandic, Flavius Frasincar, and Uzay Kaymak. 2013. Facet selection algorithms for web product search. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*. 2327–2332.

[33] Petar Veličković, Guillem Cucurull, Arantxa Casanova, Adriana Romero, Pietro Lio, and Yoshua Bengio. 2017. Graph attention networks. *arXiv preprint arXiv:1710.10903* (2017).

[34] Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*. 363–372.

[35] Jianling Wang, Kaize Ding, Liangjie Hong, Huan Liu, and James Caverlee. 2020. Next-item recommendation with sequential hypergraphs. In *Proceedings of the 43rd international ACM SIGIR conference on research and development in information retrieval*. 1101–1110.

[36] Xiang Wang, Xiangnan He, Meng Wang, Fuli Feng, and Tat-Seng Chua. 2019. Neural graph collaborative filtering. In *Proceedings of the 42nd international ACM SIGIR conference on Research and development in Information Retrieval*. 165–174.

[37] Jiancan Wu, Xiang Wang, Fuli Feng, Xiangnan He, Liang Chen, Jianxun Lian, and Xing Xie. 2021. Self-Supervised Graph Learning for Recommendation. In *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 726–735.

[38] Le Wu, Xiangnan He, Xiang Wang, Kun Zhang, and Meng Wang. 2021. A Survey on Neural Recommendation: From Collaborative Filtering to Content and Context Enriched Recommendation. *arXiv preprint arXiv:2104.13030* (2021).

[39] Liang Wu, Diane Hu, Liangjie Hong, and Huan Liu. 2018. Turning clicks into purchases: Revenue optimization for product search in e-commerce. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 365–374.

[40] 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*. 1693–1702.

[41] Naganand Yadati, Madhav Nimishakavi, Prateek Yadav, Vikram Nitin, Anand Louis, and Partha Talukdar. 2018. HyperGCN: A new method of training graph convolutional networks on hypergraphs. *arXiv preprint arXiv:1809.02589* (2018).

[42] Jing Yao, Zhicheng Dou, Jun Xu, and Ji-Rong Wen. 2020. RLPer: A Reinforcement Learning Model for Personalized Search. In *Proceedings of The Web Conference 2020*. 2298–2308.

[43] Ting Zhang, Bang Liu, Di Niu, Kunfeng Lai, and Yu Xu. 2018. Multiresolution graph attention networks for relevance matching. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*. 933–942.

[44] Yongfeng Zhang, Xu Chen, Qingyao Ai, Liu Yang, and W Bruce Croft. 2018. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th acm international conference on information and knowledge management*. 177–186.

[45] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR meets graph embedding: a ranking model for product search. In *The World Wide Web Conference*.

2390–2400.

[46] Dengyong Zhou, Jiayuan Huang, and Bernhard Schölkopf. 2006. Learning with hypergraphs: Clustering, classification, and embedding. *Advances in neural information processing systems* 19 (2006), 1601–1608.

[47] 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*. 1111–1120.

[48] Yujia Zhou, Zhicheng Dou, and Ji-Rong Wen. 2020. Enhancing Re-finding Behavior with External Memories for Personalized Search. In *Proceedings of the 13th International Conference on Web Search and Data Mining*. 789–797.

[49] Hongmin Zhu, Fuli Feng, Xiangnan He, Xiang Wang, Yan Li, Kai Zheng, and Yongdong Zhang. 2020. Bilinear graph neural network with neighbor interactions. *arXiv preprint arXiv:2002.03575* (2020).