IHGNN: Interactive Hypergraph Neural Network for Personalized Product Search

Dian Cheng1*, Jiawei Chen1†, Wenjun Peng1, Wenqin Ye1, Fuyu Lv2, Tao Zhuang2, Xiaoyi Zeng2, Xiangnan He1.
1University of Science and Technology of China, 2Alibaba 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.

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, 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 (u2, p2) happened: q1, or q3, 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 component design of IHGNN.

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.

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 \((u, p, q) \notin \mathcal{L} \). The goal of personalized product search is to learn a score function \( f: \mathcal{U} \times \mathcal{P} \times \mathcal{Q} \rightarrow \mathbb{R} \) 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 \( \mathcal{G} = (\mathcal{V}, \mathcal{E}, \mathcal{H}) \) 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 \( \mathcal{H} \) describes the connectivity of the hypergraph, with entries defined as:

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

(1)

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

---

\(^1\)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 placeholder for better description.
We first construct a hypergraph $G_h$ (1) a node set $V$, whose element represents a user, product or query (i.e., $V = \{u, e, q\}$); (2) a hyperedge set $E$, whose element represents a ternary relation depicting there exists an historical interaction among them (i.e., $E = \{e \in E | h(u, e, v) = 1\}$).

The constructed hypergraph could preserve complete ternary information from the related hyperedge to refine the representation of node $v$ (i.e., $N_v = \{a \in V | \exists e \in E, h(u, e, v) = 1 & h(a, e, v) = 1\}$).

### 3.1 Hypergraph Construction

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

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 $q \in Q$, we utilize the query content information and use a mean pooling over word embeddings $z_w$ to generate the query embedding $z_q \in R^d$:

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

For each user $u \in U$ (or product $p \in P$), we directly generate its embedding $z_u \in R^d$ (or $z_p \in R^d$) from an embedding look-up table $E \in R^{((|U|+|P|) \times d)}$ (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 $e \in E_v$, we define the aggregation function as follow:

$$m_e = [z_u \parallel z_p \parallel z_q]W$$

where $m_e$ denotes the information contained by the medium $e$, $u, p, q$ denote the user, product, query node that $e$ connects, $W$ denotes the concatenation operation, $W \in R^{3d \times d}$ 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:

$$z_v^{(1)} = \frac{\sum_{e \in E_v} m_e}{d(v)}$$


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 product, the product would generate a quite useful semantic (“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., \( u, p, q \in V_e \)) that the hyperedge \( e \) connects. We have:

\[
\begin{align*}
    f^{e1}_e &= u || z_q || z_p \\
    f^{e2}_e &= (z_u \odot z_q) || (z_u \odot z_p) || (z_q \odot z_p) \\
    f^{e3}_e &= z_u \odot z_q \odot z_p
\end{align*}
\]  

(5)

Where \( \odot \) stands for element-wise product, \( f^{e1}_e, f^{e2}_e, f^{e3}_e \) respectively capture 1,2,3-order feature interaction of neighbors. Analogously, we use a linear layer to aggregate the interaction information as:

\[
m_e = [f^{e1}_e || f^{e2}_e || f^{e3}_e] W_{e3}
\]  

(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., \( u, p, q \in V_e \& e \in E_0 \)) rather than among all neighbors (i.e., \( u, p, q \in N_0 \)). 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:

\[
z^{(l)}_v = \frac{\sum_{e \in E_v} m_e^{(l)}}{d(v)}
\]  

(7)

where the hyperedge information \( m_e^{(l)} \) can be calculated from \( z^{(l-1)}_u, z^{(l-1)}_p, z^{(l-1)}_q \) with equation (3) or (6). Finally we concatenate the representations learned by different layers to generate the entity’s final representation:

\[
z^*_v = z^{(0)}_v || z^{(1)}_v || ... || z^{(L)}_v
\]  

(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 [22] 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:

\[
\hat{y}_{uqp} = \text{sigmoid} \left( (\lambda z^*_u + (1-\lambda) z^*_p)^T z^*_q \right)
\]  

(9)

where \( \lambda \) 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:

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

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

**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).

**CIKM CUP2016 Track2.** It is a product search dataset from CIKM-Cup2016 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.

### Table 1: Dataset Statistics.

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Users</th>
<th>Queries</th>
<th>Products</th>
<th>Interactions</th>
</tr>
</thead>
<tbody>
<tr>
<td>Ali-1Core</td>
<td>200,000</td>
<td>102,816</td>
<td>220,779</td>
<td>940,946</td>
</tr>
<tr>
<td>Ali-5Core</td>
<td>39,976</td>
<td>46,098</td>
<td>39,326</td>
<td>403,982</td>
</tr>
<tr>
<td>CIKM CDs_5</td>
<td>23,882</td>
<td>3,256</td>
<td>23,550</td>
<td>339,341</td>
</tr>
<tr>
<td>CDs_5</td>
<td>112,379</td>
<td>509</td>
<td>67,602</td>
<td>1,296,885</td>
</tr>
</tbody>
</table>

### 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 datasets.

**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).

**Amazon simulated dataset.** This is a product search dataset from CIKM-Cup2016 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,
which gives different personalization weights to different history items.
- **GraphSRRL** [22] improves HEM with utilizing three manually-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 performance by calculating the ratio of positive products appearing in 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 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 i9-9900K 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.

### 4.4 Results and Analysis (RQ1)

Table 2 presents the performance of 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...
IHGN: Interactive Hypergraph Neural Network for Personalized Product Search

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

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.

<table>
<thead>
<tr>
<th>Node</th>
<th>Weighted prop.</th>
<th>Feature order</th>
</tr>
</thead>
<tbody>
<tr>
<td>IHGNN-up</td>
<td>u, p</td>
<td>×</td>
</tr>
<tr>
<td>IHGNN-qp</td>
<td>q, p</td>
<td>×</td>
</tr>
<tr>
<td>HyperGCN</td>
<td>u, q, p</td>
<td>×</td>
</tr>
<tr>
<td>IHGNN-O1</td>
<td>u, q, p</td>
<td>√</td>
</tr>
<tr>
<td>IHGNN-O2</td>
<td>u, q, p</td>
<td>√</td>
</tr>
<tr>
<td>IHGNN-O3</td>
<td>u, q, p</td>
<td>√</td>
</tr>
</tbody>
</table>

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 collaborative 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.

Table 4: Results of ablation study.

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

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.

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 an 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 these 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 (HGGNs) 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 weights 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 (WK210000019) and the Alibaba Innovative Research project (ATT50DHZ420003).
REFERENCES


