Structural Relationship Representation Learning with Graph Embedding for Personalized Product Search

Shang Liu
Nanyang Technological University, Singapore.
shang.liu@ntu.edu.sg

Wanli Gu
Meituan-Dianping Group, China.
guwanli@meituan.com

Gao Cong
Nanyang Technological University, Singapore.
gaocong@ntu.edu.sg

Fuzheng Zhang
Meituan-Dianping Group, China.
zhangfuzheng@meituan.com

ABSTRACT

To provide more accurate personalized product search (PPS) results, it is compulsory to go beyond modeling user-query-item interaction. Graph embedding techniques open the potential to integrate node information and topological structure information. Existing graph embedding enhanced PPS methods are mostly based on entity-relation-entity graph learning. In this work, we propose to consider structural relationship in users' product search scenario with graph embedding by latent representation learning. We argue that explicitly modeling the structural relationship in graph embedding is essential for more accurate PPS results. We propose a novel method, Graph embedding based Structural Relationship Representation Learning (GraphSRRL), which explicitly models the structural relationship in users-queries-products interaction. It combines three key conjunctive graph patterns to learn graph embedding for better PPS. In addition, GraphSRRL facilitates the learning of affinities between users (resp. queries or products) in the designed geometric operation in low-dimensional latent space. We conduct extensive experiments on four datasets to evaluate GraphSRRL for PPS. Experimental results show that GraphSRRL outperforms the state-of-the-art algorithm on real-world search datasets by at least 50.7% in term of Hit@10 and 48.7% in terms of NDCG@10.

CCS CONCEPTS

• **Information systems** → **Information retrieval**; **Personalization**; *Retrieval models and ranking*.

KEYWORDS

Product search; Neural networks; Graph embedding; Structural relationship representation learning

ACM Reference Format:
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 and Knowledge Management (CIKM '20), October 19–23, 2020, Virtual Event, Ireland.* ACM, New York, NY, USA, 10 pages. https://doi.org/10.1145/3340531.3411936

1 INTRODUCTION

E-commerce or business service platforms, such as Amazon and Yelp, support e-shopping for users to access substantial goods and foods. In a typical e-shopping scenario, users submit a query (e.g., loose T-shirts) to express their needs and explore the retrieved products to find the items of their interests. Therefore, the relevance of product search results affects both satisfaction degree of users to the e-shopping platform and the revenue of the e-commerce retailer.

Personalization is very important in product search where users' purchase decision is largely influenced by their preferences. For example, when a young user searches for a " loose T-shirt" on an e-commerce platform, S/he is more likely to purchase a fashionable style or a shirt from a brand s/he is interested in. *Personalized Product Search* (PPS) aims to generate personalized product suggestions for given queries [14, 15], which plays a vital role in many e-commerce platforms.

Recently, a few PPS algorithms [2, 14, 28, 30] have been proposed. In general, existing PPS methods can be categorized into three types. The first type is based on latent vector space learning [2, 15, 28] to learn the latent representation of users, queries and products. Users' preferences are matched with both user's textual query and other information such as textual reviews or visual image. The second type is learning user's static or dynamic preference [1, 14, 30]. Users' preferences are learned from user's historical queries and interaction information. The third type is to integrate graph embedding into product search to effectively utilize external heterogeneous information [3, 34]. This category of methods utilize heterogeneous information such as query or product co-occurrence and product category to improve the search performance. For example, Ai et al. [3] propose the Dynamic Relation Embedding Model (DREM) for PPS in which product-relation-product graph is constructed to learn the product representation in a latent space.

While these methods have significantly improved the performance of PPS, existing PPS methods still have two shortcomings.

First, they only characterize three-way user-query-item interaction, and do not exploit the structural relationship in user's product search scenario. As shown in Figure 1, both u_1 and u_2 pose query q_1 and have an engagement with product p_1. The structural relationship in the interaction like this is prevalent in product search scenario. However, existing PPS methods fail to model them and cannot take into account the structural relationship representation. Second, existing graph embedding enhanced methods (e.g., [3]) do not explicitly learn the representation to capture affinities between users (resp. queries, or products), which is of importance to improve performance of PPS. Without considering these two aspects, the learned representation may not be sufficient for PPS task.

In this work, we tackle the challenge by exploiting the structural relationship representation learning from users-queries-products interactions, which naturally preserve collaborative signal between users/queries/products and interaction information in the relationship path to improve PPS. We term these structural relationships as "*conjunctive graph pattern*" [16]. For example, as shown in Figure 1 there are three key patterns. Note that when branches have three or more we can randomly sample two branches among them to get the following key patterns:

- $\{u_i, u_j\} \rightarrow q_i \rightarrow p_i$.
- $u_i \rightarrow \{q_i, q_j\} \rightarrow p_i$.
- $u_i \rightarrow q_i \rightarrow \{p_i, p_j\}$.

Consider the limitations of existing solutions, we believe it is of critical importance to develop a graph embedding enhanced PPS model that can exploit conjunctive pattern information in an effective and explicit manner. Specifically, we propose a new method named Graph embedding based Structural Relationship Representation Learning model (GraphSRRL), which is equipped with two designs to correspondingly address the challenges in PPS: 1) combine three key conjunctive graph patterns for graph embedding learning, which performs embedding propagation to capture structural relationship for interaction. 2) design geometric operation (e.g., projection, intersection) in low-dimensional latent space to consider affinities between users/queries/products in pattern selectivity and the embedding learning process. GraphSRRL is conceptually advantageous to existing methods in that it is an embedding-based framework that can effectively learn the representation in structural relationships and affinities between users (resp. queries or products) in geometric operation, which are then incorporated into PPS.

The contributions of this work are summarised as follows:

- We propose the idea of explicitly modeling the structural relationships in graph embedding to provide better personalized product search and define three types of conjunctive graph patterns in user-query-product interaction learning.
- We develop a new method GraphSRRL, which explores three key conjunctive graph patterns with geometric operation such as projection operator and intersection operator in an effective and explicit manner.
- We conduct extensive experiments on four search datasets, demonstrating that GraphSRRL outperforms state-of-the-art PPS methods in accuracy. Experimental results show that GraphSRRL outperforms the state-of-the-art algorithm on real-world datasets by at least 50.7% in term of Hit@10 and 48.7% in terms of NDCG@10.

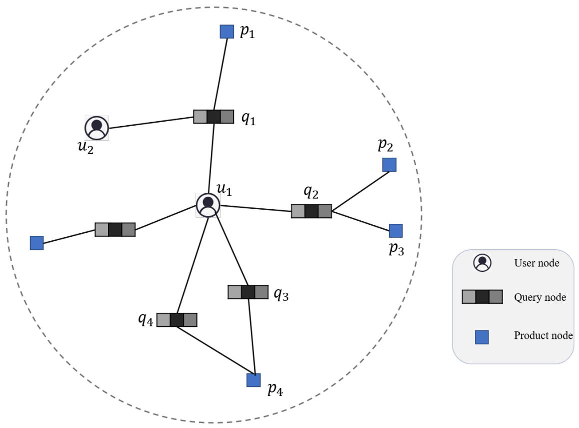


Figure 1: Example of graph construction.

2 RELATED WORK

2.1 Product Search

Product search has been an active and hot research problem. The earlier efforts aim to associate user queries with structured product information stored in relation databases [8, 10], such as brands, categories, etc. For example, Duan et al. [9] propose a probabilistic generative model to optimize the ranking by analysing e-commerce search log for useful knowledge to fill the gap between queries and product entities. Later on, more advanced retrieval models that support more complicated search data and optimization objectives are proposed for product search. For example, several studies [4, 20] discuss issues and strategies of applying learning-to-rank techniques to the product search scenario. Hu et al. [18] propose to use reinforcement learning to learn an optimal ranking policy for better utilizing the correlation between different ranking steps. Van Gysel et al. [28] propose a latent vector space model (LSE) to jointly learn distribution representation of products and queries for matching. In addition, some proposals [29, 32] aim to optimize search results by using different product retrieval metrics or constructing a Latent Dirichlet Allocation model with product information. Zhang et al. [34] propose to fuse information of query or product graph in neural retrieval model to improve product search results.

Personalized Product Search In addition to matching between query and item, it is critical to consider user's preference for product search, which will return more relevant results, persuading users to purchase. Ai et al. [2] propose a hierarchical embedding model to jointly learn latent semantic representations of users, queries, and products for personalized product search. Guo et al. [15] consider both visual and textual latent spaces, which are expected to preserve the visual similarity and semantic similarity of products, for personalized product search. Ai et al. [1] constructs query-dependent user embedding with attention mechanism over user's historical interactions. Guo et al. [14] explore long and short term user preference learning model for personalized product search. Xiao et al.

[30] utilize a probabilistic metric learning approach to infer dynamic semantic representation for entities, which aim to capture the dynamic user preferences for better personalized product search. Ai et al [3] propose to integrate product graph information into personalized product search, which is based on the constructed product-category-product graph and the learned representation is not tailed for PPS. Recently conversational based product search [33, 35] is explored. Despite the compelling success achieved by previous work, little attention has been paid to learn structural relationship in user's product search scenario for improving personalized product search results. In this paper, we propose a graph embedding based neural network model to fill this gap.

In this paper, we focus on how to effectively utilize graph representation learning in personalized product search. To our best knowledge, our work is the first that explicitly models the structural relationship in graph embedding in users-queries-products interaction learning to provide better personalized product search.

2.2 Neural Search and Graph Embedding based search

Our work is also related to the research of neural search methods, and graph embedding based search methods for other information retrieval tasks. We next review some of these methods.

Neural Search Recent years have seen big advance in applying neural network technology to different search tasks, such as ad-hoc retrieval, and recommendation. Neural search models are demonstrated to be effective for they can learn and incorporate embedding features in ranking [25, 27]. For example, Huang et al. [19] propose a deep neural network structure to project queries and documents into a common low-dimensional space where the relevance of a document given a query is computed as the distance between them. Borisov et al. [5] introduce a neural click model to model user browsing behavior for both click prediction task and relevance prediction task. Mitra et al. [24] propose a document ranking model composed of two separate deep neural networks, one that matches the query and the document using a local representation, and the other that matches the query and the document using learned distributed representations.

Graph Embedding based Search Method Graph embedding techniques have been proven to be capable of capturing graph structure information [13, 17], which has been integrated with various information retrieval applications. In the task of information retrieval, the user, query, and item interaction contains the click or purchase, which can construct a graph data. For example, Creswell et al. [6] apply a Markov random walk model to a large click log, producing a probabilistic ranking of documents for a given query. Li et al. [22] and Ren et al. [26] use click graph to learn query intent for web search. Gao et al. [11] propose smoothing models to expand click-through data: query clustering via Random Walk on click graphs to overcome sparsity problem.

Our work is largely inspired by these neural models and graph embedding models. However, to our knowledge, none of the existing work explores the so called "conjunctive structure patterns" in graph embedding to improve search ranking as we do in this work.

Table 1: Definitions of notations.

Notation	description
$\mathcal{U}, \mathcal{Q}, \mathcal{P}$	user set, query set, product set.
u, q, p	a specific user, query, product.
$\boldsymbol{u}, \boldsymbol{q}, \boldsymbol{p} \in \mathrm{R}^{\mathrm{d}}$	embedding representation of u, q, p.
d	dimensionality of the embedding vector
v, τ	a node, a edge type.
z	embedding representation of a node.
φ, ψ	geometric operators.
$\boldsymbol{W} \in \mathrm{R}^{d \times d}, \boldsymbol{b} \in \mathrm{R}^{\mathrm{d}}$	the weight matrix and bias vector.

3 PROBLEM STATEMENT

3.1 Problem Definition

We formulate the personalized product search problem in this paper as follows. Given a dataset with product set $\mathcal{P}$, a user set $\mathcal{U}$, and a possible query set $\mathcal{Q}$, in which a product $p_k \in \mathcal{P}$ is clicked, viewed or purchased by a user $u_i \in \mathcal{U}$ after the user searches a query $q_j \in \mathcal{Q}$. The user-query-product interaction tensor $Y \in R^{I \times J \times K}$ is defined where $y_{u,q,p} = 1$ indicates that user u is engaged with product p when searching query q. Additionally, we can transfer the interaction data to a graph representation to facilitate explicit structural relationship [31] learning in user-query-product interaction. The graph is comprised of users-queries-products interaction, which is illustrated in Figure 1. For example, as shown in the figure, user u_1 and user u_2 search a same query q_1 and interact with product p_1, which shows the similar preference of these two users on the given query q_1.

Formally, suppose the query intent of a query q is represented with a vector $\boldsymbol{q} \in \mathrm{R}^d$, the user preference of a user u is represented with $\boldsymbol{u} \in \mathrm{R}^d$, and the engaged product as $\boldsymbol{p} \in \mathrm{R}^d$. Given the users-queries-products interaction graph $\mathcal{G}$, we aim to predict the probabilistic score of whether a user u will interact with product p when the user issues query q. Our goal is to learn prediction function $\hat{y}_{\boldsymbol{u},\boldsymbol{q},\boldsymbol{p}} = \mathcal{F}(\boldsymbol{u}, \boldsymbol{q}, \boldsymbol{p} | \Theta, Y, \mathcal{G})$, where $\hat{y}_{\boldsymbol{u},\boldsymbol{q},\boldsymbol{p}}$ denotes the probabilistic score that user u will engage with product p when searching query q, and Θ is the model parameters of function $\mathcal{F}$.

Some notations frequently used throughout this paper are listed in Table 1. Matrices are denoted with bold symbols.

3.2 Query Representation Method

Let q be the query submitted by u, w_q be a word of the query, and $\boldsymbol{q} \in \mathrm{R}^d$ be the embedding representation of the query. We calculate $\boldsymbol{q}$ with a function of q as:

$$\boldsymbol{q} = f(q) = f(\{\boldsymbol{w}_q | w_q \in q\}), \tag{1}$$

where $\boldsymbol{w}_q \in \mathrm{R}^d$ is the embedding of word w_q in q. Then the non-linear projection method is defined as:

$$f(q) = tanh(\boldsymbol{W}^q \cdot \frac{\sum_{\boldsymbol{w}_q \in q} \boldsymbol{w}_q}{|q|} + \boldsymbol{b}^q), \tag{2}$$

where $|q|$ is the length of the query, $\boldsymbol{W}^q \in \mathrm{R}^{d \times d}$ and $\boldsymbol{b}^q \in \mathrm{R}^{\mathrm{d}}$ are the weight matrix and bias vector.

3.3 Conjunctive graph patterns

Different from existing graph construction with entity-relation-entity pattern [34], e.g., product-category-product relation, we aim to model structural relationship (as illustrated in Section 1) in user product search scenario.

In this work we propose to extract three core conjunctive graph patterns for relationship representation learning. To give concrete example using the users' product search interactions (Figure 1), consider scenario "user u_i and user u_j engage with the same product p_i when giving the search query q_i". We would write this scenario as:

$$s = p_i, \exists q_i : SEARCH(u_i, q_i) \wedge SEARCH(u_j, q_i) \wedge MATCH(q_i, p_i), \tag{3}$$

and we say that the answer or denotation of this scenario is the the set of all product nodes that are likely to be connected to nodes u_i and u_j in response to the existentially quantified query q_i. For personalized product search, we use $\wedge$ between edges that have types SEARCH and MATCH to denote that it not only considers the match of query and product but also the user's preferences. Through modeling patterns like this we aim to explicitly represent similar preference between users on search. For the scenario to be valid, its dependency graph must be a directed acyclic graph (DAG), with the anchor nodes (u_i and u_j in this case) as source nodes of the DAG and answer target as the sink nodes (p_i in this case).

The second conjunctive graph pattern considers scenario "user u_i searches a query q_i and then engages with product p_i and also interacts with product p_j" (as in Figure 1), which would be written as:

$$s = p_i, p_j, \exists q_i : SEARCH(u_i, q_i) \wedge MATCH(q_i, p_i) \wedge MATCH(q_i, p_j) \tag{4}$$

Through modeling patterns like this we aim to explicitly learn similarity between products that a user engages given the same query.

The third conjunctive graph pattern considers scenario " user u_i searches a query q_i or q_j and then both engages with product p_i" (as in Figure 1), which would be written as:

$$s = p_i, \exists (q_i, q_j) : SEARCH(u_i, q_i) \wedge SEARCH(u_i, p_j) \wedge MATCH(q_i, p_i) \wedge MATCH(q_i, p_j) \tag{5}$$

Through modeling patterns like this we aim to explicitly learn similarity between queries that a user gives for the same target product.

We will introduce the mathematical operating process of these three types of patterns in Section 4 for graph embedding to learn relationship representation, which is then utilized to enhance personalized product search.

Note that our designed conjunctive graph pattern learning method is different from meta-path based method [7] which cannot learn relations in intersection pattern.

4 OUR APPROACH: GRAPHSRRL

The key idea behind our approach GraphSRRL is that we learn how to embed the three types of conjunctive graph patterns into a

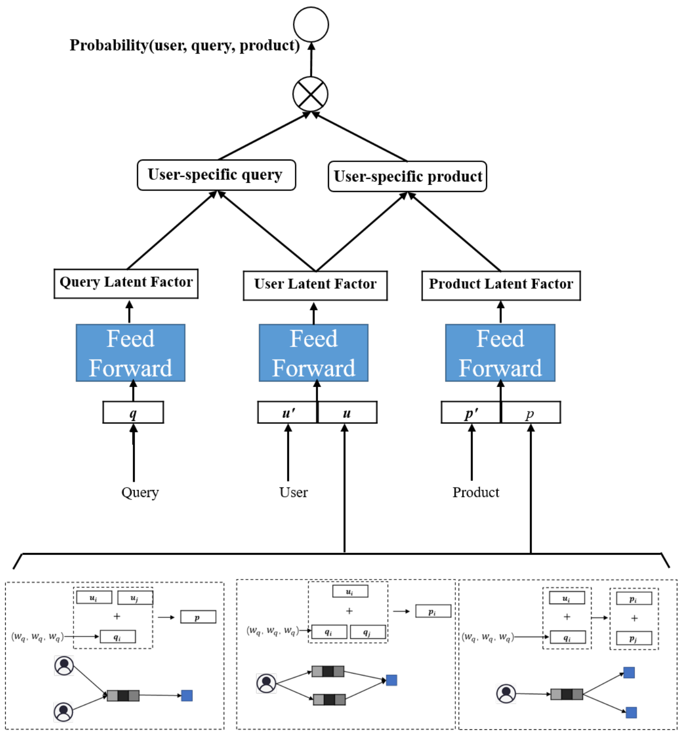


Figure 2: The framework of GraphSRRL. The bottom half and top half part illustrate the graph embedding and personalized search module, respectively, which are connected by the transmitting of nodes embedding representation

low-dimensional space with graph embedding for enhanced personalized product search. The framework of GraphSRRL is illustrated in Figure 2, which consists of two main components: graph embedding module and personalized search module. (1) The graph embedding module in the lower part utilizes the three designed conjunctive graph patterns to learn node embedding for relationship representation learning, which also facilitate the learning of affinities between users (resp. queries or products). The representation information is then introduced into personalized search module. (2) The personalized search module takes the user, query and product as well as the representations learned from graph embedding as input, and uses multi-layer perceptron (MLP) to integrate corresponding information. The extracted short and dense features for user, query and product are fed to MLP network to learn user-specific query representation and user-specific product representation, respectively. Then we feed them into another MLP together to compute the predicted probability score.

4.1 Graph Embedding Module

Graph embedding is to utilize the three designed conjunctive graph patterns to learn relationship representation and embed user, query and product nodes into continuous vector spaces which preserve node information and structural relationship information. This is

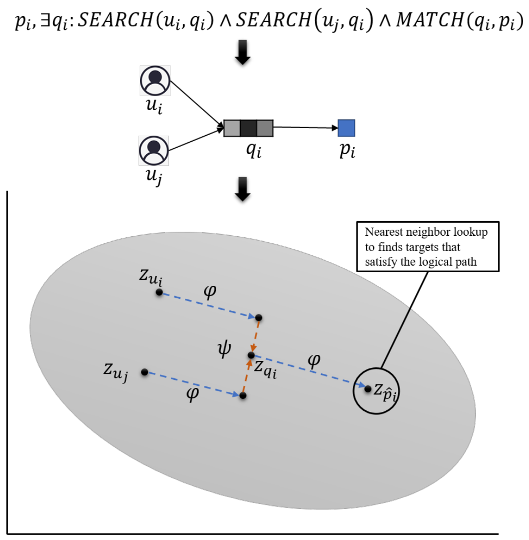

$p_i, \exists q_i: SEARCH(u_i, q_i) \wedge SEARCH(u_j, q_i) \wedge MATCH(q_i, p_i)$
u_i
u_j
q_i
p_i
Nearest neighbor lookup to finds targets that satisfy the logical path
z_{u_i}
φ
ψ
z_{q_i}
φ
z_{u_j}
φ
$z_{\hat{p}_i}$

Figure 3: The first conjunctive graph pattern learning example. Algorithm starts with embeddings of the anchor nodes and iteratively applies geometric operation φ an ψ to generate a predicted embedding target $z_{\hat{p}}$. Finally, we use the generated embedding to predict the likelihood that a node satisfies the relationship path, e.g., by nearest neighbor search in the embedding space.

achieved by representing structural relationship in graph as geometric operators that are jointly optimized in a low-dimensional embedding space along with a set of nodes embedding.

Take the first conjunctive graph pattern as an example (Figure 3). The algorithm maps structural relationship to an predicted target $z_{\hat{p}} \in \mathrm{R}^d$ using two differentiable operators, φ and ψ, described below. We define two geometric operators: Geometric projection operator φ and Geometric intersection operator ψ.

Geometric projection operator, φ: Given an embedding z and edge type τ (e.g., *SEARCH* and *MATCH*), the projection operator outputs a new embedding $z' = \varphi(z, \tau)$. We implement φ as:

$$\varphi(z, \tau) = \boldsymbol{W}^{\tau}\boldsymbol{z} + \boldsymbol{b}^{\tau}, \tag{6}$$

where $W^{\tau} \in \mathrm{R}^{d\times d}$ and $b^{\tau} \in \mathrm{R}^d$ are trainable parameters for edge type τ. In the base case, if a node embedding z_v and edge type τ are given as input, then it returns the embedding of the neighbor node set $N(v, \tau)$.

Geometric intersection operator, ψ_1: Suppose we are given two embeddings z_1 and z_2, both of which connect to a new node v with the same edge type τ. The geometric interaction operator produces a new embedding $\boldsymbol{z}'$, which performs set intersection in the embedding space. We define ψ_1 as:

$$\psi_1(\boldsymbol{z}_1, \boldsymbol{z}_2, \boldsymbol{z}_v) = NN_k(Concat^1(\boldsymbol{z}_i, i = 1, 2)) + \boldsymbol{W}^{\tau}\boldsymbol{z}_v + \boldsymbol{b}^{\tau}, \tag{7}$$

where NN_k is a k-layer feedforward neural network, $Concat^1$ is a concatenation operator on the first dimension, and W^{τ}, and B^{τ} are the trainable transformation parameters for each node type τ.

On the other hand, when we are given an anchor embedding z, which connects to two new nodes v_1 and v_2 with the same edge type τ. The second geometric interaction operator produces a new embedding $\boldsymbol{z'}$. We define ψ_2 as:

$$\psi_2(\boldsymbol{z}, \boldsymbol{z}_{v_1}, \boldsymbol{z}_{v_2}) = \boldsymbol{W}^{\tau}\boldsymbol{z} + NN_k(Concat^1(\boldsymbol{z}_{v_i}, i = 1, 2)) + \boldsymbol{b}^{\tau}, \tag{8}$$

where the parameters have similar meaning with those in Equation 7.

In particular, we want to generate predicted embedding $z_{\hat{p}}$ and target product node embedding z_p, so that the likelihood or 'score' is given by the distance or similarity metric between these embeddings:

$$Score(z_{\hat{p}}, z_p) = f_{GE}(z_{\hat{p}}, z_p), \tag{9}$$

where $\boldsymbol{z}_p$ is the embedding feature vector for product node p. The distance or similarity function adopted in this paper is normalized inner product $f_{GE}(\boldsymbol{z}_{\hat{p}}, \boldsymbol{z}_p)$, although other forms like cosine distance could also be used here.

Therefore, the learning process of the first conjunctive graph pattern (c_1) in Equation 3 can be denoted as follows:

$$\begin{aligned} \hat{p}_i &= NN_k \begin{bmatrix} \boldsymbol{W}_1\boldsymbol{u}_i \\ \boldsymbol{W}_2\boldsymbol{u}_j \end{bmatrix} + \boldsymbol{W}_3\boldsymbol{q}_i, \\ score_{c_1} &= f_{GE}(\hat{\boldsymbol{p}}_i, \boldsymbol{p}_i), \end{aligned} \tag{10}$$

The learning process of the second conjunctive graph pattern (c_2) in Equation 4 can be denoted as follows:

$$\begin{aligned} \hat{p}_i &= \boldsymbol{W}_1\boldsymbol{u}_i + NN_k \begin{bmatrix} \boldsymbol{W}_2\boldsymbol{q}_i \\ \boldsymbol{W}_3\boldsymbol{q}_j \end{bmatrix}, \\ score_{c_2} &= f_{GE}(\hat{\boldsymbol{p}}_i, \boldsymbol{p}_i) \end{aligned} \tag{11}$$

The learning process of the third conjunctive graph pattern (c_3) in Equation 5 can be denoted as follows:

$$\begin{aligned} \hat{p}_i &= \boldsymbol{W}_1\boldsymbol{u}_i + \boldsymbol{W}_2\boldsymbol{q}_i, \\ \widetilde{p}_i &= NN_k \begin{bmatrix} \boldsymbol{W}_3\boldsymbol{p}_i \\ \boldsymbol{W}_4\boldsymbol{p}_j \end{bmatrix}, \\ score_{c_3} &= f_{GE}(\hat{\boldsymbol{p}}_i, \widetilde{\boldsymbol{p}}_i), \end{aligned} \tag{12}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_1$ $\boldsymbol{W}_3$ and $\boldsymbol{W}_4 \in \mathrm{R}^{d\times d}$ are all the trainable weights. The learning loss for graph embedding with three core conjunctive graph patterns c_1, c_2 and c_3 is as follows:

$$\begin{aligned} \mathcal{L}_{GE} = &-\Big(\sum_{u_i,u_j,q_i,p_i\in\mathcal{G}} score_{c1} - \sum_{u_i,u_j,q_i,p'_i\notin\mathcal{G}} score_{c_1}\Big) \\ &-\Big(\sum_{u_i,q_i,q_j,p_i\in\mathcal{G}} score_{c2} - \sum_{u_i,q_i,q_j,p'_i\notin\mathcal{G}} score_{c_2}\Big) \\ &-\Big(\sum_{u_i,q_i,p_i,p_j\in\mathcal{G}} score_{c3} - \sum_{u_i,q_i,p_i,p'_j\notin\mathcal{G}} score_{c_3}\Big) \end{aligned} \tag{13}$$

These three types of conjunctive graph patterns are utilized to explicitly learn relationship representation and embed user, query, and product nodes into continuous vector spaces, which will integrate node information and topological relationship in users-queries-products graph.

4.2 Personalized Search Module

The final user latent Factor is calculated with $\boldsymbol{u} = FFN(\begin{bmatrix} \boldsymbol{u}_{ps} \\ \boldsymbol{u}_{ge} \end{bmatrix})$, where $\boldsymbol{u}_{ge}$ is introduced from graph embedding and $\boldsymbol{u}_{ps}$ is randomly initialized with user ID then learned with the training process. $FFN(\cdot)$ is the 2-layered MLP feed-forward neural network. Similarly product latent factor: $\boldsymbol{p} = FFN(\begin{bmatrix} \boldsymbol{p}_{ps} \\ \boldsymbol{p}_{ge} \end{bmatrix})$, where $\boldsymbol{p}_{ps}$ is randomly initialized with product ID then learned with training process. Note that other diverse attributes information such as user's demographics or product's categories can be also added in embedding $\boldsymbol{u}_{ps}$ and $\boldsymbol{p}_{ps}$, which we will leave for exploration of future work and in this paper we mainly focus on relationship between users, queries, and products. Query latent factor: $\boldsymbol{q} = FFN(\boldsymbol{q}_{words})$ where $\boldsymbol{q}_{words}$ is calculated from words embedding with Equation 1.

User-specific query representation. The user-specific query representation $\boldsymbol{h}_{uq}$ is learned via a fully connected layer based on the user latent factor $\boldsymbol{u}_i$ and query latent factor $\boldsymbol{q}_i$,

$$\boldsymbol{h}_{uq} = \delta(\boldsymbol{W}_1^h \boldsymbol{u}_i + \boldsymbol{W}_2^h \boldsymbol{q}_i + \boldsymbol{b}) \tag{14}$$

where $\boldsymbol{W}_1$, $\boldsymbol{W}_2$ and $\boldsymbol{b}$ denote the weights matrices and bias vector in the fully connect layer. $\delta(\cdot)$ is the activation function.

User-specific product representation. The user-specific product representation $\boldsymbol{h}_{up}$ is learned via a fully connected layer based on the user latent factor $\boldsymbol{u}_i$ and product latent factor $\boldsymbol{p}_i$,

$$\boldsymbol{h}_{up} = \delta(\boldsymbol{W}_3^h \boldsymbol{u}_i + \boldsymbol{W}_4^h \boldsymbol{p}_i + \boldsymbol{b}) \tag{15}$$

After having user specific query representation $\boldsymbol{h}_{uq}$ and user specific product representation $\boldsymbol{h}_{up}$, we combine the two pathways by a predicting function f_{ps}, for example,inner product or an H-layer MLP. The final predicted probability score of user u engaging product p when searching query q is: $y_{uqp} = \delta(f_{ps}(h_{uq}, h_{up}))$, where sigmoid $\delta(\cdot)$ is the activation function for the output layer.

4.3 Learning Algorithm

The complete loss function of GraphSRRL is as follows:

$$\begin{aligned} \mathcal{L} =& \mathcal{L}_{PS} + \mathcal{L}_{GE} \\ =& \sum \mathcal{J}(y_{uqp}, \hat{y}_{uqp}) + \mathcal{L}_{GE} \end{aligned} \tag{16}$$

To learn the model parameters for $\mathcal{L}_{PS}$, it is trained as a binary classification problem. We view $y = 1$ as a positive label, which means user u engages with product p when searching query q otherwise it is a negative one. The personalized search module predicts the likelihood that user u engages with product p when searching query q. With sigmoid function $\delta(\cdot)$ as the activation function for the output layer, we constrain the output $\hat{y}$ in $[0, 1]$. Based on the above setting, we minimize the cross-entropy loss for true labels and sampled negative instances:

$$\mathcal{J}(y, \hat{y}) = \sum_{(u,q,p)\in Y^+ \cup Y^-} -y\log(\hat{y}) - (1-y)\log(1-\hat{y}), \tag{17}$$

where $\boldsymbol{Y}^+$ and $\boldsymbol{Y}^-$ denote the observed user-query-product triples and unobserved user-query-product triples, respectively. We uniformly sample negative product instances $\boldsymbol{Y}^-$ from unobserved products for user-query pair according to the fixed sampling ratio 10:1 to the number of observed interactions in each iteration. An Adaptive Moment Estimation (Adam) algorithm [21] is adopted to optimize the loss function.

Table 2: Statistics for the four personalized product search datasets

	#user	#query	#item	#interaction
Kindle_Store	68,223	24,148	61,934	982,619
Electronics	192,403	1,317	63,001	1,788,612
CIKMCup	18,113	23,247	27,303	444,549
Meituan	156,309	670,752	49,142	297,348

5 EXPERIMENTS

We conduct extensive experiments on the public available datasets and real-world Meituan (one of popular restaurant and entertainment search platforms in China) search dataset to compare GraphSRRL against state-of-the-art personalized search methods. We first briefly introduce the four product search datasets used in experiments, followed by baseline algorithms, evaluation protocols, and parameter settings. Finally, we report and analyze the experimental results, with comparison to the baseline methods.

5.1 Datasets

We utilize the following four datasets in our experiments for personalized product search, in which the first two are simulated search datasets from Amazon datasets, the third is the public available dataset and the last one is from Meituan-Dianping Group.

Amazon stimulated datasets. The simulated Amazon search dataset is used by Ai et al.[2, 3] on personalized product search task. Following the paradigm used by Gysel et al. [28], Ai et al extract query from the category information of each product for personalized product search research. The user-query pairs are constructed by linking user-item pairs with each items' queries. In our experiments, we used two subsets from the Amazon product dataset[1], which are *Kindle_Store* and *Electronics*, to construct the Amazon simulated search datasets.

CIKMCup dataset. The dataset is from CIKMCup 2016 Track 2[2], which is the only publicly available real-world dataset of product search. The dataset contains five months of logs of an e-commerce search engine from 1st Jan to 1st Jun, 2016. We filter out the interaction records without user ID, query key words or item ID.

Meituan dataset. The dataset is provided by Meituan.com[3], which is one of the most popular search platforms for restaurants and entertainments in China. The dataset contains one day's search records in one of the big city in China, and all data have been anonymized.

The statistic information for these four datasets is shown in Table 2.

[1]https://jmcauley.ucsd.edu/data/amazon/
[2]https://competitions.codalab.org/competitions/11161
[3]https://www.meituan.com/

5.2 Baselines

We compare GraphSRRL model with the following baselines, in which the first baseline is product search method while the rest models are for personalized product search.

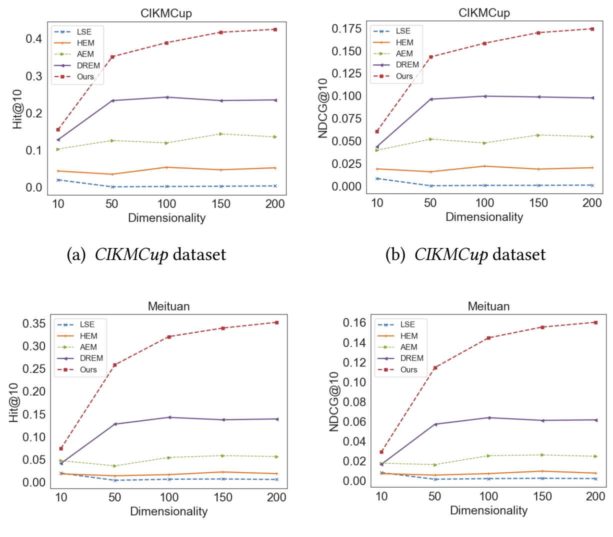
- **LSE** [28] is a representative of product search method, which learns queries and items representations in latent space with their associated keywords or text. In particular, LSE extracts n-grams from reviews of an items and projects them into a latent space with their word embeddings with a non-linear projection function. Products are retrieved based on their similarity with query in the latent space. We compare personalized product search methods with LSE to show the effectiveness of personalization.
- **HEM** [2] is proposed for personalized product search which extends LSE by adding the element of user preference to the product search. HEM learns the embeddings of users, queries, and items by maximizing the likelihood of observed user-query-items triplets.
- **AEM** [1] is a representative of user preference learning methods, which constructs query-dependent user embeddings with attention mechanism over user's historical records.
- **DREM** [3] This model is the state-of-the-art approach for the personalized product search. DREM is also a representative of graph embedding based PPS methods, which integrates multi-relational product information into personalized search. The product embedding is learned from entity/relationship graph.

5.3 Experimental Setup

We split the data by time to simulate the real situation to train and test all models. Specifically, we train all models on the first 70% interaction, and use the next 10% as valid set and last remaining interactions as test set. We measure the performance of algorithms by computing Hit ratio at 10(Hit@10), Mean Reciprocal Rank (MRR), and Normalized Discounted Cumulative Gain at top 10(NDCG@10). Hit@10 is a recall-focused metric measuring whether user interacts with one item in the top 10 results given a query. MRR evaluates the overall ranking performance. NDCG@10 focuses on position of the first retrieved product that user interacts. For every interaction, the ranking of ground truth product is calculated with respect to all items in the dataset. Higher values are better.

For LSE[4], HEM[5], AEM and DREM [6] algorithms, we use the publicly available source codes provided by their inventors to conduct our experiments and find the best hyper-parameters for them via extensive grid search. We implement GraphSRRL [7] with Pytorch[8]. We initialize all embedding vectors randomly and all parameters are initialized using the Xavier uniform method [12]. All algorithm are run for 15 epochs (most models converged after 10 epoches). We used the Adam [21] optimizer with initial learning rate of 0.01 for GraphSRRL and other methods. The regularize parameter is set 0.0001 for all methods. We tune the embedding size d of [10, 50, 100, 150, 200]. We use 50 for all algorithms by default across the comparison. All the models are trained from scratch without any pre-training with a batch size of 256.

[4]https://github.com/cvangysel/SERT.
[5]https://ciir.cs.umass.edu/downloads/HEM/.
[6]https://github.com/utahIRlab/ESRT
[7]https://github.com/Shawn-hub-hit/GraphSRRL-master
[8]https://pytorch.org/



(a) *CIKMCup* dataset (b) *CIKMCup* dataset

(c) *Meituan* dataset (d) *Meituan* dataset

Figure 4: Performances in terms of Hit@10 and NDCG@10 w.r.t embedding size on *CIKMCup* and *Meituan* search datasets. GraghSRRL consistently outperforms other baselines with the various embedding sizes.

5.4 Results and Analysis

Table 3 compares performance of GraphSRRL with the four product search methods in terms of MRR, NDCG@10, and Hit@10 on personalized product search task. From the table, we make the following observations:

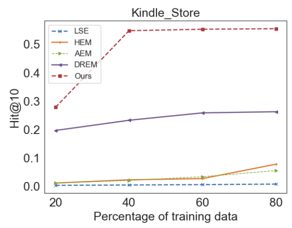
(a) LSE has the worst performance, while all the personalized product search models perform better. In particular, HEM extends LSE with a personalized component. This demonstrates that considering user's preference in product search will yield better results. This is expected because product search is a personal behavior in which user's personal preferences play a vital role.

(b) In general, the proposed method GraphSRRL significantly outperforms all the baselines, including the knowledge graph embedding based method DREM, across all metrics on the four datasets. We calculate the percentage of improvement of GraphSRRL over the best baseline, and we find that the average improvement of GraphSRRL is at least 70.25% in terms of Hit@10 and 73.87% in terms of NDCG@10 across the four datasets. The reason could be that the novel designed relationship representation learning based on conjunctive graph patterns explicitly integrate node relationship and topological structural relationship information into personalized product search.

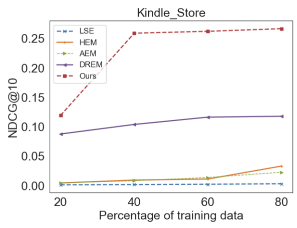
(c) Using graph embedding information to explore a more substantial node relationship can improve the performance of personalized product search. This can be seen from the experimental result

Table 3: Personalized product search experimental results. The best algorithm in each column is highlighted in bold. The last row shows the percentage improvement of GraphSRRL over the best baseline (annotated with * notation). † indicates statistically significant improvement (p < 0.01) by the pairwise t-test over the best baseline.

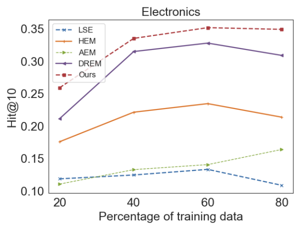
	Kindle_Store			Electronics			CIKMCup			Meituan		
	Hit@10	NDCG@10	MRR	Hit@10	NDCG@10	MRR	Hit@10	NDCG@10	MRR	Hit@10	NDCG@10	MRR
LSE	0.008	0.003	0.005	0.109	0.049	0.059	0.001	0.000	0.001	0.004	0.002	0.003
HEM	0.078	0.033	0.039	0.214	0.096	0.110	0.035	0.016	0.022	0.014	0.006	0.009
AEM	0.056	0.023	0.027	0.243	0.109	0.124	0.126	0.052	0.053	0.036	0.016	0.021
DREM	0.258*	0.115*	0.131*	0.309*	0.140*	0.159*	0.233*	0.096*	0.102*	0.128*	0.057*	0.066*
Ours	**0.555**†	**0.267**†	**0.316**†	**0.349**†	**0.161**†	**0.187**†	**0.352**†	**0.143**†	**0.147**†	**0.258**†	**0.115**†	**0.126**†
Gain	115.1%	131.3%	140.3%	13.0%	15.4%	18.0%	50.7%	48.7%	44.3%	102.3%	100.1%	91.8%



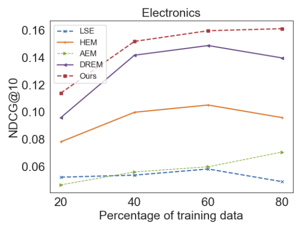
(a) *Kindle_Store* dataset



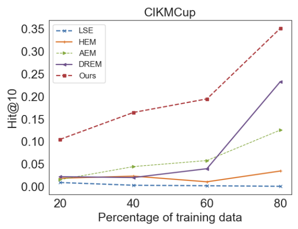
(b) *Kindle_Store* dataset



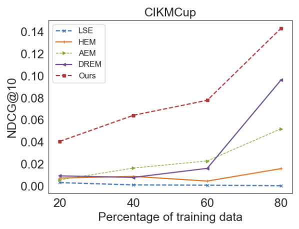
(c) *Electronic* dataset



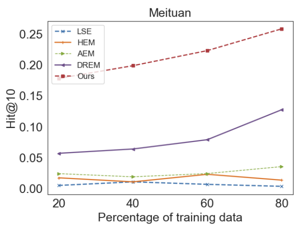
(d) *Electronic* dataset



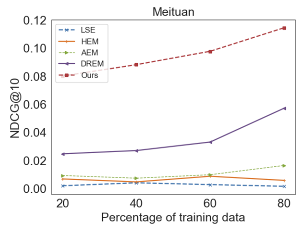
(e) *CIKMCup* dataset



(f) *CIKMCup* dataset



(g) *Meituan* dataset



(h) *Meituan* dataset

Figure 5: Performances of all algorithms in terms of Hit@10 and NDCG@10 by varying the percentage of training data on the four datasets. GraphSRRL consistently has the highest scores.

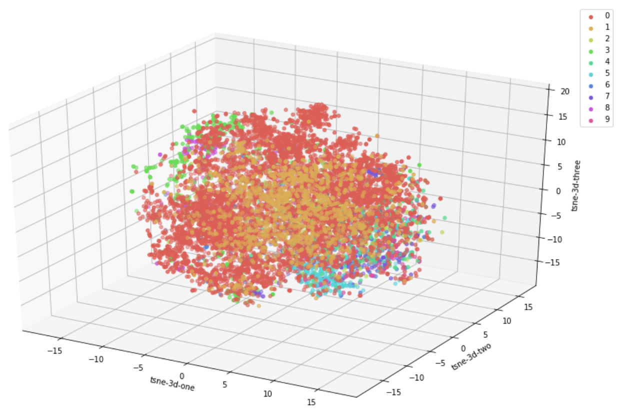
that both GraphSRRL and DREM outperform the models that do not use graph information (namely, HEM and AEM).

(d) As can be seen from Table 3, GraphSRRL shows consistent improvement compared with other methods in both simulated product search dataset and real-world product search dataset, which demonstrates the robustness of the proposed algorithm to the datasets.

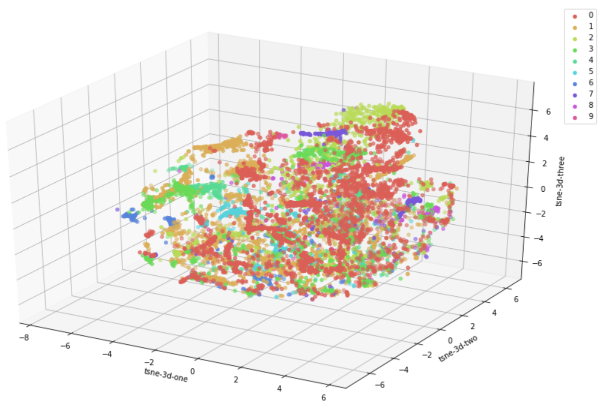
5.5 Sensitivity Analysis

Effect of embedding size. We explore the effect of the embedding size on the performance. To do this, we vary the embedding dimension from 10 to 200 and calculate Hit@10 and NDCG@10 on the two real-world search datasets Meituan and CIKMCup. The results of MRR have the same trend and thus are omitted. The effect trend on other datasets is similar. Figure 4 shows the results. We find that the search performance of GraphSRRL first increases dramatically and then increase slowly with the increase of the embedded size. This could be because over-fitting occurs when the embedded size increases significantly. When the size of embedding dimension increases from 150 to 200, GraphSRRL reaches a plateau but still outperforms the best baselines. Even with a small embedding size of 50, GraphSRRL significantly outperforms baselines with a large embedding dimension. This shows that the proposed algorithm is able to maintain significant improvements of personalized product search performance over the state-of-the-art algorithm.

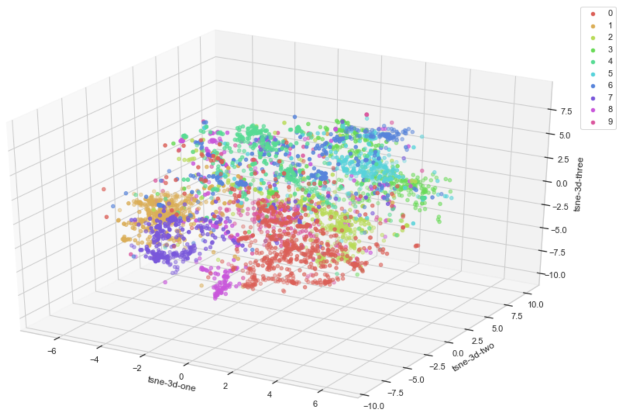
Effect of proportion of training data. In this experiment, we vary the percentage of training data from 20% to 80%. In each case, we take the next 20% interaction as testing set for all competitors. This is to compare the performance on same test data. Figures 5(a-h) show the results in Hit@10 and NDCG@10 of all the algorithms on the four search datasets, as the training data size is increased. The performance of GraphSRRL is stable and does not vary significantly across data points. We note that the performance of GraphSRRL consistently outperforms the baseline models, irrespective of the training data size.



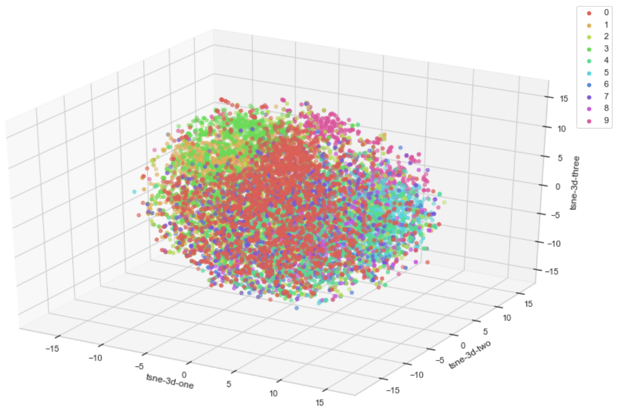
(a) *Kindle_Store* dataset



(b) *Electronic* dataset



(c) *CIKMCup* dataset



(d) *Meituan* dataset

Figure 6: The product embedding visualization. Different colors denote different categories. We find that GraphSRRL has better interpretability of product embedding on the real-world *CIKMCup* and *Meituan* product search datasets than *Kindle_Store* and *Electronics* datasets which are simulated from Amazon review data

5.6 Quality of product embedding

To see whether the product embeddings learned by GraphSRRL are interpretable and can distinguish different types of products, we develop a test for the quality of product embedding. To highlight the effect of visualization, we extract product embedding of 10 categories which have the most products on *Kindle_Store*, *Electronics*, *CIKMCup*, *Meituan* datasets and map those embeddings into 3D with T-SNE [23]. Figure 6 shows the results. From the figure, we can see that GraphSRRL can distinguish different type of products in embedding. Note that GraphSRRL does not use the category information of product but only user-query-product interaction and query words. This shows the high effectiveness of GraphSRRL by integrating structural relationship representation learning with conjunctive graph patterns.

Comparing the visualization results between different datasets, we find that GraphSRRL has better interpretability of product embedding on the real-world *CIKMCup* and *Meituan* product search datasets than *Kindle_Store* and *Electronics* datasets which are simulated from Amazon reviews data. We speculate the reason as follows: the query in *Kindle_Store* and *Electronics* which is extracted from product category randomly cannot reflect the real-world user's search intent and user-query-product structural relationship. For example, when a user searches a query "Apple EarPods“ then buys and reviews an earphone, the category "electronics" which is extracted as the simulated query of user will miss the exact user's intent and structural relationship in user-query-product interaction. Therefore our proposed GraphSRRL performs better on real-world product search datasets, which utilizes and models the real users-queries-products interaction effectively by structural relationship representation learning.

6 CONCLUSIONS AND FUTURE WORK

This paper introduces a novel graph embedding-based structural relationship representation learning approach for personalized product search, which incorporates users-queries-products interactive graph information in neural network models. We propose three types of core conjunctive graph patterns in users-queries-products interactive graph and design corresponding geometric operators to learn the representation, which preserve both affinities between users (resp queries or products) and the structural relationship in users-queries-products interaction. We conduct extensive experiments on four search datasets. Experimental results show that GraphSRRL outperforms significantly other state-of-the-art personalized product search methods. Visualization results show that the latent embeddings of GraphSRRL are highly interpretable on real-world product search datasets. Although GraphSRRL does not employ the product category information, it can distinguish different types of products in embedding by effectively utilizing and modeling users-queries-products interaction.

For future work, we plan to investigate other types of neural information retrieval networks in the GraphSRRL framework. We will also consider to incorporate more substantial information in the users-queries-products interaction graph.

7 ACKNOWLEDGMENTS

This work was supported in part by a MOE Tier-1 grant RG114/19(S) and an NTU ACE grant.

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*. ACM, 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*. ACM, 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] Kamelia Aryafar, Devin Guillory, and Liangjie Hong. 2017. An ensemble-based approach to click-through rate prediction for promoted listings at Etsy. In *Proceedings of the ADKDD'17*. 1–6.

[5] Alexey Borisov, Ilya Markov, Maarten De Rijke, and Pavel Serdyukov. 2016. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*. 531–541.

[6] Nick Craswell and Martin Szummer. 2007. Random walks on the click graph. In *Proceedings of the 30th Annual International ACM SIGIR conference on Research and Development in information retrieval*. 239–246.

[7] Yuxiao Dong, Nitesh V Chawla, and Ananthram Swami. 2017. Metapath2vec: Scalable representation learning for heterogeneous networks. In *Proceedings of the 23rd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 135–144.

[8] Huizhong Duan and ChengXiang Zhai. 2015. Mining coordinated intent representation for entity search and recommendation. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*. 333–342.

[9] 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.

[10] 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.

[11] Jianfeng Gao, Wei Yuan, Xiao Li, Kefeng Deng, and Jian-Yun Nie. 2009. Smoothing clickthrough data for web search ranking. In *Proceedings of the 32nd International ACM SIGIR Conference on Research and Development in Information Retrieval*. 355–362.

[12] Xavier Glorot and Yoshua Bengio. 2010. Understanding the difficulty of training deep feedforward neural networks. In *Proceedings of the 13th International Conference on Artificial Intelligence and Statistics*. 249–256.

[13] Aditya Grover and Jure Leskovec. 2016. Node2vec: Scalable feature learning for networks. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*. 855–864.

[14] 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.

[15] 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.

[16] Will Hamilton, Payal Bajaj, Marinka Zitnik, Dan Jurafsky, and Jure Leskovec. 2018. Embedding logical queries on knowledge graphs. In *Advances in Neural Information Processing Systems*. 2026–2037.

[17] Will Hamilton, Zhitao Ying, and Jure Leskovec. 2017. Inductive representation learning on large graphs. In *Advances in Neural Information Processing Systems*. 1024–1034.

[18] Yujing Hu, Qing Da, Anxiang Zeng, Yang Yu, and Yinghui Xu. 2018. Reinforcement learning to rank in e-commerce search engine: Formalization, analysis, and application. In *Proceedings of the 24th ACM SIGKDD International Conference on Knowledge Discovery & Data Mining*. 368–377.

[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 clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*. 2333–2338.

[20] Shubhra Kanti Karmaker Santu, Parikshit Sondhi, and ChengXiang Zhai. 2017. On application of learning to rank for e-commerce search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*. 475–484.

[21] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[22] Xiao Li, Ye-Yi Wang, and Alex Acero. 2008. Learning query intent from regularized click graphs. In *Proceedings of the 31st International ACM SIGIR Conference on Research and Development in Information Retrieval*. 339–346.

[23] Laurens van der Maaten and Geoffrey Hinton. 2008. Visualizing data using t-SNE. *Journal of Machine Learning Research* 9, Nov (2008), 2579–2605.

[24] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. 2017. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*. 1291–1299.

[25] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Jingfang Xu, and Xueqi Cheng. 2017. Deeprank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*. 257–266.

[26] 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.

[27] Parikshit Sondhi, Mohit Sharma, Pranam Kolari, and ChengXiang Zhai. 2018. A taxonomy of queries for e-commerce search. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*. 1245–1248.

[28] 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*. ACM, 165–174.

[29] 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.

[30] Teng Xiao, Jiaxin Ren, Zaiqiao Meng, Huan Sun, and Shangsong Liang. 2019. Dynamic metric learning for personalized product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 1693–1702.

[31] Hee-Geun Yoon, Hyun-Je Song, Seong-Bae Park, and Se-Young Park. 2016. A translation-based knowledge graph embedding preserving logical property of relations. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*. 907–916.

[32] Jun Yu, Sunil Mohan, Duangmanee Putthividhya, and Weng-Keen Wong. 2014. Latent dirichlet allocation based diversified retrieval for e-commerce search. In *Proceedings of the 7th ACM International Conference on Web Search and Data Mining*. 463–472.

[33] 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.

[34] Yuan Zhang, Dong Wang, and Yan Zhang. 2019. Neural IR meets graph Embedding: A ranking model for product search. In *Proceeding of the 30th World Wide Web Conference*. 2390–2400.

[35] Jie Zou and Evangelos Kanoulas. 2019. Learning to ask: Question-based sequential bayesian product search. In *Proceedings of the 28th ACM International Conference on Information and Knowledge Management*. 369–378.