Learning a Hierarchical Embedding Model for Personalized Product Search

Qingyao Ai[1], Yongfeng Zhang[1], Keping Bi[1], Xu Chen[2], W. Bruce Croft[1]
[1]College of Information and Computer Sciences, University of Massachusetts Amherst
Amherst, MA, USA 01003-9264
{aiqy,yongfeng,kbi,croft}@cs.umass.edu
[2]School of Software, Tsinghua University
Beijing, China 100084
xu-ch14@mails.tsinghua.edu.cn

ABSTRACT

Product search is an important part of online shopping. In contrast to many search tasks, the objectives of product search are not confined to retrieving relevant products. Instead, it focuses on finding items that satisfy the needs of individuals and lead to a user purchase. The unique characteristics of product search make search personalization essential for both customers and e-shopping companies. Purchase behavior is highly personal in online shopping and users often provide rich feedback about their decisions (e.g. product reviews). However, the severe mismatch found in the language of queries, products and users make traditional retrieval models based on bag-of-words assumptions less suitable for personalization in product search. In this paper, we propose a hierarchical embedding model to learn semantic representations for entities (i.e. words, products, users and queries) from different levels with their associated language data. Our contributions are three-fold: (1) our work is one of the initial studies on personalized product search; (2) our hierarchical embedding model is the first latent space model that jointly learns distributed representations for queries, products and users with a deep neural network; (3) each component of our network is designed as a generative model so that the whole structure is explainable and extendable. Following the methodology of previous studies, we constructed personalized product search benchmarks with Amazon product data. Experiments show that our hierarchical embedding model significantly outperforms existing product search baselines on multiple benchmark datasets.

KEYWORDS

Product Search, Personalization, Latent Space Model, Representation Learning

1 INTRODUCTION

Product search represents a special retrieval scenario where users submit queries to retrieve products from a search engine. The most direct application of product search is online shopping. E-shopping has become an important part of our lives today. About 8% (more than 300 billion dollars) of U.S. retail sales came from e-commerce and 71% of U.S. customers shopped online in 2015[1]. In a typical e-shopping scenario, users express their needs through queries submitted to a product search engine and explore the retrieved results to find items of interest (e.g., search on Amazon.com). Therefore, the quality of product search directly affects both user satisfaction with online shopping and the profits of e-commerce companies.

In contrast to traditional ad-hoc retrieval tasks, the concept of relevance can be highly personal in product search. Ad-hoc retrieval tasks, such as web search, focus on retrieving documents that satisfy a user's information need, which is usually related to the query topic. Although personalization is important in web search, it is not as fundamental as it is in product search since users actually want to purchase items from the result list, which is a more personal behavior. On the one hand, while multiple items could be topic-related with a user's query, only a few are actually purchased and different individuals have different opinions even on the same product (such as music CDs). Product search without considering users' differences will not satisfy the needs of all customers. On the other hand, personalization has explicit benefits for e-commerce companies as it potentially increases the chance of users to see the products that they are likely to buy. Retrieving relevant products is less important than finding potential items for purchase because the latter brings direct profits to sellers. Even a small improvement on personalized product search could be worth millions of dollars.

Personalization in product search has both potentials and pitfalls. Users of e-shopping websites often provide rich feedback about their purchases. The reviews written by customers provide information about both product properties and user preferences, which give the search engine more opportunities to learn and understand each individual. Using the review text, however, is not trivial because of the the significant vocabulary mismatch between the language of queries, products and users [30]. For example, the words used in reviews of a TV may not be found in the descriptions of a camera. Without capturing their semantic meanings, user reviews cannot provide useful information for personalized product search on a new query.

The main focus of this paper is to tackle the problem of personalized product search based on language data (i.e. words and reviews). Despite its importance in e-commerce, personalized product search has not been extensively studied so far. To the best of our knowledge, previous work focuses on product recommendation

[1]https://www.readycloud.com/info/ecommerce-statistics-all-retailers-should-know

SIGIR '17, August 07-11, 2017, Shinjuku, Tokyo, Japan

DOI: http://dx.doi.org/10.1145/3077136.3080813

in a non-search scenario [18, 19] or general product search without personalization [30]. To fill this gap, we propose a hierarchical embedding model specifically designed for personalized product search. Inspired by recent progress in distributed representation learning [2, 15], we construct a deep neural network and jointly learn latent representations for queries, products and users. Our hierarchical embedding model has three merits. First, it is a vector space model that represents queries, products and users with latent representations. The vocabulary mismatch problems in personalized product search can be effectively alleviated by conducting product retrieval in our latent semantic space. Second, our model is intentionally designed as a generative model. The likelihood of observed user-query-item triples can be directly inferred with their distributed representations, which makes the whole framework explainable and extendable. Last, our model is trained with stochastic gradient decent, which is efficient for training on GPUs and deployment in real systems. Following the methodology proposed by Gysel et al. [30], we constructed personalized product search benchmarks on Amazon product data and conducted empirical experiments to evaluate the effectiveness of our model. Our hierarchical embedding model significantly outperforms baselines including unigram-based retrieval models and the state-of-the-art latent space model for product retrieval. This demonstrates the potential of personalization with language data in product search.

2 RELATED WORK

There are three lines of research that are directly related to our work: product search, search personalization, and latent space models for information retrieval.

2.1 Product Search

The search function is important for exploring and finding products [14]. Most of the basic product information (i.e. brands, types and categories) can be structured and stored with relational databases. Considerable work has been done on searching products based on their structured aspects [17]. Despite their important applications in e-commerce, searching with structured data is not enough to satisfy the needs of e-shopping users. First, queries of product search users are often in natural language and are hard to structure. Duan et al. [10] noticed that, while languages like SQL are effective for querying structured databases, people tend not to use them in practice because they are difficult to learn. They [9, 10] propose a probabilistic mixture model to analyze product search logs from attribute levels and extend product databases with language modeling approaches to enable conditional search on specifications. Duan et al. [8] also tried to learn query representations for structured product data. Second, there is a gap between the language of product descriptions and free-form user queries. Nurmi et al. [23] reported that users' shopping lists often differ from the product information maintained by retailers. They designed a grocery retrieval system to directly retrieve products using the shopping lists written in natural language. Gysel et al. [30] also noticed the vocabulary mismatch problem existing in product search and introduced a latent vector space model that maps queries and products into a hidden semantic space for product retrieval. These studies are important steps toward language-based product search, but they ignore the effect of users in online shopping and only use the topic relevance between queries and products as their measurement of retrieval quality. In this paper, we focus on personalizing product search for each individual and use the actual purchase behavior as our gold standard.

2.2 Search Personalization

To the best of our knowledge, there are two types of personalized search: search on a personal collections (e.g. email search [6, 24]), and search on a general corpus with personalized result lists (e.g. web search [11, 22, 29]). The product search discussed in this paper is more related to studies on the latter since product collections are equally accessible for most users. In web search, research on personalization focuses on constructing user models with user-specific contents (i.e. queries), behaviors (i.e. click, clicked pages), contexts (i.e. location, time) and uses them to refine the ranked list produced by a global retrieval model [11]. Agichtein et al. [1] studied the clicks of users and introduced a behavior model to measure user preferences with the features extracted from queries, web pages and click-through logs. Teevan et al. [29] measure the ambiguity of web queries and propose a ranking model that incorporates personalizations with different strengths on different queries. In this paper, we focus on a different search scenario where users provide explicit feedback (product reviews) on some search results. To the best of our knowledge, our work is the first study that uses deep neural networks to learn user models from language data for personalized product search.

2.3 Latent Space Models

Latent space models have been widely studied for information retrieval. The basic idea of latent space models is to project both queries and documents into a high dimensional semantic space so that we can directly match their conceptual meanings and avoid vocabulary mismatch problems. Deerwester et al. [7] introduced Latent Semantic Indexing (LSI) and constructed latent vectors for words and documents by factorizing the corpus matrix of term frequency with singular value decomposition (SVD). Hofmanm[13] and Blei et al. [4] proposed pLSI and LDA by assuming that words are sampled from a fixed number of topics and documents are topic distributions. More recently, distributed representation learning with deep neural networks has attracted more attentions. Mikolov et al. [20] proposed a word2vec model which can efficiently learn high quality word embeddings on a large corpus. Le and Mikolov [15] constructed paragraph vector models to simultaneously learn distributed representations for words and documents. It has been shown that the paragraph vector model with distributed bag-of-words assumption (PV-DBOW) implicitly factorizes a tf-icf matrix and constructs a language model that is effective for semantic matching in information retrieval [2]. Inspired by these studies, we design a hierarchical embedding model to jointly learn distributed representations of words, queries, products and users for personalized product search.

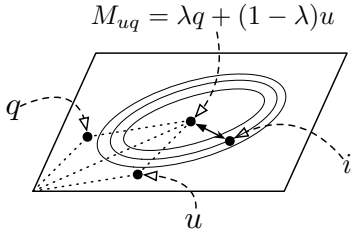


Figure 1: Personalized product search in a latent space with query q, user u, personalized search model M_{uq} and item i.

3 HIERARCHICAL EMBEDDING MODEL FOR PERSONALIZED PRODUCT SEARCH

In this section, we discuss how we tackle the problem of personalized product search with our hierarchical embedding model. In our model, queries, users and items are projected into a single latent space so that their relationships can be directly measured by their similarities. We propose a unified framework which jointly learns different level embeddings through maximizing the likelihood of purchased user-item pair given corresponding queries.

3.1 Personalized Product Search in Latent Semantic Space

For personalized product search, we consider two important factors when designing our retrieval model. The first is *query intent*, which determines whether an item is relevant to a query in general. The second is *user preference*, which decides whether an item satisfies the special need of a particular user. Although the preference of a user may vary depending on the intent of a query, it is unrealistic to construct query-dependent user models because we do not have adequate training data for each user-query pair. For simplicity, we assume that user preferences are independent from query intents and build query-independent user models for personalized product search.

To conduct product search in semantic space and to balance the profit and risk of personalization, we project both queries and users into a single latent space and explicitly control their weights in personalized product search model. Inspired by the design of word embedding models [20, 21], we design the latent representations of queries and users to have good compositionality so that the personalized search model could be directly computed as the linear combination of query models and user models. Formally, suppose that the query intent of a query q in semantic space is represented with a vector $\boldsymbol{q} \in \mathbb{R}^\alpha$ and the user preference of a user u is represented with $\boldsymbol{u} \in \mathbb{R}^\alpha$, we define the personalized search model for (u, q) as:

$$\boldsymbol{M_{uq}} = \lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u} \tag{1}$$

where λ is a hyper-parameter that controls the weight of query model $\boldsymbol{q}$ and user model $\boldsymbol{u}$.

We search products with $\boldsymbol{M_{uq}}$ following the framework of vector space retrieval models. Vector space models measure the relevance of query-document pair with the similarity of their vector representations. Similarly, we rank items according to the similarity between their latent representations and $\boldsymbol{M_{uq}}$. Let $\boldsymbol{i} \in \mathbb{R}^\alpha$ be the latent representation of item i, then the score of i with model M_{uq} can be computed as:

$$Score(i|u, q) = f(\boldsymbol{i}, \boldsymbol{M_{uq}}) = f(\boldsymbol{i}, \lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u}) \tag{2}$$

where f is a similarity function predefined for the latent space of queries, users and items. An illustration of our personalized product search in vector space is shown in Figure 1. The similarity function f in latent space models can be arbitrarily designed in many forms. In our experiments, we tried both cosine similarity and dot product (the sum of element-wise multiplications). We observed that cosine similarity yielded better performance in most cases.

3.2 Hierarchical Embedding Model

We now describe our hierarchical embedding model for personalized product search in detail. In our model, queries, users and items are represented with their associated text data. We define language models for users and items based on their distributed representations and assume that items are generated from a personalized search model constructed with query and user embeddings. We jointly learn embeddings for words, queries, users and items with this hierarchical structure by directly maximizing the likelihood of observed query-user-item triples.

The overall structure of our hierarchical embedding model is shown in Figure 2. Our model can be broadly separated into three parts. The first part of our model maps words to their corresponding word embeddings and constructs distributed representations for users and items by requiring them to predict the words from their associated reviews (R_u and R_i). The second part of our model builds query embeddings with query keywords and function ϕ. Finally, the third part of our model fine-tunes the representations of queries, users and items by requiring the composition of query and user embeddings – the personalized search model M_{uq} – to predict the purchased item. Given this structure, we can directly compute the likelihood of a query-user-item triple and train our model by maximizing the log likelihood of training data.

Embedding-based User/Item Language Model. Inspired by the paragraph vector models [15], we learn the distributed representations for users and items by constructing language models with word embeddings. Formally, given $\boldsymbol{e} \in \mathbb{R}^\alpha$ as the latent representation of an entity (which could be either a user or an item) and $\boldsymbol{w} \in \mathbb{R}^\alpha$ as the embedding of a word w, the probability that w is generated from the language model of e is defined as:

$$P(w|e) = \frac{\exp(\boldsymbol{w} \cdot \boldsymbol{e})}{\sum_{w' \in V_w} \exp(\boldsymbol{w'} \cdot \boldsymbol{e})} \tag{3}$$

where V_w is the corpus vocabulary and $P(w|e)$ is computed as a softmax over $\boldsymbol{e}$ and $\boldsymbol{w}$. For simplicity, we assume that words can be generated by user models and item models independently.

The use of embedding-based language models have two merits. First, through matching with distributed representations, the embedding-based language model alleviates the problem of vocabulary mismatch. It can directly measure the semantic similarity between words and entities in latent space. Second, the construction of embedding-based language models requires no priori knowledge about the corpus's topic distribution (e.g., the topic number in LDA [4]). It can automatically cluster entities based on the characteristics of input data.

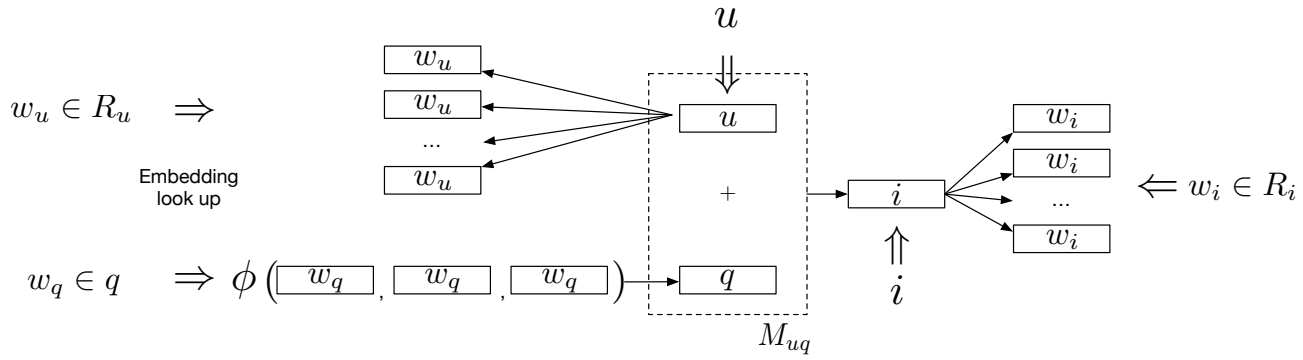


Figure 2: The structure of our hierarchical embedding model for personalized product search. R_u, R_i denote the sets of reviews for user u and item i; w_u, w_i, w_q denote the words in R_u, R_i and query q; and M_{uq} is the personalized search model constructed with query models and user models.

Query Embeddings. As the number of possible queries is very large, query embeddings learned off-line cannot be generalized in practice. Therefore, to construct distributed representations for queries on the fly, we define a projection function ϕ for queries and use the embeddings of query words as its inputs:

$$\boldsymbol{q} = \phi(\{w_q | w_q \in q\}) \tag{4}$$

Previous studies have proposed several methods to combine word embeddings to form a query embedding. The most simplest way is to aggregate and average the embeddings of query words directly [31], which can be formulated as:

$$\phi(\{w_q | w_q \in q\}) = \frac{\sum_{w_q \in q} \boldsymbol{w_q}}{|q|} \tag{5}$$

where $|q|$ is the length of query q. An extension of this method is to add a non-linear projection layer over the average word embeddings and form a new embedding vector [30]:

$$\phi(\{w_q | w_q \in q\}) = \tanh(W \cdot \frac{\sum_{w_q \in q} \boldsymbol{w_q}}{|q|} + b) \tag{6}$$

where $W \in \mathbb{R}^{\alpha \times \alpha}$ and $b \in \mathbb{R}^{\alpha}$ are parameters learned on a separate training set. Further, a more complex model that considers query structures is to sequentially input the query words into a recurrent neural network (RNN) and use the final network state as the latent query representation [25]:

$$\begin{aligned} \boldsymbol{o_t} &= (1 - \boldsymbol{a_t}) \odot \boldsymbol{o_{t-1}} + \boldsymbol{a_t} \odot \boldsymbol{s_t} \\ \boldsymbol{a_t} &= \sigma(W_a^w \boldsymbol{w_q^t} + W_a^s \boldsymbol{o_{t-1}}) \\ \boldsymbol{r_t} &= \sigma(W_r^w \boldsymbol{w_q^t} + W_r^s \boldsymbol{o_{t-1}}) \\ \boldsymbol{s_t} &= \tanh\left(W^w \boldsymbol{w_q^t} + W^s(\boldsymbol{r_t} \odot \boldsymbol{o_{t-1}})\right) \end{aligned} \tag{7}$$

where $\boldsymbol{s_t} \in \mathbb{R}^{\alpha}$ is the state vector on t step, w_q^t is the tth word in query q, $\odot$ is the element-wise product, $\sigma(x) = \frac{1}{1+e^{-x}}$ is a sigmoid function and $W^x, W^s, W_a^x, W_a^s, W_r^x, W_r^s \in \mathbb{R}^{\alpha \times \alpha}$ are parameters in the RNN with Gated Recurrent Unit [5]. The $\phi(\{w_q | w_q \in q\})$, namely the embedding of q, is equal to the final network state $\boldsymbol{s_{|q|}}$.

As far as we know, there is no query embedding method that satisfies the needs of all search scenarios. For example, the mean vector of word embeddings works well on short queries while the recurrent network performs better on long queries with complex linguistic structures. In our experiments, we explored all the methods above to identify the most effective model for query embedding in personalized product search.

Item Generation Model. To fine tune the embeddings of queries, users and items, we further construct an generative model for items that requires user embeddings and query embeddings to predict the items related to them. For users, related items are those purchased by the user; for queries, related items are those relevant to the query. The probability that an item is generated from a user model and a query model together is computed with their embedding representations and a softmax function:

$$P(i|u,q) = Score(i|u,q) = \frac{\exp\left(\boldsymbol{i} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\right)}{\sum_{i' \in V_i} \exp\left(\boldsymbol{i'} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\right)} \tag{8}$$

where V_i is the set of all possible items and λ is the weight of query model in the final ranking function (Equation 2).

The design of the item generation model aims to regularize the representations of users, queries and items so that relevant query-item pairs and preferable user-item pairs have high similarity in the final embedding space. Also, it forms a hierarchical structure that connects the learning of embeddings from different levels. With it, we can directly compute the likelihood of observed user-query-item triples in our hierarchical embedding model.

3.3 Joint Learning Framework

As mentioned previously, we learn the distributed representations of queries, users and items in our hierarchical embedding model by maximizing the likelihood of observed user-query-item triples. Let R_u and R_i be the sets of product reviews that are associated with user u and item i respectively, then the log likelihood of observing a query-user-item triple with corresponding reviews in our model can be computed as

$$\mathcal{L}(R_u, R_i, u, i, q) = \log P(R_u, R_i, u, i, q) \tag{9}$$

In our model, words in R_i are generated by the language model of i and words in R_u are generated by the language model of u. Thus, R_i is independent of u, q, R_u while R_u is independent of i, q, R_i.

Because we assume that user preferences and query intents are independent in personalized product search, we have the following:

$$\begin{aligned}\mathcal{L}(R_u, R_i, u, i, q) &= \log\Big(P(R_i, i|u, q)P(R_u, u, q)\Big)\\ &= \log\Big(P(R_i|i)P(i|u, q)P(R_u|u)P(u)P(q)\Big)\\ &= \log\Big(P(i|u, q)P(u)P(q)\prod_{w_i\in R_i}P(w_i|i)\prod_{w_u\in R_u}P(w_u|u)\Big)\\ &= \log P(i|u, q) + \sum_{w_i\in R_i}\log P(w_i|i) + \sum_{w_u\in R_u}\log P(w_u|u)\end{aligned} \tag{10}$$

where $P(u)$ and $P(q)$ are predefined as uniform distributions, which could be ignored in the computation of log likelihood. Therefore, the log likelihood of a query-user-item triple is actually the sum of log likelihood for the user language model, the item language model and the item generation model.

Directly computing the log likelihood of a query-user-item triple, however, is not practical due to the softmax function used in our hierarchical embedding model (Equation 3&8). For efficient training, we adopt a negative sampling strategy to approximate the softmax function in our model. Negative sampling was first proposed by Mikolov et al. [20] and has been extensively used in machine learning and information retrieval [2, 15]. It has been shown to be effective for approximating softmax functions and factorizing the mutual information matrix of two related entities [16]. The basic idea of negative sampling is to sample data from the corpus with a predefined distribution and form negative samples to approximate the denominator of softmax functions. In our model, the negative samples for language models are the words randomly sampled from the corpus. The log likelihood of a user model or an item model with negative sampling is:

$$\begin{aligned}\log P(w_i|i) &= \log\sigma(\boldsymbol{w_i}\cdot\boldsymbol{i}) + k\cdot\mathbb{E}_{w'\sim P_w}[\log\sigma(-\boldsymbol{w'}\cdot\boldsymbol{i})]\\ \log P(w_u|u) &= \log\sigma(\boldsymbol{w_u}\cdot\boldsymbol{u}) + k\cdot\mathbb{E}_{w'\sim P_w}[\log\sigma(-\boldsymbol{w'}\cdot\boldsymbol{u})]\end{aligned} \tag{11}$$

where k is the number of negative samples and P_w is a noise distribution for words. In our experiments, we define P_w as the unigram distribution raised to the 3/4rd power [20]. Similarly, we compute the log likelihood of our item generation model by conducting negative sampling on items:

$$\begin{aligned}\log P(i|u, q) &= \log\sigma\Big(\boldsymbol{i}\cdot(\lambda\boldsymbol{q} + (1-\lambda)\boldsymbol{u})\Big)\\ &\quad + k\cdot\mathbb{E}_{i'\sim P_i}\Big[\log\sigma\Big(-\boldsymbol{i'}\cdot(\lambda\boldsymbol{q} + (1-\lambda)\boldsymbol{u})\Big)\Big]\end{aligned} \tag{12}$$

where P_i is the uniform noise distribution for items.

We use stochastic gradient descent to learn the parameters of our hierarchical embedding model. All embeddings are trained simultaneously with this joint learning framework. Also, similar to previous studies [2, 30], we add L2 regularizations on the distributed representations of words, users and items. The final optimization goal is:

$$\begin{aligned}\mathcal{L}' &= \sum_{u,i,q}\mathcal{L}(R_u, R_i, u, i, q) + \gamma\Big(\sum_{w\in V_w}\boldsymbol{w}^2 + \sum_{u\in V_u}\boldsymbol{u}^2 + \sum_{i\in V_i}\boldsymbol{i}^2\Big)\\ &= \sum_{u,i,q}\Bigg(\sum_{w_i\in R_i}\Big(\log\sigma(\boldsymbol{w_i}\cdot\boldsymbol{i}) + k\cdot\mathbb{E}_{w'\sim P_w}[\log\sigma(-\boldsymbol{w'}\cdot\boldsymbol{i})]\Big)\\ &\qquad + \sum_{w_u\in R_u}\Big(\log\sigma(\boldsymbol{w_u}\cdot\boldsymbol{u}) + k\cdot\mathbb{E}_{w'\sim P_w}[\log\sigma(-\boldsymbol{w'}\Big)\cdot\boldsymbol{u})]\\ &\qquad + \log\sigma\Big(\boldsymbol{i}\cdot(\lambda\boldsymbol{q} + (1-\lambda)\boldsymbol{u})\Big)\\ &\qquad + k\cdot\mathbb{E}_{i'\sim P_i}\Big[\log\sigma\Big(-\boldsymbol{i'}\cdot(\lambda\boldsymbol{q} + (1-\lambda)\boldsymbol{u})\Big)\Big]\Bigg)\\ &\quad + \gamma\Big(\sum_{w\in V_w}\boldsymbol{w}^2 + \sum_{u\in V_u}\boldsymbol{u}^2 + \sum_{i\in V_i}\boldsymbol{i}^2\Big)\end{aligned} \tag{13}$$

where γ is the strength of L2 regularization; V_w, V_u and V_i are the set of all possible words, users and items respectively.

4 EXPERIMENTAL SETUP

In this section, we introduce our experimental settings for personalized product search. We describe how to extract search queries from product corpus and give details about our data partitions. We also describe the baseline methods used in our experiments and the training settings for our model.

4.1 Datasets

We used the Amazon product dataset[2] as our experiment corpus. This dataset is a well-known benchmark for product recommendation. It includes millions of customers and products as well as rich metadata such as reviews, product descriptions and product categories. In our experiments, we used four subsets from the Amazon product dataset, which are *Electronics*, *Kindle Store*, *CDs & Vinyl* and *Cell Phones & Accessories*. The first three are large-scale datasets that cover three common types of products (electronic devices, books and music). The last one is a small dataset which is used to test our models in situations where text data are limited. Specifically, we use the 5-core data provided by McAuley et al. [18] where each user and each item has at least 5 associated reviews. In these datasets, a user has to purchase an item before writing a review for it. Therefore, we extract purchase user-item pairs directly based on user reviews. The objective of personalized product search in our experiments is to find items that are both relevant to the query and purchased by the user.

4.2 Query Extraction

As far as we know, there is no publicly available dataset that contains search queries for product search. Previous studies in e-shopping has described directed product search as users search for "a producer's name, a brand or a set of terms which described the category of the product" [27]. Therefore, a common query-extraction method for product search research is to extract queries from the category information of each product.

Following the paradigm used by Gysel et al. [30], we extract the search queries for each item with a three-step process. First, we

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

Table 1: Statistics for the 5-core data for *Electronics, Kindle Store, CDs & Vinyl* and *Cell Phones & Accessories* [18].

	Electronics	*Kindle Store*	*CDs & Vinyl*	*Cell Phones & Accessories*
Corpus				
Number of reviews	1,689,188	982,618	1,097,591	194,439
Review length	118.27±158.12	112.21±129.52	174.57±177.05	93.50±131.65
Number of items	63,001	61,934	64,443	10,429
Review per item	26.81±75.82	15.87±21.42	17.03±28.15	18.64±34.24
Number of users	192,403	68,223	75,258	27,879
Review per user	8.78±8.26	14.40±24.61	14.58±39.13	6.97±4.55
Queries				
Number of queries	989	4,603	694	165
Query length	6.40±1.64	7.07±1.89	5.71±1.62	5.93±1.57
Queries per item	1.02±0.23	5.08±2.04	4.04±1.92	1.11±0.38
Queries per user	8.13±5.84	35.65±37.48	21.75±16.53	4.95±2.60
Train/Test				
Number of reviews	1,275,432/413,756	720,006/262,612	804,090/293,501	150,048/44,391
Number of queries	904/85	3313/1290	534/160	134/31
Number of user-query pairs	1,204,928/5,505	1,490,349/232,668	1,287,214/45,490	114,177/665
Relevant items per pairs	1.12±0.48/1.01±0.09	1.87±3.30/1.48±1.94	2.57±6.59/1.30±1.19	1.52±1.13/1.00±0.05

Table 2: Example queries extracted following the paradigm proposed by Gysel et al. [30] from Amazon product data.

Electronics:
– video games playstation accessory kit
– software operate system microsoft window
Kindle Store:
– store kindle ebook cookbook food wine bake dessert
– books health fitness weight loss diet
CDs & Vinyl:
– musical instrument general accessory sheet music folder
– digital music hard rock thrash speed metal
Cell Phones & Accessories:
– cell phone accessory international charger
– cell phone accessory case sleeve

extract category information for each item from the metadata of products. Then, we concatenate the terms from a single hierarchy of categories to form a topic string. Final, stopwords and duplicate words are removed from the topic string and we use it as a query for the corresponding item. To ensure the quality of extracted queries, we ignore the category hierarchies with only one level as those categories are usually non-descriptive for items (e.g. "CDs & Vinyl"). Also, we try to maintain more terms from the sub-categories by removing duplicate words sequentially from the first level to the last level (e.g. *Camera, Photo* → *Digital Camera Lenses* would be converted to "photo digital camera lenses"). Some example queries are shown in Table 2.

For personalized product search, we construct user-query pairs by linking user-item pairs with each item's queries. If a user purchased an item, the pairing of this user with any query associated with the item are valid user-query pairs. Only the items that are purchased by the user and belong to the query are considered as relevant to the user-query pair. For simplicity, we do not conduct any filtering or initial retrieval in our experiments and use all possible items within each dataset as the candidate items for each query. Therefore, the relevant items for each user-query pair are very sparse and the personalized product search settings in our experiments are difficult by nature. More statistics about the subsets of Amazon product data are shown in Table 1.

4.3 Evaluation Methodology

We partitioned each dataset into a training set and a test set according to the following instructions. First, we randomly hide 30% of reviews for each user from the training process. User-item pairs from those reviews are used to represent purchase behaviors in the test data. Second, we randomly select 30% queries as the initial test query set. After that, if all queries of a training item are in the test query set, we randomly select one query and put it back to the training query set. Therefore, each item has at least one query in the training data. Finally, we match all test queries with users to form the final test data. The basic intuition of our setting is to ensure that every query and query-user-item triple in the test set is new and unobserved in the training process. Although the number of queries is limited, we have adequate user-query pairs due to the large number of users. The statistics for data partitions in each Amazon dataset are also shown in Table 1.

For each user-query pair, we compute evaluation metrics based on the top 100 items retrieved by each model. The ranking metrics we used are mean average precision (MAP), mean reciprocal rank (MRR) and normalized discounted cumulative gain (NDCG). Reciprocal rank is the precision on the rank of the first relevant result, which is actually the inversed rank value for the first user purchase in the retrieved items. In other words, MRR indicates the expected number of items that a user needs to explore before finding the "right" product. NDCG is a common metric for multi-label ranking problems. Although we only have binary labels in our settings of personalized product search, the value of NDCG shows how

good the ranking is compared to the optimal ranked list. In our experiments, we compute NDCG at 10.

4.4 Baselines

For model evaluation, we used three types of baselines: the query likelihood model [26] (namely the standard language modeling approach), an extended query likelihood with user models, and the latent semantic entity model [30]. The first two are retrieval models based on bag-of-words representations and the last one is a state-of-the-art latent space model for product search.

Query Likelihood Model. The query likelihood model (QL) is a language modeling approach proposed by Ponte and Croft [26]. It is an unigram model that ranks documents based on the log likelihood of query words in the document's language models. Given a query Q, the probability that Q is generated from a document D is computed as

$$P_{QL}(Q|D) = \sum_{w \in Q} tf_{w,Q} \log \frac{tf_{w,D} + \mu P(w|C)}{|D| + \mu} \tag{14}$$

where $tf_{w,D}$ is the frequency of word w in D, $|D|$ is the length of D, μ is a parameter for Dirichlet smoothing and $P(w|C)$ is a background language model computed as the frequency of w divided by the total number of terms in the corpus C. In our experiments, the document for an item is constructed with the item's reviews. The value of μ are tuned around the average length of each document in the training data (from 1000 to 3000).

Extended Query Likelihood with User Models. The original QL model is not a personalized retrieval model, so we extended it to consider the effect of users in personalized product search. Based on similar assumptions, we define a user-query likelihood model (UQL) that ranks documents according to both the likelihood of query words and the words associated with each user. Formally, let U be the set of words written by a user u, then the likelihood of user-query pair (U, Q) in document model D is

$$P_{UQL}(U, Q|D) = \lambda P_{QL}(Q|D) + (1 - \lambda) P_{QL}(U|D) \tag{15}$$

Similar to Equation 2, we use λ to control the weights of U in retrieval. We tuned λ from 0.0 to 1.0 and show the results in Section 5.1&5.2. To improve efficiency, we removed stop words and used fifty of the most frequent words in U to compute $P_{UQL}(U, Q|D)$.

Latent Semantic Entity. The latent semantic entity model (LSE) proposed by Gysel et al. [30] is a latent space model specifically designed for product search. LSE learns item representations with their associated text data. Specifically, it extracts n-grams from the reviews of an item and projects them into a latent entity space with their word embeddings:

$$f_E(s) = \tanh(W_E \cdot (\frac{1}{|s|} \sum_{w \in s} \boldsymbol{w}) + b) \tag{16}$$

where $|s|$ is the length of a n-gram s, $\boldsymbol{w} \in \mathbb{R}^\alpha$ is the word embedding of word w, $f_E(s) \in \mathbb{R}^\beta$ is the representation of s in the latent entity space, and $W_E \in \mathbb{R}^{\alpha \times \beta}$, $b \in \mathbb{R}^\beta$ are parameters learned in the training process. LSE constructs distributed representation $\boldsymbol{e}$ for item e by maximizing the similarity between e and its n-grams in the latent entity space. Similarly to our hierarchical model, LSE uses negative sampling to define its loss function. However, our model approximates item embeddings by sampling negative words for each item while LSE approximates n-gram representations by sampling negative items for each n-gram. From the perspective of a generative model, the basic assumption of LSE is that each n-gram is a potential query that could generate the corresponding item. Therefore, LSE can directly use Equation 16 to compute the latent representations of queries and do product search by ranking items with their similarities to the query embedding. For simplicity, we set equal sizes for word embeddings and item embeddings ($\alpha = \beta$) in LSE and tuned them from 100 to 500. The best embedding size is 400 for *Electronics*, 300 for *Kindle Store*, 500 for *CDs & Vinyl* and 400 for *Cell Phones & Accessories*.

4.5 Model Training

Both LSE and our models are trained on a Nvidia Titan X GPU with 20 epochs. We set the initial learning rate as 0.5 and gradually decreased it to 0.0 in the training process. We used stochastic gradient decent with batch size 64 and clipped the global norm of parameter gradients with 5 to avoid unstable gradient updates. To speed up training on large datasets (*Electronics*, *Kindle Store* and *CDs & Vinyl*), we subsampled words with probability $10^4 \cdot cf_w/|C|$ where cf_w is the corpus frequency of word w and $|C|$ is the length of the corpus. For LSE and our models, we set negative sampling number as 5 and tuned L2 regularization strength γ from 0.0 to 0.005. We tuned the weight of query model λ (Equation 2&15) from 0.0 to 1.0 and tested embedding size from 100 to 500. The effect of λ and embedding size are shown in Section 5.2&5.3. The training of LSE and our models (except HEM_{RNN}) usually takes 7-8 hours to finish 20 epoch (about 100k words per second) on our largest dataset (*Electronics*). The source code can be found in the link below[3].

5 RESULTS AND DISCUSSION

Now we report our results on personalized product search benchmarks. We first show the overall retrieval performance of our hierarchical embedding models and baselines on different Amazon product datasets. Then we discuss the effect of user models in personalized product search. After that, we analyze the parameter sensitivity of embedding size in our models.

5.1 Retrieval Performance

Table 3 shows the overall results of baselines and our models on the personalized product search benchmarks of Amazon data *Electronics*, *Kindle Store*, *CDs & Vinyl* and *Cell Phones & Accessories*. In the Table 3, QL represents the query likelihood model [26]; UQL represents the extended query likelihood with user models; LSE represents the model of Latent Semantic Entity [30], and HEM denotes our hierarchical embedding models with ϕ function as mean vector (*mean*, Equation 5), projected mean (*pm*, Equation 6) and recurrent neural network (*RNN*, Equation 7). We conducted significant tests over QL, UQL and LSE for all models. All metrics reported in Table 3 are computed based on user purchases, which means that the personalized product search task is difficult by nature and even a small improvement could potentially lead to large profits for e-shopping companies.

[3] https://ciir.cs.umass.edu/downloads/HEM/

Table 3: Comparison of baselines and our hierarchical embedding models on the Amazon product search datasets. MAP and MRR are computed with top 100 items while NDCG is computed with top 10 items. $*$, + and ‡ denote significant differences to QL, UQL and LSE in Fisher randomization test [28] with $p \leq 0.01$. The best performance is highlighted in boldface.

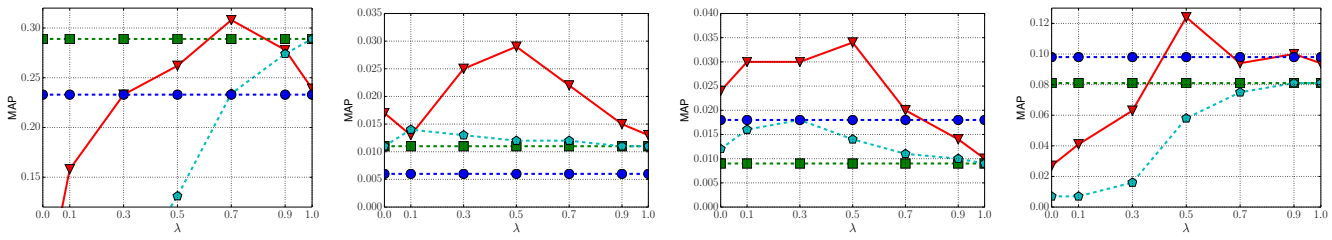
	Electronics			*Kindle Store*			*CDs & Vinyl*			*Cell Phones & Accessories*		
Model	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG	MAP	MRR	NDCG
QL	$0.289^{\dagger}$	$0.289^{\dagger}$	$0.316^{\dagger}$	$0.011^{\dagger}$	$0.012^{\dagger}$	$0.013^{\dagger}$	0.009	0.011	0.010	0.081	0.081	0.092
UQL	$0.289^{\dagger}$	$0.289^{\dagger}$	$0.316^{\dagger}$	$0.014^{*\dagger}$	$0.016^{*\dagger}$	$0.019^{*\dagger}$	0.018^{*}	0.021^{*}	0.021^{*}	0.081	0.081	0.092
LSE	0.233	0.234	0.239	0.006	0.007	0.007	0.018^{*}	0.022^{*}	0.020^{*}	0.098^{*+}	0.098^{*+}	0.084
HEM_{mean}	0.071	0.071	0.091	$0.015^{*+\dagger}$	$0.019^{*+\dagger}$	$0.018^{*\dagger}$	$0.029^{*+\dagger}$	$0.035^{*+\dagger}$	$0.034^{*+\dagger}$	0.047	0.047	0.053
HEM_{pm}	$\mathbf{0.308}^{*+\dagger}$	$\mathbf{0.309}^{*+\dagger}$	$\mathbf{0.329}^{\dagger}$	$0.029^{*+\dagger}$	$0.035^{*+\dagger}$	$0.033^{*+\dagger}$	$\mathbf{0.034}^{*+\dagger}$	$\mathbf{0.040}^{*+\dagger}$	$\mathbf{0.040}^{*+\dagger}$	$\mathbf{0.124}^{*+}$	$\mathbf{0.124}^{*+}$	$\mathbf{0.153}^{*+\dagger}$
HEM_{RNN}	0.198	0.198	0.214	$\mathbf{0.033}^{*+\dagger}$	$\mathbf{0.039}^{*+\dagger}$	$\mathbf{0.038}^{*+\dagger}$	$0.023^{*+\dagger}$	$0.027^{*+\dagger}$	$0.026^{*+\dagger}$	0.053	0.053	0.071

As shown in Table 3, the overall performance of baselines and our models varies considerably on different product datasets. According to the results for baseline models (QL, UQL and LSE), *Electronics* and *Cell Phones & Accessories* are "easy" datasets while *Kindle Store* and *CDs & Vinyl* are "hard" datasets. Empirically, there are multiple reasons that make *Electronics* and *Cell Phones & Accessories* much easier then *Kindle Store* and *CDs & Vinyl* in personalized product search. From the perspective of data content, *Kindle Store* and *CDs & Vinyl* contain items about books and music while *Electronics* and *Cell Phones & Accessories* consist of items about electronic devices. The tastes of books and music are more personal and difficult to capture compared to electronic devices. Also, the average reviews per item in *Kindle Store* and *CDs & Vinyl* are lower (15.87 and 17.03) than those in *Electronics* and *Cell Phones & Accessories* (26.81 and 18.64), which makes the modeling of items less adequate in both baselines and our models. From the perspective of queries, most items in *Electronics* and *Cell Phones & Accessories* are related only to 1 query while items in *Kindle Store* and *CDs & Vinyl* are related to 4 or 5 queries on average. For each user, there are more items that belong to the same queries but haven't been purchased in *Kindle Store* and *CDs & Vinyl*. The language for queries in *Electronics* and *Cell Phones & Accessories* showed high correlations with the language for user purchases. For example, the MAP of QL is much higher on *Electronics* and *Cell Phones & Accessories* (0.289 and 0.081) than it is on *Kindle Store* and *CDs & Vinyl* (0.011 and 0.008).

The relative performance of unigram models (QL and UQL) compared to latent space models (LSE, HEM) also varies on different datasets. On "easy" datasets such as *Electronics* and *Cell Phones and Accessories*, the performance of QL and UQL is comparable or better than the latent space baseline (LSE) and some variations of our hierarchical embedding models (HEM_{mean} and HEM_{RNN}). On difficult datasets like *Kindle Store* and *CDs & Vinyl*, however, vocabulary mismatch problems are more severe and unigram models are significantly worse than latent space models. Overall, our best model (HEM_{pm}) outperformed QL and UQL on all four datasets. The improvement of MAP over QL and UQL is 0.019 (7%) on *Electronics*, 0.018 (164%) and 0.015 (107%) on *Kindle Store*, 0.026 (325%) and 0.016 (89%) on *CDs & Vinyl*, and 0.043 (53%) on *Cell Phones and Accessories*. These results indicate that exact keyword matching is not enough to predict user purchases in product search. In many cases, the semantic relationships between queries, users and products considerably affect the purchase decisions of users.

Compared to LSE, we notice that the HEM models indeed produce better results on the tasks of personalized product search. Our best model (HEM_{pm}) outperformed LSE on MAP for 0.075 (32%) on *Electronics*, 0.023 (383%) on *Kindle Store*, 0.016 (89%) on *CDs & Vinyl* and 0.026 (27%) on *Cell Phones & Accessories*. There are two potential reasons for the good performance of our models. First, compared to LSE, our models explicitly construct user models with user's reviews. Purchase is a personal behavior and user models enable us to retrieve products according the preference of each individual. Second, our models are designed based on more general assumptions for queries, users and items. In LSE, each n-gram is considered as a potential query. Gysel et al. [30] conducted negative sampling by sampling items for each n-gram, which basically assumes that items are generated from models of n-grams. In contrast, we assume that words are generated from the models of items and items are generated from both query models and user models. We believe that items are more complex concepts and should be placed at a higher level than basic semantic units like words and n-grams.

The main differences between the variations of our hierarchical embedding models in Table 3 are their ϕ functions for query embedding. According to our experiments, HEM_{pm} is the most effective and robust model while HEM_{mean} is the worst one. Previous studies [31, 32] have shown that, despite the good compositionality of word embeddings, aggregating word embeddings directly to form query embeddings for information retrieval is not promising. In our experiments, we observed inferior performance for HEM_{mean} in Table 3. After adding a non-linear projection layer over the average word embeddings, however, our HEM_{pm} obtained significantly better results on almost all datasets. This indicates that the relation between queries and words is non-linear in semantic space. Also, we notice that the performance of the projected mean (*pm*) on three of our datasets is even better than RNN, which is considered to be a complex and powerful neural network in general. One possible reason is that the queries used in our personalized product search benchmarks are mostly keyword-based queries. As discussed in previous studies [12, 30], keyword-based queries in document retrieval and entity retrieval tend to be simple in structure and do not have complicated compositional meanings. Therefore, using neural networks as complex as RNN in our hierarchical embedding models brings little benefit to the process of query modeling and potentially increases the risk of model over-fitting.



(a) *Electronics* (b) *Kindle Store* (c) *CDs & Vinyl* (d) *Cell Phones & Accessories*

Figure 3: The performance of HEM$_{pm}$ and baselines on the Amazon personalized product search benchmark datasets with different query model weight λ. The red solid line with triangles represents the numbers of HEM$_{pm}$; the blue, green and cyan dashed lines with circles, squares and pentagons are results for LSE, QL and UQL respectively.

5.2 Personalization Weight

In our hierarchical embedding models, we define a hyper-parameter λ to control the weight of user models in personalized product search. To analyze the effect of personalization in our models, we plot the MAP value of baselines and HEM$_{pm}$ with respect to λ in Figure 3. When λ is equal to 1.0, our model purely relies on query models to retrieve items for all users; when λ is equal to 0.0, our model only uses user models to find items for each individual.

As we can see in Figure 3, the optimal performance of our hierarchical embedding models is a tradeoff between query relevance and user preference. The personalized search model of the hierarchical embedding model is the linear composition of query models (query embeddings) and user models (user embeddings). When we do not consider queries in personalized product search ($\lambda = 0.0$), HEM$_{pm}$ ranks items purely based on the preference of users and had poor performance on all datasets. When we conducted product search without personalization ($\lambda = 1.0$), HEM$_{pm}$ obtained fair results on *Electronics* and *Cell Phones & Accessories* but still performed worse than our best models. The best value of λ for HEM$_{pm}$ is 0.7 on *Electronics*, 0.5 on *Kindle Store*, *CDs & Vinyl* and *Cell Phones & Accessories*. As λ increased from 0.0 to 1.0, the performance of UQL increased in the beginning and decreased after 0.1 on *Kindle Store* and 0.3 *CDs & Vinyl*. On *Electronics* and *Cell Phones & Accessories*, however, the best UQL is the UQL with $\lambda = 1.0$, which is actually a QL model without using user reviews.

As discussed previously, the types of products not only influence the difficulty of product search but also affect the usefulness of personalization. According to our experiments, the needs of personalization on books (*Kindle Store*) and music (*CDs & Vinyl*) are higher than those on electronic devices (*Electronics*). Also, because each item belongs to one query in *Electronics* and *Cell Phones & Accessories* on average, the test queries are strong filters for items by themselves, which partially explains why using only the query models ($\lambda = 1.0$) still produced good results on these datasets.

5.3 Embedding Size

To analyze the effect of embedding sizes and potentially provide guides for future studies, we show the results of our hierarchical embedding models with different embedding sizes in Figure 4. The horizontal axes represent the size of embedding vectors for queries, users and items in our experiments.

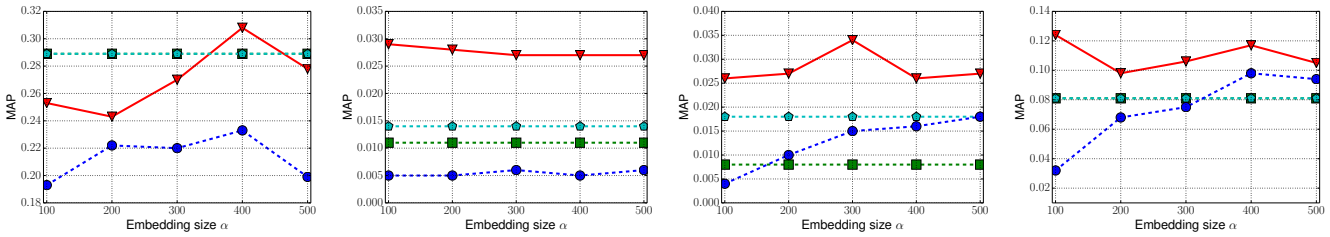
Similar to personalization weights, we observed that the needs of high dimensional embedding vectors vary on different datasets. On *Cell Phones & Accessories* and *Kindle Store*, HEM$_{pm}$ with embedding size 100 obtained the best performance on MAP. Higher embedding sizes brought no improvement but higher training cost and overfitting risks on these datasets. On *Electronics* and *CDs & Vinyl*, we observed better performance of HEM$_{pm}$ with embedding size larger than 100. The best embedding sizes for *Electronics* and *CDs & Vinyl* are 400 and 300. Overall, the performance of HEM$_{pm}$ are robust to the change of embedding sizes and outperformed the baseline models in most cases.

Arora et al. [3] conducted both empirical and theoretical analyses on word embedding models and argued that low dimension vectors (i.e. 300 dimensions) were already enough to encode the information needed by many natural language processing tasks. Because our models incorporate more complicated relationships between queries, users and items, larger embedding sizes could potentially lead to better performance in personalized product search. Nonetheless, we suggest starting with relatively low dimensional vectors and increasing embedding sizes latter if necessary.

6 CONCLUSION AND FUTURE WORK

In this paper, we introduce a hierarchical embedding model for personalized product search. Our model is a latent space retrieval model which projects queries, users and items into a semantic space and conducts product retrieval according to the semantic similarity between items and the composition of query and user models (the personalized search model). We design our neural embedding model in a generative way so that the distributed representations of queries, users, and items can be learned through optimizing the likelihood of observed query-user-item triples. Our results showed that our model significantly outperformed the state-of-the-art baselines on Amazon benchmarks and indicate that personalization with review text is fruitful for product search.

In our experiments, the performance of personalized product search and the importance of user models vary considerably from one dataset to another. We explained this phenomenon empirically according to the general statistics of different datasets such as the number of reviews per item, the number of queries per items etc., but more quantitative analyses are still required if we want to understand each dataset in detail. Those analyses would provide



(a) *Electronics* (b) *Kindle Store* (c) *CDs & Vinyl* (d) *Cell Phones & Accessories*

Figure 4: The performance of HEM$_{pm}$ and baselines on the Amazon personalized product search benchmark datasets with different embedding size α. The red solid line with triangles represents the numbers of HEM$_{pm}$; the blue, green and cyan dashed lines with circles, squares and pentagons are results for LSE, QL and UQL respectively.

important guidelines for system design in e-shopping websites. Also, in our work, we only explored the potential of language data in personalized product search. In e-shopping websites, however, there is other user feedback that we haven't used, such as clicks, ratings and frequently asked questions (FAQ). How to effectively combine information from different resources for personalized product search remains to be an open question. We believe that studies in this direction would have great potential and real influence on e-commerce in the future.

7 ACKNOWLEDGMENTS

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF IIS-1160894. Any opinions, findings and conclusions or recommendations expressed in this material are those of the authors and do not necessarily reflect those of the sponsor.

REFERENCES

[1] Eugene Agichtein, Eric Brill, Susan Dumais, and Robert Ragno. 2006. Learning user interaction models for predicting web search result preferences. In *Proceedings of the 29th ACM SIGIR*. ACM, 3–10.
[2] Qingyao Ai, Liu Yang, Jiafeng Guo, and W Bruce Croft. 2016. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the ACM ICTIR'16*. ACM, 133–142.
[3] Sanjeev Arora, Yuanzhi Li, Yingyu Liang, Tengyu Ma, and Andrej Risteski. 2015. Rand-walk: A latent variable model approach to word embeddings. *arXiv preprint arXiv:1502.03520* (2015).
[4] David M Blei, Andrew Y Ng, and Michael I Jordan. 2003. Latent dirichlet allocation. *Journal of machine Learning research* 3, Jan (2003), 993–1022.
[5] Kyunghyun Cho, Bart Van Merriënboer, Caglar Gulcehre, Dzmitry Bahdanau, Fethi Bougares, Holger Schwenk, and Yoshua Bengio. 2014. Learning phrase representations using RNN encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).
[6] Nick Craswell, Hugo Zaragoza, and Stephen Robertson. 2005. Microsoft Cambridge at TREC 14: Enterprise Track.. In *TREC*.
[7] Scott Deerwester, Susan T Dumais, George W Furnas, Thomas K Landauer, and Richard Harshman. 1990. Indexing by latent semantic analysis. *Journal of the American society for information science* 41, 6 (1990), 391.
[8] Huizhong Duan and ChengXiang Zhai. 2015. Mining Coordinated Intent Representation for Entity Search and Recommendation. In *Proceedings of the 24th ACM CIKM*. ACM, 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 CIKM*. ACM, 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] Susan T Dumais. 2014. Personalized Search: Potential and Pitfalls.. In *NTCIR*.
[12] Jiafeng Guo, Yixing Fan, Qingyao Ai, and W Bruce Croft. 2016. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM CIKM*. ACM, 55–64.
[13] Thomas Hofmann. 1999. Probabilistic latent semantic indexing. In *Proceedings of the 22nd ACM SIGIR*. ACM, 50–57.
[14] Bernard J Jansen and Paulo R Molina. 2006. The effectiveness of Web search engines for retrieving relevant ecommerce links. *Information Processing & Management* 42, 4 (2006), 1075–1098.
[15] Quoc V Le and Tomas Mikolov. 2014. Distributed Representations of Sentences and Documents.. In *ICML*, Vol. 14. 1188–1196.
[16] Omer Levy and Yoav Goldberg. 2014. Neural word embedding as implicit matrix factorization. In *Advances in neural information processing systems*. 2177–2185.
[17] Soon Chong Johnson Lim, Ying Liu, and Wing Bun Lee. 2010. Multi-facet product information search and retrieval using semantically annotated product family ontology. *Information Processing & Management* 46, 4 (2010), 479–493.
[18] Julian McAuley, Rahul Pandey, and Jure Leskovec. 2015. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD*. ACM, 785–794.
[19] Julian McAuley, Christopher Targett, Qinfeng Shi, and Anton van den Hengel. 2015. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th ACM SIGIR*. ACM, 43–52.
[20] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. 2013. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).
[21] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. 2013. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems*. 3111–3119.
[22] Meredith Ringel Morris, Jaime Teevan, and Steve Bush. 2008. Enhancing collaborative web search with personalization: groupization, smart splitting, and group hit-highlighting. In *Proceedings of the 2008 ACM conference on Computer supported cooperative work*. ACM, 481–484.
[23] Petteri Nurmi, Eemil Lagerspetz, Wray Buntine, Patrik Floréen, and Joonas Kukkonen. 2008. Product retrieval for grocery stores. In *Proceedings of the 31st ACM SIGIR*. ACM, 781–782.
[24] Paul Ogilvie and Jamie Callan. 2005. Experiments with Language Models for Known-Item Finding of E-mail Messages.. In *TREC*.
[25] Hamid Palangi, Li Deng, Yelong Shen, Jianfeng Gao, Xiaodong He, Jianshu Chen, Xinying Song, and Rabab Ward. 2016. Deep sentence embedding using long short-term memory networks: Analysis and application to information retrieval. *IEEE/ACM Transactions on ASLP* 24, 4 (2016), 694–707.
[26] Jay M Ponte and W Bruce Croft. 1998. A language modeling approach to information retrieval. In *Proceedings of the 21st ACM SIGIR*. ACM, 275–281.
[27] Jennifer Rowley. 2000. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing* 17, 1 (2000), 20–35.
[28] Mark D Smucker, James Allan, and Ben Carterette. 2007. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM CIKM*. ACM, 623–632.
[29] Jaime Teevan, Susan T Dumais, and Daniel J Liebling. 2008. To personalize or not to personalize: modeling queries with variation in user intent. In *Proceedings of the 31st ACM SIGIR*. ACM, 163–170.
[30] Christophe Van Gysel, Maarten de Rijke, and Evangelos Kanoulas. 2016. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM CIKM*. ACM, 165–174.
[31] Ivan Vulić and Marie-Francine Moens. 2015. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th ACM SIGIR*. ACM, 363–372.
[32] Hamed Zamani and W Bruce Croft. 2016. Estimating embedding vectors for queries. In *Proceedings of the ACM ICTIR'16*. ACM, 123–132.