Neural Generative Models and Representation Learning for Information Retrieval

Qingyao Ai

NEURAL GENERATIVE MODELS AND REPRESENTATION LEARNING FOR INFORMATION RETRIEVAL

A Dissertation Presented

by

QINGYAO AI

Submitted to the Graduate School of the
University of Massachusetts Amherst in partial fulfillment
of the requirements for the degree of

DOCTOR OF PHILOSOPHY

September 2019

College of Information and Computer Sciences

NEURAL GENERATIVE MODELS AND REPRESENTATION LEARNING FOR INFORMATION RETRIEVAL

A Dissertation Presented

by

QINGYAO AI

Approved as to style and content by:

W. Bruce Croft, Chair

James Allan, Member

Erik G. Learned-Miller, Member

Rajesh Bhatt, Member

Jiafeng Guo, Member

James Allan, Department Head
College of Information and Computer Sciences

To mom and dad

ACKNOWLEDGMENTS

First and foremost, I would like to thank my adviser, W. Bruce Croft, without whom this dissertation would not have been possible. Bruce taught me how to conduct research and, more importantly, how to think critically and independently. He gave me all the support I needed for developing my research projects, and he also gave me great freedom to explore the research problems that I'm excited about. These help me build my own research agenda and have a deep influence on my future career.

I would like to thank my committee members, James Allan, Erik Learned-Miller, Rajesh Bhatt, and Jiafeng Guo. They provided important support and valuable comments on my thesis proposal and dissertation, which help me better formulate the problems and successfully defend the thesis. Especially, I would like to thank Jiafeng for helping me overcome many problems in the early days of my Ph.D. He gave me great guidance on how to develop a research idea and write a good research paper. He is not only a great friend but also a wonderful mentor to me.

I would like to thank all the mentors and friends who helped me in the past of my research career. I thank Yiqun Liu for introducing me to the field of information retrieval; I thank Susan Dumais and Nick Craswell for teaching me how to conduct solid work and studies in Microsoft Research; I thank Xuanhui Wang and Michael Bendersky for giving me the opportunities to work on cutting edge problems with the best people in Google; and I thank Daniel Hill and Choonhui Teo for helping me and giving me better understanding of real users and customers during my time at Amazon Search.

I would like to thank all the staffs of our lab, Kate Morruzzi, Jean Joyce, Dan Parker, Stephen Harding, Michael Zarozinski and Glenn Stowell for their great technical and logistical support. I cannot finish my Ph.D. such easily without their help. Also, I thank our Graduate Program Manager Eileen Hamel, who helped me a lot for my Ph.D. program.

I thank all my friends and colleagues in the lab for creating such a productive and friendly working environment. In alphabet order: Keping Bi, Hamed Bonab, Ethem Can, Daniel Cohen, Laura Dietz, Shiri Dori-Hacohen, John FoleyHelia Hashemi, Myung-ha Jang, Jiepu Jiang, Youngwoo Kim, Chia-Jung Lee, Yen-Chieh Lien, Ali Montazeralghaem, Shahrzad Naseri, Chen Qu, Sheikh Muhammad Sarwar, Manmeet Singh, Lakshmi Vikraman, Liu Yang, and Hamed Zamani.

Finally, I would like to thank my parents and my girlfriend Pingping for their immense love and support for me. They are the light that guides me in the darkest moments. I cannot accomplish this without them. It is my great fortune and happiness to have them in my life.

This work was supported in part by the Center for Intelligent Information Retrieval and in part by NSF grant number 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.

ABSTRACT

NEURAL GENERATIVE MODELS AND REPRESENTATION LEARNING FOR INFORMATION RETRIEVAL

SEPTEMBER 2019

QINGYAO AI

B.S., TSINGHUA UNIVERSITY

M.S., UNIVERSITY OF MASSACHUSETTS, AMHERST

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

Information Retrieval (IR) concerns about the structure, analysis, organization, storage, and retrieval of information. Among different retrieval models proposed in the past decades, generative retrieval models, especially those under the statistical probabilistic framework, are one of the most popular techniques that have been widely applied to Information Retrieval problems. While they are famous for their well-grounded theory and good empirical performance in text retrieval, their applications in IR are often limited by their complexity and low extendability in the modeling of high-dimensional information. Recently, advances in deep learning techniques provide new opportunities for representation learning and generative models for information retrieval. In contrast to statistical models, neural models have much more flexibility because they model information and data correlation in latent spaces without explicitly relying on any prior knowledge. Previous studies on pattern recognition and natural language processing have shown that semantically meaningful representations of text, images, and many types of information can be acquired with neural models through supervised or unsupervised training. Nonetheless, the effectiveness of neural models for information retrieval is mostly unexplored. In this thesis, we study how to develop new generative models and representation learning frameworks with neural models for information retrieval. Specifically, our contributions include three main components: (1) *Theoretical Analysis*: We present the first theoretical analysis and adaptation of existing neural embedding models for ad-hoc retrieval tasks; (2) *Design Practice*: Based on our experience and knowledge, we show how to design an embedding-based neural generative model for practical information retrieval tasks such as personalized product search; And (3) *Generic Framework*: We further generalize our proposed neural generative framework for complicated heterogeneous information retrieval scenarios that concern text, images, knowledge entities, and their relationships. Empirical results show that the proposed neural generative framework can effectively learn information representations and construct retrieval models that outperform the state-of-the-art systems in a variety of IR tasks.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1

INTRODUCTION

Information Retrieval is a field concerned with the structure, analysis, organization, storage, and retrieval of information. Typically, given an explicit or implicit information need, the goal of information retrieval systems is to efficiently and effectively find materials that are *relevant* in the current search or recommendation context. In web search, for example, this means retrieving and ranking web pages based on whether they can satisfy the user's need expressed with text-based queries. Therefore, there are two fundamental challenges in the studies of information retrieval – how to represent information, and how to retrieve information by matching materials with information needs.

1.1 Information Representation and Generative Models

Ever since the beginning of human civilization, people are searching for effective methods to express information. One of the most well-known examples is the creation of language. Language is a special communication method that stores, inherits, and transmits information with symbols (i.e., words) and special rules that connects them (i.e., grammar). Based on this paradigm, the simplest way to retrieve relevant information is to express both materials and information needs with language and match them accordingly.

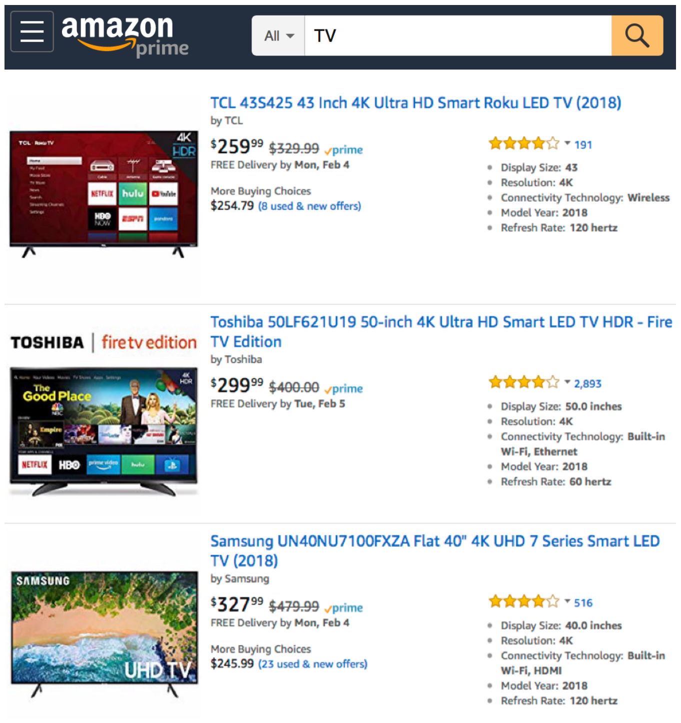
To conduct efficient language matching with machines, the *bag-of-words* representation has been proposed and widely used in information retrieval systems. The idea of bag-of-words representations is to represent information with a set of words so that each material and information need can be converted to fixed-length vectors that are processable by machines. By sacrificing the context information hidden in the linguistic structure of language, bag-of-words representations achieves great efficiency while maintaining surprisingly good effectiveness at the same time. They can precisely keep the key information of each material while maintaining simple data structures that are easy to store and index. Thus, considerable information retrieval models have been proposed based on bag-of-words representations.

Broadly speaking, existing retrieval models based on bag-of-words representations can be broadly categorized into two groups – discriminative models and generative models. Discriminative models usually extract matching features for each pair of material (e.g., document) and information need (e.g., query) to analyze their relevance. One of the first discriminative models proposed for information retrieval is the vector space model [88], which retrieve and rank documents based on their similarities to the current query in bag-of-words representations. More complicated discriminative methods include learning-to-rank models that apply machine learning algorithms to predict the relevance score of each material based on their feature vectors [61]. Generative models, on the other hand, focus on modeling the generation process of relevant information (*relevance model*) based on statistical probabilistic frameworks so that documents can be retrieved and ranked by their probabilities to be generated from the relevance model. In ad-hoc retrieval scenarios, where each document is a piece of text and users search documents with queries composed by several keywords, example generative models include the binary independence retrieval model [84] and the language modeling approach [80]. Due to their good empirical performance and well-grounded theoretical frameworks, generative retrieval models are popular and widely studied in the research community of information retrieval.

Nonetheless, there are several problems of existing generative retrieval models constructed based on the bag-of-words representations that restrict their applications in information retrieval. One of the most well-known examples is vocabulary mismatch. In many cases, the words people used to construct their queries may not be the same as those used by the creator of the corpus. We refer to this phenomenon as the *vocabulary gap*. While we can easily identify the appearance of a keyword in documents with bag-of-words representations and inverted indexing techniques, it is difficult to analyze the semantic relationships between words and documents with them. Therefore, finding relevant information with bag-of-words representations could be challenging when there are vocabulary gap between queries and documents, even when the words they used are semantically related.

The most straightforward method to handle vocabulary mismatch is by constructing information representations that can capture the semantic meanings of words and documents. Therefore, the studies of latent semantic representations have received a lot of attention in information retrieval. Instead of representing words with high-dimensional sparse vectors that are orthogonal to each other (e.g., bag-of-words), latent semantic representations create low-dimensional dense vectors for text so that their semantic similarities and relationships can be mathematically measured in the latent space. For example, a famous line of studies is the topic modeling approaches that convert words and documents into probabilistic distributions over a fixed number of hidden topics. In the topic space, words and documents could have non-binary relationships that can be measured with the similarity of their topic distributions. Again, statistical generative models are popular in the scope of latent representation learning due to their well-grounded theories and explainable frameworks.

Unfortunately, while the latent representations learned from topic models can alleviate the vocabulary mismatch problem of statistical generative models based on bag-of-words, they suffer from high computational cost and, more importantly, low generalization ability in information retrieval tasks. In many cases, the needs of information retrieval problems involve more than language and topic matching. For



TCL 43S425 43 Inch 4K Ultra HD Smart Roku LED TV (2018)
by TCL
$259.99 $329.99 prime
FREE Delivery by Mon, Feb 4
More Buying Choices
$254.79 (8 used & new offers)
191
- Display Size: 43
- Resolution: 4K
- Connectivity Technology: Wireless
- Model Year: 2018
- Refresh Rate: 120 hertz

Toshiba 50LF621U19 50-inch 4K Ultra HD Smart LED TV HDR - Fire TV Edition
by Toshiba
$299.99 $400.00 prime
FREE Delivery by Tue, Feb 5
2,893
- Display Size: 50.0 inches
- Resolution: 4K
- Connectivity Technology: Built-in Wi-Fi, Ethernet
- Model Year: 2018
- Refresh Rate: 60 hertz

Samsung UN40NU7100FXZA Flat 40" 4K UHD 7 Series Smart LED TV (2018)
by Samsung
$327.99 $479.99 prime
FREE Delivery by Mon, Feb 4
More Buying Choices
$245.99 (23 used & new offers)
516
- Display Size: 40.0 inches
- Resolution: 4K
- Connectivity Technology: Built-in Wi-Fi, HDMI
- Model Year: 2018
- Refresh Rate: 120 hertz

Figure 1.1: An example of how heterogeneous information creates challenges for generative models in information retrieval tasks.

instance, Figure 1.1 shows an example search result page of Amazon.com where we issue a query of “TV” in order to purchase a television. Usually, having the query word “TV” in the product’s title is not the reason why we prefer this product over others. There is a variety of information hidden in the appearance, prices, and brands of each product that affect the final purchase decisions of each user, and this information is not explicitly expressed in text data. Designing a generative model that can jointly model heterogeneous information from different sources, however, is difficult under existing statistical frameworks. It can require significant human effort and expertise to identify the intrinsic relationships between two types of information. Therefore, new paradigms are needed for the study of generative models and representation learning for information retrieval.

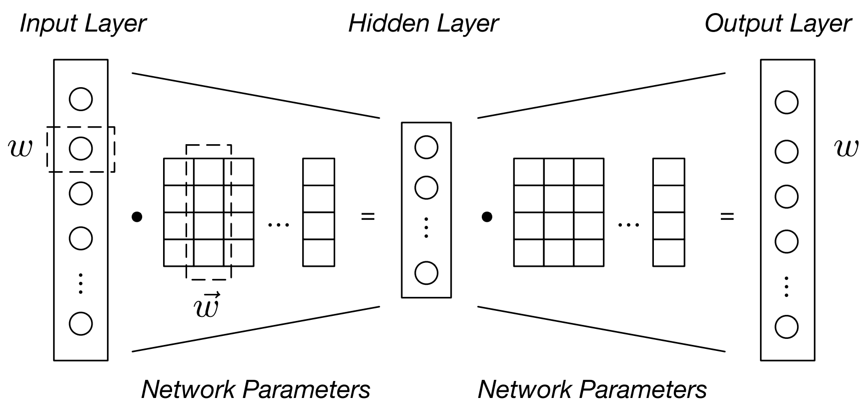


Figure 1.2: An example of neural language models based on a multi-layer neural network. The input layer is the bag-of-words vector of a piece of text (where w is one possible word in it), and the output layer is the distribution of generation probability for each word given the text and the model.

1.2 Neural Embedding Framework

Recently, advances in deep learning techniques provide new opportunities for representation learning and generative models in information retrieval. Deep learning, which is vaguely inspired by information processing and communication patterns in biological nervous systems, represents a broader family of machine learning methods based on a set of algorithms that attempt to model high-level abstractions of data with artificial neural networks[1]. One of the most well-known deep learning architectures is the *deep neural network* (DNN) that are constructed with multi-layer perceptrons built on linear or non-linear functions. Typically, the structure of a deep neural network includes an input layer, which converts raw data into a machine-processable format, a couple of hidden layers, which transform and process the input data with linear/non-linear projections, and an output layer, which produces the final results needed for each specific task.

Figure 1.2 shows an example structure of DNN built for the task of language modeling. Here, the object of the model is to predict the probability of observing

[1]`https://en.wikipedia.org/wiki/Deep_learning`

each word in the vocabulary within a certain piece of text. The input layer of the network is a raw representation of the text (e.g., bag-of-words vectors), the output layer of the network is a probability distribution over words in the vocabulary, and the input and output are connected with a hidden layer and several projection functions parameterized with vectors or matrixes of variables. Usually, network parameters are learned by optimizing a task-specific loss function defined over the model output in the *training* process. Depending on how the loss functions are designed, deep neural networks and other deep learning architectures can be broadly categorized as *supervised* methods, which compute training loss based on human annotations, *unsupervised* methods, which construct loss functions with unlabeled data, and *semi-supervised* methods, which train neural networks with both human annotated data and unlabeled data.

In fact, DNN-based language models are similar to traditional topic modeling approaches as they both create latent representations for words and texts. As we can see in Figure 1.2, the input data (i.e., the bag-of-words vector of the text) in the input layer is multiplied with a matrix of parameters to construct a latent representation in the hidden layer. Suppose that w is a word in the vocabulary V, which represents one dimension in the input vector, then the computation of the hidden representation $\vec{h}$ given the text d can be formulated as

$$\vec{h} = \sum_{w \in V} \#(w, d) \cdot \vec{w} \tag{1.1}$$

where $\#(w, d)$ is the number of times that w appears in d, and $\vec{w}$ is one column of the parameter matrix that corresponds to the dimension of w in the input vector. In other words, the DNN model creates dense vector $\vec{w}$ for each word w in its parameter space so that we can compute the latent representation of the text d (i.e. $\vec{h}$) based on the distributed representations of words. Thus, we refer to the distributed representation of w as the *embedding* of w, and models that represent each information entity with a numerical vector in the network space as *Neural Embedding Models*.

Previous studies have proven that neural embedding models can effectively acquire semantically meaningful representations of words and text. Ever since Miklov et al. [72] introduced the famous word embedding technique *word2vec* in 2013, neural embedding models have significantly changed the landscape of many research fields including natural language processing and social network analysis. It has been shown that the distributed representations of words built with neural networks and unsupervised learning can significantly outperform traditional models built with human annotations and statistical analysis in synonyms identification (e.g., "large" and "big"), word analogy analysis (e.g., "big" to "biggest" as "small" to "smallest"), and many natural language processing applications such as machine translation and reading comprehension. Also, given a proper design of the network, the word representations learned with neural embedding models could capture important semantic relationships and have good compositionality (e.g., the embedding of "queen" is similar to the embedding of "king" minus "man").

Behind the power of neural embedding models is a flexible model framework. While it is not the first paradigm proposed for latent semantic representation learning, the neural embedding framework is no doubt the "simplest" one. The design of a neural embedding model could require no prior knowledge of the corpus's topic distributions, and have no restriction on how the network topological structure should be constructed. All parameters can be learned directly from optimizing predefined loss functions with standard gradient descent methods. This essentially means that we could apply any linguistic rules or heuristic knowledge into the design of the model. For example, Devlin et al. [31] introduce a novel method that builds word embedding with neural transformer networks and combines the optimization objectives of language modeling and next sentence prediction in a single model. The proposed BERT model successfully beats the state-of-the-art representation learning techniques in a variety of natural language processing tasks.

Because the computation of artificial neural networks mostly built on simple functions such as matrix multiplication and maximum pooling, the training of neural embedding models could be highly efficient and scalable. Today, advanced Graphics Processing Units (GPUs) can handle trillions of simple mathematical operations in seconds. This means that we could train a neural model with millions of parameters on millions of documents in a couple of hours. In addition, most computations in the training of neural embedding models are orthogonal, which can be easily distributed among thousands of machines without affecting the final performance of the model. With distributed programming frameworks such as Tensorflow[2] and PyTorch [3], neural models have already been successfully deployed on large computer clusters and serve for the online products of Google, Microsoft, Facebook, and etc.

1.3 Neural Generative Models for Information Retrieval

In the thesis, we develop new techniques that combine the merits of generative models and neural embedding models for information retrieval. Our goal is to create a generic representation learning framework that can capture complicated relationships between structured and unstructured data. Specifically, we construct distributed representations with neural embedding models by directly modeling the generation of relevant information in search. The final outcome of this thesis is a neural generative model with flexible structures that can be easily extended and applied to a majority of information retrieval tasks.

[2]`https://www.tensorflow.org`

[3]`https://pytorch.org`

As discussed in the rest of this section, this thesis identifies and tackles three fundamental challenges for the applications of neural generative models in information retrieval: (1) how to theoretically adapt neural embedding models for retrieval proposes, (2) how to construct neural models under a generative retrieval framework, and (3) how to generalize the neural generative framework to capture complicated relationships between arbitrary types of data for information retrieval.

1.3.1 How to adapt neural embedding models for information retrieval

Although the introduction of distributed representation learning with neural embedding models is revolutionary in the community of natural language processing, its value had not been well-recognized in information retrieval until recent years. As discussed previously, neural embedding models are highly flexible in terms of model structures and parameter optimization. While this makes them handy in tasks with straightforward goals (e.g., predict the next term in a sentence given previous terms), it also creates challenges for its adaption to information retrieval – there is limited theoretical guidance for how to design learning objectives for search applications. Information retrieval concerns about the evaluation and ranking of information based on their relevance – an abstract relationship between queries and documents that involves multiple factors, such as keyword matching, term salience, topic diversity, etc. Without theoretical design principles, it is difficult to develop well-grounded neural embedding models that suit the need of information retrieval. Previous studies that directly applying word embeddings learned from natural language processing tasks to information retrieval applications often lead to poor experiment results and low system extendability. In this dissertation, we explicate the theoretical foundation of neural embedding models and their optimization process, and propose several modifications to existing neural embedding techniques that can improve their performance in information retrieval tasks.

1.3.2 How to combine neural embeddings with generative models

While the combination of classic bag-of-words representations and distributed representations learned with unsupervised neural embedding models indeed brings benefits to statistical generative retrieval models, this paradigm is considered suboptimal in practice. To the best of our knowledge, most unsupervised neural embedding models are optimized based on the linguistic semantics and relationships of text. The objectives of information retrieval problems, however, often involve factors beyond the literal meanings of queries and documents. For instance, in product search, users who search for "cell phone" would not purchase a product only because it has the terms "cell phone" in its description. Capturing the implicit relevance between queries and documents requires us to organically integrate the power of neural embedding models with generative retrieval frameworks and directly optimize them for information retrieval problems. To address this problem, we propose an embedding-based neural generative model that simultaneously models the generation of information and their relevance with distributed information representations.

1.3.3 How to build a generic neural generative framework

In general information retrieval scenarios, complicated relationships and correlations often exist between information entities from different sources. Exploiting heterogeneous information with statistical generative retrieval models, however, is challenging. In most cases, it requires significant human effort and expertise to identify the intrinsic relationships between two types of information. Most existing studies focus on a simplified problem in which documents only contain homogeneous information stored in multiple fields (e.g. text in title, description, and body) [77, 54, 53]. They assume that information is independent in different fields and try to find heuristic weighting schemes that combine the results of generative models built in each field separately. There have been some attempts [51, 117] to tackle more complicated tasks such as cross-media retrieval, but their methods also rely on human heuristics to combine information. Thus, these models are domain-specific and cannot be generalized to incorporate other types of information. In this thesis, we extend neural generative models by creating a generic representation learning framework to incorporate heterogeneous information and multi-relational data for information retrieval.

1.4 Contributions

The contribution of this thesis can be summarized as follows:

- **Theoretical analysis on neural embedding models for IR**. In Chapter 3, we conduct a theoretic analysis of neural embedding models for information retrieval. Specifically, we focus on a famous unsupervised embedding model, i.e. the *Paragraph Vector Model* [58], and a classic *ad-hoc retrieval* task where users specify their information need through a query that initiates a search for documents that are likely to be relevant to them [9]. While paragraph vector models show impressive performance in many natural language processing tasks, integrating it with traditional retrieval models, however, produces unstable performance and limited improvements. In the first chapter of this thesis, we formally discuss three intrinsic problems of the original paragraph vector model that restrict its performance in retrieval tasks: (1) The unregulated training process of the paragraph vector is vulnerable to short document over-fitting that produces length bias in the final retrieval model; (2) The corpus-based negative sampling of the paragraph vector leads to a weighting scheme for words that overly suppresses the importance of frequent words; And (3) the lack of word-context information makes paragraph vector unable to capture word substitution relationships. Accordingly, we propose three modifications to the paragraph vector model that make it more suitable for ad-hoc retrieval. The analysis results and proposed techniques provide important theoretical guidance for the future adaption of neural embedding models for information retrieval.

- **A design example of embedding-based neural generative retrieval models**. To combine the merits of generative retrieval models and neural embedding models for information retrieval, in Chapter 4, we propose an embedding-based neural generative model that simultaneously models the generation of information and their relevance with distributed information representations. Our experimental test bed is a product search task where users issue text queries to find items that satisfy their information needs and purchase intents. In product search, user behaviors are often affected by their personal preferences and there is a severe mismatch between the language of queries, products, and users. To handle this problem, we propose to learn semantic representations for products, users and queries from different levels with a hierarchical neural embedding model. Specifically, we build a multi-level probabilistic generative model that requires the representations of products to predict the words in their reviews and the representations of users and queries to predict the products that have been purchased in each search session. Ranking is conducted based on the probability of each product to be generated by the user model and the query model. The proposed model not only has the good theoretical foundations and explainability of generative models but also inherits the advantages of neural embedding models in terms of effectiveness and flexibility.

- **An extendable and explainable neural representation learning framework**. To further develop a generic neural generative framework for information retrieval, in Chapter 5, we propose to jointly model both entities and their relationships with neural embeddings. Again, we use product search as our experimental test bed. Product search is a well-established information retrieval tasks which naturally involve structured (e.g., prices, ratings, brands, categories, etc.) and unstructured data (e.g., titles, reviews, images, etc.) from heterogeneous sources. Understanding the relationships between these data is difficult under a statistical generative framework but essential for the effectiveness of a product search engine. To address this problem, we propose an extendable and explainable neural generative framework that creates a session-dependent knowledge graph based on multi-relational data. Despite their types, all entities are embedded into latent semantic spaces and all relationships are modeled as linear transpositions that connect one entity to another. For any entity pair (e_h, e_t) with relation r (where e_h is often referred to as the *head entity* and e_t is referred to as the *tail entity*), we create generative models that generate the tail entities e_t by transforming the head entities e_h with relation transposition functions based on r and their distributed representations. We also define "search and purchase" as a dynamic relation between users and products so that we can retrieve items according to their probability to be generated from the user model and "search and purchase" relation. The proposed neural generative model is extendable as it can incorporate any types of information in its training process, and it also creates a graph embedding structure that naturally supports logical inference and reasoning.

1.5 Dissertation Outline

The remainder of the dissertation is organized as follows:

- In Chapter 2, we survey the related work of this thesis, which include the generative retrieval models based on bag-of-words representations (e.g., language modeling approaches, BM25, etc.) and the latent semantic models for information retrieval (e.g., Latent Semantic Indexing, LDA-based language modeling approach, etc.). We also provide necessary background information for deep learning techniques and different retrieval applications.

- In Chapter 3, we present our effort on adapting unsupervised neural embedding models for ad-hoc document retrieval. This include a theoretical analysis of the optimization process of the paragraph vector models and empirical experiments on combining neural language models with generative retrieval models based on statistical probabilistic frameworks.

- In Chapter 4, we discuss how to construct embedding-based neural generative models for product search. We formulate the relationship between products, queries and users in a generative way and learn distributed representations for them simultaneously by optimizing the posterior likelihood of observed user interactions in product search.

- In Chapter 5, we further extend the embedding-based neural generative model to a generic representation learning framework for search on multi-relational data and heterogeneous information. The proposed model framework has good extensibility and transparency, and our experiments show that it achieves the state-of-the-art performance on product retrieval through the joint modeling of multiple product information such as reviews, brands, categories, etc.

- In Chapter 6, we summarize our contributions and discuss the potential directions for future studies.

CHAPTER 2
BACKGROUND AND RELATED WORK

In this chapter, we review the research topics related to this thesis. They are statistical generative models, deep learning techniques, and the information retrieval applications we use for our test bed. We also briefly discuss the existing neural models for information retrieval.

2.1 Statistical Generative Models for IR

The research on statistical generative models has a long history in the information retrieval (IR) community started from the mid-1900s. Numerous studies in the early years of IR focused on developing probabilistic frameworks to rank documents according to the posterior probability of relevance [26], which we refer to as the *classic probabilistic models*. Towards the end of the 1990s, a new family of probabilistic models that try to estimate a probability distribution that captures statistical regularities of the document's language emerged and quickly became popular [63]. These models, which we refer to as the *language modeling approaches* for IR, share a common root with statistical generative models used in many research fields such as Natural Language Processing and Speech Recognition. Their design principles have been used as the foundation of many generative retrieval models including those discussed in this dissertation. In the meantime, another line of studies that focus on constructing latent semantic information representations with statistic probabilistic framework gradually receive more and more attention in the field. By capturing the semantic meanings of information beyond bag-of-words, these studies kick off a new direction for the design and application of generative models for information retrieval. In this section, we give a brief introduction of different statistical generative models proposed for retrieval and representation learning in IR.

2.1.1 Classic Probabilistic Models

In classic probabilistic models, relevance is considered as a basic, dichotomous criterion variable defined independently with information retrieval systems [83]. Given this idea, the retrieval of information can be formulated as a process of ranking documents according to the probability ranking principle (PRP) as

> ***Probability Ranking Principle*** *(PRP)* [83]: *If a reference retrieval system's response to each request is a ranking of the documents in the collections in order of decreasing probability of usefulness to the user who submitted the request, where the probabilities are estimated as accurately as possible on the basis of whatever data has been made available to the system for this purpose, then the overall effectiveness of the system to its users will be the best that is obtainable on the basis of that data.*

Therefore, the key to probabilistic retrieval models is to compute the probability that a document would be relevant to a query.

Let d, q and r be random variables that denote a document, a query and their relevance, respectively. In the classic probabilistic models for IR, the probability that d and q have relevance r ($P(d, q|r)$) is formulated as $P(d, q|r) = P(q|r) \cdot P(d|q, r)$ [26], where the relevance model r is separately modeled with the documents to rank. One of the best-known examples is the Binary Independence Retrieval Model (BIR), which is also known as the Okapi model, the City model and the Robertson-Sparck Jones model [70, 26]. In the BIR models, documents are represented as bag-of-words vectors and term occurrences are assumed to be conditionally independent given a relevance class. Thus, the probability that d is generated by the relevance model of q can be directly estimated by how likely its terms will appear in q's relevant documents. Because we cannot acquire this relevance information in advance, many techniques have been proposed to estimate it based on the statistics of words and corpus, including Tree Dependence Model [23], 2-Poisson Model [44, 84, 85], and BM25 [86].

Take BM25 as an example. The BM25 ranking function is inspired by the theory of relevance generation in the BIR model [83] and the 2-Poisson model [84]. It assumes that term frequencies are distributed according to a mixture of two Poissons, but estimates the probability that document d is generated by the relevance model of a query q with a heuristic function tuned empirically in the experiments. Specifically, BM25 ranks documents with the probability of d given q and relevance r as

$$P(d|q, r = 1) \approx \sum_{w \in q \cap d} \cdot \#(w, d) \frac{(k+1)\#(w, d)}{k\big((1-b) + b\frac{|d|}{|d|_{avg}}\big) + \#(w, d)} \log \frac{N - df_w + 0.5}{df_w + 0.5} \quad (2.1)$$

where $\#(w, d)$ is the term frequency of word w in d, $|d|$ is the length of d, $|d|_{avg}$ is the average document length of the corpus, df_w is the number of documents in which w appears, and N is the total number of documents in the corpus. The effect of term frequency and document length in final results are controlled by hyper-parameter k and b. As we can see, the final forms of classic probabilistic models usually are constructed based on term statistics and are straight forward to compute.

2.1.2 Language Modeling Approach

The first language modeling approach for IR was proposed by Ponte and Croft [80] in the late 1990s, and it initiated a new branch of studies on statistical generative retrieval models in 2000s. Different from classic probabilistic models, language modeling approaches do not explicitly separate the relevance model from the modeling of documents. Instead, it factors the probability of a document d having relevance r to a query q as $P(d, q|r) = P(d|r) \cdot P(q|d, r)$ [26]. In other words, language modeling approaches assume that a query is generated from relevant documents that satisfy the corresponding information need, and a retrieval system can retrieve relevant documents by ranking them according to how likely they can generate the query.

To estimate $P(q|d,r)$, the original language modeling approach treats queries and documents as bags of words and computes the query likelihood as the posterior probability of observing each query word (unigram) given the document [80]. Specifically, the probability of a query q given a document d can be derived with maximum likelihood estimation as

$$P(q|d) = \prod_{w \in q} P(w|d) = \prod_{w \in q} \frac{\#(w,d)}{|d|} \tag{2.2}$$

Unfortunately, naive methods that construct language models based on the bag-of-words representations fail when the query words are not observed in documents. Therefore, an important direction of research on language modeling approaches is how to effectively smooth the estimation of word probabilities. To solve this problem, most existing work adopts a simple paradigm that incorporates a corpus language model for unobserved query words. Example techniques include the Jelinek-Mercer method, absolute discounting, and Bayesian smoothing with Dirichlet priors [112]. For instance, the famous *Query Likelihood* model (QL) with Dirichlet smoothing is defined as

$$P(q|d) = \sum_{w \in q} \#(w,q) \log \frac{\#(w,d) + \mu P(w|C)}{|d| + \mu} \tag{2.3}$$

where $P(w|C)$ is the posterior probability of observing word w in the corpus C, which is usually computed as the term frequency of w in C divided by the number of words in C; $|d|$ is the length of document d; μ is a hyper-parameter for Dirichlet distribution; and $\#(w,q)$ and $\#(w,d)$ are the term frequency of w in q and d, respectively.

While the first language modeling approach for IR is constructed based on unigrams, more sophisticated models that consider bigrams, trigrams and their importance were developed under the same framework later in the 2000s [39, 68, 32, 11]. For example, Gao et al. [39] introduce a dependence language model that estimates the generation probability of bigrams for information retrieval. Metzler et al. [68] propose the sequential dependence model that jointly combine the language models of unigrams and bigrams in ordered and unordered windows with the markov random field. These models generally produce the state-of-the-art performance in a variety of retrieval tasks including ad-hoc retrieval and verbose query analysis [57, 69, 12, 13].

2.1.3 Latent Semantic Modeling

As discussed in Section 1.1, statistical generative models with bag-of-words representations often suffer from the vocabulary mismatch between queries and documents. Because classic probabilistic models and language modeling approaches for IR compute document scores purely based on term statistics, they cannot explicitly model the semantic relationships between words and documents. Thus, another line of studies on statistical generative models focuses on developing better representations that capture the high-level semantic meanings of information. One of the famous examples is the Latent Semantic Indexing (LSI) techniques proposed by Deerwester et al.[29] in 1990. The idea of LSI is to construct a term frequency matrix for words and documents in the corpus and factorize it with SVD analysis. The output vectors of SVD for each dimension of the matrix can be treated as latent representations for each word and document. In this latent parameter space, words and documents with similar meanings tend to be close to each other. Later, Hoffman [47] constructs a Probabilistic Latent Semantic Indexing (pLSI) that represents documents as mixtures of topics. Given a fixed topic pool Z, the probability of a word w generated by the language model of document d can be computed as

$$P(w|d) = \sum_{z \in Z} P(w|z)P(z|\theta_d) \tag{2.4}$$

where z is the language model of a topic in Z and θ_d is the topic model of d. Further, Blei et al. [16] extended pLSI by drawing topic mixtures from a conjugate Dirichlet priori. The topic distributions of words and documents can further be used as their latent semantic representations. The semantic relationships between information can be directly measured in latent space so that vocabulary mismatch would be less of a problem. Previous studies have shown that these latent semantic representations are effective for language smoothing in IR applications [62, 99].

The neural generative models studied in this thesis share a similar design principle with the topic models as they both try to model the generation of words from documents at a semantic level. However, the existing topic models are built under the statistical probabilistic framework while our proposed model directly models the correlations between words and documents with neural networks.

2.2 Deep Learning

The advance of deep learning techniques has significantly changed the landscape of machine learning studies and applications in the 2010s. Artificial neural network models have produced the state-of-the-art performance in a variety of Computer Vision (CV), Pattern Recognition (PR) and Natural Language Processing (NLP) tasks, and its effectiveness in IR problems have been recognized by more and more researchers today. In this section, we introduce some basic concepts of deep learning that are necessary for the understanding of this dissertation.

2.2.1 Neural Architecture

Deep learning refers to a broader family of machine learning methods based on algorithms that tackle specific tasks or model high-level abstractions of data with artificial neural networks[1]. As far as we know, there is no theoretical restriction

[1] `https://en.wikipedia.org/wiki/Deep_learning`

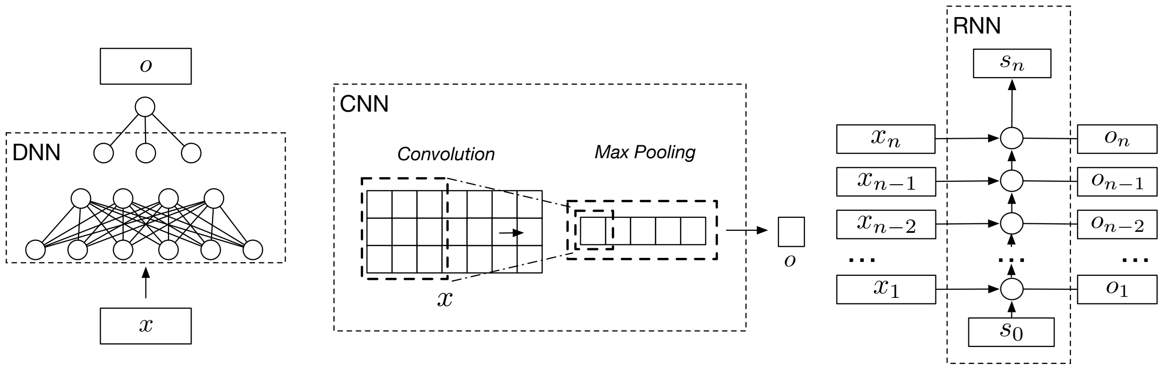


Figure 2.1: An illustration of three popular architectures of deep learning models, which are the Deep Neural Network (DNN), the Convolutional Neural Network (CNN), and the Recurrent Neural Network (RNN). The inputs and outputs of each network are referred to as x and o, respectively.

on how the topological structure of a neural network should be constructed, so the design of deep learning models could be fairly flexible and diverse. In general, there are three types of neural networks that are popular among many machine learning related tasks, which are Deep Neural Network (DNN), Convolutional Neural Network (CNN), and Recurrent Neural Network (RNN).

DNN refers to the multi-layer neural networks that transform the input data with stacks of linear or non-linear projections in order to achieve certain objects (e.g., label predictions, data compression, etc.). As shown in Figure 2.1, given the input data x, the DNN model converts x to the output data o with multi-layer perceptrons. Both the input and the output of the DNN could have arbitrary dimensions depending on the applications.

CNN is similar to DNN as they both have multi-layer networks in their main structures. The main difference is that CNN models usually include one or multiple layers of convolution processes that project the input data with sliding windows and kernel functions. Therefore, CNN is more capable of capturing local data patterns with different granularity [2]. Also, in order to convert the final output into a fixed size vector or matrix, CNN models often apply a pooling layer over the convolution layer.

For instance, the max pooling of CNN in Figure 2.1 would convert the convolution output into a single value o by picking the maximum number in the vector.

RNN is a special deep learning architecture that does not explicitly have multiple layers in its network structure. Instead, it uses a "single" layer network to sequentially read and encode input data as a list. Let the input data be a list of vectors $\{x_1, x_2, ..., x_n\}$ and the initial state of RNN be s_0, then RNN takes one input vector x_i in the ith step and interacts it with the state vector s_{i-1} to produce an output o_i and the new state s_i, as shown in Figure 2.1. Due to this structure, RNN is particularly good at sequential modeling and context understanding [1].

In this dissertation, most proposed models under our neural generative framework are designed with architectures similar to DNN. As discussed in Section 1.2, DNN models naturally construct latent representations for input data with their network parameters. Our goal is to learn such representations in a generative manner and directly optimize the performance of retrieval systems.

2.2.2 Parameter Optimization

The training of a neural model usually involve three steps – defining a loss function, computing the training loss and parameter gradients, and update the parameters. Depending on how the training data are constructed, the training process of neural models can be broadly categorized into three groups: supervised training, unsupervised training, and semi-supervised training. Supervised training refer to the methods where loss functions are computed based on model outputs and human annotated data. For example, the training of a digit recognition model is considered as a supervised method as we manually provide labels for each input picture and teach the model to recognize the digit in pictures accordingly. Unsupervised training refer to the cases where loss functions are computed directly based on the input data without human annotation. In Section 1.2, we describe a DNN model where the input is the bag-of-words representation of a document and the output is a probability distribution of words. Because we can directly train the model by requiring it to predict the words that have been observed in the documents, it is an unsupervised method that doesn't use any human labeled data. Semi-supervised methods, in contrast, is a combination of supervised training and unsupervised training. It usually includes parameter optimizations on a small set of human annotated data as well as on a large set of unlabeled data. Most proposed models described in this thesis could be seen as semi-supervised methods.

In order to update model parameters in the training process, a variety of optimization strategies have been proposed for deep learning. The most popular one is the Stochastic Gradient Descent (SGD) that computes parameter gradients with respect to the training loss on batches of data and update parameters by applying the gradients with a fixed learning rate. There are also complicated methods that jointly consider the current parameter gradients and history information in previous training steps, such as Adagrad [35] and Adam [55].

2.3 Information Retrieval Applications

Information retrieval applications such as web search and product search have significantly changed people's lives in the last 30 years. Depending on how the queries and materials are stored and represented, search problems in IR can be categorized into two groups: search on homogeneous information and search on heterogeneous information.

2.3.1 Homogeneous Information and Ad-hoc Retrieval

Search on homogeneous information refers to the scenarios where information needs and materials are constructed and stored in similar forms. One of the most well-known examples is *ad-hoc retrieval* where both queries and documents are represented with unstructured text. In a standard ad-hoc retrieval task, queries are formulated with a couple of keywords while documents are formulated with sentences and articles with variable lengths. Because they share the same information units (i.e., words), matching queries and documents in their raw forms is theoretically principled and straight forward to do. Therefore, ad-hoc retrieval models often focus on modeling the matching patterns between queries and documents directly [38]. For example, classic probabilistic models and language modeling approaches for IR computes document scores based on matching query unigrams with document unigrams [84, 86, 80]. Complicated statistical retrieval models incorporate bigram matching and trigram matching in the construction of ranking functions [36, 27]. Also, there are many studies focusing on modeling the term dependency and matching proximity in words and documents [39, 73, 75, 68]. In this dissertation, we use ad-hoc retrieval as our initial test bed for the study of neural generative retrieval models.

2.3.2 Heterogeneous Information and Product Search

In many IR tasks, documents may contain or be related to information from different structures and multiple sources. For example, a document could have text structured in multiple fields such as title and body (e.g., Web pages); in more complicated cases, a document could consist of information from different sources such as image, category, etc. (e.g. products on e-shopping websites). Integrating information in heterogeneous forms for effective retrieval has been an important research topic in IR.

Depending on the sources of data, studies on heterogeneous information retrieval can be broadly split into two directions: retrieving information from a single source with both structured and unstructured forms, i.e., the *semi-structured data*, and retrieving information from multiple sources. Semi-structured data are common today as most articles and web pages involve both unstructured text and structured fields such as titles, descriptions, etc. Despite that they are expressed with words sharing the same semantic meanings, information from different fields often have different effects on how people determine the relevance of a document with respect to a query. In fact, using evidence from document structures to improve search engines is well studied in the IR community [100, 79, 82, 110]. A variety of models has been proposed to tackle the problem of combining information from multiple fields for document retrieval. For instance, Wilkinson [100] introduced a number of hypotheses to combine section-level and document level information, including a weighted sum, a max-pooling process and a combination of section scores and document scores. There is also a number of studies that try to construct probabilistic retrieval models for semi-structured data [79, 74, 94]. Robertson et al. [82] extended the original BM25 model based on the bag-of-words assumption [86] and proposed a BM25F that combines the frequency information across fields to produce a balanced BM25 score. Ogilvie and Callan [77] proposed a couple of methods to combine document representations for a known-item search task under the language modeling framework. Kim et al. [54] developed a probabilistic model for XML retrieval, and Kim and Croft [53] later proposed to weight different document fields with a model based on relevance feedback.

Most existing work on semi-structured data simplifies the problem by focusing on text only. Part of the reason is that modeling information from multiple sources is challenging under a statistical probability framework. However, as web applications accumulate more and more information in heterogeneous forms such as text, image, and knowledge graphs, the problem of how to model heterogeneous information from different sources for retrieval tasks has become increasingly important. Recently, some studies were conducted on cross-media retrieval, a special case of heterogeneous information retrieval with a focus on media data [102, 117, 107, 106]. They developed separate modules for each type of data and mapped them into a semantic space.

The retrieval of documents is conducted in a similar paradigm to vector space models [88] which ranks documents by their distances to the query representation. Jeon et al. [51] propose a Fixed Annotation-based Cross-media Relevance Model (FACMRM) for cross-media retrieval. In FACMRM, images are described with a vocabulary of blobs generated from image features using clustering. Retrieval is conducted with a probabilistic model that ranks images with the probability of generating a word given the blobs in an image. Despite their differences, these methods are constructed with human heuristics that require significant domain expertise to build, which makes them difficult to be extended for new information sources. Constructing a unified generative framework that is both effective and extendable for information retrieval is still an open question.

In this thesis, we use the task of *product search* to explore the potentials of neural generative models for search on heterogeneous information. Product search is a well-established retrieval task which focuses on retrieving relevant products for customers to satisfy their purchase intents. Previous studies on product search mainly focus on retrieving products based on their structured aspects such as brand, category, etc. For example, Lim et al. [60] propose a document profile model to suggest semantic tags for each item based on their structural aspects, so that we can retrieve products by matching queries with multiple product aspects simultaneously. Despite their success, searching with structured data cannot satisfy the need of e-shopping users today as their intents become more and more complicated. As shown by Duan et al. [34], writing structured search queries (e.g., SQL) is usually considered hard and inconvenient for search users. In most cases, queries submitted to product search engines are free-form text that is difficult to structure. However, there often exists a large vocabulary gap between the descriptions of products and user queries [97, 6]. Nurmi et al. [76] find that the words customers used to write their shopping lists are often different from those sellers used to describe their products. Because of these problems, IR researchers have proposed to construct semantic latent space for product search and conduct matching between queries and products with their latent representations. For instance, Yu et al. [108] construct a Latent Dirichlet Allocation model with product information and use it to diverse e-commerce search results. Van Gysel et al. [97] introduce a Latent Semantic Embedding model that maps and matches n-grams from queries and product descriptions into a hidden space. Guo et al. [43] propose a TranSearch model that can directly match text queries with product images. Also, there is a variety of studies [8, 52, 48] on extracting different text and product features and feeding them into learning-to-rank models for the optimization of different product retrieval objectives. Wu et al. [101] manually extract multiple statistic features from product search logs and construct an ensemble tree model to predict user clicks. Wu et al. [103] developed a special ranking loss function that optimizes product search engines by maximizing the revenue generated from online transactions. While they are effective for product search on unstructured free text, it is difficult to extend these methods with structured metadata and their relationships. As discussed in Chapter 4 and 5, we propose to combine neural embedding techniques with generative models for product search and develop a generic representation learning framework for heterogeneous information retrieval.

2.4 Neural Models for Information Retrieval

Recently, inspired by the advances in deep learning techniques, neural models have received considerable attention in the IR community. They have been proven to be effective at extracting relevance signals directly from low-level data representations such as raw text, and have good performance in a variety of IR tasks such as ad-hoc retrieval [3], question answering [105], and recommendation [45]. Based on their optimization goals, existing neural models for IR can be broadly categorized as discriminative models and generative models.

2.4.1 Discriminative Neural Retrieval Models

The basic idea of discriminative neural models for IR is to treat retrieval tasks as a classification or regression problem in which the goal is to find the most relevant document in the candidate sets. Thus, the loss functions for parameter optimization in discriminative retrieval models are often designed based on the labels of each document or the comparison of two or more documents (e.g. hinge loss [93]). Also, the structure of their networks usually focuses on modeling the match between queries and documents. For example, Huang et al. [49] proposed to represent documents with trigrams and introduced a Deep Structured Semantic Model that constructs a deep neural network for short text retrieval. The trigram representations of documents are directly fed into a multi-layer network and the loss function is computed using a softmax function over the network outputs of one relevant document and multiple negative samples. Guo et al. [41] conducted a theoretical analysis of IR tasks and proposed a Deep Relevance Matching Model that explicitly models the interaction patterns between queries and relevant documents. They notice that both the exact matching and semantic matching of words in queries and documents are important for understanding their relevance, and combine the two matching signals together to create a neural matching model for ad-hoc retrieval. Later on, many papers [90, 30, 21, 104] have been published in this research direction following similar design paradigms. Also, there are many studies on applying deep learning technology on other complicated IR problems such as user modeling [64, 20] and context-aware relevance modeling [1, 10].

Although these discriminative neural models produce state-of-the-art performances in many retrieval applications, they are completely data-driven and it is difficult for a human to understand what exactly they do to evaluate the relevance of documents. Thus, another line of studies on neural retrieval models focus on directly modeling the generation of relevant information, which we refer to as the generative neural retrieval models.

2.4.2 Generative Neural Retrieval Models

Neural generative models, especially neural language models, have been extensively studied in Natural Language Processing (NLP). One of the initial works in this field is the neural probabilistic language model proposed by Bengio et al. [14]. Similar to the DNN model presented in Figure 1.2, the neural probabilistic language model takes the bag-of-words representations of context words as the input of a multi-layer neural network to predict the next word in the text. More recently, Mikolov et al. [72, 71] introduced an efficient embedding model (word2vec) that can acquire semantically meaningful distributed representations for words. Following a similar training paradigm, it learns word embeddings in a generative framework where words are generated from their context. Later, Le and Mikolov [58] proposed the paragraph vector models that learn distributed representations for documents through constructing neural language models. Previous studies have shown that the paragraph vector models can significantly outperform classic topic modeling approaches such as pLSI [47] and LDA [16] in word analogy tasks and document clustering. Despite the great impact of neural generative models in the NLP community, there were few studies that successfully apply neural generative models for IR tasks in the early 2010s. Most neural retrieval models in the early days simply use the similarities between the distributed representations of words and documents learned with unsupervised learning to estimate their relevance. Without proper adaptations, these models tend to produce sub-optimal retrieval performance in practice.

In this dissertation, we explore the potential of neural generative models for IR. Specifically, we study the theoretical foundation of several neural models and propose a generic generative framework that can learn distributed information representations through optimizing the retrieval performance of IR systems in specific tasks such as ad-hoc document retrieval and product search.

CHAPTER 3

ADAPTING NEURAL EMBEDDING MODELS FOR AD-HOC RETRIEVAL

3.1 Introduction

As discussed in Chapter 1, most tasks in information retrieval (IR) benefit from representations that do not treat individual words and documents as unique symbols but reflect their semantic relationships. A common paradigm is to project both words and documents to a latent semantic space and perform matching or language estimation accordingly. This has led to a range of research that incorporates topic models into ad-hoc retrieval tasks. For example, the cluster-based retrieval model [62] and the LDA-based retrieval model [99] have been used to smooth the probability estimation in language modeling approaches with a cluster-based topic model and a Latent Dirichlet Allocation model, respectively. Both methods obtained consistent improvement over the original language models [80].

Recent advances in neural embedding models potentially provide new methods to acquire semantically meaningful representations for words and documents. In particular, Le et al. [58] propose a paragraph vector (PV) model that can jointly learn word and document embeddings through estimating a document level language model. In contrast to topic models, PV does not define a fixed number of topics as a priori. Documents and words are flexibly clustered through the learning of embedding vectors. Meanwhile, PV can be trained with stochastic gradient decent algorithm (SGD), which is simple yet efficient for large-scale learning problems. Previous studies showed that PV has superior performance on several linguistic tasks [28] and great potential for IR [58].

Since PV estimates a document language model, a natural idea is to incorporate it into the language model framework for IR tasks. However, according to our initial experiments, directly combining the original PV with language modeling approaches produces unstable performance and limited improvement.

In this chapter, we conduct both a theoretic and empirical analysis of PV-DBOW to define its limitation as a language model for IR. Specifically, we notice three problems when incorporating the original PV-DBOW into language modeling approaches. First, the unregulated learning objective makes PV-DBOW vulnerable to over-fitting. This version of the model tends to retrieve more short documents as training iterations increase. Second, the corpus-frequency based negative sampling strategy of PV-DBOW leads to a ICF-like weighting scheme for words in documents, which overly suppresses frequent words. Third, PV-DBOW does not capture word-context information, which makes it unable to model word substitution. By not capturing the substitution relations between words, PV-DBOW produces sub-optimal vectors for words and documents which leads to inferior language estimation. In addition to the detailed analysis of these problems, we also provide clear explanations of how they are addressed by L2 regularization, document-frequency based negative sampling, and a joint learning objective. Results on TREC collections indicate that the proposed modifications improve both the effectiveness and robustness of PV-based retrieval models.

The rest of the chapter is structured as follows. Section 3.2 introduces the basic structure of PV-based retrieval models. Section 3.3 presents the analysis of the problems and modifications for PV-based retrieval models. The proposed modifications are validated with experiments in Section 3.4. Finally, we conclude this chapter in Section 3.5.

3.2 Paragraph Vector Model for IR

In this section, we describe the details of how to apply the original PV model for information retrieval. We focus on a specific type of PV model with distributed bag-of-words assumption (PV-DBOW) due to its direct connection with language models of documents.

3.2.1 PV-DBOW

The original PV-DBOW was proposed by Le et al. [58]. The concept of "paragraph" stands for texts with varied lengths, which can be sentences, paragraphs and, in our case, the whole documents. PV-DBOW assumes the independence between words in a document and uses the document to predict each observed word in it. In this way, PV-DBOW learns both document and word embeddings by estimating a document level language model. Specifically, each document d is first projected into a semantic space and then trained to predict its words w. With the bag-of-words assumption, the generative probability of word w in document d is obtained through a softmax function over vocabulary V_w:

$$P(w|d) = \frac{exp(\vec{w} \cdot \vec{d})}{\sum_{w' \in V_w} exp(\vec{w'} \cdot \vec{d})} \tag{3.1}$$

where $P(w|d)$ denotes the probability of word w given document d, $\vec{w}$ and $\vec{d}$ denote the vector representations for w and d.

To reduce the cost of gradient computation for Equation (3.1) given a large vocabulary, Mikolov et al. [72] proposed a ***negative sampling*** strategy. The idea of negative sampling is to randomly sample several words from the corpus according to a predefined noise distribution, and use these words to approximate the denominator of Equation (3.1). With negative sampling, the global objective of PV-DBOW that sums over all possible word-document pairs is:

$$\begin{aligned}\ell = &\sum_{w \in V_w} \sum_{d \in V_d} \#(w,d) \log(\sigma(\vec{w} \cdot \vec{d})) \\ &+ \sum_{w \in V_w} \sum_{d \in V_d} \#(w,d)(k \cdot E_{w_N \sim P_V}[\log \sigma(-\vec{w_N} \cdot \vec{d})])\end{aligned} \tag{3.2}$$

where $\#(w,d)$ is the frequency of observed word-document pairs, V_d represents the corpus of documents, k is the number of negative samples, $\sigma(x)$ is the sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$ and $E_{w_N \sim P_V}[\log \sigma(-\vec{w_N} \cdot \vec{d})]$ is the expected value of $\log \sigma(-\vec{w_N} \cdot \vec{d})$ given the noise distribution P_V.

The embedding process of PV-DBOW captures high level semantic information and conveys two major advantages over traditional topic models such as LDA. First, PV-DBOW does not have a fixed number of topics. Documents and words are automatically clustered through the training process without any prior assumptions. Second, PV-DBOW can be efficiently trained through SGD, which is more scalable to a large corpus than traditional probabilistic topic models. According to our experience, training PV-DBOW on a million documents is ten times faster than training LDA with Gibbs sampling on the same collection.

3.2.2 PV-based Retrieval Model

For each document, PV-DBOW builds a language model that directly estimates the probability of word given a certain document ($P(w|d)$). Therefore, a natural way to use PV-DBOW in the IR scenario is to combine its language estimation with traditional language modeling approaches. Inspired by the idea of the LDA-based retrieval model [99], we use PV-DBOW for language model smoothing in the query likelihood model (QL). Suppose that the word probability estimated with QL (with Dirichlet smoothing) and PV-DBOW are $P_{QL}(w|d)$ and $P_{PV}(w|d)$, the final word probability $P(w|d)$ is obtained through Jelinek-Mercer smoothing:

$$P(w|d) = (1-\lambda)P_{QL}(w|d) + \lambda P_{PV}(w|d) \tag{3.3}$$

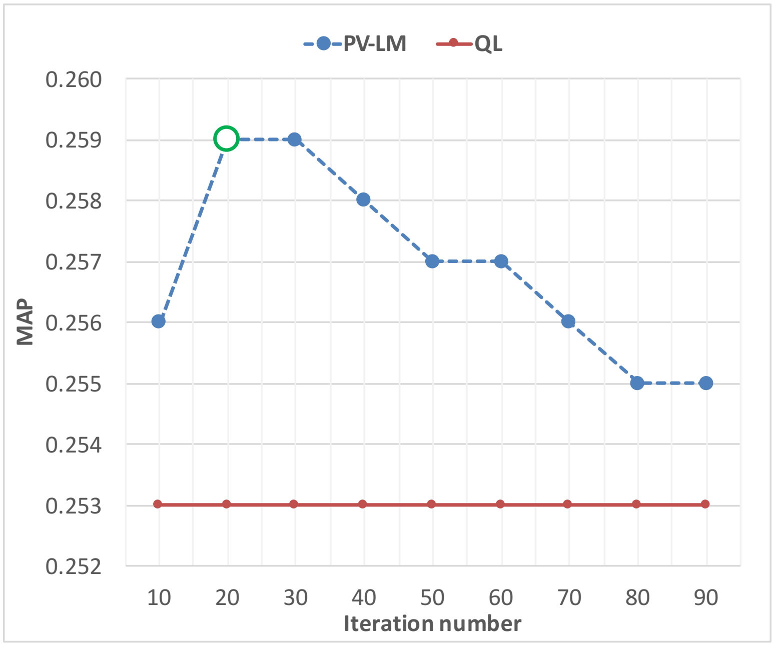


Figure 3.1: The MAP of QL and the PV-based retrieval model with the original PV-DBOW on Robust04 with title queries in respect of different training iteration.

where λ is the parameter that controls smoothing strength. In our experiments, we tried other smoothing methods such as Dirichlet smoothing, but we observed no significant difference between them in retrieval performance.

3.2.3 Stability of the Model

Our initial experiment show that the PV-based retrieval model indeed outperforms QL model, but its improvement is unstable throughout the training process. On Robust04, we trained PV-DBOW with 300 dimensions and evaluated QL and the PV-based retrieval model with a 5-fold cross validation on title queries (detailed settings are described in Section 3.4.2 and 3.4.3). The best mean average precision (MAP) of the PV-based retrieval model with the original PV-DBOW is 0.259, while that for QL model is 0.253. The difference between the PV-based retrieval model and QL is significant ($p < 0.05$), which demonstrates the effectiveness of language smoothing with PV-DBOW. However, we noticed that the performance of PV-based

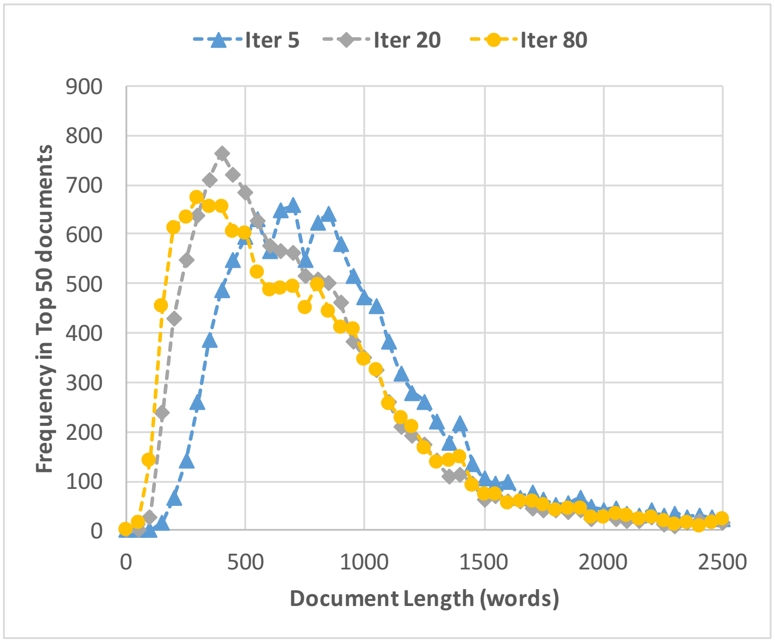


Figure 3.2: The distribution of documents in respect of document length for top 50 documents retrieved by PV-based retrieval model on Robust04 (title queries). Documents with more than 2500 words are ignored.

retrieval model is highly sensitive to the training iterations of PV-DBOW. As shown in Figure 3.1, the MAP of the PV-based retrieval model increases in the beginning, but starts to decrease after 20 iterations. The final performance in the 80 iterations is only slightly better than QL. In the worst cases, the performance improvement from PV-DBOW is inconsistent and marginal, which motivates us to further analyze the limitations of PV-DBOW in language estimation.

3.3 Problems and Modifications

In this section, we conduct an analysis of the reasons for the unstable performance and marginal improvements of the original PV-based retrieval model. Based on this analysis, we talk about the corresponding modifications and show how these modifications affect the language estimation of the PV model.

3.3.1 Over-fitting on Short Documents

As shown in Section 3.2.3, one interesting phenomenon is that the performance of the PV-based retrieval model does not converge along with the training iterations. To analyze the possible reasons, we conducted experiments over the top retrieved results of the PV models. Figure 3.2 shows the distribution of documents with respect to document length in the top 50 documents retrieved by the PV-based retrieval model on Robust04 with title queries. We equally split the domain of document length (0 to 2500) into 50 bins and ignore documents longer than 2500 words (which accounts for less than 4% of the top 50 documents). To avoid confusion, all the models depicted in Figure 3.2 only use the probability produced by PV-DBOW in language estimation (namely $\lambda = 1$ in Equation (3.3)). As shown in Figure 3.2, the distribution of documents with respect to document length gradually moves to the left as training iterations increase for PV-DBOW. The median document length for PV-DBOW is 750-800 under 5 iterations, 600-650 under 20 iterations, and 550-600 under 80 iterations. The results indicate that the training process of PV-DBOW introduces increasingly stronger bias toward short documents in the final retrieval model.

To understand the fundamental reason for this length bias, let us look back at the learning process of the PV-DBOW model. As shown in Equation (3.1), the prediction task of PV-DBOW requires the model to assign higher probability to words that occur in a document than others. In other words, the model will try to align the document vector to the word vectors that appear in the document. This alignment is much easier for short documents since on average the word vectors in short documents would be more concentrated than that in long documents. In practice, concentrated word vectors lead to concentrated gradient directions for document vectors. The partial derivative of the global objective with respect to a certain document d is computed as follows:

$$\begin{aligned}\frac{\partial \ell}{\partial d} = &\sum_{w \in V_w} \#(w,d) \log(\sigma(-\vec{w} \cdot \vec{d}))\vec{w} \\ &- \sum_{w \in V_w} \#(w,d)(k \cdot E_{w_N \sim P_V}[\log \sigma(\vec{w_N} \cdot \vec{d})]\vec{w_N})\end{aligned} \tag{3.4}$$

Despite the part with w_N (which is randomly sampled according to a global noise distribution), we can see that the gradient of d is a weighted sum of its word vectors. Because short documents have fewer words, their gradients could easily converge to a direction that is not far from all the word vectors. This would result in more rapid increase of norms for short document vectors. Therefore, given an observed word, the probability produced by short documents will become higher and higher, leading to a potential over-fitting.

To verify this, we further plot the variation of the learned document vectors with respect to the document length under different learning iterations. Figure 3.3 shows the distribution of vector norms for 10,000 documents randomly sampled from Robust04. For documents with more than 1,000 words, vector norms in PV-DBOW with 5, 20 and 80 iterations show no significant difference. However, for documents with fewer than 1,000 words, the norms of document vectors increases rapidly as iteration number increases.

This analysis shows that the original PV-DBOW suffers from the over-fitting problem along with the training process, and this over-fitting problem is more severe for short documents. A direct method to solve the over fitting problems is to regularize the learning objective of PV-DBOW. Because the over-fitting problem is mainly caused by the unrestricted document vectors, we add an L2 regularizer over the document vectors. More formally, the local objective function for each (w, d) pair with regularization is now as follows:

$$\ell'(w,d) = \ell(w,d) - \frac{\gamma}{|d|}||\vec{d}||^2 \tag{3.5}$$

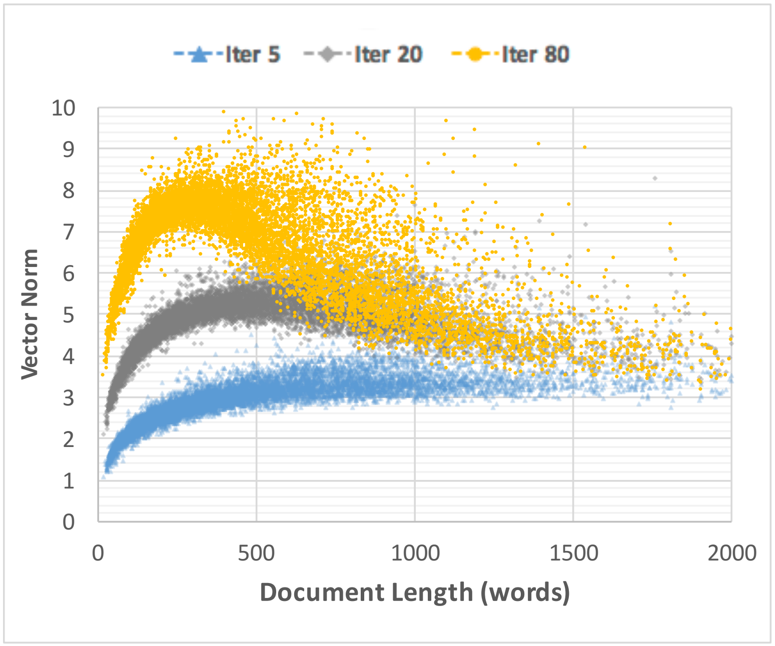


Figure 3.3: The distribution of vector norms in respect of document length for 10,000 documents randomly sampled from Robust04.

where $\ell(w, d)$ represents the local objective function for PV-DBOW, $||\vec{d}||$ denotes the norm of vector $\vec{d}$ and γ denotes a hyper-parameter that control the strength of regularization. Because each iteration of PV-DBOW goes through each word once, a length factor $\frac{1}{|d|}$ where $|d|$ denotes the number of words in document d (namely the length of d) is used to guarantee the same regularization term for all the documents in the training corpus.

The effect of L2 regularization on the language model of PV-DBOW is twofold. First, with L2 regularization, the vector norms for both short documents and long documents are roughly the same along with training iterations. Severe over-fitting on short documents no longer exists in long term training. Second, the restriction on vector norms makes the probability distribution in Equation (3.1) smoother, which potentially benefits the language smoothing of PV-based retrieval models.

3.3.2 Improper Noise Distribution

To analyze the reason for the limited performance improvement of the PV-based retrieval model, we first look at the learning objective of PV-DBOW. Inspired by the analysis of the skip-gram model in Levy et al. [59], we derive the local objective for a specific word-document pair from Equation (3.2) as:

$$\ell(w,d) = \#(w,d) \log \sigma(\vec{w} \cdot \vec{d}) + k\#(d)P_V(w) \log \sigma(-\vec{w} \cdot \vec{d}) \tag{3.6}$$

where $\#(d)$ represents the length of d. Define $x = \vec{w} \cdot \vec{d}$, then the objective's partial derivative on x would be:

$$\frac{\partial \ell(w,d)}{\partial x} = \#(w,d) \cdot \sigma(-x) - k \cdot \#(d) \cdot P_V(w) \cdot \sigma(x) \tag{3.7}$$

Let the partial derivative equal to zero, then the only valid solution for Equation (3.7) is

$$\vec{w} \cdot \vec{d} = \log(\frac{\#(w,d)}{\#(d)} \cdot \frac{1}{P_V(w)}) - \log k \tag{3.8}$$

We can see that the original PV-DBOW model conducts implicit factorization over the term-document co-occurrence matrix. The noise distribution of negative sampling actually decides how we weight the terms in a document. The original negative sampling [72] adopts empirical word distribution in the whole corpus as the noise distribution P_V, which is defined as:

$$P_V(w_N) = \frac{\#w_N}{|C|} \tag{3.9}$$

where $\#(w_N)$ is the corpus frequency of w_N and $|C|$ is the size of the corpus. In Equation (3.8), $\frac{\#(w,d)}{\#(d)}$ is the normalized TF of w in d, and $\frac{1}{P_V(w)}$ (namely $\frac{|C|}{\#w}$) is the ICF value of w. Therefore, the original PV-DBOW with negative sampling is optimizing for a variation of TF-ICF weighting scheme.

However, TF-ICF is not a popular weighting scheme in IR. One direct reason is that ICF-based term weighting computes the discriminative ability of words only according to their frequency in the corpus and does not consider any form of document structure information. Empirically, a word with high corpus frequency could still be discriminative if it only appears in a small group of documents. This partially explains why PV-DBOW performs well on NLP tasks but not on IR tasks.

Based on these above ideas, one approach to address the problem of PV-DBOW is applying a document-frequency (DF) based negative sampling strategy. More formally, we replace P_V in the original negative sampling with a new noise distribution P_D as follows:

$$P_D(w_N) = \frac{\#D(w_N)}{|N|} \tag{3.10}$$

where $\#D(w_N)$ denotes the document frequency of w_N and $|N| = \sum_{w' \in V_w} \#D(w')$. After substituting P_V with P_D in Equation (3.8), we get the new optimal solution as

$$\vec{w} \cdot \vec{d} = \log(\frac{\#(w,d)}{\#(d)} \cdot \frac{|N|}{\#D(w)}) - \log(k) \tag{3.11}$$

Because $\frac{|N|}{\#D(w)}$ is a variant of the inverse document frequency (IDF) of w, PV-DBOW with DF-based negative sampling is factorizing a shifted matrix of TF-IDF, which is usually considered to be a better scheme for term weighting than TF-ICF [81].

We further plot both the corpus-frequency and document-frequency based distributions in Figure 3.4 (P_V and P_D respectively). Similar to Church and Gale [24], we observe considerable difference between these sampling distributions, especially on frequent words. As we can see from Figure 3.4, P_V grows in an exponential way and assigns much higher sample probability to frequent words compared to P_D, which

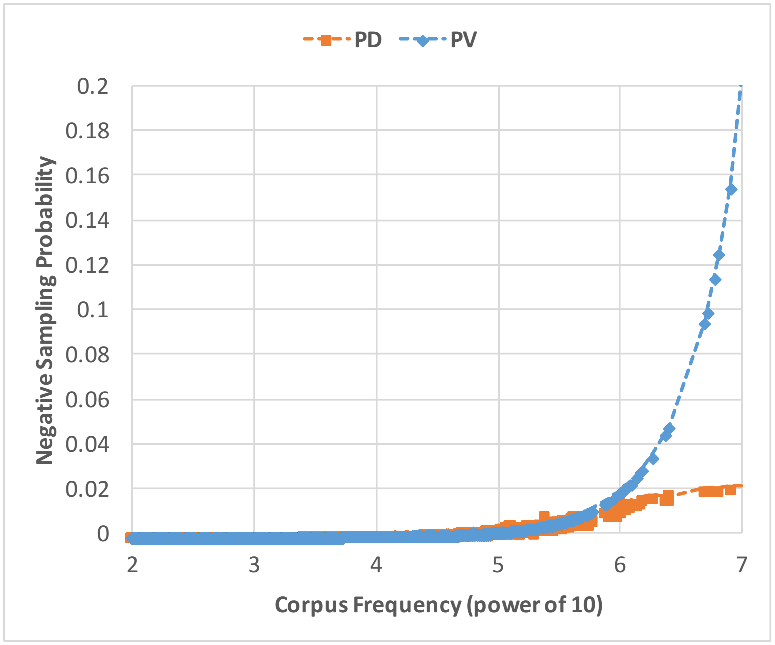


Figure 3.4: The distribution of the original negative sampling (P_V) and the document-frequency based negative sampling (P_D). The horizontal axis represents log value of word frequency with log base 10.

may over-penalize frequent words in the learning of language model. For example, in Robust04 query 339 ("alzheimers drug treatment"), the probability estimated by PV-DBOW with corpus-frequency based negative sampling for "alzheimers" (0.042) is higher than "drug" (0.002) in document FT933-3956, even when "drug" appears two times more than "alzheimers". This suppression makes "drug" less important for the final ranking and consequentially hurts the performance of this query. With document-frequency based negative sampling, the term weighting is moderated and produces more reasonable language estimation (0.056 for "alzheimers" and 0.069 for "drugs" in FT933-3956).

In practice, negative sampling with a very skew distribution is suboptimal for the approximation of softmax function in the learning objective of PV-DBOW. This is the reason why Mikolov et al. [72] applied a unigram distribution raised to the

Table 3.1: The cosine similarities between *clothing*, *garment* and four relevant documents in Robust04 query 361 ("clothing sweatshops"). EPV-DR represents the PV-based retrieval model with document-frequency based negative sampling and L2 regularization. EPV-DRJ is EPV-DR with a joint objective.

	EPV-DR		EPV-DRJ	
	clothing	*garment*	*clothing*	*garment*
clothing	1.000	0.632	1.000	0.638
LA112689-0194 $TF_{clothing}$=2, $TF_{garment}$=26	0.044	0.134	0.107	0.169
LA112889-0108 $TF_{clothing}$=0, $TF_{garment}$=10	-0.003	0.100	0.126	0.155
LA021090-0137 $TF_{clothing}$=7, $TF_{garment}$=9	0.052	0.092	0.147	0.119
LA022890-0105 $TF_{clothing}$=6, $TF_{garment}$=6	0.066	0.079	0.107	0.107

power of 0.75. Similarly, we adopt a power version of document frequency that uses $\#D(w)^{\eta}(0 \leq \eta \leq 1)$ to replace $\#D(w)$ in Equation (3.10).

3.3.3 Insufficient Modeling for Word Substitution

From the analysis in the prior section, we find that the optimal solution of PV-DBOW's objective function (Equation (3.6)) is actually an implicit factorization over the term-document matrix. As shown in [92], models that leverage distributed information over the term-document matrix mainly capture words' syntagmatic relations but ignore paradigmatic relations. Syntagmatic relations relate words that co-occur in the same text region. For example, "NBA" is related to "basketball" because they often co-occur in same documents. Paradigmatic relations, namely substitution relations, relate words that often share similar context but may not co-occur in documents. For example, "subway" and "underground" are synonyms and often occur in similar contexts, but American people usually use "subway" while British people tend to use "underground". The original PV-DBOW aligns word vectors to document vectors so that words with high co-occurrence tend to have similar representations. However, it cannot model the semantic similarity between words that occur with similar context but not in the same document.

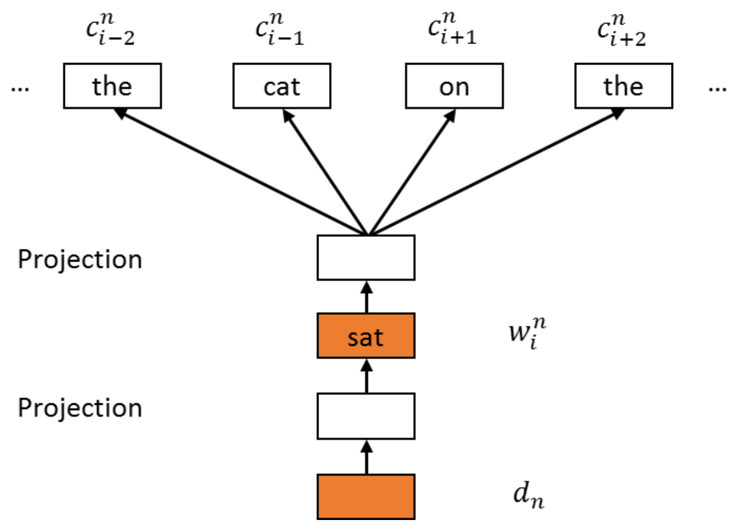


Figure 3.5: The structure of two-layer PV-DBOW. The document is trained to predict the observed word and then the observed word is trained to predict its context.

Word paradigmatic information, or word substitution relation is important for IR because it directly alleviates the problem of term mismatch. Term mismatch is common in IR tasks because a query term mismatches 40% to 50% of relevant documents on average [116]. A language model that cannot capture word substitution relation would be vulnerable to the mismatch problem and have limited smoothing ability. Here we take Robust04 query 361 ("clothing sweatshops") as an example. In this query, "garment" is frequent in relevant documents while "clothing" is not. Table 3.1 lists the cosine similarities between "clothing", "garment" and four relevant documents in the enhanced PV-based retrieval model with document-frequency based negative sampling and L2 regularization (EPV-DR). Intuitively, "clothing" should receive a similar probability to "garment" because they are synonyms. However, EPV-DR assigns much lower cosine similarities for "clothing" than "garment", which consequentially decreases the probability of "clothing" in these relevant documents and lowers their final ranks.

To model word substitution relations, we apply a joint learning objective for PV-DBOW as suggested in [28, 92]. As shown in Figure 3.5, the first layer of the model uses the document vector to predict the observed word. Then, the second layer of the model uses the observed word to predict its context. More formally, the local objective of the PV-DBOW with the joint objective function can be expressed as

$$
\begin{aligned}
\ell = & \log(\sigma(\vec{w_i} \cdot \vec{d})) + k \cdot E_{w_N \sim P_D}[\log \sigma(-\vec{w_N} \cdot \vec{d})] \\
& + \sum_{\substack{j=i-L \\ j \neq i}}^{i+L} \log(\sigma(\vec{w_i} \cdot \vec{c_j})) + k \cdot E_{c_N \sim P_D}[\log \sigma(-\vec{w_i} \cdot \vec{c_N})]
\end{aligned}
\tag{3.12}
$$

where $\vec{c_j}$ is the context vector for word w_j, c_N denotes the sampled context and L represents the context window size.

From a learning perspective, adding the prediction objective between words and context actually regularizes the learning objective of PV-DBOW. This regularization usually results in better representations for words and documents according to previous studies [28, 92]. In Table 3.1, after incorporating EPV-DR with the joint objective (EPV-DRJ), the cos similarities between "clothing" and those four relevant documents increase considerably. Even LA112889-0108 (the document in which "clothing" never appears) now has similar cosine similarities for "clothing" and "garment". Therefore, the language estimation of EPV-DRJ based retrieval model gives higher probabilities for "clothing" in those documents and increases the final retrieval performance.

3.4 Experiments

In this section, we conduct empirical experiments to verify the effectiveness of different modifications on PV-DBOW for IR.

3.4.1 Data Set and Baselines

Two TREC collections (Robust04 and GOV2) have been used to evaluate the retrieval performance of PV-based retrieval models and proposed modifications. The

Table 3.2: Statistics of experimental data sets.

Collection	#Docs	#Words	Size	TREC Topics
Robust-04	528K	253M	1.9G	351-450, 601-700
Gov2	25,205K	24,007M	426G	701-850

statistics of Robust04 and GOV2 are provided in Table 3.2. We use the Galago search engine[1] to index the corpus and stemmed terms with the Krovetz stemmer [56]. Stop words in queries are removed in advance as suggested in [50]. To better understand the effectiveness of paragraph vector models in information retrieval, we include results from two baselines, i.e. the query likelihood model [80] and the LDA-based retrieval model [99].

Query likelihood (QL) [80] is a basic language modeling approach for information retrieval. It constructs document models with bag-of-words representation and ranks documents according to the log likelihood of query words given the document models. Standard query likelihood model with Dirichlet smoothing [111] can be formulated as Equation (3.13):

$$P_{QL}(Q|D) = \sum_{w \in Q} tf_{w,Q} log \frac{tf_{w,D} + \mu P(w|C)}{|D| + \mu} \tag{3.13}$$

where $tf_{w,Q}$ is the number of times that w occurs in the query, $tf_{w,D}$ is the number of times that w occurs in the document, $|D|$ is the length of the document, μ is a parameter for Dirichlet smoothing and $P(w|C)$ is a background language model that is computed as the number of w in the whole corpus divided by the corpus size. To simplify the parameter tuning for both baselines and PV-based retrieval models, we do not tune μ in our experiments and use the average value of the 5-fold validation on Robust04 and GOV2 from Huston and Croft [50]. Specifically, for Robust04 collection, we set $\mu = 934$ for title queries and $\mu = 2166$ for description

[1]`http://www.lemurproject.org/galago.php`

queries. For GOV2 collection, we set $\mu = 1481$ for title queries and $\mu = 2107$ for description queries.

LDA-based retrieval model (LDA-LM) [16]: LDA is a popular topic model based on a formal generative model of documents. It draws the document-topic distribution $\hat{\theta}$ and topic-word distribution $\hat{\phi}$ from two conjugate Dirichlet priors and models the posterior estimation of word w in document d as:

$$P_{Lda}(w|d) = \sum_{z=1}^{K} P(w|z,\hat{\phi})P(z|d,\hat{\theta}) \tag{3.14}$$

where K is the number of topics in LDA model. Proposed by Wei and Croft [99], LDA-based retrieval models combines the original document model from QL with LDA model as:

$$P(w|d) \quad = \quad (1-\lambda)P_{QL}(w|d) + \lambda P_{Lda}(w|d) \tag{3.15}$$

where $P_{QL}(w|d)$ is the maximum likelihood estimation of word w in document d with the query likelihood model and $P_{lda}(w|d)$ is the posterior estimation of w given d in the LDA model. In experiments, we use Gibbs sampling to estimate the parameters of LDA and empirically set topic number as $K = 800$. Following previous study [40], the symmetric Dirichlet priors in LDA are set as $\alpha = \frac{50}{K}$ and $\beta = 0.01$.

3.4.2 Evaluation Framework

We employ four standard retrieval metrics for evaluation: mean average precision (MAP), normalized discounted cumulative gain at 20 (nDCG@20) and precision at 20 (P@20). Due to the limited number of annotated queries in our experiment collections, we conduct 5-fold cross-validation. We follow the same settings as Huston and Croft[50] and split the query topics for each collections randomly into 5 folds. We tune λ (the combination weight for the LDA-based retrieval model and PV-based retrieval models) with 4 of the 5 folds and test on the remaining 1 fold. The reported numbers are the average value over all test folds. As suggested by Smucker et al.[91], statistical significance is computed with Fisher randomization test with threshold 0.05.

For efficient computation, we adopt a re-ranking strategy. The initial retrieval is performed with query likelihood model to obtain 2,000 candidate documents. Then re-ranking is performed with different models. The final evaluation is carried out on the top 1,000 results.

We trained LDA and paragraph vector models with documents in Robust04 and GOV2 separately. However, handling large scale dataset like GOV2 is computational expensive for LDA. For fair comparison, we randomly sampled 500k documents (including the candidates retrieved by QL) from GOV2 and trained LDA and paragraph vector models on the sampled subset.

3.4.3 Settings for Paragraph Vector Models

We tested four types of PV-based retrieval models:

- PV-LM: the PV-based retrieval model with PV-DBOW proposed by Le et al. [58].
- EPV-R-LM: the PV-LM model with L2 regularization.
- EPV-DR-LM: the EPV-R-LM model with document-frequency based negative sampling.
- EPV-DRJ-LM: the EPV-DR-LM model with a joint learning objective.

The tuning of all hyper-parameters in PV-DBOW requires considerable effort and is not the core of this paper, so we set most parameters same with the default settings

Table 3.3: Results on Robust04 collection measured by mean average precision (MAP), normalized discounted cumulative gains at 20 (NDCG@20), and precision at 20 (P@20). $*$, $+$ means significant difference over QL, LDA-LM respectively at 0.05 significance level measured by Fisher randomization test.

	Robust04 collection					
	Topic titles			Topic descriptions		
Method	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
QL	0.253	0.415	0.369	0.246	0.391	0.334
LDA-LM	0.258^{*}	0.421	0.374^{*}	0.247	0.392	0.336
PV-LM	0.259^{*}	0.418	0.371	0.247	0.392	0.335
EPV-R-LM	0.260^{*}	0.417	0.371	0.251^{*}	0.397^{*}	0.340^{*}
EPV-DR-LM	0.262^{*}	0.418	0.368	0.252^{*+}	0.397^{*}	0.338^{*}
EPV-DRJ-LM	$\mathbf{0.267}^{*+}$	$\mathbf{0.425}^{*}$	$\mathbf{0.376}^{*}$	$\mathbf{0.253}^{*+}$	$\mathbf{0.404}^{*+}$	$\mathbf{0.347}^{*+}$

from skip-gram word embedding model proposed in [72][2] except for iteration number. The iteration number is tuned offline with PV-LM from 10 to 80 (10 per step) on Robust04 titles. We observed the best performance under 20 iterations and fix this number for all PV-based retrieval models.

Modification-specific hyper-parameters are tuned separately for EPV-R-LM, EPV-DR-LM and EPV-DRJ-LM. For models with document-frequency based negative sampling, we tuned η from 0.0 to 1.0 (0.1 per step). The best performance for EPV-DR-LM and EPV-DRJ is 0.4 and 0.1. For models with L2 regularization, we tested γ from 0.1, 1, 10 and 100. The best performance is consistently obtained with 10 in EPV-R-LM, EPV-DR-LM and EPV-DRJ-LM.

3.4.4 Results and Discussion

3.4.4.1 Overall Performance

We refer the results on Robust04 in Table 3.3 and further extend the evaluation of baselines and PV-based retrieval models on GOV2 in Table 3.4. As observed by previous studies [99, 62, 4], topic level estimation is beneficial for language modeling

[2]https://code.google.com/p/word2vec/

Table 3.4: Comparison of different models over GOV2 collection measured by mean average precision (MAP), normalized discounted cumulative gains at 20 (NDCG@20), and precision at 20 (P@20). $*$, $+$ means significant difference over QL, LDA-LM respectively at 0.05 significance level measured by Fisher randomization test. The best performance is highlighted in boldface.

	GOV2 collection					
	Titles			Descriptions		
Method	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
QL	0.295^{+}	0.409	0.510^{+}	0.249^{+}	0.371	0.470
LDA-LM	0.290	0.406	0.505	0.245	**0.376**	0.468
PV-LM	0.294	0.409	0.510^{+}	0.246	0.364	0.463
EPV-R-LM	0.295^{+}	0.410	0.511^{+}	0.250^{+}	0.368	0.467
EPV-DR-LM	0.296^{+}	0.412	0.512	0.250^{+}	0.371	0.470
EPV-DRJ-LM	$\mathbf{0.297}^{+}$	$\mathbf{0.415}^{*+}$	$\mathbf{0.519}^{*+}$	$\mathbf{0.252}^{*+}$	0.371	**0.472**

approach. Both LDA-LM and PV-LM outperform QL on Robust04 titles and descriptions. The relative improvements in respect of MAP for LDA-LM are 2.4% on titles and 2.0% on descriptions; for PV-LM are 2.4% on titles and 0.4% on descriptions. The performance of LDA-LM and PV-LM show no significant difference. After adding L2 regularization, document-frequency based negative sampling and a joint objective, the performance of PV-LM increases and finally outperforms all baselines in both Robust04 and GOV2. On Robust04, the relative improvements of MAP for EPV-R-LM, EPV-DR-LM and EPV-DRJ-LM over PV-LM are 0.0%, 1.2% and 3.1% on titles, 0.0%, 2.0% and 2.4% on descriptions; on GOV2, the relative improvements of MAP for EPV-R-LM, EPV-DR-LM and EPV-DRJ-LM over PV-LM are 0.3%, 0.7% and 1.0% on titles, 1.6%, 1.6% and 2.4% on descriptions.

We notice that topic level smoothing tends to be more effective on short queries than long queries. Both LDA-LM and PV-based retrieval models achieve better improvement over QL in title queries than in description queries. For example, the best PV-based retrieval model, EPV-DRJ-LM, outperforms QL with 5.5% on Robust04 titles but 2.5% on Robust04 descriptions in respect of MAP. An explanation for this phenomenon is that vocabulary mismatch is more severe in short queries. With fewer words in short queries, the missing of one word could hurt the maximum likelihood estimation of QL. In contrast, long queries like descriptions usually have sufficient terms to express their query intents and are more robust to mismatch problems. The introduction of semantic matching to long queries could bring less benefit but more noise. We will give more examples in Section 3.4.4.4.

In our experiments, GOV2 receives less benefit from semantic smoothing (compared to Robust04). The incorporation of LDA even damages the performance of language modeling approach in most metrics. One potential reason is that GOV2 consists of web pages, which have a complex and noisy topic distribution compared to news articles in Robust04. Because our experiments restrict the number of topic in LDA to 800 (due to efficiency), the topics learned by LDA may be too vague and coarse for language estimation. In comparison, although the dimension of the vectors is 300, the number of topics in paragraph vector models is not limited. Because documents are automatically clustered without prior assumptions about topic distribution, PV-DBOW could capture finer semantic relations in a noisy environment. In our experiments, the EPV-DRJ-LM outperforms both QL and LDA-LM in most metrics.

3.4.4.2 Iteration Number

Our analysis shows that the number of training iterations in PV-DBOW have a considerable effect on the language estimation of PV-based retrieval models. To study the effect of training iterations, we depict the MAP value of PV-based retrieval models under different iteration numbers on Robust04 titles in Figure 3.6.

As shown in Figure 3.6, the over-fitting problem of PV-LM without L2 regularization is evident as iteration number increases. The best performance of PV-LM (0.259) is observed at 20 iterations, but it drops to 0.255 at 90 iterations. In contrast,

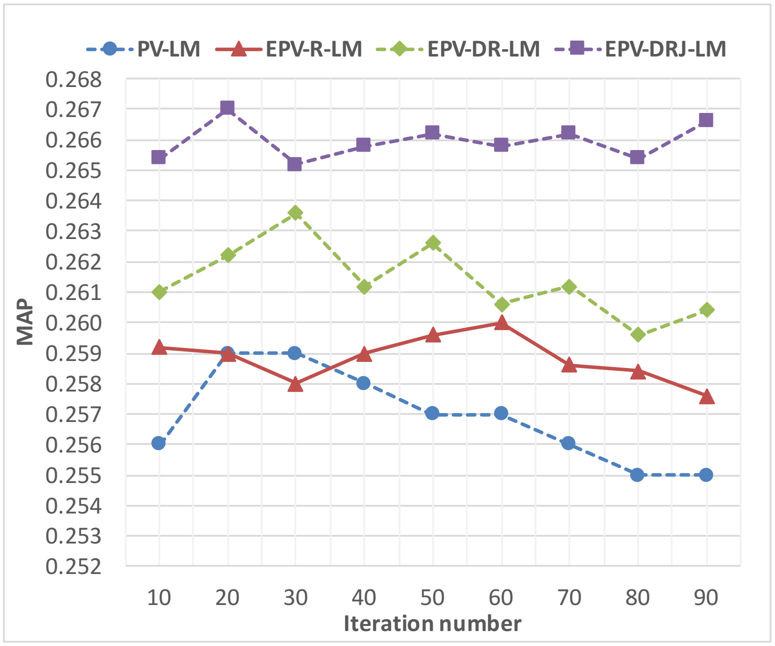


Figure 3.6: MAP variation of PV-based retrieval models with respect to iteration number. The horizontal axis represents the number of training iterations, and the vertical axis represents MAP on Robust04 title queries.

the results of PV-based retrieval models with L2 regularization (EPV-R-LM, EPV-DR-LM and EPV-DRJ-LM) are steady across different iteration numbers. The MAP of EPV-R-LM slightly wave around 0.259 and consistently outperforms PV-LM after 30 iterations.

Although the L2 regularization can effectively solve the over fitting problem of PV-based retrieval models, it does not significantly improve retrieval performance. By incorporating document-frequency based negative sampling strategy and the join objective, we observed improvement in the MAP scores on Robust04. These results indicate that those modifications together can significantly improve the robustness and effectiveness of PV-based retrieval models.

3.4.4.3 Vector Dimensionality

Previous studies find that higher dimensional vector representation can improve the performance of neural embedding models in NLP tasks [71]. To understand the

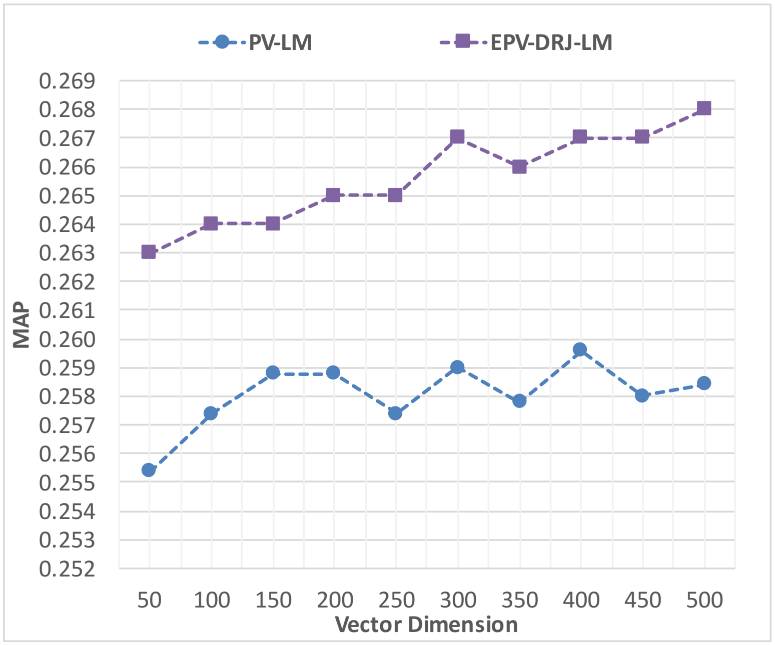


Figure 3.7: MAP variation of PV-based retrieval models with respect to vector dimensions. The horizontal axis represents vector dimensions, and the vertical axis represents MAP on Robust04 title queries.

effect of vector dimensionality, we test PV-based retrieval models with different vector sizes on Robust04 titles and show the results in Figure 3.7.

In Figure 3.7, the vector size in PV-DBOW shows a minor correlation with the performance of PV-based retrieval models. Although the MAP value of EPV-DRJ-LM increases slowly from 0.263 to 0.268 when vector dimensionality changes from 50 to 500, the performance of PV-LM fluctuates between 0.256 and 0.259. The improvement caused by increasing vector dimensionality is not consistent in different PV-based retrieval models. Zuccon et al. [118] find that vector dimensionality in word embedding has a minor effect on model performance in ad-hoc retrieval. Similarly, we notice that the setting of dimensionality for PV-based retrieval models is not as important as it is for LDA-LM [99] in language estimation. A potential explanation is that the dimensionality of document vectors is not explicitly linked with the topic number in paragraph vector models. Even with low-dimensional vectors, paragraph vector models can still model a complex topic structure. In our experiments, the EPV-DRJ-LM with 50 dimensions still outperforms the LDA-LM with 800 topics on Robust04 (MAP 0.263 v.s. 0.259).

3.4.4.4 Case Studies

To further illustrate how paragraph vector models work for information retrieval, we conduct case studies to show the advantages and disadvantages of PV-based retrieval models.

The advantages of PV-based retrieval models mostly come from its semantic matching process. We use Robust04 title query 317 (”unsolicited faxes”) as an example. In Robust04, only three documents have ”unsolicited” and ”faxes” simultaneously and two of them contain each word exactly once. QL failed in this case (MAP 0.186) because it cannot reasonably differentiate the relevance of documents that do not have ”unsolicited” or ”faxes”. By projecting documents into semantic concepts, paragraph vector models and LDA provide finer information for the query words and the mismatched documents. As a result, the MAP for EPV-DRJ-LM and LDA-LM in query 317 outperform QL by 75.3% (0.186 to 0.326) and 19.4% (0.186 to 0.222). The results show that both the PV-based and LDA-based retrieval models can improve retrieval performance by involving semantic matching information in language modeling approaches, while PV models can provide even better estimation than LDA model.

However, the semantic matching in PV-based retrieval models may sometimes not work well on long queries. One representative example in our experiments is Robust04 query 614: *flavr savr tomato* (the title), *find information about the first genetically modified food product to go on the market flavr savr also flavor saver tomato developed by calgene* (the description with stopwords removed). With the query title, EPV-DRJ-LM performs better than QL (MAP 0.522 v.s. 0.174) because most documents in Robust04 do not contain the exact matching of these query words (only four documents contain ”flavr” or ”savr”). However, the situation changes when replacing the query title with the query description. One reason is that the query description expands the query title with high quality words that can significantly boost the performance of exact matching model (such as “genetical”, “food” and “calgene”). In our experiments, the MAP value of QL is increased by 336.8% (0.174 to 0.76), but the gain for EPV-DRJ-LM is only 44.1% (from 0.522 to 0.752 in MAP). Generally, long queries describe query intents with sufficient information. In this case, semantic matching may bring less benefit but more noise to retrieval models.

3.5 Conclusion

In this chapter, we study PV-DBOW with both theoretic and empirical analysis to understand its limitation as a language model for IR. We discuss three problems that restrict the effectiveness of PV-DBOW in IR scenario: over-fitting on short documents, improper negative sampling strategy and the lack of word substitution modeling. To address these problems, three modifications for the original PV-DBOW have been proposed. We analyze how these modifications affect the language estimation in PV-DBOW and how they improve the performance of PV-based retrieval models. Experiments and case studies on standard TREC collections are presented to better illustrate and backup our analysis.

Although the discussions of this paper mainly focuses on PV-DBOW for IR, some results are also instructive for future work on other neural embedding models. First, the noise distribution of negative sampling can significantly affect the performance of PV-based retrieval models. With formal inductions, we show that different noise distributions lead PV-DBOW to optimize a different weighting scheme. In this way, one may easily adapt neural embedding models to incorporate different information for different tasks. Second, the norms of embedding vectors contain important information for IR. Previous work mainly focuses on the cosine similarities between embedding vectors, but our analysis show that the norms of embedding vectors also influence the language estimation of PV models. Vector norms in neural embedding models are related to both word frequency and document structures, which could be potentially useful for future studies.

CHAPTER 4

EMBEDDING-BASED NEURAL GENERATIVE MODEL FOR PERSONALIZED PRODUCT SEARCH

4.1 Introduction

In the previous chapter, we conduct theoretical analysis of a well-known neural generative model (i.e., the paragraph vector model) and propose several adaption techniques to improve its effectiveness in ad-hoc retrieval. Nonetheless, ad-hoc retrieval is a special search task that only concerns about homogeneous queries and documents, while IR problems in practice often involve heterogeneous information from multiple sources. For example, the description page of a product on Amazon.com usually consist of semi-structured data such as titles, descriptions, reviews, etc. How to conduct effective representation learning and information retrieval on such heterogeneous data is still an open question for the IR community. As a second step towards a generic neural representation learning framework for information retrieval, in this chapter, we explore the potential of neural generative models for a well-established IR task – personalized product search.

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

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

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 potential 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 [97]. 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.

In this chapter, we tackle the problem of personalized product search with neural generative models 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 in a non-search scenario [66, 65] or general product search without personalization [97]. To fill this gap, we propose a *Hierarchical Embedding Model* (HEM) specifically designed for personalized product search. Inspired by recent progress in distributed representation learning [58, 3], 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. [97], 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.

4.2 Hierarchical Embedding Model for 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.

4.2.1 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 (e.g., the intent of a query "digital camera" is to purchase a digital camera). The second is *user preference*, which decides whether an item satisfies the special need of a particular user (e.g., a user who has an iPhone may prefer accessories designed for Apple products). 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 [72, 71], 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:

$$\boldsymbol{M_{uq}} = \lambda \boldsymbol{q} + (1 - \lambda)\boldsymbol{u} \tag{4.1}$$

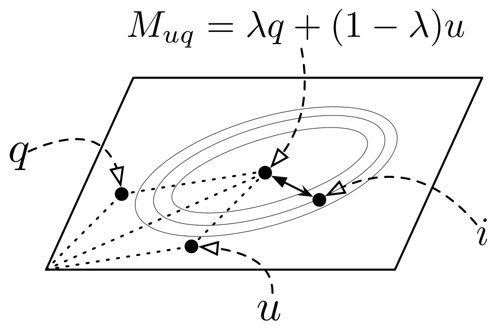


Figure 4.1: Personalized product search in a latent space with query q, user u, personalized search model M_{uq} and item i.

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:

$$Score(i|u,q) = f(\boldsymbol{i}, \boldsymbol{M_{uq}}) = f(\boldsymbol{i}, \lambda\boldsymbol{q} + (1-\lambda)\boldsymbol{u}) \tag{4.2}$$

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 4.1. 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.

4.2.2 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

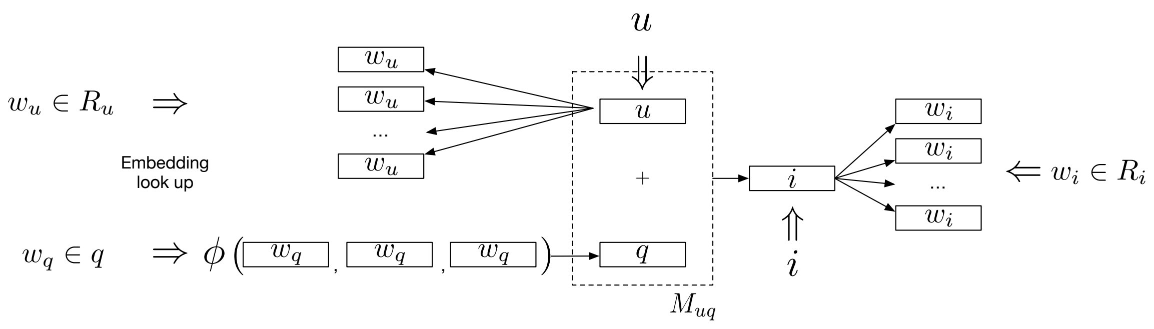


Figure 4.2: The structure of our hierarchical embedding model for personalized product search. R_u, R_i denote the sets of reviews for user u and item i; w_u, w_i, w_q denote the words in R_u, R_i and query q; and M_{uq} is the personalized search model constructed with query models and user models.

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 4.2. 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.

4.2.2.1 Embedding-based User/Item Language Model

Inspired by the paragraph vector models [58], 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:

$$P(w|e) = \frac{\exp(\boldsymbol{w} \cdot \boldsymbol{e})}{\sum_{w' \in V_w} \exp(\boldsymbol{w'} \cdot \boldsymbol{e})} \tag{4.3}$$

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 [17]). It can automatically cluster entities based on the characteristics of input data.

4.2.2.2 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:

$$\boldsymbol{q} = \phi(\{w_q | w_q \in q\}) \tag{4.4}$$

Previous studies have proposed several methods to combine word embeddings to form a query embedding. The simplest way is to aggregate and average the embeddings of query words directly [98], which can be formulated as:

$$\phi(\{w_q|w_q \in q\}) = \frac{\sum_{w_q \in q} \boldsymbol{w_q}}{|q|} \tag{4.5}$$

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 [97]:

$$\phi(\{w_q|w_q \in q\}) = \tanh(W \cdot \frac{\sum_{w_q \in q} \boldsymbol{w_q}}{|q|} + b) \tag{4.6}$$

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 [78]:

$$\begin{aligned}
\boldsymbol{o_t} &= (1 - \boldsymbol{a_t}) \odot \boldsymbol{o_{t-1}} + \boldsymbol{a_t} \odot \boldsymbol{s}_t \\
\boldsymbol{a_t} &= \sigma(W_a^w \boldsymbol{w_q^t} + W_a^s \boldsymbol{o_{t-1}}) \\
\boldsymbol{r_t} &= \sigma(W_r^w \boldsymbol{w_q^t} + W_r^s \boldsymbol{o_{t-1}}) \\
\boldsymbol{s_t} &= \tanh\big(W^w \boldsymbol{w_q^t} + W^s(\boldsymbol{r_t} \odot \boldsymbol{o_{t-1}})\big)
\end{aligned} \tag{4.7}$$

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 [22]. 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.

4.2.2.3 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:

$$P(i|u,q) = Score(i|u,q) = \frac{\exp\big(\boldsymbol{i} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\big)}{\sum_{i' \in V_i} \exp\big(\boldsymbol{i'} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\big)} \tag{4.8}$$

where V_i is the set of all possible items and λ is the weight of query model in the final ranking function (Equation 4.2).

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.

4.2.3 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

$$\mathcal{L}(R_u, R_i, u, i, q) = \log P(R_u, R_i, u, i, q) \tag{4.9}$$

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:

$$\begin{aligned}\mathcal{L}(R_u, R_i, u, i, q) &= \log \big(P(R_i, i|u, q)P(R_u, u, q)\big) \\ &= \log \big(P(R_i|i)P(i|u, q)P(R_u|u)P(u)P(q)\big) \\ &= \log \big(P(i|u, q)P(u)P(q)\prod_{w_i \in R_i} P(w_i|i)\prod_{w_u \in R_u} P(w_u|u)\big) \\ &= \log P(i|u, q) + \sum_{w_i \in R_i} \log P(w_i|i) + \sum_{w_u \in R_u} \log P(w_u|u)\end{aligned} \tag{4.10}$$

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 4.3&4.8). 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. [71] and has been extensively used in machine learning and information retrieval [58, 3]. It has been shown to be effective for approximating softmax functions and factorizing the mutual information matrix of two related entities [59]. 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:

$$\begin{aligned} \log P(w_i|i) &= \log \sigma(\boldsymbol{w_i} \cdot \boldsymbol{i}) + k \cdot \mathbb{E}_{w' \sim P_w}[\log \sigma(-\boldsymbol{w'} \cdot \boldsymbol{i})] \\ \log P(w_u|u) &= \log \sigma(\boldsymbol{w_u} \cdot \boldsymbol{u}) + k \cdot \mathbb{E}_{w' \sim P_w}[\log \sigma(-\boldsymbol{w'} \cdot \boldsymbol{u})] \end{aligned} \tag{4.11}$$

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 [71]. Similarly, we compute the log likelihood of our item generation model by conducting negative sampling on items:

$$\begin{aligned} \log P(i|u,q) =& \log \sigma\big(\boldsymbol{i} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\big) \\ &+ k \cdot \mathbb{E}_{i' \sim P_i}\big[\log \sigma\big(-\boldsymbol{i'} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\big)\big] \end{aligned} \tag{4.12}$$

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 [3, 97], we add L2 regularizations on the distributed representations of words, users and items. The final optimization goal is:

$$
\begin{aligned}
\mathcal{L}' =& \sum_{u,i,q} \mathcal{L}(R_u, R_i, u, i, q) + \gamma\Big(\sum_{w \in V_w} \boldsymbol{w}^2 + \sum_{u \in V_u} \boldsymbol{u}^2 + \sum_{i \in V_i} \boldsymbol{i}^2\Big) \\
=& \sum_{u,i,q} \Bigg(\sum_{w_i \in R_i} \big(\log \sigma(\boldsymbol{w_i} \cdot \boldsymbol{i}) + k \cdot \mathbb{E}_{w' \sim P_w}[\log \sigma(-\boldsymbol{w'} \cdot \boldsymbol{i})]\big) \\
& \quad + \sum_{w_u \in R_u} \big(\log \sigma(\boldsymbol{w_u} \cdot \boldsymbol{u}) + k \cdot \mathbb{E}_{w' \sim P_w}[\log \sigma(-\boldsymbol{w'}) \cdot \boldsymbol{u})] \\
& \quad + \log \sigma\big(\boldsymbol{i} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\big) \\
& \quad + k \cdot \mathbb{E}_{i' \sim P_i}\big[\log \sigma\big(-\boldsymbol{i'} \cdot (\lambda \boldsymbol{q} + (1-\lambda)\boldsymbol{u})\big)\big] \Bigg) \\
& + \gamma\Big(\sum_{w \in V_w} \boldsymbol{w}^2 + \sum_{u \in V_u} \boldsymbol{u}^2 + \sum_{i \in V_i} \boldsymbol{i}^2\Big)
\end{aligned}
\tag{4.13}
$$

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.

4.3 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.

4.3.1 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

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

Table 4.1: Statistics for the 5-core data for *Electronics*, *Kindle Store*, *CDs&Vinyl* and *Cell Phones&Accessories* [65]. For example, $a \pm v$ means that the average value is a and the standard deviation is v.

	Electronics	*Kindle Store*
Corpus		
Number of reviews	1,689,188	982,618
Review length	118.27±158.12	112.21±129.52
Number of items	63,001	61,934
Review per item	26.81±75.82	15.87±21.42
Number of users	192,403	68,223
Review per user	8.78±8.26	14.40±24.61
Queries		
Number of queries	989	4,603
Query length	6.40±1.64	7.07±1.89
Queries per item	1.02±0.23	5.08±2.04
Queries per user	8.13±5.84	35.65±37.48
Train/Test		
Number of reviews	1,275,432/413,756	720,006/262,612
Number of queries	904/85	3313/1290
Number of user-query pairs	1,204,928/5,505	1,490,349/232,668
Relevant items per pairs	1.12±0.48/1.01±0.09	1.87±3.30/1.48±1.94

	CDs & Vinyl	*Cell Phones & Accessories*
Corpus		
Number of reviews	1,097,591	194,439
Review length	174.57±177.05	93.50±131.65
Number of items	64,443	10,429
Review per item	17.03±28.15	18.64±34.24
Number of users	75,258	27,879
Review per user	14.58±39.13	6.97±4.55
Queries		
Number of queries	694	165
Query length	5.71±1.62	5.93±1.57
Queries per item	4.04±1.92	1.11±0.38
Queries per user	21.75±16.53	4.95±2.60
Train/Test		
Number of reviews	804,090/293,501	150,048/44,391
Number of queries	534/160	134/31
Number of user-query pairs	1,287,214/45,490	114,177/665
Relevant items per pairs	2.57±6.59/1.30±1.19	1.52±1.13/1.00±0.05

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. [65] 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.

4.3.2 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" [87]. 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. [97], we extract the search queries for each item with a three-step process. First, we 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* $\rightarrow$ *Digital Camera Lenses* would be converted to "photo digital camera lenses"). Some example queries are shown in Table 4.2.

Table 4.2: Example queries extracted following the paradigm proposed by Gysel et al. [97] 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

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 4.1.

4.3.3 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 4.1.

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.

4.3.4 Baselines

For model evaluation, we used three types of baselines: the query likelihood model [80] (namely the standard language modeling approach), an extended query likelihood with user models, and the latent semantic entity model [97]. 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 [80]. 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

$$P_{QL}(Q|D) = \sum_{w \in Q} tf_{w,Q} \log \frac{tf_{w,D} + \mu P(w|C)}{|D| + \mu} \tag{4.14}$$

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

$$P_{UQL}(U, Q|D) = \lambda P_{QL}(Q|D) + (1 - \lambda) P_{QL}(U|D) \tag{4.15}$$

Similar to Equation 4.2, 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 4.4.1&4.4.2. 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. [97] 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:

$$f_E(s) = \tanh(W_E \cdot (\frac{1}{|s|}\sum_{w \in s} \boldsymbol{w}) + b) \tag{4.16}$$

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 4.16 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*.

4.3.5 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 4.2&4.15) 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 4.4.2&4.4.3. 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].

Table 4.3: 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 $\ddagger$ denote significant differences to QL, UQL and LSE in Fisher randomization test [91] with $p \leq 0.01$. The best performance is highlighted in boldface.

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

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

4.4 Results and Discussion

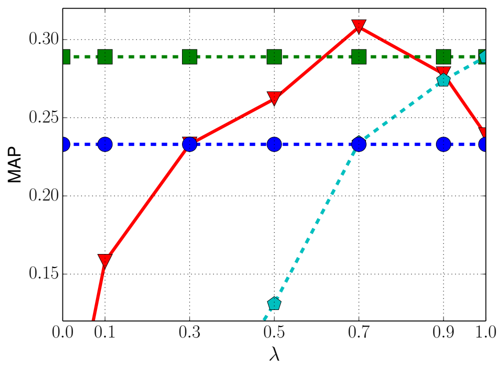
Now we report our results on personalized product search benchmarks. We first show the overall retrieval performance of our hierarchical embedding models and baselines on different Amazon product datasets. Then we discuss the effect of user models in personalized product search. After that, we analyze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

4.4.1 Retrieval Performance

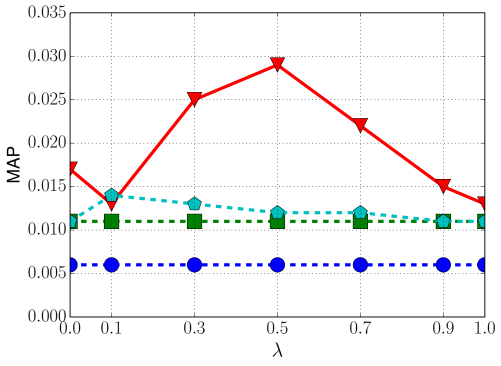
Table 4.3 shows the overall results of baselines and our models on the personalized product search benchmarks of Amazon data *Electronics*, *Kindle Store*, *CDs & Vinyl* and *Cell Phones & Accessories*. In the Table 4.3, QL represents the query likelihood model [80]; UQL represents the extended query likelihood with user models; LSE represents the model of Latent Semantic Entity [97], and HEM denotes our hierarchical embedding models with ϕ function as mean vector (*mean*, Equation 4.5), projected mean (*pm*, Equation 5.7) and recurrent neural network (*RNN*, Equation 4.7). We conducted significant tests over QL, UQL and LSE for all models. All metrics reported in Table 4.3 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.

As shown in Table 4.3, 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).

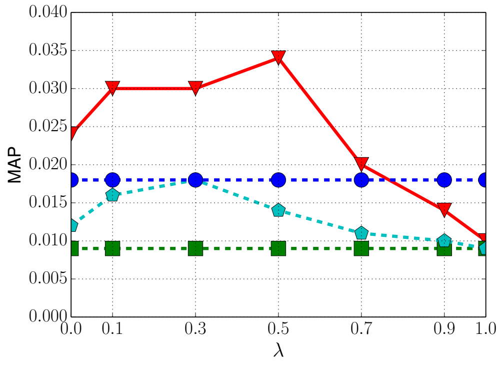
The relative performance of unigram models (QL and UQL) compared to latent space models (LSE, HEM) also varies on different datasets. On "easy" datasets such as *Electronics* and *Cell Phones and Accessories*, the performance of QL and UQL is comparable or better than the latent space baseline (LSE) 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



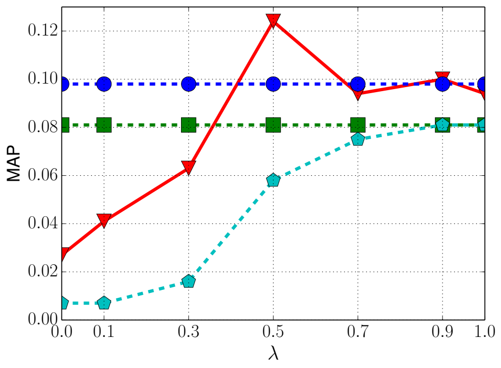
(a) *Electronics*



(b) *Kindle Store*



(c) *CDs & Vinyl*



(d) *Cell Phones & Accessories*

Figure 4.3: 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.

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. [97] 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 4.3 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 [109, 98] 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 4.3. 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 [97, 41], 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.

4.4.2 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 4.3. 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 4.3, 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.

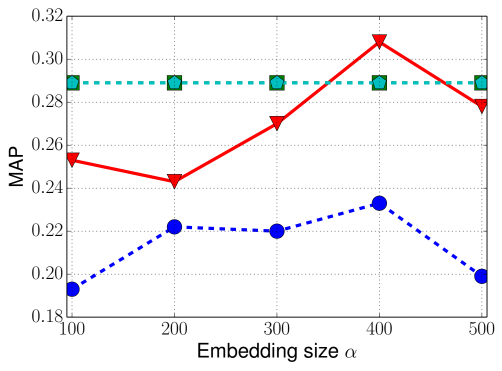
As discussed previously, the types of products not only influence the difficulty of product search but also affect the usefulness of personalization. According to our experiments, the needs of personalization on books (*Kindle Store*) and music (*CDs & Vinyl*) are higher than those on electronic devices (*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.

4.4.3 Embedding Size

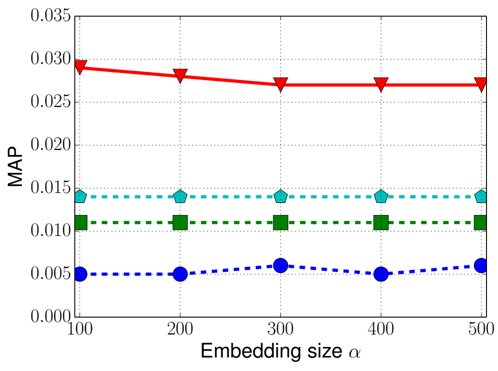
To analyze the effect of embedding sizes and potentially provide guides for future studies, we show the results of our hierarchical embedding models with different embedding sizes in Figure 4.4. The horizontal axes represent the size of embedding vectors for queries, users and items in our experiments.

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.

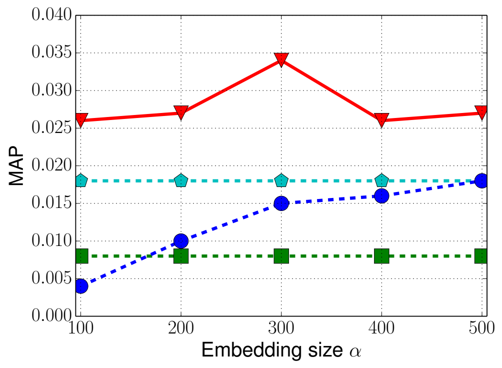
Arora et al. [7] conducted both empirical and theoretical analyses on word embedding models and argued that low dimension vectors (i.e. 300 dimensions) were



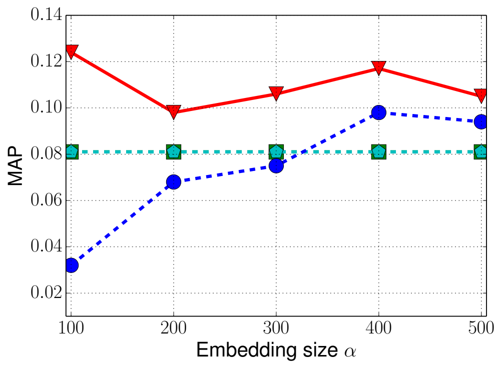
(a) *Electronics*



(b) *Kindle Store*



(c) *CDs & Vinyl*



(d) *Cell Phones & Accessories*

Figure 4.4: 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.

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.

4.5 Conclusion

In this chapter, 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.

CHAPTER 5

EXTENDABLE AND EXPLAINABLE NEURAL REPRESENTATION LEARNING FRAMEWORK

5.1 Introduction

The hierarchical embedding model proposed in Chapter 4 is a well-defined example of neural generative models. It is constructed with semi-structured language data including query strings, product descriptions, and user reviews, and it directly optimizes the probability of retrieving a relevant product given a user query based on a generative framework. Nonetheless, information retrieval problems in practice often involve more complicated needs and information relationships.

For example, showing relevant items on the top of result pages is not enough to guarantee the effectiveness of product search. Existing studies based on this paradigm often simplify the problem of product search by assuming that users will purchase an item as long as it is observed and relevant [97, 6]. They ignore the fact that there is a significant gap between the item relevance perceived by search engines and e-shopping users. Because purchasing is expensive and highly personal [6], users often need a good reason to justify their purchases. As modern product search systems become increasingly sophisticated, it is difficult for normal users to understand why search engines retrieve certain items for them. A direct consequence is that users may not perceive a retrieved item as relevant even when it satisfies their search intents.

To actually optimize user purchases, a good product search engine needs to retrieve relevant products as well as providing good explanations of why retrieved items should be interesting to users. Previous studies on product recommendation have shown that providing appropriate explanations significantly improves user acceptance for recommended items [46, 95]. It benefits recommendation systems in multiple ways including user satisfaction, system transparency, debugging complexity, etc. [15, 25, 96, 115]. Despite its potential, the explainability of retrieval systems has not been well studied in product search. There are two problems that limit the development of explainable product search systems. First, purchasing is a complicated behavior as it depends on multiple factors and heterogeneous information such as user reviews, product metadata, and search context. To provide high-quality explanations, we need to consider the relationship between users and products from multiple angles (e.g., brands, categories, etc.). As far as we know, no existing retrieval model can directly incorporate heterogeneous information for product search. Second, producing a readable explanation requires the system to have logical reasoning. For example, the system should be able to infer that "*Bob* likes *Apple* products" after seeing him search and purchase multiple products from *Apple*. An explanation is reasonable and effective only when it is formulated based on well-grounded logic, while how to construct such a product retrieval model with logical reasoning ability is still an open question for the IR community.

As discussed in Chapter 1, statistical generative retrieval frameworks tend to have good explainablity while deep learning models could be highly effective and flexible in terms of representation learning. Thus, in this chapter, we propose a generic neural representation learning framework that combines the merits of both statistical generative models and deep neural work to tackle the problem of explainable product search. Inspired by the studies of relation prediction in knowledge base [19, 18], we create a unified knowledge graph on multiple types of product data, and conduct retrieval with it. Our motivation is to integrate multi-relational product information for search, and generate explanations with logic inference on the knowledge graph. Empirical experiments and analysis with Amazon benchmark datasets show that incorporating different product knowledge with DREM has significant potential for explainable product search.

Our main contributions can be summarized as follows:

- We propose a Dynamic Relation Embedding Model to construct a session-dependent knowledge graph for product retrieval.
- We propose a Soft Matching Algorithm to efficiently extract explainable paths with knowledge embeddings for search explanations.
- We conducted both retrieval experiments and case studies to verify the effectiveness of the proposed approach in product retrieval and explainable search.

The rest of this chapter is organized as follows. In Section 5.2 and 5.3, we introduce our approach and how to extract explanations for product search. Then, we describe our experimental setup and results in Section 5.4 and 5.5. Finally, we conclude our work in Section 5.6.

5.2 Model Description

We now provide detailed descriptions of the Dynamic Relation Embedding Model (DREM) for explainable product search. DREM jointly models different user/product knowledge, and creates a knowledge graph with both static and dynamic relationships. In this section, we first provide an overview of DREM and describe how to conduct product search with it. Then we discuss the modeling of static and dynamic entity relationships in detail.

5.2.1 Overview

As discussed previously, explainable product search requires retrieval systems to be capable of modeling and conducting logical inference with different product information. The relationships between product-related entities usually are complicated

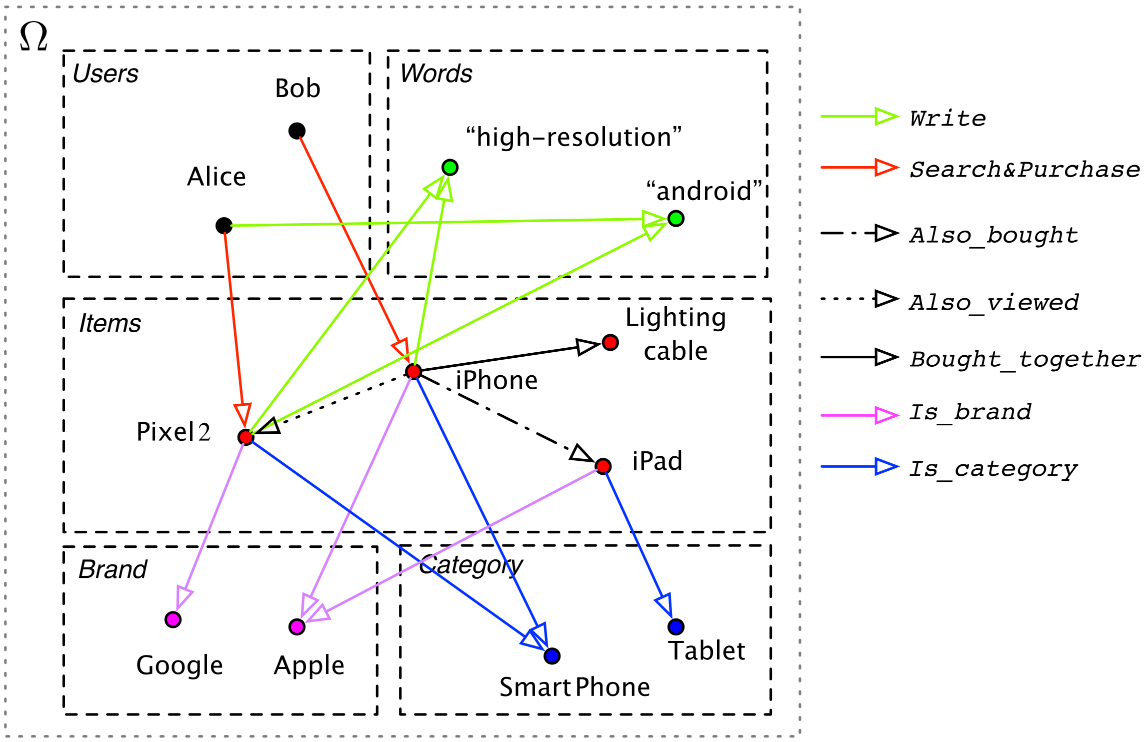


Figure 5.1: An example of the knowledge graph created by DREM for product search.

and not injective. For example, an item could belong to multiple categories and a category could include multiple items. One of the most popular methods to model such multi-relational data is to construct a *knowledge graph*. In a knowledge graph, each node represents an entity and each edge represents the existence of a certain relationship between two entities. With this design, a knowledge graph satisfies the need of logical inference because any relationship between an arbitrary pair of entities can be inferred by the path between them. In this chapter, we design DREM to construct a knowledge graph for product data, and conduct explainable product search accordingly.

An example of a knowledge graph created by DREM is shown in Figure 5.1. In DREM, each user, product and related entities are represented with vectors in a single latent space Ω. Two entities are linked with an edge if there is a relationship between them. Each edge is labeled with a unique symbol to denote the type of the relationship. In order to conduct product search with DREM, we create a special edge named as *Search&Purchase* to model the relationship between users and items. In a search session, a user will be *Search&Purchase* with an item if he or she purchases the item. Thus, the problem of product search is to find items that are likely to have the *Search&Purchase* relationship with the users.

Inspired by previous work on relationship prediction [18], we assume that all relationships can be viewed as translations from one entity to another. Suppose that there exists a relationship r between two entities x and y. Let x be the head entity and y be the tail entity, then we can directly get y by translating x with r as

$$\boldsymbol{y} = \boldsymbol{x} + \boldsymbol{r}$$

Therefore, any relationship in Ω can be treated as a linear transformation between entities and represented with a latent vector that shares the same dimensionality with Ω. We refer to the latent vectors of entities and their relationships in Ω as *entity embeddings* and *relationship embeddings*, respectively.

The key of DREM is to effectively infer the embedding representations of entities and relationships. Although a variety of methods has been proposed for learning knowledge base embeddings [19, 18], none of them is applicable to DREM because the knowledge graph of DREM is not static. Usually, user intent varies in different search sessions. The relationship between users and items cannot be determined without the search context. Therefore, *Search&Purchase* is a dynamic relationship that must be computed for product search on the fly. In the next sections, we describe how to jointly model static and dynamic relationships in DREM.

5.2.2 Static Relation Modeling

Given the formulation of entities and relationships in the latent space, a generic solution to estimate the embedding parameters is to use an EM-like algorithm [67].

For example, we can iteratively minimize the empirical loss of (x, r) by computing $\boldsymbol{r}$ as the mean of $\boldsymbol{y} - \boldsymbol{x}$ for all entity pairs with r, and computing x as the mean of $\boldsymbol{y} - \boldsymbol{r}$ for all entity pairs with both x and r. Such a method, however, is not appropriate for search problems because it does not explicitly differentiate entities with and without the relationship. A trivial solution that gives similar representations to all x and y in Ω could still have a low empirical loss of (x, r) in practice. Based on this consideration, we propose to learn DREM by maximizing the posterior probability of observed relationships and minimizing the unobserved ones.

Let r be a static relationship between head entity $x \in X_r$ and tail entity $y \in Y_r$. X_r and Y_r are the sets of all possible entities that are of the same types with x and y, respectively. We refer to r as static because it holds for x and y universally regardless of the search context. Inspired by the study of embedding-based generative framework [71, 5, 6], we define the probability of observing tail entity y given head entity x and relationship r as

$$P(y|x, r) = \frac{\exp\big((\boldsymbol{x} + \boldsymbol{r}) \cdot \boldsymbol{y}\big)}{\sum_{y' \in Y_r} \exp\big((\boldsymbol{x} + \boldsymbol{r}) \cdot \boldsymbol{y'}\big)} \tag{5.1}$$

where $\boldsymbol{r} \in \mathbb{R}^{\alpha}$, $\boldsymbol{x} \in \mathbb{R}^{\alpha}$ and $\boldsymbol{y} \in \mathbb{R}^{\alpha}$ are the embedding representations of r, x and y with α dimensions. We directly optimize DREM through maximizing the log likelihood of observed $(x, r, y) \in S_{(x,r,y)}$ triples for all static relationships as

$$\mathcal{L}(S_{(x,r,y)}) = \sum_{(x,r,y) \in S_{(x,r,y)}} \log P(y|x, r) \tag{5.2}$$

where $S_{(x,r,y)}$ is the set of all observed static (x, r, y) triples in the training data. As shown in Equation (5.1), $P(y|x, r)$ is a softmax function over y, which essentially assumes that $\sum_{y \in Y_r} P(y|x, r) = 1$. Therefore, the maximization of $\mathcal{L}(S_{(x,r,y)})$ will minimize the probability of unobserved (x, r, y).

Optimizing $\mathcal{L}(S_{(x,r,y)})$ directly, however, is prohibitive in practice. The computational complexity of $\mathcal{L}(S_{(x,r,y)})$ is $\mathcal{O}(\alpha|S_{(x,r,y)}||Y_r|)$, and the size of $S_{(x,r,y)}$ and Y_r can be large (e.g., there are millions of items in Amazon product datasets). To efficiently train DREM on large-scale data, we adopt a negative sampling strategy to approximate $P(y|x,r)$ in $\mathcal{L}(S_{(x,r,y)})$. Negative sampling was first proposed by Mikolov et al. [72] and has been widely applied in machine learning and information retrieval tasks [72, 58, 3, 114, 6]. The idea of negative sampling is to approximate the denominator of softmax functions by randomly sampling some negative samples from the corpus. Specifically, we sample negative instance y' from Y_r and compute $\log P(y|x,r)$ as

$$\log P(y|x,r) = \log\sigma\big((\boldsymbol{x}+\boldsymbol{r})\cdot\boldsymbol{y}\big) + k\cdot\mathbb{E}_{y'\sim P_r}[\log\sigma\big(-(\boldsymbol{x}+\boldsymbol{r})\cdot\boldsymbol{y'}\big)] \tag{5.3}$$

where k is the number of negative instances, $\sigma(x)=\frac{1}{1+e^{-x}}$ is the sigmoid function and P_r is a noise distribution for $y\in Y_r$.

In fact, as shown by previous studies [59], the optimization of $\mathcal{L}(S_{(x,r,y)})$ with the negative sampling strategy is theoretically principled as it essentially guides the model to factorizing the matrix of mutual information between relations and entities. Let $\ell(x,r,y)$ be the expected loss on a specific relation triple $(x,r,y)\in S_{(x,r,y)}$ based on Equation (5.2) and Equation (5.3), then we have

$$\ell(x,r,y) = \#(x,r,y)\cdot\log\sigma(\boldsymbol{y}\cdot(\boldsymbol{x}+\boldsymbol{r})) + k\cdot\#(x,r)\cdot P_r(y)\cdot\log\sigma(-\boldsymbol{y}\cdot(\boldsymbol{x}+\boldsymbol{r})) \tag{5.4}$$

where $\#(x,r,y)$ and $\#(x,r)$ are the numbers of observed relation triple (x,r,y) and pair (x,r) in $S_{(x,r,y)}$. If we derive the partial gradient of $\ell(x,r,y)$ with respect to $\boldsymbol{y}\cdot(\boldsymbol{x}+\boldsymbol{r})$ and let it be zero, we can easily get the following result:

$$\boldsymbol{y}\cdot(\boldsymbol{x}+\boldsymbol{r}) = \log(\frac{\#(x,r,y)}{\#(x,r)}\cdot\frac{1}{P_r(y)}) - \log k \tag{5.5}$$

In this work, we follow the common practice of defining $P_r(y)$ as the normalized frequency of y in all observed (x, r, y) for r, so the right hand side of Equation (5.5) is actually a shifted version of pointwise mutual information between (x, r) and y, and optimizing $\mathcal{L}(S_{(x,r,y)})$ with negative sampling is similar to factorizing the mutual information matrix of $(x, r, y) \in S_{(x,r,y)}$.

5.2.3 Dynamic Relation Modeling

In DREM, we create a relationship between users and products named as *Search&Purchase*. Due to the nature of search tasks, this relationship is dynamic and cannot be determined without the search context. For example, Canon cameras could be linked with users when the query is "digital camera", but not when it is "mobile phone". Therefore, the embeddings of *Search&Purchase* are session-dependent and have to be computed on the fly. For simplicity, we represent the context of a product search session with the query submitted by the user. Note that other session information such as previous queries and clicks [89] can also be incorporated into the framework if needed.

Let q be the query submitted by user u, $\{w_q\}$ be the words of the query, and $\boldsymbol{v}$ be the embedding representation of the relationship *Search&Purchase*. Then we can compute $\boldsymbol{v}$ with a function of q as

$$\boldsymbol{v} = f(q) = f(\{w_q | w_q \in q\}) \tag{5.6}$$

Previous studies have proposed several methods to model search intents with queries in latent space [109, 98, 97, 6]. In Chapter 4, we have explored and compared three options including averaged word vectors [98], non-linear projections [97], and recurrent neural networks (RNN) [78]. In their experiments, the non-linear projection method usually produces the best and most robust results. Suppose that the latent space of DREM has α dimensions, then the non-linear projection method defines $f(q)$ as

$$f(q) = \tanh(W \cdot \frac{\sum_{w_q \in q} \boldsymbol{w_q}}{|q|} + b) \tag{5.7}$$

where $|q|$ is the length of q, $\boldsymbol{w_q}$ is the embedding of w_q, and $W \in \mathbb{R}^{\alpha \times \alpha}$, $b \in \mathbb{R}^{\alpha}$ are parameters to be learned in the training process. In this work, we employ this non-linear projection function to compute $f(q)$. We tried other query embedding functions [98, 78] as well, but observed no significant performance improvement in our retrieval experiments.

Similar to the modeling of static relationships, we use a softmax function to compute the conditional probability of item i given user u with the dynamic relationship v as:

$$P(i|u, \boldsymbol{v}) = \frac{\exp\big((\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{i}\big)}{\sum_{i' \in I} \exp\big((\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{i'}\big)} \tag{5.8}$$

where I is the set of all items. Again, we employ the negative sampling strategy to approximate the log likelihood of observed (u, v, i) triples as

$$\log P(i|u, \boldsymbol{v}) = \log \sigma\big((\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{i}\big) + k \cdot \mathbb{E}_{i' \sim P_i}[\log \sigma\big(-(\boldsymbol{u} + \boldsymbol{v}) \cdot \boldsymbol{i'}\big)] \tag{5.9}$$

where P_i is an uniform distribution for $i \in I$. Let $D_{(u,v,i)}$ be the set of observed (u, v, i) triples in the training data, then the final optimization goal of DREM is to maximize the log likelihood of $D_{(u,v,i)}$ and $S_{(x,r,y)}$ as

$$
\begin{aligned}
\mathcal{L} = & \sum_{(u,v,i)\in D_{(u,v,i)}} \log P(i|u,v) \quad + \sum_{(x,r,y)\in S_{(x,r,y)}} \log P(y|x,r) \\
= & \sum_{(u,v,i)\in D_{(u,v,i)}} \log \sigma\big((\boldsymbol{u}+\boldsymbol{v})\cdot \boldsymbol{i}\big) + k\cdot \mathbb{E}_{i'\sim P_i}[\log \sigma\big(-(\boldsymbol{u}+\boldsymbol{v})\cdot \boldsymbol{i'}\big)] \\
& + \sum_{(x,r,y)\in S_{(x,r,y)}} \log \sigma\big((\boldsymbol{x}+\boldsymbol{r})\cdot \boldsymbol{y}\big) + k\cdot \mathbb{E}_{y'\sim P_r}[\log \sigma\big(-(\boldsymbol{x}+\boldsymbol{r})\cdot \boldsymbol{y'}\big)] \\
= & \sum_{(u,q,i)\in D_{(u,q,i)}} \log \sigma\big((\boldsymbol{u}+f(q))\cdot \boldsymbol{i}\big) + k\cdot \mathbb{E}_{i'\sim P_i}[\log \sigma\big(-(\boldsymbol{u}+f(q))\cdot \boldsymbol{i'}\big)] \\
& + \sum_{(x,r,y)\in S_{(x,r,y)}} \log \sigma\big((\boldsymbol{x}+\boldsymbol{r})\cdot \boldsymbol{y}\big) + k\cdot \mathbb{E}_{y'\sim P_r}[\log \sigma\big(-(\boldsymbol{x}+\boldsymbol{r})\cdot \boldsymbol{y'}\big)]
\end{aligned}
\tag{5.10}
$$

where *Search&Purchase* (u, q, i) is the only dynamic relation in $D_{(u,v,i)}$. In DREM, $\boldsymbol{u}$, $\boldsymbol{i}$ and embeddings for all other entities are jointly learned with the parameters W and b in Equation (5.7). To conduct product search for a specific user u with query q, we simply rank products $i \in I$ with their estimated purchase probability $P(i|u, \boldsymbol{v})$.

Empirically, the weight of static and dynamic relationships do not need to be equal in the model optimization. To explicitly control their relative importance in the final entity representations, we add a hyper-parameter λ in Equation (5.10) as

$$
\begin{aligned}
\mathcal{L} = & \lambda \sum_{(u,q,i)\in D_{(u,q,i)}} \log \sigma\big((\boldsymbol{u}+f(q))\cdot \boldsymbol{i}\big) + k\cdot \mathbb{E}_{i'\sim P_i}[\log \sigma\big(-(\boldsymbol{u}+f(q))\cdot \boldsymbol{i'}\big)] \\
& + (1-\lambda)\sum_{(x,r,y)\in S_{(x,r,y)}} \log \sigma\big((\boldsymbol{x}+\boldsymbol{r})\cdot \boldsymbol{y}\big) + k\cdot \mathbb{E}_{y'\sim P_r}[\log \sigma\big(-(\boldsymbol{x}+\boldsymbol{r})\cdot \boldsymbol{y'}\big)]
\end{aligned}
\tag{5.11}
$$

For simplicity, we assign equal weights for all relationships in most cases ($\lambda = 0.5$), but we discuss the results of DREM with respect to different λ in Section 5.5.1.3. Also, in this paper, we assume that all users and items have appeared in $D_{(u,v,i)}$ or $S_{(x,r,y)}$ at least once. We leave the exploration of cold-start product search for future studies.

5.2.4 Time Complexity

The construction of DREM includes two phases: the offline training of entity/relation embeddings and the online testing on unobserved user-query pairs. The time complexity in the training phase mainly depends on the dimensionality of embedding vectors and the size of training data. For each static relationship triple, the computation of local loss (Equation (5.3)) is $\mathcal{O}(k\alpha)$, where k is the number of negative samples, and α is the size of each embedding vector. For each dynamic relationship triple, the computation of the relation embedding $\boldsymbol{v}$ (Equation (5.7)) is $\mathcal{O}\big((|q| + \alpha)\alpha\big)$, and the computation of local loss (Equation (5.9)) is $\mathcal{O}\big((|q| + \alpha + k)\alpha\big)$. Thus, the time complexity of training DREM in one epoch is $\mathcal{O}\big((\mathbb{E}_q[|q|] + \alpha + k)\alpha|D_{(u,v,i)}| + k\alpha|S_{(x,r,y)}|\big)$, where $\mathbb{E}_q[|q|]$ is the average number of words in each query, and $|D_{(u,v,i)}|$ and $|S_{(x,r,y)}|$ is the number of observed dynamic and static relation triples, respectively. Because k and α are hyper-parameters, the computation cost of DREM is linear to the size of the training data, which is considered to be efficient in general.

For online testing, each item must be assigned with a score to generate the ranked list for a given user-query pair. As discussed in Section 5.2.3, we rank items according to the estimated purchase probability $P(i|u, \boldsymbol{v})$ in Equation (5.8). Because $\exp(x)$ is a monotone increasing function and the denominator of the softmax function is equal for all items, we can directly rank items based on the dot product between $\boldsymbol{i}$ and $\boldsymbol{u} + \boldsymbol{v}$, which has $\mathcal{O}\big((|q| + \alpha)\alpha\big)$ complexity. Because we only need to compute $\boldsymbol{v}$ once per query, the computation cost for each testing user-query pair is $\mathcal{O}\big((|q| + \alpha + |I|)\alpha\big)$, where $|I|$ is the total number of items in the product collection. Since $|I|$ is much larger than $|q| + \alpha$, the overall complexity is approximately $\mathcal{O}(|I|\alpha)$. To further improve the efficiency, one can reduce $|I|$ by adding additional retrieval phases to filter out irrelevant documents before applying DREM. We leave these for future studies.

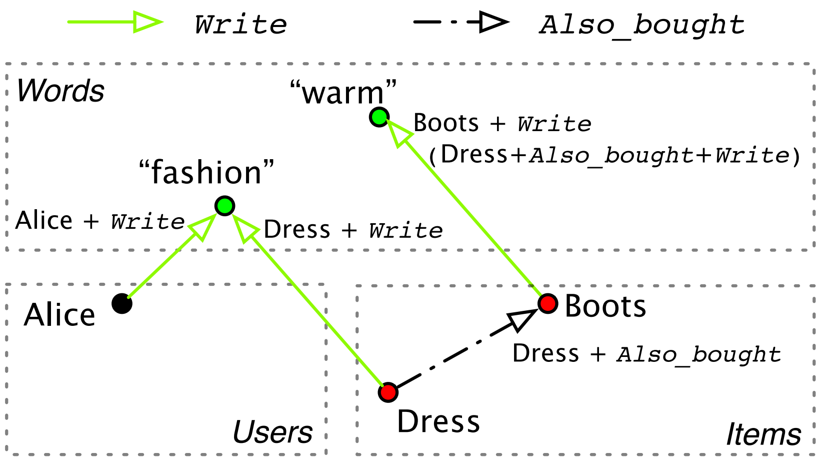


Figure 5.2: An example explanation path from user *Alice* to item *Dress* through word "fashion" in DREM.

5.3 Explanation Extraction

An important advantage of DREM is its support for explainable product search. With the knowledge graph, we can directly infer entity relationships and provide explanations of why retrieved items should be interesting to the users. In this section, we discuss how to construct explanation paths in DREM and extract possible explanations for search results in product search.

5.3.1 Explanation Path

We formulate the problem of explaining why item i is retrieved for user u as finding an explanation path between i and u in the knowledge graph. Figure 5.2 shows an example search session where we retrieve a dress for user *Alice*. As shown in the figure, both the dress and *Alice* are linked with the word "fashion" by the relationship *Write* in the knowledge graph. Based on this observation, we can say that "we retrieve this dress for *Alice* because she often writes about *fashion* in her reviews and *fashion* is frequently used to describe the dress by other users".

Formally, let Ω_x^r and Ω_y^r be the subspaces of Ω for the head and tail entities of relationship r, respectively. Then we define an explanation path from u to i as two lists of relationships $\{r_u^k\}$ (size n) and $\{r_i^j\}$ (size m) that connects u and i:

$$\boldsymbol{u} + \sum_{k=1}^{n} \boldsymbol{r_u^k} = \boldsymbol{e} = \boldsymbol{i} + \sum_{j=1}^{m} \boldsymbol{r_i^j} \tag{5.12}$$

where the head entity space of r_u^1 is same to the entity space of user u ($\Omega_x^{r_u^1} = \Omega_u$), the head entity space of r_i^1 is same to the entity space of item i ($\Omega_x^{r_i^1} = \Omega_i$), the tail entity space of r_u^n is same to the tail entity space of r_i^m ($\Omega_y^{r_u^n} = \Omega_y^{r_i^m}$), and $\Omega_y^{r_u^{k-1}} = \Omega_x^{r_u^k}$ ($k \in [2, n]$), $\Omega_y^{r_i^{j-1}} = \Omega_x^{r_i^j}$ ($j \in [2, m]$). The relationship r_u^k and r_i^j can either be an identity relationship Φ (which projects an entity to itself) or any relationship in the observed data $S_{(x,r,y)}$ and $D_{(u,v,i)}$. Here, e is an entity in $\Omega_y^{r_i^m}$ ($\Omega_y^{r_u^n}$) that links u and i with $\{r_u^k\}$ and $\{r_i^j\}$. Given this explanation path, we can generate a search explanation as "we retrieve item i for user u because u has relationships $\{r_u^k\}$ with e, and i is also linked with e through $\{r_i^j\}$". Therefore, the key of explainable product search is the finding of $\{r_u^k\}$ and $\{r_i^j\}$ given the user u and the retrieved item i.

5.3.2 Extraction Algorithm

Finding an explanation path, however, is difficult for an arbitrary (u, i) pair. Because we only observe a limited number of relationship triples in the training data, the knowledge graph built on product data usually is sparse [113, 6, 114] . In most cases, it is impossible to find two sets of relationships $\{r_u^k\}$ and $\{r_i^j\}$ that directly link the user u to the item i. To tackle this problem, we propose a Soft Matching Algorithm (SMA) to extract explanation paths in DREM.

Let Ω_e be a subspace of Ω that contains all entities with the type of e, and e_u, e_i be the projections of u and i in Ω_e given particular relation paths, then we define the soft matching score for u and i through $e \in \Omega_e$ as

$$S(e|u,i) = \log\big(P(e|e_u)P(e|e_i)\big) = \log P(e|e_u) + \log P(e|e_i) \tag{5.13}$$

where $P(e|e_u)$ and $P(e|e_i)$ are the probability of observing e given e_u and e_i. Intuitively, $P(e|e_u)$ and $P(e|e_i)$ can be model with any functions that measure the similarity between e, e_u, and e_i. In DREM, a straightforward method to compute $P(e|e_u)$ and $P(e|e_i)$ is to adopt the embedding-based generative framework as described in Equation (5.1). This, however, ignores the length of the path from u to i, and could potentially favor long and less meaningful search explanations in practice. To explicitly encourage short explanation paths, we add a decay factor β and define $P(e|e_u)$ and $P(e|e_i)$ as

$$P(e|e_u)=\frac{\exp(\boldsymbol{e_u}\cdot\boldsymbol{e}-\beta n)}{\sum_{e'\in\Omega_e}\exp(\boldsymbol{e_u}\cdot\boldsymbol{e'})}, P(e|e_i)=\frac{\exp(\boldsymbol{e_i}\cdot\boldsymbol{e}-\beta m)}{\sum_{e'\in\Omega_e}\exp(\boldsymbol{e_i}\cdot\boldsymbol{e'})} \tag{5.14}$$

where β is a hyper-parameter that controls the effect of probability decay, and n, m are the length of path $p_u = \{r_u^k\}$, $p_i = \{r_i^j\}$ that translate u, i to e_u, e_i. In this work, we set β as 1.

A summary of SMA for explanation extraction is shown in Algorithm 1. Let $G = \{\Omega_e, r\}$ be a graph where each node Ω_e denotes a subspace of entity type e, and each edge r denotes a relationship that connects two nodes with weight 1. First, given an arbitrary pair (u,i), we find the shortest path from Ω_u to Ω_e and Ω_i to Ω_e as $p_u[e]$ and $p_i[e]$ with the Dijkstra algorithm [33]. Second, we compute their translations e_u, e_i in Ω_e with $p_u[e]$, $p_i[e]$ and their soft matching scores with $e \in \Omega_e$ using Equation (5.13) and (5.14). Finally, we sort entity e from each $\Omega_e \subset \Omega$ with their matching probabilities in descending order, and select the best path $\{p_u[e], e, p_i[e]\}$ to generate search explanations. We manually ignore the path that only contains *Search&Purchase* because it does not provide any information for search explanation.

ALGORITHM 1: Soft Matching Algorithm (SMA)

```
Input: $u$, $i$, $\beta$, $G = \{\Omega_e, r\}$
Output: $path\_set, score\_set$
Procedure Main()
1    Initialize $p_u \leftarrow \{\}, p_i \leftarrow \{\}, path\_set \leftarrow \{\}, score\_set \leftarrow \{\}$
2    for $\Omega_e \in G$ do
3        $p_u[e] = Dijkstra(\Omega_u, \Omega_e, G)$
4        $p_i[e] = Dijkstra(\Omega_u, \Omega_e, G)$
     end
5    for $\Omega_e \in G$ do
6        for $e \in \Omega_e$ do
7            $score\_set[e] = S(e|u, i)$ // Equation (5.13) and (5.14).
8            $path\_set[e] = \{p_u[e], e, p_i[e]\}$.
         end
     end
9    return $path\_set, score\_set$
Function Dijkstra($\Omega_x, \Omega_y, G$)
10   $p_{xy} \leftarrow$ the shortest path from $\Omega_x$ to $\Omega_y$ in $G$;
11   return $p_{xy}$;
```

5.4 Experimental Setup

In this section, we describe the details of our experiment settings. We conduct experiments with Amazon product datasets and compare our method with state-of-the-art product search systems including both text-based models [80] and latent space models [97, 6].

5.4.1 Datasets

The Amazon product dataset[1] is a well-known benchmark for product search and recommendation [97, 6, 114]. It contains information for millions of customers, products and associated metadata including descriptions, reviews, brands, and categories. In our experiments, we used four subsets of the Amazon product data, which are *Electronics*, *Kinde Store*, *CDs & Vinyl*, and *Cell Phones & Accessories*. We use the

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

Table 5.1: Statistics for the 5-core datasets for *Electronics*, *Kindle Store*, *CDs&Vinyl* and *Cell Phones&Accessories* [65]. For example, $a \pm v$ means that the average value is a and the standard deviation is v.

	Electronics	*Kindle Store*
Corpus		
Vocabulary size	142,922	95,729
Number of reviews	1,689,188	982,618
Number of users	192,403	68,223
Number of items	63,001	61,934
Number of brands	3,525	1
Number of categories	983	2,523
Relationships		
Write per user	777.23±1748.6	1174.23±3682.39
Write per item	2373.62±5860.33	1293.47±1916.72
Also_bought per item	36.70±38.56	82.56±29.92
Also_viewed per item	4.36±9.44	0.16±1.66
Bought_together per item	0.59±0.72	0.00±0.04
Is_brand per item	0.47±0.50	0.00±0.00
Is_category per item	4.39±0.95	9.85±2.61
Train/Test		
Number of reviews	1,275,432/413,756	720,006/262,612
Number of queries	904/85	3313/1290
Number of user-query pairs	1,204,928/5,505	1,490,349/232,668
Relevant items per pair	1.12±0.48/1.01±0.09	1.87±3.30/1.48±1.94
	CDs & Vinyl	*Cell Phones & Accessories*
Corpus		
Vocabulary size	202,959	22,493
Number of reviews	1,097,591	194,439
Number of users	75,258	27,879
Number of items	64,443	10,429
Number of brands	1,414	955
Number of categories	770	206
Relationships		
Write per user	1846.88±7667.51	500.01±979.78
Write per item	2156.83±4024.15	1336.64±2698.30
Also_bought per item	57.28±39.22	56.53±35.82
Also_viewed per item	0.27±1.86	1.24±4.29
Bought_together per item	0.68±0.80	0.81±0.77
Is_brand per item	0.21±0.41	0.52±0.50
Is_category per item	7.25±3.13	3.49±1.08
Train/Test		
Number of reviews	804,090/293,501	150,048/44,391
Number of queries	534/160	134/31
Number of user-query pairs	1,287,214/45,490	114,177/665
Relevant items per pair	2.57±6.59/1.30±1.19	1.52±1.13/1.00±0.05

5-core data provided by McAuley et al. [65] where each user and product has at least 5 purchases and 5 reviews.

5.4.1.1 Query Extraction

To the best of our knowledge, no large-scale query log is available on the Amazon dataset. A common paradigm used by previous studies is to extract queries from the category information of each product. Similar to Van Gysel et al. [97], we adopt a two-step process to extract search queries for each user. First, given an arbitrary user and his purchase history, we extract the hierarchical category information of each item with more than two levels. Second, we remove duplicated words and stopwords from a single hierarchy of categories and concatenate the terms to form a topic string. The topic string is then treated as a query submitted by the user which leads to a purchase of the corresponding item. Because users often search for "a producer's name, a brand or a set of terms which describe the category of the product" in e-shopping [87], queries extracted with this paradigm are usually sufficient to simulate real-world product search queries [97, 6].

5.4.1.2 Entities and Relationships

In this work, we consider five types of entities and their relationships in product search. The entities we used are *user*, *item*, *word*, *brand* and *category*. We ignore words that have appeared for less than 5 times in the corresponding corpus. Also, we split hierarchical category information of each product into multiple distinct categories and replace each category as a unique symbol in the training data. For example, a two-level category hierarchy *Camera, Photo* $\rightarrow$ *Digital Camera Lences* will be considered as two separate entities and anonymized as foo_1 and foo_2. An item that belongs to this category hierarchy will be connected to both foo_1 and foo_2.

The relationships used in our experiments include

- *Write* : Word w was written by user u in their reviews ($u \rightarrow w$) or written for item i in the item's reviews ($i \rightarrow w$).
- *Also_bought*: Users who purchased item i_1 previously also purchased item i_2 ($i_1 \rightarrow i_2$).
- *Also_viewed*: Users who viewed item i_1 previously also viewed item i_2 ($i_1 \rightarrow i_2$).
- *Bought_together*: Item i_1 was purchased with item i_2 in a single transaction ($i_1 \rightarrow i_2$).
- *Is_brand*: Item i belongs to brand b ($i \rightarrow b$).
- *Is_category*: Item i belongs to category c ($i \rightarrow c$).

The statistics of entities and relationships in the Amazon product datasets are summarized in Table 5.1. Similar to previous studies [113, 6, 114], the observed relation triples in our data are highly sparse.

5.4.2 Baselines

To demonstrate the effectiveness of the DREM as a product search model, we incorporate five baselines in our experiments: the language modeling approach for IR [80], a probabilistic retrieval model (BM25) [85], a ensemble learning-to-rank model (LambdaMART) [101], the Latent Semantic Entity model [97], and the Hierarchical Embedding Model [6].

5.4.2.1 QL

The language modeling approach for IR, which is often referred to as the Query Likelihood model (QL), is first proposed by Ponte and Croft [80]. It is a unigram model that ranks documents based on the posterior probability of observing the query words given a document's language model estimated with maximum likelihood estimation. In this paper, we concatenate the title, description and reviews of an item in the training data to form a document for it, and compute its ranking scores with respect to a query q based on the language modeling approach with Dirichlet smoothing [111] as:

$$\log(P(q|d)) = \sum_{w \in q} \#(w,q) \log \frac{\#(w,d) + \mu \frac{\#(w,C)}{|C|}}{|d| + \mu}$$

where $\#(w,q)$, $\#(w,d)$, and $\#(w,C)$ are the frequencies of word w in the query q, document d, and the corpus C, respectively; $|d|$ and $|C|$ are the lengths of d and C; and μ is a hyper-parameter that controls the effect of Dirichlet smoothing.

5.4.2.2 BM25

Built on the bag-of-words representations of queries and documents, BM25 is a classic probabilistic retrieval model proposed by Robertson and Walker. [85]. It assumes a 2-Poisson distribution for observed words in the corpus, and ranks documents with a statistical scoring function as

$$BM25(q,d) = \sum_{w \in q} IDF(w,C) \cdot \frac{\#(w,q) \cdot (k_1 + 1)}{\#(w,q) + k_1 \cdot (1 - b + b \cdot \frac{|d|}{avg(|d|)})}$$

where $IDF(w,C)$ is the inverse document frequency of word w in the corpus C, $avg(|d|)$ is the average document length, and k_1 and b are two hyper-parameters for the ranking function. Similar to QL, we concatenate the title, description, and reviews in the training data to form a document for each product.

5.4.2.3 LambdaMART

As a representative study on applying learning-to-rank techniques to product search, Wu et al. [101] construct a LambdaMART model for product search by manually extracting a variety of ranking features for each item with their text data and user behavior logs. In this paper, we construct a learning-to-rank baseline with LambdaMART following the same pipeline used by Wu et al. [101]. Due to the limits of Amazon review datasets, we cannot compute certain features such as session features

Table 5.2: A summary of the ranking features extracted for constructing a learning-to-rank model on the Amazon product search dataset.

Global Statistic Features	
Length	The length of product title, descriptions, reviews.
Purchase	The total number of purchases on each item in the training set.
Distinct Purchase	The distinct number of users who have purchased a certain item in the training set.
Query-item Features	
TF	The average term frequency of query terms in product title, descriptions, reviews, and the whole document (title+description+reviews).
IDF	The average inverse document frequency of query terms in product title, descriptions, reviews, and the whole document (title+description+reviews).
TF-IDF	The average value of $tf \cdot idf$ of query terms in product title, descriptions, reviews, and the whole document (title+description+reviews).
BM25	The scores of BM25 [85] on product title, descriptions, reviews, and the whole document (title+description+reviews).
LMABS	The scores of Language Model (LM) [80] with absolute discounting [112] on product titles, descriptions, reviews, and the whole document (title+description+reviews).
LMDIR (QL)	The scores of LM with Dirichlet smoothing [112] (which is same with QL) on product titles, descriptions, reviews, and the whole document (title+description+reviews).
LMJM	The scores of LM with Jelinek-Mercer [112] on product titles, descriptions, reviews, and the whole document (title+description+reviews).

and time features, but we manage to reproduce most global statistic features and query-item features proposed by Wu et al. [101]. Detailed feature descriptions are listed in Table 5.2.

5.4.2.4 LSE

The Latent Semantic Entity model (LSE) is the first latent space model proposed for product search by Van Gysel et al. [97]. It encodes queries and n-grams with a non-linear projection function similar to Equation (5.7). It also learns the embedding representations of items by maximizing the similarity between an item and the encoded n-grams extracted from the corresponding item reviews. Specifically, for each n-gram s in the product review of an item i, the similarity between s and i in LSE is computed as

$$P(i|s) = \sigma(\boldsymbol{i} \cdot f(s))$$

where $\boldsymbol{i}$ is the representation of i in the latent space, $f(x)$ is the non-linear projection function in Equation (5.7), and $\sigma(x)$ is a sigmoid function $\sigma(x) = \frac{1}{1+e^{-x}}$. Products are retrieved based on their similarity with the query in the latent space.

5.4.2.5 HEM

The Hierarchical Embedding Model (HEM) proposed by Chapter 4 is a state-of-the-art retrieval model for personalized product search. It is constructed based on a generative framework which assumes that reviews are generated by the language model of users/items and purchases are generated by the joint model of users and queries. Similar to DREM, HEM learns the embeddings of users, items, and queries by maximizing the likelihood of observed review data, and ranks items based on their posterior probability given the user and the query. However, HEM only considers the information of users, items, and product reviews, and does not differentiate the relations between different types of entities in the optimization process.

5.4.3 Evaluation Methodology

To train and test different product search models, we partitioned each dataset into a training set and a test set. Following the methodology used in Chapter 4, we randomly hide 30% of the user reviews from the training data and use their corresponding purchase information as the ground truth for testing. We randomly select 30% queries as test queries, and if all queries for an item were selected as test queries, we randomly pick one from the test query set and put it back to the training data. After that, we match the queries with user-item pairs in the test set to construct the final test data. An item is relevant to a user-query pair if and only if it is relevant to the query and has been purchased by the user. In this setting, all query-user-item triples in the test set are unobserved in the training process. More statistics about our data partitions are shown in Table 5.1.

To evaluate retrieval performance in our experiments, we adopt three metrics, which include the mean average precision (MAP), the mean reciprocal rank (MRR) and the normalized discounted cumulative gain (NDCG). For each user-query pair, we only retrieve 100 items to generate the rank list. Both MAP and MRR are computed based on the whole rank list, while NDCG is computed only based on the top 10 items. Significant differences are measured by the Fisher randomization test [91] with $p < 0.01$.

5.4.4 Implementation Details

For QL and BM25, we used galago[2] to index and retrieve items. For LambdaMART, we manually extract features from raw data and build the model with an open-source learning-to-rank package ranklib[3]. And for LSE and HEM, we used the same implementation that has been used in Chapter 4. QL, BM25, LSE, and HEM

[2]`https://sourceforge.net/p/lemur/wiki/Galago/`

[3]`https://sourceforge.net/p/lemur/wiki/RankLib/`

conduct product search based on the text information of items, which is the same as DREM built with the *Write* relationship only. To further analyze the usefulness of other relationships, we tested DREM built on *Write* together with other relationships. We refer to DREM with *Also_bought*, *Also_viewed*, *Bought_together*, *Is_brand* and *Is_category* as DREM_{AB}, DREM_{AV}, DREM_{BT}, DREM_{Bnd} and DREM_{Cat}, respectively. DREM with *Write* only and the DREM with all relationships are referred to as DREM_{NoMeta} and DREM_{All}.

The latent space models (LSE, HEM, and DREM) are trained with stochastic gradient descent with batch size 64. We manually clip the norm of batch gradients with 5 to avoid unstable parameter updates. We train each model with 20 epochs and gradually decrease the learning rate from 0.5 to 0 in the training process. For baselines, we tuned the Dirichlet smoothing parameter μ of QL from 1000 to 3000, and the BM25 scoring parameter k_1 and b from 0.5 to 4 and 0.25 to 1, respectively. The number of trees and leaf nodes in LambdaMART are set as 1000 and 10, respectively, and we tuned the personalization weight η of HEM from 0 to 1. We also tuned the embedding size α for LSE, HEM, and DREM from 100 to 500. In order to better illustrate the importance of different product relationships in different datasets, we fix the dynamic relation weight λ in Equation (5.11) as 0.5 for most experiments, but we will discuss its effect in Section 5.5.1.3.

5.5 Results and Discussions

In this section, we report the results of our experiments. We first present and discuss the retrieval performance of DREM and baseline models. Then we provide a case study to analyze the effectiveness of DREM for explainable product search.

Table 5.3: Comparison of baselines and DREM on the Amazon product search datasets. $*$, $+$ and $\dagger$ denote significant differences to all baselines (QL, BM25, LambdaMART, LSE, and HEM), DREM$_{NoMeta}$, and all tested models, respectively, in Fisher randomization test [91] with $p \leq 0.01$. The best performance is highlighted in boldface.

	Electronics			*Kindle Store*		
Model	MAP	MRR	NDCG	MAP	MRR	NDCG
QL	0.289	0.289	0.316	0.011	0.012	0.013
BM25	0.283	0.280	0.304	0.021	0.013	0.014
LambdaMART	0.180	0.181	0.237	0.028	0.029	0.018
LSE	0.233	0.234	0.239	0.006	0.007	0.007
HEM	0.308^{*+}	0.309^{*+}	0.329^{*+}	0.029	0.035^{*}	0.033^{*}
DREM$_{NoMeta}$	0.291	0.291	0.319	0.036^{*}	0.044^{*}	0.042^{*}
DREM$_{AB}$	0.283	0.283	0.312	0.043^{*+}	0.052^{*+}	0.050^{*+}
DREM$_{AV}$	0.318^{*+}	0.319^{*+}	0.349^{*+}	0.035^{*}	0.043^{*}	0.041^{*}
DREM$_{BT}$	0.320^{*+}	0.321^{*+}	0.346^{*+}	0.037^{*}	0.045^{*}	0.042^{*}
DREM$_{Bnd}$	0.314^{*+}	0.315^{*+}	0.340^{*+}	0.037^{*}	0.044^{*}	0.043^{*}
DREM$_{Cat}$	0.299^{+}	0.300^{+}	0.360^{*+}	0.048^{*+}	0.056^{*+}	0.056^{*+}
DREM$_{All}$	**0.366**$^{*+\dagger}$	**0.367**$^{*+\dagger}$	**0.408**$^{*+\dagger}$	**0.057**$^{*+\dagger}$	**0.067**$^{*+\dagger}$	**0.067**$^{*+\dagger}$
	CDs & Vinyl			*Cell Phones & Accessories*		
Model	MAP	MRR	NDCG	MAP	MRR	NDCG
QL	0.009	0.011	0.010	0.081	0.081	0.092
BM25	0.027	0.018	0.016	0.083	0.081	0.115
LambdaMART	0.054^{*+}	0.057^{*+}	0.051^{*+}	0.121	0.121	0.148
LSE	0.018	0.022	0.020	0.098	0.098	0.084
HEM	0.034	0.040	0.040	0.124^{*+}	0.124^{*+}	0.153^{*+}
DREM$_{NoMeta}$	0.034	0.041	0.040	0.107	0.107	0.127
DREM$_{AB}$	0.046^{+}	0.054^{+}	0.054^{+}	0.098	0.098	0.120
DREM$_{AV}$	0.034	0.041	0.040	0.095	0.096	0.096
DREM$_{BT}$	0.037^{+}	0.044^{+}	0.042^{+}	0.089	0.089	0.096
DREM$_{Bnd}$	0.035	0.041	0.040	0.134^{*+}	0.134^{*+}	0.152^{+}
DREM$_{Cat}$	0.059^{*+}	0.068^{*+}	0.070^{*+}	0.193^{*+}	0.193^{*+}	0.229^{*+}
DREM$_{All}$	**0.074**$^{*+\dagger}$	**0.084**$^{*+\dagger}$	**0.086**$^{*+\dagger}$	**0.249**$^{*+\dagger}$	**0.249**$^{*+\dagger}$	**0.282**$^{*+\dagger}$

5.5.1 Retrieval Performance

Table 5.3 summarizes the results of our product search experiments on the four subsets of Amazon product data. We group the models into three groups – the baseline models (QL, BM25, LambdaMART, LSE, HEM); DREM with *Write* and another relationship among *Also_bought*, *Also_viewed*, *Bought_together*, *Is_brand* and *Is_category* (DREM$_{AB}$, DREM$_{AV}$, DREM$_{BT}$, DREM$_{Bnd}$, DREM$_{Cat}$); and the DREM with *Write* only or with all the relationships (DREM$_{NoMeta}$, DREM$_{All}$)

5.5.1.1 Overall Results

As we can see from the table, the relative performance of bag-of-words models (QL, BM25) and latent space models without personalization (LSE) varies across different datasets. While QL and BM25 have comparable performance on all datasets, LSE outperformed them on *CDs & Vinyl* but performed worse than them on *Electronics* and *Kindle Store*. As shown by previous studies [41, 42], the main difference between unigram models and latent space models is their ability to conduct semantic matching. The latter performs well when vocabulary mismatch between queries and documents is severe, while the former works better in other cases. Our results indicate that the severity of vocabulary mismatch is low on *Electronics* or *Kindle Store*, but high on *CDs & Vinyl*. By incorporating personalization, HEM consistently outperformed QL and LSE on all the datasets tested in our experiments. Because purchasing is a highly personalized behavior, incorporating user information can help HEM better understand the search intents of each user and retrieve items that suits different individuals. The results for LambdaMART are more complicated. While it achieved superior performance on *CDs & Vinyl*, it also produced bad results on *Electronics*. Further analysis of ranking features are needed in order to understand why learning-to-rank models perform differently on different product categories, which is beyond the scope of this dissertation, and we will leave it for future studies.

After incorporating other product knowledge information discussed in Section 5.4.1, DREM$_{All}$ significantly outperformed all baseline models on all datasets. Its obtained 19%, 97%, 118% and 101% improvements with respect to MAP over HEM on *Electronics*, *Kinde Store*, *CDs & Vinyl* and *Cell Phones & Accessories*, respectively. This demonstrate the usefulness of multi-relational product data and the effectiveness of DREM as a product retrieval model.

In Table 5.3, the performance of HEM and DREM$_{NoMeta}$ is competitive in most cases. HEM and DREM$_{NoMeta}$ are both constructed based on users, items and their associated reviews. The only difference between them is the method they used to model entity relationships. HEM directly uses user embeddings to predict both review words and purchased items, while DREM$_{NoMeta}$ uses relationship embeddings to project users into the space of words and items separately. According to our results, the two paradigms are equally effective for product search and neither of them is consistently better than the other. However, DREM is more powerful in terms of extendability because it creates a knowledge graph that can integrate different kinds of product information for retrieval tasks.

5.5.1.2 Usefulness of Different Relationships

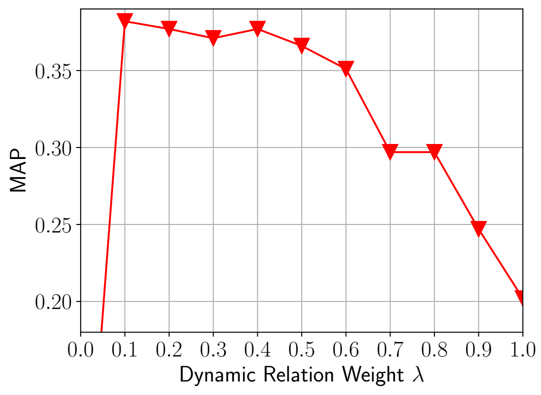
In our experiments, we analyze the importance of different relationships by training DREMs with each of the relationships separately. As shown in Table 5.3, the importance of relationships varies considerably on different datasets. On *Electronics*, nearly all types of product knowledge brought benefits to DREM except DREM$_{AB}$, which is built on the *Write* and *Also_bought* relationships. As shown by in Chapter 4, the importance of personalization for product search is less significant on *Electronics* than on other datasets. Two co-purchased items in *Electronics* are less likely to satisfy the same type of user preference or search intent. For example, users may not intend to buy a keyboard when they search for “mouse”, despite that they often buy keyboards before or after the purchase of a mouse. Therefore, the relationship *Also_brought* introduces less information but more noise for DREM on *Electronics*.

In contrast to *Electronics*, the incorporation of *Also_bought* significantly improved the retrieval performance of DREM on *Kinde Store* and *CDs & Vinyl*. DREM$_{AB}$ outperformed DREM$_{NoMeta}$ by 19% and 35% with respect to MAP on *Kinde Store* and *CDs & Vinyl*, respectively. This indicates that co-purchased items often fit the same need of users in *Kinde Store* and *CDs & Vinyl*. This is reasonable because *Kinde Store* and *CDs & Vinyl* consist of books and music, on which people usually have consistent tastes. If a CD is relevant to a query, then other frequently co-purchased CDs are also likely to be relevant.

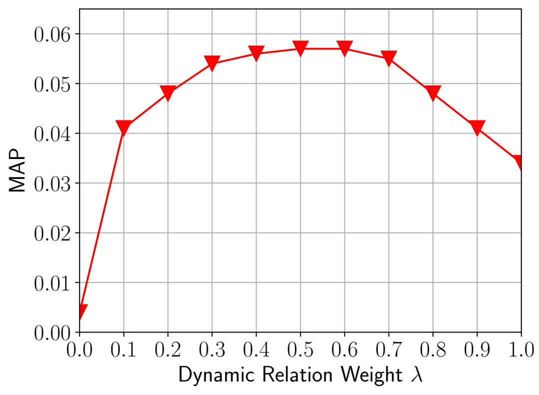
As shown in Table 5.3, we observed that *Is_brand* is more useful for product search on *Cell Phones & Accessories* than on other datasets. On *Cell Phones & Accessories*, DREM$_{Bnd}$ significantly outperformed DREM$_{NoMeta}$ with a 25% improvement on MAP. According to a recent report[4], most people have high loyalty to the manufacturer of their phones and 56% of people who currently possess a smartphone used to own a phone from the same manufacturer. Thus, it is not surprising to see that *Is_brand* exhibits high correlations with user purchases in *Cell Phones & Accessories*.

Although we have split each hierarchical category into distinct categories and anonymized them in model construction, there might be a concern that incorporating category entities in DREM may hurt the fairness of the evaluation since the test queries are generated based on the hierarchy of categories. In our experiment, we indeed observed that DREM with *Is_category* (DREM$_{Cat}$) performed better than the DREM with other relationships on *Kinde Store*, *CDs & Vinyl* and *Cell Phones & Accessories*. However, it's worth noticing that DREMs without *Is_category* also significantly outperformed the state-of-the-art baseline methods. In Table 5.3, DREM$_{BT}$

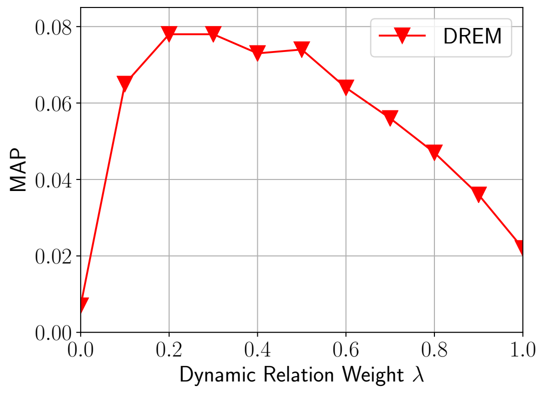
[4]`https://www.statista.com/statistics/716086/smartphone-brand-loyalty-in-us/`



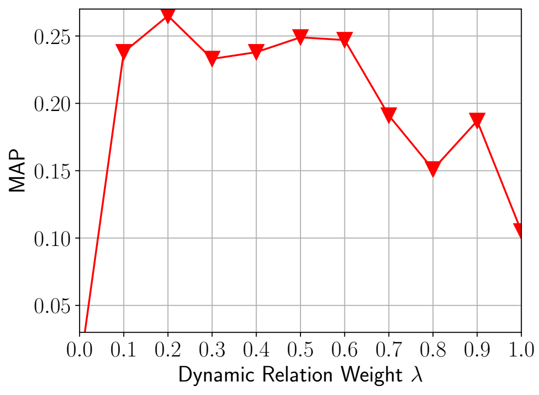
(a) *Electronics*



(b) *Kindle Store*



(c) *CDs & Vinyl*



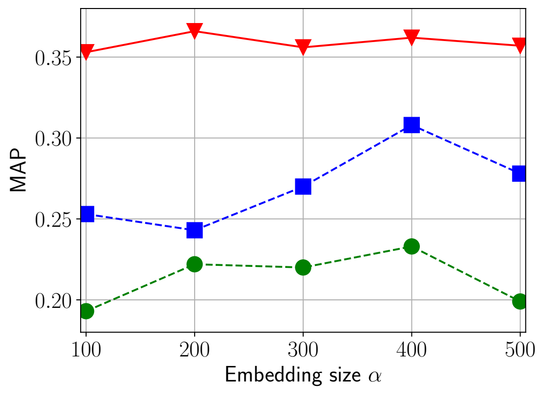
(d) *Cell Phones & Accessories*

Figure 5.3: The performance of DREM with different dynamic relation weight λ

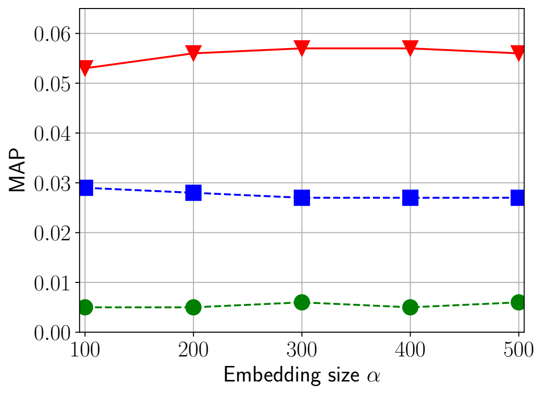
on *Electronics*, DREM$_{AB}$ on *Kinde Store*, DREM$_{AB}$ on *CDs & Vinyl* and DREM$_{Bnd}$ on *Cell Phones & Accessories* obtained 4%, 48%, 35% and 8% improvements on MAP over the best baseline (HEM), respectively. Again, these results indicate the effectiveness of DREM and the usefulness of multi-relational product data for product search.

5.5.1.3 Parameter Sensitivity

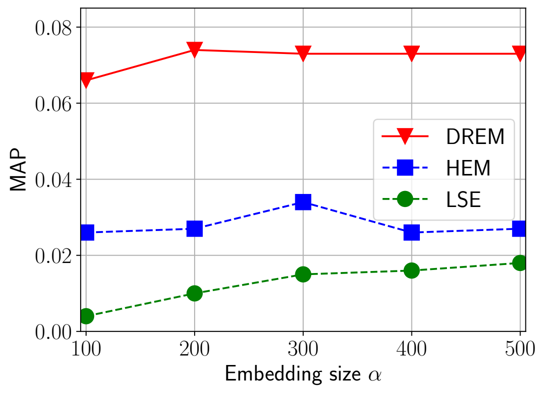
There are two hyper-parameters used in the training of DREM – the dynamic relation weight λ in Equation (5.11) and the embedding size α. To analyze the pa-



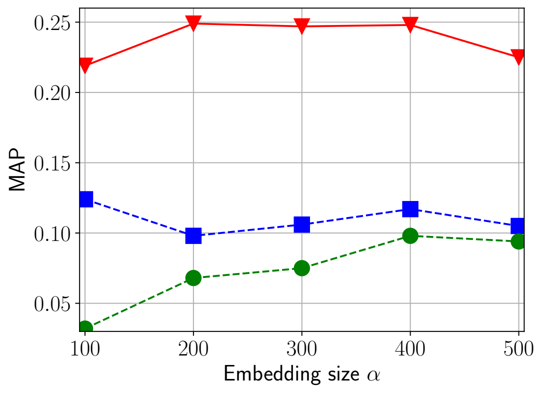
(a) *Electronics*



(b) *Kindle Store*



(c) *CDs & Vinyl*



(d) *Cell Phones & Accessories*

Figure 5.4: The performance of DREM and baselines with different embedding size α. The red solid line with triangles represents the numbers for DREM$_{All}$; the green and blue dashed lines with circles and squares are results for LSE and HEM, respectively.

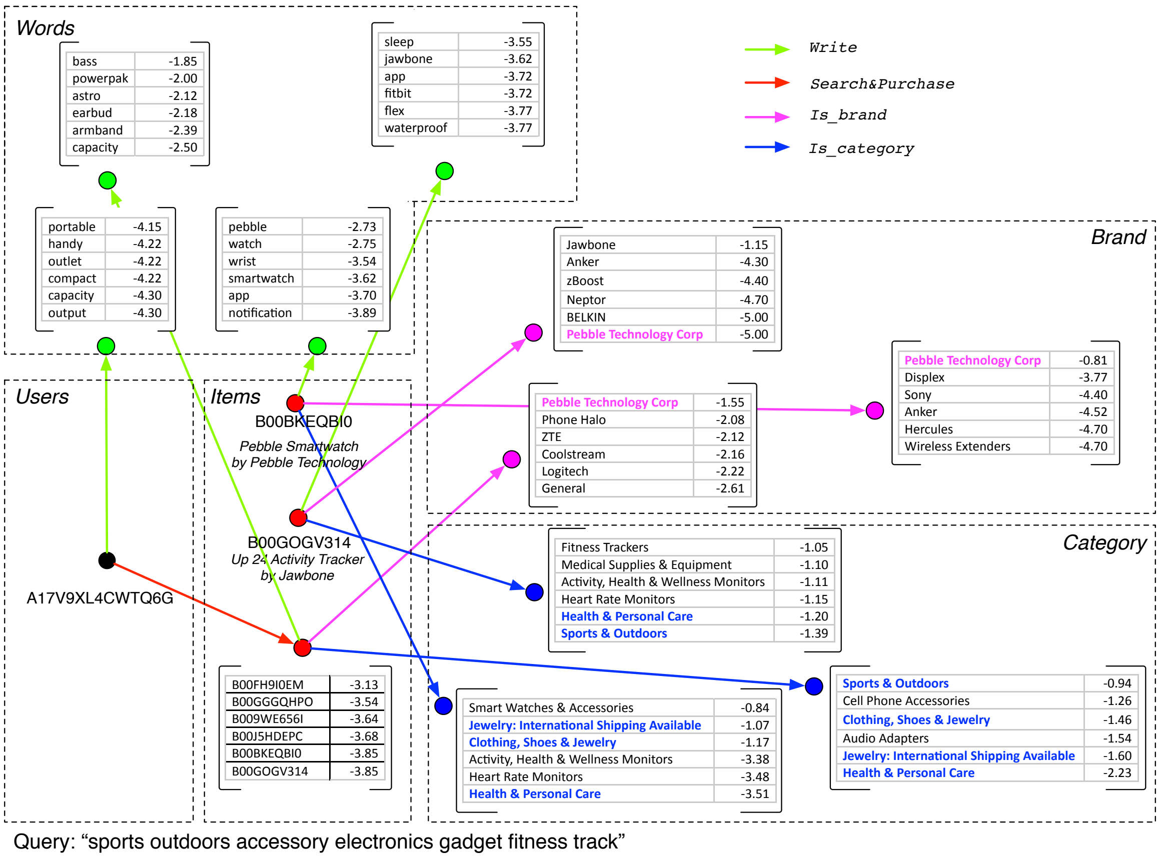
rameter sensitivity of DREM, we plot the MAP of DREM$_{All}$ with different parameter settings in Figure 5.3 and Figure 5.4.

Figure 5.3 shows the performance of DREM$_{All}$ on different product categories with respect to the dynamic relation weight λ ranged from 0 to 1. When $\lambda = 0$, DREM$_{All}$ learns nothing on the dynamic relationships, and search queries would have no influence on the final search results, which means that the model will be degraded from a search model to recommendation model. As expected, the retrieval performance of DREM$_{All}$ with $\lambda = 0$ is significantly worse than other models. When $\lambda = 1$, DREM$_{All}$ does not incorporate any information from static relationships. While it performs reasonable well compared to the text-based baseline models such as QL and LSE, it produces inferior performance compared to DREM$_{All}$ with smaller λ. As shown in Figure 5.3, DREM$_{All}$ usually achieves the best performance when λ is larger than 0.1 but less than 0.7. This demonstrates that both dynamic and static relationship information are valuable for product search.

Figure 5.4 plots the retrieval performance of both baseline methods (LSE and HEM) and DREM$_{All}$. As we can see, the size of embeddings has minor effect on the performance of DREM. DREM$_{All}$ obtained similar results with different α and outperformed LSE and HEM with large margins. Therefore, in practice, we advise to start with a small α and increasing it when necessary.

5.5.2 Case Study

To show the effectiveness of DREM as an explainable product search model, we plot the knowledge graph created by DREM$_{All}$ on *Cell Phones & Accessories* for query "sports outdoors accessory electronics gadget fitness track" in Figure 5.5. We show the nodes and translations of user "A17V9XL4CWTQ6G", item "B00GOGV314" (*Up 24 Activity Tracker* by *Jawbone*), and item "B00BKEQBI0" (*Pebble Smartwatch* by *Pebble Technology*) in different entity subspace. Each edge in the graph represents a


Words
Users
Items
Brand
Category
Write
Search&Purchase
Is_brand
Is_category
B00BKEQBI0
Pebble Smartwatch by Pebble Technology
B00GOGV314
Up 24 Activity Tracker by Jawbone
A17V9XL4CWTQ6G
Query: “sports outdoors accessory electronics gadget fitness track”

bass	-1.85
powerpak	-2.00
astro	-2.12
earbud	-2.18
armband	-2.39
capacity	-2.50

sleep	-3.55
jawbone	-3.62
app	-3.72
fitbit	-3.72
flex	-3.77
waterproof	-3.77

portable	-4.15
handy	-4.22
outlet	-4.22
compact	-4.22
capacity	-4.30
output	-4.30

pebble	-2.73
watch	-2.75
wrist	-3.54
smartwatch	-3.62
app	-3.70
notification	-3.89

Jawbone	-1.15
Anker	-4.30
zBoost	-4.40
Neptor	-4.70
BELKIN	-5.00
Pebble Technology Corp	-5.00

Pebble Technology Corp	-1.55
Phone Halo	-2.08
ZTE	-2.12
Coolstream	-2.16
Logitech	-2.22
General	-2.61

Pebble Technology Corp	-0.81
Displex	-3.77
Sony	-4.40
Anker	-4.52
Hercules	-4.70
Wireless Extenders	-4.70

Fitness Trackers	-1.05
Medical Supplies & Equipment	-1.10
Activity, Health & Wellness Monitors	-1.11
Heart Rate Monitors	-1.15
Health & Personal Care	-1.20
Sports & Outdoors	-1.39

B00FH9I0EM	-3.13
B00GGGQHPO	-3.54
B009WE656I	-3.64
B00J5HDEPC	-3.68
B00BKEQBI0	-3.85
B00GOGV314	-3.85

Smart Watches & Accessories	-0.84
Jewelry: International Shipping Available	-1.07
Clothing, Shoes & Jewelry	-1.17
Activity, Health & Wellness Monitors	-3.38
Heart Rate Monitors	-3.48
Health & Personal Care	-3.51

Sports & Outdoors	-0.94
Cell Phone Accessories	-1.26
Clothing, Shoes & Jewelry	-1.46
Audio Adapters	-1.54
Jewelry: International Shipping Available	-1.60
Health & Personal Care	-2.23

Figure 5.5: The knowledge graph created by DREM$_{All}$ on *Cell Phones & Accessories* for query “sports outdoors accessory electronics gadget fitness track”. Six top retrieved entities and the corresponding probabilities (Equation (5.14)) are shown for each node.

particular type of relationship. Entities are connected with their translations through edges with solid arrows. For clarity, we hide the item-item relationships (*Also_bought*, *Also_viewed* and *Bought_together*) in the graph. On each node, we show a list of six results selected from the top retrieved entities with soft matching (Equation (5.13) and (5.14)). Entities shared by multiple lists in the same subspace are highlighted with colors. We use u, i_j, i_p to denote the node of the user, *Up 24 Activity Tracker* and *Pebble Smartwatch*, and use $\vec{SP}$, $\vec{B}$, $\vec{C}$ to denote the relationships of *Search&Purchase*, *Is_brand* and *Is_category*.

As shown in Figure 5.5, given the Soft Matching Algorithm, we can find the following explanation paths from user "A17V9XL4CWTQ6G" to *Pebble Smartwatch* "B00BKEQBI0":

- $u + \vec{SP} + \vec{B} \rightarrow$**Pebble Technology**$\leftarrow i_p + \vec{B}$ with $S(e|u,i) = -2.36$.
- $u + \vec{SP} + \vec{C} \rightarrow$**Clothing, Shoes, Jewelry**$\leftarrow i_p + \vec{C}$ with $S(e|u,i) = -2.63$.
- $u + \vec{SP} + \vec{C} \rightarrow$**Jewelry:International Ship**$\leftarrow i_p + \vec{C}$ with $S(e|u,i) = -2.67$.
- $u + \vec{SP} + \vec{C} \rightarrow$**Health&Personal Care**$\leftarrow i_p + \vec{C}$ with $S(e|u,i) = -5.84$.

With simple templates, we can create four explanations for why the user should be interested in *Pebble Smartwatch* as

- "Based on your profile and query, you may like to see somethings by *Pebble Technology*, and *Pebble Smartwatch* is a top product of this brand." ($S(e|u,i) = -2.36$)
- "Based on your profile and query, you may like to see somethings in *Clothing, Shoes, Jewelry*, and *Pebble Smartwatch* is a top product in this category." ($S(e|u,i) = -2.63$)
- "Based on your profile and query, you may like to see somethings in *Jewelry:International Ship*, and *Pebble Smartwatch by Pebble Technology* is a top product in this category." ($S(e|u,i) = -2.67$)
- "Based on your profile and query, you may like to see somethings in *Health&Personal Care*, and *Pebble Smartwatch* is a top product in this category." ($S(e|u,i) = -5.84$)

Similarly, for *Up 24 Activity Tracker* "B00GOGV314", we have the following explanation paths that connect it to the search user "A17V9XL4CWTQ6G":

- $u+\vec{SP}+\vec{C}\rightarrow$**Sports&Outdoors**$\leftarrow i_j+\vec{C}$ with $S(e|u,i) = -2.33$:

  "Based on your profile and query, you may like to see somethings in *Sports&Outdoors*, and *Up 24 Activity Tracker* is a top product in this category."

- $u+\vec{SP}+\vec{C}\rightarrow$**Health&Personal Care**$\leftarrow i_j+\vec{C}$ with $S(e|u,i) = -3.43$:

  "Based on your profile and query, you may like to see somethings in *Health&Personal Care*, and *Up 24 Activity Tracker* is a top product in this category."

- $u+\vec{SP}+\vec{B}\rightarrow$**Pebble Technology**$\leftarrow i_j+\vec{B}$ with $S(e|u,i) = -5.81$

  "Based on your profile and query, you may like to see somethings by *Pebble Technology*, which is a top brand related to *Up 24 Activity Tracker by Jawbone*."

Given more information on the query and corresponding products, we find that most explanations above are actually reasonable. According to the query, the user is looking for electronic fitness trackers. *Pebble Technology* is a company famous for its fitness tracking devices, while *Pebble Smartwatch* is one of its bestsellers. Also, when the user is searching for fitness trackers within the domain of *Cell Phones & Accessories*, it is likely that he or she is interested in wearable devices with health tracking functions. *Pebble Smartwatch* is a wearable device well-known for its multi-functionality and stylish design, while *Up 24 Activity Tracker* is one of its competitors that focuses on health tracking functions and has a cheaper price. It is reasonable to recommend the former based on its popularity in *Clothing, Shoes, Jewelry*, while recommend the latter based on its relationship with *Health&Personal Care*. In fact, the query word "fitness" is more related to *Health&Personal Care*, and the user purchased *Up 24 Activity Tracker* in the end.

5.6 Conclusion

In this chapter, we present our initial attempt to tackle the problem of explainable product search with a generic neural representation learning framework. We propose a Dynamic Relation Embedding Model that jointly learns embedding representations for entities/relationships and creates session-dependent knowledge graphs. The proposed model is extendable as it can easily incorporate arbitrary types of information and relationships by adding new nodes and edges into the graphs. Empirical experiments show that our approach significantly outperforms the state-of-the-art product retrieval methods and has the ability to produce reasonable explanations for search results. This indicates that the construction of dynamic knowledge graph with multi-relational product data is beneficial for both the effectiveness and explainability of retrieval models.

CHAPTER 6
SUMMARY AND FUTURE WORK

In this chapter, we provide a brief summary of our work. We begin by summarizing how to build neural generative models for information retrieval based on the optimization theory and our empirical experience introduced in the previous chapters. Then, we present a comprehensive summary of our experimental results on ad-hoc retrieval and product search. Finally, we discuss potential directions for future studies.

6.1 Constructing Neural Generative Models for IR

The advantages of neural generative models are their flexibility and effectiveness in learning representations specifically designed for the need of their tasks. In this dissertation, we develop a generic neural generative representation learning framework for information retrieval. Specifically, to design a neural generative model under the proposed framework in practice, we recommend taking the following steps:

1. **Construct data graphs**. Usually, information is expressed with information units and the connections between them. In this thesis, we propose to build neural generative models for information retrieval by creating a data graph for both structured and unstructured data. For example, in ad-hoc retrieval, we treat documents and words as separate information units and construct data graphs by connecting them based on their co-occurrence.

2. **Make data assumptions**. Generative representation learning frameworks naturally structure data in hierarchical manners when they require certain entities

or models to generate other information units (e.g., generating words from language models). In this thesis, we propose to construct data assumptions based on the data graph and the specific needs of each IR task. Particularly in product search, we propose to model the relationships between users, queries, and products by requiring the joint model of users and queries to generate the corresponding purchased products in each search session.

3. **Select optimization and learning strategy**. Neural models can only learn what we tell them to learn. Thus, the selection of optimization objectives and learning strategies often have a direct influence on the performance of an IR system. In this thesis, we conduct theoretical analysis of a well-established neural generative model (i.e., the paragraph vector model [58]) based on negative sampling [71]. We derive the closed-form expression of its learning objective and propose several techniques to adapt it for IR. We also discover that the norms of embedding vectors often reflect important information of the data or the optimization process. We further propose a couple of regularization techniques to control the learning process of neural generative models for its robustness and effectiveness in IR tasks.

6.2 Summary of Experimental Results

Following the steps described in Section 6.1, we show how to develop neural generative models for multiple types of IR problems including search on homogeneous information (i.e., ad-hoc retrieval) and search on heterogeneous information (i.e., product search). In this section, we summarize the key results of our experiments.

6.2.1 Ad-hoc Retrieval with Homogeneous Information

Table 6.1 and Table 6.2 summarize our experimental results on applying neural generative models (the paragraph vector model in this case) to ad-hoc retrieval.

Table 6.1: Test set results on Robust04 and GOV2 collection with topic titles as the search queries. Retrieval performance is measured by mean average precision (MAP), normalized discounted cumulative gains at 20 (NDCG@20), and precision at 20 (P@20). $*$, $+$ means significant difference over QL [80], LDA-LM [99] respectively at 0.05 significance level measured by Fisher randomization test.

	Robust04			GOV2		
Method	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
QL	0.253	0.415	0.369	0.295^{+}	0.409	0.510^{+}
LDA-LM	0.258^{*}	0.421	0.374^{*}	0.290	0.406	0.505
PV-LM	0.259^{*}	0.418	0.371	0.294	0.409	0.510^{+}
EPV-D-LM	0.260^{*}	0.417	0.371	0.295^{+}	0.410	0.511^{+}
EPV-DR-LM	0.262^{*}	0.418	0.368	0.296^{+}	0.412	0.512
EPV-DRJ-LM	**0.267**$^{*+}$	**0.425***	**0.376***	**0.297**$^{+}$	**0.415**$^{*+}$	**0.519**$^{*+}$

Table 6.2: Test set results on Robust04 and GOV2 collection with topic descriptions as the search queries. Retrieval performance is measured by mean average precision (MAP), normalized discounted cumulative gains at 20 (NDCG@20), and precision at 20 (P@20). $*$, $+$ means significant difference over QL [80], LDA-LM [99] respectively at 0.05 significance level measured by Fisher randomization test.

	Robust04			GOV2		
Method	MAP	nDCG@20	P@20	MAP	nDCG@20	P@20
QL	0.246	0.391	0.334	0.249^{+}	0.371	0.470
LDA-LM	0.247	0.392	0.336	0.245	**0.376**	0.468
PV-LM	0.247	0.392	0.335	0.246	0.364	0.463
EPV-R-LM	0.251^{*}	0.397^{*}	0.340^{*}	0.250^{+}	0.368	0.467
EPV-DR-LM	0.252^{*+}	0.397^{*}	0.338^{*}	0.250^{+}	0.371	0.470
EPV-DRJ-LM	**0.253**$^{*+}$	**0.404**$^{*+}$	**0.347**$^{*+}$	**0.252**$^{*+}$	0.371	**0.472**

In these tables, QL [80] and LDA-LM [99] represent two types of statistical generative retrieval models based on bag-of-words representations and latent semantic representations learned with topic models; PV-LM refers to the naive model that directly combines the statistical language modeling approach for IR with the original paragraph vector model proposed for NLP; and EPV-D-LM, EPV-DR-LM, and EPV-DRJ-LM represents three types of enhanced paragraph vector models for ad-hoc retrieval with our proposed Document-frequency based negative sampling strategy (D), vector norm Regularization (R), and Joint learning objectives (J) of word syntagmatic and paradigmatic relationships. Table 6.1 shows the performance of different models with short queries (i.e., topic titles) that only contain several words while Table 6.2 shows the performance of different models with long queries (i.e., topic descriptions) containing tens of words. In both cases, we observe that the naive combination of the paragraph vector model with the statistic language modeling approach for IR (i.e., PV-LM) performs similar or worse than the traditional topic modeling approach LDA-LM. However, with the proposed adaption techniques introduced in Chapter 3, all enhanced paragraph vector models achieve superior performance over LDA-LM in ad-hoc retrieval. This demonstrates the effectiveness of the proposed adaption techniques and the potential of neural generative models for IR.

6.2.2 Product Search with Heterogeneous Information

Table 6.3 shows the performance of different baselines and our proposed neural generative models – the hierarchical embedding model (HEM) and the dynamic relation embedding model (DREM) – in product search. The baselines include statistic generative retrieval models (QL [80] and BM25 [86]), learning-to-rank algorithms (LambdaMART) based on hand-crafted features [101], and the state-of-the-art neural embedding model (LSE) [97] for product search. As we can see in the table, HEM, which jointly learns the distributed representations of users, products, and queries

Table 6.3: Comparison of different product search baselines with the hierarchical embedding model (HEM) and the dynamic relation embedding model (DREM) on the Amazon search datasets with products from different categories. Mean average precision (MAP) and mean reciprocal rank (MRR) are computed with top 100 items, while normalized discounted cumulative gain (NDCG) is computed with top 10 items. $*$ and $+$ denote significant differences to all baselines (QL, BM25, LambdaMART, LSE) and all tested models, respectively, in Fisher randomization test [91] with $p \leq 0.01$. The best performance is highlighted in boldface.

	Electronics			*Kindle Store*		
Model	MAP	MRR	NDCG	MAP	MRR	NDCG
QL	0.289	0.289	0.316	0.011	0.012	0.013
BM25	0.283	0.280	0.304	0.021	0.013	0.014
LambdaMART	0.180	0.181	0.237	0.028	0.029	0.018
LSE	0.233	0.234	0.239	0.006	0.007	0.007
HEM	0.308*	0.309*	0.329*	0.029	0.035*	0.033*
DREM	**0.366**$^{*+}$	**0.367**$^{*+}$	**0.408**$^{*+}$	**0.057**$^{*+}$	**0.067**$^{*+}$	**0.067**$^{*+}$
	CDs & Vinyl			*Cell Phones & Accessories*		
Model	MAP	MRR	NDCG	MAP	MRR	NDCG
QL	0.009	0.011	0.010	0.081	0.081	0.092
BM25	0.027	0.018	0.016	0.083	0.081	0.115
LambdaMART	0.054	0.057	0.051	0.121	0.121	0.148
LSE	0.018	0.022	0.020	0.098	0.098	0.084
HEM	0.034	0.040	0.040	0.124*	0.124*	0.153*
DREM	**0.074**$^{*+}$	**0.084**$^{*+}$	**0.086**$^{*+}$	**0.249**$^{*+}$	**0.249**$^{*+}$	**0.282**$^{*+}$

with language data, significantly outperforms QL, BM25, and LSE in all cases. It also outperforms LambdaMART on *Electronics*, *Kindle Store*, and *Cell Phones & Accessories*. Note that LambdaMART is considered to be the state-of-the-art retrieval models for product search in both academia and industry. By incorporating more heterogeneous information with a generic relation embedding framework, DREM further outperforms HEM and all product search baselines with huge margins. This, again, indicates that the proposed neural generative representation learning framework has great potential for information retrieval in practice, especially for tasks that concern about heterogeneous information.

6.3 Future Work

While the study of neural generative models and representation learning for information retrieval is still in an early stage, we believe that this is a fruitful research direction. This thesis presents our initial effort towards the development of a generic neural generative framework for IR. There are many challenges in practice that we haven't discussed yet. Next, we briefly describe several research problems that we would like to study in the future.

- **Automatic Network Design**. Although designing neural models is relatively simple compared to traditional generative retrieval models based on statistical probabilistic analysis, it still requires significant mathematical background and deep understanding of the data. As more and more types of information are created on the Web, it becomes non-trivial to manually craft a data graph that can capture all types of information entities and their connections in practical IR systems. Therefore, how to construct a neural generative framework that can automatically discover and model the relationships between information is an important question for the IR community.

- **Theoretically Principled Search Result Explanation**. In Chapter 5, we propose a dynamic relation embedding model that simultaneously retrieves products and creates explanations of why the products should be interesting to the users. While the explanations generated by the proposed models are reasonable in many cases, they are not the direct reason why the products are retrieved. In other words, there are correlations but no causal relationship between the retrieval results and the result explanations in DREM. We are currently working on developing a theoretically principled search framework that unifies the retrieval process with result explanation generation so that users can see the exact reason why certain items are retrieved for their queries.

- **Efficient Retrieval with Neural Generative Models**. Low efficiency is a well-recognized problem of existing neural approaches for IR. Because most neural retrieval models, including the neural generative models introduced in this thesis, use dense vectors to represent semantics and information relationships, they are not eligible for efficient data organization techniques such as inverted indexing. While a couple of pre-computing and sparse encoding methods have been proposed to speed up the computation of dense matrix multiplications [37, 110], how to reduce the computation cost of neural generative models is still an open question. Breakthroughs in this direction could have important impacts on the design and applications of IR systems in practice.

BIBLIOGRAPHY

[1] Ai, Qingyao, Bi, Keping, Guo, Jiafeng, and Croft, W Bruce. Learning a deep listwise context model for ranking refinement. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018), ACM, pp. 135–144.

[2] Ai, Qingyao, OConnor, Brendan, and Croft, W Bruce. A neural passage model for ad-hoc document retrieval. In *European Conference on Information Retrieval* (2018), Springer, pp. 537–543.

[3] Ai, Qingyao, Yang, Liu, Guo, Jiafeng, and Croft, W Bruce. Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM on International Conference on the Theory of Information Retrieval* (2016), ACM, pp. 133–142.

[4] Ai, Qingyao, Yang, Liu, Guo, Jiafeng, and Croft, W. Bruce. Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *Proceedings of the 39th annual international ACM SIGIR conference on Research and development in information retrieval* (2016), ACM.

[5] Ai, Qingyao, Yang, Liu, Guo, Jiafeng, and Croft, W Bruce. Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval* (2016), ACM, pp. 869–872.

[6] Ai, Qingyao, Zhang, Yongfeng, Bi, Keping, Chen, Xu, and Croft, W Bruce. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference* (2017), ACM, pp. 645–654.

[7] Arora, Sanjeev, Li, Yuanzhi, Liang, Yingyu, Ma, Tengyu, and Risteski, Andrej. Rand-walk: A latent variable model approach to word embeddings. *arXiv preprint arXiv:1502.03520* (2015).

[8] Aryafar, Kamelia, Guillory, Devin, and Hong, Liangjie. An ensemble-based approach to click-through rate prediction for promoted listings at etsy. In *Proceedings of the ADKDD'17* (2017), ACM, p. 10.

[9] Baeza-Yates, Ricardo, Ribeiro-Neto, Berthier, et al. *Modern information retrieval*, vol. 463. ACM press New York, 1999.

[10] Bello, Irwan, Kulkarni, Sayali, Jain, Sagar, Boutilier, Craig, Chi, Ed, Eban, Elad, Luo, Xiyang, Mackey, Alan, and Meshi, Ofer. Seq2slate: Re-ranking and slate optimization with rnns. *arXiv preprint arXiv:1810.02019* (2018).

[11] Bendersky, Michael, Metzler, Donald, and Croft, W Bruce. Learning concept importance using a weighted dependence model. In *Proceedings of the third ACM international conference on Web search and data mining* (2010), ACM, pp. 31–40.

[12] Bendersky, Michael, Metzler, Donald, and Croft, W Bruce. Parameterized concept weighting in verbose queries. In *Proceedings of the 34th international ACM SIGIR conference on Research and development in Information Retrieval* (2011), ACM, pp. 605–614.

[13] Bendersky, Michael, Metzler, Donald, and Croft, W Bruce. Effective query formulation with multiple information sources. In *Proceedings of the fifth ACM international conference on Web search and data mining* (2012), ACM, pp. 443–452.

[14] Bengio, Yoshua, Ducharme, Réjean, Vincent, Pascal, and Jauvin, Christian. A neural probabilistic language model. *Journal of machine learning research 3*, Feb (2003), 1137–1155.

[15] Bilgic, Mustafa, and Mooney, Raymond J. Explaining recommendations: Satisfaction vs. promotion. In *Beyond Personalization Workshop, IUI* (2005), vol. 5, p. 153.

[16] Blei, David M., Ng, Andrew Y., and Jordan, Michael I. Latent dirichlet allocation. *J. Mach. Learn. Res. 3* (Mar. 2003), 993–1022.

[17] Blei, David M, Ng, Andrew Y, and Jordan, Michael I. Latent dirichlet allocation. *Journal of machine Learning research 3*, Jan (2003), 993–1022.

[18] Bordes, Antoine, Usunier, Nicolas, Garcia-Duran, Alberto, Weston, Jason, and Yakhnenko, Oksana. Translating embeddings for modeling multi-relational data. In *Advances in neural information processing systems* (2013), pp. 2787–2795.

[19] Bordes, Antoine, Weston, Jason, Collobert, Ronan, Bengio, Yoshua, et al. Learning structured embeddings of knowledge bases. In *AAAI* (2011), vol. 6, p. 6.

[20] Borisov, Alexey, Markov, Ilya, de Rijke, Maarten, and Serdyukov, Pavel. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web* (2016), International World Wide Web Conferences Steering Committee, pp. 531–541.

[21] Cheng, Heng-Tze, Koc, Levent, Harmsen, Jeremiah, Shaked, Tal, Chandra, Tushar, Aradhye, Hrishi, Anderson, Glen, Corrado, Greg, Chai, Wei, Ispir, Mustafa, et al. Wide & deep learning for recommender systems. In *Proceedings of the 1st Workshop on Deep Learning for Recommender Systems* (2016), ACM, pp. 7–10.

[22] Cho, Kyunghyun, Van Merriënboer, Bart, Gulcehre, Caglar, Bahdanau, Dzmitry, Bougares, Fethi, Schwenk, Holger, and Bengio, Yoshua. Learning phrase representations using rnn encoder-decoder for statistical machine translation. *arXiv preprint arXiv:1406.1078* (2014).

[23] Chow, C, and Liu, Cong. Approximating discrete probability distributions with dependence trees. *IEEE transactions on Information Theory 14*, 3 (1968), 462–467.

[24] Church, Kenneth W, and Gale, William A. Poisson mixtures. *Natural Language Engineering 1*, 02 (1995), 163–190.

[25] Cramer, Henriette, Evers, Vanessa, Ramlal, Satyan, Van Someren, Maarten, Rutledge, Lloyd, Stash, Natalia, Aroyo, Lora, and Wielinga, Bob. The effects of transparency on trust in and acceptance of a content-based art recommender. *User Modeling and User-Adapted Interaction 18*, 5 (2008), 455.

[26] Croft, W Bruce, and Lafferty, John. *Language modeling for information retrieval*, vol. 13. Springer Science & Business Media, 2013.

[27] Croft, W Bruce, Turtle, Howard R, and Lewis, David D. The use of phrases and structured queries in information retrieval. In *Proceedings of the 14th annual international ACM SIGIR conference on Research and development in information retrieval* (1991), ACM, pp. 32–45.

[28] Dai, Andrew M, Olah, Christopher, Le, Quoc V, and Corrado, Greg S. Document embedding with paragraph vectors. In *NIPS Deep Learning Workshop* (2014).

[29] Deerwester, Scott C., Dumais, Susan T, Landauer, Thomas K., Furnas, George W., and Harshman, Richard A. Indexing by latent semantic analysis. *JAsIs 41*, 6 (1990), 391–407.

[30] Dehghani, Mostafa, Zamani, Hamed, Severyn, Aliaksei, Kamps, Jaap, and Croft, W Bruce. Neural ranking models with weak supervision. *arXiv preprint arXiv:1704.08803* (2017).

[31] Devlin, Jacob, Chang, Ming-Wei, Lee, Kenton, and Toutanova, Kristina. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805* (2018).

[32] Diaz, Fernando, and Metzler, Donald. Improving the estimation of relevance models using large external corpora. In *Proceedings of the 29th annual international ACM SIGIR conference on Research and development in information retrieval* (2006), ACM, pp. 154–161.

[33] Dijkstra, Edsger W. A note on two problems in connexion with graphs. *Numerische mathematik 1*, 1 (1959), 269–271.

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

[35] Duchi, John, Hazan, Elad, and Singer, Yoram. Adaptive subgradient methods for online learning and stochastic optimization. *Journal of Machine Learning Research 12*, Jul (2011), 2121–2159.

[36] Fagan, Joel L. Automatic p hrase indexing for document retrieval: an examination of syntactic and non-syntactic methods. In *ACM SIGIR Forum* (2017), vol. 51, ACM, pp. 51–61.

[37] Fan, Yixing, Guo, Jiafeng, Lan, Yanyan, Xu, Jun, Pang, Liang, and Cheng, Xueqi. Learning visual features from snapshots for web search. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management* (2017), ACM, pp. 247–256.

[38] Fan, Yixing, Guo, Jiafeng, Lan, Yanyan, Xu, Jun, Zhai, Chengxiang, and Cheng, Xueqi. Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval* (2018), ACM, pp. 375–384.

[39] Gao, Jianfeng, Nie, Jian-Yun, Wu, Guangyuan, and Cao, Guihong. Dependence language model for information retrieval. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (2004), ACM, pp. 170–177.

[40] Griffiths, Thomas L., and Steyvers, Mark. Finding scientific topics. *PNAS 101*, suppl. 1 (2004), 5228–5235.

[41] Guo, Jiafeng, Fan, Yixing, Ai, Qingyao, and Croft, W Bruce. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management* (2016), ACM, pp. 55–64.

[42] Guo, Jiafeng, Fan, Yixing, Ai, Qingyao, and Croft, W Bruce. Semantic matching by non-linear word transportation for information retrieval. In *Proceedings of the 25th ACM International Conference on Information and Knowledge Management* (2016), ACM.

[43] Guo, Yangyang, Cheng, Zhiyong, Nie, Liqiang, Xu, Xin-Shun, and Kankanhalli, Mohan. Multi-modal preference modeling for product search. In *2018 ACM Multimedia Conference on Multimedia Conference* (2018), ACM, pp. 1865–1873.

[44] Harter, Stephen Paul. *A probabilistic approach to automatic keyword indexing.* PhD thesis, University of Chicago, 1974.

[45] He, Xiangnan, Liao, Lizi, Zhang, Hanwang, Nie, Liqiang, Hu, Xia, and Chua, Tat-Seng. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web* (2017), International World Wide Web Conferences Steering Committee, pp. 173–182.

[46] Herlocker, Jonathan L, Konstan, Joseph A, and Riedl, John. Explaining collaborative filtering recommendations. In *Proceedings of the 2000 ACM conference on Computer supported cooperative work* (2000), ACM, pp. 241–250.

[47] Hofmann, Thomas. Probabilistic latent semantic indexing. In *Proceedings of the 22nd annual international ACM SIGIR conference on Research and development in information retrieval* (1999), ACM, pp. 50–57.

[48] Hu, Yujing, Da, Qing, Zeng, Anxiang, Yu, Yang, and Xu, Yinghui. 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 &#38; Data Mining* (New York, NY, USA, 2018), KDD '18, ACM, pp. 368–377.

[49] Huang, Po-Sen, He, Xiaodong, Gao, Jianfeng, Deng, Li, Acero, Alex, and Heck, Larry. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Conference on information & knowledge management* (2013), ACM, pp. 2333–2338.

[50] Huston, Samuel, and Croft, W Bruce. A comparison of retrieval models using term dependencies. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management* (2014), ACM, pp. 111–120.

[51] Jeon, Jiwoon, Lavrenko, Victor, and Manmatha, Raghavan. Automatic image annotation and retrieval using cross-media relevance models. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), ACM, pp. 119–126.

[52] Karmaker Santu, Shubhra Kanti, Sondhi, Parikshit, and Zhai, ChengXiang. 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* (2017), ACM, pp. 475–484.

[53] Kim, Jin Young, and Croft, W Bruce. A field relevance model for structured document retrieval. In *European Conference on Information Retrieval* (2012), Springer, pp. 97–108.

[54] Kim, Jinyoung, Xue, Xiaobing, and Croft, W Bruce. A probabilistic retrieval model for semistructured data. In *European Conference on Information Retrieval* (2009), Springer, pp. 228–239.

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

[56] Krovetz, Robert. Viewing morphology as an inference process. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval* (1993), ACM, pp. 191–202.

[57] Lavrenko, Victor, and Croft, W Bruce. Relevance-based language models. In *ACM SIGIR Forum* (2017), vol. 51, ACM, pp. 260–267.

[58] Le, Quoc, and Mikolov, Tomas. Distributed representations of sentences and documents. In *International Conference on Machine Learning* (2014), pp. 1188–1196.

[59] Levy, Omer, and Goldberg, Yoav. Neural word embedding as implicit matrix factorization. In *Advances in Neural Information Processing Systems* (2014), pp. 2177–2185.

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

[61] Liu, Tie-Yan. Learning to rank for information retrieval. *Foundations and Trends in Information Retrieval 3*, 3 (2009), 225–331.

[62] Liu, Xiaoyong, and Croft, W Bruce. Cluster-based retrieval using language models. In *Proceedings of the 27th annual international ACM SIGIR conference on Research and development in information retrieval* (2004), ACM, pp. 186–193.

[63] Liu, Xiaoyong, and Croft, W Bruce. Statistical language modeling for information retrieval. Tech. rep., MASSACHUSETTS UNIV AMHERST CENTER FOR INTELLIGENT INFORMATION RETRIEVAL, 2005.

[64] Loyola, Pablo, Liu, Chen, and Hirate, Yu. Modeling user session and intent with an attention-based encoder-decoder architecture. In *Proceedings of the Eleventh ACM Conference on Recommender Systems* (2017), ACM, pp. 147–151.

[65] McAuley, Julian, Pandey, Rahul, and Leskovec, Jure. Inferring networks of substitutable and complementary products. In *Proceedings of the 21th ACM SIGKDD* (2015), ACM, pp. 785–794.

[66] McAuley, Julian, Targett, Christopher, Shi, Qinfeng, and van den Hengel, Anton. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th ACM SIGIR* (2015), ACM, pp. 43–52.

[67] McLachlan, Geoffrey, and Krishnan, Thriyambakam. *The EM algorithm and extensions*, vol. 382. John Wiley & Sons, 2007.

[68] Metzler, Donald, and Croft, W Bruce. A markov random field model for term dependencies. In *Proceedings of the 28th annual international ACM SIGIR conference on Research and development in information retrieval* (2005), ACM, pp. 472–479.

[69] Metzler, Donald, and Croft, W Bruce. Latent concept expansion using markov random fields. In *Proceedings of the 30th annual international ACM SIGIR conference on Research and development in information retrieval* (2007), ACM, pp. 311–318.

[70] Metzler Jr, Donald A. *Beyond bags of words: effectively modeling dependence and features in information retrieval.* University of Massachusetts Amherst, 2007.

[71] Mikolov, Tomas, Chen, Kai, Corrado, Greg, and Dean, Jeffrey. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781* (2013).

[72] Mikolov, Tomas, Sutskever, Ilya, Chen, Kai, Corrado, Greg S, and Dean, Jeff. Distributed representations of words and phrases and their compositionality. In *Advances in neural information processing systems* (2013), pp. 3111–3119.

[73] Mishne, Gilad, and De Rijke, Maarten. Boosting web retrieval through query operations. In *European Conference on Information Retrieval* (2005), Springer, pp. 502–516.

[74] Myaeng, Sung Hyon, Jang, Don-Hyun, Kim, Mun-Seok, and Zhoo, Zong-Cheol. A flexible model for retrieval of sgml documents. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (1998), ACM, pp. 138–145.

[75] Nallapati, Ramesh, and Allan, James. Capturing term dependencies using a language model based on sentence trees. In *Proceedings of the eleventh international conference on Information and knowledge management* (2002), ACM, pp. 383–390.

[76] Nurmi, Petteri, Lagerspetz, Eemil, Buntine, Wray, Floréen, Patrik, and Kukkonen, Joonas. Product retrieval for grocery stores. In *Proceedings of the 31st ACM SIGIR* (2008), ACM, pp. 781–782.

[77] Ogilvie, Paul, and Callan, Jamie. Combining document representations for known-item search. In *Proceedings of the 26th annual international ACM SIGIR conference on Research and development in informaion retrieval* (2003), ACM, pp. 143–150.

[78] Palangi, Hamid, Deng, Li, Shen, Yelong, Gao, Jianfeng, He, Xiaodong, Chen, Jianshu, Song, Xinying, and Ward, Rabab. 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.

[79] Piwowarski, Benjamin, and Gallinari, Patrick. A machine learning model for information retrieval with structured documents. In *International Workshop on Machine Learning and Data Mining in Pattern Recognition* (2003), Springer, pp. 425–438.

[80] Ponte, Jay M, and Croft, W Bruce. A language modeling approach to information retrieval. In *Proceedings of the 21st annual international ACM SIGIR conference on Research and development in information retrieval* (1998), ACM, pp. 275–281.

[81] Robertson, Stephen. Understanding inverse document frequency: on theoretical arguments for idf. *Journal of documentation 60*, 5 (2004), 503–520.

[82] Robertson, Stephen, Zaragoza, Hugo, and Taylor, Michael. Simple bm25 extension to multiple weighted fields. In *Proceedings of the thirteenth ACM international conference on Information and knowledge management* (2004), ACM, pp. 42–49.

[83] Robertson, Stephen E. The probability ranking principle in ir. *Journal of documentation 33*, 4 (1977), 294–304.

[84] Robertson, Stephen E, van Rijsbergen, Cornelis J, and Porter, Martin F. Probabilistic models of indexing and searching. In *Proceedings of the 3rd annual ACM conference on Research and development in information retrieval* (1980), Butterworth & Co., pp. 35–56.

[85] Robertson, Stephen E, and Walker, Steve. Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Proceedings of the 17th annual international ACM SIGIR conference on Research and development in information retrieval* (1994), Springer-Verlag New York, Inc., pp. 232–241.

[86] Robertson, Stephen E, Walker, Steve, Jones, Susan, Hancock-Beaulieu, Micheline M, Gatford, Mike, et al. Okapi at trec-3. *Nist Special Publication Sp 109* (1995), 109.

[87] Rowley, Jennifer. Product search in e-shopping: a review and research propositions. *Journal of consumer marketing 17*, 1 (2000), 20–35.

[88] Salton, Gerard, Wong, Anita, and Yang, Chung-Shu. A vector space model for automatic indexing. *Communications of the ACM 18*, 11 (1975), 613–620.

[89] Shen, Xuehua, Tan, Bin, and Zhai, ChengXiang. Context-sensitive information retrieval using implicit feedback. In *Proceedings of the 28th annual international ACM SIGIR conference* (2005), ACM, pp. 43–50.

[90] Shen, Yelong, He, Xiaodong, Gao, Jianfeng, Deng, Li, and Mesnil, Grégoire. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web* (2014), ACM, pp. 373–374.

[91] Smucker, Mark D, Allan, James, and Carterette, Ben. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management* (2007), ACM, pp. 623–632.

[92] Sun, Fei, Guo, Jiafeng, Lan, Yanyan, Xu, Jun, and Cheng, Xueqi. Learning word representations by jointly modeling syntagmatic and paradigmatic relations. In *Proceedings of the 53rd Annual Annual Meeting of the Association for Computational Linguistics* (2015).

[93] Suykens, Johan AK, and Vandewalle, Joos. Least squares support vector machine classifiers. *Neural processing letters 9*, 3 (1999), 293–300.

[94] Svore, Krysta M, and Burges, Christopher JC. A machine learning approach for improved bm25 retrieval. In *Proceedings of the 18th ACM conference on Information and knowledge management* (2009), ACM, pp. 1811–1814.

[95] Tintarev, Nava, and Masthoff, Judith. A survey of explanations in recommender systems. In *Data Engineering Workshop, 2007 IEEE 23rd International Conference on* (2007), IEEE, pp. 801–810.

[96] Tintarev, Nava, and Masthoff, Judith. Designing and evaluating explanations for recommender systems. *Recommender Systems Handbook* (2011), 479–510.

[97] Van Gysel, Christophe, de Rijke, Maarten, and Kanoulas, Evangelos. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM CIKM* (2016), ACM, pp. 165–174.

[98] Vulić, Ivan, and Moens, Marie-Francine. Monolingual and cross-lingual information retrieval models based on (bilingual) word embeddings. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval* (2015), ACM, pp. 363–372.

[99] Wei, Xing, and Croft, W. Bruce. Lda-based document models for ad-hoc retrieval. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval* (New York, NY, USA, 2006), SIGIR '06, ACM, pp. 178–185.

[100] Wilkinson, Ross. Effective retrieval of structured documents. In *SIGIR94* (1994), Springer, pp. 311–317.

[101] Wu, Chen, Yan, Ming, and Si, Luo. Ensemble methods for personalized e-commerce search challenge at cikm cup 2016. *arXiv preprint arXiv:1708.04479* (2017).

[102] Wu, Fei, Zhang, Hong, and Zhuang, Yueting. Learning semantic correlations for cross-media retrieval. In *Image Processing, 2006 IEEE International Conference on* (2006), IEEE, pp. 1465–1468.

[103] Wu, Liang, Hu, Diane, Hong, Liangjie, and Liu, Huan. Turning clicks into purchases: Revenue optimization for product search in e-commerce.

[104] Xiong, Chenyan, Dai, Zhuyun, Callan, Jamie, Liu, Zhiyuan, and Power, Russell. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th ACM SIGIR* (2017), ACM, pp. 55–64.

[105] Yang, Liu, Ai, Qingyao, Guo, Jiafeng, and Croft, W Bruce. anmm: Ranking short answer texts with attention-based neural matching model. In *Proceedings of the 25th ACM international on conference on information and knowledge management* (2016), ACM, pp. 287–296.

[106] Yang, Yi, Xu, Dong, Nie, Feiping, Luo, Jiebo, and Zhuang, Yueting. Ranking with local regression and global alignment for cross media retrieval. In *Proceedings of the 17th ACM international conference on Multimedia* (2009), ACM, pp. 175–184.

[107] Yang, Yi, Zhuang, Yue-Ting, Wu, Fei, and Pan, Yun-He. Harmonizing hierarchical manifolds for multimedia document semantics understanding and cross-media retrieval. *IEEE Transactions on Multimedia 10*, 3 (2008), 437–446.

[108] Yu, Jun, Mohan, Sunil, Putthividhya, Duangmanee Pew, and Wong, Weng-Keen. Latent dirichlet allocation based diversified retrieval for e-commerce search. In *Proceedings of the 7th ACM international conference on Web search and data mining* (2014), ACM, pp. 463–472.

[109] Zamani, Hamed, and Croft, W Bruce. Estimating embedding vectors for queries. In *Proceedings of the ACM ICTIR'16* (2016), ACM, pp. 123–132.

[110] Zamani, Hamed, Dehghani, Mostafa, Croft, W Bruce, Learned-Miller, Erik, and Kamps, Jaap. From neural re-ranking to neural ranking: Learning a sparse representation for inverted indexing. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management* (2018), ACM, pp. 497–506.

[111] Zhai, Chengxiang, and Lafferty, John. A study of smoothing methods for language models applied to ad hoc information retrieval. In *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval* (2001), ACM, pp. 334–342.

[112] Zhai, Chengxiang, and Lafferty, John. A study of smoothing methods for language models applied to information retrieval. *ACM Transactions on Information Systems (TOIS) 22*, 2 (2004), 179–214.

[113] Zhang, Fuzheng, Yuan, Nicholas Jing, Lian, Defu, Xie, Xing, and Ma, Wei-Ying. Collaborative knowledge base embedding for recommender systems. In *Proceedings of the 22nd ACM SIGKDD international conference on knowledge discovery and data mining* (2016), ACM, pp. 353–362.

[114] Zhang, Yongfeng, Ai, Qingyao, Chen, Xu, and Croft, W. Joint representation learning for top-n recommendation with heterogeneous information sources. *CIKM. ACM* (2017).

[115] Zhang, Yongfeng, Lai, Guokun, Zhang, Min, Zhang, Yi, Liu, Yiqun, and Ma, Shaoping. Explicit factor models for explainable recommendation based on phrase-level sentiment analysis. In *Proceedings of the 37th international ACM SIGIR conference* (2014), ACM, pp. 83–92.

[116] Zhao, Le, and Callan, Jamie. Term necessity prediction. In *Proceedings of the 19th ACM international conference on Information and knowledge management* (2010), ACM, pp. 259–268.

[117] Zhuang, Yue-Ting, Yang, Yi, and Wu, Fei. Mining semantic correlation of heterogeneous multimedia data for cross-media retrieval. *IEEE Transactions on Multimedia 10*, 2 (2008), 221–229.

[118] Zuccon, Guido, Koopman, Bevan, Bruza, Peter, and Azzopardi, Leif. Integrating and evaluating neural word embeddings in information retrieval. In *Proceedings of the 20th Australasian Document Computing Symposium* (2015), ACM, pp. Article–No.