NEURAL MODELS FOR INFORMATION RETRIEVAL WITHOUT LABELED DATA

A Dissertation Presented

by

HAMED ZAMANI

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 MODELS FOR INFORMATION RETRIEVAL WITHOUT LABELED DATA

A Dissertation Presented

by

HAMED ZAMANI

Approved as to style and content by:

W. Bruce Croft, Chair

James Allan, Member

Erik Learned-Miller, Member

Weibo Gong, Member

James Allan, Chair of the Faculty
College of Information and Computer Sciences

To my parents.

ACKNOWLEDGMENTS

First and foremost, I would like to thank my advisor and role model, Bruce Croft, without whom this dissertation would not have been possible. Bruce provided me with the perfect balance of guidance and freedom. Most importantly, he taught me how to be a scientist and make significant impact on the community. His immense support, visionary leadership, boundless intellectual curiosity, and constant enthusiasm had, and will continue to have, tremendous impact on my academic life. I am also grateful to him for placing his trust and confidence in my abilities and giving me the opportunity to teach the Information Retrieval course (CS646) at UMass, collaborate broadly, write research grants, and organize workshops.

I must also acknowledge my committee members, James Allan, Erik Learned-Miller, and Weibo Gong. Their insightful comments improved this dissertation in many ways. In particular, I would like to thank James for his thought-provoking feedback and detailed comments on the materials presented in this dissertation. I must also thank Erik for his invaluable feedback during our close collaboration.

It was a great privilege to be part of the Center for Intelligent Information Retrieval (CIIR). I would like to thank all of my fellow graduate students at the Center. In particular, I would like to thank Qingyao Ai, Dan Cohen, John Foley, Helia Hashemi, Jiepu Jiang, and Youngwoo Kim for their nearly endless supply of profound discussions. In addition, I am sincerely indebted to all the CIIR staff, for their dedicated support of my work. In particular, I would like to thank Stephen Harding, Jean Joyce, Kate Morruzzi, Dan Parker, and Glenn Stowell for their administrative and technical support.

I had the honor of collaborating with over 50 researchers so far, as either mentor, mentee, or collaborator. I have not only learned a lot from them, but also made many great friends. In particular, I would like to thank my Master's advisor, Azadeh Shakery, for introducing me to the field of information retrieval. I would also like to thank my internship mentors and collaborators at Google Research and Microsoft Research for giving me the opportunity to see what it was like to be a researcher in industrial research labs. They include Michael Bendersky, Paul Bennett, Nick Craswell, Susan Dumais, Gord Lueck, Bhaskar Mitra, Xia Song, Saurabh Tiwary, Xuanhui Wang, and Mingyang Zhang. I must also acknowledge the help and support of Nick Craswell, Mostafa Dehghani, Fernando Diaz, and Hang Li while organizing the first international workshop on learning from limited or noisy data for information retrieval (LND4IR) at SIGIR 2018. A special thanks to Mohammad Aliannejadi, Mostafa Dehghani, Jaap Kamps, and Markus Schedl for the productive collaborations we had throughout the last few years.

Last but not least, I would not be where I am today without the amazing support, encouragement and love from my family. I am forever indebted to my parents, Razieh and Mokhtar, for giving me the opportunities that have made me who I am. They selflessly encouraged me to explore new directions in life. This journey would not have been possible if not for them, and I dedicate this milestone to them. I would like to thank my brother and sister, Mahdi and Mahsa, for their unfailing emotional support. I am grateful to have them by my side. Finally, I would like to thank my girlfriend, Helia, for her love and care while I was finishing this dissertation. This paragraph is woefully inadequate to express the depth of support and love I received from all of them.

This work was supported in part by the Center for Intelligent Information Retrieval. 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 MODELS FOR INFORMATION RETRIEVAL WITHOUT LABELED DATA

SEPTEMBER 2019

HAMED ZAMANI

B.Sc., UNIVERSITY OF TEHRAN

M.Sc., UNIVERSITY OF TEHRAN

Ph.D., UNIVERSITY OF MASSACHUSETTS AMHERST

Directed by: Professor W. Bruce Croft

Recent developments of machine learning models, and in particular deep neural networks, have yielded significant improvements on several computer vision, natural language processing, and speech recognition tasks. Progress with information retrieval (IR) tasks has been slower, however, due to the lack of large-scale training data as well as neural network models specifically designed for effective information retrieval. In this dissertation, we address these two issues by introducing task-specific neural network architectures for a set of IR tasks and proposing novel unsupervised or *weakly supervised* solutions for training the models. The proposed learning solutions do not require labeled training data. Instead, in our weak supervision approach, neural models are trained on a large set of noisy and biased training data obtained from external resources, existing models, or heuristics.

We first introduce relevance-based embedding models that learn distributed representations for words and queries. We show that the learned representations can be effectively employed for a set of IR tasks, including query expansion, pseudo-relevance feedback, and query classification.

We further propose a standalone learning to rank model based on deep neural networks. Our model learns a sparse representation for queries and documents. This enables us to perform efficient retrieval by constructing an inverted index in the learned semantic space. Our model outperforms state-of-the-art retrieval models, while performing as efficiently as term matching retrieval models.

We additionally propose a neural network framework for predicting the performance of a retrieval model for a given query. Inspired by existing query performance prediction models, our framework integrates several information sources, such as retrieval score distribution and term distribution in the top retrieved documents. This leads to state-of-the-art results for the performance prediction task on various standard collections.

We finally bridge the gap between retrieval and recommendation models, as the two key components in most information systems. Search and recommendation often share the same goal: helping people get the information they need at the right time. Therefore, joint modeling and optimization of search engines and recommender systems could potentially benefit both systems. In more detail, we introduce a retrieval model that is trained using user-item interaction (e.g., recommendation data), with no need to query-document relevance information for training.

Our solutions and findings in this dissertation smooth the path towards learning efficient and effective models for various information retrieval and related tasks, especially when large-scale training data is not available.

TABLE OF CONTENTS

Page

CHAPTER

LIST OF TABLES

Table **Page**

LIST OF FIGURES

Figure **Page**

CHAPTER 1
INTRODUCTION

Information Retrieval (IR) is a field of science concerned with finding material of mostly unstructured nature to satisfy an information need [102].[1] Information retrieval technologies have impacted and are having increasing impact on people's everyday lives. They are a major source of revenue for the key players of the tech industry, such as Google[2] and Microsoft[3]. Search engines are probably the most successful applications of information retrieval. Effective search engines mostly rely on machine learning algorithms with carefully designed hand-crafted *features* trained on *large-scale data* collected from past user interactions with the search engine [91].

Manual feature engineering has several disadvantages. First it is expensive, in terms of both time and cost. Second, extracting effective complex features is difficult or sometimes infeasible. Third, effective features are mostly application- and task-specific, therefore, they must be carefully designed for each target application and task, separately. One goal of this dissertation is to omit the need for manual feature engineering. In more detail, we propose models based on the recent advances in deep learning to learn latent features instead of manual feature engineering. Recent deep learning techniques have shown impressive performance in various computer

[1] Information retrieval refers to a wide range of information seeking and organizing needs. Among them, text retrieval has been the focus of the IR community since 1950s [142]. Text retrieval focuses on helping people find documents that are relevant to their information needs. In this dissertation, all references to information retrieval will be to text retrieval systems.

[2] `https://www.google.com/`

[3] `https://www.microsoft.com/`

vision, natural language processing, and speech recognition tasks. However, deep neural networks require large-scale training data to perform well, which is not always available. The academic community and small companies lack such data, which prevents them from working on a set of cutting-edge technologies. Even within the major tech industry, there are situations where data is extremely sparse; these include enterprise systems, new or small international markets or demographics, and personal systems [187]. As a result, dealing with limited data is an unavoidable problem in both academia and industry. Therefore, a considerable portion of this dissertation proposes approaches to deal with limited training data in its extreme case, where there is *no* training data in hand.

Most approaches introduced in this dissertation are based on a *weak supervision* solution, a machine learning strategy that does not require labeled data. In more detail, our proposed models are mostly trained based on the output of an existing unsupervised model for the underlying task [42, 180]. In the following sections, we review a set of information retrieval tasks studied in this dissertation and briefly introduce our contributions.

1.1 Distributed Representation for Information Retrieval

Representation learning for terms,[4] queries, and documents is at the core of information retrieval. Assuming vocabulary terms being the atomic units of natural languages, most information retrieval models represent each piece of text based on the vocabulary terms it contains as well as their frequencies. For instance, in the vector space model [143], each text is represented by a $|V|$ dimensional vector, where V denotes the vocabulary set. Each dimension corresponds to a vocabulary term and the similarity between two vectors is often computed using dot product or co-

[4] Although there are some subtle differences between the meanings of “word” and “term”, they are sometimes used interchangeably throughout this dissertation.

sine similarity functions. Using one dimension per vocabulary term is called *local* representation.

In local representations, it is assumed that dimensions are independent, which is not a true assumption in the context of natural language. The reason is that there exist syntactic and semantic relations between terms in the vocabulary. This assumption in traditional retrieval models has led to a long-standing problem called *vocabulary mismatch* [30]. Vocabulary mismatch refers to a common phenomenon in natural language where different vocabulary terms are used for the same (or similar) concepts or entities. Vocabulary mismatch is particularly important for information retrieval, since the language usage in queries and documents is different.

Using *distributed* representation for text is a potential solution for addressing the vocabulary mismatch problem. Distributed representation refers to representing each atomic unit of language by a pattern of activity distributed over several dimensions in a vector space model [70]. This allows us to measure the similarity between any arbitrary pair of vocabulary terms, and consequently any pair of text.

Latent semantic indexing (LSI) [39] is an early model that uses distributed representation for information retrieval. LSI decomposes a document-term matrix computed from the collection using singular value decomposition (SVD) into three matrices. The latent representations learned by SVD are then considered as distributed representations of terms and documents. Despite its clever idea, LSI is expensive or infeasible to run on large collections [102]. More recently, with the rapid growth of computational resources, distributed representations have again attracted much attention. Among various approaches for learning distributed representation for terms, word2vec [107] and GloVe [123] are notable. They are also called *word embedding* models. These models are designed based on similar idea. Unlike LSI, they use a small window context (e.g., five terms) instead of the whole document.

In this dissertation, we focus on learning distributed representations for information retrieval. We argue that the objectives of existing word embedding models are based on term adjacency, which differs from the main objective in information retrieval. In fact, the main goal in information retrieval is to capture *relevance*. Therefore, we revisit the objective of word embedding models by predicting the terms appeared in the relevant documents in response to an information need. This has led to the development of *relevance-based word embedding* models [180]. It is notable that our models do not require labeled data. In more detail, we assume that the top retrieved documents in response to each query are relevant to the query and use relevance models [87], state-of-the-art pseudo-relevance feedback models, to automatically generate large-scale *weakly supervised data*.

To evaluate the effectiveness of word embedding models for information retrieval, we propose a set of extrinsic evaluation methodologies for the tasks of query expansion [178, 3] and query classification [179]. We show that relevance-based word embedding models outperform general-purpose word embedding models in these IR tasks.

1.2 Neural Ranking Models

Developing efficient and effective retrieval models has always been at the core of information retrieval [30]. Modern search engines use a multi-stage cascaded architecture for ranking documents in response to each query [170]. The early stage rankers are supposed to be efficient with high recall. They use an inverted index to efficiently retrieve documents from a large-scale collection. On the other hand, the last stage rankers are often expensive with high precision. They only re-rank a small subset of the collection [176]. It has been over a decade since *learning to rank* models are used as the last stage rankers in the search and recommendation industry [91].

To avoid the need for manual feature engineering, a set of learning to rank models based on deep neural networks has been recently proposed to learn representation for unstructured text [56, 108]. These models, called *neural ranking models*, often require large-scale training data to perform effectively. To address this issue, Dehghani et al. [42] have recently proposed a *weak supervision* approach that uses an existing retrieval model, e.g., BM25 [134], as a weak labeler to produce a large volume of training data.

In this dissertation, we theoretically study weak supervision in the context of learning to rank [182, 183]. We model weak supervision as a noisy channel that introduces noise to the true labels. Our study not only theoretically explains why weak supervision models perform well, but also provides guidelines for training weakly supervised learning to rank models. We further propose a *standalone neural ranking model* [186] that is trained with weak supervision with no labeled data. Unlike existing learning to rank models, including neural ranking models, our model can retrieve documents from a large-scale collection. In other words, our model does not require a multi-stage cascaded architecture. This enables us to train our model fully end-to-end. We show that our model outperforms competitive baselines, while performing as efficient as term matching models, such as query likelihood [126] and BM25 [134].

1.3 Neural Query Performance Prediction

Predicting the performance of a search engine for a given query is a fundamental and challenging task in information retrieval. The *query performance prediction* (QPP) task is defined as predicting the quality of a retrieval model for a given query, when neither explicit nor implicit relevance information is available [32]. Accurate and real-time performance predictors could potentially be used in triggering a specific action in the retrieval system, such as selecting an index traversal algorithm at query time [101], choosing the correct number of documents to process in a cascaded multistage retrieval system [34], choosing the most effective ranking function per query, selecting the best variant from multiple query reformulations, or requesting more information from users in cases of potential poor retrieval performance, particularly in conversational systems.

In this dissertation, we propose a general end-to-end framework based on neural networks for query performance prediction. Our framework, called NeuralQPP [184], consists of multiple components, each analyzing a distinct aspect useful for performance prediction. We implement NeuralQPP using three components and train it using *multiple* weak supervision signals. Our model provides state-of-the-art results for the query performance prediction task.

1.4 Joint Modeling of Search and Recommendation

A quarter-century has passed since Belkin and Croft [15] discussed the similarity and unique challenges of information retrieval and information filtering systems. They concluded that their underlying goals are essentially equivalent, and thus they are two sides of the same coin. Search and recommendation are nowadays two major components of information systems. For example, in e-commerce applications, such as Amazon[5] and eBay[6], improving both search and recommendation engines is essential for increasing user satisfactions. We believe that search engines and recommender systems seek the same goal: helping people get the information they need at the right time. Therefore, joint modeling and optimization of search engines and recommender systems, if possible, could potentially benefit both systems.

In this dissertation, we propose to use joint modeling and optimization of search and recommendation [181] in order to learn a retrieval model from recommendation

[5] `https://www.amazon.com/`

[6] `https://www.ebay.com/`

data (i.e., without labeled retrieval data). This is not only a theoretically interesting problem, but also useful in practice. From theoretical perspective, it is a challenging transfer learning problem that differs from typical domain adaptation tasks. From practical perspective, it can lead to the development of search functionality for information systems that only have a recommender system. Our multi-task learning model takes advantage of weakly supervised relevance-based word embedding in addition to user-item interaction data for learning item (document) representations.

1.5 Contributions

In this dissertation, we make the following contributions:

1. We introduce relevance-based word embedding and propose maximum likelihood maximization and relevance posterior estimation as two models for learning relevance-based word embedding vectors. We further propose a weak supervision approach for training relevance-based word embedding models without labeled data (Chapter 3).

2. We propose two embedding-based query expansion models as well as an embedding-based pseudo-relevance feedback model based on the language modeling framework. These models improve the retrieval performance by considering semantic information. They also provide a set of extrinsic evaluation methodologies for evaluating word embedding algorithms for information retrieval (Chapter 3).

3. We propose a model for estimating query embedding vectors from pre-trained word representations, based on maximum likelihood estimation. We show that average word embedding, which is a simple yet effective model for query representation, is a special case of our model. Our model can also incorporate the top documents retrieved for the query in order to estimate more accurate query representations. We use our query embedding vectors for query classification

as an extrinsic evaluation methodology to evaluate word embedding algorithms (Chapter 3).

4. We theoretically study learning to rank with weak supervision. Our study has led to the definition of symmetric ranking loss functions. We study a set of pairwise loss functions and prove that the pairwise L_1 and the hinge loss are symmetric. We theoretically and empirically show that learning to rank models trained with noisy or weak labels should use a symmetric ranking loss function (Chapter 4).

5. We propose an end-to-end standalone neural ranking model trained with weak supervision, without labeled data. Our model learns a sparse representation for each query and document. We use the learned high-dimensional sparse representations to construct an inverted index in the latent space. This has led to development of the first learning to rank model that can retrieve documents from large-scale collections as efficiently as term matching models. Leveraging pseudo-relevance feedback in the learned latent space has resulted in outperforming state-of-the-art baselines (Chapter 4).

6. We design a neural network architecture for the task of query performance prediction. Inspired by the existing performance prediction models, our model analyzes retrieval score distribution, term distribution in the top retrieved documents, and semantic coherence of the top documents. To train a robust and effective model, we propose a weak supervision solution by learning from multiple weak signals. To regularize our model, we propose a component dropout technique that improves the generalization of our model. Our model consistently performs better than state-of-the-art performance prediction baselines (Chapter 5).

7. We introduce a framework for joint modeling and optimization of search and recommendation. Without query-document relevance information for training, thanks to relevance-based word embedding, our model transfer knowledge from user-item interactions (i.e., recommendation data) to learn item representation suitable for information retrieval. The retrieval component is based on bag-of-words. Our model outperforms state-of-the-art retrieval baselines and performs comparably with competitive hybrid recommendation models, in terms of recommendation performance (Chapter 6).

CHAPTER 2
RELATED WORK

In this chapter, we review related work in two sections. We first explore neural network approaches to information retrieval and then briefly review machine learning techniques that do not require labeled training data, with a focus on weak supervision training.

2.1 Neural Approaches to Information Retrieval

Deep neural network methods have been recently applied to various information retrieval applications, including ad-hoc retrieval [42, 109, 172], web search [16, 188], question answering [26, 62, 146], product search [4, 181], personal search [90, 149, 177], mobile search [5, 6], and conversational search [7, 175], and recommender systems [145, 189]. In the following, we focus on two categories of models that are most related to this dissertation, including word embedding approaches applied to information retrieval and neural ranking models. For more information on neural models to other information retrieval tasks, we refer the readers to [108].

2.1.1 Word Embedding for Information Retrieval

Learning a semantic representation for text has been studied for many years. Latent semantic indexing (LSI) [39] is considered as an early work in this area that tries to map each text to a semantic space using singular value decomposition (SVD), a well-known matrix factorization algorithm. Subsequently, Clinchant and Perronnin [25] proposed Fisher Vector (FV), a document representation framework based on continuous word embeddings, which aggregates a non-linear mapping of word vectors into a document-level representation. However, relatively simple term matching retrieval models, such as BM25 [134] and query likelihood [126], often significantly outperform the models that are solely based on semantic matching.

More recently, the rapid advances in computational resources (especially GPUs) have led to the development of efficient word embedding algorithms that are much more effective than their classic alternatives in measuring semantic similarity between words. *Word embedding*, also known as distributed representation of words, refers to a set of machine learning algorithms that learn high-dimensional real-valued dense vector representation $\vec{w} \in \mathbb{R}^d$ for each vocabulary term w, where d denotes the embedding dimensionality. GloVe [123] and word2vec [107] are two well-known word embedding algorithms that learn embedding vectors based on the same idea, but using different machine learning techniques. The idea is that the words that often appear in similar contexts are similar to each other. Based on this idea, these algorithms try to accurately predict the adjacent word(s) given a word or a context (i.e., a few words appeared in the same context window). GloVe learns the word embedding vectors by decomposing a global word-word co-occurrence matrix, while word2vec is based on a feed-forward fully-connected neural network with a single hidden layer. Levy and Goldberg [89] further showed that neural word embedding models are theoretically equivalent to implicit matrix factorization models, under some constraints.

Recently, Rekabsaz et al. [131] proposed to exploit global context in word embeddings in order to avoid topic shifting. In more detail, they suggest using document-level term co-occurrence to measure term relatedness by post filtering the related terms with high word embedding similarities using various global context measures. In another study, Diaz et al. [45] suggested using local information to train word embedding vectors. Their approach is motivated by a widely known fact in the information retrieval literature, indicating that there is a big difference between the unigram distribution of words on sub-topics of a collection and the unigram distribution estimated from the whole collection [173]. In more detail, Diaz et al. proposed training word embedding vectors on the top retrieved documents for each query. However, this model, called local embedding, is not always practical in real-word applications, since the embedding vectors must be trained at query time.

Word embedding vectors have been successfully employed in several NLP and IR tasks. Kusner et al. [83] proposed word mover's distance (WMD), a function for calculating semantic distance between two documents, which measures the minimum traveling distance from the embedded vectors of individual words in one document to the other one. Zhou et al. [197] introduced an embedding-based method for question retrieval in the context of community question answering. Vulić and Moens [168] proposed learning bilingual word embedding vectors from document-aligned comparable corpora for the task of cross-lingual information retrieval (CLIR). Zheng and Callan [196] presented a supervised embedding-based technique to re-weight terms in the existing IR models, e.g., BM25. Based on the well-defined structure of the language modeling framework in information retrieval [126], a number of methods have been introduced to employ word embedding vectors within this framework in order to improve the performance in IR tasks. For instance, Ganguly et al. [51] considered the semantic similarities between vocabulary terms to smooth document language models. Zuccon et al. [200] further proposed to employ word embeddings within the translation model for IR. Their proposed method achieves comparable performance to the mutual information-based translation language models of Karimzadehgan and Zhai [78]. The main idea behind the methods proposed by Ganguly et al. and Zuccon et al. is similar. Both methods consider semantic similarity in computing the probability of terms in the documents. Apart from estimating document language models, ALMasri et al. [8] proposed a heuristic query expansion method based on word embedding similarities. Their method is a term-by-term expansion. Kuzi et al. [84] also employed word embedding for query expansion, both with and without pseudo-relevance feedback. Sordoni et al. [155] earlier proposed a query expansion method based on concept embedding. Their method is a supervised learning approach with the quantum entropy loss function that uses click-through data. All of these approaches are based on word embedding vectors learned based on term proximity information. PhraseFinder of Jing and Croft [76] is an early work that used term proximity information for query expansion.

Most recently, word embedding algorithms went beyond the bag-of-words representation and learn contextual representation for words and sentences. ELMo [125] and BERT [43] are two successful implementations of contextual word embeddings. The network architecture in ELMo is based on recurrent neural networks (RNNs), while BERT uses Transformers [165], which are solely based on attention. The efficiency of Transformers compared to RNNs enables us to train BERT on a large-scale corpus and achieve a better performance, compared to ELMo. BERT has been recently applied to information retrieval tasks, and led to significant improvements in passage retrieval [117, 119].

Chapter 3 revisits the objective in word embedding algorithms and proposes to learn relevance-based word embedding by predicted the distribution of terms in the documents relevant to a specific information need. It also introduces a set of effective and formally motivated query expansion and query classification techniques.

2.1.2 Neural Ranking Models

Learning to rank refers to a set of machine learning algorithms for producing a ranked list of objects [91]. Conventional learning to rank models range from linear [105] to tree-based [19] to support vectors machines [77] to neural networks [18]. These models require careful feature engineering, which significantly affects their performance.

Recent studies in the area of deep learning have enabled us to learn abstract representations from raw data, such as image or text. This not only avoids the troublesome and expensive feature engineering procedure, but also allows the model to learn complex task-specific features from the data. The learning to rank models based on such neural representation learning are called *neural ranking models*.[1]

Guo et al. [55] categorized neural ranking models to representation-focused and interaction-focused models. Since representation-focused models also use the interaction information for learning the representations, Dehghani et al. [42] suggested early combination and late combination as the two categories of neural ranking models. The early combination models are designed based on the interactions between query and document as the networks' input. For instance, the deep relevance matching model (DRMM) [55] takes histogram-based features as input, representing the interactions between query and document. DeepMatch [97] is another example that maps the input to a sequence of terms and computes the matching score using a feed-forward network. Dehghani et al. [42] also proposed simple pointwise and pairwise neural ranking models based on feed-forward networks. Their models use the average word embedding of query and document as the input of the network. Xiong et al. [172] proposed a neural ranking model, called KNRM, that applies different kernel functions to the query-document interaction matrix. Dai et al. [36] extended KNRM by considering soft matching of ngrams using convolutional neural networks and proposed the Conv-KNRM model that outperforms the original KNRM model.

The late combination models, on the other hand, separately learn a representation for query and document and then compute the relevance score using a matching function applied on the learned representations. DSSM [74] is an example of late

[1] All the learning to rank models based on neural networks are sometimes referred to as neural ranking models. However, we use the phrase “neural ranking models” for the models that do not require manual feature engineering.

combination models that learns representations using feed-forward networks and then uses cosine similarity as the matching function. DSSM was further extended by making use of convolutional neural networks, called C-DSSM [150].

Furthermore, Mitra et al. [109] combined both early and late combination ideas. Their duet model consists of two components: local and distributed. The local component takes the query-document term matching matrix and produces a retrieval score, while the distributed component learns distributed representations for both query and document and compares them. The final score is obtained by summing the local and distributed retrieval scores.

The aforementioned neural ranking models assume that each document is an unstructured text, however, documents in most applications, including web search, product search, and email search, contain multiple fields (i.e., semi-structured documents). Zamani et al. [188] proposed NRM-F, a neural model for ranking semi-structured documents. NRM-F was evaluated in the context of web search and has shown significant improvements compared to state-of-the-art learning to rank models. Liu et al. [95] enhanced document representation in neural ranking models by incorporating entity representations and utilizing knowledge graphs. Besides document representation, Zamani et al. [177] proposed a context-aware neural ranking model that takes contextual information into account for learning more accurate query representation. Their network is based on a deep and wide architecture. For more information on neural ranking models, we refer the readers to the recent survey by Guo et al. [56].

All the existing learning to rank algorithms, including neural ranking models, re-rank a small subset of documents retrieved by a set of efficient early stage rankers. In Chapter 4, we propose a standalone neural ranking model, that unlike existing learning to rank models, can efficiently retrieve documents from a large-scale collection.

2.2 Machine Learning without Labeled Data

Many machine learning algorithms are trained using labeled training data. Labels can be obtained using expert annotators, crowdsourcing, explicit feedback, or implicit feedback from user interactions with a system. As machine learning models get more complex, they generally require more labeled data to train. Deep neural networks nowadays consist of hundreds of millions of parameters, which necessitates the need for large-scale training data. However, collecting large-scale labeled data is time-consuming, expensive and sometimes infeasible. In this section, we briefly review a set of learning approaches that do not require labeled training data.

Unsupervised Learning: Unsupervised learning is a type of machine learning algorithms used to draw inferences from test data. Cluster analysis is a typical unsupervised learning task. Besides clustering, there exist a number of unsupervised dimensionality reduction methods, such as principal component analysis (PCA) [72, 122] and linear discriminant analysis (LDA) [50]. In the realm of neural networks, autoencoders are popular unsupervised learning models that are employed for representation learning and dimensionality reduction by input reconstruction as the learning objective [53]. Restricted Boltzmann machines can be also trained unsupervised [71]. Generative adversarial networks (GANs) [54] are a class of unsupervised machine learning algorithms, implemented by a system of two neural networks (i.e., generative and discriminative models) contesting with each other in a zero-sum game framework. GANs have recently shown promising results in a set of computer vision tasks [53]. Unsupervised learning models only identify commonalities in the data.

Self-Supervised Learning: Self supervision refers to a learning category in which models are trained to predict a part of the data that is withheld. Language modeling is a good example of self supervision that has been used in various natural language related tasks. General-purpose word embedding algorithms, such as word2vec [107] and GloVe [123], are also categorized as self-supervised learning approaches that predict a term given a context or vice versa. Self-supervised models also do not require manually labeled data; they automatically generate training data by withholding a part of the data.

Transfer Learning: Transfer learning refers to a category of models that transfer knowledge learned in one or more source tasks and use it to improve learning in a related target task. Transfer learning is particularly useful when the target task suffers from lack of large-scale training data. Research in transfer learning began in early 1990's [127] and has attracted increasing attention. A simple approach for transfer learning is to train a model using the source training data and use it as a pre-trained model for the target task. Multi-task learning is also a subcategory of transfer learning algorithms that simultaneously optimizes multiple objectives for different tasks.

Weakly Supervised Learning: In weak supervision, the objective is the same as in the supervised setting, however instead of a ground-truth labeled training set we have a set of noisy labels, called weak labels. Weak supervision is a broad concept that refers to learning from any inaccurate, noisy, or biased labels. In this dissertation, we refer to weak supervision as a technique that does not require manual labeled data or implicit feedback. Therefore, learning from crowdsourced data or click data collected from user interactions with the system does not consider as weak supervision in this dissertation. In fact, weak supervision refers to methods that can automatically generate large-scale training data. They can be based on heuristics, external resources, or existing models. This is a student-teacher learning, in which the teacher is an unsupervised weak labeler. This can be used to either improve efficiency [27] or effectiveness [42, 79]. Distant supervision is also a type of weak supervision that makes use of an existing resource, such as knowledge base, to generate training data.

Most of the approaches presented in this dissertation are based on weak supervision. We use existing models for a target task to generate large-scale training data and leverage it for training our neural network models.

2.3 Weak Supervision for Information Retrieval

Limited training data has been a perennial problem in information retrieval [187]. This has motivated researchers to explore building models using *pseudo-labels*. For example, pseudo-relevance feedback (PRF) [13, 31] assumes that the top retrieved documents in response to a given query are relevant to the query. Although this assumption does not necessarily hold, PRF has been proven to be effective in many retrieval settings [87, 135, 191]. Building pseudo-collections and simulated queries for various IR tasks could be considered as another set of approaches that tackle this issue [11, 14].

As widely known, deep neural networks often require a large volume of training data. Recently, weak supervision has been started to be explored in information retrieval. It was firstly proposed in SIGIR 2017 by Dehghani et al. [42] for training neural ranking models and by Zamani and Croft [180] for learning IR-specific word representations. Since then, weak supervision has been studied from various perspectives for different applications. Following Dehghani et al. [42], weak supervision has been explored for ranking tasks. For instance, Nie et al. [116] proposed a convolutional neural ranking model trained with weak supervision for ad-hoc retrieval. MacAvaney et al. [100] explored a number of different methods for generating weak supervision data using the content information. Weak supervision has been also explored for passage retrieval by Xu et al. [174]. Recently, Haddad and Ghosh [59] proposed a sampling strategy to reduce the training time in weakly supervised neural ranking models.

In addition to ranking, weak supervision has been employed for some other IR tasks. For example, Xiao et al. [171] proposed a weak supervision method for query re-writing in the context of e-commerce applications. Chaidaroon et al. [23] proposed a weak supervision solution for text semantic hashing. Learning knowledge graph representation [167] is another application addressed by weak supervision training.

There is a line of research for combining supervised and weakly supervised signals to learn more accurate models. Dehghani et al. [40, 41] proposed weighting methods for controlling the learning rate of each mini-batch during training. Luo et al. [98] proposed to use click data for training weakly supervised ranking models. In addition, Shnarch et al. [151] proposed a method for blending weak supervision and strong supervision data for training a model for argumentation mining. Recent data programming frameworks, such as Snorkel [130],[2] have smoothed the path for generating weak supervision data for a wide variety of tasks.

In this dissertation, we show that weak supervision is a general solution for many information retrieval tasks. For example. we provide successful implementation of weakly supervised models for representation learning (Chapter 3), ranking (Chapter 4), and query performance prediction (Chapter 5). In addition, we provide a theoretical study for weak supervision training (Chapter 4). Learning from multiple weak supervision signals have been also explored in the dissertation (Chapter 5).

[2] `https://hazyresearch.github.io/snorkel/`

CHAPTER 3

DISTRIBUTED REPRESENTATIONS FOR INFORMATION RETRIEVAL

Information retrieval is mainly concerned with retrieving relevant documents from a large document collection in response to a query. This requires developing computational representations for queries and documents. The quality of these representations directly impacts the retrieval performance. This is why building useful query and document representations has been always at the core of information retrieval research. Learning IR-specific representations often require query-document relevance signal. In this chapter, we mainly focus on representation learning for words as the atomic units of natural languages. We propose relevance-based word embedding and discuss how to train the model using *weak supervision*. In fact, we use Lavrenko and Croft's relevance models [87] to automatically generate training data for our model.

This chapter is structured as follows: Section 3.1 briefly reviews word embedding models. Section 3.2 introduces our models for learning IR-specific distributed representations. Section 3.3 studies the application of word embedding in a number of IR tasks.

3.1 Background: Word Embedding

Learning a semantic representation for text has been studied for many years. Latent semantic indexing (LSI) [39] can be considered as early work in this area that tries to map each text to a semantic space using singular value decomposition (SVD), a well-known matrix factorization algorithm. Subsequently, Clinchant and

Perronnin [25] proposed Fisher Vector (FV), a document representation framework based on continuous word embeddings, which aggregates a non-linear mapping of word vectors into a document-level representation. However, a number of popular IR models, such as BM25 and language models, often significantly outperform the models that are based on semantic similarities. More recently, efficient algorithms for learning word representations have been proposed that model semantic similarly between words, effectively.

Word embedding, also known as distributed representation of words, refers to a set of machine learning algorithms that learn high-dimensional real-valued dense vector representation $\vec{w} \in \mathbb{R}^d$ for each vocabulary term w, where d denotes the embedding dimensionality. GloVe [123] and word2vec [107] are two well-known word embedding algorithms that learn embedding vectors based on the same idea, but using different machine learning techniques. The idea is that the words that often appear in similar contexts are similar to each other. These algorithms try to accurately predict the adjacent word(s) given a word or a context (i.e., a few words appeared in the same context window). The word2vec algorithm uses a fully-connected feed-forward network with a single hidden layer to learn distributed representation of words. On the other hand, GloVe employs a matrix factorization algorithm to decompose a global word-word co-occurrence matrix to two lower ranked matrices. Both of these two models have shown promising results in a set of natural language processing tasks.

3.2 Relevance-based Word Embedding

As introduced earlier in Section 3.1, typical word embedding algorithms, such as word2vec [107] and GloVe [123], learn high-dimensional real-valued embedding vectors based on the proximity of terms in a training corpus, i.e., co-occurrence of terms in the same context window. Although these approaches could be useful for learning the embedding vectors that can capture semantic and syntactic similarities between vocabulary terms and have shown to be useful in many NLP and IR tasks, there is a large gap between their learning objective (i.e., term proximity) and what is needed in many information retrieval tasks. For example, consider the query expansion task and assume that a user submitted the query “dangerous vehicles”. One of the most similar terms to this query based on the typical word embedding algorithms (e.g., word2vec and GloVe) is “safe”, and thus it would get a high weight in the expanded query model. The reason is that the words “dangerous” and “safe” often share similar contexts. However, expanding the query with the word “safe” could lead to poor retrieval performance, since it changes the meaning and the intent of the query.

This example together with many others have motivated us to revisit the objective used in the learning process of word embedding algorithms in order to obtain the word vectors that better match with the needs in IR tasks. The primary objective in many IR tasks is to model the notion of *relevance*. Several approaches, such as the relevance models proposed by Lavrenko and Croft [87], have been proposed to model relevance. Given the successes achieved by these models, we propose to revisit the objective for learning word embedding vectors as follows:

The objective is to accurately predict the terms that are observed in a set of documents relevant to a particular information need.

In the following, we first describe our neural network architecture, and then explain how to build a training set for learning relevance-based word embeddings. We further introduce two models, relevance likelihood maximization (RLM) and relevance posterior estimation (RPE), with different objectives using the described neural network.

3.2.1 Neural Network Architecture

We use a simple yet effective feed-forward neural network with a single linear hidden layer. The architecture of our neural network is similar to that of the word2vec

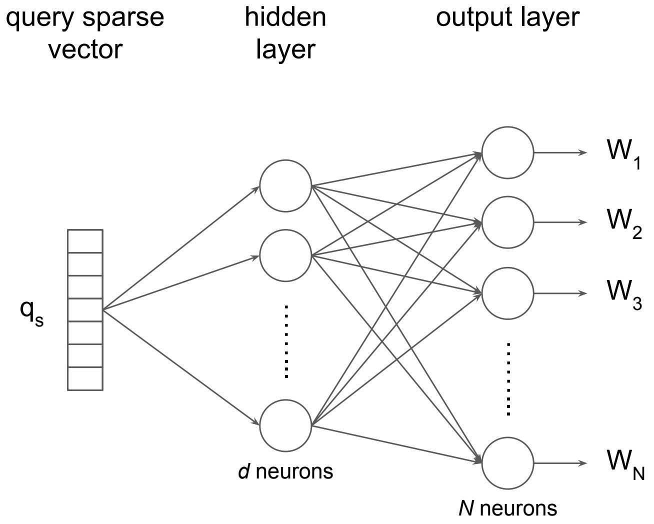


Figure 3.1: The relevance-based word embedding architecture. The objective is to learn d-dimensional distributed representation for words based on the notion of relevance, instead of term proximity. N denotes the total number of vocabulary terms.

model [107] and is depicted in Figure 3.1. The input of the model is a sparse query vector $\vec{q_s}$ with the length of N, where N denotes the total number of vocabulary terms. This vector can be obtained by a projection function given the vectors corresponding to individual query terms. In our model, we simply consider average as the projection function. Hence, $\vec{q_s} = \frac{1}{|q|}\sum_{w \in q} \vec{e_w}$, where $\vec{e_w}$ and $|q|$ denote the one-hot vector representation of term w and the query length, respectively. The hidden layer in this network maps the given query sparse vector to a query embedding vector $\vec{q}$, as follows:

$$\vec{q} = \vec{q_s} \times \mathcal{W}_Q \tag{3.1}$$

where $\mathcal{W}_Q \in \mathbb{R}^{N \times d}$ is a weight matrix for estimating query embedding vectors and d denotes the embedding dimensionality. The output layer of the network is a fully-connected layer given by:

$$\sigma(\vec{q} \times \mathcal{W}_w + b_w) \tag{3.2}$$

where $\mathcal{W}_w \in \mathbb{R}^{d \times N}$ and $b_w \in \mathbb{R}^{1 \times N}$ are the weight and the bias matrices for estimating the probability of each term. σ is the activation function which is discussed in Sections 3.2.3 and 3.2.4.

To summarize, our network contains two sets of embedding parameters, $\mathcal{W}_Q$ and $\mathcal{W}_w$. The former aims to map the query into the "query embedding space", while the latter is used to estimate the weights of individual terms.

3.2.2 Modeling Relevance for Training without Labeled Data

Relevance feedback has been shown to be highly effective in improving retrieval performance [135, 30]. In relevance feedback, a set of documents relevant to a given query is considered for estimating accurate query models. Since explicit relevance signals for a given query are not always available, pseudo-relevance feedback (PRF) assumes that the top retrieved documents in response to the given query are relevant to the query and uses these documents in order to estimate better query models. The effectiveness of PRF in various retrieval scenarios indicates that useful information can be captured from the top retrieved documents [87, 86, 191]. We make use of this well-known assumption to train our model. It should be noted that there is a significant difference between PRF and the proposed models: In PRF, the feedback model is estimated from the top retrieved documents of the given query in an online setting. In other words, PRF retrieves the documents for the initial query and then estimates the feedback model using the top retrieved documents. We propose to train the model in an offline setting. Moving from the online to the offline setting would lead to substantial improvements in efficiency, because an extra retrieval run is not required in the offline setting. To learn a model in an offline setting, we consider a fixed-length dense vector for each vocabulary term and estimate these vectors based on the information extracted from the top retrieved documents for large numbers of training queries. Note that our models do not require labeled training data. However, if explicit or implicit relevance signals, such as click data, are available, without loss of generality, they can be employed for training the models.

To formally describe our training data, let $T = \{(q_1, \mathcal{R}_1), (q_2, \mathcal{R}_2), \cdots, (q_m, \mathcal{R}_m)\}$ be a training set with m training queries. The i^{th} element of this set is a pair of query q_i and the corresponding pseudo-relevance feedback distribution. These distributions are estimated based on the top k retrieved documents (in our experiments, we set k to 10) for each query. The distributions can be estimated using any PRF model, such as those proposed in [87, 157, 185, 191]. In our model, we focus on a variant of the relevance models (i.e., RM3) [2, 87] that estimates the relevance distribution as:

$$p(w|\mathcal{R}_i) \propto \sum_{d \in F_i} p(w|d) \prod_{w' \in q_i} p(w'|d) \tag{3.3}$$

where F_i denotes a set of top retrieved documents for query q_i. Note that the probability of terms that do not appear in the top retrieved documents is equal to zero.

3.2.3 Relevance Likelihood Maximization Model

In this model, the goal is to learn the relevance distribution $\mathcal{R}$. Given a set of training data, we aim to find a set of parameters $\theta_{\mathcal{R}}$ in order to maximize the likelihood of generating relevance model probabilities for the whole training set. The likelihood function is defined as follows:

$$\prod_{i=1}^{m} \prod_{w \in V_i} \widehat{p}(w|q_i; \theta_{\mathcal{R}})^{p(w|\mathcal{R}_i)} \tag{3.4}$$

where $\widehat{p}$ is the relevance distribution that can be obtained given the learning parameters $\theta_{\mathcal{R}}$ and $p(w|\mathcal{R}_i)$ denotes the relevance model distribution estimated for the i^{th} query in the training set (see Section 3.2.2 for more detail). V_i denotes a subset of vocabulary terms that appeared in the top ranked documents retrieved for the query q_i. The reason for iterating over the terms that appeared in this set instead of the whole vocabulary set V is that the probability $p(w|\mathcal{R}_i)$ is equal to zero for all terms $w \in V - V_i$.

In this method, we model the probability distribution $\widehat{p}$ using the softmax function (i.e., the function σ in Equation (3.2)) as follows:

$$\widehat{p}(w|q; \theta_{\mathcal{R}}) = \frac{\exp(\vec{w}^T \vec{q})}{\sum_{w' \in V} \exp(\vec{w'}^T \vec{q})} \tag{3.5}$$

where $\vec{w}$ denotes the learned embedding vector for term w and $\vec{q}$ is the query vector came from the output of the hidden layer in our network (see Section 3.2.1). According to the softmax modeling and the log-likelihood function, we have the following objective:

$$\arg\max_{\theta_{\mathcal{R}}} \sum_{i=1}^{m} \sum_{w \in V_i} p(w|\mathcal{R}_i) \left(\log \exp(\vec{w}^T \vec{q_i}) - \log \sum_{w' \in V} \exp(\vec{w'}^T \vec{q_i}) \right) \tag{3.6}$$

Computing this objective function and its derivatives would be computationally expensive (due to the presence of the normalization factor $\sum_{w' \in V} \exp(\vec{w'}^T \vec{q})$ in the objective function). Since all the word embedding vectors as well as the query vector are changed during the optimization process, we cannot simply omit the normalization term as is done in [179] for estimating query embedding vectors based on pre-trained word embedding vectors. To make the computations more tractable, we consider a hierarchical approximation of the softmax function, which was introduced by Morin and Bengio [112] in the context of neural network language models and then successfully employed by Mikolov et al. [107] in the word2vec model.

The hierarchical softmax approximation uses a binary tree structure to represent the vocabulary terms, where each leaf corresponds to a unique word. There exists a unique path from the root to each leaf, and this path is used for estimating the probability of the word representing by the leaf. Therefore, the complexity of calculating softmax probabilities goes down from $O(|V|)$ to $O(\log(|V|))$ which is the height of the tree. This leads to a huge improvement in computational complexity. We refer the reader to [110, 112] for the details of calculating the hierarchical softmax approximation.

3.2.4 Relevance Posterior Estimation Model

As an alternative to maximum likelihood estimation, we can estimate the relevance posterior probability. In the context of pseudo-relevance feedback, Zhai and Laffery [191] assumed that the language model of the top retrieved documents is estimated based on a mixture model. In other words, it is assumed that there are two language models for the feedback set: the relevance language model[1] and a background noisy language model. They used an expectation-maximization algorithm to estimate the relevance language model. In this model, we make use of this assumption in order to cast the problem of estimating the relevance distribution $\mathcal{R}$ as a classification task: *Given a pair of word w and query q, does w come from the relevance distribution of the query q?* Instead of $p(w|\mathcal{R})$, this model estimates $p(R=1|w,q;\theta_{\mathcal{R}})$ where R is a Boolean variable and $R=1$ means that the given term-query pair (w,q) comes from the relevance distribution $\mathcal{R}$. $\theta_{\mathcal{R}}$ is a set of parameters that is going to be learned during the training phase.

Therefore, the problem is cast as a binary classification task that can be modeled by logistic regression (which means the function σ in Equation (3.2) is the sigmoid function):

$$\widehat{p}(R=1|\vec{w},\vec{q};\theta_{\mathcal{R}}) = \frac{1}{1+e^{(-\vec{w}^T\vec{q})}} \tag{3.7}$$

where $\vec{w}$ is the relevance-based word embedding vector for term w. Similar to the previous model, $\vec{q}$ is the output of the hidden layer of the network, representing the query embedding vector.

[1] The phrase “topical language model” was used in the original work [191]. We call it “relevance language model” to have consistent definitions in our both models.

In order to address this binary classification problem, we consider a cross-entropy loss function. In theory, for each training query, our model should learn to model relevance for the terms appearing in the corresponding pseudo-relevant set and non-relevance for all the other vocabulary terms, which could be impractical, due to the large number of vocabulary terms. Similar to [107], we propose to use the noise contrastive estimation (NCE) [58] which hypothesizes that we can achieve a good model by only differentiating the data from noise via a logistic regression model. The main concept in NCE is similar to those proposed in the divergence from randomness model [9] and the divergence minimization feedback model [191]. Based on the NCE hypothesis, we define the following negative cross-entropy objective function for training our model:

$$\begin{aligned} \arg\max_{\theta_{\mathcal{R}}} \sum_{i=1}^{m} \Bigg[& \sum_{j=1}^{\eta^+} \mathbb{E}_{w_j \sim p(w|\mathcal{R}_i)} \left[\log \widehat{p}(R = 1 | \vec{w_j}, \vec{q_i}; \theta_{\mathcal{R}}) \right] \\ & + \sum_{j=1}^{\eta^-} \mathbb{E}_{w_j \sim p_n(w)} \left[\log \widehat{p}(R = 0 | \vec{w_j}, \vec{q_i}; \theta_{\mathcal{R}}) \right] \Bigg] \end{aligned} \tag{3.8}$$

where $p_n(w)$ denotes a noise distribution and $\eta = (\eta^+, \eta^-)$ is a pair of hyper-parameters to control the number of positive and negative instances per query, respectively. We can easily calculate $\widehat{p}(R = 0|\vec{w_j}, \vec{q_i}) = 1 - \widehat{p}(R = 1|\vec{w_j}, \vec{q_i})$. The noise distribution $p_n(w)$ can be estimated using a function of unigram distribution $U(w)$ in the whole training set. Similar to [107], we use $p_n(w) \propto U(w)^{3/4}$ which has been empirically shown to work effectively for negative sampling.

It is notable that although this model learns embedding vectors for both queries and words, it is not obvious how to calculate the probability of each term given a query; because Equation (3.7) only gives us a classification probability and we cannot simply use the Bayes rule here (since, not all probability components are known). This model can perform well when computing the similarity between two terms or two queries, but not a query and a term. However, we can use the model presented in [179] to estimate the query model using the word embedding vectors (not the ones learned for query vectors) and then calculate the similarity between a query and a term.

3.3 Applications to Information Retrieval

Word embedding vectors enable us to compute the similarity of any two words in the vocabulary that exist in the training set. Since most IR models rely on exact term matching, these vectors can potentially be useful in a number of IR tasks. Sections 3.3.1 and 3.3.2 study the effectiveness of word embedding vectors in query expansion and query classification tasks, respectively.

3.3.1 Query Expansion

Vocabulary mismatch, i.e., the mismatch of different vocabulary terms that refer to the same concept, is a long-standing problem in information retrieval. For instance, in the language modeling framework [126], the retrieval score of a query-document pair is often computed using negative KL-divergence as follows [85]:

$$\text{score}(Q, D) = -D(\theta_Q || \theta_D) = -\sum_{w \in q} p(w|\theta_Q) \log \frac{p(w|\theta_Q)}{p(w|\theta_D)} \tag{3.9}$$

where θ_Q and θ_D denote the query and the document language models, respectively. The query language model is often estimated using maximum likelihood estimation, while in the document language model the maximum likelihood estimation is often smoothed with the collection language model [192]. This addresses the zero probability problem for the query terms not appearing in the document. As formulated above, the retrieval score is only computed for the words in the query, which shows that vocabulary mismatch affects the retrieval score. Query expansion is a well-known technique to alleviate this problem. In query expansion, the goal is to estimate a more accurate query model (i.e., θ_Q) by adding new terms to the query, alongside their weights. In the following, we explain how word embedding vectors can be used for query expansion.

3.3.1.1 Embedding-based Query Expansion Models

We propose two novel estimations for the query language models by making use of semantic similarity between the terms coming from the similarity between the word embedding vectors. These two estimations have different simplifying assumptions. The first method considers that there is a conditional independence assumption between the query terms; while the second method assumes that the semantic similarity between two given terms is independent of the query. Interestingly, as we will see in the following, the first assumption leads to an expansion method based on multiplicative similarity, i.e., the expanded terms should be similar to all query terms. On the other hand, the second assumption leads to an additive similarity (a mixture model).

EQE1: Conditional Independence of Query Terms: To estimate $p(w|\theta_Q)$, we first consider the Bayes rule as follows:

$$p(w|\theta_Q) = \frac{p(\theta_Q|w)p(w)}{p(Q)} \propto p(\theta_Q|w)p(w) \tag{3.10}$$

In the above equation, we ignore $p(Q)$ since it is independent of the term w. We assume that query terms are independent of each other, but we keep their dependence on w. Therefore, we can estimate the query language model as follows:

$$p(w|\theta_Q) \propto p(q_1, q_2, \ldots, q_k|w)p(w) = p(w)\prod_{i=1}^{k} p(q_i|w) \tag{3.11}$$

where $q_1, q_2, \ldots, q_k$ are the query terms and k is the query length. To compute $p(q_i|w)$, we consider the word embedding similarities which can capture the semantic similarity between vocabulary terms. To do so, we compute this probability as follows:

$$p(q_i|w) = \frac{\delta(q_i, w)}{\sum_{w' \in V} \delta(w', w)} \tag{3.12}$$

where δ and V denote the similarity function (e.g., the softmax function over the dot product similarity) and the vocabulary set, respectively. Considering the law of total probability, we compute $p(w)$ using the following equation:[2]

$$p(w) = \sum_{w' \in V} p(w, w') \propto \sum_{w' \in V} \delta(w, w') \tag{3.13}$$

The generated language model θ_Q can be linearly interpolated with the maximum likelihood estimation of the query language model with the coefficient of α.

EQE2: Query-Independent Term Similarities: To estimate the query language model θ_Q, we first use the law of total probability as follows:

$$p(w|\theta_Q) = \sum_{w' \in V} p(w, w'|\theta_Q) = \sum_{w' \in V} p(w|w', \theta_Q)p(w'|\theta_Q) \tag{3.14}$$

In the above calculations, we use the Bayes rule. We assume that the semantic similarity between the terms is independent of the query language model. Therefore, $p(w|w', \theta_Q)$ can be computed as follows:

$$p(w|w', \theta_Q) = p(w|w') = \frac{\delta(w, w')}{\sum_{w'' \in V} \delta(w'', w')} \tag{3.15}$$

where δ denotes the semantic similarity between two given terms and can be calculated by transforming the cosine similarity values using the sigmoid function.

[2] Intuitively, a word with higher semantic similarity with all the other words will achieve higher probability. In other words, general terms are supposed to have high probabilities according to this definition, which is also consistent with the other definitions for $p(w)$, such as the frequency ratio of the word w in a large corpus.

Considering the maximum likelihood estimation, we can rewrite Equation 3.14 as follows:

$$p(w|\theta_Q) \propto \sum_{w' \in Q} \frac{\delta(w, w')}{\sum_{w'' \in V} \delta(w'', w')} \times \frac{c(w', Q)}{|Q|} \tag{3.16}$$

where $c(w', Q)$ and $|Q|$ denote the count of term w' in the query and the query length, respectively.

Similar to the previous embedding-based estimation of query models, the generated query language model can be linearly interpolated with the maximum likelihood estimation of the query language model with a coefficient of α.

3.3.1.2 Embedding-based Relevance Models

Pseudo-relevance feedback has been shown to be highly effective in improving the retrieval performance [10, 87, 99, 191]. In PRF, it is assumed that the top-retrieved documents are relevant to the query, and thus they can be used to improve the query language model accuracy. We propose an embedding-based relevance model (ERM), a method inspired by the relevance model approach proposed by Lavrenko and Croft [87], which has been shown to be one of the most effective and robust PRF methods [99].

We compute the feedback language model as follows:

$$p(w|\theta_F) \propto \sum_{D \in F} p(w, Q, D) = \sum_{D \in F} p(Q|w, D) p(w|D) p(D) \tag{3.17}$$

where θ_F and F respectively denote the feedback language model and the set of feedback documents, i.e., the top-retrieved documents. To compute $p(Q|w, D)$, we consider both term matching and semantic similarities. This probability can be computed via a linear interpolation with the coefficient of β:

$$p(Q|w, D) = \beta \; p_{tm}(Q|w, D) + (1 - \beta) \; p_{sem}(Q|w, D) \tag{3.18}$$

where p_{tm} and p_{sem} denote the probabilities coming from the term matching similarities and the semantic similarities, respectively. Similar to RM3 [2, 87], $p_{tm}(Q|w, D)$ can be estimated by considering the independence assumption of terms (query terms and w) as follows:

$$p_{tm}(Q|w, D) = \prod_{i=1}^{k} p(q_i|D) \tag{3.19}$$

where q_i denotes the i^{th} term of the query Q with the length of k. To compute $p_{sem}(Q|w, D)$, we assume that query terms are independent of each other, but we keep their dependence to the term w and document D. Therefore, we can calculate this probability as follows:

$$p_{sem}(Q|w, D) = \prod_{i=1}^{k} p_{sem}(q_i|w, D) \triangleq \prod_{i=1}^{k} \frac{\delta(q_i, w)c(q_i, D)}{Z} \tag{3.20}$$

where δ computes the semantic similarity between two given terms and $c(q_i, D)$ is the count of term q_i in the document D. Z is a normalization factor, which is only needed to be a summation over the terms appeared in the document D (instead of all vocabulary terms), and thus it is not computationally expensive.

Similar to RM3, we compute $p(w|D)$ (see Equation (3.17)) using the maximum likelihood estimation (MLE) smoothed by a reference language model.[3] We assume that there is no prior knowledge about the pseudo-relevant documents, and thus we calculate $p(D)$ using a uniform distribution. It is notable that the proposed embedding-based relevance model satisfies the "semantic effect" constraint recently proposed by Montazeralghaem et al. [111]. The updated query language model can be calculated using the linear interpolation of the original query language model with the computed feedback language model:

[3] We also considered embedding-based semantic similarities in computing this probability using the law of total probability ($p(w|D) = \sum_{w' \in V} p(w, w'|D)$), but there is no significant improvement over the MLE estimation in our experiments. Therefore, we keep it as simple as possible.

Table 3.1: Collections statistics.

collection	queries (title only)	#docs	avgdl	#qrels
Associated Press 88-89	TREC 1-3 Ad-Hoc Track, topics 51-200	165k	287	15,838
TREC Disks 4&5 minus Congressional Record	TREC 2004 Robust Track, topics 301-450 & 601-700	528k	254	17,412
2004 crawl of .gov domains	TREC 2004-06 Terabyte Track, topics 701-850	25m	648	26,917

$$p(w|\theta^*_Q) = \alpha\ p(w|\theta_Q) + (1-\alpha)\ p(w|\theta_F) \tag{3.21}$$

Note that the original query language model can be computed using either the maximum likelihood estimation, or one of the embedding based query language models proposed in Section 3.3.1.1.

3.3.1.3 Experiments

Data: To evaluate the proposed methods, we used three standard TREC collections: AP (Associated Press 1988-1989), Robust (TREC Robust Track 2004 collection), and GOV2 (TREC Terabyte Track 2004-2006 collection). The first two collections contain high-quality news articles and the last one is a large web collection. The statistics of these collections are reported in Table 3.1. We considered the title of topics as queries in our experiments. The standard INQUERY stopword list was used in all experiments, and no stemming was performed.

We employed the KL-divergence retrieval model [85] with the Dirichlet prior smoothing method. All experiments were carried out using the Galago toolkit.[4] In all the experiments, we used the word embeddings produced by the GloVe method [123] trained on a 6 billion token collection (i.e., the Wikipedia dump 2014 plus the Gi-

[4] http://www.lemurproject.org/galago.php

gawords 5). Note that we considered only the queries where the embedding vectors of all query terms are available. The employed embedding vectors contain all query terms for 146 (out of 150), 241 (out of 250), and 147 (out of 150) queries in AP, Robust, and GOV2, respectively.

Parameters Setting: In all the experiments, the Dirichlet prior smoothing parameter μ was set to Galago's default value of 1500. In the experiments related to the pseudo-relevance feedback, the number of feedback documents was set to the typical value of 10. In all the experiments (except in those explicitly mentioned), the parameters α (the linear interpolation coefficient), m (the number of terms added to the queries), and β (see Section 3.3.1.2) were set using 2-fold cross-validation to optimize MAP over the queries of each collection. We swept the parameters α and β between $\{0.1, \ldots, 0.9\}$. The value of the parameter m and the number of feedback documents (for PRF experiments) were also selected from $\{10, 20, ..., 100\}$. The free hyper-parameters of the baselines were also set using the same procedure. In all experiments unless explicitly mentioned, the dimension of embedding vectors was set to 200.

Evaluation Metrics: Mean Average Precision (MAP) of the top-ranked 1000 documents is selected as the main evaluation metric to evaluate the retrieval effectiveness. Furthermore, we also consider the precision of the top 5 and 10 retrieved documents (P@5 and P@10). Statistically significant differences of performances are determined using the two-tailed paired t-test computed at a 95% confidence level based on the average precision per query.

To evaluate the robustness of methods, we consider the robustness index (RI) [28] which is defined as $\frac{N_+ - N_-}{|Q|}$, where $|Q|$ denotes the number of queries, and N_+/N_- shows the number of queries improved/decreased by each method compared to the maximum likelihood estimation baseline. To avoid the influence of very small average precision differences in the RI values, we only consider the improvements/losses higher than 10% (relative). The RI value is in the $[-1, 1]$ interval and higher RI values determine more robust methods.

Experiment I: Comparison of Embedding-based Query Expansion Models Against Other Embedding-based Models in Information Retrieval: In the first set of experiments, we compare the proposed global embedding-based query expansion methods (EQE1 and EQE2) against other embedding-based models in information retrieval. To do so, we consider the following baselines: (i) query likelihood with no query expansion (QL), (ii) embedding-based document language model smoothing (called GLM), (iii) a heuristic-based query expansion model based on word embeddings (VEXP) [51], and (iv) a query expansion method based on the embedding similarity of words with average word embedding of query terms (AWE) [8]. In this set of experiments, we only focus on the embedding vectors produced by GloVe as explained earlier.

The results achieved by the proposed methods (EQE1 and EQE2) and the baselines are reported in Table 3.2. According to this table, both proposed methods outperform all the baselines in all collections, in terms of MAP, P@5, P@10, and RI. The t-test results show that the MAP improvements of EQE1 compared to the baselines are significant, except the VEXP model in the Robust collection. Although in some cases EQE2 outperforms EQE1, we can generally claim that in most cases EQE1 has superior performance. Note that EQE1 is based on multiplicative similarity, while EQE2 is based on additive similarity. Multiplicative similarity means that the expanded terms should be close to all query terms.

As shown in Table 3.2, the improvements over the MLE baseline in AP and Robust are higher than those in the GOV2 collection. The reason could be related to the corpus used for extracting the embedding vectors. This corpus mostly contains formal

Table 3.2: Comparing the proposed embedding-based query expansion methods (EQE1 and EQE2) with the baselines. The superscript 1/2/3/4 denotes that the MAP improvements over QL/GLM/VEXP/AWE are statistically significant. The highest value in each row is marked in bold.

Dataset	**Metric**	**QL**	**GLM**	**VEXP**	**AWE**	**EQE1**	**EQE2**
AP	MAP	0.2236	0.2254	0.2338	0.2304	0.2388^{1234}	**0.2391**^{1234}
	P@5	0.4260	0.4369	0.4412	0.4356	0.4397	**0.4466**
	P@10	0.4014	0.4051	0.4038	0.4058	**0.4075**	0.4014
	RI	–	0.10	0.18	0.14	**0.32**	**0.32**
Robust	MAP	0.2190	0.2244	0.2253	0.2224	**0.2292**^{124}	0.2257^{1}
	P@5	0.4606	0.4523	0.4722	0.4680	**0.4739**	0.4622
	P@10	0.3979	0.3929	0.4133	0.4066	0.4162	**0.4183**
	RI	–	0.22	0.17	0.14	**0.30**	0.22
GOV2	MAP	0.2696	0.2684	0.2687	0.2657	**0.2745**^{1234}	0.2727^{4}
	P@5	0.5592	0.5537	0.5932	0.5537	**0.5959**	0.5810
	P@10	0.5531	0.5483	0.5537	0.5503	**0.5660**	0.5517
	RI	–	-0.14	0.10	-0.18	**0.20**	0.08

texts. Therefore, the context of the employed word embedding vectors is more similar to the context of the newswire collections rather than the GOV2 collection.

In Table 3.3, we compare our embedding-based relevance model with three different initial retrievals (QL+ERM, EQE1+ERM, and EQE2+ERM) against query likelihood and RM3, a state-of-the-art variant of the relevance models proposed by Lavrenko and Croft [87]. According to the results, RM3 significantly outperforms QL in all collections in terms of MAP. This shows the effectiveness of pseudo-relevance feedback in information retrieval. The results suggest that QL+ERM outperforms RM3 (QL+RM1), in terms of MAP, P@5, P@10, and RI, in all collections (the same RI value achieved in the Robust collection). The MAP improvements of QL+ERM over RM3 are always statistically significant. QL+ERM achieves high P@5 and P@10 values compared to RM3, especially in GOV2. This shows the importance of capturing semantic similarities for the PRF task. Among all the considered feedback methods, EQE1+ERM outperforms all the other methods in AP and GOV2, in terms of MAP,

Table 3.3: Evaluating the proposed methods in the pseudo-relevance feedback scenario. The superscript 1/2 denotes that the MAP improvements over QL/RM3 are statistically significant. The highest value in each row is marked in bold.

Dataset	Metric	QL	RM3	QL+ERM	EQE1+ERM	EQE2+ERM
AP	MAP	0.2236	0.3051	0.3102^{12}	**0.3178**^{12}	0.3140^{12}
	P@5	0.4260	0.4644	0.4699	**0.4822**	0.4644
	P@10	0.4014	0.4500	0.4521	**0.4568**	0.4479
	RI	–	0.47	**0.52**	0.47	**0.52**
Robust	MAP	0.2190	0.2677	0.2711^{12}	0.2731^{12}	**0.2750**^{12}
	P@5	0.4606	0.4581	0.4639	**0.4797**	0.4730
	P@10	0.3979	0.4191	0.4241	0.4307	**0.4369**
	RI	–	0.31	0.31	0.32	**0.36**
GOV2	MAP	0.2696	0.2938	0.3005^{12}	**0.3012**^{12}	0.2957^{1}
	P@5	0.5592	0.5592	0.5823	**0.5850**	0.5782
	P@10	0.5531	0.5599	0.5830	**0.5844**	0.5782
	RI	–	0.15	**0.22**	0.20	0.20

Table 3.4: The corpora used for training the embedding vectors.

ID	corpus	#tokens	#vocab
Wiki	Wikipedia 2014 & Gigawords 5	6b	400k
Web 42b	Web crawl	42b	1.9m
Web 840b	Web crawl	840b	2.2m

P@5, and P@10. In Robust, EQE2+ERM performs well, in terms of MAP and P@10. Note that as reported in [99], the RM3 method is a robust PRF method, and the experiments show that RM3 is less robust than the proposed methods.

Experiment II: Sensitivity of Embedding-based Query Expansion Performance to the Word Embedding Quality: To analyze the robustness of the proposed methods to the choices made in training the word embedding vectors, we consider three external corpora: Wiki, Web 42b, and Web 840b. The Wiki corpus mostly contains articles with formal language; while the other two corpora are two web collections containing 42 and 840 billion tokens. The statistics of these corpora

Table 3.5: The MAP values achieved by EQE1, EQE2, and ERM (MLE+ERM) with different corpora for training the embedding vectors (dimension = 300).

Dataset	**Method**	**Wiki**	**Web 42b**	**Web 840b**
AP (146 queries)	EQE1	0.2402	0.2356	0.2362
	EQE2	0.2408	0.2352	0.2400
	ERM	0.3106	0.3094	0.3081
Robust (240 queries)	EQE1	0.2294	0.2255	0.2273
	EQE2	0.2271	0.2237	0.2266
	ERM	0.2713	0.2705	0.2683
GOV2 (146 queries)	EQE1	0.2745	0.2729	0.2767
	EQE2	0.2726	0.2713	0.2743
	ERM	0.3013	0.2989	0.3021

are reported in Table 3.4.[5] The dimension of embedding vectors learned from these corpora is 300. The results are reported in Table 3.5. According to this table, the results are robust to the corpus that is used for training the word embedding vectors. There is no significant differences between the values obtained by employing different corpora for learning the embedding vectors. In the GOV2 collection, the Web 840b corpus seems to be slightly better than the other ones. Despite the large gap between the size of Wiki and the other two corpora, the results achieved by Wiki are higher than those obtained by the other ones, in the newswire collections.

Experiment III: Query Expansion with Relevance-based Word Embedding: In this set of experiments, we focus on EQE1 as our best global embedding-based query expansion model and evaluate the performance of different word embedding algorithms. To do so, we compare the proposed relevance-based word embedding models (RLM and RPE) with word2vec and GloVe. All the models were trained on the target retrieval collections. The results are reported in Table 3.6. According to the results, although word2vec performs slightly better than GloVe, no signifi-

[5] The embedding data is available at `http://nlp.stanford.edu/projects/glove/`.

Table 3.6: Evaluating relevance-based word embeddings in the context of query expansion. The superscripts 0/1/2 denote that the MAP improvements over MLE/word2vec/GloVe are statistically significant. The highest value in each row is marked in bold.

Collection	**Metric**	**MLE**	**word2vec**	**GloVe**	**Rel.-based Embedding**	
					RLM	**RPE**
AP	MAP	0.2197	0.2420	0.2389	$\mathbf{0.2580}^{012}$	0.2543^{012}
	P@20	0.3503	0.3738	0.3631	$\mathbf{0.3886}^{012}$	0.3812^{034}
	NDCG@20	0.3924	0.4181	0.4098	$\mathbf{0.4242}^{012}$	0.4226^{012}
Robust	MAP	0.2149	0.2215	0.2172	$\mathbf{0.2450}^{012}$	0.2372^{012}
	P@20	0.3319	0.3337	0.3281	$\mathbf{0.3476}^{012}$	0.3409^{0}
	NDCG@20	0.3863	0.3881	0.3844	$\mathbf{0.3982}^{012}$	0.3955^{0}
GOV2	MAP	0.2702	0.2723	0.2709	$\mathbf{0.2867}^{012}$	0.2855^{012}
	P@20	0.5132	0.5172	0.5128	$\mathbf{0.5367}^{012}$	0.5358^{012}
	NDCG@20	0.4482	0.4509	0.4485	$\mathbf{0.4576}^{02}$	0.4557^{0}

cant differences can be observed between their performances. Both relevance-based embedding models outperform the baselines in all the collections, which shows the importance of taking relevance into account for training embedding vectors. These improvements are often statistically significant compared to all the baselines. The relevance likelihood maximization model (RLM) performs better than the relevance posterior estimation model (RPE) in all cases and the reason is related to their objective function. RLM learns the relevance distribution for all terms, while RPE learns the classification probability of being relevance for vocabulary terms (see Section 3.2).

To get a sense of what is learned by each of the embedding models, Tables 3.7 and 3.8 report the top 10 expansion terms for two sample queries from the TREC Robust Track in 2004. According to these tables, the terms added to the queries by the word2vec model are syntactically or semantically related to individual query terms. For the query "indian american museum" as an example, the terms "history", "art", and "culture" are related to the query term "museum", while the terms "united" and "states" are related to the query term "american". In contrast, looking at the

Table 3.7: Top 10 expansion terms obtained by the word2vec and the relevance-based word embedding models for an example query: “indian american museum”.

word2vec	**Rel.-based Embedding**	
	RLM	**RPE**
powwows	chumash	heye
smithsonian	heye	collection
afro	artifacts	chumash
mesoamerica	smithsonian	smithsonian
smithsonians	collection	york
native	washington	new
heye	institution	apa
hopi	york	native
mayas	native	americans
cimam	apa	history

expansion terms obtained by the relevance-based word embeddings, we can see that some relevant terms to the whole query were selected. For instance, “chumash” (a group of native americans)[6], “heye” (the national museum of the American Indian in New York), “smithsonian” (the national museum of the American Indian in Washington DC), and “apa” (the American Psychological Association that actively promotes American Indian museums). A similar observation can be made for the other sample query (i.e., “tibet protesters”). For example, the word “independence” is related to the whole query that was only selected by the relevance-based word embedding models, while the terms “protestors”, “protests”, “protest”, and “protesting” that are syntactically similar to the query term “protesters” were considered by the word2vec model. We believe that these differences are due to the learning objective of the models. Interestingly, the expansion terms added to each query by the two relevance-based models look very similar, but according to Table 3.6, their performances are quite different. The reason is related to the weights given to each term by the two

[6] see `https://en.wikipedia.org/wiki/Chumash_people`

Table 3.8: Top 10 expansion terms obtained by the word2vec and the relevance-based word embedding models for an example query: “tibet protesters”.

word2vec	**Rel.-based Embedding**	
	RLM	**RPE**
tibetan	tibetan	tibetan
lhasa	lama	tibetans
demonstrators	tibetans	lama
tibetans	lhasa	independence
marchers	dalai	lhasa
lhasas	independence	dalai
jokhang	protest	open
demonstrations	open	protest
dissidents	zone	zone
barkhor	followers	jokhang

models. The weights given to the expansion terms by RPE are very close to each other because its objective is to just classify each term and all of these terms are classified with a high probability as “relevant”.

In the next experiment, we consider the methods that use the top retrieved documents for query expansion: the relevance model (RM3) [2, 87], and the local embedding approach proposed by Diaz et al. [45] with the general idea of training word embedding models on the top ranked documents retrieved in response to a given query. Similar to [45], we use the word2vec model to train word embedding vectors on top 1000 documents. The results are reported in Table 3.9. The ERM model that uses the relevance-based word embedding (RLM[7]) outperforms all the other methods. These improvements are statistically significant in most cases. By comparing the results obtained by local embedding and those reported in Table 3.6, it can be observed that there are no substantial differences between the results for local embedding and word2vec. This is similar to what is reported by Diaz et al. [45] when the

[7] We only consider RLM which shows better performance compared to RPE in query expansion.

Table 3.9: Evaluating relevance-based word embedding in pseudo-relevance feedback scenario. The superscripts 1/2/3 denote that the MAP improvements over RM3/EQE1 with Local Embedding/ERM with Local Embedding are statistically significant. The highest value in each row is marked in bold.

Collection	**Metric**	**RM3**	**EQE1**	**ERM**	
			Local Emb.	**Local Emb.**	**RLM**
AP	MAP	0.2927	0.2412	0.3047	**0.3119**^{12}
	P@20	0.4034	0.3742	0.4105	**0.4233**^{12}
	NDCG@20	0.4368	0.4173	0.4411	**0.4495**^{123}
Robust	MAP	0.2593	0.2235	0.2643	**0.2761**^{123}
	P@20	0.3486	0.3366	0.3498	**0.3605**^{123}
	NDCG@20	0.4011	0.3868	0.4080	**0.4173**^{123}
GOV2	MAP	0.2863	0.2748	0.2924	**0.2986**^{123}
	P@20	0.5318	0.5271	0.5379	**0.5417**^{12}
	NDCG@20	0.4503	0.4576	0.4584	**0.4603**^{123}

embedding vectors are trained on the top documents in the target collection, similar to our setting. Note that the relevance-based model was also trained on the target collection.

An interesting observation from Tables 3.6 and 3.9 is that the RLM performance (without using pseudo-relevant documents) in Robust and GOV2 is very close to the RM3 performance, and is slightly better in the GOV2 collection. Note that RM3 needs two retrieval runs and uses top retrieved documents, while RLM only needs one retrieval run. This is an important issue in many real-world applications, since the efficiency constraints do not always allow them to have two retrieval runs per query.

3.3.2 Query Classification

Query classification, also known as query categorization, has the goal of classifying search queries into a number of pre-defined categories. Two types of classification have been studied. In the first, the labels are query types, such as navigational queries, informational queries, and transactional queries, e.g., [17, 88]. The other task is classifying queries based on their topics. Query classification methods are often based on the k-nearest neighbors approach; they classify each query based on the labels of the most similar queries by majority voting. In the following, we estimate distributed query representations from the learned word embedding vectors, and use the obtained representations for computing query similarity.

3.3.2.1 Estimating Embedding Vectors for Queries

Estimating accurate query models is a crucial component in every retrieval framework. It has been extensively studied in existing retrieval models and several approaches have been proposed for estimating query models, especially in the language modeling framework. Here, we focus on query representation in a pre-trained embedding semantic space: *how to estimate accurate embedding vectors for queries?* Formally, let E denote a set of d-dimensional embedding vectors for each vocabulary term w. Given a query $q = \{w_1, w_2, \cdots, w_k\}$ with the length of k, the problem is to estimate a d-dimensional vector $\vec{q}$, henceforth called *query embedding vector*, for the query q.

We propose a probabilistic framework to estimate query embedding vectors based on maximum likelihood estimation. The main idea behind our approach is to maximize the likelihood of an accurate query language model and a probabilistic distribution that can be calculated using the embedding semantic space for each query vector. To do so, let $\delta(\cdot, \cdot)$ denote a similarity function that computes the similarity between two embedding vectors. Hence, the probability of each term w given a query vector $\vec{q}$, henceforth called the *query embedding distribution*, can be calculated as:

$$p(w|\vec{q}) = \frac{\delta(\vec{w}, \vec{q})}{Z} \tag{3.22}$$

where $\vec{w} \in E$ denotes the embedding vector of the given term w. The normalization factor Z can be calculated as follows:

$$Z = \sum_{w' \in V} \delta(\vec{w'}, \vec{q}) \tag{3.23}$$

where V denotes the set of all vocabulary terms.[8] On the other hand, assume that there is a query language model θ_q for the query q, that shows how much each word contributes to the query. Our claim is that a query embedding vector $\vec{q^*}$ is a proper query embedding vector, if the query embedding distribution (see Equation (3.22)) is "close to" the query language model θ_q. In other words, our purpose is to find a query embedding vector that maximizes the following log-likelihood function:

$$\vec{q^*} = \arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) \log p(w|\vec{q}) \tag{3.24}$$

The high computational complexity of the normalization factor in calculating the query embedding distribution (see Equation (3.23)) makes optimizing the log-likelihood function expensive. Note that since the normalization factor Z depends on the query embedding vector, it cannot be computed offline. Therefore, similar to many other optimization problems, we need to relax our objective function. To this end, we assume that the normalization factor Z in Equation (3.22) is equal for all query vectors. Although this simplifying assumption is not true, our observations indicate that this is not a harmful assumption. To give an intuition about the validity of this assumption, we consider many ($> 200,000$) random query vectors and calculate the normalization factor Z for all of these query vectors. The mean and standard deviation for all Z values are $(1.7 \pm 0.15) * 10^4$. The mean value is an order of magnitude larger than the standard deviation, and this shows that most of the Z values are close

[8] In this work, we assume that the embedding vectors of all query terms are available.

to the mean value, which indicates that our assumption is reasonable. It is worth noting that all the following calculations can be done without this assumption, but with high computational cost.

Therefore, based on this relaxation, we can re-write our objective function as follows:

$$\arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) \log \delta(\vec{w}, \vec{q}) \tag{3.25}$$

As shown in Equation (3.25), our framework consists of two main components: the query language model θ_q and the similarity function δ. Since the output of $\delta(\vec{w}, \vec{q})$ is dependent on the query vector $\vec{q}$, the function δ can affect the way that we can optimize the objective function.

The similarity function δ for computing the similarity between two embedding vectors can be calculated using the softmax function as follows:

$$\delta(\vec{w}, \vec{w'}) = \exp\left(\frac{\sum_{i=1}^{d} \vec{w}_i \vec{w'}_i}{\|\vec{w}\| \|\vec{w'}\|}\right) \tag{3.26}$$

where d denotes the dimension of embedding vectors. Without loss of generality, assume that the embedding vectors of all vocabulary terms are unit vectors, and thus their norms are equal to 1. Therefore, our objective function (see Equation (3.25)) can be re-written as follows:

$$\arg\max_{\vec{q}} \sum_{w \in V} p(w|\theta_q) . \frac{\sum_{i=1}^{d} \vec{w}_i \vec{q}_i}{\|\vec{q}\|} \tag{3.27}$$

To make this objective function even simpler, we add a constraint to force the query vector be a unit vector. In other words, we consider the following constraint: $\|q\| = 1$. Based on this constraint, we obtain the Lagrange function as follows:

$$\mathcal{L}(\vec{q}, \lambda) = \sum_{w \in V} \left(p(w|\theta_q) \sum_{i=1}^{d} \vec{w}_i \vec{q}_i \right) + \lambda \left(1 - \sum_{i=1}^{d} (\vec{q}_i)^2 \right) \tag{3.28}$$

where λ denotes the Lagrange multiplier. Using the mathematical optimization method of Lagrange multipliers, we compute the first derivatives of the Lagrange function as follows:

$$\begin{cases} \frac{\partial \mathcal{L}}{\partial \vec{q_i}} = \sum_{w \in V} \vec{w_i} p(w|\theta_Q) - 2\lambda \vec{q_i} \\ \frac{\partial \mathcal{L}}{\partial \lambda} = 1 - \sum_{i=1}^{d} (\vec{q_i})^2 \end{cases} \tag{3.29}$$

where $\vec{q_i}$ denotes the i^{th} element of the query vector $\vec{q}$. By setting the above partial derivatives to zero, we can find the stationary point of our objective function as below:

$$\vec{q_i} = \frac{\sum_{w \in V} \vec{w_i} p(w|\theta_Q)}{\sqrt{\sum_{j=1}^{d} \left(\sum_{w \in V} \vec{w_i} p(w|\theta_Q)\right)^2}} \tag{3.30}$$

Therefore, the query embedding vector can be calculated using the above closed-form formula.

Estimating Query Language Model using Maximum Likelihood: Maximum likelihood estimation (MLE) is a simple yet effective approach for estimating query language models. MLE for a given query q can be calculated by relative counts:

$$p(w|\theta_q) = \frac{\text{count}(w, q)}{|q|} \tag{3.31}$$

where θ_q, $\text{count}(w, q)$, and $|q|$ denote the unigram query language model, the count of term w in the query q, and the query length, respectively.

As shown earlier, each element of the query embedding vector can be calculated using Equation (3.30). In this equation, the denominator is just a normalization factor to force the embedding vector be a unit vector. Now, assume that the query language model θ_q is estimated using maximum likelihood estimation. Therefore, the query embedding vector is calculated as:

$$\vec{q_i} \propto \sum_{w \in q} \frac{\text{count}(w, q)}{|q|} . \vec{w_i} \tag{3.32}$$

where $\vec{q_i}$ denotes the i^{th} element of the query vector. In the above equation, the summation is just over the query terms, since the count of other terms is equal to zero, and thus they will not affect the result. The above equation is equivalent to average word embedding (AWE). To summarize, when the similarity of embedding vectors is calculated using the softmax function and the query language model is estimated using maximum likelihood estimation, the proposed framework will produce the AWE method.

Estimating Query Language Model using Pseudo-Relevance Feedback: Pseudo-relevance feedback (PRF) has been shown to be highly effective at improving retrieval effectiveness. PRF in the language modeling framework estimates a query language model from a small set of top-retrieved documents. In PRF, in addition to updating the weights of query terms, a number of new terms will be added to the query. We consider the relevance model with the i.i.d. sampling assumption [87], a state-of-the-art PRF method, to estimate the query language model as follows:

$$p(w|\theta_q) \propto \sum_{d \in F} p(w|d) \prod_{w' \in q} p(w'|d) \tag{3.33}$$

where F denotes the set of feedback documents. The probability of each term in the document (e.g., $p(w|d)$) can be computed by smoothing the maximum likelihood estimation probability. The top m terms with highest probabilities are usually selected in the feedback language models. We call the query embedding vectors estimated using the PRF distributions, *pseudo query vectors* (PQV).

3.3.2.2 Query Classification with k-Nearest Neighbors

To classify each query, we consider a very simple approach based on query embedding vectors. We first compute the probability of each category/label given each query q and then select the top t categories with highest probabilities. In fact, this method is based on k-nearest neighbors classification. The probability $p(C_i|q)$ can be easily computed using the following formula:

$$p(C_i|q) = \frac{\delta(\vec{C_i}, \vec{q})}{\sum_j \delta(\vec{C_j}, \vec{q})} \propto \delta(\vec{C_i}, \vec{q}) \tag{3.34}$$

where C_i denotes the i^{th} category. $\vec{C_i}$ is the centroid vector of all query embedding vectors with the label of C_i. In the above equation, we drop the normalization factor, since it is the same for all categories. For PQV methods, we linearly interpolate the above probability with those computed using the MLE method with the interpolation coefficient of α.

3.3.2.3 Experiments

Data: We consider the dataset that was previously employed to evaluate the query classification approaches submitted to the KDD Cup 2005: Internet user search query categorization [93]. This evaluation set contains 800 web queries that were issued by real users. These queries were randomly selected and do not contain junk and non-English words/phrases. The queries were tagged by three individual human editors. The KDD Cup 2005 organizers pre-defined 67 categories (labels) and each editor selected up to 5 labels among them for each query. The embedding vectors of all query terms of 700 out of 800 queries are available in our embedding collection. We only consider these 700 queries in our evaluations. The spelling errors in queries are corrected in a pre-processing phase.

In our evaluations, we consider 5-fold cross-validation over the queries and the reported results are the average of all results obtained over the test folds. In each step we have 560 and 140 training and test queries, respectively.

In the experiments related to PQV, we use the Robust collection (see Table 3.1 for details) to retrieve pseudo-relevant documents. In all experiments, stopwords

Table 3.10: Evaluating embedding algorithms via query classification. The superscripts 1/2 denote that the improvements over word2vec/GloVe with the same query embedding method are significant. The superscript * denotes that the improvements over the AWE method with all embeddings are significant. The highest value in each column is marked in bold.

Query Emb. Method	Word Embedding	Precision	F1-measure
AWE	word2vec	0.3712	0.4008
	GloVe	0.3643	0.3912
	Rel.-based Embedding - RLM	0.3943^{12}	0.4267^{12}
	Rel.-based Embedding - RPE	0.3961^{12}	0.4294^{12}
PQV	word2vec	0.3803	0.4227
	GloVe	0.3777	0.4142
	Rel.-based Embedding - RLM	0.4069^{12*}	0.4460^{12*}
	Rel.-based Embedding - RPE	$\mathbf{0.4086}^{12*}$	$\mathbf{0.4483}^{12*}$

were removed from the queries and no stemming was performed. We employed the INQUERY stopword list.

Parameters Setting: In all experiments unless explicitly mentioned, the parameters α (the linear interpolation coefficient), and n (the number of feedback terms considered in the PQV methods) were tuned on the training data. We swept the parameter α between $\{0.1, \ldots, 0.9\}$. The values of parameter n were also selected from $\{10, 20, ..., 100\}$.

Evaluation Metrics: We consider two widely used evaluation metrics that were also used in KDD Cup 2005 [93]: precision and F1-measure. Since the labels assigned by the three human editors differ in some cases, all the label sets should be taken into account. We compute these two metrics in the same way as was used to evaluate the KDD Cup 2005 submissions [93]. Statistically significant differences are determined using the two-tailed paired t-test computed at a 95% confidence level.

Results: We compare our models against the word2vec and GloVe methods. In our experiment, we use two different query embedding approach, one using average word embedding of query terms (AWE) and one based on the pseudo-query vector (PQV) that uses the top 10 retrieved documents to estimate query embedding. The results are reported in Table 3.10, where the relevance-based embedding models significantly outperform the baselines in terms of both metrics. An interesting observation here is that contrary to the query expansion experiments, RPE performs better than RLM in query classification. The reason is that in query expansion the weight of each term is considered in order to generate the expanded query language model. Therefore, in addition to the order of terms, their weights should be also effective for improving the retrieval performance with query expansion. In query classification, we only assign a few categories to each query, and thus as long as the order of categories is correct, the similarity values between the queries and the categories do not matter. The results also demonstrate the pseudo-query vector method significantly outperforms the AWE method.

3.4 Summary

In summary, we presented a neural network architecture, similar to word2vec [107], for learning relevance-based word embedding. Our model is based on the bag-of-words assumption. We presented two objectives for training the model: relevance likelihood maximization (RLM) and relevance posterior estimation (RPE). We trained our model using weak supervision. In more detail, we employed the Lavrenko and Croft's relevance models [87] (i.e., RM3) for generating training data.

The learned word representations were extrinsically evaluated using two downstream tasks: query expansion (with and without pseudo-relevance feedback) and query classification. Our experiments on multiple datasets demonstrate the effectiveness of the proposed solutions to the bag-of-words word embedding models, such as word2vec [107] and GloVe [123].

CHAPTER 4
NEURAL RANKING MODELS

Developing efficient and effective retrieval models has always been at the core of information retrieval research. In this chapter, we study how to train learning to rank models without labeled data using *weak supervision*. In more detail, we use existing term matching retrieval models, such as BM25 or query likelihood, to automatically generate pairwise training data. The goal is to learn a retrieval model using the generated data, such that it outperforms the weak labeler. This chapter provides theoretical evidence and guideline for weakly supervised neural ranking models. In addition, we present empirical results showing that the proposed model outperforms state-of-the-art retrieval models.

This chapter is structured as follows: Section 4.1 theoretically studies weak supervision for learning to rank models. Leveraging weak supervision, Section 4.2 proposes a standalone neural ranking model, called SNRM, as an efficient and effective retrieval model.

4.1 Learning to Rank with Weak Supervision

Despite the advances in computer vision, speech recognition, and NLP tasks using unsupervised deep neural networks, such advances have not been observed in *ranking*, as a core information retrieval task. A plausible explanation is the complexity of the ranking problem in IR, in the sense that it is not obvious how to learn a ranking model from queries and documents when no supervision in form of the relevance information is available. To overcome this issue, Dehghani et al. [42] have recently proposed to leverage large amounts of data to infer noisy or weak labels and use that signal for learning models without labeled data. In more detail, they used an existing retrieval model, such as BM25, to retrieve documents in response to a large number of queries. They further trained a simple neural ranking model based on the generated training data and showed significant improvements compared to the weak labeler.

In the following, we present a set of theoretical results on learning to rank from weakly supervised data. We define symmetric ranking loss functions and further study the problem under uniformity and non-uniformity assumptions. Finally, we study a set of pairwise loss functions to determine which ones are symmetric.

4.1.1 Preliminaries

In the following, we briefly review preliminary information used in the rest of the chapter. In more detail, we first formulate the task of learning to rank objects, and then introduce the risk minimization framework for statistical learning.

4.1.1.1 Learning to Rank Formulation

Suppose $\mathcal{X}$ is the input space and $\mathcal{Y}$ is the output space for a ranking problem. Each element of $\mathcal{X}$, denoted as $\mathbf{x}$, is a list of n feature vectors corresponding to n objects that should be ranked, i.e., $\mathbf{x} = [x_1, x_2, \cdots, x_n]$. Each element of $\mathcal{Y}$, denoted as $\mathbf{y}$, is also a list of n labels corresponding to the objects appeared in $\mathbf{x}$, i.e., $\mathbf{y} = [y_1, y_2, \cdots, y_n]$. Let $P(X, Y)$ be an unknown joint distribution, where random variables X and Y respectively take $\mathbf{x}$ and $\mathbf{y}$ as their values.

Assume that $\mathcal{M}$ is a ranking model that takes $\mathbf{x}$ as input and generates a rank list from the objects appearing in $\mathbf{x}$. Without loss of generality, we assume that $\mathcal{M}$ produces a score for each object in $\mathbf{x}$ and the objects are then sorted based on their scores in descending order. Therefore, $\mathcal{M}(\mathbf{x})$ can be written as $[\mathcal{M}(x_1), \mathcal{M}(x_2), \cdots, \mathcal{M}(x_n)]$.

In typical statistical learning to rank problems, we are given a training set $\mathcal{T} = \{(\mathbf{x}_1, \mathbf{y}_1), (\mathbf{x}_2, \mathbf{y}_2), \cdots, (\mathbf{x}_m, \mathbf{y}_m)\}$ with m elements, each representing a rank list for n

objects. The training instances are assumed to be drawn *iid* according to an unknown distribution over $\mathcal{X} \times \mathcal{Y}$. In document ranking, e.g., ad-hoc retrieval, $\mathbf{x}_i$ is equal to $\{(q_i, d_{i1}), (q_i, d_{i2}), \cdots, (q_i, d_{in})\}$, where q_i denotes the i^{th} query in the training set and d_{ij} denotes the j^{th} candidate document that should be ranked in response to the query q_i. $\mathbf{y}_i$ is also a list of n labels representing the relevance judgments for the corresponding query-document pairs.

4.1.1.2 Risk Minimization Framework

In this subsection, we briefly explain the risk minimization framework in statistical learning. The risk function for a given ranking model $\mathcal{M}$ and loss function $\mathcal{L}$ is defined as follows:

$$R_{\mathcal{L}}(\mathcal{M}) = \mathbb{E}_P[\mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y})] = \int\int \mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y}) P(\mathbf{x}, \mathbf{y}) \, \mathrm{d}\mathbf{x} \, \mathrm{d}\mathbf{y} \tag{4.1}$$

where $\mathbb{E}$ denotes expectation and its subscript is a distribution (or a random variable) with respect to which the expectation is taken. As mentioned earlier, P denotes an unknown true distribution over the $\mathcal{X} \times \mathcal{Y}$ space. $\mathcal{L}(\cdot, \cdot)$ is a loss function that computes the difference between its inputs which are two lists with the size of n.

Given the training data $\mathcal{T}$, the empirical risk is defined as follows:

$$\widehat{R}_{\mathcal{L}}(\mathcal{M}) = \mathbb{E}_{(\mathbf{x},\mathbf{y}) \in \mathcal{T}}[\mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y})] = \frac{1}{m} \sum_{i=1}^{m} \mathcal{L}(\mathcal{M}(\mathbf{x}_i), \mathbf{y}_i) \tag{4.2}$$

Under the risk minimization framework, the objective is to learn a ranking model $\mathcal{M}$ that is a global minimizer of the risk function $\widehat{R}_{\mathcal{L}}$, which depends on the ranking loss function $\mathcal{L}$.[1]

[1] Since ranking metrics are often non-differentiable, a surrogate loss function is often used. We can assume that $\mathcal{L}$ is a surrogate ranking loss.

4.1.2 Problem Statement

As pointed out earlier in this chapter, weak supervision is a learning strategy that does not require labeled training data. Let M be a weak supervision signal that automatically produces a label for any given input. This gives us a set of weakly labeled training data $\widehat{\mathcal{T}} = \{(\mathbf{x}_1, \widehat{\mathbf{y}}_1), (\mathbf{x}_2, \widehat{\mathbf{y}}_2), \cdots, (\mathbf{x}_m, \widehat{\mathbf{y}}_m)\}$, where $\widehat{\mathbf{y}}_i = [M(x_{i1}), M(x_{i2}), \cdots, M(x_{in})]$. The input feature vectors $\mathbf{x}$ in weakly labeled data are the same as the ones in $\mathcal{T}$ described in Section 4.1.1.1.

The weakly supervised labels are generated automatically and thus are not accurate. In the following, we theoretically study how to effectively train a ranking model on such noisy labels. Without loss of generality, we can look at weak supervision as a noisy channel that applies some noise on the actual true labels. Therefore, $\widehat{\mathbf{y}} = M(\mathbf{x})$ is modeled as a noisy channel that introduces noise on the true label $\mathbf{y}$.

Let us first define noise tolerance in ranking based on the risk minimization framework as follows:

Definition 1. [Noise Tolerance in Learning to Rank] Under the ranking loss function $\mathcal{L}$, risk minimization is noise tolerant if:

$$Pr[\mathcal{M}(\mathbf{x}) \overset{\text{rank}}{=} \mathbf{y}] = Pr[\widehat{\mathcal{M}}(\mathbf{x}) \overset{\text{rank}}{=} \mathbf{y}] \qquad \forall (\mathbf{x}, \mathbf{y}) \in \mathcal{T} \tag{4.3}$$

where $\cdot \overset{\text{rank}}{=} \cdot$ denotes that the two given score lists are equal in terms of ranking (i.e., the corresponding objects are in the same order when sorted by their scores). $\mathcal{M}$ and $\widehat{\mathcal{M}}$ respectively denote the ranking models trained on the true data $\mathcal{T}$ and the weakly supervised data $\widehat{\mathcal{T}}$.

4.1.3 Symmetric Ranking Loss

Motivated by the symmetry condition defined for binary classification [52], we define symmetric ranking loss function as follows:

Definition 2. [Symmetric Ranking Loss Function] A ranking loss function $\mathcal{L}$ is symmetric, if it satisfies the following constraint:

$$\sum_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y}) = c \quad \forall \mathbf{x}, \forall \mathcal{M} \tag{4.4}$$

where c is a constant number.

Note that the above definition assumes that the output space $\mathcal{Y}$ is finite and discrete, which is a reasonable assumption for a ranking task, where the order of objects matters and the number of items is finite (i.e., equal to n in our setting). In case of binary relevance judgments, the output space $\mathcal{Y}$ is $\{0,1\}^n$, thus $|\mathcal{Y}| = 2^n$.

Theorem 1. In case of binary relevance judgments (labels), any ranking loss function $\mathcal{L}$ based on a pairwise classification loss $\mathcal{L}_{pair}$ is symmetric, if the following condition, called the sufficient symmetric pairwise loss (SSPL) constraint, holds:

$$\mathcal{L}_{pair}(\mathcal{M}(x) - \mathcal{M}(x'), -1) + \mathcal{L}_{pair}(\mathcal{M}(x) - \mathcal{M}(x'), 1) = c \quad \forall x, x', \forall \mathcal{M} \tag{4.5}$$

where c is a constant number.[2]

Proof. If the ranking loss function $\mathcal{L}$ is computed based on pairwise misclassifications, then without loss of generality, we have:

$$\begin{aligned}\sum_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y}) &= \frac{1}{Z}\sum_{\mathbf{y}\in\mathcal{Y}}\sum_{i=1}^{n}\sum_{j=i+1}^{n} \mathcal{L}_{pair}(\mathcal{M}(x_i) - \mathcal{M}(x_j), y_i - y_j) \\ &= \frac{1}{Z}\sum_{i=1}^{n}\sum_{j=i+1}^{n}\sum_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}_{pair}(\mathcal{M}(x_i) - \mathcal{M}(x_j), y_i - y_j) \qquad (4.6)\end{aligned}$$

[2] We assume that the pairwise loss for two objects with the same label is zero (or constant).

where Z is a constant normalization factor. Since the labels are binary, thus $\mathcal{Y} = \{0,1\}^n$. The terms inside the summations in Equation (4.6) only depend on the i^{th} and the j^{th} element of $\mathbf{y}$. Therefore, we can rewrite the above equations as below:

$$\begin{aligned}
&= \frac{2^{n-2}}{Z} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \sum_{(y_i,y_j)\in\{0,1\}^2} \mathcal{L}_{pair}(\mathcal{M}(x_i) - \mathcal{M}(x_j), y_i - y_j) \\
&= \frac{2^{n-2}}{Z} \sum_{i=1}^{n} \sum_{j=i+1}^{n} \mathcal{L}_{pair}(\mathcal{M}(x_i) - \mathcal{M}(x_j), -1) - \mathcal{L}_{pair}(\mathcal{M}(x_i) - \mathcal{M}(x_j), 1) \qquad (4.7)
\end{aligned}$$

Note that the pairwise loss for two objects with the same labels is assumed to be zero. Given the SSPL condition mentioned in Equation (4.5), we rewrite the above equation as:

$$\Rightarrow \sum_{\mathbf{y}\in\mathcal{Y}} \mathcal{L}(\mathcal{M}(\mathbf{x}), \mathbf{y}) = \frac{2^{n-2}}{Z} \sum_{i=1}^{n} \sum_{j=i+1}^{n} c = c' \qquad (4.8)$$

which shows that $\mathcal{L}$ is a symmetric ranking function, and completes the proof. □

In Section 4.1.6, we study a number of well-known pairwise loss functions and discuss whether they satisfy the SSPL constraint. At this step, it is sufficient to know that there exist some loss functions that satisfy the SSPL constraint, and thus the following theoretical findings are useful in practice.

4.1.4 Weak Supervision as Uniform Noisy Channel

We generalize Ghosh et al.'s findings [52] on binary classification to ranking and we assume that independent from the input $\mathbf{x}$, the noisy channel applies a *uniform* noise on the true label and produces the weak label. Although, this is a strong assumption that does not hold in many real-world situations, it gives insights into understanding learning from weak supervision, and is a first step towards more complex situations (e.g., the non-uniformity assumption).

Theorem 2. In learning to rank from weak supervision, where the weak signal is drawn from the true label with a uniform noise, let $\mathcal{L}$ be a symmetric ranking loss function (see Definition 2). Then, $\mathcal{L}$ is noise tolerant (see Definition 1) under the noise probability $\rho < \frac{2^n-1}{2^n}$.

Proof. We prove this theorem based on the risk minimization framework. To do so, we show that any ranking model $\widehat{\mathcal{M}}^*$ that is a global minimizer for the empirical risk function on the weak data is also a global minimizer of the true empirical risk.

The empirical risk function for a ranking model $\mathcal{M}$ over the weak data $\widehat{\mathcal{T}}$ is defined as:

$$\widehat{R}'_{\mathcal{L}}(\mathcal{M}) = \mathbb{E}_{(\mathbf{x},\widehat{\mathbf{y}})\in\widehat{\mathcal{T}}}[\mathcal{L}(\mathcal{M}(\mathbf{x}),\widehat{\mathbf{y}})] = \mathbb{E}_{\mathbf{x}}\ \mathbb{E}_{\mathbf{y}|\mathbf{x}}\ \mathbb{E}_{\widehat{\mathbf{y}}|\mathbf{x},\mathbf{y}}[\mathcal{L}(\mathcal{M}(\mathbf{x}),\widehat{\mathbf{y}})] \tag{4.9}$$

Given the uniform noise assumption, the expected value of the loss for the weak supervision label is equal to the loss for the true label with the probability of $1-\rho$ and the probability of $\frac{\rho}{2^n-1}$ for any other labels with binary judgments. Hence, the empirical risk $\widehat{R}'_{\mathcal{L}}(\mathcal{M})$ is equal to:

$$\mathbb{E}_{\mathbf{x}}\ \mathbb{E}_{\mathbf{y}|\mathbf{x}}\left[(1-\rho)\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y}) + \frac{\rho}{2^n-1}\sum_{\mathbf{y}'\in\mathcal{Y}\setminus\{\mathbf{y}\}}\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y}')\right] \tag{4.10}$$

Since $\mathbb{E}_{\mathbf{x}}\ \mathbb{E}_{\mathbf{y}|\mathbf{x}} = \mathbb{E}_{\mathbf{x},\mathbf{y}}$ and $\mathcal{L}$ is a symmetric ranking loss, given the definition of empirical risk for true labels, we have:

$$\begin{aligned}\widehat{R}'_{\mathcal{L}}(\mathcal{M}) &= (1-\rho)\widehat{R}_{\mathcal{L}}(\mathcal{M}) + \frac{\rho}{2^n-1}(c-\widehat{R}_{\mathcal{L}}(\mathcal{M}))\\ &= \frac{c\rho}{2^n-1} + (1-\frac{2^n\rho}{2^n-1})\widehat{R}_{\mathcal{L}}(\mathcal{M})\end{aligned} \tag{4.11}$$

Now assume that $\widehat{\mathcal{M}}^*$ is a global minimizer for the risk function $\widehat{R}'_{\mathcal{L}}$, thus for any ranking model $\mathcal{M}$, we have:

$$\widehat{R}'_{\mathcal{L}}(\widehat{\mathcal{M}}^*) - \widehat{R}'_{\mathcal{L}}(\mathcal{M}) \leq 0 \tag{4.12}$$

On the other hand, from Equation (4.11) we have:

$$\widehat{R}'_{\mathcal{L}}(\widehat{\mathcal{M}}^*) - \widehat{R}'_{\mathcal{L}}(\mathcal{M}) = (1 - \frac{2^n \rho}{2^n - 1})(\widehat{R}_{\mathcal{L}}(\widehat{\mathcal{M}}^*) - \widehat{R}_{\mathcal{L}}(\mathcal{M})) \tag{4.13}$$

According to Equations (4.12) and (4.13) and because $\rho < \frac{2^n-1}{2^n}$, then $\widehat{R}_{\mathcal{L}}(\widehat{\mathcal{M}}^*) - \widehat{R}_{\mathcal{L}}(\mathcal{M}) \leq 0$ and thus $\widehat{\mathcal{M}}^*$ is also a global minimizer for the true empirical risk function $\widehat{R}_{\mathcal{L}}$. Therefore, risk minimization under symmetric ranking losses is noise tolerant, which completes the proof. □

This theorem shows that symmetric ranking losses are robust to uniform noise used to generate weak supervision labels. Note that the only condition is ρ being less than $\frac{2^n-1}{2^n}$. This means that the weak signal should be better than random, which is not a restrictive condition. Interestingly, this finding is independent of the size of training data.

4.1.5 Weak Supervision as Non-uniform Noisy Channel

Section 4.1.4 assumes that the error probability of weak labeler is the same for all training instances. This means that the quality of weak supervision signal is the same for all queries, which is not a true assumption in practice, i.e., some queries are more difficult and some are easier to respond. In Theorem 3, we relax this assumption and find an upper bound for the empirical risk function.

Theorem 3. Let $\mathcal{L}$ be a symmetric ranking loss function (see Definition 2). For each pair $(\mathbf{x}, \widehat{\mathbf{y}}) \in \widehat{\mathcal{T}}$, assume that the weak label $\widehat{\mathbf{y}}$ is equal to the true label with a probability of $\rho_{\mathbf{x}}$ which depends on the input $\mathbf{x}$. Then, the empirical risk function $\widehat{R}_{\mathcal{L}}(\widehat{\mathcal{M}}^*)$ is upper bounded by $\widehat{R}_{\mathcal{L}}(\mathcal{M}^*)/(1 - \frac{2^n \rho_{max}}{2^n-1})$, where $\widehat{\mathcal{R}}_{\mathcal{L}}$ is empirical risk on the true data, ρ_{max} is the maximum error probability, and $\widehat{\mathcal{M}}^*$ and $\mathcal{M}^*$ are the global minimizers of the weak risk $\widehat{R}'_{\mathcal{L}}$ and the true risk $\widehat{R}_{\mathcal{L}}$, respectively.

Proof. Similar to Equation (4.10), the empirical risk function for any ranking model $\mathcal{M}$ over the weakly labeled data $\widehat{\mathcal{T}}$ is defined as:

$$\begin{aligned}
\widehat{R}'_{\mathcal{L}}(\mathcal{M}) &= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[(1-\rho_{\mathbf{x}})\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y}) + \frac{\rho_{\mathbf{x}}}{2^n-1}\sum_{\mathbf{y}'\in\mathcal{Y}\setminus\{\mathbf{y}\}}\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y}')\right] \\
&= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[(1-\rho_{\mathbf{x}})\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y}) + \frac{\rho_{\mathbf{x}}}{2^n-1}(c-\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y}))\right] \\
&= \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\frac{c\rho_{\mathbf{x}}}{2^n-1}\right] + \mathbb{E}_{\mathbf{x},\mathbf{y}}\left[(1-\frac{2^n\rho_{\mathbf{x}}}{2^n-1})\mathcal{L}(\mathcal{M}(\mathbf{x}),\mathbf{y})\right] \qquad (4.14)
\end{aligned}$$

Note that the term $\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\frac{c\rho_{\mathbf{x}}}{2^n-1}\right]$ is independent of the ranking model. Let $\widehat{\mathcal{M}}^*$ and $\mathcal{M}^*$ be the global minimizers of the empirical risk functions $\widehat{R}'_{\mathcal{L}}$ and $\widehat{R}_{\mathcal{L}}$, respectively. Therefore, we have:

$$\widehat{R}'_{\mathcal{L}}(\widehat{\mathcal{M}}^*) - \widehat{R}'_{\mathcal{L}}(\mathcal{M}^*) \leq 0 \qquad (4.15)$$

According to Equation (4.14), we can rewrite the above inequality as:

$$\begin{aligned}
&\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\left(1-\frac{2^n\rho_{\mathbf{x}}}{2^n-1}\right)\left(\mathcal{L}(\widehat{\mathcal{M}}^*(\mathbf{x}),\mathbf{y}) - \mathcal{L}(\mathcal{M}^*(\mathbf{x}),\mathbf{y})\right)\right] \leq 0 \\
&\Rightarrow \min_{\rho_{\mathbf{x}}}\left\{1-\frac{2^n\rho_{\mathbf{x}}}{2^n-1}\right\}\mathbb{E}_{\mathbf{x},\mathbf{y}}\left[\left(\mathcal{L}(\widehat{\mathcal{M}}^*(\mathbf{x}),\mathbf{y}) - \mathcal{L}(\mathcal{M}^*(\mathbf{x}),\mathbf{y})\right)\right] \leq 0 \\
&\Rightarrow (1-\frac{2^n\rho_{max}}{2^n-1})\mathbb{E}_{\mathbf{x},\mathbf{y}}[\mathcal{L}(\widehat{\mathcal{M}}^*(\mathbf{x}),\mathbf{y})] \leq \mathbb{E}_{\mathbf{x},\mathbf{y}}[\mathcal{L}(\mathcal{M}^*(\mathbf{x}),\mathbf{y})] \\
&\Rightarrow \widehat{R}_{\mathcal{L}}(\widehat{\mathcal{M}}^*) \leq \frac{1}{1-\frac{2^n\rho_{max}}{2^n-1}}\widehat{R}_{\mathcal{L}}(\mathcal{M}^*) \qquad (4.16)
\end{aligned}$$

Therefore, the true empirical risk for $\widehat{\mathcal{M}}^*$ is upper bounded by $\widehat{R}_{\mathcal{L}}(\mathcal{M}^*)/(1-\frac{2^n\rho_{max}}{2^n-1})$. This completes the proof. □

Theorem 3 shows that the ratio of empirical risk for the global minimizer of the weak risk to to the one for the global minimizer of the true risk is upper bounded by $\frac{1}{1-\frac{2^n\rho_{max}}{2^n-1}}$.

Remark 1. Theorem 3 shows that if the minimum empirical risk on the true labeled data is equal to 0, then the model $\widehat{\mathcal{M}}^*$ is the global minimizer of the empirical risk on the true labeled data. Therefore, if $\widehat{R}_{\mathcal{L}}(\mathcal{M}^*) = 0$, then any symmetric ranking loss $\mathcal{L}$ is robust to non-uniform noise.

4.1.6 A Study of Pairwise Loss Functions

In the following, we study a number of pairwise loss functions to identify the ones that satisfy the SSPL constraint introduced by Theorem 1. Without loss of generality, assume that $\mathcal{M}(x) \in [0, 1] : \forall x$ which can be obtained via a sigmoid function. With some relaxation of notation throughout this section, for a pair of objects (o, o') with feature vectors (x, x'), let $s_{o \geq o'} = \mathcal{M}(x) - \mathcal{M}(x')$ denote the score of o being ranked higher than o' by the ranking model $\mathcal{M}$. Therefore, $s_{o \geq o'} = -s_{o < o'} \in [-1, 1]$. Let $y_{o \geq o'} \in \{-1, 1\}$ be a pairwise label indicating whether o should be ranked higher than o' or not.

Lemma 1. In pairwise learning to rank if $\mathcal{M}(x) \in [0, 1] : \forall x$, hinge loss and mean absolute error (L_1 loss) satisfy the SSPL constraint. On contrary, cross entropy loss and mean square error (L_2 loss) do not satisfy the SSPL constraint.

Proof. In the following, we study the loss functions mentioned in the lemma one by one.

- **Hinge loss**: Hinge loss, also known as the max-margin loss, is defined as follows:

$$\max\{0, 1 - y_{o \geq o'} s_{o \geq o'}\} \tag{4.17}$$

where $y_{o \geq o'} \in \{-1, 1\}$. Given the above definition, we have:

$$\begin{aligned} &\mathcal{L}_{hinge}(s_{o \geq o'}, -1) + \mathcal{L}_{hinge}(s_{o \geq o'}, 1) \\ &= \max\{0, 1 - s_{o \geq o'}\} + \max\{0, 1 + s_{o \geq o'}\} \end{aligned} \tag{4.18}$$

Since $s_{o \geq o'} \in [-1, 1]$, the above equation is equal to 2, and thus hinge loss satisfies the SSPL constraint.

- **Mean absolute error or $\mathbf{L_1}$ loss**: Given the definition of L_1 loss, we have:

$$\begin{aligned} &\mathcal{L}_{MAE}(s_{o \geq o'}, -1) + \mathcal{L}_{MAE}(s_{o \geq o'}, 1) \\ &= |1 - s_{o \geq o'}| + |-1 - s_{o<o'}| + |-1 - s_{o \geq o'}| + |1 - s_{o<o'}| \end{aligned} \tag{4.19}$$

Since $s_{o \geq o'} = -s_{o<o'} \in [-1, 1]$, then the above equation is equal to 4, and thus mean absolute error satisfies the SSPL constraint.

- **Cross entropy loss**: Given the definition of cross entropy loss, we have:

$$\begin{aligned} &\mathcal{L}_{ce}(s_{o \geq o'}, -1) + \mathcal{L}_{ce}(s_{o \geq o'}, 1) \\ &= \log p_{o \geq o'} + \log(1 - p_{o \geq o'}) \\ &= \log p_{o \geq o'}(1 - p_{o \geq o'}) \end{aligned} \tag{4.20}$$

Note that we should use the pairwise probability $p_{o \geq o'}$ for the cross entropy loss. The above equation is neither constant, nor bounded. Therefore, the cross entropy loss function does not satisfy the SSPL constraint.

- **Mean square error or $\mathbf{L_2}$ loss**: Given the definition of L_2 loss, we have:

$$\begin{aligned} &\mathcal{L}_{MSE}(s_{o \geq o'}, -1) + \mathcal{L}_{MSE}(s_{o \geq o'}, 1) \\ &= 2(1 - s_{o \geq o'})^2 + 2(1 + s_{o \geq o'})^2 \end{aligned} \tag{4.21}$$

Thus, the L_2 loss function does not satisfy the SSPL constraint. However, the above equation is bounded by $[4, 8]$.

□

4.1.7 Experiments

In this section, we first provide the results on a synthetic noisy data, and then experiment with real weak supervision data for the ad-hoc retrieval task.

4.1.7.1 Evaluation on Synthetic Data

In the first set of experiments, we empirically validate our theoretical findings. To do so, we create a synthetic data based on the MQ2008 dataset, which is a part of the LETOR 4.0 dataset [128].[3] Each training and test instance in this dataset contains 46 features, extracted via various retrieval techniques. In our experiments, we performed 5-fold cross-validation based on the queries. We trained a fully-connected feed-forward neural network on the training data. The network consists of two hidden layers with 500 and 100 neurons. Relu was used as the non-linear activation function in the hidden layers, and sigmoid was used as the output activation. We trained the model using a pairwise setting; any pair of documents with different labels with respect to each query was considered as a pairwise training instance. The model was trained for one epoch using the Adam optimizer [80], and the learning rate was selected from $\{0.0001, 0.0005, 0.001\}$ based on the performance on the validation set. The batch size was set to 128. We evaluated the performance of the models in terms of normalized discounted cumulative gain for the top 20 documents (nDCG@20).

Uniform noise: In the first set of experiments, we applied a uniform noise on the training data. The pairwise noise probability was swept from 0.0 to 0.45. Note that the pairwise noise should be less than 0.5 which means that the weak labeler should perform better than random. We evaluated models with the same neural architecture, but different pairwise loss functions. The results are plotted in Figure 4.1. As depicted, performance of the models based on the L_2 and cross entropy (CE) loss func-

[3] The LETOR dataset is available at `https://www.microsoft.com/en-us/research/project/letor-learning-rank-information-retrieval/`.

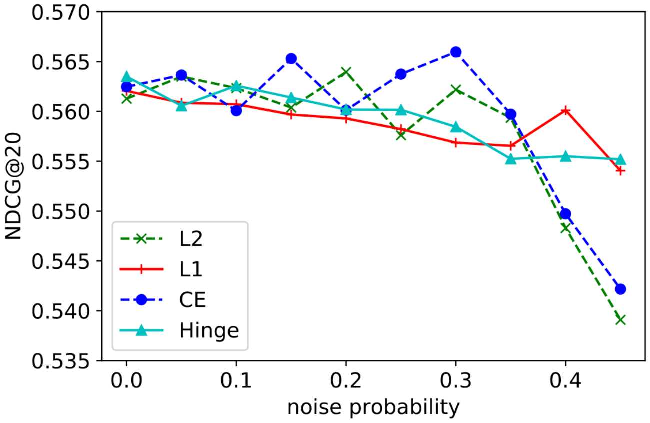


Figure 4.1: The retrieval performance on MQ2008 with respect to the uniform noise probability ($\rho < 0.5$).

tions significantly drops when the noise probability increases. However, the models with the L_1 and the hinge loss functions are robust to uniform noise. This observation empirically validates our theoretical findings on the robustness of symmetric ranking losses to uniform noise.

Non-uniform noise: In the next set of experiments, we applied non-uniform noise on the training data. In other words, the probability of noise varies across queries. We swept the maximum noise probability from 0.0 to 0.9. The results are plotted in Figure 4.2. According to this plot, the performance of all the models significantly drop when the maximum noise probability passes the 0.7 threshold. Figure 4.2 shows that the models with symmetric ranking losses (i.e., L_1 and hinge loss) perform substantially better than those based on the L_2 and cross entropy loss functions, when the maximum noise probability is high (i.e., > 0.7).

4.1.7.2 Evaluation on Weak Supervision Data

In this subsection, we focus on a real weak supervision setting for ad-hoc retrieval. To do so, we trained a fully-connected feed-forward pairwise model (FNRM), exactly the same as the RankModel introduced in [42]. Network parameters were optimized

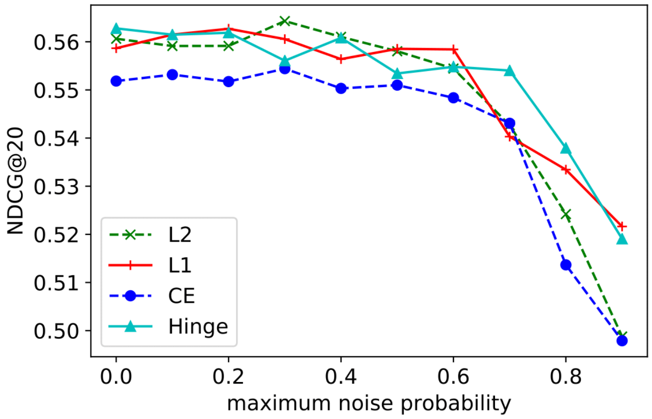


Figure 4.2: The retrieval performance on MQ2008 with respect to the non-uniform maximum noise probability.

using the Adam optimizer [80]. In this experiment, the learning rate and the batch size were selected from $[5e{-}5, 1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3]$ and $\{32, 64, 128\}$, respectively. The hidden layer sizes were selected from $\{100, 300, 500\}$. We initialized the word embedding matrix by pre-trained GloVe [123] vectors learned from Wikipedia dump 2014 plus Gigawords 5.[4] The embedding dimensionality was set to 300. All retrieval experiments were carried out using the Galago search engine [30]. We performed 2-fold cross-validation over the queries in each collection for hyper-parameter tuning.

We collected our training queries from AOL query logs [121]. We only used the query strings, and no session and click information was obtained from the query logs. We filtered out the navigational queries containing URL substrings, i.e., “http”, “www.”, “.com”, “.net”, “.org”, “.edu”. All non-alphabetical characters were removed from the queries. As a sanity check, we made sure that no queries from the training set appear in our evaluation query sets. Applying all of these constraints leads to over 6 million unique queries as our training query set. We used query likelihood [126] with Dirichlet prior smoothing [191] as the weak supervision signal. In

[4] https://nlp.stanford.edu/projects/glove/

Table 4.1: Retrieval performance of weakly supervised neural ranking models (FNRM) with different loss functions. The highest value per column is marked in bold, and the superscripts 0/1/2 denote statistically significant improvements compared to QL/FNRM-CE/FNRM-L2, respectively.

Method	Robust			ClueWeb		
	MAP	P@20	nDCG@20	MAP	P@20	nDCG@20
QL	0.2499	0.3556	0.4143	0.1044	0.3139	0.2294
FNRM-CE	0.2743^{0}	0.3682^{0}	0.4272^{0}	0.1233^{0}	0.3286^{0}	0.2308^{0}
FNRM-L2	0.2765^{0}	0.3696^{0}	0.4290^{0}	0.1214^{0}	0.3271^{0}	0.2315^{0}
FNRM-L1	$\mathbf{0.2831}^{012}$	$\mathbf{0.3769}^{012}$	$\mathbf{0.4333}^{012}$	0.1321^{012}	$\mathbf{0.3368}^{012}$	0.2386^{012}
FNRM-Hinge	0.2815^{012}	0.3752^{012}	0.4327^{012}	$\mathbf{0.1329}^{012}$	0.3351^{012}	$\mathbf{0.2392}^{012}$

more detail, for each training query, we retrieved 100 documents from the target evaluation collection using the query likelihood model and created our pairwise training instances based on the query likelihood scores.

We evaluate our models using the following two TREC collections: The first collection, Robust, consists of thousands of news articles and is considered as a homogeneous collection. Robust was previously used in TREC 2004 Robust Track. The second collection, ClueWeb, is a challenging and large-scale web collection containing heterogeneous and noisy documents. ClueWeb (i.e., ClueWeb09-Category B) is a common web crawl collection that only contains English web pages. ClueWeb was previously used in TREC 2009-2012 Web Track. We cleaned the ClueWeb collection by filtering out the spam documents, using the Waterloo spam scorer[5] [29] with the threshold of 60%. Stopwords were removed from all collections. For Robust, TREC topics 301-450 & 601-700, and for ClueWeb, topics 1-200 were used for the experiments.

Results: The results reported in Table 4.1 show that all weakly supervised models outperform the query likelihood (QL) model, which is our weak labeler. The results

[5] http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/

demonstrate that the models with L_1 and hinge loss functions significantly outperform the models with L_2 or cross entropy as the loss function. The statistical differences are computed using the two-tailed paired t-test at 95% confidence interval ($p\_value < 0.05$). Recall that the L_1 loss and the hinge loss satisfy the SSPL constraint, while the L_2 and the cross entropy loss function do not. This is an empirical validation on real weak supervision data for our theory presented in this chapter.

4.2 A Standalone Neural Ranking Model with Weak Supervision

In this section, we introduce SNRM as a standalone neural ranking model based on learning a latent sparse representation. Section 4.2.1 describes the desirable properties of the model. Section 4.2.2 details the neural network architecture. The general design of our model consists of three phases: (i) the training phase (Section 4.2.3), (ii) the offline inverted index construction phase (Section 4.2.4), and (iii) the retrieval phase with or without feedback (Sections 4.2.5 and 4.2.6).

4.2.1 Design Desiderata

Designing a standalone neural ranking model that can retrieve documents from a large-scale collection, instead of re-ranking a small set of documents returned by a first stage ranker, could potentially be used in various retrieval engines. Conventional term matching retrieval models, such as TF-IDF, BM25, or query likelihood, derive query and document representations based on the atomic units of natural languages (e.g., words); hence they carry the desirable property of *sparsity* which helps efficient retrieval on large-scale collections. They each use an inverted index built on the search collection. Based on the idea of using an inverted index for efficient retrieval, we introduce SNRM, a neural network model for learning standalone rankers based on *latent sparse* representations.

Similar to representation-focused neural ranking models, e.g., [42, 74, 150, 188], our neural framework consists of three major components: the query representation ϕ_Q, the document representation ϕ_D, and the matching function ψ. The retrieval score for each query-document pair (q, d) is then computed as follows:

$$\text{retrieval score}(q, d) = \psi\ (\phi_Q(q),\ \phi_D(d)) \tag{4.22}$$

Our goal to build a responsive standalone ranker leads to a number of requirements on these three components:

- ϕ_D is a query independent component, and thus can be computed offline. In contrast to previous neural ranking models that learn low-dimensional dense representations, ϕ_D should output a high-dimensional sparse vector. This will allow us to construct an inverted index based on the learned representations.

- The components ϕ_Q and ψ should be run at query or inference time (i.e., when producing the ranked output). To obtain a real-time ranking model, it is necessary to minimize the amount of computation in these two components.

- The i^{th} element of ϕ_Q and ϕ_D should represent the same latent feature. In other words, the two components ϕ_Q and ϕ_D should provide representations in the same semantic space, which also implies $|\phi_Q| = |\phi_D|$. Considering the first property, this property also means that the query representation should be sparse.

- The matching function ψ should return zero if there is no overlap between the indices of non-zero elements in ϕ_Q and ϕ_D. This property, in addition to the previous one, is necessary to build and use an inverted index for efficient retrieval.

- Functions ϕ_Q and ϕ_D use the full capacity of the output space and do not map queries and documents to a particular subspace. In other words, for each dimension in the output space, there exists at least one query q or document d such that $\phi_Q(q)$ or $\phi_D(d)$ leads to a non-zero value for that dimension.

We claim that if the design of a neural ranking model satisfies the above properties, and the learned query and document representations are sufficiently sparse, we are then able to construct an inverted index from the learned representation by ϕ_D and retrieve documents in response to a given query as a standalone ranking, without the need for a first stage retrieval model.

4.2.2 Network Architecture

Since the representations learned by ϕ_Q and ϕ_D in SNRM should represent the same semantic space, we design our architecture to share parameters between these components. Furthermore, we need different levels of sparsity in the query and document representations. This is mainly motivated by efficiency, as a key feature of a standalone ranker, since the matching function will compute the retrieval score for all the documents that have a non-zero value for at least one of the non-zero latent elements in the query representation. So the sparser the representation of the query is, the less computation is expected. Furthermore, queries are intuitively much shorter than documents and contain less information, so to express queries, it makes sense to have fewer non-zero elements in their representations compared to the document representations. Using a simple fully-connected feed-forward network for implementing these components is not an appropriate solution as in this case (due to the shared parameters between ϕ_Q and ϕ_D) both query and document representations would become similar in terms of sparsity ratio, i.e., percentage of zero elements (see Equation (4.25)). This results in long queries in the learned latent space, leading to an expensive matching function.

Based on this argument, we need a representation learning model in which the representation sparsity is a function of the input length. We propose an architecture based on ngram representation learning. The intuition behind our model is that we first learn a sparse representation for each continuous n words in the given document or query. The learned sparse representations are then aggregated via average pooling. In fact, our document representation (and similarly our query representation) is obtained as follows:

$$\phi_D(d) = \frac{1}{|d| - n + 1} \sum_{i=1}^{|d|-n+1} \phi_{\text{ngram}}(w_i, w_{i+1}, \cdots, w_{i+n-1}) \tag{4.23}$$

where $w_1, w_2, \cdots, w_{|d|}$ denote the terms appearing in document d with the same order. ϕ_{ngram} learns a sparse representation for the given ngram. The query representation is also computed using the same function. This approach provides two important advantages:

- The number of representations averaged in Equation (4.23) has a direct relationship with the input document/query length. Therefore, the level of the sparsity in the final learned representations depends on the length of the input text. This results in more sparse representations for queries in comparison to documents, which is desired.

- Using the sliding window for encoding the input words as a set of ngrams helps to capture the local interaction among terms, hence the model considers term dependencies. Term dependencies have been widely known to be useful for improving the retrieval performance [106].

We model our ngram representation function ϕ_{ngram} by a fully-connected feed-forward network that reads the input words using a sliding window over the sequence of their embeddings and encodes their ngrams. To this end, we first collect the embedding vectors for each term in the given ngram from an embedding matrix $\mathcal{E} \in \mathbb{R}^{|V| \times m}$ where V and m denote the vocabulary set and the embedding dimensionality, respectively. As discovered in Chapter 3, using relevance-based embedding vectors trained to optimize IR objectives can lead to significant improvements, compared to those trained by general-purpose word embedding vectors, such as word2vec [107] or GloVe [123]. Therefore, $\mathcal{E}$ is part of our network parameters that is learned in an end-to-end training setting. The collected n embedding vectors for the given ngram are then concatenated and fed into a stack of fully-connected layers. These fully-connected layers have an hourglass structure that forces the information of the input data to be compressed and passed through a small number of units in the middle layers that are meant to learn the low dimensional manifold of the data. Then, the number of hidden units increases in upper layers to give us a high-dimensional output, e.g., $20,000$. Note that in terms of the structure and the dimension of layers, the model looks like an autoencoder. However, unlike autoencoders, we have no reconstruction loss. We will discuss our training in Section 4.2.3. We employ rectified linear unit (ReLU) as the activation function in our model.

Our neural architecture for query and document representation can also be seen as a multi-layer one-dimensional convolutional network in which the window sizes for all layers except the first one are set to 1. The window size for the first layer is equal to n, and the strides are all set to 1. Figure 4.3 illustrates how a sequence of words is mapped to a latent sparse representation. This building block of the model is then used in the offline process of encoding documents to sparse latent representations to build an inverted index and also used at inference time to encode submitted queries. The architecture of this block is parallelizable, which supports an efficient procedure for encoding queries at inference time.

We define the matching function ψ as the dot product of the query and document representations. It can be simply proved that dot product has all the properties mentioned earlier in Section 4.2.1. Making the model as efficient as possible is our main reason behind using such a simple function to measure the query-document

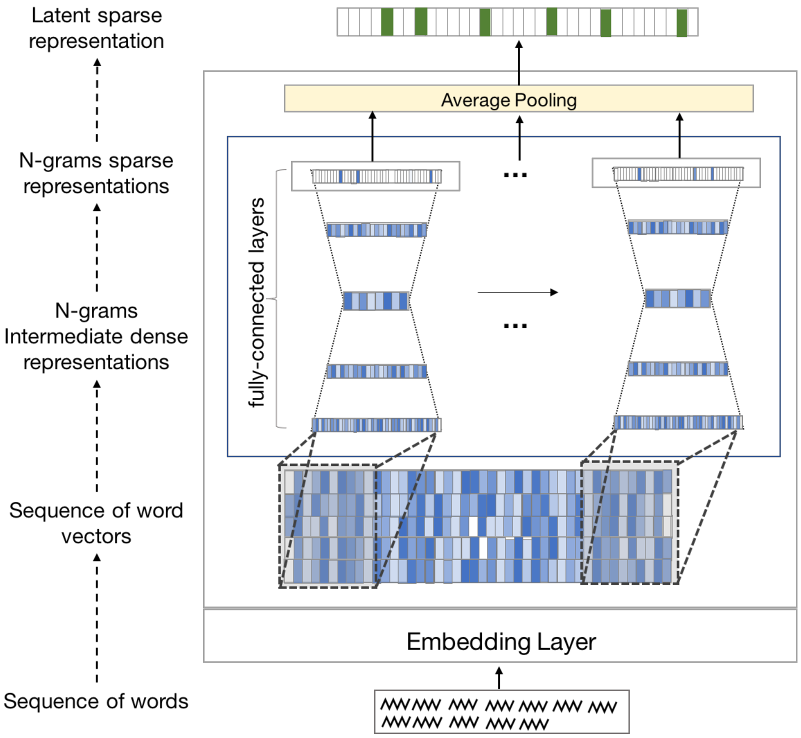

Latent sparse representation
N-grams sparse representations
N-grams Intermediate dense representations
Sequence of word vectors
Sequence of words
Average Pooling
fully-connected layers
...
Embedding Layer

Figure 4.3: Learning a latent sparse representation for a document.

relevance scores. This component plays a key role in efficiency of the model, since it is frequently used at inference time (see Section 4.2.5).

4.2.3 Training

To train the SNRM framework we have two objectives: the retrieval objective with the goal of improving retrieval accuracy and the sparsity objective with the goal of increasing sparseness in the learned query and document representations.

Let $T = \{(q_1, d_{11}, d_{12}, y_1), \cdots, (q_N, d_{N1}, d_{N2}, y_N)\}$ denote a set of N training instances; each containing a query string q_i, two candidate documents d_{i1} and d_{i2}, and a label $y_i \in \{-1, 1\}$ indicating which document is more relevant to the query. In the following, we explain how we optimize our model to achieve both of the mentioned goals.

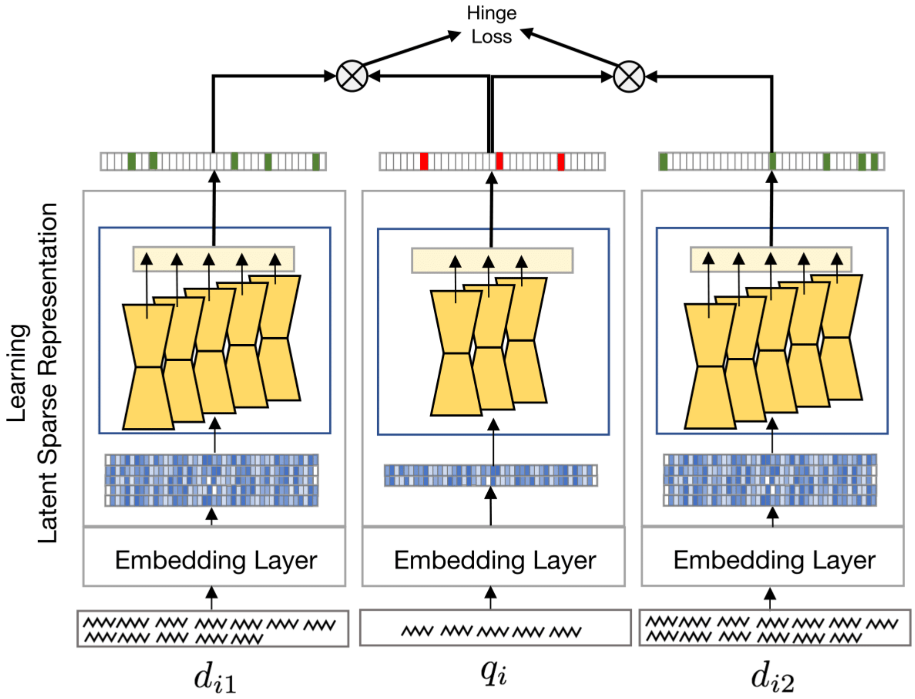


Figure 4.4: General schema of the SNRM at training time.

Retrieval Objective: We train our model using a pairwise setting as depicted in Figure 4.4. Motivated by our theoretical results presented in Section 4.1, we employ hinge loss (max-margin loss function) that has been widely used in the learning to rank literature for pairwise models [91]. Hinge loss is a linear loss function that penalizes examples violating a margin constraint. The hinge loss for the i^{th} training instance is defined as follows:

$$\mathcal{L} = \max\left\{0, \epsilon - y_i\left[\psi(\phi_Q(q_i), \phi_D(d_{i1})) - \psi(\phi_Q(q_i), \phi_D(d_{i2}))\right]\right\} \tag{4.24}$$

where ϵ is a hyper-parameter determining the margin of hinge loss.

Sparsity Objective: In addition to improving the retrieval accuracy, our model aims at maximizing the *sparsity ratio*, which is defined as follows:

$$\text{sparsity ratio } (\vec{v}) = \frac{\text{total number of zero elements in } \vec{v}}{|\vec{v}|} \tag{4.25}$$

Defining $0^0 = 0$, maximizing the sparsity ratio is equivalent to minimizing the L_0 norm:

$$L_0(\vec{v}) = \sum_{i=1}^{|\vec{v}|} |\vec{v}_i|^0 \tag{4.26}$$

Therefore, minimizing the L_0 norm for the final query and document representations is also one of our main objectives. However, the L_0 norm is non-differentiable, which makes it impossible to train our model with the backpropagation algorithm. In fact, L_0 minimization is a non-convex optimization problem, and even finding a solution that approximates the true minimum for L_0 is NP-hard [114]. Therefore, a tractable surrogate loss function should be considered. An alternative would be minimizing L_1 norm (i.e., $L_1(\vec{v}) = \sum_{i=1}^{|\vec{v}|} |\vec{v}_i|$). Although it is clear that we can minimize L_1 as a term in our loss function, it is not immediately obvious how minimizing L_1 would lead to sparsity in the query and document representations. Employing the L_1 norm to promote sparsity has a long history, dating back at least to 1930s for the Fourier transform extrapolation from partial observations [37]. L_1 has been also employed in the information theory literature for recovering band-limited signals [47]. Later on, sparsity minimization has received significant attention as a method for hyper-parameter optimization in regression, known as the Lasso [160].

The theoretical justification for the hypothesis that L_1 minimization would lead to sparsity in our model relies on two points: (1) the choice of rectified linear unit (ReLU) as the activation function forces the non-positive elements to be zero ($\text{ReLU}(x) = \max\{0, x\}$), and (2) the gradient of $L_1(\vec{v})$ for an element of $\vec{v}$ is constant and thus independent of its value. Therefore, the gradient optimization approach used in the backpropagation algorithm [140] reduces the elements of the query and document representation vectors independent of their values. This moves small values toward zero and thus the desired sparsity is obtained.

Loss Function: The final loss function for the i^{th} training instance is defined as follows:

$$\mathcal{L}(q_i, d_{i1}, d_{i2}, y_i) + \lambda \; L_1(\phi_Q(q_i)||\phi_D(d_{i1})||\phi_D(d_{i2})) \tag{4.27}$$

where $||$ means concatenation. The hyper-parameter λ controls the sparsity of the learned representations.

Training with Weak Supervision: Following Dehghani et al. [42], we train our model with weak supervision that benefits from the pseudo-labels obtained by an existing retrieval model, called weak labeler. In more detail, given a large set of queries and a collection of documents, we first retrieve the documents for each training query q_i using the query likelihood retrieval model [126] with Dirichlet prior smoothing [192] as our weak labeler. Each training instance $(q_i, d_{i1}, d_{i2}, y_i)$ is then obtained by sampling two candidate documents from the result list or one from the result list and one random negative sample from the collection. y_i is defined as:

$$y_i = \text{sign}\left(p_{QL}(q_i|d_{i1}) - p_{QL}(q_i|d_{i2})\right) \tag{4.28}$$

where p_{QL} denotes the query likelihood probability.

4.2.4 Inverted Index Construction

The training phase in SNRM is followed by an inverted index construction phase. In this phase, as shown in Figure 4.5, we first feed each document in the collection into the trained document representation component. We look at each index of the learned representation as a "latent term". In other words, let M denote the dimensionality of document representation, e.g., $20,000$. Thus, we assume that there exist M latent terms. Therefore, if the i^{th} element of $\phi_D(d)$ is non-zero, then the document d would be added to the inverted index for the latent term i. The value of this element is the weight of the latent term i in the learned high-dimensional latent vector space.

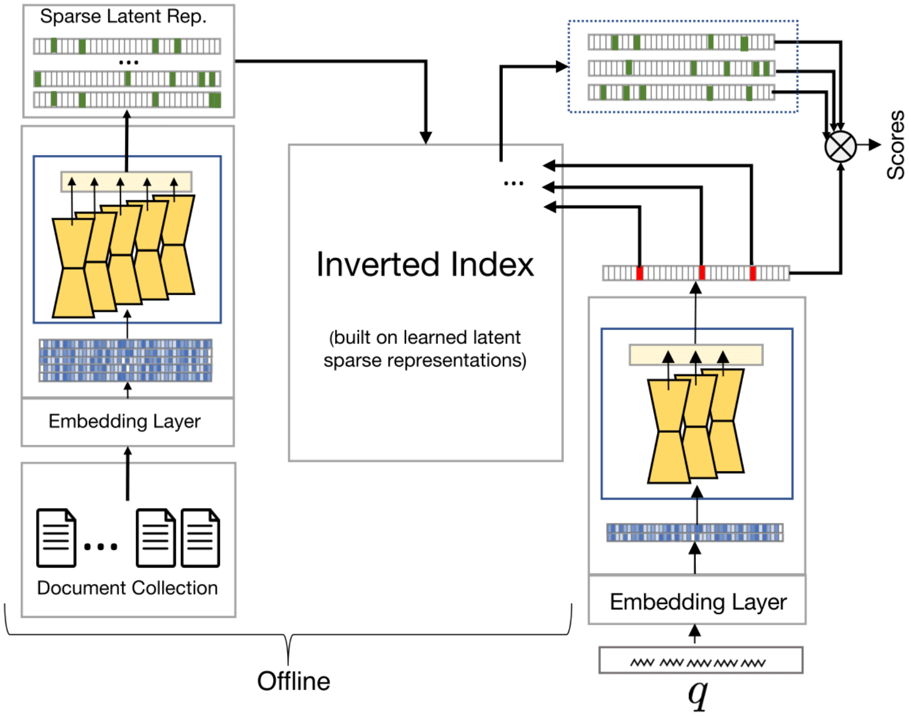


Figure 4.5: General schema of the SNRM after training.

Since documents of a collection can be assumed to be independent, the inverted index construction phase will be memory efficient, and we can feed documents to the document representation component using mini-batches. Note that in case of incorporating new documents, there is no need to train the network from scratch; we can obtain representation for every new document. In Section 4.2.8, we have some experiments that support this claim. However, due to the temporal property of natural languages and invention of new vocabulary terms, the model can be re-trained, periodically. This indicates that SNRM is applicable in real-world scenarios.

4.2.5 Retrieval

As illustrated in Figure 4.5, at inference time, we first feed the query q into the query representation sub-network ϕ_Q to obtain the learned sparse representation $\vec{q}$. Given the dot product definition of ψ at training time, the retrieval score for each document d at inference time is computed as:

$$\text{retrieval score}(q, d) = \sum_{\vec{q_i}|>0} \vec{q}_i \vec{d}_i \tag{4.29}$$

which is a summation over the non-zero elements of $\vec{q}$. With the constructed inverted index, the documents with non-zero elements in the i^{th} index can be retrieved efficiently. Similar to the existing term matching retrieval models, such as query likelihood and BM25, retrieval scores can be computed via the Map Reduce framework [38].

4.2.6 Pseudo-Relevance Feedback

Pseudo-relevance feedback (PRF) is an approach that assumes that the top retrieved documents in response to a given query are relevant, and uses those documents for query expansion. This is also known as a local approach for query expansion [173]. PRF has been proven to be highly effective in various retrieval tasks [30]. We can also take advantage of PRF in SNRM. To do so, we use the Rocchio's relevance feedback algorithm [135] for vector space models. Let $\{d_1, d_2, \cdots, d_k\}$ be the top k documents retrieved by SNRM in response to the query q. The updated query vector is computed as:

$$\vec{q}^* = \vec{q} + \alpha \frac{1}{k} \sum_{i=1}^{k} \vec{d}_i \tag{4.30}$$

where α controls the weight of the feedback vector. Following previous work on PRF [87, 99, 185, 191], we only keep the top t terms with the highest values in the updated query vector $\vec{q}^*$. We then retrieve documents as described in Section 4.2.5.

4.2.7 SNRM Summary

In this subsection, we look at the proposed model as a whole and discuss what can actually be learned by the model. From a high-level perspective, SNRM first maps each text to a dense representation in a low-dimensional semantic space and then transforms it to a high-dimensional sparse representation. Based on our retrieval function (see Equation (4.29)), our model retrieves documents based on the "bag of latent terms" assumption. However, query and document representations are obtained by aggregating ngram representations that capture local term dependencies. Therefore, what SNRM does is mapping the input text from a natural language in which sequences of words matter to a new "latent language" in which term sequences should not play a significant role. Conceptually speaking, SNRM is learning a new "language" with new "vocabulary" terms as its atomic components, and we can benefit from the sparsity of these atomic components in our retrieval framework.

4.2.8 Experiments

In the following we empirically evaluate and analyze SNRM. We first introduce the data, the experimental setup, and the evaluation metrics. We then report a range of experiments to evaluate and better understand the proposed model.

Data: Similar to Section 4.1.7.2, we use Robust and ClueWeb to evaluate SNRM. For weak supervision training, we again use the AOL query logs [121]. For more information about the collection, the training query set, and the pre-processing steps, refer to Section 4.1.7.2.

Experimental Setup: We implemented and trained our models using TensorFlow [1].[6] The network parameters were optimized with Adam [80] based on the backpropagation algorithm [140]. In our experiments, the learning rate and the batch size were selected from $[5\text{e}{-5}, 1\text{e}{-4}, 5\text{e}{-4}, 1\text{e}{-3}, 5\text{e}{-3}]$ and $\{32, 64, 128\}$, respectively. Two or three hidden layers were used for the ngram representation network (ϕ_{ngram}) where n is set to 5. The hidden layer sizes were selected from {100, 300, 500}. The output layer size was also chosen from $\{5\text{k}, 10\text{k}, 20\text{k}\}$.[7] We select the parameter λ (see Equation (4.27)) from the [1e−7, 5e−9] interval. The dropout keep probability

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

[7] Due to the GPU memory constraints, we used a maximum of 20k dimensions.

was selected from $\{0.6, 0.8, 1\}$. We initialized the embedding matrix $\mathcal{E}$ by pre-trained GloVe [123] vectors learned from Wikipedia dump 2014 plus Gigawords 5. The embedding dimension was set to 300. All retrieval experiments were carried out using the Galago search engine [30].[8] We performed 2-fold cross-validation over the queries in each collection for tuning the hyper-parameters of our models as well as baselines.

Evaluation Metrics: To study the effectiveness of SNRM, we report four standard evaluation metrics: mean average precision (MAP) of the top ranked 1000 documents, precision of the top 20 retrieved documents (P@20), normalized discounted cumulative gain [75] calculated for the top 20 retrieved documents (nDCG@20), and recall in the top 1000 documents (Recall). Statistically significant differences of MAP, P@20, nDCG@20, and Recall values were computed using the two-tailed paired t-test with Bonferroni correction at a 95% confidence level.

Experiment I: Comparison of SNRM against the baselines: We compare our model against the following baselines:

- **QL:** The query likelihood retrieval model [126] with Dirichlet prior smoothing [192]. The smoothing parameter μ is a hyper-parameter selected from $\{100, 300, 500, 1000, 1500, 2000\}$.

- **SDM:** The sequential dependence model of Metzler and Croft [106] that takes advantage of term dependencies using Markov random fields in the language modeling framework. The weight of the unigram query component, the ordered window, and the unordered window were selected from $[0, 1]$ with the step size of 0.05, as the hyper-parameters of the model. We made sure that they sum to 1.

- **RM3:** A variant of the relevance model proposed by Lavrenko and Croft [87] that is considered as a state-of-the-art pseudo-relevance feedback model [99]. We selected

[8] `https://www.lemurproject.org/galago.php`

the number of feedback documents from $\{5, 10, 15, 20, 30, 50\}$, the feedback term count from $\{10, 20, \cdots, 100\}$, and the feedback coefficient from $[0, 1]$ with the step size of 0.05.

- **FNRM:** A pairwise neural ranking model recently proposed by Dehghani et al. [42] (i.e., Rank Model) that uses fully-connected feed-forward networks. This model is based on the bag of words assumption and uses weighted average of word embedding vectors for query and document representations. As suggested in [42], this model reranks the top 2000 documents retrieved by query likelihood. Similar to SNRM, this model is trained using the hinge loss in a pairwise setting. The hyper-parameters of this model as listed in the original paper [42] were optimized exactly in the same way as our model.

- **CNRM:** A neural ranking model based on convolutional networks to take term dependencies into account. In fact, this model uses a convolutional layer on top of the word embedding representations, and then an average pooling layer followed by multiple fully-connected layers is employed. This model is similar to CDSSM [150] and NRM-F [188], except in the use of word embedding instead of trigram hashing. This model is also trained with the same weak supervision data. Hyper-parameter tuning and training setting of this model are similar to those of FNRM.

Table 4.2 reports the results for the proposed model against the baselines. According to the results, neural ranking models trained with query likelihood as the weak supervision signal (i.e., FNRM, CNRM, and SNRM) significantly outperform the query likelihood model. This indicates that the neural models can generalize their observations from the weak labeler. This happens since the query likelihood model is restricted to term matching and thus suffers from the vocabulary mismatch problem, however, these neural models can do semantic matching learned from a large set of data labeled by QL as the weak labeler. This learning strategy makes it possible to

Table 4.2: Performance of the proposed models and baselines. The highest value per column is marked in bold, and the superscripts 1/2/3/4/5/6 denote statistically significant improvements compared to QL/SDM/RM3/FNRM/CNRM/SNRM, respectively.

Method	**Robust**			
	MAP	**P@20**	**nDCG@20**	**Recall**
QL	0.2499	0.3556	0.4143	0.6820
SDM	0.2524	0.3679^{1}	0.4242^{1}	0.6858
RM3	0.2865^{12}	0.3773^{12}	0.4295^{12}	0.7494^{12}
FNRM	0.2815^{12}	0.3752^{12}	0.4327^{12}	0.7234^{12}
CNRM	0.2801^{12}	0.3764^{12}	0.4341^{123}	0.7183^{12}
SNRM	0.2856^{12}	0.3766^{12}	0.4310^{12}	0.7481^{1245}
SNRM with PRF	**0.2971**^{123456}	**0.3948**^{123456}	**0.4391**^{123456}	**0.7716**^{123456}

Method	**ClueWeb**			
	MAP	**P@20**	**nDCG@20**	**Recall**
QL	0.1044	0.3139	0.2294	0.3286
SDM	0.1078	0.3141	0.2320	0.3385^{1}
RM3	0.1068	0.3157	0.2309	0.3298
FNRM	0.1329^{123}	0.3351^{123}	0.2392^{13}	0.3426^{123}
CNRM	0.1286^{123}	0.3317^{123}	0.2337^{1}	0.3345^{13}
SNRM	0.1290^{123}	0.3336^{123}	0.2351^{13}	0.3393^{135}
SNRM with PRF	**0.1475**^{123456}	**0.3461**^{123456}	**0.2482**^{123456}	**0.3618**^{123456}

train generalized neural models with no labeled training data. They can potentially improve their weak labelers. The results achieved by SNRM and RM3 are comparable on the Robust collection, while SNRM significantly outperforms RM3 on the ClueWeb collection. Our results also suggest that SNRM performs on par with the FNRM and CNRM baselines, in terms of MAP, P@20, and nDCG@20. It is important to keep in mind that these two baselines are expensive (or even infeasible) to run on large set of documents and thus, as suggested in [42], they only re-rank the top 2000 documents retrieved by query likelihood as the first stage ranker. Therefore, their performance is bounded by the first stage performance, in terms of recall. However, our model, which is able to retrieve documents from its own constructed latent

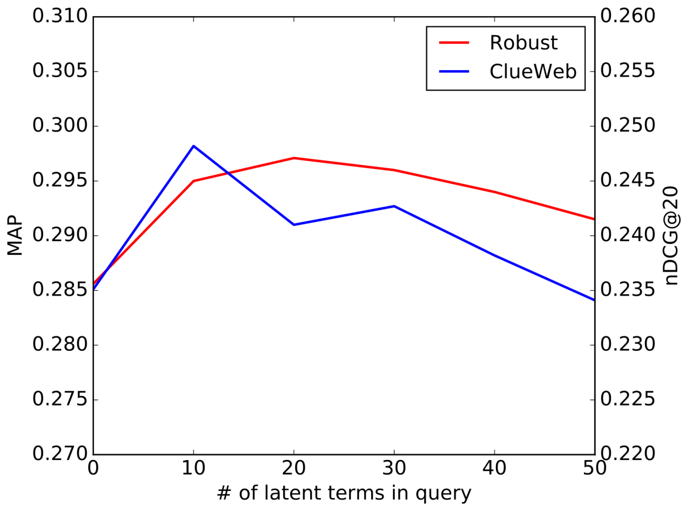


Figure 4.6: Sensitivity of SNRM with PRF to the number of non-zero elements in the updated query vector.

inverted index, can bring up additional relevant documents, even those with no strict vocabulary overlap with the query. It is notable that the Recall@2000 for the QL model is 0.7409 and 0.3551 on Robust and ClueWeb, respectively. Interestingly, the Recall@1000 achieved by SNRM on the Robust collection is higher than the obtained recall by QL on the top 2000 documents. Since FNRM and CNRM are re-ranking the top 2000 documents, our Recall@1000 is even higher than the upper-bound value that can be achieved by any re-ranking baselines, including FNRM and CNRM, on the Robust collection. Note that given the shallow-depth pooling done for assessing the ClueWeb documents, it is not an ideal collection for studying recall-oriented metrics.

It is also worth noting that having a stack of rankers is a common practice in large-scale real-world search engines. Although most existing neural ranking models focus on re-ranking as the final ranker in the stack, our model can be used as an early stage ranker, and improving recall is desirable in such a ranker. Although our learning objective does not directly optimize recall, our model improves the baselines in terms of this metric. This indicates that the learned "latent terms" may carry semantic information that is useful for information retrieval purposes.

Experiment II: Analyzing SNRM with Pseudo-Relevance Feedback: In the next set of experiments, we study the effect of PRF in the new semantic space learned by SNRM. We selected the hyper-parameters α (the feedback weight in Equation (4.30)), the number of feedback documents, and the number of feedback "terms" (the number of non-zero elements in the updated query vector) using the same cross-validation procedure explained earlier in the experiments.

According to Table 4.2, PRF shows promise in the learned latent space. The reason is that it uses local information obtained from the top retrieved documents. In fact, the top retrieved documents help us to find a better query representation compared to the one obtained by the original short query. SNRM with PRF outperforms all the baselines, including RM3. All the improvements are statistically significant.

The performance of the proposed model with respect to the number of of non-zero elements in the updated query vector (i.e., parameter t) is shown in Figure 4.6. As suggested by TREC 2004 Robust Track and TREC 2009-2012 Web Track, we use MAP for Robust and nDCG@20 for ClueWeb as the main evaluation metrics. According to Figure 4.6, the best value for parameter t is 10 for ClueWeb and 20 for Robust, hence dependent on the collection.

Experiment III: Analyzing Sparsity in SNRM: Increasing the sparsity in the leaned representations is one of the objectives of the model and we translate this into minimizing the L_1 norm. In order to study whether minimizing the L_1 norm promotes sparsity in the representations, we plot the L_1 norm of the learned representations, as well as the sparsity ratio for the input query and documents with respect to the training steps. The definition of sparsity ratio is given in Equation (4.25). In this experiment, we set the output dimensionality to 10k and the parameter λ (see Equation 4.27) to 1e$-$7. The results for the model trained on the Robust collection are plotted in Figure 4.7. These curves show that decreasing in the L_1 norm increases the sparsity in both query and document representations. Besides this observation, it is

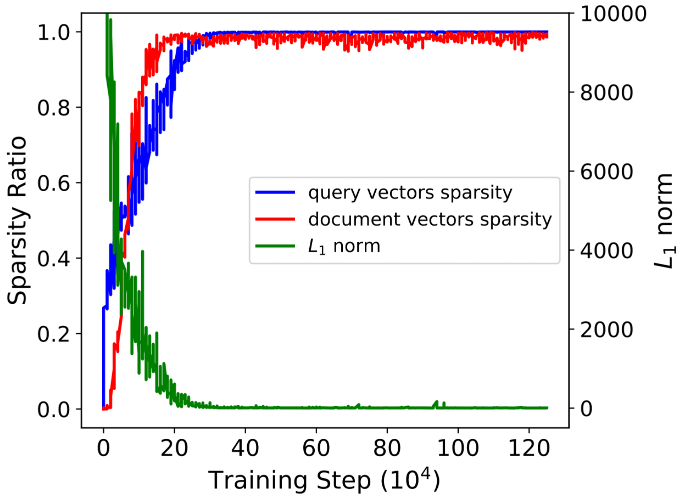


Figure 4.7: Sparsity ratio for query and document representations plus the L_1 norm with respect to the training steps, for SNRM trained on the Robust collection with $10,000$ output dimensionality and $\lambda = 1\text{e}{-7}$.

also shown in Figure 4.7 that the sparsity in query representations is higher compared to the document representations.

Experiment IV: Analyzing the Efficiency of SNRM: As mentioned earlier, the efficiency brought by term matching models comes from the use of inverted index, which is made possible by the sparsity nature of natural languages. Figure 4.8 shows that similar to the natural languages, our learned representations also drawn from a Zipfian-like distribution.[9] In addition, Table 4.3 reports the number of non-zero elements (i.e., the number of unique latent terms) in the representations learned for queries and documents of Robust and ClueWeb. According to the results, the learned query vectors are much sparser than the document vectors. This shows that the input length affects the sparsity of the learned vectors, which is necessary for an efficient retrieval. The sparsity ratio in Robust is higher than that in ClueWeb. This is due to the document length and also the diversity of the documents.

[9] The small sample shown in Figure 4.8 doesn't exactly exhibit the Zipfian distribution, but it shows the skewed nature that makes the use of an efficient inverted index possible.

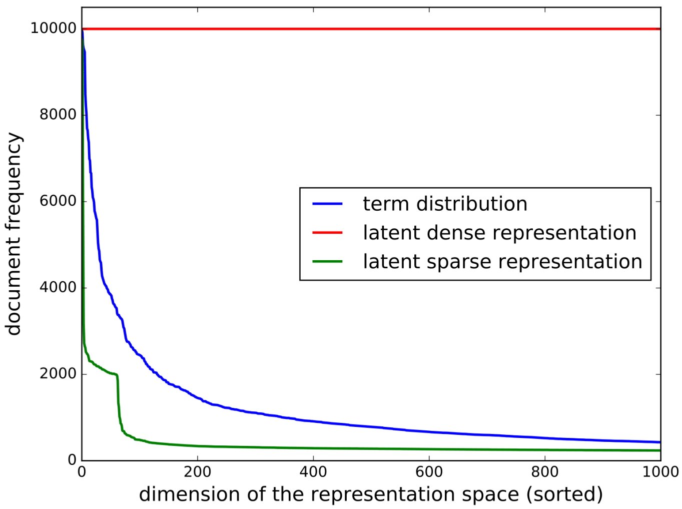


Figure 4.8: The document frequency for the top 1000 dimensions in the actual term space (blue), the latent dense space (red), and the latent sparse space (green) for a random sample of 10k documents from the Robust collection.

Although the shown properties of the learned representations guarantee an efficient use of inverted index for retrieval, we also study the retrieval time for the proposed model compared to a simple term matching model. To do so, we construct a Galago index from the learned representations and implement our retrieval function (see Equation (4.29)) as a retrieval model in Galago. The per query running time for SNRM is computed as the query representation time plus the retrieval time. The query representation time is equal to the running time for pre-processing the query text plus a forward pass through the network from query text to the final sparse representation of the query. The retrieval time is the running time for retrieving documents for the obtained query representation from the constructed Galago index. This experiment was run on a machine with a Core i7-4790 CPU @ 3.60GHz and 32GiB RAM. The average and the standard deviation of retrieval time per query for both Robust and ClueWeb collections are reported in Table 4.4. According to the table, SNRM performs as efficiently as QL, with sub-second response time on the large ClueWeb collection and much faster on the small Robust collection.

Table 4.3: Number of non-zero elements in the query and document representations with $10,000$ output dimensionality.

# Unique	**Robust**		**ClueWeb**	
latent terms...	**Mean**	**Std. dev.**	**Mean**	**Std. dev.**
per document	97.96	447.57	130.24	561.53
per query	3.37	3.04	3.87	4.51

Table 4.4: Efficiency of SNRM compared to query likelihood, in terms of average run time (milliseconds) per query.

Method	**Robust**		**ClueWeb**	
	Mean	**Std. dev.**	**Mean**	**Std. dev.**
QL	35.14	18.43	662.86	746.68
SNRM	46.12	23.11	612.73	640.98

Experiment V: Analyzing the Effect of Sparsity on Retrieval Performance: Figure 4.9 plots the retrieval performance as well as the sparsity ratio achieved by varying the parameter λ (see Equation (4.27)). As shown in the plot, when λ is set to 1e$-$5, the model only focuses on reducing the sparsity, meaning that the learned vector for some queries and documents become all zero. This results in a poor retrieval performance. On the other hand, when representations have enough non-zero elements, the retrieval performance of the model is stable. For instance, in this case, the performance achieved by 99% and 91% sparseness ratios are close.

Experiment VI: Analyzing the Effect of Unseen Documents on the SNRM Performance: As claimed in Section 4.2.4, our approach can be also used to index the documents not seen during the training time. To do so, we randomly removed 1% (over 5k documents) and 5% (over 26k documents) from the Robust collection in two different settings. We then trained SNRM using the obtained collections. Once the training was done, we indexed the whole Robust collection with the trained models.

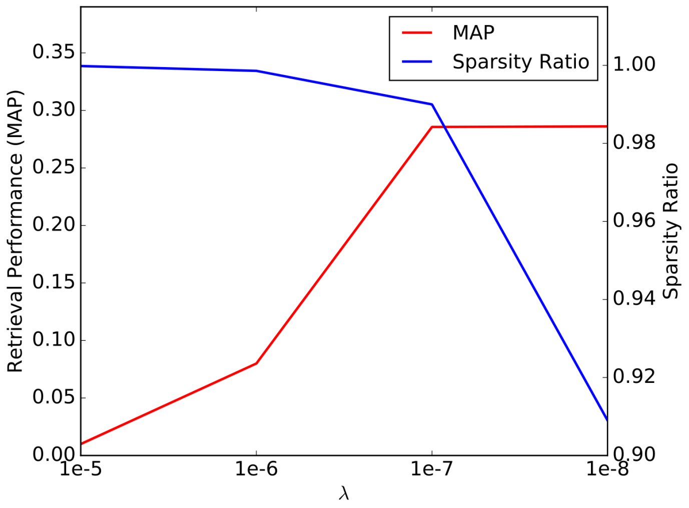


Figure 4.9: Retrieval performance and sparsity ratio on the Robust collection with respect to different values of parameter λ. The output dimensionality was set to 10,000.

Table 4.5: Performance of SNRM on the Robust collection with respect to different amount of random document removal at training time. The superscript $^\triangledown$ denotes significant performance loss in comparison with the setting where no document is removed (i.e., no removal).

% removal	**MAP**	**P@20**	**nDCG@20**	**Recall**
no removal	0.2971	0.3948	0.4391	0.7716
1% removal	0.2953	0.3953	0.4401	0.7691
5% removal	$0.2776^\triangledown$	$0.3807^\triangledown$	$0.4227^\triangledown$	$0.7349^\triangledown$

The results are reported in Table 4.5. According to the results, we do not observe a performance loss when 1% of the documents were omitted from the collection. This indicates the robustness of SNRM in indexing unseen documents, which is a practical point in real-world scenarios where the collections frequently change or new documents are added to the collection (e.g., web). However, 5% document removal leads to significant performance drop. Due to the size of the collection, removing 5% of the documents may result in removing a set of vocabulary terms from the collection, and thus the model cannot learn proper latent representations for the documents containing unseen vocabulary terms. This suggests that in real-world scenarios with dynamic collections, the model should be trained periodically, which is already a common practice.

4.3 Summary

In summary, we studied weak supervision for training neural ranking models in the context of ad-hoc retrieval. We provided a formal modeling of weak supervision using noisy channels. Based on the risk minimization framework, we proved that weakly supervised learning to rank models with symmetric ranking loss functions are robust to noise. In addition to our theoretical results, we designed a standalone neural ranking model that learns sparse representation for each query and document and builds an inverted index for the learned representations. Our model can efficiently retrieve documents from a large corpus and outperform state-of-the-art retrieval models, such as RM3 [87], SDM [106], and FNRM [42]. Notably, our model improves the recall by retrieving the relevant document with few query terms (if any). This also allows the model to improve the retrieval performance, in terms of precision-oriented metrics.

CHAPTER 5

NEURAL QUERY PERFORMANCE PREDICTION

In this chapter, we propose a neural network architecture for the task of query performance prediction. Due to the lack of large-scale training data for this task, we present a *weak supervision* solution by considering multiple existing query performance prediction methods as multiple weak supervision signals. We study how to learn from multiple weak signals and provide state-of-the-art performance prediction results for ad-hoc retrieval. Section 5.1 provides background information about the query performance prediction task. Section 5.2 introduces NeuralQPP, a neural network model for query performance prediction. Section 5.3 explains how to train NeuralQPP without labeled data by exploiting multiple existing performance prediction methods to serve as weak supervision signals. Section 5.4 evaluate the performance of the proposed method.

5.1 Background: Query Performance Prediction

Quality estimation is a fundamental task that can help to improve effectiveness or efficiency in various applications, such as machine translation [156], and automatic speech recognition [21, 115]. When it comes to search engines, the task is called query performance or query difficulty prediction. This task has been widely studied in the IR literature [22, 32, 33, 35, 63, 64, 82, 152, 154, 153, 159, 198, 199]. The task of query performance prediction (QPP) is defined as predicting the retrieval effectiveness of a search engine given an issued query with no implicit or explicit relevance information.

Query performance prediction approaches can be partitioned into two disjoint sets: pre-retrieval and post-retrieval approaches. Pre-retrieval QPP approaches predict the performance of each query based on the content and the context of the query in addition to the corpus statistics. Pre-retrieval predictors are often derived from linguistic or statistical information. Part-of-speech tags, as well as syntactic and morphological features of query terms are among the linguistic features used for query performance prediction [113]. Inverse document frequency and average query term coherence are examples of statistical information used for this task [32, 65]. Hauff et al. [64] provided a through overview of the pre-retrieval QPP approaches.

Alternately, post-retrieval QPP approaches, which are the focus of this chapter, estimate the query performance by analyzing the result list returned by the retrieval engine in response to the query. Carmel and Yom-Tov [22] categorized post-retrieval predictors as clarity-based, robustness-based, and score-based approaches:

- Clarity-based approaches [32, 33] estimate the query performance by measuring the coherence (clarity) of the result list with respect to the collection. These approaches assume that the more focused the result list, the more effective the retrieval.

- Robustness-based approaches predict the query performance by estimating the *robustness* of the result list. Robustness can be measured in various ways. For example, Zhou and Croft [199] measured it based on query perturbation in a Query Feedback (QF) model. In other work, the same authors measured the ranking robustness through document perturbation by injecting noise into the top results [198]. Both query and document perturbations were also studied by Vinay et al. [166]. Aslam and Pavlu [12] studied the ranking robustness based on retrieval engine perturbation. Apart from perturbation approaches, Diaz [44] measured the ranking robustness using the cluster hypothesis [164] by regularizing the retrieval score of each document given its most similar documents. This approach is called spatial autocorrelation.

- A variety of post-retrieval approaches predict the query performance by analyzing the retrieval score distribution, and are commonly referred to as score-based approaches. Among these, the Weighted Information Gain (WIG) of Zhou and Croft [199] and the Normalized Query Commitment (NQC) of Shtok et al. [154] are the most popular models, and are considered state-of-the-art. WIG measures the divergence of the mean retrieval score from the collection score and NQC measures the standard deviation of the retrieval scores normalized by the collection score. Retrieval score distribution has been further employed in other models for the QPP task, e.g., [35, 124]. Most recently, Roitman et al. [138] proposed a bootstrapping approach to provide a robust standard deviation estimator for retrieval scores.

There also exist models that combine multiple predictors from multiple categories. The utility estimation framework (UEF) of Shtok et al. [152] is an example of this QPP family, which is based on statistical decision theory. Making use of both pseudo-effective and pseudo-ineffective reference lists was further studied by Kurland et al. [82], Shtok et al. [153], and Roitman [136].

A number of supervised approaches have been also studied for query performance prediction. For instance, Raiber and Kurland [129] proposed a learning to rank model based on Markov random fields. Most recently, Roitman et al. [137] introduced a supervised combining approach based on coordinate ascent.

5.2 A Neural Network Architecture for Query Performance Prediction

In this section, we propose a *general* query performance prediction framework based on neural networks. The framework is called *NeuralQPP* and is independent of the retrieval engine. NeuralQPP consists of K components that cover different and complementary aspects of query performance prediction. Each component c_i is a sub-network in NeuralQPP, that produces a d_i-dimensional real-valued dense representation, denoted as ρ_i. The learned representations are expected to provide useful information for the query performance prediction task. The obtained ρ_is are then aggregated using an aggregation function Λ which outputs a d-dimensional dense vector. This vector is finally fed into a prediction function Γ that returns a real number representing the predicted performance. All the sub-networks in the NeuralQPP framework are trained simultaneously in an end-to-end fashion.

In summary, the performance of a query is predicted as follows:

$$\Gamma(\Lambda(\rho_1, \rho_2, \cdots, \rho_K)) \tag{5.1}$$

where ρ_i is the output of the component c_i.

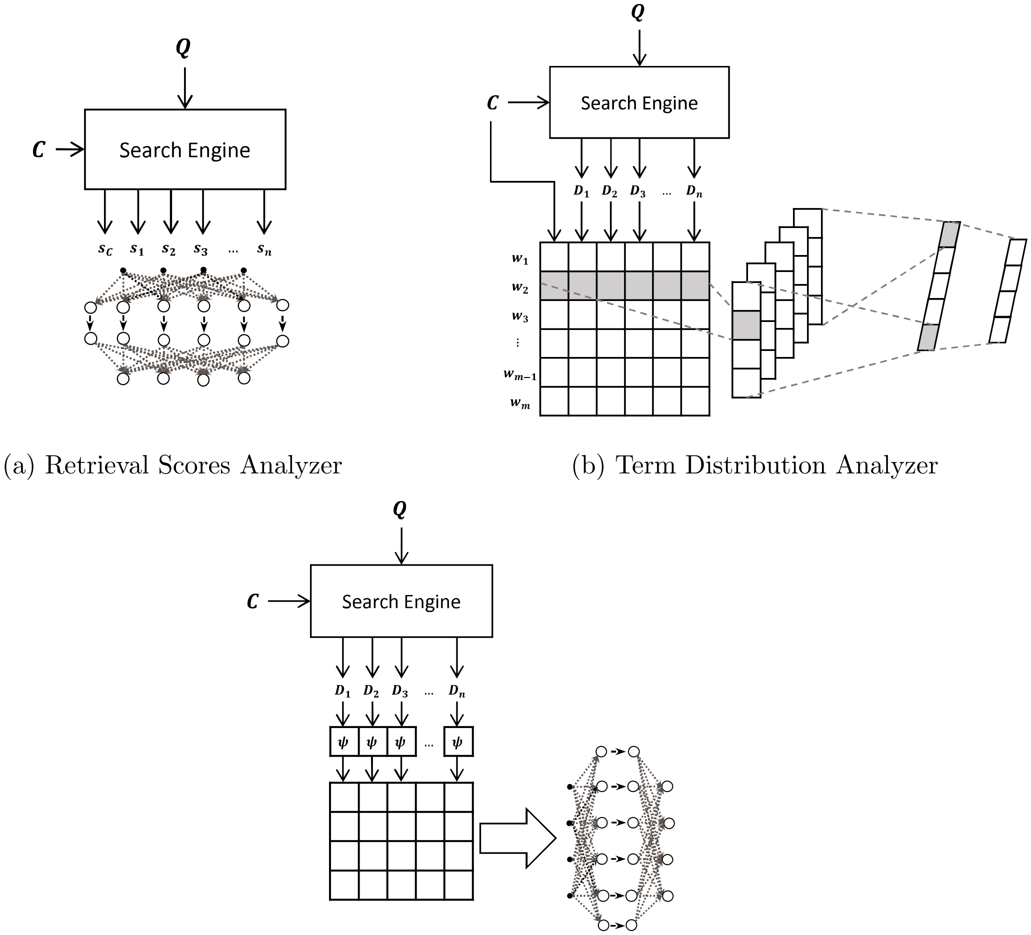
In order to minimize the number of hyper-parameters and perform minimal network engineering, we implement the aggregation function Λ as a weighted average:

$$\Lambda(\rho_1, \rho_2, \cdots, \rho_K) = \frac{1}{K} \sum_{i=1}^{K} \omega_i \rho_i \tag{5.2}$$

where ω_i controls the influence of each component in the final prediction. The network parameters ω_i are trained as part of the NeuralQPP model. Note that this definition of Λ forces the dimensions of all ρ_is to be equal.

We model the function Γ as a fully connected feed-forward neural network that takes the output of Λ and produces a real number representing the predicted performance. We use rectified linear unit (ReLU) as our activation function for hidden layers to learn non-linear functions, and sigmoid for the output layer. To prevent overfitting, we use dropout in all hidden layers. The number of hidden layers and their sizes are hyper-parameters of the model.

In the following, we introduce the three components implemented for NeuralQPP ($K = 3$). The first component analyzes the retrieval score distribution, while the second component considers the term distributions in pseudo-effective and pseudo-ineffective document sets. The last component analyzes the semantic information obtained from the top retrieved documents.


Q
C
Search Engine
s_C s_1 s_2 s_3 … s_n
Q
C
Search Engine
D_1 D_2 D_3 … D_n
w_1
w_2
w_3
⋮
w_{m-1}
w_m
Q
C
Search Engine
D_1 D_2 D_3 … D_n
ψ ψ ψ … ψ

(a) Retrieval Scores Analyzer

(b) Term Distribution Analyzer

(c) Semantic Analyzer

Figure 5.1: NeuralQPP consists of the three components depicted above. The representations learned by each of these components are then aggregated using the arithmetic mean and then fed into a fully-connected feed-forward network that produces a single score for query performance prediction.

5.2.1 Component I: Retrieval Scores Analyzer

Inspired by the score-based approaches described in Chaoter 5.1, such as WIG [199] and NQC [154], our first component, called the retrieval score analyzer, estimates the query performance given the retrieval scores for the top n documents returned by the search engine in response to a query q. As shown in Figure 5.1a, this component takes a vector $\vec{s}$ with $n+1$ dimensions as input, such that:

$$s_i = \begin{cases} \text{score}(q, \mathcal{C}) & \text{if } i = 1 \\ \text{score}(q, D_{i-1}) & o.w. \end{cases} \tag{5.3}$$

where $\mathcal{C}$ and D_{i-1} denote the collection and the $(i-1)^{\text{th}}$ document in the result list returned by the search engine. 'score' denotes the scoring function used by the retrieval engine and score$(q, \mathcal{C})$ is computed as the retrieval score for a document constructed by concatenating all documents in the collection. The order of concatenation does not matter for bag of words models. We feed the constructed vector $\vec{s}$ into a fully-connected feed-forward neural network. In summary, this component computes a non-linear abstract representation of the retrieval score distribution, suitable for the query performance prediction task.

5.2.2 Component II: Term Distribution Analyzer

Inspired by the clarity-based approaches [32] described in Section 5.1, a term distribution analyzer component predicts the query performance using term distribution information. The component's architecture is presented in Figure 5.1b. In this component, we first create a matrix $A = [a_{ij}]$ with $n+1$ columns where the first column corresponds to the collection (as a pseudo-ineffective document set) and each of the remaining n columns corresponds to each of the top n documents retrieved in response to the query q (as pseudo-effective documents). The matrix A has m rows, each corresponding to a vocabulary term from a set W containing the top m terms with the highest cumulative count in the top n retrieved documents ($|W| = m$). Each element of the matrix A is calculated as:

$$a_{ij} = \begin{cases} \Pr(w_i|\theta_C) & \text{if } j = 1 \\ \Pr(w_i|\theta_{D_{j-1}}) & o.w. \end{cases} \tag{5.4}$$

where w_i, θ_C, and $\theta_{D_{j-1}}$ respectively denote the i^{th} term in the vocabulary set W, the collection's unigram language model, and the unigram language model of the $(i-1)^{\text{th}}$ retrieved document. The language models are estimated using maximum likelihood estimation. Since this component is responsible for term distribution analysis, we can assume that vocabulary terms are independent. Therefore a non-linear mapping function $\phi : \mathbb{R}^{n+1} \rightarrow \mathbb{R}^f$ is applied on each row of the matrix A. The parameters of this mapping function are shared for all m terms (rows of the matrix). Indeed, this is similar to applying a convolutional layer with the window size and stride of 1. The input channel size and the filter size are equal to m and f, respectively. Therefore, this layer outputs a $f \times m$ matrix. A sub-sampling phase is further applied. We take the maximum value of the f features learned for each term (max-pooling), which results in a m-dimensional vector. This vector is then fed to a fully-connected feed-forward network for dimension reduction and learning an abstract representation of term distributions, expected to be suitable for query performance prediction.

5.2.3 Component III: Semantic Analyzer

The semantic analyzer component, shown in Figure 5.1c, takes the documents returned by the retrieval engine and measures the query performance based on their distributed representations. For instance, this component can measure how semantically coherent or diverse the returned documents are. The intuition behind this is that coherence and diversity in the returned documents correlate with the ambiguity of the query, since a query may carry multiple meanings or intents. Previous QPP models that analyze the coherence of the result list, e.g., clarity [32], are based on term occurrence (similar to our term distribution analyzer component). Thus, this component provides a novel way of looking at the problem.

In this component, we first represent each document in a latent semantic space, and then learn a set of latent features based on the learned representations. Our document representation function ψ consists of two major functions: (1) an embedding function $\mathcal{E} : V \rightarrow \mathbb{R}^l$ that maps each term from the vocabulary set V to a l-dimensional embedding space, and (2) a global term weighting function $\mathcal{W} : V \rightarrow \mathbb{R}$ that maps each vocabulary term to a real-valued number showing its global importance. The document representation function ψ represents a document $D = \{w_1, w_2, \cdots, w_{|D|}\}$ as follows:

$$\psi(D; \mathcal{E}, \mathcal{W}) = \sum_{i=1}^{|D|} \widehat{\mathcal{W}}(w_i) \cdot \mathcal{E}(w_i) \tag{5.5}$$

which is the weighted element-wise summation over the term embedding vectors. A normalized weight $\widehat{\mathcal{W}}$ is learned for each term using a softmax function as follows:

$$\widehat{\mathcal{W}}(w_i) = \frac{\exp(\mathcal{W}(w_i))}{\sum_{j=1}^{|D|} \exp(\mathcal{W}(w_j))} \tag{5.6}$$

This approach of document representation is based on the bag of words assumption. Despite its simplicity, it was shown to perform well for ad-hoc retrieval tasks [42].

We flatten, concatenate, and feed the representations learned for the top n retrieved documents ($\{\psi(D_1), \psi(D_2), \cdots, \psi(D_n)\}$) into a fully-connected feed-forward network in order to obtain a non-linear abstract representation that demonstrates useful information for query performance prediction extracted from semantic representation of documents in the result list.

5.3 Learning from Multiple Weak Supervision Signals

Here, we describe how to train the proposed neural query performance prediction model with no labeled training data. Indeed, we first propose to train our model using multiple weak supervision signals in Section 5.3.1, and later propose a component dropout technique to regularize our model in Section 5.3.2. Finally, Section 5.3.3 introduces the weak supervision signals employed to train the NeuralQPP model.

5.3.1 Training

Let $\mathcal{M}$ be a retrieval model that retrieves documents from the collection $\mathcal{C}$ in response to a given query. In this work, we propose to train a model with multiple weak supervision signals, which is categorized as an unsupervised learning approach. To do so, we first obtain a set of queries $\mathcal{Q}$ and N weak labelers: N unsupervised query performance prediction models that can provide us complementary information. Predicting the performance of each query $q_i \in \mathcal{Q}$ over the collection $\mathcal{C}$ using the weak supervision signals results in a training set $T = \{(q_i, \pi^n_{\mathcal{M}}(q_i; \mathcal{C}), \mathcal{Y}_i) : q_i \in \mathcal{Q}\}$ where $\pi^n_{\mathcal{M}}(q_i; \mathcal{C})$ denotes the list of the top n documents retrieved by $\mathcal{M}$ in response to the query q_i, and $\mathcal{Y}_i$ denotes a list of predicted performances for q_i by each of the weak supervision signals (thus, $|\mathcal{Y}_i| = N$).

A straightforward solution for learning from multiple weak labels would be casting the problem to learning from a single weak label by aggregating the N weak labels to end up with a single label for each query. This aggregation can be done, for example, by averaging the labels for pointwise settings, or by majority voting for pairwise settings. Another simple solution would be training N distinct models each using one of the weak labels and then aggregating their outputs at inference time by summation.

We argue that neither of these solutions are optimal, since they both incur information loss (which is also justified in our experiments). Therefore, we aim to train our model by optimizing across all weak labels at the same time. Our proposed solution simultaneously optimizes N loss functions, each corresponding to a weak label. Hence, we define our loss function as a linear interpolation of N loss functions:

$$\mathcal{L} = \sum_{k=1}^{N} \alpha_k \mathcal{L}_k \tag{5.7}$$

where $\alpha = [\alpha_1, \alpha_2, \cdots, \alpha_N]$ is a vector of hyper-parameters controlling the influence of each weak label in the final loss function. We investigate two learning settings in our experiments: pointwise and pairwise.

Pointwise learning. In a pointwise setting, we use mean absolute error (MAE) as the loss function. The absolute error for a query q_i in the training set is defined as follows:

$$\mathcal{L}_k(q_i) = |\mathcal{Y}_{ik} - \widehat{P}_k(q_i; \mathcal{M}, \mathcal{C}, \theta)| \tag{5.8}$$

where $\widehat{P}$ denotes the query performance score predicted by our model with the parameter set θ for the given query.

Pairwise learning. Since the task of query performance prediction is often defined and evaluated as a ranking task [32, 154, 199] (ranking queries with respect to their performances), we can train our model using a pairwise setting. Therefore, each training instance consists of a random pair of queries from the training set T. To this end, we employ hinge loss (max-margin loss function) that has been widely used in the learning to rank literature for pairwise models [91]. Hinge loss is a linear loss function that penalizes examples violating the margin constraint. The hinge loss for a query pair q_i and q_j is defined as follows:

$$\mathcal{L}_k(q_i, q_j) = \max\{0, 1 - \text{sign}(\mathcal{Y}_{ik} - \mathcal{Y}_{jk})(\widehat{P}_k(q_i; \mathcal{M}, \mathcal{C}, \theta) - \widehat{P}_k(q_j; \mathcal{M}, \mathcal{C}, \theta))\} \tag{5.9}$$

In the following, we describe how each $\widehat{P}_k$ is computed.

5.3.2 Component Dropout

In training our model with multiple weak labels, we are faced with two major issues: (1) The predictions $\widehat{P}_1, \widehat{P}_2, \cdots, \widehat{P}_N$ should not be equal; otherwise, this would be equivalent to aggregating the weak labels and training the model using the aggregated labels. On the other hand, $\widehat{P}_1, \widehat{P}_2, \cdots, \widehat{P}_N$ should be produced by a single model that would be used at inference time. (2) As can be seen in Section 5.3.3, some of the employed weak supervision signals can be exactly computed using a sub-network of NeuralQPP. For instance, the retrieval scores analyzer component can compute NQC. Therefore, when we use NQC as a weak supervision signal, the network tries to predict the output only based on the retrieval scores analyzer component. This prevents the model from generalizing well. To overcome these two issues, we propose a component dropout approach that also regularizes our model.

Assume that we aim at training a NeuralQPP model with K components using N weak supervision signals. Let p^{ik}_{drop} denote the probability of dropping the effect of the i^{th} component for the k^{th} weak supervision signal ($1 \leq i \leq K$ and $1 \leq k \leq N$).

For each training instance, we construct a binary matrix B with the dimensionality of $K \times N$, whose elements are sampled from Bernoulli distributions as follows:

$$B_{ik} \sim \text{Bern}(1 - p^{ik}_{drop}) \tag{5.10}$$

Each element B_{ik} indicates whether the i^{th} component should be kept for the k^{th} weak supervision signal or not. We make sure that at least one component is kept, such that $\sum_{i=1}^{K} B_{ik} > 0$. Therefore, we compute each prediction $\widehat{P}_k$ as $\Gamma(\Lambda_k(\rho_1, \rho_2, \cdots, \rho_K))$, where Λ_k at training time is computed as:

$$\Lambda_k(\rho_1, \rho_2, \cdots, \rho_K) = \frac{1}{\sum_{i=1}^{K} B_{ik}} \sum_{i=1}^{K} B_{ik}\omega_i\rho_i \tag{5.11}$$

This results in different predictions for $\widehat{P}_1, \widehat{P}_2, \cdots, \widehat{P}_N$ at training time. At the inference time, no component must be dropped, so the matrix B is filled with 1s. In this case, Section 5.11 is equivalent to Section 5.2.

The proposed component dropout technique is similar to the field-level dropout approach, recently proposed by Zamani et al. [188] to prevent over-dependence on high-precision fields (e.g., clicked queries) in neural ranking models for semi-structured documents. The presented technique not also avoids overfitting on a weak supervision signal, but also allows us to use multiple weak signals.

5.3.3 Weak Supervision Signals

To train a generalized model, a natural decision would be to select weak labelers based on different intuitions, assumptions, and consumed information. This enables the neural model to observe complementary information in order to improve its generalization. Hence, we select a clarity-based approach, a score-based approach, and a combining approach (see Section 5.1 for more information about these categories) as the weak supervisors for our NeuralQPP model. The chosen weak labelers are described below:

Clarity: Clarity [32] is one of the early methods for query performance prediction that is based on the language modeling framework [126]. In more detail, this method estimates the query performance as follows:

$$clarity(q; \mathcal{C}, \mathcal{M}) = \sum_{w \in V} p(w|\mathcal{R}_q) \log \frac{p(w|\mathcal{R}_q)}{p(w|\theta_{\mathcal{C}})} \tag{5.12}$$

where V denotes the vocabulary set, $\mathcal{R}_q$ represents the query language model estimated using relevance models [87], and $\theta_{\mathcal{C}}$ represents the reference language model estimated using a maximum likelihood estimation over the whole collection. Intuitively, this model measures the coherence of term distributions in the top retrieved documents with respect to the collection. The term distribution analyzer component (see Figure 5.1b) is expected to learn such a measurement. To generate this weak label, we set the number of retrieved documents to 200.

Normalized Query Commitment (NQC): NQC [154] measures the query performance by computing the normalized standard deviation of the retrieval scores assigned to the top retrieved documents, as follows:

$$NQC(q;\mathcal{C},\mathcal{M}) = \frac{\sqrt{\frac{1}{n}\sum_{D\in\pi^n_{\mathcal{M}}(q;\mathcal{C})}(\text{score}(q,D)-\hat{\mu})^2}}{\text{score}(q,\mathcal{C})} \tag{5.13}$$

where $\pi^n_{\mathcal{M}}(q;\mathcal{C})$ is the result list containing the top n retrieved documents in response to the query q. $\hat{\mu}$ denotes the mean retrieval scores in $\pi^n_{\mathcal{M}}(q;\mathcal{C})$. The intuition behind this model is that query drift can potentially be estimated by measuring the diversity of the retrieval scores. The retrieval score analyzer component (see Figure 5.1a) also gives us such a measurement. To generate this weak label, the number of retrieved documents is again set to 200.

Utility Estimation Framework (UEF): UEF [152] is a theoretical framework by Shtok et al. based on statistical decision theory. UEF estimates the utility that each retrieved document provides w.r.t. the initiated query, as follows:

$$UEF(q;\mathcal{C},\mathcal{M}) \approx \text{sim}(\pi^n_{\mathcal{M}}(q;\mathcal{C}), \pi^n_{\mathcal{M}}(\mathcal{R}_q;\pi^n_{\mathcal{M}}(q;\mathcal{C})))\ \Pr(\mathcal{R}_q|I_q) \tag{5.14}$$

where $\pi^n_{\mathcal{M}}(\mathcal{R}_q;\pi^n_{\mathcal{M}}(q;\mathcal{C}))$ is the original result list re-ranked by the relevance model's estimation of the query language model ($\mathcal{R}_q$). The function 'sim' computes the similarity between two rank lists. We used Pearson's ρ coefficient as a ranking similarity measurement, as is the standard for QPP comparisons. To estimate the representativeness probability $\Pr(\mathcal{R}_q|I_q)$, we used Zhou and Croft's WIG approach [199] for the unigram language model[1]. It is computed as follows:

$$\Pr_{WIG}(\mathcal{R}_q|I_q) \propto \frac{1}{\sqrt{|q|}}\frac{1}{n}\sum_{D\in\pi^n_{\mathcal{M}}(q;\mathcal{C})}(\text{score}(q,D)-\text{score}(q,\mathcal{C})) \tag{5.15}$$

[1] The original WIG approach is based on the term dependence model [106]. This bag-of-words variant is used as our third weak signal, and has been shown to be highly effective [22].

Table 5.1: Statistical properties of the four collections used.

collection	**queries (title only)**	**#docs**	**avgdl**	**#qrels**
Associated Press 88-89	TREC 1-3 Ad-Hoc Track, topics 51-200	165k	287	15,838
TREC Disks 4&5 minus Congressional Record	TREC 2004 Robust Track, topics 301-450 & 601-700	528k	254	17,412
2004 crawl of .gov domains	TREC 2004-06 Terabyte Track, topics 701-850	25m	648	26,917
ClueWeb 09 - Category B	TREC 2009-2012 Web Track, topics 1-200	50m	1506	18,771

Note that the original UEF approach uses multiple samples to produce a relevance model, although, we used a single sampling to produce this weak label. To obtain this weak label, n is set to 100.

5.4 Experiments

To empirically study the effectiveness of the proposed model, we first introduce our datasets and then explain how our model is evaluated. We then describe our experimental setup in detail for further reproducibility. We finally discuss our empirical results.

5.4.1 Data

Collections: We evaluate our models using four TREC collections: The first two collections (AP and Robust) consist of thousands of news articles and are considered homogeneous collections. AP and Robust were previously used in the TREC 1-3 Ad-Hoc Tracks and the TREC 2004 Robust Track, respectively. The second two collections (GOV2 and ClueWeb) are large-scale web collections containing heterogeneous documents. GOV2 consists of the ".gov" domain web pages, crawled in 2004. ClueWeb (i.e., ClueWeb09-Category B) is a common web crawl collection that only contains English web pages. GOV2 and ClueWeb were previously used in TREC 2004-2006 Terabyte Track and TREC 2009-2012 Web Track, respectively. The statistics of these collections as well as the corresponding TREC topics are reported in Table 5.1. We use only the topic titles as queries.

We cleaned the ClueWeb collection by filtering out the spam documents. The spam filtering phase was done using the Waterloo spam scorer[2] [29] with the threshold of 60%. Stopwords were removed from all collections and no stemming was performed.

Training Queries: Similar to Section 4.1.7.2, we use the AOL query logs [121]. For more information about the training query set, and the pre-processing steps, refer to Section 4.1.7.2.

5.4.2 Evaluation

Following prior work on query performance prediction [32, 35, 138, 152, 154, 199], we evaluate our models by computing the correlation between the predicted performance and the actual average precision for the top 1000 documents retrieved per query (AP@1000). In our main experiment, we also report the correlation with the true nDCG values [75] for the top 20 documents. Note that nDCG@20 is a preferred evaluation metric for the ClueWeb collection due to the shallow pooling performed during relevance assessments [24, 96]. We predict the performance of the query likelihood model [126] with Dirichlet prior smoothing ($\mu = 1500$) [192] implemented in the Galago search engine [30].

We use the standard measures from previous research to compute the correlation between predictions and actual performance. Pearson's ρ coefficient as a linear correlation metric and Kendall's τ coefficient as a ranking-based correlation metric are

[2] `http://plg.uwaterloo.ca/~gvcormac/clueweb09spam/`

used. Statistically significant results are reported for two confidence intervals: 95% ($p\_value < 0.05$) and 99% ($p\_value < 0.01$).

Following prior work [154, 153, 138], to evaluate our models as well as the baselines, we first generate 30 equal-size random splits for each collection. In each split, the first fold is used for hyper-parameter optimization using grid search; the hyper-parameter setting that led to the highest Pearson's ρ correlation on predicting the actual AP@1000 values was selected for evaluation on the second fold. This process was repeated for all 30 splits, and the average performance over the second folds are reported. This enables us to perform the paired t-test with Bonferroni correction to identify statistically significant differences between the performance of two QPP models ($p\_value < 0.05$).

5.4.3 Experimental Setup

We implemented and trained our models using TensorFlow [1]. The network parameters were optimized with the Adam optimizer [80] based on the back-propagation algorithm [140]. In our experiments, the learning rate was selected from $[1e{-}5, 5e{-}5, 1e{-}4, 5e{-}4, 1e{-}3, 5e{-}3]$ and the batch size was set to 128. Either two or three hidden layers were used for each sub-networks of the NeuralQPP framework. The layer sizes were selected from $\{100, 300, 500\}$. We select the parameter vector α (see Equation (5.7)) from $\{0.2, 0.4, 0.6, 0.8\}$ and the dropout and the component dropout probabilities (see Equation (5.10)) from $\{0, 0.2, 0.4, 0.6, 0.8\}$. We initialized the embedding matrix $\mathcal{E}$ (see Equation (5.5)) by pre-trained GloVe [123] vectors trained on Wikipedia dump 2014 plus Gigawords 5. The embedding dimension was set to 100.

5.4.4 Results and Discussions

In our experiments, we first evaluate our model against state-of-the-art unsupervised QPP approaches. We then analyze each component of the designed neural network. We further study the influence of incorporating multiple weak supervision

Table 5.2: Performance of query performance prediction models on four collections, in terms of the Pearson's ρ and the Kendall's τ correlations. The results are reported for estimating the performance of each query in terms of AP@1000 as the target metric. The highest value in each column is marked in bold, and the superscripts † / ‡ denote statistically significant improvements compared to all baselines at 95% / 99% confidence intervals.

Method	**Target Metric: AP@1000**							
	AP		**Robust**		**GOV2**		**ClueWeb**	
	P-ρ	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**
Clarity	0.556	0.428	0.410	0.292	0.319	0.205	0.046	0.068
QF	0.585	0.438	0.418	0.274	0.494	0.324	0.273	0.130
WIG	0.547	0.391	0.444	0.294	0.462	0.321	0.238	0.202
σ_k	0.514	0.349	0.438	0.271	0.341	0.292	0.323	0.183
$n(\sigma_{x\%})$	0.524	0.289	0.380	0.218	0.342	0.255	0.188	0.139
NQC	0.540	0.369	0.445	0.283	0.424	0.321	0.308	0.139
SMV	0.505	0.349	0.401	0.274	0.357	0.279	0.326	0.156
RSD	0.594	0.406	0.455	0.352	0.444	0.276	0.193	0.096
CombSum	0.584	0.444	0.483	0.338	0.486	0.317	0.313	0.151
UEF	0.647	0.468	0.565	0.364	0.502	0.315	0.341	0.195
NeuralQPP (Pointwise)	0.613	0.432	0.582†	0.370	0.517	0.322	0.362‡	0.219
NeuralQPP (Pairwise)	**0.697**‡	**0.483**‡	**0.611**‡	**0.408**‡	**0.540**‡	**0.357**‡	**0.367**‡	**0.229**†

signals in the NeuralQPP model. In our final experiments, we explore how NeuralQPP performs in terms of predicting the performance of various retrieval models.

Experiment I: Comparison against the Baselines: In the first set of experiments, we evaluate our models against popular and state-of-the-art query performance prediction baselines, including:

- Clarity [32]: See Section 5.3.3 for the details of this model.
- Query Feedback (QF): a high-performing QPP approach by Zhou and Croft [199] that measures the intersection of the result lists obtained by the original query and an estimated query from the top retrieved documents. Intuitively, this approach

Table 5.3: Performance of query performance prediction models on four collections, in terms of the Pearson's ρ and the Kendall's τ correlations. The results are reported for estimating the performance of each query in terms of nDCG@20 as the target metric. The highest value in each column is marked in bold, and the superscripts † / ‡ denote statistically significant improvements compared to all baselines at 95% / 99% confidence intervals.

Method	**Target Metric: nDCG@20**							
	AP		**Robust**		**GOV2**		**ClueWeb**	
	P-ρ	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**
Clarity	0.437	0.293	0.321	0.221	0.097	0.073	0.040	0.050
QF	0.442	0.296	0.379	0.263	0.322	0.208	0.205	0.084
WIG	0.427	0.287	0.335	0.207	0.308	0.225	0.255	0.175
σ_k	0.428	0.260	0.365	0.240	0.252	0.223	0.300	0.127
$n(\sigma_{x\%})$	0.428	0.210	0.273	0.158	0.232	0.196	0.185	0.082
NQC	0.464	0.290	0.390	0.255	0.257	0.203	0.270	0.102
SMV	0.438	0.266	0.371	0.256	0.335	**0.241**	0.282	0.121
RSD	0.459	0.315	0.394	0.286	0.339	0.203	0.199	0.095
CombSum	0.470	0.318	0.434	0.334	0.349	0.213	0.286	0.162
UEF	0.435	0.302	0.501	0.332	0.311	0.188	0.300	0.159
NeuralQPP (Pointwise)	0.442	0.321	0.528†	**0.350**†	0.346	0.232	0.341‡	0.201‡
NeuralQPP (Pairwise)	**0.492**‡	**0.336**†	**0.539**‡	0.343†	**0.371**†	0.239	**0.352**‡	**0.218**‡

looks at the retrieval engine as a noisy channel and estimates the quality of the channel by measuring the amount of corruption in the result lists.

- Weighted Information Gain (WIG): a popular approach introduced by Zhou and Croft [199] that computes the information gain of the top retrieved documents compared to the collection. WIG predicts the query performance by analyzing the mean retrieval score and the collection's score.

- σ_k: a simple yet effective QPP model that computes the standard deviation of the retrieval scores for the top k retrieved documents. This model has been explored by Pérez-Iglesias and Araujo [124].

- $n(\sigma_{x\%})$: another approach based on the standard deviation, proposed by Cummins et al. [35], that uses a dynamic number of documents per query. This approach computes the standard deviation of the top retrieved documents whose retrieval scores are at least $x\%$ of the one obtained by the highest ranked document.

- Normalized Query Commitment (NQC) [154]: See Section 5.3.3 for the details of this model.

- Score Magnitude and Variance (SMV): a more recent QPP approach by Tao and Wu [158] that considers not only the "variance"[3] over the retrieval scores, but also the score magnitude.

- Robust Standard Deviation (RSD): a recent QPP method proposed by Roitman et al. [138] that computes multiple weighted standard deviations based on a bootstrapping approach. As suggested by the authors, we used WIG [199], as the sample weighting function.

- CombSum: a simple aggregation approach applied on top of the predictions generated by all the above baselines. For this model, we first normalize the scores generated by each model. This is a linear combining approach.

- Utility Estimation Framework (UEF) [152]: See Section 5.3.3 for the details of this model. The choice of representativeness probability in UEF is considered as a hyper-parameter and selected from {NQC, QF, WIG}.

As described in Section 5.4.2, all of the hyper-parameters of the baselines were optimized in the same way as the proposed models. In particular, the number of top retrieved documents is a common hyper-parameter in all of them. We selected this hyper-parameter from $\{5, 10, 15, 20, 25, 50, 100, 300, 500, 1000\}$.

[3] SMV does not exactly compute the variance. Instead, its formulation is more similar to the WIG's [199].

The results for the above baselines and the proposed NeuralQPP model with two training settings (pointwise and pairwise) are reported in Tables 5.2 and 5.3. Note that neither the baselines nor the proposed approaches require labeled training data. The first observation from these two tables is that there is no clear winner among the baselines. From the baseline results, predicting the query performance on the web collections is generally a much harder task when compared to the newswire collections. This is mostly due to the collection size, the variety of topics it covers, and the amount of noise in the collection. Although previous work mostly focused on predicting the performance of queries in terms of average precision for a deep ranking cut-off, we also provide the results for an additional evaluation metric (nDCG@20) that computes the query performance for a shallow ranking cut-off. An interesting observation here is that estimating the query performance in terms of nDCG@20 is a harder task, since the predicted performance of various methods achieve a lower correlation with the actual nDCG@20 values in comparison with the AP@1000 values.

Our second observation from Tables 5.2 and 5.3 is that the pairwise setting in the NeuralQPP model works much better than the pointwise setting. The reason might be related to the nature of the labels we use for training our models. In fact, weak supervision provides a set of noisy labels, and maximizing the likelihood of generating the labels by a neural model is not necessarily a proper choice; instead, optimizing a pairwise loss function gives more freedom to the model to obtain useful features to discriminate two queries. This enables the model to perform much better than the weak labels in almost all cases. A similar observation was made by Dehghani et al. [42] when training neural ranking models with weak supervision signals in the ad-hoc retrieval task.

Our third observation from the results is that NeuralQPP outperforms all the baselines, including the combining approaches, for most collections. The improvements achieved by the NeuralQPP model trained with a pairwise loss function are

Table 5.4: Performance of the NeuralQPP's individual components as well as the Component Dropout technique in case of existing multiple components. The Pearson's ρ and the Kendall's τ correlations are reported for the AP of the top 1000 documents per query. The highest value in each column is marked in bold, and the superscripts ‡ denotes statistically significant improvements compared to all individual components at a 99% confidence interval.

Component(s)	**AP**		**Robust**	
	P-ρ	**K-τ**	**P-ρ**	**K-τ**
Retrieval score analyzer	0.536	0.388	0.442	0.289
Term distribution analyzer	0.541	0.447	0.419	0.319
Semantic Analyzer	0.471	0.353	0.485	0.307
All without Component Dropout	0.636‡	0.462	0.571‡	0.367‡
All with Component Dropout	**0.697**‡	**0.483**‡	**0.611**‡	**0.408**‡
Component(s)	**GOV2**		**ClueWeb**	
	P-ρ	**K-τ**	**P-ρ**	**K-τ**
Retrieval score analyzer	0.351	0.280	0.346	0.188
Term distribution analyzer	0.308	0.212	0.056	0.073
Semantic Analyzer	0.378	0.210	0.090	0.084
All without Component Dropout	0.485‡	0.308‡	0.349	0.193
All with Component Dropout	**0.540**‡	**0.357**‡	**0.367**‡	**0.229**‡

statistically significant in nearly all cases. This indicates the effectiveness of the proposed neural model and training for query performance prediction.

We hereafter focus on predicting AP@1000 for each query. We also focus on the pairwise setting to train our model, which has superior performance.

Experiment II: Analysis of the NeuralQPP Components: As pointed out earlier, we propose three components to develop the NeuralQPP model (see Section 5.2). In this set of experiments, we evaluate the performance of each of these components, individually. We also evaluate the effectiveness of the proposed component dropout technique to regularize the model with multiple components. In this experiment, we train our model with the pairwise setting and with all weak labels.

Table 5.5: Performance of NeuralQPP trained with different weak labels, in terms of correlation with the actual AP@1000 values. The highest value in each column is marked in bold, and the superscript † / ‡ denote statistically significant improvements compared to all individual weak signals as well as both All-MV and All-Ind methods at 95% / 99% intervals.

Weak	**AP**		**Robust**		**GOV2**		**ClueWeb**	
Label	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**
Clarity	0.581	0.443	0.437	0.361	0.330	0.268	0.095	0.103
NQC	0.572	0.369	0.461	0.312	0.439	0.336	0.353	0.184
UEF	0.682	0.480	0.597	0.381	0.527	0.334	0.348	0.191
All-MV	0.694	0.477	0.520	0.351	0.454	0.316	0.357	0.187
All-Ind	0.591	0.454	0.447	0.362	0.384	0.316	0.116	0.128
All-CD	**0.697**	**0.483**	**0.611**†	**0.408**†	**0.540**†	**0.357**‡	**0.367**†	**0.229**‡

Table 5.4 reports the results for this experiment. According to the results, the performance achieved by each of the individual components exhibits high variance. For instance, the term distribution analyzer component achieved the highest performance on the AP collection, however, the performance achieved by the retrieval score analyzer component on the ClueWeb collection are far higher than those achieved by the other two components. This shows that various components can capture different aspects required for achieving improved performance on different collections. Our results also validate that our model successfully makes use of the information captured by multiple components – employing all components together outperforms all individual components. Furthermore, the results indicate that the component dropout technique is effective in all cases and leads to improved performance. All of the improvements obtained by NeuralQPP with all three components and with the component dropout technique are statistically significant when compared to each individual component.

Experiment III: Analysis of the Weak Supervision Signals: In the next set of experiments, we empirically study how employing multiple weak supervision signals (see Section 5.3.3) affects the NeuralQPP performance. To achieve this aim, we use our model with all three components trained by each of the weak supervision signals, individually. The results obtained by these models are reported in the first section of Table 5.5. In the second section, we present the results for two simple models that consider all weak signals (see Section 5.3.1 for more detail). The first model, All-MV, aggregates the outputs for all of the weak labelers. In fact, for each training pair, All-MV selects the label by majority voting over the output of all weak labelers. The second model, All-Ind, learns three separate NeuralQPP models, each by a single weak label, and then produces the final prediction by summing the output of these individually learned models. In the last section of the table, we report the results achieved by our model, All-CD (CD stands for component dropout).

We report the results of this experiment in Table 5.5. By looking at the results presented in both Tables 5.2 and 5.5, we can observe a clear correlation between the performance obtained by each method: Clarity, NQC, and UEF (see Table 5.2), and those achieved by NeuralQPP trained with each of these models as a weak signal (see Table 5.5), respectively. For example, NeuralQPP trained with the Clarity model as the weak signal performs well on the newswire collections, compared to the web collections. The Clarity method itself also behaves similarly. Table 5.5 also demonstrates that training with multiple weak signals leads to a higher generalization, and thus a more accurate performance predictor. These improvements are statistically significant on the Robust, GOV2, and ClueWeb collections.

Our results also demonstrate that the proposed approach for learning from multiple weak labelers is more effective than both All-MV and All-Ind. In particular, All-Ind has poor overall performance, because the models are learned individually and the scale of their outputs are not necessarily in the same scale.[4] Therefore, one

[4] Normalizing the scores for each split improves the performance for All-Ind, however, it performs worse than All-MV. The reason is that All-Ind components are trained separately.

Table 5.6: Performance of the NeuralQPP model for predicting the average precision of the top 1000 documents for popular retrieval models.

Retrieval	**AP**		**Robust**		**GOV2**		**ClueWeb**	
Model	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**	**P-ρ**	**K-τ**
QL	0.697	0.483	0.611	0.408	0.540	0.357	0.367	0.229
TF-IDF	0.671	0.480	0.619	0.412	0.562	0.386	0.355	0.210
BM25	0.718	0.503	0.624	0.412	0.483	0.322	0.310	0.197

of the models may bias the final prediction. As mentioned in Section 5.3.1 both All-MV and All-Ind suffer from information loss provided by multiple weak labels.

Experiment IV: Predicting the Performance of Various Retrieval Models: In the last set of experiments, we study the ability of NeuralQPP to predict the performance of various retrieval models. The results for evaluating the performance of three popular retrieval models: query likelihood (QL) [126], BM25, and TF-IDF. The results reported in Table 5.6 demonstrate that the NeuralQPP performs well in predicting the performance of all of these retrieval models.

5.5 Summary

In this chapter, we introduced learning from multiple weak supervision signals. In more detail, we focused on the query performance prediction task, and used three existing QPP methods to generate weak supervision data. In addition, we proposed a neural network architecture for query performance prediction. Our model analyzes retrieval scores as well as syntactic and semantic coherency of retrieved documents.

Our experiments on four TREC collections (AP, Robust, GOV2, and ClueWeb) demonstrate that the proposed solution trained with a pairwise loss function outperforms state-of-the-art QPP methods for ad-hoc retrieval.

CHAPTER 6

JOINT MODELING OF SEARCH AND RECOMMENDATION

In this chapter, we introduce the task of joint modeling and optimization of search and recommendation. The goal is to learn a retrieval model without any query-document relevance information. We design a model based on relevance-based word embedding, which is a *weakly supervised* model for learning IR-specific word representations (see Chapter 3). In addition, our framework (called JSR) takes advantage of user-item interaction data (e.g., recommendation data) for learning more accurate item representations.In fact, JSR is a multi-task learning framework with two tasks: retrieval and recommendation, where only recommendation data is available. The retrieval objective is based on input text reconstruction, which is unsupervised. In the following, we first motivate the task (Section 6.1), and then propose a joint search-recommendation framework (Section 6.2). Section 6.3 reports the experimental results for both retrieval and recommendation. We finally list a number of potential applications for the proposed framework in Section 6.4.

6.1 Motivation

Learning to rank models have been successfully employed for various retrieval tasks, such as web search, personal search, and question answering [91]. They are mostly trained with either explicit query-document relevance signals, or implicit feedback collected from the user interactions with the retrieval system. Due to the high cost of collecting human annotations, the latter is the natural choice in industrial systems. Existing learning to rank models heavily rely on query-document (or user-query-document) interactions, such as clickthrough data. However, there exist other types of user interactions with the system that can be potentially useful in developing retrieval models. For instance, in many scenarios, users have countless interactions with the items without using the search engine, e.g., browsing and clicking on items, or interacting with the output of recommendation engines.

Although user-item interactions have been previously utilized for user modeling in recommender systems [69, 133, 144] and search personalization [103, 118, 162], learning a retrieval model from user-item interaction data has not yet been explored. Belkin and Croft [15] pointed out the similarities and unique challenges of information retrieval (IR) and information filtering (IF) systems, since 1990s. They concluded that their underlying goals are essentially equivalent, and thus they are two sides of the same coin. This has motivated us to study how to learn a retrieval model from user-item interaction, which has been historically used for developing recommender systems.

Learning an accurate retrieval model from user-item interactions (i.e., collaborative filtering data) has several real-world applications. For example, in media streaming services, such as Netflix[1] and Spotify[2], where rich user-item interaction data exists, building an accurate search engine learned from such large-scale user-item interaction data is desired.

In this chapter, we propose a model that learns to predict user-item interactions (collaborative filtering) and reconstruct the items' description text based on their learned representations. In other words, our model learns item representations that not only simulate user behaviors to predict future user-item interactions, but can also be mapped to a natural language space to be further used for retrieval.

[1] `https://www.netflix.com/`

[2] `https://www.spotify.com/`

6.2 Learning a Retrieval Model from User-Item Interactions

In this section, we formalize the problem definition, and introduce the proposed joint search and recommendation framework (JSR). We further discuss the employed method for speeding up the training procedure. We additionally provide a matrix factorization interpretation of the model in order to find the connection between JSR and the recommender systems literature. We finally summarize our proposed solution.

6.2.1 Problem Statement

Let $U = \{u_1, u_2, \cdots, u_m\}$ and $I = \{i_1, i_2, \cdots, i_n\}$ denote a set of m users and n items, respectively. Let $D = \{y_{ui} : u \in U, i \in I\}$ be a set of user-item interaction data, where y_{ui} represents the label corresponding to the interaction that the user u had with the item i. Labels can be either numeric (e.g., star rating) or binary (e.g., like/dislike). Instead of explicit feedback, y_{ui} can be also captured from user interactions with the recommender system as implicit feedback. Although implicit feedback can be also numeric (e.g., interaction count or time), binary labels are more common, such as clicking on a link, watching a movie, or listening to a music. Therefore, we assume that labels are binary and were collected from implicit feedback, which is the most realistic setting in many applications [48, 68, 132]. Our framework can be easily developed for numeric labels as well.

In addition to user-item interactions, we assume that there is an additional set $I_T = \{t_1, t_2, \cdots, t_n\}$ containing a textual description for each item in I. These descriptions are often easy to collect, for example, from item descriptions, meta-data, or user-generated tags and reviews. Textual item description has been previously used for content-based and hybrid recommendation [20].

Given D and I_T, the goal is to develop a joint search and recommendation model. In more detail, the learned model should be able to (1) predict the future user-item

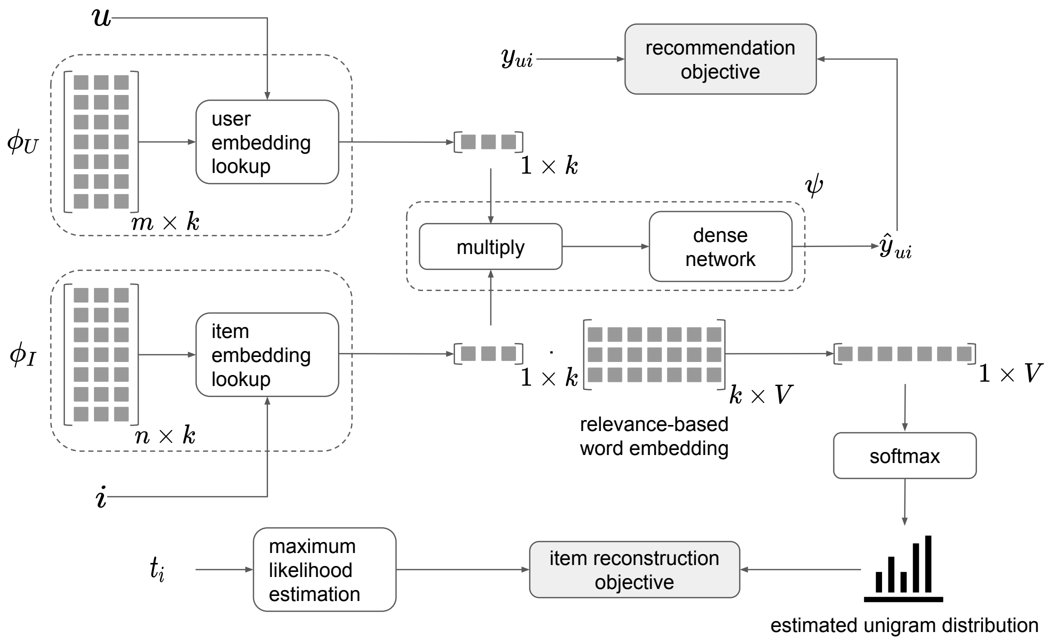


Figure 6.1: A high-level overview of the JSR framework that consists of three major components $\phi_{\mathbf{U}}$, $\phi_{\mathbf{I}}$, and ψ. JSR is trained using two objective functions: a recommendation objective and an item text reconstruction objective.

interactions (i.e., recommendation) and (2) retrieve relevant items given a natural language search query.

6.2.2 The JSR Framework

A natural implementation for joint modeling of search and recommendation is to optimize retrieval and recommendation objectives, simultaneously. However, this cannot be possible without query-item relevance information. Therefore, JSR consists of the following two objectives: a recommendation objective and an item reconstruction objective. The high-level overview of JSR is depicted in Figure 6.1.

6.2.2.1 Recommendation

Our recommendation component is based on collaborative filtering which relies on user-item interactions. Collaborative filtering has shown to be effective in many recommendation scenarios. JSR estimates a recommendation score for each user-item pair as follows:

$$\hat{y}_{ui} = \psi(\phi_U(u), \phi_I(i)) \tag{6.1}$$

where $\hat{y}_{ui}$ is the model's prediction. The component ϕ_U (ϕ_I) learns a k-dimensional latent representation for each user (item). As demonstrated in Figure 6.1, we implement the recommendation component ψ by feeding the Hadamard product of the user and item representations to a fully-connected feed-forward neural network (called the dense network) with few hidden layers. Note that the Hadamard product is the element-wise multiplication of two matrices. We use ReLU as the activation function in the hidden layers and employ dropout in all hidden layers to avoid overfitting. The output activation in the dense network is a sigmoid function. The number of hidden layers, their sizes, and the dropout probability are hyper-parameters of the model.

We formulate the recommendation objective using a pointwise loss function. Following He et al. [68], we use a binary cross-entropy loss function that has shown effective performance in neural collaborative filtering for implicit data. The reason is that some popular loss functions in collaborative filtering, such as mean squared error, assume that the data is drawn from a Gaussian distribution, which does not hold in many settings [141]. In addition to the effectiveness of binary cross-entropy, it is a probabilistic loss derived from the log-likelihood maximization and can be easily combined with our second probabilistic objective. This loss function has been also used for training neural retrieval models in previous work [42, 74, 186].

To formally define the loss function, let us first define b as a mini-batch of training data sampled from the training set D (see Section 6.2.1) expanded with η random negative samples per user-item interaction. η is a hyper-parameter. The loss function for the mini-batch b is defined as:

$$L_{bce} = -\frac{1}{|b|} \sum_{(u,i) \in b} y_{ui} \log \hat{y}_{ui} + (1 - y_{ui}) \log(1 - \hat{y}_{ui}) \tag{6.2}$$

We have also tried pairwise cross-entropy loss, however, no significant improvement has been observed in our experiments. Therefore, we have decided to keep the pointwise model given its simplicity.

6.2.2.2 Item Text Reconstruction

The goal of item reconstruction is to make sure that the learned user and item representations can be mapped into a natural language space, and thus can be used for item retrieval. To this end, we use the textual descriptions of items (i.e., the set I_T; see Section 6.2.1 for more information). In more detail, we maximize the probability of generating the textual description of each item from the learned item representations. We focus on unigram language model representation which has been shown to be effective in information retrieval [126, 190].[3] In more detail, let $\mathcal{E} \in \mathbb{R}^{|V| \times k}$ denote a word embedding matrix with the same dimensionality as the user and item embeddings (k), where V is the vocabulary set. $\mathcal{E}$ is a pre-trained relevance-based word embedding matrix [180]. Relevance-based word embedding models are based on the bag-of-words assumption and represent each word in a k-dimensional space to capture relevance information. The model is trained based on weak supervision and does not require any label data. In fact, relevance-based word embedding models use the relevance models [87] as a weak supervision signal. We use relevance-based word embedding, since the goal of the model is to learn representations that are suitable for further retrieval purposes. For more information on relevance-based embedding, we refer the reader to [180] (see Chapter 3). Multiplication of each item embedding vector $\vec{i} \in \mathcal{I}$ to the transpose of the embedding matrix $\mathcal{E}$ results in a $|V|$-dimensional representation for the corresponding item. Therefore, our model estimates a unigram language model for each item i as follows:

[3] Some models, such as [120], treat users and items as queries and documents and use language modeling approaches for recommendation, which are out of the scope of this work.

$$\theta_i = \text{softmax}(\mathcal{I}_{[i]}.\mathcal{E}^T) \tag{6.3}$$

The aim is to maximize the likelihood of generating the item description text. This is equivalent to minimizing the following cross-entropy for the mini-batch b:

$$L_{mlr} = -\frac{1}{|b|} \sum_{(u,i) \in b} \sum_{w \in t_i} \frac{\text{count}(w, t_i)}{|t_i|} \log p(w|\theta_i) \tag{6.4}$$

where $t_i \in I_T$ is a textual description for item i.

6.2.2.3 Optimization

We train the model using a gradient descent-based optimizer. The parameters that should be learned include the user embedding matrix $\mathcal{U}$, the item embedding matrix $\mathcal{I}$, and the parameters of the dense network for recommendation. Note that the embedding matrix $\mathcal{E}$ is pre-trained and fixed. We use Adam optimizer [80] in our experiments to minimize the following loss function:

$$\mathcal{L} = L_{bce} + \alpha L_{mlr} \tag{6.5}$$

where α is a hyper-parameter controlling the weight of the item reconstruction loss.

6.2.3 Training Efficiency in JSR

Due to the large number of terms in a vocabulary (e.g., 500k), the unigram language model estimated by JSR is computationally expensive, because of the summation in the denominator of the softmax operator. Since this summation should be computed for every single item in the mini-batch at each training step, it substantially slows down the training process. To address this issue, we approximate the softmax operator using *hierarchical softmax*, which has been introduced by Morin and Bengio [112] for neural language modeling and successfully employed by Mikolov et al. [107] for word representation learning. This approximation uses a binary tree structure to represent vocabulary terms. Each leaf corresponds to a unique vocabulary term and there exists a unique path from the root of the tree to each leaf. This path is used for estimating the probability of the vocabulary term representing by the leaf. Since the height of the tree is $O(\log(|V|))$, the complexity of softmax calculation goes down from $O(|V|)$ to $O(\log(|V|))$. This results in a huge improvement in computational complexity. We refer the reader to [110, 112] for the details of hierarchical softmax approximation.

6.2.4 Matrix Factorization Interpretation

Matrix factorization is the dominant approach in collaborative filtering. In this section, we briefly discuss how JSR can be interpreted as a matrix factorization model. For simplicity, assume that the dense network in our model (see Figure 6.1) is simply a linear regression model whose weights are set to 1. Therefore, feeding the Hadamard product of two vectors to this network is equivalent to their inner product. In this case, JSR can be modeled as:

$$\mathcal{U}^*, \mathcal{I}^* = \arg\min_{\mathcal{U},\mathcal{I}} L_1(\mathcal{A}, \mathcal{U}\mathcal{I}^T) + \lambda L_2(\mathcal{T}, \mathcal{I}\mathcal{E}^T) \tag{6.6}$$

where $\mathcal{A} \in \mathbb{R}^{m \times n}$ denotes the user-item interaction matrix, containing the training data. Each row j of the matrix $\mathcal{T} \in \mathbb{R}^{n \times |V|}$ is a unigram language model for the item j, estimated using maximum likelihood estimation. L_1 is a recommendation loss that minimizes the user-item interaction reconstruction error, while L_2 is an item description reconstruction loss that minimizes the distance between the learned item representations and textual descriptions of the items. L_2 can be also seen as a *regularizer* that prevents the collaborative filtering model from overfitting. Item embedding has been previously used by Liang et al. [94] and by Train et al. [161]

to regularize collaborative filtering, however, their models are based on item-item co-occurrences, which is different from ours.

Due to the non-linear operations in the dense network, JSR is the generalized version of this matrix factorization interpretation. It is notable that due to the large size of matrix $\mathcal{T}$, implementing this matrix factorization method is infeasible or difficult, which also motivates our choice of neural network implementation.

6.2.5 Item Retrieval using JSR

Once the model is trained, we compute the language model θ_i (see Equation (6.3)) for all items. To improve the retrieval efficiency, we take the top N vocabulary terms with the highest probability for each item (called $W_{\text{top}}(i)$) and normalize the language model as follows:

$$p(w|\hat{\theta}_i) = \begin{cases} \frac{p(w|\theta_i)}{\sum_{w' \in W_{\text{top}}(i)} p(w'|\theta_i)} & \text{if } w \in W_{\text{top}}(i) \\ 0 & \text{otherwise} \end{cases} \tag{6.7}$$

We constructed an inverted index from each word w in the vocabulary to a list of items as follows:

$$w \rightarrow \Big\{(i, p(w|\hat{\theta}_i)) : \forall i \in I \text{ such that } w \in W_{\text{top}}(i)\Big\} \tag{6.8}$$

To compute the retrieval score for a natural language query q at the test time, we use a KL-divergence retrieval model [85] with Jelinek-Mercer smoothing:

$$\text{retrieval score}(q, i) = \sum_{w \in q} p(w|\theta_q) \log \Big[\lambda p(w|\hat{\theta}_i) + (1 - \lambda) p(w|C)\Big] \tag{6.9}$$

where $\lambda \in [0, 1]$ is the smoothing parameter and $p(w|C)$ denotes the probability of the term in the collection, and can be computed as:

$$p(w|C) = \frac{1}{Z} \sum_{i \in I} p(w|\hat{\theta}_i) \tag{6.10}$$

where Z is a normalization factor. We develop two retrieval models based on different implementation of the query language model (θ_q). Similar to [126], the JSR-QL is our first retrieval model in which $p(w|\theta_q)$ is computed using maximum likelihood estimation (i.e., $p(w|\theta_q) = \frac{\text{count}(w,q)}{|q|}$). The second retrieval model, called JSR-RM, is a pseudo-relevance feedback algorithm based on Lavrenko and Croft's relevance models [87]. In other words, we first retrieve documents using JSR-QL and then compute the relevance language model (i.e., RM3) and finally retrieve documents based on the re-estimated query language model. Refer to [2, 87] for the detailed implementation of RM3.

6.2.6 Summary

The model optimizes two objectives simultaneously, one user-item interaction (recommendation) objective and one item text reconstruction objective. This allows the model to transfer knowledge from user-item interactions to the item representations. We use unigram language model for representing items and estimate it using a hierarchical softmax function which has shown to be highly efficient. The language model is obtained from the learned item representation multiplied by a relevance-based word embedding matrix, which results in item language models that are suitable for retrieval purposes. Our matrix factorization interpretation of the framework shows that item text reconstruction can be seen as a regularization for the recommendation model. Since we use relevance-based word embedding, this regularization is biased towards representations that are useful for retrieval.

Table 6.1: Statistics of the data used in our experiments.

Data	**#users**	**#items**	**#interaction**	**sparsity**	**#queries**
MovieLens 20M	138,493	27,278	20,000,263	99.471%	1236
Amazon					
- Electronics	192,403	63,001	1,689,188	99.986%	989
- Kindle Store	68,223	61,934	989,618	99.977%	4603
- Cell Phones	27,879	10,429	194,439	99.933%	165

6.3 Experiments

In this section, we evaluate the proposed framework using a wide range of datasets. We first introduce our training data and parameter setting details. We further evaluate our model in terms of both retrieval and recommendation performance. We finally present a set of empirical analysis to have better understanding of the results.

6.3.1 Training

This section describes the details for training the JSR framework, including the training data and the experimental setup.

6.3.1.1 Training Data

We study the performance of our model on multiple public datasets. We use *MovieLens 20M* [61] which is the largest version of the MovieLens datasets to date and contains over 20 million total interactions with the minimum interaction of 20 per user. MovieLens 20M is a standard dataset for evaluating collaborative filtering models in the context of movie recommendation.[4] We also use three diverse product categories in the context of e-commerce services. We adopt the five-core Amazon review dataset [66, 104], that covers user-item interactions (review, rating, helpfulness votes, etc) on 24 product categories from May 1996 to July 2014.[5] We have selected

[4] MovieLens 20M can be found at `https://grouplens.org/datasets/movielens/20m/`.

[5] The Amazon review dataset can be found at `http://jmcauley.ucsd.edu/data/amazon/`.

three product categories with different size and sparsity to observe the performance of the model on different conditions. They include Electronics, Kindle Store, and Cell Phones & Accessories. The largest category contains over 1.5 million interactions, while the number of interactions in the smallest category does not reach 200 thousands. The minimum number of interactions per user in these categories is 5. Similar to previous work [48, 68], we binarized the labels in all datasets by representing each user-item interaction with label 1 as an implicit feedback. Table 6.1 reports the statistics of the datasets.

To construct the set I_T (see Section 6.2.1), we concatenated the tags that users have provided for each movie in the MovieLens 20M dataset. For the Amazon datasets, we selected the most helpful review of each item (according to their helpfulness scores provided by users) as its textual description. If there was no review with positive helpfulness score, we randomly selected one of the reviews. We cleaned up the data by removing non-alphabetic characters and stopwords from item descriptions.

Following previous work [68, 92, 132], we adopt a leave-one-out evaluation methodology for evaluation. For each user, we held-out the latest interaction as the test recommendation data and utilized the remaining data for training.

6.3.1.2 Parameter Setting

We implemented our model using TensorFlow [1]. In all experiments, the network parameters were optimized using the Adam optimizer [80]. For hyper-parameter optimization, we selected the latest interaction of each user in the training set as a validation set. We performed grid search and chose the hyper-parameters based on the loss value obtained on the validation set. After the hyper-parameter selection process, we train the model with the chosen hyper-parameter values on the original training set. The learning rate was selected from $\{1 \times 10^{-5}$, 5×10^{-4}, 1×10^{-4}, 5×10^{-4}, $1 \times 10^{-3}\}$. The batch size was selected from $\{32, 64, 128, 256, 512\}$. The dropout keep probability was selected from $\{0.7, 0.8, 0.9, 1.0\}$. The dimensionality of word, user, and item embedding vectors were set to 50 and the word embedding matrix was initialized by a relevance-based word embedding model, called relevance likelihood maximization (RLM) [180]. The embedding vectors were trained using the ClueWeb collection as described in [180]. We set the number of negative samples per interaction (i.e., parameter η) to 4. The number of hidden layers in the dense network and their output sizes were selected from $\{1, 2, 3\}$ and $\{10, 20, 50, 100\}$, respectively.

6.3.2 Evaluating the Retrieval Performance

In this section, we describe our experimental methodology and results for evaluating the retrieval performance of JSR.

6.3.2.1 Evaluation Data

This experiment is challenging, since, to the best of our knowledge, there is no public dataset containing both user-item interactions and query-item relevance information on the same item set. Therefore, we created a dataset for evaluating the model trained on the MovieLens 20M dataset, and used an automatic evaluation methodology employed by [4, 163, 181] for evaluating the models trained on the Amazon datasets.

Movie retrieval dataset: We adapt the known-item search data created by Hagen et al. [60]. The data contains difficult real-world information needs collected from Yahoo! Answers. From this data, we only selected the information needs with the category “Movies”. The relevance judgments contain a single relevant document per information need. The relevant documents were selected from the ClueWeb collection. By manual annotation, we linked each of the relevant documents in the dataset to the corresponding movie ID in the MovieLens 20M dataset. We finally filtered out the queries whose answers were not found in MovieLens 20M. This results in 1236 queries, written by real users in the Yahoo! Answers website, each has exactly one

Table 6.2: Two sample queries and their associated relevant movies from the retrieval dataset.

Query	**Relevant movie**
I can't remember the name of the movie where this teenager is really good at this space shooter arcade game, then one day aliens contact him and he figures out it was training him and he becomes a space fighter pilot.	The Last Starfigher (1984)
It's bugging me because I forgot the name of it. Its about a boy who has elephantitis. And there is a blind girl who feels his face. And then at the end I think he kills himself or something. What is it called I forgot.	Mask (1985)

relevant movie from the MovieLens 20M dataset. Two query examples are listed in Table 6.2.

Product retrieval dataset: The Amazon product data does not contain search queries, thus cannot be directly used for evaluating retrieval models. As Rowley [139] investigated, directed product search queries contain either a producer's name, a brand, or a set of terms describing the product category. Following this observation, Van Gysel et al. [163] proposed to automatically generate queries based on the product categories. To be exact, for each item in a category c, a query q is generated based on the terms in the category hierarchy of c. Then, all the items within that category are marked as relevant for the query q. The detailed description of the query generation process can be found in [4]. Although the queries in this data were automatically constructed, since the query generation process has been done based on observations from real user queries, it has become a standard approach for evaluating product search, and has been used by the research community [4, 57, 163, 181, 195].

6.3.2.2 Evaluation Metrics

In the movie retrieval dataset, there is only one relevant item per query. This characteristic makes many of the common IR evaluation metrics unsuitable. Therefore, for this dataset, we use mean reciprocal rank (MRR) and normalized discounted cumulative gain (nDCG) [75] of the top 10 retrieved items (nDCG@10) as the main evaluation metrics. However, in the Amazon datasets, there exist multiple relevant items per query. In such cases, mean average precision (MAP) and nDCG@10 are used as main evaluation metrics.

Statistically significant differences of performances were computed using the two-tailed paired t-test with Bonferroni correction at a 99% confidence level.

6.3.2.3 Experimental Setup

The hyper-parameters of all the retrieval baselines as well as the proposed model were selected using two-fold cross-validation over queries. To improve reproducibility, the data split was done based on the query IDs (even and odd). In hyper-parameter optimization, the regularization parameter α and the smoothing parameter λ were selected from $(0, 1)$.

6.3.2.4 Results and Discussion

We compare the retrieval performance of the proposed method to the following baselines:

- Query Likelihood (QL) [126]: This is a language model-based retrieval model that uses the Dirichlet prior method [192] for document language model smoothing.

- BM25 [134]: This is a simple yet effective probabilistic retrieval model derived from a 2-Poisson distribution approximation for document modeling.

- Relevance Model (RM3) [87]: This is an effective pseudo-relevance feedback model based on the language modeling framework that uses the top retrieved documents for query expansion.

- Embedding-based Relevance Model (ERM) [178]: This is a state-of-the-art pseudo-relevance feedback method that takes advantage of pre-trained word embedding

Table 6.3: Retrieval performance of JSR and the baselines. The highest value per column is marked in bold, and the superscript * denotes statistically significant improvements compared to all the baselines.

Method	**MovieLens 20M**		**Amazon - Electronics**	
	MRR	**nDCG@10**	**MAP**	**nDCG@10**
QL	0.052	0.114	0.372	0.421
BM25	0.056	0.118	0.351	0.408
RM3	0.064	0.125	0.386	0.443
ERM	0.068	0.123	0.400	0.458
PRP+	0.094	0.193	0.431	0.511
PRP+ & ERM	0.102	0.209	0.457	0.560
JSR-QL	0.121*	0.317*	0.511*	0.608*
JSR-RM	**0.133***	**0.325***	**0.518***	**0.614***
Method	**Amazon - Kindle Store**		**Amazon - Cell Phones**	
	MAP	**nDCG@10**	**MAP**	**nDCG@10**
QL	0.181	0.210	0.236	0.267
BM25	0.185	0.219	0.244	0.275
RM3	0.193	0.219	0.256	0.286
ERM	0.213	0.235	0.284	0.312
PRP+	0.251	0.297	0.328	0.370
PRP+ & ERM	0.306	0.366	0.347	0.381
JSR-QL	0.348*	0.396*	0.366*	0.408*
JSR-RM	**0.362***	**0.407***	**0.381***	**0.425***

vectors. Similar to JSR, ERM also uses the relevance-based word embedding [180] which has shown superior performance to models like word2vec [107] and GloVe [123]. For more information, see Chapter 3.3.1.2.

Since the authors are not aware of any retrieval baseline that use user-item interactions, we adapt the probabilistic relevance propagation (PRP) of Shakery and Zhai [147]. PRP uses hyperlinks in the Web for retrieval score estimation and language model smoothing [148]. Based on the user-item interaction matrix, we constructed a graph of users and items and applied the PRP algorithm. This model utilizes user-item interaction information in retrieval. We call the model PRP+. We also enhance this model by using embedding-based relevance model (ERM) [178] as for pseudo-relevance feedback (called PRF+ & ERM).

Each of the baselines has a number of hyper-parameters, such as smoothing parameter (μ), the number of feedback terms, the feedback coefficient, etc. As mentioned earlier, we tune all of these hyper-parameters using a two-fold cross-validation over the retrieval queries. In all the baselines, we used the same textual descriptions used in training our model as the content of items (see Section 6.3.1 for more information). To have a fair evaluation, we do not consider supervised retrieval models in our baselines, because our model does not use any query-document relevance signal.

The retrieval performance of the methods are presented in Table 6.3. The results indicate the difficulty of the movie retrieval dataset compared to the others. This is due to the nature of the queries (see Table 6.2). Comparing the results obtained by the JSR and PRP+ models against the other baselines demonstrates the importance of user-item interaction for retrieval tasks. According to the table, both JSR models outperform competitive well-tuned retrieval baselines. The improvements are statistically significant in all cases. Moreover, JSR-RM achieves the best retrieval results. The highest performance is achieved on the Amazon - Electronics dataset, which is the largest product dataset, in terms of the number of user-item interactions.

6.3.3 Evaluating the Recommendation Performance

In this section, we evaluate the recommendation performance of JSR and compare it against state-of-the-art collaborative and hybrid filtering models.

6.3.3.1 Evaluation Data

As pointed out earlier in Section 6.3.1, we evaluate the recommendation performance based on a leave-one-out strategy. In more detail, the last interaction of each user was chosen as a test data. We further followed the common strategy of taking 100 random negative sample items per user, as been widely used in the literature [48, 49, 68, 81].

6.3.3.2 Evaluation Metrics

To evaluate the recommendation performance, we use normalized discounted cumulative gain (nDCG) [75] and hit ratio (HR). The cut-off for these metrics is set to 10. HR measures whether the test item is present on the top 10 list. Note that since there is only one relevant item per user in the leave-one-out strategy, HR is equivalent to recall. On the other hand, nDCG is a ranking metric accounting for the position of the hit by assigning higher scores to hits at top ranks. Although nDCG is often used for evaluating items with graded relevance labels, we use this metric to have our results comparable with previous work [48, 68]. We calculated both metrics for each test user and reported the average score. Similar to the retrieval experiments, statistically significant differences of performances were computed using the two-tailed paired t-test with Bonferroni correction at a 99% confidence level.

6.3.3.3 Results and Discussion

We compare the effectiveness of JSR to the following collaborative filtering baselines:

- ItemPopularity: This simple baseline computes the popularity of each item based on the number of interactions on the item in the training set. The items are then ranked based on their popularity at test time. This is a non-personalized method to benchmark the recommendation performance.

- Bayesian Personalized Ranking (BPR) [132]: This is a competitive matrix factorization method, adapted for learning from implicit feedback. BPR takes advantage of a pairwise ranking loss.

- Element-wise Alternating Least Squares (eALS) [66]: This is an effective matrix factorization method for item recommendation with implicit feedback. This baseline takes all unobserved items as negative instances and weights them based on their popularity. It has shown superior performance to the weighted matrix factorization (WMF) method [73] that uses a uniform weighting over negative instances.

- Neural Collaborative Filtering (NCF) [68]: This neural network baseline is a combination of a generalized matrix factorization and a fully-connected feed-forward network that uses a cross-entropy loss function for collaborative filtering with implicit feedback.

We also compare our model to the following hybrid recommendation baselines:

- Collaborative Deep Learning (CDL) [169]: This neural network hybrid recommendation model jointly performs deep representation learning for the content information and collaborative filtering for the user-item interactions.

- Collaborative Filtering with Additional Stacked Denoising Autoencoders (CF-aSDAE) [46]: This hybrid recommendation model uses stacked denoising autoencoders to jointly model deep users and items' latent factors from side information and collaborative filtering from the interaction matrix.

Each of the baselines has a number of hyper-parameters, such as learning rate and the number of latent factors. We tune all of these hyper-parameters using the same procedure as the proposed method using the same validation data (see Section 6.3.1.2).

Table 6.4 reports the recommendation performance achieved by JSR and the baselines. According to the table, BPR, eALS, and NCF are all competitive collaborative filtering baselines, and there is no clear winner among them. NCF outperforms the other baselines in three datasets (MovieLens 20M, Amazon-Kindle Store, and

Table 6.4: Recommendation performance of JSR and the baselines. The highest value per column is marked in bold, and the superscript * denotes statistically significant improvements compared to all the collaborative filtering baselines (i.e., ItemPopularity, BPR, eALS, and NCF).

Method	**MovieLens 20M**		**Amazon - Electronics**	
	nDCG	**HR**	**nDCG**	**HR**
ItemPopularity	0.5226	0.8196	0.3227	0.5044
BPR	0.6086	0.8633	0.4820	0.7639
eALS	0.6171	0.8820	0.4906	0.7834
NCF	0.6247	0.9050	0.4841	0.7624
CDL	0.6321	0.9183	0.5081	0.8021
CF-aSDAE	0.6319	0.9177	0.5109	0.8073
JSR	**0.6345***	**0.9210***	**0.5125***	**0.8100***
Method	**Amazon - Kindle Store**		**Amazon - Cell Phones**	
	nDCG	**HR**	**nDCG**	**HR**
ItemPopularity	0.1875	0.3340	0.2214	0.3766
BPR	0.3881	0.5845	0.3692	0.5038
eALS	0.3863	0.5809	0.3746	0.5152
NCF	0.3910	0.5999	0.3775	0.5203
CDL	0.4310	0.6570	0.4112	0.5898
CF-aSDAE	0.4285	0.6503	**0.4183**	0.6022
JSR	**0.4335***	**0.6627***	0.4176*	**0.6085***

Amazon-Cell Phones), while eALS shows superior performance among the baselines in the Amazon-Electronics data. This might be due to the sparsity of the collections, i.e., Amazon-Electronics is the sparsest dataset (see Table 6.1). The eALS baseline takes negative samples based on the popularity of items, while the other uses a uniform sampling method. The results also show that JSR significantly outperforms all the collaborative filtering baselines in all the datasets. Our model also shows a comparable (and in most cases slightly better) performance to the hybrid recommendation baselines. This indicates that although JSR only uses the item descriptions for regularization, it effectively takes advantage of side information. Note that the main goal of JSR is not utilizing textual descriptions of items for improving the recommendation.

Table 6.5: The top 10 words selected by JSR for four sample movies. The words are sorted in descending order in terms of their weights. The table should be viewed in color.

The Lord of the Rings (1978)	**Batman Returns (1992)**	**Gandhi (1982)**	**The Mask (1994)**
fantasy	batman	documentary	cartoon
magic	character	film	parody
movies	superhero	directed	movie
wizard	horror	prize	black[6]
animation	thriller	award	comic
potter	starring	supporting	comedy
cartoon	fantasy	films	film
fiction	movie	movie	monster
classic	joker	fiction	thriller
novel	comedy	drama	shows

6.3.4 Additional Empirical Analysis

In this section, we provide additional empirical analysis to have a better understanding the model's performance. We first do a case study by looking at the top terms chosen by the model for a few items. We further investigate the impact of different word embedding vectors on the model's performance.

6.3.4.1 Analyzing the Learned Representations

To provide a deeper analysis on the quality of the learned representation, we report the top 10 terms with the highest weight in the learned representations for four sample movies (for the model trained on the MovieLens 20M data) in Table 6.5. In the table, we classify each word into one of the following categories:

- Green: related to the movie genre
- Gray: related to the movie content (e.g., characters, scenes, and specific features)

[6] The Mask is often referred to as "black comedy".

Table 6.6: The corpora used for training the embedding vectors.

ID	**corpus**	**# tokens**
GloVe - Wiki	Wikipedia 2014 & Gigawords 5	6b
GloVe - 840b	Web crawl	840b
GloVe - ClueWeb	Web crawl	70b

- Blue: related to other movies similar to the movie
- Yellow: miscellaneous features (e.g., awards, etc.)
- Red: not directly related (i.e., the word is not directly related to the movie)
- White: general terms (e.g., movie and film)

There are some common words that appear in most movie representations, such as "movie", "film", "films". In fact, due to the nature of the MovieLens 20M dataset, these terms are considered as general terms. There exist more than one green cell in each column of the table, which indicates that JSR pays attention to the genre of the movies. The movie Gandhi has won eight Academy Awards and interestingly, JSR gives high weights to the words such as "prize" and "award" (yellow cells). This might be due to the fact that many users who watched Gandhi were interested in award-winning movies. Surprisingly, we observe the term "potter" among the top 10 terms for the movie The Lord of the Rings. This is most likely selected because the users who watched The Lord of the Rings also watched the Harry Potter movies. In addition, there are some terms that we do not have direct explanations for. For example, it is not clear why the term "comedy" is selected for the movie Batman Returns. They might be due to the user interactions with other movies. They might also be due to the model's errors. The dark comedy aspect of this movie could be also the reason.

Table 6.7: Retrieval performance of JSR with different word embedding initialization. The highest value per column is marked in bold, and the superscript * denotes statistically significant improvements compared to all the baselines.

Embedding	**MovieLens 20M**		**Amazon - All**	
	MRR	**nDCG**	**MAP**	**nDCG**
GloVe - Wiki	0.088	0.177	0.342	0.386
GloVe - 820b	0.104	0.229	0.358	0.415
GloVe - ClueWeb	0.096	0.193	0.351	0.408
RLM	**0.133***	**0.325***	**0.389***	**0.443***

6.3.4.2 Investigating the Impact of Word Embedding Vectors

As mentioned multiple times in this chapter, our model uses relevance-based word embedding. In this section, we investigate the performance of the model when using general-purpose word embedding vectors, such as GloVe [123]. Table 6.7 shows the results for the model with various word embedding matrix. For the sake of space, in this experiments, we report the result for all the Amazon datasets together. The description of the training corpora is presented in Table 6.6. According to the results, relevance-based word embedding (RLM) leads to significantly better retrieval results. The reason is that the objective in such models were specifically designed for information retrieval purposes. Note that none of these word embedding models require label data for training.

6.4 Potential Applications for JSR

In this section, we review a number of potential applications in which the proposed JSR framework can be potentially useful.

Interpretability: Interpretation in machine learning models would lead to better understanding the models' behavior. It helps researchers and engineers identify what has been learned by the model, what biases have affected the model, and how to improve the performance. It is especially desired in collaborative filtering models, since they heavily rely on learning latent features for users and items. Since our model learns a representation that can be mapped to a unigram language model, it can be potentially used for improving the interpretability of filtering models.

Transparency and Explainability: Providing accurate explanations for every recommendation made by a collaborative filtering model has been recently proven to be effective in recommendation performance [67]. In addition to making the users aware of the reasons behind all the suggestions, explainability improves the transparency of the recommender system. Previous work has shown that users prefer to receive an explanation for any recommendation [194]. Learning user and item representations that can be mapped to a natural language space could potentially ease the process of explaining recommendations.

User Profiling: Table 6.5 shows that JSR learns interpretable representations for items. Therefore, it can be also used for user profiling; representing the user's interests with natural language. The system can also allow the user to manually modify the learned natural language representation for improving the recommendation.

Universal Representation Across Domain and Modality: Natural language is a universal representation; meaning that each movie, music, image, book, etc. can be represented using natural language. On the other hand, cross-domain and cross-modal recommendations are still challenging. Mapping multi-domain item representations to a natural language space would potentially close the gap between these domains and modalities.

Conversational Recommendation: Natural language is the most effortless and flexible communication method for human. In conversational recommender systems, e.g., [195], users describe their information needs in order to receive accurate recommendations. Therefore, learning natural language representations for items could be of potential use in conversational recommendation.

6.5 Summary

In this chapter, we introduced joint modeling and optimization of search engines and recommender systems. With a focus on learning without query-document relevance data, we employed weakly supervised relevance-based word embedding in our joint framework, which is based on multi-task learning. In fact, our framework minimizes two objectives, simultaneously: user-item interaction reconstruction and item text reconstruction. Our experiments in the domains of e-commerce and movies demonstrate that substantial retrieval improvements can be achieved using user-item interaction data. In addition to performance improvement, our model has several potential applications, such as transparency and explainability in recommendation, user profiling, and conversational recommendation.

CHAPTER 7

CONCLUSIONS AND FUTURE WORK

This chapter provides a broad summary of our work. We begin by summarizing how to use weak supervision in information retrieval and reiterate our primary contributions. We further conclude by proposing several potential research directions for future work.

7.1 Overview of Weak Supervision for Information Retrieval

Complex machine learning models, including deep neural networks, often require large-scale training data for effective training. However, such large-scale data is not available in many information retrieval tasks [187]. *Weak supervision* is a solution for training machine learning models in case of scarcity in training data. The proposed weak supervision solution consists of the following steps:

(a) **Collecting or generating unlabeled data:** In most applications, unlabeled data is easy to collect or generate. The first step in weak supervision is simply obtaining an unlabeled training set X, which is relatively large-scale and covers a wide range of input instances.

(b) **Generating labels:** In weak supervision, we automatically label the collected unlabeled data. An example of such models in ad-hoc retrieval is term matching models, such as BM25 [134]. Using this model, called weak labeler, we automatically generate labels for the obtained unlabeled data to produce the weak supervision training set $(X, \widehat{Y})$. One can take advantage of multiple weak labelers.

(c) **Designing task-specific model:** The student models should be task-specific and designed in a way that generalize the automatically generated training set, in order to outperform the weak labeler.

(d) **Weak supervision training:** The last step involves training of the designed machine learning model using the generated weak supervision data. The optimization objective should be carefully designed to be robust to label noise in addition to avoid overfitting.

7.2 Key Findings and Results

The proposed weak supervision solution is a general solution we applied to a number of core information retrieval tasks, including representation learning, ranking, performance prediction, and transfer learning from recommendation data. In this section, we review our key findings.

Learning distributed representations for information retrieval We proposed relevance-based word embedding models for learning IR-specific distributed representation for terms and queries. The model has shown superior performance compared to state-of-the-art bag-of-words embedding methods, such as word2vec [107] and GloVe [123], in a number of IR tasks, including query expansion and query classification. In more detail, we obtained 5-10% improvements in our query expansion experiments using the AP, Robust, and GOV2 collections, and 6-7% improvements in our query classification experiments using the KDD Cup 2005 dataset [93]. Although the focus of this dissertation is on unsupervised training, the effectiveness of the proposed solution trained on large-scale click data has been recently investigated by Zhang et al. [193].

Neural ranking models We proposed a neural ranking model for ad-hoc document retrieval that learns sparse representations for queries and documents for inverted indexing. Our model can retrieve documents from a large collection, instead of re-ranking a small set of documents retrieved by a first stage retrieval model. We additionally provide theoretical guidelines for designing and optimizing weakly supervised neural ranking models. In more detail, we proved that symmetric ranking loss functions are more effective for weakly supervised ranking models. Experiments on Robust and ClueWeb collections showed that our proposed neural ranking model significantly outperforms state-of-the-art retrieval models, such as RM3 [87], SDM [106], and FNRM [42]. We obtained 18-42% improvements compared to a well-tuned query likelihood model [126].

Neural query performance prediction We designed a neural network architecture for post-retrieval query performance prediction. We trained our model using multiple weak supervision signals, resulted in state-of-the-art query performance prediction over a number of TREC collections, including AP, Robust, GOV2, and ClueWeb.

Joint modeling of search and recommendation We designed a joint model for optimizing search engines and recommender systems at the same time. Our model takes advantage of weakly supervised relevance-based word embedding and user-item interaction data (i.e., recommendation data) for learning a retrieval model. Our experiments on an Amazon product dataset and a movie dataset show that transferring knowledge from recommendation data for developing retrieval models gives us 29-95% relative improvement.

7.3 Future Work

We believe that neural information retrieval models and weak supervision training are important advances in information retrieval research. Although, in this dissertation, we tackled many critical issues in these areas, several challenges still remain. In the following, we briefly describe some of these challenges:

Sample selection for weak supervision We provided several successful implementation of weak supervision training in information retrieval applications. The presented solutions require large-scale training data, thus they take a significant amount of time for training. An open research direction is to either select or weight training instances that are useful for training to reduce the training time. Dehghani et al. [40, 41] presented supervised solutions for weighting training instances, however, addressing this issue in an unsupervised setting is still relatively unstudied. For instance, using query performance prediction techniques for weighting training instances in learning to rank models could be a possible solution.

Weak supervision as an easy-to-use framework Learning an accurate and robust weakly supervised model is tightly coupled with the accuracy of the weak labeler(s). Both label generation and model training components require task-specific machine learning expertise. Future work can focus on developing *general* solutions for weak supervision training. Motivated by the proposed joint search and recommendation framework, this framework is expected to contain various transfer learning techniques for transferring knowledge from training data in another domain or application to the target weakly supervised task. This framework will make weak supervision easily accessible to a broader audience.

Efficient and effective neural ranking Although the proposed standalone neural ranking model (SNRM) introduces a solution to retrieve documents instead of just re-ranking a small set of documents, there still remains a number of open research questions. For example, the learned sparse representations are in a continuous space. However, the inverted index compression techniques widely used in search engines rely on the discrete representation of text. Learning discrete sparse representation for queries and documents would be a desired future direction. In addition, the model does not use contextual embedding representations, which has recently shown promising results in a number of NLP and IR tasks [43, 117, 119, 125]. Extending the proposed framework to contextual embedding representation is also an interesting research direction that can potentially lead to significant improvements.

BIBLIOGRAPHY

[1] M. Abadi, A. Agarwal, P. Barham, E. Brevdo, Z. Chen, C. Citro, G. S. Corrado, A. Davis, J. Dean, M. Devin, S. Ghemawat, I. Goodfellow, A. Harp, G. Irving, M. Isard, Y. Jia, R. Jozefowicz, L. Kaiser, M. Kudlur, J. Levenberg, D. Mané, R. Monga, S. Moore, D. Murray, C. Olah, M. Schuster, J. Shlens, B. Steiner, I. Sutskever, K. Talwar, P. Tucker, V. Vanhoucke, V. Vasudevan, F. Viégas, O. Vinyals, P. Warden, M. Wattenberg, M. Wicke, Y. Yu, and X. Zheng. TensorFlow: Large-scale machine learning on heterogeneous systems, 2015. Software available from tensorflow.org.

[2] N. Abdul-jaleel, J. Allan, W. B. Croft, F. Diaz, L. Larkey, X. Li, D. Metzler, M. D. Smucker, T. Strohman, H. Turtle, and C. Wade. UMass at TREC 2004: Novelty and HARD. In *Proceedings of the 2004 Text Retrieval Conference*, TREC '04, Gaithersburg, Maryland, USA, 2004.

[3] Q. Ai, H. Zamani, S. Harding, S. Naseri, J. Allan, and W. B. Croft. UMass at TREC 2017 Common Core Track. In *Proceedings of the 2017 Text Retrieval Conference*, TREC '17, Gaithersburg, Maryland, USA, 2017.

[4] Q. Ai, Y. Zhang, K. Bi, X. Chen, and W. B. Croft. Learning a hierarchical embedding model for personalized product search. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 645–654, Shinjuku, Tokyo, Japan, 2017. ACM.

[5] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. In situ and context-aware target apps selection for unified mobile search. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 1383–1392, Torino, Italy, 2018. ACM.

[6] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. Target apps selection: Towards a unified search framework for mobile devices. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 215–224, Ann Arbor, MI, USA, 2018. ACM.

[7] M. Aliannejadi, H. Zamani, F. Crestani, and W. B. Croft. Asking clarifying questions in open-domain information-seeking conversations. In *Proceedings of the 42Nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 475–484, Paris, France, 2019. ACM.

[8] M. ALMasri, C. Berrut, and J.-P. Chevallet. A comparison of deep learning based query expansion with pseudo-relevance feedback and mutual information. In *Proceedings of the 38th European Conference on IR Research*, ECIR '16, pages 709–715, Padua, Italy, 2016. Springer International Publishing.

[9] G. Amati and C. J. Van Rijsbergen. Probabilistic models of information retrieval based on measuring the divergence from randomness. *ACM Trans. Inf. Syst.*, 20(4):357–389, Oct. 2002.

[10] M. Ariannezhad, A. Montazeralghaem, H. Zamani, and A. Shakery. Iterative estimation of document relevance score for pseudo-relevance feedback. In *Proceedings of the 39th European Conference on IR Research*, ECIR '17, pages 676–683, Aberdeen, UK, 2017. Springer International Publishing.

[11] N. Asadi, D. Metzler, T. Elsayed, and J. Lin. Pseudo test collections for learning web search ranking functions. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1073–1082, Beijing, China, 2011. ACM.

[12] J. A. Aslam and V. Pavlu. Query hardness estimation using jensen-shannon divergence among multiple scoring functions. In *Proceedings of the 29th European Conference on IR Research*, ECIR '07, pages 198–209, Rome, Italy, 2007. Springer-Verlag.

[13] R. Attar and A. S. Fraenkel. Local feedback in full-text retrieval systems. *J. ACM*, 24(3):397–417, July 1977.

[14] L. Azzopardi, M. de Rijke, and K. Balog. Building simulated queries for known-item topics: An analysis using six european languages. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 455–462, Amsterdam, The Netherlands, 2007. ACM.

[15] N. J. Belkin and W. B. Croft. Information filtering and information retrieval: Two sides of the same coin? *Commun. ACM*, 35(12):29–38, Dec. 1992.

[16] A. Borisov, I. Markov, M. de Rijke, and P. Serdyukov. A neural click model for web search. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 531–541, Montreal, Quebec, Canada, 2016. International World Wide Web Conferences Steering Committee.

[17] A. Broder. A taxonomy of web search. *SIGIR Forum*, 36(2):3–10, Sept. 2002.

[18] C. Burges, T. Shaked, E. Renshaw, A. Lazier, M. Deeds, N. Hamilton, and G. Hullender. Learning to rank using gradient descent. In *Proceedings of the 22nd International Conference on Machine Learning*, ICML '05, pages 89–96, Bonn, Germany, 2005. ACM.

[19] C. J. Burges. From ranknet to lambdarank to lambdamart: An overview. Technical report, Microsoft, June 2010.

[20] R. Burke. Hybrid recommender systems: Survey and experiments. *User Modeling and User-Adapted Interaction*, 12(4):331–370, Nov. 2002.

[21] J. G. C. de Souza, H. Zamani, M. Negri, M. Turchi, and F. Daniele. Multitask learning for adaptive quality estimation of automatically transcribed utterances. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL '15, pages 714–724, Denver, Colorado, May–June 2015. Association for Computational Linguistics.

[22] D. Carmel and E. Yom-Tov. *Estimating the Query Difficulty for Information Retrieval.* Morgan and Claypool Publishers, 1st edition, 2010.

[23] S. Chaidaroon, T. Ebesu, and Y. Fang. Deep semantic text hashing with weak supervision. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 1109–1112, Ann Arbor, MI, USA, 2018. ACM.

[24] C. L. A. Clarke, N. Craswell, and E. M. Voorhees. Overview of the TREC 2012 web track. In *Proceedings of the 2012 Text Retrieval Conference*, TREC '12, Gaithersburg, Maryland, USA, 2012.

[25] S. Clinchant and F. Perronnin. Aggregating continuous word embeddings for information retrieval. In *Proceedings of the Workshop on Continuous Vector Space Models and their Compositionality*, pages 100–109, Sofia, Bulgaria, 2013. Association for Computational Linguistics.

[26] D. Cohen and W. B. Croft. End to end long short term memory networks for non-factoid question answering. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 143–146, Newark, Delaware, USA, 2016. ACM.

[27] D. Cohen, J. Foley, H. Zamani, J. Allan, and W. B. Croft. Universal approximation functions for fast learning to rank: Replacing expensive regression forests with simple feed-forward networks. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 1017–1020, Ann Arbor, MI, USA, 2018. ACM.

[28] K. Collins-Thompson. Reducing the risk of query expansion via robust constrained optimization. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 837–846, Hong Kong, China, 2009. ACM.

[29] G. V. Cormack, M. D. Smucker, and C. L. Clarke. Efficient and effective spam filtering and re-ranking for large web datasets. *Inf. Retr.*, 14(5):441–465, Oct. 2011.

[30] B. Croft, D. Metzler, and T. Strohman. *Search Engines: Information Retrieval in Practice*. Addison-Wesley Publishing Company, 1st edition, 2009.

[31] W. B. Croft and D. J. Harper. Document retrieval systems. chapter Using Probabilistic Models of Document Retrieval Without Relevance Information, pages 161–171. Taylor Graham Publishing, London, UK, UK, 1988.

[32] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Predicting query performance. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 299–306, Tampere, Finland, 2002. ACM.

[33] S. Cronen-Townsend, Y. Zhou, and W. B. Croft. Precision prediction based on ranked list coherence. *Inf. Retr.*, 9(6):723–755, Dec. 2006.

[34] J. S. Culpepper, C. L. A. Clarke, and J. Lin. Dynamic cutoff prediction in multi-stage retrieval systems. In *Proceedings of the 21st Australasian Document Computing Symposium*, ADCS '16, pages 17–24, Caulfield, VIC, Australia, 2016. ACM.

[35] R. Cummins, J. Jose, and C. O'Riordan. Improved query performance prediction using standard deviation. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 1089–1090, Beijing, China, 2011. ACM.

[36] Z. Dai, C. Xiong, J. Callan, and Z. Liu. Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 126–134, Marina Del Rey, CA, USA, 2018. ACM.

[37] M. A. Davenport, M. F. Duarte, Y. C. Eldar, and G. Kutyniok. Introduction to compressed sensing. In *Compressed Sensing: Theory and Applications*, pages 1–64. Cambridge University Press, 2012.

[38] J. Dean and S. Ghemawat. Mapreduce: Simplified data processing on large clusters. *Commun. ACM*, 51(1):107–113, 2008.

[39] S. Deerwester, S. T. Dumais, G. W. Furnas, T. K. Landauer, and R. Harshman. Indexing by latent semantic analysis. *Journal of the American Society for Information Science*, 41(6):391–407, 1990.

[40] M. Dehghani, A. Mehrjou, S. Gouws, J. Kamps, and B. Schölkopf. Fidelity-weighted learning. In *Proceedings of the 6th International Conference on Learning Representations*, ICLR '18, 2018.

[41] M. Dehghani, A. Severyn, S. Rothe, and J. Kamps. Learning to learn from weak supervision by full supervision. In *Proceedings of the NeurIPS 2017 workshop on Meta-Learning*, MetaLearn '17, Long Beach, California, USA, 2017.

[42] M. Dehghani, H. Zamani, A. Severyn, J. Kamps, and W. B. Croft. Neural ranking models with weak supervision. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 65–74, Shinjuku, Tokyo, Japan, 2017. ACM.

[43] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL '19, pages 4171–4186, Minneapolis, Minnesota, 2019. Association for Computational Linguistics.

[44] F. Diaz. Performance prediction using spatial autocorrelation. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 583–590, Amsterdam, The Netherlands, 2007. ACM.

[45] F. Diaz, B. Mitra, and N. Craswell. Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics*, ACL '16, pages 367–377, Berlin, Germany, 2016. Association for Computational Linguistics.

[46] X. Dong, L. Yu, Z. Wu, Y. Sun, L. Yuan, and F. Zhang. A hybrid collaborative filtering model with deep structure for recommender systems. In *Proceedings of the 2017 AAAI Conference on Artificial Intelligence*, AAAI '17, pages 1309–1315, San Francisco, California, USA, 2017. AAAI Press.

[47] D. L. Donoho and B. F. Logan. Signal recovery and the large sieve. *SIAM J. Appl. Math.*, 52(2):577–591, 1992.

[48] T. Ebesu, B. Shen, and Y. Fang. Collaborative memory network for recommendation systems. In *The 41st International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '18, pages 515–524, Ann Arbor, MI, USA, 2018. ACM.

[49] A. M. Elkahky, Y. Song, and X. He. A multi-view deep learning approach for cross domain user modeling in recommendation systems. In *Proceedings of the 24th International Conference on World Wide Web*, WWW '15, pages 278–288, Florence, Italy, 2015. International World Wide Web Conferences Steering Committee.

[50] R. A. Fisher. The use of multiple measurements in taxonomic problems. *Annals of Eugenics*, 7(2):179–188, 1936.

[51] D. Ganguly, D. Roy, M. Mitra, and G. J. Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 795–798, Santiago, Chile, 2015. ACM.

[52] A. Ghosh, N. Manwani, and P. Sastry. Making risk minimization tolerant to label noise. *Neurocomput.*, 160(C):93–107, July 2015.

[53] I. Goodfellow, Y. Bengio, and A. Courville. *Deep Learning*. The MIT Press, 2016.

[54] I. J. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, and Y. Bengio. Generative adversarial nets. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NeurIPS '14, pages 2672–2680, Montreal, Canada, 2014. MIT Press.

[55] J. Guo, Y. Fan, Q. Ai, and W. B. Croft. A deep relevance matching model for ad-hoc retrieval. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 55–64, Indianapolis, Indiana, USA, 2016. ACM.

[56] J. Guo, Y. Fan, L. Pang, L. Yang, Q. Ai, H. Zamani, C. Wu, W. B. Croft, and X. Cheng. A deep look into neural ranking models for information retrieval. *Information Processing & Management*, 2019.

[57] Y. Guo, Z. Cheng, L. Nie, Y. Wang, J. Ma, and M. Kankanhalli. Attentive long short-term preference modeling for personalized product search. *ACM Trans. Inf. Syst.*, 37(2):19:1–19:27, Jan. 2019.

[58] M. U. Gutmann and A. Hyvärinen. Noise-contrastive estimation of unnormalized statistical models, with applications to natural image statistics. *J. Mach. Learn. Res.*, 13(1):307–361, Feb. 2012.

[59] D. Haddad and J. Ghosh. Learning more from less: Towards strengthening weak supervision for ad-hoc retrieval. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 857–860, Paris, France, 2019. ACM.

[60] M. Hagen, D. Wägner, and B. Stein. A corpus of realistic known-item topics with associated web pages in the clueweb09. In *Proceedings of the 37th European Conference on IR Research*, ECIR '15, pages 513–525, Vienna, Austria, 2015. Springer International Publishing.

[61] F. M. Harper and J. A. Konstan. The movielens datasets: History and context. *ACM Trans. Interact. Intell. Syst.*, 5(4):19:1–19:19, Dec. 2015.

[62] H. Hashemi, M. Aliannejadi, H. Zamani, and W. B. Croft. ANTIQUE: A nonfactoid question answering benchmark. *CoRR*, abs/1905.08957, 2019.

[63] H. Hashemi, H. Zamani, and W. B. Croft. Performance prediction for nonfactoid question answering. In *Proceedings of the 2019 ACM International Conference on the Theory of Information Retrieval*, ICTIR '19, Santa Clara, California, USA, 2019. ACM.

[64] C. Hauff, D. Hiemstra, and F. de Jong. A survey of pre-retrieval query performance predictors. In *Proceedings of the 17th ACM Conference on Information and Knowledge Management*, CIKM '08, pages 1419–1420, Napa Valley, California, USA, 2008. ACM.

[65] J. He, M. Larson, and M. de Rijke. Using coherence-based measures to predict query difficulty. In *Proceedings of the 30th European Conference on IR Research*, ECIR '08, pages 689–694, Glasgow, UK, 2008. Springer Berlin Heidelberg.

[66] R. He and J. McAuley. Ups and downs: Modeling the visual evolution of fashion trends with one-class collaborative filtering. In *Proceedings of the 25th International Conference on World Wide Web*, WWW '16, pages 507–517, Montreal, Quebec, Canada, 2016. International World Wide Web Conferences Steering Committee.

[67] X. He, T. Chen, M.-Y. Kan, and X. Chen. Trirank: Review-aware explainable recommendation by modeling aspects. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 1661–1670, Melbourne, Australia, 2015. ACM.

[68] X. He, L. Liao, H. Zhang, L. Nie, X. Hu, and T.-S. Chua. Neural collaborative filtering. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 173–182, Perth, Australia, 2017. International World Wide Web Conferences Steering Committee.

[69] W. Hill, L. Stead, M. Rosenstein, and G. Furnas. Recommending and evaluating choices in a virtual community of use. In *Proceedings of the SIGCHI Conference on Human Factors in Computing Systems*, CHI '95, pages 194–201, Denver, Colorado, USA, 1995. ACM Press/Addison-Wesley Publishing Co.

[70] G. E. Hinton, J. L. McClelland, and D. E. Rumelhart. Parallel distributed processing: Explorations in the microstructure of cognition, vol. 1. chapter Distributed Representations, pages 77–109. MIT Press, Cambridge, MA, USA, 1986.

[71] G. E. Hinton and R. R. Salakhutdinov. Reducing the dimensionality of data with neural networks. *Science*, 313(5786):504–507, 2006.

[72] H. Hotelling. Analysis of a complex of statistical variables with principal components. *Journal of Educational Psychology*, 24:417–441, 1933.

[73] Y. Hu, Y. Koren, and C. Volinsky. Collaborative filtering for implicit feedback datasets. In *Proceedings of the 2008 Eighth IEEE International Conference on Data Mining*, ICDM '08, pages 263–272, Pisa, Italy, 2008. IEEE Computer Society.

[74] P.-S. Huang, X. He, J. Gao, L. Deng, A. Acero, and L. Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 2333–2338, San Francisco, California, USA, 2013. ACM.

[75] K. Järvelin and J. Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Trans. Inf. Syst.*, 20(4):422–446, Oct. 2002.

[76] Y. Jing and W. B. Croft. An association thesaurus for information retrieval. In *Intelligent Multimedia Information Retrieval Systems and Management*, RIAO '94, pages 146–160, New York, NY, USA, 1994.

[77] T. Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the Eighth ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '02, pages 133–142, Edmonton, Alberta, Canada, 2002. ACM.

[78] M. Karimzadehgan and C. Zhai. Estimation of statistical translation models based on mutual information for ad hoc information retrieval. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 323–330, Geneva, Switzerland, 2010. ACM.

[79] Y. Kim and J. Allan. Unsupervised explainable controversy detection from online news. In *Proceedings of the 41st European Conference on IR Research*, ECIR '19, pages 836–843. Springer International Publishing, 2019.

[80] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. In *Proceedings of the 3rd International Conference on Learning Representations*, ICLR '15, San Diego, CA, USA, 2015.

[81] Y. Koren. Factorization meets the neighborhood: A multifaceted collaborative filtering model. In *Proceedings of the 14th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '08, pages 426–434, Las Vegas, Nevada, USA, 2008. ACM.

[82] O. Kurland, A. Shtok, D. Carmel, and S. Hummel. A unified framework for post-retrieval query-performance prediction. In *Proceedings of the Third International Conference on Advances in Information Retrieval Theory*, ICTIR '11, pages 15–26, Bertinoro, Italy, 2011. Springer-Verlag.

[83] M. J. Kusner, Y. Sun, N. I. Kolkin, and K. Q. Weinberger. From word embeddings to document distances. In *Proceedings of the 32nd International Conference on International Conference on Machine Learning - Volume 37*, ICML'15, pages 957–966, Lille, France, 2015.

[84] S. Kuzi, A. Shtok, and O. Kurland. Query expansion using word embeddings. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1929–1932, Indianapolis, Indiana, USA, 2016. ACM.

[85] J. Lafferty and C. Zhai. Document language models, query models, and risk minimization for information retrieval. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 111–119, New Orleans, Louisiana, USA, 2001. ACM.

[86] V. Lavrenko, M. Choquette, and W. B. Croft. Cross-lingual relevance models. In *Proceedings of the 25th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '02, pages 175–182, Tampere, Finland, 2002. ACM.

[87] V. Lavrenko and W. B. Croft. Relevance based language models. In *Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '01, pages 120–127, New Orleans, Louisiana, USA, 2001. ACM.

[88] U. Lee, Z. Liu, and J. Cho. Automatic identification of user goals in web search. In *Proceedings of the 14th International Conference on World Wide Web*, WWW '05, pages 391–400, Chiba, Japan, 2005. ACM.

[89] O. Levy and Y. Goldberg. Neural word embedding as implicit matrix factorization. In *Proceedings of the 27th International Conference on Neural Information Processing Systems*, NeurIPS '14, pages 2177–2185, Montreal, Canada, 2014. MIT Press.

[90] C. Li, M. Zhang, M. Bendersky, H. Deng, D. Metzler, and M. Najork. Multiview embedding-based synonyms for email search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 575–584, Paris, France, 2019. ACM.

[91] H. Li. *Learning to Rank for Information Retrieval and Natural Language Processing.* Morgan & Claypool Publishers, 2011.

[92] S. Li, J. Kawale, and Y. Fu. Deep collaborative filtering via marginalized denoising auto-encoder. In *Proceedings of the 24th ACM International on Conference on Information and Knowledge Management*, CIKM '15, pages 811–820, Melbourne, Australia, 2015. ACM.

[93] Y. Li, Z. Zheng, and H. K. Dai. Kdd cup-2005 report: Facing a great challenge. *SIGKDD Explor. Newsl.*, 7(2):91–99, Dec. 2005.

[94] D. Liang, J. Altosaar, L. Charlin, and D. M. Blei. Factorization meets the item embedding: Regularizing matrix factorization with item co-occurrence. In *Proceedings of the 10th ACM Conference on Recommender Systems*, RecSys '16, pages 59–66, Boston, Massachusetts, USA, 2016. ACM.

[95] Z. Liu, C. Xiong, M. Sun, and Z. Liu. Entity-duet neural ranking: Understanding the role of knowledge graph semantics in neural information retrieval. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 2395–2405, Melbourne, Australia, 2018. Association for Computational Linguistics.

[96] X. Lu, A. Moffat, and J. S. Culpepper. The effect of pooling and evaluation depth on ir metrics. *Inf. Retr.*, 19(4):416–445, Aug. 2016.

[97] Z. Lu and H. Li. A deep architecture for matching short texts. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NeurIPS '13, pages 1367–1375, Lake Tahoe, Nevada, 2013. Curran Associates Inc.

[98] C. Luo, Y. Zheng, J. Mao, Y. Liu, M. Zhang, and S. Ma. Training deep ranking model with weak relevance labels. In *Databases Theory and Applications*, pages 205–216. Springer International Publishing, 2017.

[99] Y. Lv and C. Zhai. A comparative study of methods for estimating query language models with pseudo feedback. In *Proceedings of the 18th ACM Conference on Information and Knowledge Management*, CIKM '09, pages 1895–1898, Hong Kong, China, 2009. ACM.

[100] S. MacAvaney, A. Yates, K. Hui, and O. Frieder. Content-based weak supervision for ad-hoc re-ranking. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 993–996, Paris, France, 2019. ACM.

[101] J. Mackenzie, J. S. Culpepper, R. Blanco, M. Crane, C. L. A. Clarke, and J. Lin. Query driven algorithm selection in early stage retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 396–404, Marina Del Rey, CA, USA, 2018. ACM.

[102] C. D. Manning, P. Raghavan, and H. Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008.

[103] N. Matthijs and F. Radlinski. Personalizing web search using long term browsing history. In *Proceedings of the Fourth ACM International Conference on Web Search and Data Mining*, WSDM '11, pages 25–34, Hong Kong, China, 2011. ACM.

[104] J. McAuley, C. Targett, Q. Shi, and A. van den Hengel. Image-based recommendations on styles and substitutes. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 43–52, Santiago, Chile, 2015. ACM.

[105] D. Metzler and W. Bruce Croft. Linear feature-based models for information retrieval. *Information Retrieval*, 10(3):257–274, Jun 2007.

[106] D. Metzler and W. B. Croft. 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*, SIGIR '05, pages 472–479, Salvador, Brazil, 2005.

[107] T. Mikolov, I. Sutskever, K. Chen, G. Corrado, and J. Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems*, NeurIPS '13, pages 3111–3119, Lake Tahoe, Nevada, 2013. Curran Associates Inc.

[108] B. Mitra and N. Craswell. An introduction to neural information retrieval. *Foundations and Trends in Information Retrieval*, pages 1–117, April 2018.

[109] B. Mitra, F. Diaz, and N. Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1291–1299, Perth, Australia, 2017. International World Wide Web Conferences Steering Committee.

[110] A. Mnih and G. E. Hinton. A scalable hierarchical distributed language model. In *Proceedings of the 22nd International Conference on Neural Information Processing Systems*, NeurIPS '09, pages 1081–1088. Curran Associates, Inc., 2009.

[111] A. Montazeralghaem, H. Zamani, and A. Shakery. Axiomatic analysis for improving the log-logistic feedback model. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 765–768, Pisa, Italy, 2016. ACM.

[112] F. Morin and Y. Bengio. Hierarchical probabilistic neural network language model. In *Proceedings of the Tenth International Workshop on Artificial Intelligence and Statistics*, pages 246–252. Society for Artificial Intelligence and Statistics, 2005.

[113] J. Mothe and L. Tanguy. Linguistic features to predict query difficulty. In *ACM SIGIR Workshop on Predicting Query Difficulty - Methods and Applications*, pages 7–10, Salvador de Bahia, Brazil, 2005.

[114] S. Muthukrishnan. Data streams: Algorithms and applications. *Foundations and Trends in Theoretical Computer Science*, 1(2):117–236, 2005.

[115] M. Negri, M. Turchi, J. G. C. de Souza, and F. Daniele. Quality estimation for automatic speech recognition. In *Proceedings of the 25th International Conference on Computational Linguistics*, COLING '14, pages 1813–1823, Dublin, Ireland, 2014. Dublin City University and Association for Computational Linguistics.

[116] Y. Nie, A. Sordoni, and J.-Y. Nie. Multi-level abstraction convolutional model with weak supervision for information retrieval. In *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '18, pages 985–988, Ann Arbor, MI, USA, 2018. ACM.

[117] R. Nogueira and K. Cho. Passage re-ranking with BERT. *CoRR*, abs/1901.04085, 2019.

[118] M. G. Noll and C. Meinel. Web search personalization via social bookmarking and tagging. In *Proceedings of the 6th International Semantic Web Conference*, ISWC '07, pages 367–380, Busan, Korea, 2007. Springer Berlin Heidelberg.

[119] H. Padigela, H. Zamani, and W. B. Croft. Investigating the successes and failures of BERT for passage re-ranking. *CoRR*, abs/1905.01758, 2019.

[120] J. Parapar, A. Bellogín, P. Castells, and A. Barreiro. Relevance-based language modelling for recommender systems. *Inf. Process. Manage.*, 49(4):966–980, July 2013.

[121] G. Pass, A. Chowdhury, and C. Torgeson. A picture of search. In *Proceedings of the 1st International Conference on Scalable Information Systems*, InfoScale '06, Hong Kong, China, 2006. ACM.

[122] K. Pearson. On lines and planes of closest fit to systems of points in space. *The London, Edinburgh, and Dublin Philosophical Magazine and Journal of Science*, 2(11):559–572, 1901.

[123] J. Pennington, R. Socher, and C. Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543, Doha, Qatar, 2014. Association for Computational Linguistics.

[124] J. Pérez-Iglesias and L. Araujo. Standard deviation as a query hardness estimator. In *String Processing and Information Retrieval*, SPIRE '10, pages 207–212, Los Cabos, Mexico, 2010. Springer Berlin Heidelberg.

[125] M. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, NAACL '18, pages 2227–2237, New Orleans, Louisiana, 2018. Association for Computational Linguistics.

[126] J. M. Ponte and W. B. Croft. A language modeling approach to information retrieval. In *Proceedings of the 21st Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '98, pages 275–281, Melbourne, Australia, 1998. ACM.

[127] L. Y. Pratt. Discriminability-based transfer between neural networks. In *Proceedings of the 6th International Conference on Neural Information Processing Systems*, NeurIPS '93, pages 204–211. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 1993.

[128] T. Qin and T. Liu. Introducing LETOR 4.0 datasets. *CoRR*, abs/1306.2597, 2013.

[129] F. Raiber and O. Kurland. Query-performance prediction: Setting the expectations straight. In *Proceedings of the 37th International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '14, pages 13–22, Gold Coast, Queensland, Australia, 2014. ACM.

[130] A. Ratner, S. H. Bach, H. Ehrenberg, J. Fries, S. Wu, and C. Ré. Snorkel: Rapid training data creation with weak supervision. *Proc. VLDB Endow.*, 11(3):269–282, Nov. 2017.

[131] N. Rekabsaz, M. Lupu, A. Hanbury, and H. Zamani. Word embedding causes topic shifting; exploit global context! In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 1105–1108, Shinjuku, Tokyo, Japan, 2017. ACM.

[132] S. Rendle, C. Freudenthaler, Z. Gantner, and L. Schmidt-Thieme. Bpr: Bayesian personalized ranking from implicit feedback. In *Proceedings of the Twenty-Fifth Conference on Uncertainty in Artificial Intelligence*, UAI '09, pages 452–461, Montreal, Quebec, Canada, 2009. AUAI Press.

[133] P. Resnick, N. Iacovou, M. Suchak, P. Bergstrom, and J. Riedl. Grouplens: An open architecture for collaborative filtering of netnews. In *Proceedings of the 1994 ACM Conference on Computer Supported Cooperative Work*, CSCW '94, pages 175–186, Chapel Hill, North Carolina, USA, 1994. ACM.

[134] S. Robertson, S. Walker, S. Jones, M. Hancock-Beaulieu, and M. Gatford. Okapi at trec-3. In *TREC '96*, pages 109–126, Gaithersburg, Maryland, USA, 1996.

[135] J. J. Rocchio. Relevance Feedback in Information Retrieval. In *The SMART Retrieval System: Experiments in Automatic Document Processing*, pages 313–323. 1971.

[136] H. Roitman. An enhanced approach to query performance prediction using reference lists. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 869–872, Shinjuku, Tokyo, Japan, 2017. ACM.

[137] H. Roitman, S. Erera, O. Sar-Shalom, and B. Weiner. Enhanced mean retrieval score estimation for query performance prediction. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 35–42, Amsterdam, The Netherlands, 2017. ACM.

[138] H. Roitman, S. Erera, and B. Weiner. Robust standard deviation estimation for query performance prediction. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '17, pages 245–248, Amsterdam, The Netherlands, 2017. ACM.

[139] J. Rowley. Product search in e-shopping: A review and research propositions. *Journal of Consumer Marketing*, 17(1):20–35, 2000.

[140] D. E. Rumelhart, G. E. Hinton, and R. J. Williams. Learning representations by back-propagating errors. *Nature*, 323:533–536, 1986.

[141] R. Salakhutdinov and A. Mnih. Probabilistic matrix factorization. In *Proceedings of the 20th International Conference on Neural Information Processing Systems*, NeurIPS '07, pages 1257–1264, Vancouver, British Columbia, Canada, 2007. Curran Associates Inc.

[142] G. Salton. *Automatic Information Organization and Retrieval.* McGraw Hill Text, 1968.

[143] G. Salton, A. Wong, and C. S. Yang. A vector space model for automatic indexing. *Commun. ACM*, 18(11):613–620, Nov. 1975.

[144] B. Sarwar, G. Karypis, J. Konstan, and J. Riedl. Item-based collaborative filtering recommendation algorithms. In *Proceedings of the 10th International Conference on World Wide Web*, WWW '01, pages 285–295, Hong Kong, Hong Kong, 2001. ACM.

[145] M. Schedl, H. Zamani, C.-W. Chen, Y. Deldjoo, and M. Elahi. Current challenges and visions in music recommender systems research. *International Journal of Multimedia Information Retrieval*, 7(2):95–116, Jun 2018.

[146] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 373–382, Santiago, Chile, 2015. ACM.

[147] A. Shakery and C. Zhai. A probabilistic relevance propagation model for hypertext retrieval. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 550–558, Arlington, Virginia, USA, 2006. ACM.

[148] A. Shakery and C. Zhai. Smoothing document language models with probabilistic term count propagation. *Inf. Retr.*, 11(2):139–164, Apr. 2008.

[149] J. Shen, M. Karimzadehgan, M. Bendersky, Z. Qin, and D. Metzler. Multi-task learning for email search ranking with auxiliary query clustering. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 2127–2135, Torino, Italy, 2018. ACM.

[150] Y. Shen, X. He, J. Gao, L. Deng, and G. Mesnil. Learning semantic representations using convolutional neural networks for web search. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14 Companion, pages 373–374, Seoul, Korea, 2014. ACM.

[151] E. Shnarch, C. Alzate, L. Dankin, M. Gleize, Y. Hou, L. Choshen, R. Aharonov, and N. Slonim. Will it blend? blending weak and strong labeled data in a neural network for argumentation mining. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics*, ACL '18, pages 599–605, Melbourne, Australia, July 2018. Association for Computational Linguistics.

[152] A. Shtok, O. Kurland, and D. Carmel. Using statistical decision theory and relevance models for query-performance prediction. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '10, pages 259–266, Geneva, Switzerland, 2010. ACM.

[153] A. Shtok, O. Kurland, and D. Carmel. Query performance prediction using reference lists. *ACM Trans. Inf. Syst.*, 34(4):19:1–19:34, June 2016.

[154] A. Shtok, O. Kurland, D. Carmel, F. Raiber, and G. Markovits. Predicting query performance by query-drift estimation. *ACM Trans. Inf. Syst.*, 30(2):11:1–11:35, May 2012.

[155] A. Sordoni, Y. Bengio, and J.-Y. Nie. Learning concept embeddings for query expansion by quantum entropy minimization. In *Proceedings of the Twenty-Eighth AAAI Conference on Artificial Intelligence*, AAAI'14, pages 1586–1592, Quebec City, Quebec, Canada, 2014. AAAI Press.

[156] L. Specia, M. Turchi, N. Cancedda, M. Dymetman, and N. Cristianini. Estimating the sentence-level quality of machine translation systems. In *Proceedings of the Annual Conference of European Association for Machine Translation*, EAMT '09, pages 28–37, 2009.

[157] T. Tao and C. Zhai. Regularized estimation of mixture models for robust pseudo-relevance feedback. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 162–169, Seattle, Washington, USA, 2006. ACM.

[158] Y. Tao and S. Wu. Query performance prediction by considering score magnitude and variance together. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 1891–1894, Shanghai, China, 2014. ACM.

[159] P. Thomas, F. Scholer, P. Bailey, and A. Moffat. Tasks, queries, and rankers in pre-retrieval performance prediction. In *Proceedings of the 22nd Australasian Document Computing Symposium*, ADCS '17, pages 11:1–11:4, Brisbane, QLD, Australia, 2017. ACM.

[160] R. Tibshirani. Regression shrinkage and selection via the lasso. *J. R. Stat. Soc. Series B*, 58:267–288, 1994.

[161] T. Tran, K. Lee, Y. Liao, and D. Lee. Regularizing matrix factorization with user and item embeddings for recommendation. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 687–696, Torino, Italy, 2018. ACM.

[162] Y. Ustinovskiy and P. Serdyukov. Personalization of web-search using short-term browsing context. In *Proceedings of the 22nd ACM International Conference on Information & Knowledge Management*, CIKM '13, pages 1979–1988, San Francisco, California, USA, 2013. ACM.

[163] C. Van Gysel, M. de Rijke, and E. Kanoulas. Learning latent vector spaces for product search. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 165–174, Indianapolis, Indiana, USA, 2016. ACM.

[164] C. J. van Rijsbergen. *Information Retrieval*. Butterworth-Heinemann, Newton, MA, USA, 2nd edition, 1979.

[165] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. u. Kaiser, and I. Polosukhin. Attention is all you need. In *Proceedings of the 30th International Conference on Neural Information Processing Systems*, NeurIPS '17, pages 5998–6008. Curran Associates, Inc., 2017.

[166] V. Vinay, I. J. Cox, N. Milic-Frayling, and K. Wood. On ranking the effectiveness of searches. In *Proceedings of the 29th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 398–404, Seattle, Washington, USA, 2006. ACM.

[167] N. Voskarides, E. Meij, R. Reinanda, A. Khaitan, M. Osborne, G. Stefanoni, P. Kambadur, and M. de Rijke. Weakly-supervised contextualization of knowledge graph facts. In *The 41st International ACM SIGIR Conference on Research &#38; Development in Information Retrieval*, SIGIR '18, pages 765–774, Ann Arbor, MI, USA, 2018. ACM.

[168] I. Vulić and M.-F. Moens. 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*, SIGIR '15, pages 363–372, Santiago, Chile, 2015. ACM.

[169] H. Wang, N. Wang, and D.-Y. Yeung. Collaborative deep learning for recommender systems. In *Proceedings of the 21th ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '15, pages 1235–1244, Sydney, NSW, Australia, 2015. ACM.

[170] L. Wang, J. Lin, and D. Metzler. A cascade ranking model for efficient ranked retrieval. In *Proceedings of the 34th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '11, pages 105–114, Beijing, China, 2011. ACM.

[171] R. Xiao, J. Ji, B. Cui, H. Tang, W. Ou, Y. Xiao, J. Tan, and X. Ju. Weakly supervised co-training of query rewriting andsemantic matching for e-commerce. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 402–410, Melbourne, VIC, Australia, 2019. ACM.

[172] C. Xiong, Z. Dai, J. Callan, Z. Liu, and R. Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 55–64, Shinjuku, Tokyo, Japan, 2017. ACM.

[173] J. Xu and W. B. Croft. Query expansion using local and global document analysis. In *Proceedings of the 19th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '96, pages 4–11, Zurich, Switzerland, 1996. ACM.

[174] P. Xu, X. Ma, R. Nallapati, and B. Xiang. Passage ranking with weak supervsion. *CoRR*, abs/1905.05910, 2019.

[175] L. Yang, H. Zamani, Y. Zhang, J. Guo, and W. B. Croft. Neural matching models for question retrieval and next question prediction in conversation. *CoRR*, abs/1707.05409, 2017.

[176] D. Yin, Y. Hu, J. Tang, T. Daly, M. Zhou, H. Ouyang, J. Chen, C. Kang, H. Deng, C. Nobata, J.-M. Langlois, and Y. Chang. Ranking relevance in yahoo search. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 323–332, San Francisco, California, USA, 2016. ACM.

[177] H. Zamani, M. Bendersky, X. Wang, and M. Zhang. Situational context for ranking in personal search. In *Proceedings of the 26th International Conference on World Wide Web*, WWW '17, pages 1531–1540, Perth, Australia, 2017. International World Wide Web Conferences Steering Committee.

[178] H. Zamani and W. B. Croft. Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 147–156, Newark, Delaware, USA, 2016. ACM.

[179] H. Zamani and W. B. Croft. Estimating embedding vectors for queries. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 123–132, Newark, Delaware, USA, 2016. ACM.

[180] H. Zamani and W. B. Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '17, pages 505–514, Shinjuku, Tokyo, Japan, 2017. ACM.

[181] H. Zamani and W. B. Croft. Joint modeling and optimization of search and recommendation. In *Proceedings of the First International Conference on Design of Experimental Search and Information Retrieval Systems*, DESIRES '18, pages 36–41, Bertinoro, Italy, 2018. CEUR.

[182] H. Zamani and W. B. Croft. On the theory of weak supervision for information retrieval. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '18, pages 147–154, Tianjin, China, 2018. ACM.

[183] H. Zamani and W. B. Croft. Towards theoretical understanding of weak supervision for information retrieval. *CoRR*, abs/1806.04815, 2018.

[184] H. Zamani, W. B. Croft, and J. S. Culpepper. Neural query performance prediction using weak supervision from multiple signals. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 105–114, Ann Arbor, MI, USA, 2018. ACM.

[185] H. Zamani, J. Dadashkarimi, A. Shakery, and W. B. Croft. Pseudo-relevance feedback based on matrix factorization. In *Proceedings of the 25th ACM International on Conference on Information and Knowledge Management*, CIKM '16, pages 1483–1492, Indianapolis, Indiana, USA, 2016. ACM.

[186] H. Zamani, M. Dehghani, W. B. Croft, E. Learned-Miller, and J. Kamps. 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*, CIKM '18, pages 497–506, Torino, Italy, 2018. ACM.

[187] H. Zamani, M. Dehghani, F. Diaz, H. Li, and N. Craswell. Sigir 2018 workshop on learning from limited or noisy data for information retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 1439–1440, Ann Arbor, MI, USA, 2018. ACM.

[188] H. Zamani, B. Mitra, X. Song, N. Craswell, and S. Tiwary. Neural ranking models with multiple document fields. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 700–708, Marina Del Rey, CA, USA, 2018. ACM.

[189] H. Zamani, M. Schedl, P. Lamere, and C.-W. Chen. An analysis of approaches taken in the acm recsys challenge 2018 for automatic music playlist continuation. *ACM Trans. Intell. Syst. Technol.*, 2019. (to appear).

[190] C. Zhai. *Statistical Language Models for Information Retrieval.* Now Publishers Inc., Hanover, MA, USA, 2008.

[191] C. Zhai and J. Lafferty. Model-based feedback in the language modeling approach to information retrieval. In *Proceedings of the Tenth International Conference on Information and Knowledge Management*, CIKM '01, pages 403–410, Atlanta, Georgia, USA, 2001. ACM.

[192] C. Zhai and J. Lafferty. A study of smoothing methods for language models applied to information retrieval. *ACM Trans. Inf. Syst.*, 22(2):179–214, Apr. 2004.

[193] H. Zhang, X. Song, C. Xiong, C. Rosset, P. N. Bennett, N. Craswell, and S. Tiwary. Generic intent representation in web search. In *Proceedings of the 42nd International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '19, pages 65–74, Paris, France, 2019. ACM.

[194] Y. Zhang and X. Chen. Explainable recommendation: A survey and new perspectives. *CoRR*, abs/1804.11192, 2018.

[195] Y. Zhang, X. Chen, Q. Ai, L. Yang, and W. B. Croft. Towards conversational search and recommendation: System ask, user respond. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, CIKM '18, pages 177–186, Torino, Italy, 2018. ACM.

[196] G. Zheng and J. Callan. Learning to reweight terms with distributed representations. In *Proceedings of the 38th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '15, pages 575–584, Santiago, Chile, 2015. ACM.

[197] G. Zhou, T. He, J. Zhao, and P. Hu. Learning continuous word embedding with metadata for question retrieval in community question answering. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing*, ACL-IJCNLP '15, pages 250–259, Beijing, China, 2015. Association for Computational Linguistics.

[198] Y. Zhou and W. B. Croft. Ranking robustness: A novel framework to predict query performance. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 567–574, Arlington, Virginia, USA, 2006. ACM.

[199] Y. Zhou and W. B. Croft. Query performance prediction in web search environments. In *Proceedings of the 30th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '07, pages 543–550, Amsterdam, The Netherlands, 2007. ACM.

[200] G. Zuccon, B. Koopman, P. Bruza, and L. Azzopardi. Integrating and evaluating neural word embeddings in information retrieval. In *Proceedings of the 20th Australasian Document Computing Symposium*, ADCS '15, pages 12:1–12:8, Parramatta, NSW, Australia, 2015. ACM.