Leibniz Universität Hannover

Masters Thesis

Reproducibility and Interpretability of Neural Ranking Models

Author:
Zeon Trevor Fernando

Supervisors:
Prof. Dr. Avishek Anand
M.Sc. Jaspreet Singh

Examiners:
Prof. Dr. Avishek Anand
Prof. Dr. Wolfgang Nejdl

A thesis submitted in fulfillment of the requirements
for the degree of M.Sc. Internet Techonologies and Information Systems

in the

Institut für Verteilte Systeme - Fachgebiet Wissensbasierte Systeme

March 2019

Declaration of Authorship

I, Zeon Trevor Fernando, declare that this thesis titled, 'Reproducibility and Interpretability of Neural Ranking Models' and the work presented in it is my own. I confirm that this work submitted for assessment is my own and is expressed in my own words. Any uses made within it of the works of other authors in any form (e.g., ideas, equations, figures, text, tables, programs) are properly acknowledged at any point of their use. A list of the references employed is included.

Signed: _____

Date: _____

Abstract

In recent years, many deep neural network models have been proposed for the ad-hoc retrieval task. They have shown to perform better than traditional information retrieval baselines such as BM25, query-likelihood (QL) and some state-of-the-art learning-to-rank approaches that requires a lot of hand-crafted feature engineering. In this thesis, we first implement a select few neural IR approaches (DRMM, MatchPyramid, PACRR-DRMM, NPRF-DRMM) using the standard TREC newswire collection (Robust04) to see how well we can reproduce the results highlighted in the respective papers.

In the second part of the thesis, we focus on the question–why a *document* is relevant to a query using a particular neural model. In the ML community, several approaches for explaining decisions made by deep neural networks have been proposed – including DeepSHAP which modifies the DeepLift algorithm to estimate the relative importance (shapley values) of input features for a given decision by comparing the activations in the network for a given image against the activations caused by a reference input. In image classification, the reference input tends to be a plain black image. While DeepSHAP has been well studied for image classification tasks, we see how we can adapt it to explain the output of Neural Retrieval Models (NRMs). In particular, what is a good "black" image in the context of IR? In this thesis, we explored various reference input document construction techniques. Additionally, we compared the explanations generated by DeepSHAP to LIME (a model agnostic approach) and found that the explanations differ considerably. This study raises concerns regarding the robustness and accuracy of explanations produced for NRMs. With this work we aim to shed light on interesting problems surrounding interpretability in NRMs and highlight areas of future work.

Acknowledgements

First, I would like to thank my supervisor Prof. Dr. Avishek Anand for giving me the opportunity to work on this interesting and challenging topic for my thesis and also for his constant guidance and invaluable ideas. I am also extremely grateful for his understanding and support in helping me change my topic mid-way through my thesis.

I would like to thank Jaspreet Singh for his invaluable suggestions that helped me successfully train most of these neural ranking models and for always being available to talk through various approaches and ideas.

I would also like to thank Prof. Dr. Wolfgang Nejdl for giving me the opportunity to work at L3S Research Center during my masters on a lot of interesting and diverse research problems ranging from Information Retrieval, Machine Learning to User Interface Design.

I would like to thank my friends for their constant support and help during this journey that had many ups and downs.

Finally, I am grateful to my parents for supporting me in the decisions I made. I am deeply thankful for their constant source of encouragement and belief in me and also Cheryl, my wonderful and talented sister who picks my brain from time to time to understand the fundamentals of programming.

Contents

List of Figures

List of Tables

Abbreviations

CBOW	**C**ontinuous **B**ag **O**f **W**ords
CNN	**C**onvolutional **N**eural **N**etwork
DESM	**D**ual **E**mbedding **S**pace **M**odel
DRMM	**D**eep **R**elevance **M**atching **M**odel
DSSM	**D**eep **S**tructured **S**emantic **M**odel
IR	**I**nformation **R**etrieval
LIME	**L**ocal **I**nterpretable **M**odel-agnostic **E**xplanations
LM	**L**anguage **M**odel
LTR	**L**earning-**T**o-**R**ank
MLP	**M**ulti-**L**ayer **P**erceptron
NN	**N**eural **N**etwork
NPRF	**N**euro **P**seudo **R**elevance **F**eedback
NRM	**N**eural **R**etrieval **M**odel
OOV	**O**ut-**O**f-**V**ocabulary
QL	**Q**uery **L**ikelihood
RNN	**R**ecurrent **N**eural **N**etwork

Chapter 1

Introduction

In recent years, deep neural networks have achieved state of the art results in various tasks or domains, such as, computer vision, speech recognition and machine translation tasks [LeCun et al., 2015]. These models have shown to be effective at learning abstract representations from the raw input (large training datasets), and the models have sufficient generalizability to be able to tackle different learning problems. These are desirable properties for the problem of relevance ranking in Information Retrieval (IR) and in recent years there have been a substantial number of works that applied these neural methods in IR displaying advances in state of the art.

Information retrieval is a core component in many real-world applications–Web search, digital libraries, e-commerce platforms and so on. The problem of relevance ranking in IR (*ad-hoc* retrieval) is defined as–given a query and set of candidate documents, a scoring function determines the degree of relevance of the document to the query and finally produces a ranking list by sorting the relevance scores. Many different ranking models have been proposed including vector space models (TF-IDF), probabilistic retrieval models (BM25, QL) and learning to rank (LTR) approaches that applies machine learning to the ranking function achieving great improvements in retrieval effectiveness [Liu et al., 2009]. However, successful LTR models relies heavily on hand-crafted feature engineering that is time-confusing, incomplete and over-specified. Thus, neural IR models that employ the use of shallow or deep neural networks for IR have been proposed as a solution to tackle the feature engineering problem of learning to rank, by using only automatically learned features from raw text input of the query and document. These

neural IR models use vector representations of text, and have a large number of parameters that needs to be tuned with large-scale datasets in training [Mitra et al., 2017]. Therefore, unlike traditional IR models, these models are more "data-hungry" and the performance increases as training data increases.

All existing neural IR can be broadly categorized based on whether they influence the query representation that encodes information need, or the document representation that captures distribution of information contained or estimate the mutual relevance based on patterns of matching between the query and document. They are categorized, as *representation-based* approaches (DSSM [Huang et al., 2013], CDSSM [Shen et al., 2014a,b]), and *interaction-based* approaches (DRMM [Guo et al., 2016], DUET [Mitra et al., 2017], MatchPyramid [Pang et al., 2016], PACRR [Hui et al., 2017, 2018], K-NRM [Xiong et al., 2017]). The representation-based approaches aimed to learn document representations to match with query representations, while interaction-based approaches learn from matching matrices between query and document. Recently, there have been works that focused on architectures that try to model the human relevance judgement process (DeepRank [Pang et al., 2017b]) and on learning relevance signals at different granularities (i.e. passage-level or document-wide) [Fan et al., 2018]. The various neural ranking models learned from generating the features automatically have already outperformed several state-of-the-art LTR models with tens of hand crafted features [Fan et al., 2018, Pang et al., 2017b]. There have been some interesting works whose focus has been on the training strategies of neural ranking models as not everyone has access to commercial large-scale training datasets [Dehghani et al., 2018a, 2017] and learning alternative indexing schemes for neural IR models [Zamani et al., 2018] that moves from the neural re-ranking strategies (*telescoping*) adopted by most previous approaches.

With the spurt in performance of *deep learning* in many domains, there has come a new wave of approaches trying to explain the decisions made by these complex machine learning models. Explainability and interpretability is at the heart of how one can deploy NNs approaches in real-word applications and how humans will interact with them. These explanations can help debug models, determine training biases or understand decisions in simpler terms to build trust. With the recent progress of neural IR approaches that has shown state-of-the-art performance on certain benchmarks, the need to understand and analyze why these models are performing better or how they are

connected to the classical IR approaches has become an important topic raised in the IR community [Craswell et al., 2017, 2018].

1.1 Motivation

Reproducibility The rapid increase of new neural ranking models proposed in the literature, with each using different pre-processing strategies, different hyper-parameter choices, different benchmark datasets (public or large-scale private clickthrough data) and different scale of computational resources, highlights the issue of reproducibility and applicability of these new approaches. This has led to the focus of reproducibility of these methods that have been discussed in the IR community lately at relevant workshops [Craswell et al., 2017, 2018] and also there has been the release of the large-scale dataset (MS-MARCO) [Nguyen et al., 2016] for the passage re-ranking task[1] and TREC 2019 deep learning track[2] for benchmarking emerging neural IR approaches.

Interpretability The explainability of neural ranking models is largely unexplored. There have been few papers that focused on the interpretability of neural ranking models [Singh and Anand, 2019] and on the diagnosis or analysis of different model components of the model empirically [Cohen et al., 2018, Pang et al., 2017a] or with the use of diagnostic datasets based on formal retrieval constraints [Rennings et al., 2019]. It is interesting to see to what extent we can use the explainability approaches proposed in other domains (image or text classification) to understand the decisions of neural networks, which can be applied in the context of IR. Thus, its a challenging and promising direction of research in neural IR.

1.2 Contributions

The work in this thesis can be divided into two parts–the first part focuses mainly on the reproducibility of various neural ranking models (both *representation-based* and *interaction-based*); and the second part focuses on the question–*why* a particular document is relevant to a query for the various neural ranking models implemented part of the reproducibility experiments.

[1] https://github.com/dfcf93/MSMARCO/blob/master/Ranking/README.md
[2] https://trec.nist.gov/pubs/call2019.html

We address the following questions in this thesis:

- What are the challenges in the reproducibility of the neural ranking models on the standard TREC Robust04 benchmark collection? Are we able to reproduce the same retrieval performance as mentioned in the papers? If not, why?

- How can we adapt the *model-introspective* approach DeepSHAP to explain the output of neural ranking models?
- How different are the DeepSHAP explanations across the various implemented neural ranking models?
- How do the explanations from DeepSHAP compare with the explanations from the *model-agnostic* approach LIME?

1.3 Outline

The rest of the thesis is organized as follows: In chapter 2, we extensively cover all the related work of neural ranking models. In chapter 3, we describe the fundamentals needed to understand the existing neural IR approaches and the interpretability approaches that we used in this thesis to understand the neural IR models. In chapter 4, we discuss in detail the various neural ranking models along with the experimental setup and a discussion of the results for each model. In chapter 5, we describe how existing interpretability approaches are adapted for the context of *ad-hoc* retrieval and how they can be used to explain neural ranking models along with the experimental evaluation of our approaches. Finally, in chapter 6 we conclude our work and propose directions for future research on the topic.

Chapter 2

Related Work

In this chapter, we will discuss the various *deep learning* approaches that have been applied to information retrieval. We initially review the literature that covers the use of pre-trained word embeddings in traditional IR approaches in section 2.1. We then discuss neural network architectures designed specifically for relevance ranking in IR in section 2.2. These models have included insights from traditional retrieval models as neural building blocks. The neural ranking approaches are broadly categorized into two groups: *representation-based* models (section 2.2.1) and *interaction-based* models (section 2.2.2) [Guo et al., 2016]. A comparison of the neural ranking model architectures can be observed in Table 2.2 and the datasets used in the experimental setups can be seen in Table 2.1.

Finally, in section 2.3 we discuss the various *interpretability* approaches that have been used to provide explanations to understand the predictions of machine learning models. We also focus on works that have been proposed recently which apply *interpretability* approaches to understand ranking models in section 2.3.1.

2.1 Word Embeddings in IR

In this section, we cover earlier work that incorporated pre-trained word embeddings into traditional ad-hoc retrieval approaches. The papers are categorized into *Implicit*–using

vector similarity in the embedding space that is then used in language modelling frameworks and *Explicit*–using word embeddings aggregated for distributed representation of large units of text and then ranking, as discussed in [Onal et al., 2018].

Implicit In the Generalized Language Model (GLM) [Ganguly et al., 2015] approach they incorporated the semantic similarity between query and document or collection terms measured through the cosine similarity between word embeddings (`word2vec` CBOW) into the query-likelihood language modelling retrieval approach. They build on the "noisy channel" translation model [Berger and Lafferty, 1999], where they no longer assume that the query term is generated by sampling independently from either the document or collection but through a generative process that may transform a different term t' from either the document or collection into the observed query term t. In the experiments conducted on TREC 6-8 and Robust using Lucene show improvements over both unigram query-likelihood (LM with Jelinek Mercer smoothing) and LDA smoothed LM.

Similarly, Neural Translation Language Model (NTLM) [Zuccon et al., 2015] also integrates word embeddings into the Berger and Lafferty [1999] translation model approach to query-likelihood IR. They estimate the translation probability between terms as the cosine similarity between the terms divided by the sum of the cosine similarities between the translating term and all terms in the vocabulary. Previous state-of-the-art translation models used mutual information (MI) to calculate the translation probabilities. They conduct experiments on TREC datasets AP87-88, WSJ87-92, DOTGOV, and MedTrack and show that NTLM moderately improves over the state-of-the-art (SOTA) MI approaches. The results show that NTLM gets moderate improvements over SOTA based on small improvements over a large number of queries instead of large differences over few queries. Analyzing the various hyper-parameters of the word embeddings shows that NTLM is robust to the choices of embedding dimensionality, context window size, model objective (CBOW vs Skip-gram). Regarding the choice of training corpus for the trained embeddings and its effects on retrieval performance, they show that effectively the best performance is obtained from the same collection on which the retrieval is evaluated but using a different corpus doesn't statistically degrade the retrieval effectiveness.

Explicit In the Dual Embedding Space Model (DESM) [Mitra et al., 2016, Nalisnick et al.,

2016], the authors highlight the importance of retaining both the input and output embeddings from the `word2vec` model after training. They observed that within the same embedding space (IN or OUT), neighbors are functionally similar words. However, the neighbors of the word using the IN-OUT vector cosine similarity are topically similar words. For example, for the term *harvard*, the topically similar terms are the ones that likely co-occur with *harvard* (e.g. *faculty*, *alumini*) and the functionally similar ones are likely to occur in similar contexts (e.g. *yale*, *nyu*). In DESM, query words are represented using the IN vectors and document words with the OUT vectors. A document embedding representation is created by taking an average of the normalized document word vectors. Then, the query-document relevance is computed by taking the average cosine similarity between each query term and document embedding. The word embeddings are generated using `word2vec` CBOW model. They conduct experiments on a proprietary Web collection (Bing) with explicit and implicit relevance judgements comparing their approach with BM25 and latent semantic analysis (LSA) [Deerwester et al., 1990] baselines. All Out-Of-Vocabulary (OOV) terms are ignored for the DESM approach but retained for the baselines. In their experiments, they show that DESM is a poor standalone ranker when there is a large set of candidate documents, so they use a *telescoping* setup [Matveeva et al., 2006] where a small candidate set of documents are initially retrieved using the Bing search engine, and then re-ranked by DESM. The re-ranking results show significant improvements over the baselines, with the best performance obtained when using word embeddings trained on queries and using IN-OUT embeddings for ranking.

In Roy et al. [2016], instead of finding a single vector representation of a document, they are modelled as a mixture distribution that generates the terms of the document. This distribution is estimated using k-means clustering of the term embeddings in the document. The likelihood of the query is estimated by the average cosine distance of each query term to the cluster centroid vectors of the document. This centroid-based query likelihood function is then combined with standard LM based likelihood for ranking. It shows significant improvements over LM with Jelinek-Mercer smoothing baseline on the TREC 6-8 and Robust adhoc task topics. For efficiency, the global vocabulary is clustered using `word2vec` embeddings beforehand and the document clusters are created by grouping the terms according to their global clusters–the cluster's centroids are the average of the word vectors in that group.

Table 2.1: Datasets used in experimental setups

English data	Papers
Bing Query Logs and Web crawl	DESM (Mitra et al. [2016], Nalisnick et al. [2016]), DSSM (Huang et al. [2013]), C-DSSM (Shen et al. [2014a,b]), DUET (Mitra et al. [2017]), Conv-KNRM (Dai et al. [2018])
AOL Query Logs	Nie et al. [2018a,c], FNRM (Dehghani et al. [2017]), SNRM (Zamani et al. [2018])
PubMed 2018	PACRR-DRMM, ABEL-DRMM, POSIT-DRMM (McDonald et al. [2018])
TREC 1-2 Adhoc (AP 88-89)	Zamani and Croft [2016]
TREC 1-3 Adhoc	NTLM (Zuccon et al. [2015]), NPRF (Li et al. [2018b])
TREC 6-8 Adhoc	GLM (Ganguly et al. [2015]), Roy et al. [2016]
TREC DOTGOV	NTLM (Zuccon et al. [2015])
TREC MedTrack	NTLM (Zuccon et al. [2015])
TREC 2007-2008 Million Query	DeepRank (Pang et al. [2017b]), HiNT (Fan et al. [2018]), DeepTileBars (Tang and Yang [2019])
TREC Robust 2004	GLM (Ganguly et al. [2015]), Roy et al. [2016], Zamani and Croft [2016], Diaz et al. [2016], MatchPyramid (Pang et al. [2016]), DRMM (Guo et al. [2016]), PACRR-DRMM (McDonald et al. [2018]), NPRF (Li et al. [2018b]), Dehghani et al. [2018a,b, 2017], EPV (Ai et al. [2016a,b]), SNRM (Zamani et al. [2018])
TREC GOV2	Zamani and Croft [2016], EPV (Ai et al. [2016a,b]), DeepRank (Pang et al. [2017b]), HiNT (Fan et al. [2018])
TREC ClueWeb09-Cat-B	Diaz et al. [2016], DRMM (Guo et al. [2016]), Nie et al. [2018a,c], Dehghani et al. [2018a, 2017], Conv-KNRM (Dai et al. [2018]), DeepTileBars (Tang and Yang [2019]), SNRM (Zamani et al. [2018])
TREC WebTrack 09-11	DRMM (Guo et al. [2016])
TREC WebTrack 10-12	DeepTileBars (Tang and Yang [2019])
TREC WebTrack 09-12	K-NRM, Conv-KNRM (Dai et al. [2018]), Dehghani et al. [2018a], SNRM (Zamani et al. [2018])
TREC WebTrack 09-14	PACRR (Hui et al. [2017]), Co-PACRR (Hui et al. [2018])
Chinese data	*Papers*
Sogou Query Logs	K-NRM (Xiong et al. [2017]), Conv-KNRM (Dai et al. [2018]), DeepRank(Pang et al. [2017b])

Query Expansion In [Diaz et al., 2016, Zamani and Croft, 2016], they propose a new query expansion based language model (QLM) that uses word embeddings. In Diaz et al. [2016] they propose a query expansion language model that uses word embeddings trained on topic-constrained corpus. They show that word embeddings learnt from the entire corpus, can be very general. So they train word embeddings on subsets of documents that are retrieved for all topics using the query-likelihood retrieval model. The original query language model is interpolated with the new query expansion language model that is defined by the weights of terms computed using UU^Tq where U is the $|V|$ x d embedding matrix and q is $|V|$ x 1 term matrix. The locally trained embeddings are compared against the globally trained embeddings on TREC 1-2, Robust04, and ClueWeb09-Cat-B web corpus. Those models based on the local embeddings are shown to perform best on retrieval effectiveness of NDCG@10. They also show that word embeddings learnt on a topically-constrained sample of documents from a external large corpus achieves the highest retrieval effectiveness.

In Zamani and Croft [2016], they propose two query expansion language models–in the first, $p(w|q)$ is computed by multiplying likelihood scores $p(w|t)$ given individual query terms as they consider conditional independence between query terms whereas in the second, $p(w|q)$ is computed using an additive model over $p(w|t)$ scores. The $p(w|t)$ scores are computed based on similarity of word embeddings which is a sigmoid function applied on top of cosine similarity to increase the discriminative ability. They also propose a relevance model [Lavrenko and Croft, 2001], which computes a feeback query language that uses embedding similarities along with term matching. They compare the proposed QLMs with maximum likelihood estimation (MLE), GLM [Ganguly et al., 2015] on AP, Robust and GOV2 collections. They show that the first QLM is more effective in query expansion experiments.

2.2 Neural Network Architectures for IR

In deep IR, the computation of relevance of a document to a given query can be formalized as a text matching problem as follows. Given two texts $T1$ and $T2$, the similarity between the two can be measured using a scoring function on the representations of each text.

$$match(T1, T2) = F(\Phi(T1), \Phi(T2)) \tag{2.1}$$

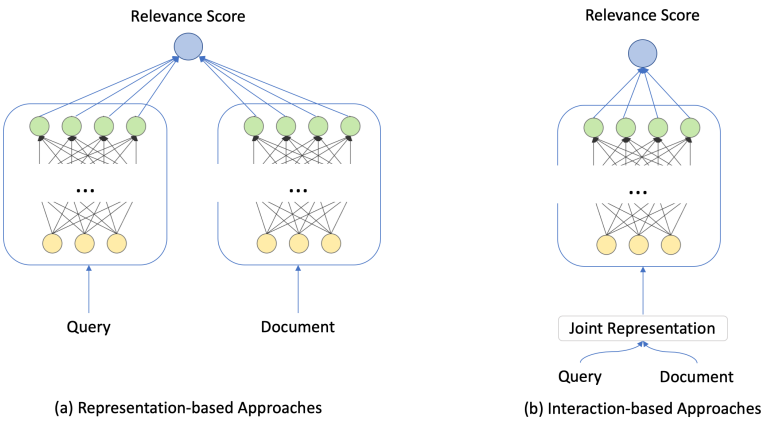

Figure 2.1: Types of Neural Network Architectures for IR.

where Φ is a function to match each text into a representation vector and F is a scoring function on the interactions between the texts. Based on this deep IR can be roughly categorized into *representation-based* and *interaction-based* models as shown in fig. 2.1.

In *representation-based* models (section 2.2.1), the focus is on learning meaningful semantic representations of the text through several hidden layers using a deep neural network (DNN) and estimating the relevance score by using a matching function on the last level representations of the query and document. In this approach, Φ is a complex representation mapping function and F is a simple matching function like cosine or dot similarity. For example, in DSSM [Huang et al., 2013], Φ is a feed forward neural network and F is cosine similarity.

In *interaction-based* models (section 2.2.2), the focus is on learning salient interaction patterns from the local interactions between the query and document that is fed as input into a DNN. In this approach, Φ is a simple mapping function while F is a complex deep model. For example, in DRMM [Guo et al., 2016], Φ is maps each text into a sequence of word vectors and F is a feed forward neural network over the interaction matrix built from the word vectors of the two texts.

Whereas, in DUET [Mitra et al., 2017] they combine both *representation-based* and *interaction-based* models by employing two separate deep neural network models–*local* model that estimates the relevance score according to exact matches between query and document terms, and *distributed* model that estimates relevance by matching dense lower-dimensional representations of both query and document text. The final relevance

score is the sum of scores obtained from the two sub-networks. They carry out experiments on commercial search log data from Bing and compare their approach against traditional baselines (BM25, QL, DM) and neural baselines (DRMM, DSSM, CDSSM, DESM) and show that it largely outperforms the baselines.

2.2.1 Representation-Based Models

The Deep Structured Semantic Model (DSSM) [Huang et al., 2013] is the earliest work in neural *representation-based* models for ad-hoc retrieval. The DSSM model is composed of a deep neural network with three non-linear layers on top of a word hashing layer. The query and document are first mapped to high-dimensional term vectors (*bag-of-words* representation). After this the term vectors are mapped to trigram vectors by the word hashing layer, in order to cope with the large size of the vocabulary (reduces vocabulary size from 500K to 30K).

In Convolutional Deep Structured Semantic Model (C-DSSM) [Shen et al., 2014b] they extend DSSM by including a convolutional neural network (CNN) with max-pooling followed by a semantic layer to get high level representations of text. It first uses word hashing to project each word into a tri-gram vector. Then the convolutional layer projects each word vector within a context into a local contextual vector. Following which max-pooling is applied that picks the most salient local features to form a fixed-length global vector for queries and documents. Since both DSSM and C-DSSM fail to capture contextual information accurately, Shen et al. [2014a] propose a Convolutional Latent Semantic Model (CLSM) that is built on top of DSSM. In this, the first layer consists of *word-n-grams* constructed using a contextual sliding window over the input word sequence followed by letter tri-gram representation that maps each word-n-gram to its tri-gram representation like in DSSM. Then, a convolutional layer transforms these tri-grams into contextual feature vectors using a convolutional matrix shared between the word-n-grams. Then, max-pooling is applied against each dimension of the contextual feature to pick salient word-n-grams to form a fixed length vector. Finally, a semantic layer performs a non-linear transformation to obtain a semantic representation of the text.

All of the above models are trained and evaluated on large-scale datasets from Bing. The models are discriminatively trained on query-document title pairs from the clickthrough

data such that the conditional likelihood of the clicked document for a given query is maximized (*cross-entropy* loss). These models are shown to outperform baselines comprising of Word Translation Model, TF-IDF, BM25 in terms of retrieval effectiveness (NDCG @1,3,10). However, in a later work [Guo et al., 2016] they show that both DSSM and C-DSSM do not perform as well as traditional IR models when using the entire document body.

In recent work [Nie et al., 2018a], they compare a representation-based approach (C-DSSM without word transformation to tri-letter representation) and interaction-based approach (MatchPyramid but with additional CNN layer) under the same test conditions (experimental setup) using weak supervision [Dehghani et al., 2017] (section 2.2.3) with AOL query logs to generate a large amount of training data to be able to reasonably train both models on the ClueWeb09-Cat-B collection. However, their experimental results are inconsistent with that of Dehghani et al. [2017], where they show that a neural model based on representation learning, weakly supervised by BM25, can give better retrieval effectiveness than the BM25 baseline. This could be due to the difference in dense input representation, where in Dehghani et al. [2017] they concatenate the IDF weighted element-wise sum of the term's embedding vectors of both the query and document. Finally, in Nie et al. [2018a] to improve the retrieval effectiveness of the representation-based model, they employ matching at different levels of abstraction instead of a matching applied on the representation obtained from the final layer for computing a relevance score like in previous works. They build interaction matrices from the embeddings or convolved vectors between the query and document at each layer and then pick the strongest p signals across each row, average it and then sum up the averages of every query representation to get a matching score for that layer. These matching scores are aggregated through a softmax gate to get a global matching score for the query and document.

In Ai et al. [2016a,b] they investigate the use of PV-DBOW [Le and Mikolov, 2014] as a document language model for retrieval. They propose three improvements over the original model for retrieval: (1) use document-frequency based rather than corpus-frequency based negative sampling so that frequent words will not be suppressed too much; (2) regularization over document representation to prevent model overfitting to short documents; and (3) joint-learning objective that considers both document-word

and word-context associations to better model word substitution relations. They show higher effectiveness over LDA-based LM and QL for Robust04 and GOV2 collections.

2.2.2 Interaction-Based Models

2.2.2.1 Bag-of-words based

Deep Relevance Matching Model (DRMM) [Guo et al., 2016] is one of the first *interaction-based* models to show improvements over traditional IR models (bag-of-words baselines). The query-document local interactions are first mapped into fixed-length *matching histograms* to distinguish between exact matching and soft matching signals using term embedding cosine similarities. These signals are fed into a *feed forward* NN and a *term-gating* network that models term importance to finally get a relevance score for the query-document pair. In their experiments over Robust04 and ClueWeb-Cat-09B collections, they show significant improvements over the baselines: BM25, QL, representation-based models (DSSM, C-DSSM) and interaction-based models (MatchPyramid). The term embeddings are trained on the respective collections and they also show empirically that as the CBOW dimensions increases, the DRMM performance first increases and then slightly drops.

Similar to DRMM, Kernel-based Neural Ranking Model (K-NRM) [Xiong et al., 2017] first creates a query-document interaction matrix that computes the cosine similarity between the term embeddings of both query and document. Then, they use multiple *gaussian kernels* to obtain different levels of exact and soft-matching features from the interaction matrix which is then used as an input to a ranking layer which is a linear layer with *tanh* activations to produce the final relevance score. The model is trained end-to-end using a max-margin learning-to-rank loss function. During *learning*, the kernels convert the loss and adjust the word embeddings to produce a soft matching signal that better seperates relevant and irrelevant documents. They conduct extensive experiments on a Chinese commercial search engine query logs (Sogou) and show significant improvement over traditional baselines (BM25), learning-to-rank approaches (RankSVM) and neural IR baselines (DRMM, C-DSSM). They also carry out an analysis that shows using a model without multi-level soft matches or embedding learning (using pre-trained `word2vec` embeddings) quickly diminishes the performance of K-NRM.

2.2.2.2 Word n-gram based

MatchPyramid [Pang et al., 2016] is one of the first works that tries to capture hierarchical matching patterns based on *n-gram* matches from the local interaction matrix of the query-document. They employ a convolutional neural network (CNN) comprising of one convolutional layer and dynamic max pooling layer on top of the interaction matrix to extract hierarchical matching patterns. Finally, the output of the CNN is fed into a MLP to produce a ranking score. They conducted extensive experiments to study the impact of pooling sizes, similarity functions (cosine, indicator, gaussian) and kernel sizes on the retrieval performance on the Robust04 collection. They achieve a better retrieval performance over traditional IR models (BM25, QL), and representation-based models (pretrained DSSM, C-DSSM), using a pooling length equivalent to paragraphs, a similarity function that differentiates between exact and semantic matches and small kernel size.

In Multi-level Abstraction Convolutional Model (MACM) [Nie et al., 2018a,c], they use a model similar to MatchPyramid but with an additional convolutional and max-pooling layers and integrates the matching scores at different levels of abstraction. They determine matching scores at each abstraction level by flattening the max-pooled layer and using a MLP to obtain a score. The matching scores are aggregated through a softmax gating mechanism that considers the importance of each abstraction level by computing the max interaction values across each row of the interaction matrix or convolution feature maps and then summing up these max interaction values to get a global importance value. They show that the model trained with weak supervision using AOL query logs can outperform BM25 baseline on the ClueWeb-Cat-09B collection. They also compare with model variants without multi-level matching like in MatchPyramid and show that it outperforms these variants as well.

Convolutional Kernel-based Neural Ranking Model (Conv-KNRM) [Dai et al., 2018] improves upon the K-NRM model by employing CNN to compose adjacent word embeddings into n-gram embeddings. The n-gram embeddings from both query and document can be combined into multiple interaction matrices that allows cross-matching of n-grams of different lengths. They then use the kernel pooling and ranking layer to combine the n-gram soft matches into a final ranking score [Xiong et al., 2017]. They conducted experiments using English search logs from Bing (WSDM'09 workshop) and Chinese search

logs from Sogou and show the advantages of this approach over the baselines: BM25, RankSVM, DRMM, C-DSSM, MatchPyramid and K-NRM. They also carry out experiments on ClueWeb-Cat-09B collection, where they use a domain adaptation method as there isn't sufficient training data. So, they pre-trained Conv-KNRM using the Bing search logs to be able to train the word embedding and convolutional layers effectively and then they use the relevance judgements from TREC 09-12 web tracks to re-train the soft matching kernel pooling and ranking layer keeping the embedding and convolutional layers 'frozen'. This approach is shown to outperform all of the baselines mentioned above.

In Position-Aware Convolutional Recurrent Relevance model (PACRR) [Hui et al., 2017], they use a combination of convolutional kernels to capture interactions that includes unigrams, bi-grams and tri-grams matches; k-max pooling along the query dimension to preserve important signals from different query terms; and recurrent layer to combine signals from different query terms to produce a global relevance assessment. Later, they improve on PACRR and introduce Co-PACRR [Hui et al., 2018] which is a context-aware variant that takes into account the local and global context of the matching signals. For this they use three components:(1) *disambiguation* component for considering the matching signal with the local context in which it occurs, this is done by introducing an input that captures the similarity between the query vector and all document contexts defined by a window; (2) *cascade k-max pooling* to account for the location of matching signals by applying k-max pooling at multiple positions in the document instead of pooling on the entire document; (3) *shuffling combination* layer to regularize the model so that it doesn't learn the absolute position of terms within the query as this could influence the aggregation layer to down-weight positions that are generally zero padded. They compare their approaches with state-of-the-art neural models including DRMM, K-NRM, local model of DUET and MatchPyramid on six years of TREC WebTrack benchmarks and show significant improvements over all baselines. They also note that their approach after re-ranking QL ranked results produce runs that are ranked within the top-3 runs on atleast 5 years based on ERR@20.

In McDonald et al. [2018], they propose several extensions to DRMM so as to account for the context in which the query terms appear. First, they introduce PACRR-DRMM a simple model that uses DRMM to independently score each of the (document-aware) query term encodings that are obtained from PACRR and aggregates these scores using

a linear layer. They then propose two additional models that uses *context-sensitive* term embeddings obtained from pre-trained embeddings by using a standard BiLSTM encoding scheme: (1) Attention-Based ELement-wise DRMM (ABEL-DRMM) in which a document representation for each q-term (d_{q_i}) is first computed by taking a sum of the context-sensitive encodings of d-terms, weighted by their attention score that is computed using a softmax over the dot product between that q-term and d-term encoding and then the (document-aware) q-term encoding is the hadamard product between d_{q_i} and q-term encoding; (2) POoled SImilariTy DRMM (POSIT-DRMM) in which document-aware q-term encoding is computed by first concatenating attention scores that are obtained from cosine similarity between q-term and d-term encodings and then creating a representation that includes the max-pooled attention score and average of the k-max pooled attention scores. They compare against BM25, and neural IR baselines (DRMM, PACRR) on the TREC Robust04 collection and show significant improvements over all baselines.

2.2.2.3 Document context based

DeepRank [Pang et al., 2017b] proposes a new architecture that better models the human judgement process while assigning relevance to a query-document pair. The model first employs a *detection strategy* to extract query-centric contexts from the document, that is, the contexts with a query term in the center. Then the *measure network* determines the local relevance between query and query-centric contexts by creating an 3D input tensor that includes query, and query-centric word representations along with the word-level interaction matrix between the word embeddings of the query & query-centric context (cosine/indicator). They then use either a CNN or 2D gated recurrent unit (2D-GRU) to capture the local relevance from the input tensor producing a *local relevance vector* containing matching patterns extracted from different kernels. Finally, the *aggregation network* aggregates the local relevances at a query term level, and then combines it by weighting the query term importance using a term gating mechanism. The query-term level relevance is computed by first grouping the query-centric contexts that have the same query word together and then using a RNN to aggregate these local relevance vectors sequentially which also considers the position of the query-centric contexts in the document. They conduct experiments on both benchmark LETOR4.0

data (MQ2007, MQ2008) and large scale Chinese clickthrough data (Sogou). The experimental results show that: (1) Existing neural IR models such as DSSM, CDSSM, DRMM perform worse than the pairwise (RankSVM, RankBoost) and listwise (ListNet, AdaRank, LambdaMart) learning-to-rank methods using hand-crafted features; (2) DeepRank significantly improves the performance across both existing NRMs and learning-to-rank baselines.

Building upon DeepRank, Fan et al. [2018] propose a HIerarchical Neural maTching model (HiNT) to learn diverse relevance patterns by a data-driven approach to allow relevance signals from different granularities (document-wide or passage-level) compete with each other to obtain a final relevance score. The model consists of two stacked components: (1) *local matching layer* which produces a set of local relevance scores between a query and each passage of a document (obtained by a fixed-sized sliding window). The relevance scores are computed by building two 3D input tensors as described in Pang et al. [2017b]–one for exact matching and one for semantic matching and both are given as input to a 2D Gated-RNN (spatial GRU) and the last hidden representation is taken as the matching output; (2) *global matching layer* is a hybrid network that aggregates passage level relevance at different granularities–*independent decision* model assumes that the passage-level relevance scores are independent and *accumulative decision* model applies LSTM sequentially across the passage-level signals to generate an accumulated relevance scores at different positions. Then, k-max pooling is applied over the dimensions to pick the top-k signals from either the passage-level or accumulated relevance signals, which is then concatenated as input to a MLP for the final relevance score. They follow a similar experimental setup as Pang et al. [2017b] and show improvements over traditional document-wide, passage-level retrieval models, learning-to-rank methods and neural IR models.

In DeepTileBars [Tang and Yang, 2019], they first split the documents into topical segments using the TextTiling algorithm [Hearst, 1994] which splits documents at positions where a change in topic is detected and they then create a query-segment interaction matrix which is fed into the neural IR model (DeepTileBars) to obtain a relevance score. Each cell in the query-segment interaction matrix comprises of 3 values–term frequency of query word in segment, inverse document frequency of query word, and similarity score (gaussian kernel) of most similar word in the segment using word embeddings. DeepTileBars first has multiple CNNs with different kernel sizes that are employed to

learn relevance patterns at different levels of granularity in the document's topic hierarchy. Then LSTMs of the same number as CNNs are used accumulate the relevance signals from the output of the CNNs, this gives relevance scores of the document at different granularities. All the different relevance scores are aggregated using a MLP to get a final global relevance score for the document. They conduct experiments on the TREC WebTrack (2010-12) and LETOR4.0 (MQ2008) and show improvements over traditional IR baselines (BM25, QL) and neural IR baselines (DRMM, MatchPyramid, DeepRank, HiNT).

2.2.3 Weak Supervision for Neural IR

Recently, Dehghani et al. [2017] propose a weak supervision method to use large amounts of unsupervised data to generate 'weak' labels that can be used as a signal to learn neural IR models. They examined neural ranking models with different architectures, training objectives (pointwise, pairwise), and different input representations from encoding query-document pairs into sparse or dense vectors to learn dense embeddings for each text. The models are trained on billions of training examples obtained by using AOL query logs as a query set and retrieving *pseudo-relevant* documents for each query using BM25 (weak supervision signal). They observed that by using just the training data obtained from the BM25 weak annotator they are able to outperform BM25 on the test dataset. This is the case only with the models that uses an embedded vector representation which concatenates the weighted element-wise sum of the term embeddings from the query and document. They also carry out analysis to understand what the model learns, if the use of pretrained embeddings from external/target corpus instead of learnt embeddings affects the performances of the models. They demonstrate that training a model with limited amount of supervised data (relevance judgements) after pre-training it on weakly supervised data gives a significant improvement in performance. This approach has been used lately to train various neural ranking models [Li et al., 2018a, Nie et al., 2018a,c, Zamani et al., 2018]. Most recently, Zamani and Croft [2018] provided a theoretical foundation for explaining the successful empirical results achieved by weakly supervised neural IR models.

"Fidelity-Weighted Learning" [Dehghani et al., 2018a,b] is a semi-supervised student-teacher approach for training deep neural networks with weakly supervised data instead

of treating the weakly labeled samples uniformly. The *student* network is initially trained on weak data to learn an initial task-dependent data representation (embedding dense vector representation in Dehghani et al. [2017]). This data representation is then used with the *teacher* network which is a Bayesian semi-supervised approach that learns to predict the strong data (limited relevance judgements). Then, the *teacher* is used to generate a new weakly supervised data with confidence scores for each pair of data. This new weak dataset is used to fine tune the parameters of the student network. The experiments on Robust04 and ClueWeb-Cat-09B show that this approach provides improvements over the weakly supervised setup in Dehghani et al. [2017].

2.2.4 Neuro-Pseudo Relevance Framework for Neural IR

Recently, in Li et al. [2018b] they propose a Neuro Pseudo Relevance Framework (NPRF) that enables the use of pseudo relevance feedback with existing neural IR models. The existing neural IR models are used as scorers to evaluate the target document with respect to the top-ranked queries and the query without changing their architectures. They show two examples of the NPRF framework with two neural IR models–DRMM and KNRM, and evaluate their performance on TREC benchmark datasets (TREC 1-3, Robust04) for ad-hoc retrieval. A brief description of the architecture is given in section 4.4 and we detail the reproducibility experiments on TREC Robust04 collection that we carried out on the NPRF-DRMM variant from this paper in section 4.4.1.

2.2.5 Learning Sparse Representations for Indexing

In [Zamani et al., 2018] they propose a Standalone Neural Ranking Model (SNRM) that can retrieve relevant documents from a large-scale collection, instead of re-ranking a small set of candidate documents that are retrieved by a first stage ranker (BM25, QL) which is the approach used in all previous neural ranking models. They propose a network architecture to learn *latent sparse* representations for each continuous n words of both queries and documents in the same semantic space. The learned sparse representations of n-grams are then aggregated using average pooling to get a sparse representation for the entire text. The fully connected feed forward network has a hourglass structure with small number of units in the middle layers to learn a low-dimensional representation of the data and the upper layers have a large number of units to get a sparse output

representation from the sequence of n embedding vectors concatenated (n-gram). These sparse representations are trained with weak supervision using a training objective that is a combination of pairwise max-margin loss and sparsity objective that minimizes the L_1-norm of the sparse representations. After training, the inverted index is constructed by considering each index of the learned representation (ϕ) as a "latent" term, so then for each document in the collection if the i^{th} index of $\phi_D(\text{d})$ is non-zero, it is added to the inverted index for the latent term i. During inference, the query sparse representation $\vec{q}$ is first obtained and for each non-zero element in $\vec{q}$ the corresponding documents from the inverted index are retrieved and scored using the dot product.

Model	I/P Repr.	Training workload	Word Embeddings	Evaluation Testset	Impl. Trainable	OOV handling	Baselines	Training Obj.
DESM	BOW + Embeddings	—	`CBOW` & `SGNS` Bing	Bing Prop., Robust04, CW	✓	skips	BM25, LSA	—
DUET	query-doc interaction matrix query & doc tri-gram vector	click-through data 200K Bing	—	Bing Prop.	weak sup.	✗	BM25, LSA, QL, DRMM, DSSM, CDSSM, DESM	log-likelihood
DSSM	Bag-of-character tri-gram vector	click-through data 16K Bing	—	Bing Prop.	weak sup.	✗	BM25, LSA, DAE	cross entropy
CDSSM	Bag-of-character tri-gram vector	click-through data 16K Bing	—	Bing Prop.	weak sup.	✗	BM25, LSA, DSSM	cross entropy
DRMM	query-doc interaction matrix (cosine)	judgements Robust04, CW WebTrack 09-11	`CBOW` Robust04	Robust04, CW	✓	Ind. match 1/0	QL, BM25, DSSM, CDSSM, MP	max margin
Match-Pyr. MP	query-doc interaction matrix (cosine, ind., dot, gaussian)	judgements Robust04	`Glove` Wikipedia	Robust04 titles	✓	NA.[1]	QL, BM25, DSSM, CDSSM	max margin
PACRR Co-PACRR	query-doc interaction matrix (cosine)	judgements CW WebTrack 09-14	`word2vec` Google-News	CW WebTrack 09-14	✓	Retrain OOV words	DUET-L, KNRM, DRMM, MP	max margin Co-PACRR (cross entropy)

Table 2.2: Comparison of Neural IR model architectures and experimental setups. (continued)

[1]Not mentioned in the paper.

Model	I/P Repr.	Training workload	Word Embeddings	Evaluation Testset	Impl. Trainable	OOV handling	Baselines	Training Obj.
K-NRM	query-doc interaction matrix (cosine)	click-through data 100K Sogou (Chinese)	Trained with model	Sogou Prop.	weak sup.	✗	BM25, QL, DSSM, CDSSM, DRMM, RankSVM	max margin
Conv-KNRM	query term embed matrix & doc term embed matrix	click-through data 100K Sogou, 100K Bing, CW 09-12	Trained with model	Sogou Prop., Bing Prop., CW 09-12	weak sup.	✗	BM25, MP, DRMM, CDSSM, KNRM, RankSVM	max margin
MACM	query-doc interaction matrix (cosine)	AOL query logs (weak sup.)	`Glove` Wikipedia	CW 09-12	weak sup.	skips	BM25, QL	max margin
FNRM Dehghani et al. [2017]	query & doc weighted avg. embedding repr.	AOL query logs	`word2vec` Google-News	Robust04, CW 09-12	weak sup.	✗	BM25	max margin
DeepRank	3D tensor of query, query-centric word repr. & interaction matrix	LETOR4.0 (MQ2007 & MQ2008), click-through data (Sogou)	`CBOW` Wikipedia	LETOR4.0, Sogou Prop.	✓	NA.	BM25-Title, RankSVM, RankBoost, ListNet, AdaRank, LambdaMart, DSSM, CDSSM, DRMM, MP	max margin
HiNT	3D tensor of query, passage word repr. & interaction matrix	LETOR4.0 (MQ2007 & MQ2008)	`CBOW` Wikipedia	LETOR4.0	✓	rand. sample (-0.02, 0.02)	BM25, AdaRank, LambdaMart, DSSM, DRMM, Duet, DeepRank	max margin
DeepTileBars	3D tensor of query-segment interaction matrix (*tf*, *idf*, *gaussian sim.*)	judgements CW WebTrack 10-12	`word2vec` Google-News	CW 10-12, LETOR4.0 (MQ2008)	✓	NA.	BM25, QL, DRMM, MP, HiNT, DeepRank	cross entropy
SNRM Zamani et al. [2018]	query and doc sparse embeddings (BOW)	AOL query logs (weak sup.)	`Glove` Wikipedia	Robust04, CW 09-12	weak sup.	✗	QL, SDM, RM3, FNRM, CDSSM	max margin + L_1-norm sparsity obj.

Table 2.2: Comparison of Neural IR model architectures and experimental setups.

2.3 Interpretability in ML

There are two main approaches to interpretability in machine learning models: *model agnostic* and *model introspective* approaches. Model agnostic approaches [Ribeiro et al., 2016, 2018] generate post-hoc explanations for the original model by treating it as a black box by learning an interpretable model on the output of the model or by perturbing the inputs or both. Model introspective approaches on one hand include "interpretable" models such as decision trees [Letham et al., 2015], attention-based networks [Xu et al., 2015], and sparse linear models [Ustun and Rudin, 2016] where there is a possibility to inspect individual model components (path in a decision tree, feature weights in linear models) to generate useful explanations. On the other hand, there are gradient-based methods like Simonyan et al. [2014] that generates attributions by considering the partial derivative of the output with respect to the input features. Following this, there were many works [Arras et al., 2017, Bach et al., 2015, Lundberg and Lee, 2017, Shrikumar et al., 2017] that generate attributions by inspecting the neural network architectures.

2.3.1 Interpretability in Ranking Models

Recently there have been few works focused on interpretability [Singh and Anand, 2019] and diagnosis of neural IR models [Cohen et al., 2018, Pang et al., 2017a, Rennings et al., 2019]. In the diagnostic approaches, they use the formal retrieval constraints ("axioms") defined for traditional retrieval models to find the differences between neural IR and learning-to-rank approaches with hand-crafted features through a manual error analysis [Pang et al., 2017a] or build diagnostic datasets based on the axioms to empirically analyse these models [Rennings et al., 2019]. Also, in Cohen et al. [2018] they take a diagnostic approach by probing neural retrieval models by taking each layers weights and using it as an input to a classifier for different types of NLP tasks (sentiment analysis, POS tagging, etc.). The performance on those tasks would provide insight into the type of information encoded in each layer.

Whereas, in Singh and Anand [2019] they built a Explainable Search System (EXS) that adapts a local model agnostic interpretability approach (LIME) [Ribeiro et al., 2016] to explain the relevance of a document for a query for various neural IR models (DESM, DRMM). They provide visual explanations that also helps answer why one document is

ranked higher than another or what is the intent of the query according to the neural ranker.

Chapter 3

Background

In this chapter, we describe the building blocks that are commonly used to design neural network architectures for IR on which we carry out various reproducibility experiments as detailed in chapter 4. We first describe the common input text representations that are commonly used by the existing neural IR models in section 3.1. Then, in section 3.2 we describe a few standard neural network architectures that are commonly used in IR using the categories described in Mitra and Craswell [2018]. Then we cover the different learning objectives and training strategies employed to effectively train these neural ranking models in section 3.3.

In the last section, we detail the two interpretability approaches–LIME (section 3.4.1) and DeepSHAP (section 3.4.2) that is used in this thesis to help understand the ranking decisions made by various neural IR models.

3.1 Input Text Representations

Neural IR models that learn representations of text, take raw text as input. Commonly used input text representations for neural network models in IR are pre-trained embeddings [Guo et al., 2016, Pang et al., 2016] or one-hot representations of terms or character n-graphs [Huang et al., 2013, Mitra et al., 2017]. Few models also take dense representations of longer chunks of text by aggregating the term pre-trained embeddings either by concatenating or taking a weighted sum over all the terms [Dehghani et al., 2017]. These pre-trained embeddings can also be fine-tuned while training the neural

models by directly optimizing it for the IR task of matching query-document pair [Xiong et al., 2017]. Learning the term embeddings end-to-end with model training helps learn soft matching patterns that can help separate relevant from irrelevant documents but this requires large amount of training data, such as, click-through data from commercial search engines or weak supervision [Dehghani et al., 2017] to learn effective representations. The pre-trained embeddings are generally learnt in an unsupervised manner either using an external large corpus or the target corpus itself using neural term embeddings models, such as, `word2vec` (CBOW or skip-gram) [Mikolov et al., 2013] or `glove` [Pennington et al., 2014].

word2vec It consists of two architectures–skip-gram architecture that consists of one hidden layer with both input and output as one-hot term vectors and the model is trained by minimizing the error in predicting a term given one of its neighbouring terms defined by a context window; and Continuous Bag-Of-Words (CBOW) which is similar to skip-gram but the task it to minimize the error in predicting the middle term given a bag-of-words representation of all its neighbouring terms in a window.

Glove It combines global and local context in the learning of embeddings by using a training objective that tries to minimize the squared difference between the dot product of the word vectors of term-neighbour pairs and the logarithm of the word's probability of co-occurrence in the corpus.

3.2 Popular Architectures for IR

3.2.1 Shift invariant neural operations

Shift invariant comprises of *convolutional* and *recurrent* architectures. The key intuition behind these architectures, is that, the meaning of an English sentence in most cases should be consistent irrespective of the location in the document where it appears.

In shift neural operations fundamentally employ a window-based approach over the input space. A fixed size window is moved over the input space with fixed stride in each step. A typically parameterised function - called a *kernel*, or *filter*, or *cell* is applied over each instance of this window. The parameters of the cell are shared across all

window instances, which implies lesser number of total parameters in the model and more supervision per parameter per training sample.

A popular cell implementation includes multiplying with a weight matrix - then the architecture is referred to as convolutional. An example of cell without parameters is *pooling* - which consists of aggregating (sum or max) over all term vectors in the window. The length of the input sequence can be variable in both cases and the length of the output of convolutional or pooling layer is a function of input length. In the *global pooling strategy* there is a single window which spans over the whole input–applied on top of the convolutional layer thus giving a fixed sized output for variable length input.

In convolutional or pooling each window is applied independently. While, in *recurrent* architecture the cell not only considers the current input window but also the output of the previous instance of the cell as its input. Popular RNN archictectures are–Elmon network, Long Short-Term Memory (LSTM) and Gated Recurrent Unit (GRU).

One thing to consider is how the window outputs are aggregated when using convolutional or recurrent layers. Convolutional layers are followed by either pooling or fully-connected layers that perform a global aggregation over all window instances. Fully-connected layer is aware of window position, while as pooling is agnostic to it. A global max-pooling operation could be applied to a variable size input, but it is not suitable for long sequences. Most existing *interaction-based* neural IR approaches in general include convolutional architectures (DUET [Mitra et al., 2017], MatchPyramid, MACM [Nie et al., 2018c], Co-PACRR, Conv-KNRM), whereas a few approaches include a convolutional architecture followed by recurrent architecture that aggregates the various local signals (PACRR, DeepRank, HiNT [Fan et al., 2018], DeepTileBars [Tang and Yang, 2019]).

3.2.2 Auto-encoders

The purpose of auto-encoders is to learn a compressed representation $\vec{x} \in \mathbb{R}^k$ from their high-dimension representation $\vec{v} \in \mathbb{R}^K$ where $k \ll K$. The model is trained by feeding in the high-dimensional input and reconstructing the same representation in the output layer. The model is trained by minimizing the error between the input $\vec{v}$ and the output

of the decoder $\vec{v'}$. Squared loss is employed popularly.

$$\mathcal{L}_{auto-encoder}(\vec{v}, \vec{v'}) = \|\vec{v} - \vec{v'}\|^2 \tag{3.1}$$

One of the earliest neural models was a deep auto-encoder trained in an unsupervised setting on an unlabeled document corpus [Salakhutdinov and Hinton, 2009]. The documents are represented as a bag of terms and uses a one-hot representation for the terms themselves–considering only the top two thousand words in the vocabulary after removing stop words. The model is pre-trained layer-by-layer and then further trained for end-to-end learning. This model generates a condensed binary vector representation of documents. Given a search query, a corresponding hash is computed and then candidate relevant documents are retrieved that match this hash vector. A standard IR model can then re-rank this candidate set of documents. One of the main drawbacks of auto-encoders is that the model is trained on document reconstruction loss which would not model the relevance signals required to match query and documents.

3.2.3 Siamese networks

These models are used for comparing short texts. It is similar to the auto-encoders architecture but it is trained on pairs of inputs. It consists of two models that project the inputs into a common embedding space (dense representation) and predefined metric (cosine, dot) is used to commute the similarity between the embedded vectors. Most of the earlier *representation-based* models employ this type of network (DSSM [Huang et al., 2013], C-DSSM [Shen et al., 2014a]).

3.2.4 Interaction based networks

In Siamese networks, both the query and documents are represented using single embedded vectors. We can instead compare individual parts of the query with individual parts of the document, to aggregate a partial evidence of relevance. This is useful, when dealing with long documents–as they contain a mixture of topics, such a strategy is more effective than representing it using a low dimensional vector. In these approaches, a sliding window is moved over the query and the document text, and each instance of the window of the query is compared with each instance of the document text window.

The terms within each window can be represented in different ways, like, one-hot vectors, pre-trained embeddings or embeddings that are updated during model training. A neural model like convolutional operates over the interaction matrix and aggregates evidence across all pairs of windows compared. These interaction based approaches have been explored for ranking long documents (DUET, MatchPyramid [Pang et al., 2016], PACRR, Co-PACRR [Hui et al., 2017, 2018]).

3.2.5 Lexical and semantic matching networks

Most of the previous networks focuses on learning good representations of the text. However, these representation-based models perform poorly on rare terms and search intents. Thus, approaches that also model lexical matches using deep neural networks perform better (DRMM [Guo et al., 2016], DUET, K-NRM [Xiong et al., 2017]). Neural models that focuses on lexical matching tend to have fewer parameters and can be learned effectively with smaller datasets. In recent neural IR approaches, the input representation consists of 3-dimensional tensors that includes different types of interaction matrices capturing different signals–exact or semantic (HiNT [Fan et al., 2018], DeepTileBars), interactions between context-sensitive or context-insensitive embeddings (POSIT-DRMM [McDonald et al., 2018]), etc.

3.3 Neural Model Learning

In this section, we describe the common learning objectives and training strategies that are employed by existing neural ranking approaches.

3.3.1 Learning objectives

Similar to learning-to-rank (LTR) approaches, the learning objectives for neural IR models can be broadly categorized into–*pointwise* and *pairwise* ranking loss objectives.

Pointwise Obj. In the pointwise approach, given a query q and $d \in D$ belonging to the ideal ranking set from the training dataset, a neural model is directly trained to estimate the relevance label $rel_q(d)$ which is generally a numerical value. The *regression* squared loss defined in eq. 3.2 [Dehghani et al., 2017] and *cross-entropy* loss defined in

eq. 3.3 [Huang et al., 2013, Hui et al., 2018, Mitra et al., 2017] are the common ranking objectives used to train existing approaches.

$$\mathcal{L}_{squared} = ||rel_q(d) - s(q,d)||^2 \tag{3.2}$$

where $s(q,d)$ is the score predicted by the model.

$$\mathcal{L}_{cross-entropy} = -log\Big(\frac{\exp(f(Q,D^+))}{\sum_{D\in N}\exp(f(Q,D))}\Big) \tag{3.3}$$

where D^+ is the relevant document and $N = \{D^+\} \cup D^-$ and D^- is a fixed number of randomly sampled irrelevant documents (either manually judged or picked from the rest of the collection minus the candidate documents retrieved using a first stage ranker).

Pairwise Obj. In this approach, given a query q, and documents (d^+, d^-) where d^+ is ranked higher than d^- with respect to the query, the neural ranking model is directly trained to learn the preference pairs or ranking between pairs of documents with respect to individual queries. The *max-margin* pairwise ranking loss is the most common ranking objective (eq. 3.4) employed by numerous neural IR models [Guo et al., 2016, Pang et al., 2016, Xiong et al., 2017].

$$\mathcal{L}_{max-margin} = max(0, 1 - (s(q,d^+) - s(q,d^-)) \tag{3.4}$$

where s(q, d) is the score returned by the neural ranking model.

3.3.2 Training strategies

Supervised learning is the most common strategy employed by the existing approaches where query-document pair labels are available. The labels are available either through relevance judgements from popular TREC ad-hoc tasks or through implicit feedback from user interactions with commercial search engines (click-through) data.

Weakly supervised learning As neural models require large amounts of data, and since annotated data is limited, in this learning strategy query-document labels are generated using an existing traditional IR model (BM25, QL) that gives pseudo labels. Most recently, Dehghani et al. [2017] proposed to train a neural ranking model with weak

supervision and show that it provides upto 35% improvement over the BM25 baseline that is used as the weak ranker or annotator.

Semi-supervised learning In this approach, a small set of high quality manually judged query-document pairs are used along with large sets of unlabeled or pseudo-labeled data. For neural models, fine-tuning weak supervision trained models using a small set of labeled data is shown in [Dehghani et al., 2018a]. Recently, Li et al. [2018a] proposed a neural model with a joint supervised and unsupervised loss objectives. The learned representation is optimized for the context of IR by minimizing a supervised loss that accounts for errors in query-document relevance matching, and a unsupervised loss that computes the document reconstruction loss (e.g. auto-encoders). They show that neural IR models jointly learnt from labeled and unlabeled data benefit from both the generic semantics in unlabeled data and target-specific features in labeled data.

3.4 Interpretability Approaches

In this section, we describe in detail two interpretability approaches–one that is *model-agnostic* (LIME) and the other a *model-introspective* (DeepSHAP) that we use in this thesis to understand the decisions made by various neural IR models (such as, exactly *why* a particular document is relevant to a given query for this model). The approach of how these interpretability methods are modified or applied to the context of IR is described in Chapter 5.

3.4.1 Local Interpretable Model-Agnostic Explanations

Local interpretable model-agnostic models (LIME) [Ribeiro et al., 2016, 2018] are used to explain individual predictions by approximating the local behavior of complex black-box models. The important characteristics underlying LIME is that the explanation model needs to be (1) *interpretable*, (2) *locally faithful* and (3) *model-agnostic* which Ribeiro et al. [2016] respectively define as (1) "provide qualitative understanding between the input variables and the response", (2) the explanation "must correspond to how the model behaves in the vicinity of the instance being predicted" and (3) "the explanation should be able to explain any model". The explanations are created using "interpretable" models (e.g. sparse linear models) trained on interpretable feature representations. The

training data is sampled from perturbing the instance to be explained weighted by its proximity to that instance and the corresponding predictions obtained from the black box model.

Approaches in this space tend to differ based on the type of black-box model (classification, regression), the interpretable feature space and the definition of locality. For example, the interpretable feature space for text based models is the space of all words in that instance represented using one-hot vectors even if the black-box model uses word embeddings. The explanation then is a visualization of the simple model depending on the context and the model itself.

LIME is an approach designed specifically to explain the output of a classifier. To illustrate their approach, consider a trained binary classifier $\mathcal{B}$ and an instance document to classify d. Assume that $\mathcal{B}$(d) is a probability distribution across the classes. The objective of LIME is to train a simple model $\mathcal{M}_d$ that minimizes $\mathcal{L}(\mathcal{B}, \mathcal{M}_d, \pi_d)$ which is a measure of how far $\mathcal{M}_d$ is in approximating $\mathcal{B}$ in the locality defined by π_d which is a proximity measure between the perturbed instance and the instance to be explained. The explanation $\xi(x)$ produced by LIME can be obtained using the following equation.

$$\xi(x) = \arg\min_{g \in G} \mathcal{L}(\mathcal{B}, \mathcal{M}_d, \pi_d) + \Omega(g) \tag{3.5}$$

G is the family of explanation models (e.g. all possible linear regression models). $\Omega(g)$ is the complexity measure of the explanation model g. For example, for linear models, $\Omega(g)$ could be the number of non-zero weights. The loss $\mathcal{L}$ to be reduced is the dissonance between the simple $\mathcal{M}_d$'s predictions and the labels produced by $\mathcal{B}$ for all $d' \in \pi_d$.

For the case of explaining text based models, $\mathcal{M}_d$ is a sparse linear regression model (*ridge regression*[1]) trained on a bag of words feature space. $d' \in \pi_d$ is generated by perturbing document d. In LIME, d' is created by removing random words from random positions in d. The locally weighted squared loss $\mathcal{L}$ of the sparse linear model is as defined in eq. 3.6, where π_d is an exponential kernel on some distance function *dist* (e.g. cosine distance for text) given in eq. 3.7.

$$\mathcal{L}(\mathcal{B}, \mathcal{M}_d, \pi_d) = \sum_{d,d' \in D} \pi_d(d')(\mathcal{B}(d) - \mathcal{M}_d(d'))^2 \tag{3.6}$$

[1] https://scikit-learn.org/stable/modules/generated/sklearn.linear_model.Ridge.html

$$\pi_d(d') = exp(\frac{-dist(d, d')}{\sigma^2}) \tag{3.7}$$

The coefficients of the features of the trained $\mathcal{M}_d$ indicates the importance of the words for that instance. Features with highest positive coefficients are assumed to be important for the model to classify the instance to the predicted class, where as those with highest negative coefficients are assumed to work for the different class. Limiting the number of words returned for the explanation is chosen based on feature-selection strategies, such as, LASSO (using the regularization parameter λ), or forward and backward selection of features.

3.4.2 DeepSHAP

DeepSHAP [Lundberg and Lee, 2017] is one of the popular *model-introspective* approaches that enables us to understand the predictions of complex deep-learning models. It produces explanations by using the input feature attributions for a given instance and the prediction of the model for that instance. DeepSHAP is a modification of the DeepLift [Shrikumar et al., 2017] algorithm to efficiently estimate the shapley values over the input feature space for a given instance. The shapley value is a term coined by Shapley [Shapley, 1953] in cooperative game theory to refer to the contribution of a feature in a prediction. More specifically, shapley values explain the contribution of an input feature towards the *difference* in the prediction made vs the average prediction value.

DeepLIFT explains the difference in output or prediction from some 'reference' output with respect to the difference of the input (to explain) from a 'reference' input. DeepLIFT uses a '**summation-to-delta**' property that states that:

$$\sum_{i=1}^{n} C_{\Delta x_i \Delta t} = \Delta t \tag{3.8}$$

where $t = f(x)$ is model output, $\Delta t = f(x) - f(r)$, $\Delta x_i = x_i - r_i$ and r is the reference input. It can be described as the amount difference-from-reference of the output that is attributed to the amount of difference-from-reference of the input. The authors define a function (eq. 3.9) analogous to partial derivatives to compute the feature importance scores and use the chain rule to backpropagate the activation differences from the output

layer to the original input.

$$m_{\Delta x \Delta t} = \frac{C_{\Delta x \Delta t}}{\Delta x} \tag{3.9}$$

The choice of reference input depends on domain specific knowledge; For example, in digit classification task on the MNIST dataset, they use a reference input of all-zeros as that is the background of the images. For object detection in images, a plain black image is often used.

Chapter 4

Reproducibility of Neural IR models

In this chapter, we will discuss in detail the architectures of various deep learning IR models that we selected to reproduce along with the experimental setup, implementations and the discussion of the results we obtained. We initially implemented two *interaction-based* models–DRMM [Guo et al., 2016] and MatchPyramid [Pang et al., 2016], as it was shown that *interaction-based* approaches clearly outperform *representation-based* approaches in terms of retrieval effectiveness in a recent empirical study [Nie et al., 2018a]. We selected DRMM as it is shown to outperform strong IR baselines and other recent deep learning based approaches [Onal et al., 2018].

As DRMM doesn't consider the contexts in which terms occur, we considered *position-aware* approaches such as MatchPyramid that mimics image recognition for text matching and a *position-aware* extension of DRMM using PACRR like *n-gram* matching features to give PACRR-DRMM [McDonald et al., 2018]. We then implement a model NPRF-DRMM that uses the recently proposed generic neural pseudo relevance feedback (NPRF) framework [Li et al., 2018b] that enables the use of PRF along with state-of-the-art neural IR models by embedding them as building blocks. The proposed framework can use existing neural IR models by using them as scorers in evaluating the relevance of a document relative to the top-ranked documents and to the query without modifying their architectures.

#Vocabulary	#Document	Avg Doc. Len.	#Query
0.6M	528K	477	250

Table 4.1: Statistics of the TREC Robust04 collection

All the reproducibility experiments are carried out on the TREC Robust04 benchmark dataset for ad-hoc retrieval. The dataset contains 528,155 documents (TREC disks 4 & 5 minus Congressional Records) and 250 queries that is used in the TREC 2004 Robust track [Voorhees, 2005]. Manual high-quality relevance judgements are available for all queries, where both relevant and non-relevant documents are marked. Statistics of the collection are included in Table 4.1.

4.1 Deep Relevance Matching Model (DRMM)

The deep relevance matching model (DRMM) for ad-hoc retrieval is an interaction-focused model that uses a joint deep architecture at the query term level for relevance matching [Guo et al., 2016]. First, they build local interactions between each pair of terms from a query and a document based on cosine similarity of term embeddings. Then for each query term, they map the variable length local interactions into a fixed-length matching histogram. Based on this matching histogram, they employ a feed forward matching network that learns hierarchical matching patterns and produces a matching score. Finally, the overall matching score is computed by aggregating the individual query term matching scores using a term gating network. The model architecture is depicted in Figure 4.1.

Formally, both query and documents are represented using a set of term vectors denoted by $q = \{w_1^{(q)}, ..., w_M^{(q)}\}$ and $d = \{w_1^{(d)}, ..., w_N^{(d)}\}$, and s denotes the final relevance score, then we have

$$
\begin{aligned}
z_i^{(0)} &= h(w_i^{(q)} \otimes d), && i = 1, ..., M \\
z_i^{(l)} &= \tanh(W^{(l)} z_i^{(l-1)} + b^{(l)}), && i = 1, ..., M,\ l = 1, ..., L \\
s &= \sum_{i=1}^{M} g_i z_i^{(L)}
\end{aligned}
$$

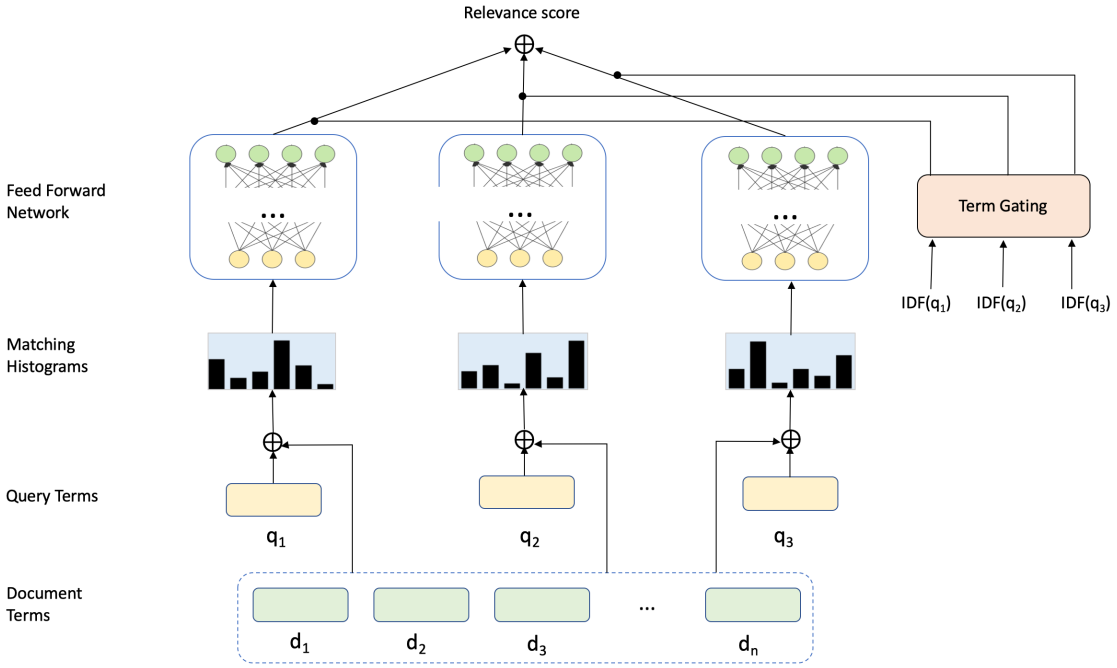

Figure 4.1: Deep Relevance Matching Model (DRMM) architecture

where $\otimes$ denotes the interaction operator (cosine similarity) between a query term and the document terms, h is the mapping function to get matching histograms, $z_i^{(l)}$ denotes the intermediate hidden layers for the i-th query term, and g_i denotes the aggregation weight from the term gating network.

Matching Histogram Mapping. Since the local interactions is within the interval [-1, 1], the histogram is obtained by discretizing the interval into a set of ordered bins and accumulating the count of local interactions for each bin. In this work, fixed size bins are used and the exact matching signal is treated as a separate bin. They explore three different ways of mapping:

Count-based Histogram (CH): Takes the count of interactions in each bin as the histogram value.

Normalized Histogram (NH): Normalize the count value in each bin by the total count, to focus on relative rather than absolute number of different levels of interaction.

LogCount-based Histogram (LCH): Takes the logarithm over the count value in each bin, to reduce the range.

Term Gating Network. The model explicitly models query term importance using a term gating network to get a aggregation weight for each term. The softmax function

is used as the gating function.

$$g_i = \frac{\exp(w_g x_i^{(q)})}{\sum_{j=1}^{M} \exp(w_g x_j^{(q)})}, \qquad i = 1, ..., M$$

where w_g is the weight vector of the gating network. Two different inputs were considered: *term vector*, where query term vectors are used; *inverse document frequency (IDF)*, where the IDF of the query term is used.

Model Training. The model is trained on triples (q, d^+, d^-), where document d^+ is ranked higher than document d^- with respect to q, minimizing pairwise max-margin loss as in eq. 4.1.

$$\mathcal{L}(q, d^+, d^-; \Theta) = \max(0, 1 - s(q, d^+) + s(q, d^-)) \tag{4.1}$$

where $s(q, d)$ denotes the predicted matching score and Θ are the parameters of the feed forward network and term gating network. The stochastic gradient descent method Adam with mini-batches (20) is used to trained the model.

4.1.1 Experimental Setup

We use the *title* of each TREC topic as queries in our experiments. The indexing and retrieval is implemented using Lucene[1]. During indexing and retrieval, both document and query words are white-spaced tokenized, lower-cased and stemmed using Krovetz stemmer available here[2]. Stopword removal is performed on the query words during retrieval from the Lucene index based on a custom stopwords list[3]. We compare the DRMM model retrieval performance against the BM25 baseline using the default parameters ($k1$:1.2 and b:0.75) from Lucene.

Term Embeddings. The 300 dimensional vectors used as term embedding input for DRMM are trained using CBOW model [Mikolov et al., 2013] on Robust04 collection. The parameters used are context window size (10), negative samples (10) and sub-sampling of frequent words with a threshold of 10^{-4} using `word2vec`[4]. The corpus is

[1] https://lucene.apache.org/
[2] https://github.com/rmit-ir/KrovetzStemmer
[3] See Lucene EnglishAnalyzer for details.
[4] https://radimrehurek.com/gensim/models/word2vec.html

Model Name	MAP	nDCG@20	P@20
BM25	0.2405	0.4038	0.347
$\text{DRMM}_{LCH\text{x}IDF}$	0.257	0.4103	0.352
$\text{DRMM}_{LCH\text{x}IDF}$ *paper*	0.268	0.423	0.381

Table 4.2: DRMM Model retrieval performance on the Robust04 collection

preprocessed removing HTML tags, stemming using Krovetz stemmer and stopword removal using NLTK[5]. We discarded from the vocabulary terms that occur less than 10 times in the collection resulting in a vocabulary of size 106K. The out-of-vocabulary (OOV) terms are ignored while preparing the model input.

Model Implementation. We implemented the DRMM model using the Keras functional api[6] with a Tensorflow backend. The network configuration of the DRMM model is set as: one histogram layer (30 nodes), two hidden layers in the feed forward matching network (5 and 1 nodes) and 1 output layer with the term gating network for the final relevance score as suggested in the paper based on hyperparameter tuning on the validation set. We implement the variant $\text{DRMM}_{LCH\text{x}IDF}$ that uses *logcount-based* histogram as input and *term vector* for computing query term importance in the term gating network. We use a step-decay adaptive learning rate that drops the learning rate by a factor of 0.9 every 10 iterations (initial learning rate=0.001). We train the model for 100 iterations, where in each iteration the model trains on 1000 mini-batches (20). Thus, the model would train on roughly 2M training pairs of data. The training pairs for each mini-batch is randomly sampled from all possible pairs of relevant and irrelevant documents from the relevance judgements for the queries in the training set.

Evaluation. We conduct 5 fold cross-validation to minimize overfitting due to the limited number of queries in the collection. Topics are randomly split into 5 folds and the parameters are tuned on 4-of-5 folds. The retrieval performance is evaluated on the final fold in each case using the optimal parameters. This process is repeated 5 times, once for each fold. Mean Average Precision (MAP) is the metric that is optimized for during training. We use a re-ranking strategy for evaluation, where first we retrieve the top-2000 documents using BM25 model and then re-rank it using the neural model to obtain the top-1000 documents that are used to compute the evaluation metrics. Each displayed metric is the average of the five 5-fold evaluation values.

[5] https://www.nltk.org/
[6] https://keras.io/getting-started/functional-api-guide/

4.1.2 Results and Discussion

We observe from Table 4.2 that our implementation of DRMM provides improvements over the BM25 baseline across all evaluation metrics. However, this implementation gives results lower than that from the paper because of the following reasons: due to different random partitions of the data, and in the original paper they re-rank the top documents returned from a well-tuned query-likelihood (QL) model. Also, they use a different sampling strategy to generate the training pair instances since there is a data imbalance problem, where the pairs available for each query are significantly different (i.e. some queries have more positive samples than others). Then, the model could be dominated by a few queries leading to lower performance.

We also empirically observe that using word embeddings not trained on the collection, such as, `word2vec`[7] or `glove`[8] embeddings results in lower performance over the evaluation metrics. This is because the coverage of these term embeddings are only 35% and 50% of the 0.1M vocabulary using `word2vec` and `glove` respectively. The filtering of the vocabulary size from 0.6M to 0.1M by removing words that occur less than 10 times in the collection is a very important preprocessing step in effectively training a DRMM model that is better than the baseline. This is because there isn't sufficient data to learn reliable term embeddings for the words that occur less than 10 times in the collection.

4.2 MatchPyramid for ad-hoc retrieval

The MatchPyramid model for ad-hoc retrieval [Pang et al., 2016] is an interaction-focused model that first builds a matching matrix from the local interactions between the terms from query and document using word embeddings. This matching matrix is then viewed as an image and fed into a 2D convolutional neural network (CNN) to extract hierarchical matching patterns. Finally the matching patterns from the CNN are fed into a multi-layer perceptron (MLP) to aggregate a relevance score. The model is known to capture different matching patterns, such as, n-grams and un-ordered n-terms. The model architecture is shown in Figure 4.2.

[7] `https://code.google.com/archive/p/word2vec/`
[8] `https://nlp.stanford.edu/projects/glove/`

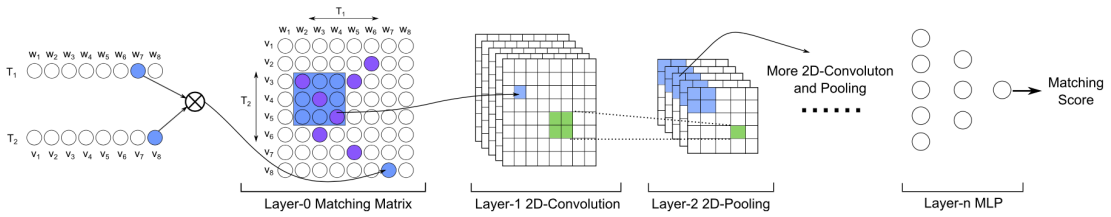

Figure 4.2: MatchPyramid architecture [Pang et al., 2016]

Matching Matrix. It is a two-dimensional matrix where each element $\mathbf{M}_{ij}$ denotes the similarity between the i-th word w_i in the query and the j-th word v_j in the document. The order in which the words appear is preserved while creating the matrix. They define four different similarity functions based on the words w_i , v_j or their corresponding embeddings: *indicator function* (MP-Ind)–checks if two words are identical; *cosine similarity* (MP-Cos)–between the word embeddings; *dot product* (MP-Dot); *gaussian kernel* (MP-Gau)–as defined in eq. 4.2.

$$\mathbf{M}_{ij} = \ e^{-||\vec{w_i} - \vec{v_j}||^2} \tag{4.2}$$

Hierarchical Convolution. This consists of convolutional and dynamic pooling layers which are commonly used in CNN. The kernel sizes in each convolutional layer determines the maximum size of n-gram features that we take into account. And the pooling layer determines the most important information that needs to be selected from the convolutional layer. This is useful especially when we consider documents that have hundreds of words of which most are background words.

Training Objective. They use a pairwise ranking loss such as hinge loss with a margin of 1.0 for training the MatchPyramid model similar to the objective defined in eq. 4.1 for the DRMM model.

4.2.1 Experimental Setup

The indexing and retrieval experimental setup for the TREC Robust04 collection is the same as described in section 4.1.1. The term embeddings used for all the models are the *glove* embeddings of 50 dimensions which are obtained by training on the Wikipedia

Model Name	MAP	nDCG@20	P@20
BM25	0.2405	0.4038	0.347
MP-Ind	0.1823	0.3353	0.286
MP-Ind *paper*	0.169	0.319	0.281
MP-Cos	0.1898	0.3328	0.276
MP-Cos *paper*	0.189	0.330	0.290

Table 4.3: MatchPyramid models retrieval performance on the Robust04 collection

corpus. We follow the exact same evaluation method as described for the DRMM model (section 4.1.1) here.

Model Implementation. We implement the model variants MP-Ind and MP-Cos using Keras. The optimization method `Adam` with mini-batches (32) is used for model training. The initial learning rate is set to 10^{-4} and uses a step-decay learning rate schedule that drops the learning rate by a factor of 0.9 after every 10 epochs. The network configurations for the MatchPyramid models are selected based on MAP on the validation set. We use one convolutional layer with kernel size 3 x 3 and feature maps set to 8, one dynamic pooling layer with pooling size 2 x 10, and two fully connected layers (128 nodes and 1 node respectively) with ReLU as the activation as suggested in [Pang et al., 2016]. We use one hierarchical convolution layer as more layers will lead to overfitting due to limited data. In our experiments, the maximum query length is set to 5 and the maximum document length is set to 500.

4.2.2 Results and Discussion

When we look at Table 4.3 we see that the MatchPyramid models cannot compete with the BM25 retrieval baseline across all measures. However, we can see that our implemented model performance is comparable to the model performance mentioned in the paper. As both MP-Ind and MP-Cos show similar performance, we can see that MP-Cos is able to encode exact matching along with semantic signals giving higher importance to exact matching over semantic similarities which is important for relevance matching. We empirically observe that it is better to use `glove` embeddings (50-dim) than the `word2vec` embeddings (300-dim) trained on the collection, this could be because the MatchPyramid models have a huge number of parameters and limited training data thus using an embedding with fewer (50) dimensions gives better performance. The best pooling size of 2 x 10 shows that it picks importance signals equivalent to the median

query length on the query side, and on the document side, it aggregates importance signals from every 50 words which is close to the average length of a paragraph.

4.3 Deep Relevance Ranking using Enhanced Document-Query Interactions (PACRR-DRMM)

In this paper [McDonald et al., 2018], the focus is on enriching DRMM with context-sensitive representations of the query, as in the original model query terms are scored relative to the document terms without taking into consideration the context in which the term occurs, its akin to a Bag-of-Words (BoW) model. Whereas, the PACRR model [Hui et al., 2017, Yates and Hui, 2017] is a *position-aware* interaction based model that uses similarity matrices that captures semantic similarity between each query term and each individual term in the document, also preserving term order. Thus, PACRR-DRMM is an extension of DRMM, that uses the query term encodings obtained from PACRR-like convolutional *n-gram* matching features as an input for the DRMM model.

PACRR. The query-document similarity matrix (*sim*) is created using the cosine similarity between the q-term and d-term embeddings. To keep the dimensions of the *sim* fixed across queries and documents of varying length, the queries are padded to max query length l_q and documents are either truncated/padded to max document length l_d[9]. Then the following pipeline is applied:

convolutional relevance matching. convolutional layers of different kernel sizes n x n (n=2, 3,...,l_g) are applied on the similarity matrix to capture positional information over text windows of different lengths (*n-grams*). For each kernel size, multiple kernel filters are used. The convolutional layer uses a stride (1, 1) along the query dimension and stride of (1, n) along the document dimension, so that it operates over consecutive terms that existed in the document.

max pooling layers. First, max pooling is applied across the dimension of the filters to preserve the strongest signal from different filters. Then, row-wise k-max pooling to capture the k strongest signals for each query term and all document terms.

[9]PACRR-*firstk* as highlighted in [Hui et al., 2017].

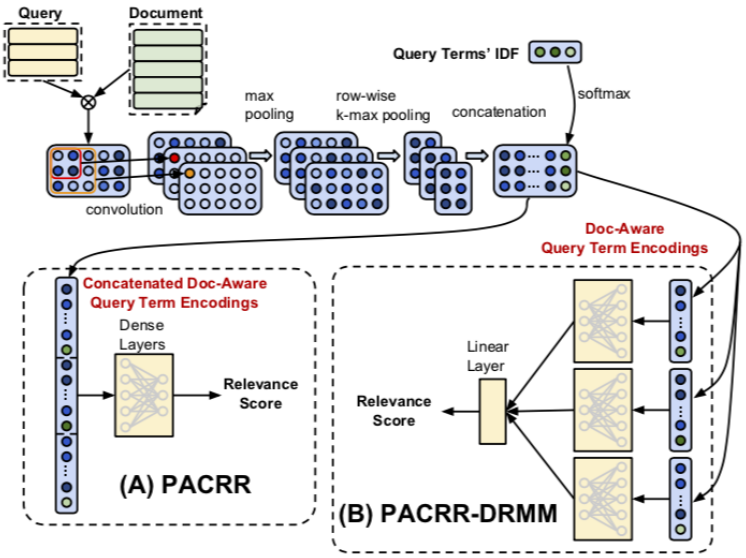

Figure 4.3: PACRR-DRMM architecture [McDonald et al., 2018]

MLP for global relevance. The resulting matrices are concatenated into a single matrix where each row is a document-aware query term encoding. To each row, the IDF of the query term is appended which is normalized by taking softmax with respect to the IDFs of all query terms. The rows are then concatenated to give a single vector that is used as input to a MLP [Hui et al., 2018] to give a relevance score.

PACRR-DRMM. In this version of the PACRR model, instead of using a MLP to score the *concatenation* of all document-aware q-term encodings, the MLP is used independently to score each q-term encoding similar to DRMM and then the resulting scores are aggregated using a linear layer. The architecture of the model along with its comparison to the PACRR model can be seen in Figure 4.3. The scores of the q-terms from MLP is not weighted using a *gating* mechanism like in DRMM, however since the IDF of q-term is appended to the encoding are a form of term-gating.

Training objective. The model (fig. 4.3) is trained by minimizing the cross entropy loss as defined in eq. 4.3 as it has been demonstrated to give better results [Dehghani et al., 2017].

$$\mathcal{L}(q, d^+, d^-; \Theta) = -log\frac{exp(rel(q, d^+))}{exp(rel(q, d^+)) + exp(rel(q, d^-))} \tag{4.3}$$

4.3.1 Experimental Setup

The indexing and retrieval experimental setup for the TREC Robust04 collection follows as described in section 4.1.1. The same evaluation method as described for the DRMM model (4.1.1) is implemented here.

Input Preprocessing. The document and query words are white-spaced tokenized, lower-cased and stemmed using Krovetz stemmer. But no stopword removal is applied on the texts. Thus, the vocabulary size is 0.3M which is bigger than that considered for both DRMM and MatchPyramid (0.1M).

Term Embeddings. This model uses 200-dimension `word2vec` embeddings trained on the collection with negative sampling and window size set to 5 and the rest of the hyperparameters set to default[10]. The word embeddings are not updated while training the PACRR-DRMM model. The OOV term is represented with an embedding that is the average of all the term embeddings in the vocabulary.

Model Implementation. The model is implemented in Keras with TensorFlow backend. It is trained using the `Adam` optimization method with learning rate of 0.001 and mini-batches (32) that contains randomly sampled irrelevant document for each relevant document found in the top-1000 BM25 retrieved results for each query. The training usually converges within 50 epochs, with the weights uniformly initialized. The irrelevant documents are also sampled from the top-1000 BM25 documents which are not marked as relevant. We follow the network configuration as specified in Appendix A of the paper. The maximum query length l_q is set to 5 and maximum document length l_d to 1000. The maximum kernel size (l_g x l_g) for the convolutional layers is set to (3 x 3) with number of filters equal to 16. For row-wise k-max pooling, k=2. A two layer MLP with ReLU activations and hidden layers with 7 nodes to score each document-aware q-term encoding. Finally, the scores of each document-aware q-term encoding are aggregated using a linear layer (1 node MLP).

4.3.2 Results and Discussion

From Table 4.4, we see that both PACRR and PACRR-DRMM shows improvements over the BM25 baseline across all evaluation metrics. We also see that the model trained

[10] https://radimrehurek.com/gensim/models/word2vec.html

Model Name	MAP	nDCG@20	P@20
BM25	0.2405	0.4038	0.347
PACRR	0.260	0.442	0.372
PACRR-DRMM	0.263	0.445	0.374
PACRR-DRMM *paper*	0.259	0.444	0.373

Table 4.4: PACRR-DRMM model retrieval performance on the Robust04 collection

using the architecture configuration mentioned above, gives a retrieval performance that is quite close to the model trained in the paper across all the metrics. This is because we used the same partitions[11] of the topics shared by the authors that was used for cross-validation along with the pre-processed documents ("HTML" tags removed) for each fold, the top-k documents retrieved from BM25 using Galago[12] and the pre-trained word embeddings. We observe that PACRR-DRMM performs slightly better than PACRR, this is likely due to the fewer parameters of the MLP layer which is shared between all q-term encodings and as it uses a shorter input representation.

4.4 Neural Pseudo-Relevance Feedback (NPRF-DRMM)

In this work [Li et al., 2018b], they proposed a generic neural pseudo relevance feedback framework (NPRF) that helps the application of PRF with existing neural IR models (NIRM). For a given query and target document, NPRF produces a final relevance score by considering the interactions of the target document with the query as well as the top-n feedback documents from the initial ranking (e.g. BM25, QL). The proposed framework directly incorporates existing neural IR models without the need to modify their architectures as shown in Figure 4.4.

Extracting document interactions. Given the target document d and each feedback document $d_q \in D_q$, $rel_d(d_q, d)$ is a NIRM that is used to compute the n relevance scores. Since DRMM takes as input matching histograms (LCHxIDF) from the cosine similarity between pairs of term embeddings from d and d_q without considering term dependencies, so d_q is summarized by keeping the top-k terms based on *tf-idf* scores to remove noisy or irrelevant terms.

[11] https://archive.org/download/deep_relevance_ranking_data/robust04_data.tar.gz
[12] http://www.lemurproject.org/galago.php

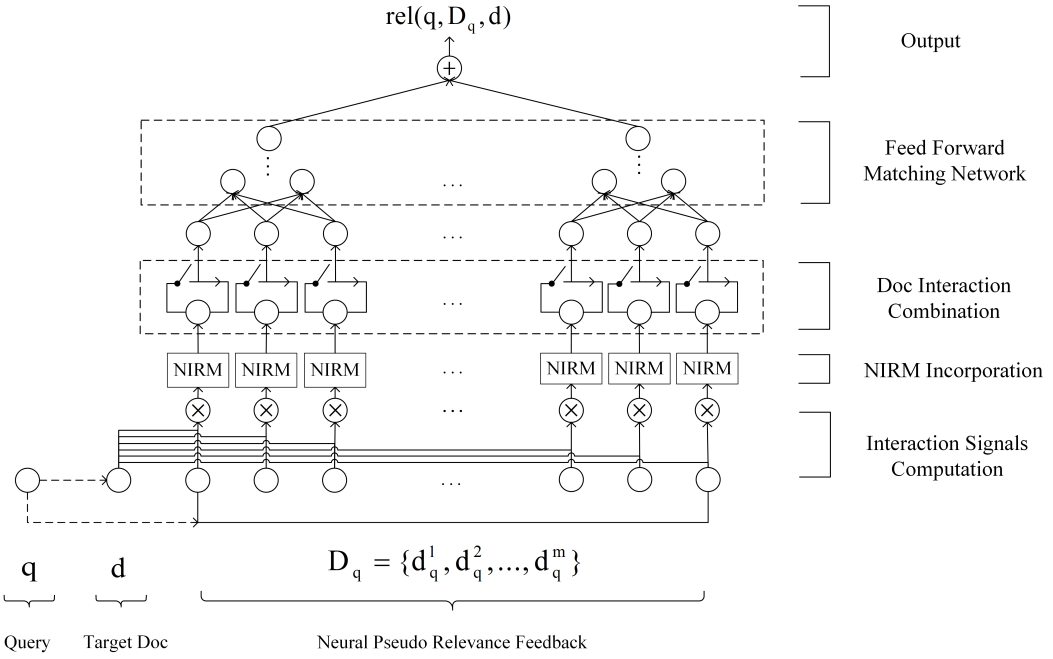

Figure 4.4: Neuro Pseudo Relevance Framework (NPRF) architecture [Li et al., 2018b]

Combining document interactions. When combing the relevance scores for the feedback documents, the relevance of d from the initial ranking $rel_q(q, d_q)$ is also important. Thus, weighted relevance document score $rel'_d(d_q, d)$ is computed using the equation 4.4 with min-max normalized $rel_q(q, d_q)$ score.

$$rel'_d(d_q, d) = rel_d(d_q, d)(0.5 + 0.5 * rel_q(q, d_q)) \tag{4.4}$$

The $rel'_d(d_q, d)$ of each $d_q \in D_q$ is combined into a single relevance score using two variants: (i) direct summation, and (ii) using feed forward network with *tanh* activations.

Training Objective. A pairwise ranking loss such as hinge loss with a margin of 1.0 is used for training the model similar to the objective defined in eq. 4.1.

4.4.1 Experimental Setup

The indexing and retrieval experimental setup for the TREC Robust04 collection is the same as described in section 4.1.1. The term embeddings used are the `word2vec` embeddings of 300 dimensions trained on the Robust04 collection using the setup described in section 4.1.1. Akin to the evaluation setup for DRMM described in section 4.1.1, the proposed NPRF_{ds}-DRMM model is used to re-rank the top-2000 documents retrieved

Model Name	MAP	nDCG@20	P@20
BM25	0.2405	0.4038	0.347
NPRF_{ds}-DRMM	0.2869	0.4585	0.4006
NPRF_{ds}-DRMM *paper*	0.2904	0.4502	0.4064

Table 4.5: NPRF-DRMM model retrieval performance on the Robust04 collection

from BM25 and uses the same 5-fold cross validation setup to measure the model's retrieval performance.

Model Implementation. The model implemented in Keras is trained using the `Adam` optimizer with a batch size of 20 and an initial learning rate of 10^{-3} followed by step-decay learning rate schedule that drops the rate by a factor of 0.9 after every 25 iterations. Usually, the training converges within 30 epochs or maximum 250 iterations (on each iteration the model trains on 100 mini-batches). The variant that we implemented is NPRF_{ds}-DRMM that is based on the direct summation of the weighted feedback document relevance scores as that was shown to have the best performance across all the variants [Li et al., 2018b] on the TREC Robust04 collection. The top-10 documents from BM25 are used as the pseudo relevant feedback document input D_q for the model, where each $d_q \in D_q$ is summarized by the top-20 terms based on *tf-idf*. The network configuration for the DRMM component is using the $\text{DRMM}_{LCH\text{x}IDF}$ variant with the original configuration ([Guo et al., 2016]) that includes a histogram input layer of 30 nodes, two hidden layers in MLP (5 and 1 node respectively), and one output node with term gating (*IDF*-softmax values) for the feedback document relevance score. A different sampling strategy is used to generate the training pair instances that takes into consideration the data imbalance problem thus generating about the same number of pairs across all the queries even if some queries have more positive samples than others[13].

4.4.2 Results and Discussion

From Table 4.5, we see that NPRF_{ds}-DRMM shows significant improvements over the BM25 baseline across all evaluation metrics. We also see that the model trained using the architecture configuration mentioned above, gives a retrieval performance that is quite close to the model trained in the paper across all the metrics. This is because we

[13] https://github.com/ucasir/NPRF/blob/master/utils/pair_generator.py#L100

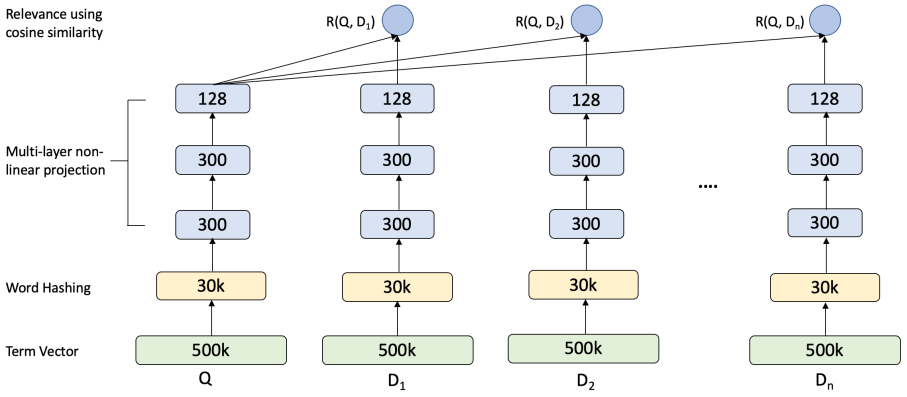

Figure 4.5: Deep Structured Semantic Model (DSSM) architecture

used the same partitions of the topics shared by the authors as part of the code[14]. The small differences in the metrics is because the preprocessing that we used is different from that mentioned in the paper where they used a porter stemmer instead of krovetz stemmer and also the word embeddings are trained differently by using the pool of top-2000 documents returned from BM25 for each individual queries as suggested by Diaz et al. [2016] instead of the entire Robust04 collection as we used in section 4.1.1.

4.5 Deep Structured Semantic Models (DSSM)

The Deep Structured Semantic Model (DSSM) [Huang et al., 2013] is based on Siamese networks used for short text matching. The model is trained on pairs of query and documents titles, both texts represented as a bags-of-character-trigraphs. This architecture consists of two deep models–for the query as well as the document–with fully-connected layers and cosine distance as the similarity function. They train the model on click-through data where each sample consists of a query q, positive documents d^+ (document clicked by user on SERP page) and a set of negative documents D^- randomly sampled uniformly from the collection. The model is trained by minimizing the cross-entropy loss after taking a softmax over the model outputs for all the candidate documents,

$$\mathcal{L}_{DSSM}(q, d^+, D^-) = \log\left(\frac{e^{\gamma \cdot \cos(\vec{q}, \vec{d^+})}}{\sum_{d \in D} e^{\gamma \cdot \cos(\vec{q}, \vec{d})}}\right) \\ where, D = d^+ \cup D^- \tag{4.5}$$

[14] https://github.com/ucasir/NPRF/blob/master/model/nprf_drmm_config.py

4.5.1 Experimental Setup

The indexing and retrieval experimental setup for the TREC Robust04 collection is the same as described in Section 4.1.1. Since DSSM needs large scale training data (large click-through dataset), due to its huge parameter size, we try to train the model using two strategies: pretraining the triletter embeddings using the Robust04 collection instead of training it during model training, and using weak supervision to train the model with AOL query logs as suggested by Dehghani et al. [2017]. The models implemented in Keras are trained using the `Adam` optimizer with varying batch sizes and learning rates. The parameters of the DNN is shown in Figure 4.5, the only difference is that the number of nodes in the word hashing layer depends on the triletters obtained from the Robust04 collection.

4.5.1.1 Pretraining Triletter Embeddings

The input for the DSSM model is the bag-of-character triletter vector which is of much lower dimensionality (∼15k) than term vectors which are usually the size of the vocabulary (∼500k), thus enabling us to effectively train a deep neural network (DNN). Due to the limited number of queries and smaller size of the training dataset from Robust04 collection in comparison to the Bing Web collection that is used in the paper, we first train the triletter embeddings from the collection so that the limited training data can then be used to effectively learn dense representations for both queries and documents.

The 300 dimensional triletter embeddings are trained using the skip-gram model with negative sampling (SGNS) [Mikolov et al., 2013]. The corpus is preprocessed removing HTML tags, but no stemming and stopword removal is applied. Each word (e.g. *good*) in the collection is first represented with starting and ending marks (e.g. *#good#*) after which they are split into triletter *n-grams* (e.g. *#go, goo, ood, od#*). The corpus then comprises of documents represented using the triletters instead of words which are then used to train the embeddings. The parameters used are context window size (5), negative samples (10), minimum triletter frequency (5) and subsampling of frequent words with a threshold of 10^{-4} using `word2vec`[15] (`sg=1` and `hs=0`). The model is trained for 20 epochs over the corpus. The parameters for the model were chosen based on Bojanowski

[15] https://radimrehurek.com/gensim/models/word2vec.html

et al. [2017], where they describe how to compute word embeddings by training *n-gram* embeddings and then representing each word by the sum of these representations.

4.5.1.2 Weak supervision with AOL query logs

Training query set. We prepare the training query set using the setup described in Dehghani et al. [2017], which comprises of unique queries that appear in the AOL query logs. We filter out navigational queries containing URL substrings ("http","www.", ".com", ".net",".org", ".edu"). All non-alphanumeric characters from the queries were removed. All queries that are present in the evaluation Robust04 queries are removed from the training set. Finally, only those queries that have at-least 10 documents retrieved by BM25 from Robust04 are kept, thus giving us a set of 6.15 million queries are after preprocessing. We also created smaller subsets of queries (1k, 10k) to see if we can effectively train a small model and 1M as that should already give us sufficient data to train the model.

Training data. In Dehghani et al. [2017], they retrieve the top-1000 documents using BM25 for each query in the training query set and generate $|Q|$x1000^2 ($\sim$6E13) pairwise training pairs. Due to limited computational resources and time, we used different sampling strategies to generate the pairs:

- For each positive document that we consider from the top-10, we sample 4 negative documents from the rest of the top-1000 documents.
- For each positive document from the top-10, we sample 3 negative documents randomly from the rest of the collection and 1 negative document from the rest of the top-1000 based on probability distribution over the BM25 scores.
- For each positive document from top-10, we sample 4 negative documents from the rest of the collection.
- For a positive document from top-1, we randomly sample 4 negative documents from the rest of the collection.

Document text. In the DSSM paper, they train the model on query-document title pairs where title is extracted from the title field from the index. Thus, we also tried

training the model on different document text versions since the Robust04 collection doesn't have a specific title field in many of the documents (∼65%)– full-text of document, the first 3 sentences of the document as the title, the first 2 sentences as the title and the first 3 sentences plus the snippet as title for the positive documents.

4.5.2 Results and Discussion

Using the various strategies described above to train the DSSM model, we couldn't effectively train a model that performs better than BM25 baseline[16]. The poor effectiveness of the DSSM model (*representation-focused*) is also highlighted in previous studies [Guo et al., 2016, Pang et al., 2016]. One possible reason for the poor effectiveness is that it is difficult to learn a good global representation of the documents in the Robust04 collection to match with the representation learnt for a related query.

However, these results are inconsistent with that of Dehghani et al. [2017] where they show a model based on representation learning and weakly supervised with BM25, gives a performance better than BM25. Our results with weak supervision reflects the same performance as in this study [Nie et al., 2018a], where they hypothesize that the difference in performance could be due to the huge difference in computation resources where they train a very large number of training pairs (∼6E13) for a large number of epochs. One other reason could be that the size of the dense input representation has a much smaller dimensionality (300-dimension normalized weighted element-wise summation of the term embeddings) whereas for DSSM it is about 15k.

4.6 Performance Comparison of Trained Models

In this section, we briefly compare the retrieval performance of the implemented neural retrieval models as seen in Table 4.6. We can see that all of the implemented neural retrieval models perform better than the traditional BM25 baseline as highlighted in previous work, except for MatchPyramid which is similar to the conclusion they get in the original paper [Pang et al., 2016].

[16] https://www.comet.ml/neural-ir/dssm-weak-supervision

Models	Model Name	MAP	nDCG@20	P@20
Traditional IR	BM25	0.2405	0.4038	0.347
DRMM	$\text{DRMM}_{LCH\text{x}IDF}$	0.257	0.4103	0.352
MatchPyramid	MP-Ind	0.1823	0.3353	0.286
	MP-Cos	0.1898	0.3328	0.276
PACRR-DRMM	PACRR	0.260	0.442	0.372
	PACRR-DRMM	0.263	0.445	0.374
NPRF	NPRF_{ds}-DRMM	0.2869	0.4585	0.4006

Table 4.6: Comparison of retrieval performance across different retrieval models on Robust04 collection

We can clearly observe that *position-aware* neural IR models, like, PACRR and PACRR-DRMM that takes into consideration the context of the query terms perform better than DRMM which computes the interactions between the query and document terms ignoring the context. Also, we see that the NPRF model (NPRF_{ds}-DRMM) which incorporates pseudo relevance feedback into the existing DRMM model, improves on the DRMM retrieval performance across all evaluation metrics. We also provide a comparison of the hyper-parameters between the implemented neural retrieval models in Table 4.7.

Hyper-parameters	DRMM	MatchPyramid	PACRR-DRMM	NPRF-DRMM
Initial learning rate (lr)	10^{-3}	10^{-4}	10^{-3}	10^{-3}
Step-decay (lr)	✓	✓	✗	✓
Drop factor (lr)	0.9 (10 iter.)	0.9 (10 iter.)	—	0.9 (25 iter.)
Mini-batch size	20	32	32	20
Optimizer	Adam	Adam	Adam	Adam
Training iterations	100	200	50	250
Nr. mini-batches per iter.	1000	1000	—	100

Table 4.7: Comparison of hyper-parameter values across the different retrieval models

4.7 Deployment of Trained Models

We have deployed some of the trained models (DRMM, MP-COS, PACRR-DRMM, NPRF_{ds}-DRMM) using the Flask[17] web framework. The Flask server first loads the pretrained Keras model that shows the best performance on MAP for Robust04 and exposes two REST API endpoints–for scoring a given query-document pair and for preprocessing the given text into the input format required by the pretrained model. The work flow of how the models are deployed is displayed in Figure. 4.6.

[17] http://flask.pocoo.org/

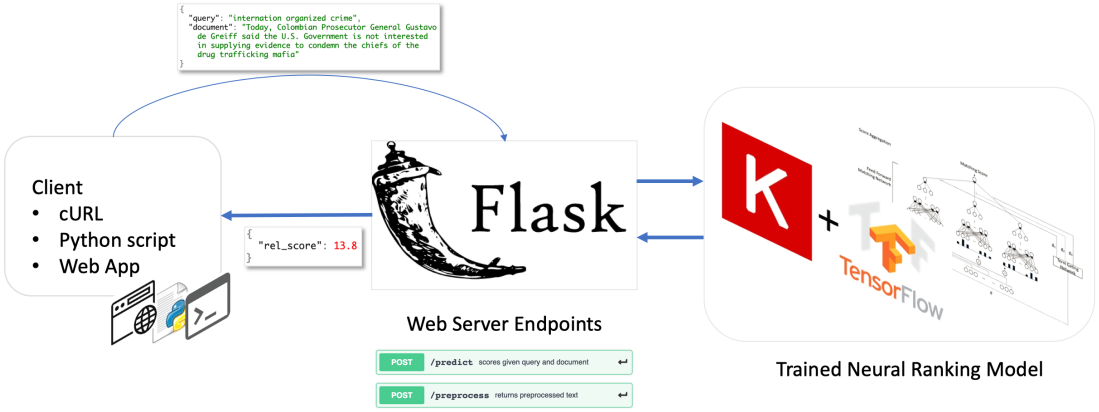

Figure 4.6: Deployment of NRMs workflow

The endpoint for scoring can be used by any system that has a *telescoping* setup that has to re-rank the initial set of retrieved documents (e.g. BM25 or QL) using one of the trained models. The second endpoint for preprocessing the given text is useful for interpretable visualizations such as heatmaps that highlights words in the document that are relevant/irrelevant to the query.

Chapter 5

Interpretability in IR

In this chapter, we will initially describe how the interpretability approaches detailed in chapter 3–LIME (section 3.4.1) and DeepSHAP (section 3.4.2) are applied for *ad-hoc* text retrieval and ranking in the context of complex neural ranking models.

In this work, we are interested in how DeepSHAP can be adapted to explain the output of neural retrieval models (NRM). In particular, what is a good "*reference*" input in the context of IR? In computer vision, a plain black image is a good reference input to understand the decisions of a image classifier. So we developed various reference input document construction methods as described in the section 5.2 for 3 different neural rankers (DRMM, MatchPyramid, PACRR-DRMM). We would also like to investigate if DeepSHAP's explanations are highly sensitive or robust to the different reference input distributions that were constructed. Also, do these reference input distributions perform differently depending on the neural ranking model. Additionally, we compare the explanations produced by DeepSHAP to that of LIME (model-agnostic) detailed in section 5.1. We ponder on the question, if both these approaches produce different or similar explanations, as they both give *local* explanations.

In order to understand the various questions raised above, we describe the experimental setup in section 5.3. The results of the various experiments are discussed in detail in section 5.4.

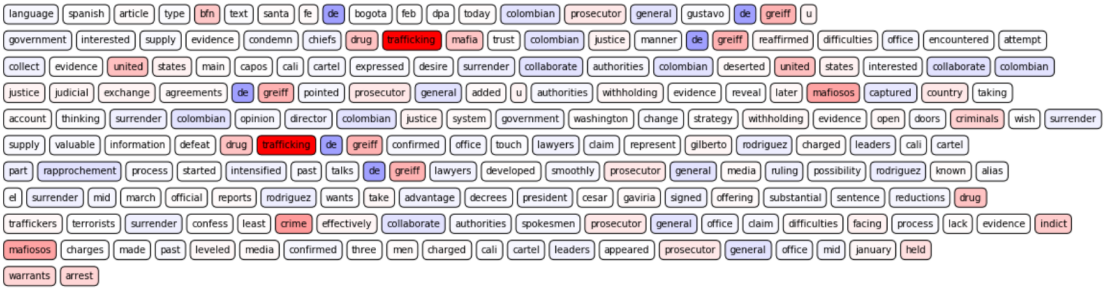

Figure 5.1: Heatmap visualization over the words in the doc 'FBIS3-10082' for the query 'international organized crime' using the output of LIME for the DRMM ranking model. *Words highlighted in 'red' indicate 'relevance' and those in 'blue' for the 'irrelevant' class.*

5.1 LIME for IR

The LIME [Ribeiro et al., 2016] approach is designed mainly to explain the output of classifiers, whereas we would like to understand *why* a particular document returned from a neural retrieval model is relevant to a query. Recently, there has been work [Singh and Anand, 2019] which adapted LIME to show how ranking can be cast as a classification problem where the neural retrieval model predicts a class distribution (*relevant* or *irrelevant*) for a given query-document pair. We discuss their *score-based* approach that we implemented here, consider a query q and the list of *top-k* documents D_q^k returned from the neural retrieval model $\mathcal{R}$ (point-wise ranker) obtained by scoring each document from a set of candidate documents retrieved from the index (BM25) and sorting. To explain a document d from D_q^k, LIME trains an explanation model $\mathcal{M}_d$ on the perturbed documents d'. To model it into a classification problem they consider a random variable $\mathcal{X}$ to indicate the possible classes - *relevant* and *irrelevant*. Below is the formula used to compute $P(\mathcal{X} = relevant|q, d', \mathcal{R})$ using the *score-based* approach,

$$1 - \frac{\mathcal{R}(q, d_1) - \mathcal{R}(q, d')}{\mathcal{R}(q, d_1)} \tag{5.1}$$

where $d_1 \in D_q^k$ is the top-ranked document in the list, if $\mathcal{R}(q, d') \geq \mathcal{R}(q, d_1)$ then $P(\mathcal{X} = relevant) = 1$. Note that $P(\mathcal{X} = irrelevant|q, d', \mathcal{R}) = 1 - P(\mathcal{X} = relevant|q, d', \mathcal{R})$.

Explanation Model In this approach the interpretable feature space is the set of words and since the explanation model $\mathcal{M}_d$ is a linear ridge regression model, the sign and magnitude of the coefficients of $\mathcal{M}_d$ indicate which words in d are strong indicators

of relevance and could be used as visual explanations such as heatmaps over the input words as shown in Figure 5.1.

5.2 DeepSHAP for IR

In the context of IR, DeepSHAP can be used to explain *why* a document is relevant to query (according to a given neural retrieval model) by computing the shapley values for words in the document. The words with high shapley values indicate that they are important towards this prediction of relevance. However, to accurately compute the shapley values using DeepSHAP–a reference input is needed. What makes a good *background* image in the context of IR?

Unlike classification tasks, in ranking we have at least 2 inputs which are in most cases the query and document tokens. In this work, we fix the reference input for the query to be same as that of the query-document instance to be explained and experiment with various reference inputs for the document. The intuition behind doing so is to gain an average reference output in the locality of the query.

The various document reference inputs that we considered in our experiments are:

OOV The reference document consists of 'OOV' tokens. For, DRMM and MatchPyramid models the embedding vector for 'OOV' comprises of all-zeros which is similar to the background image used for MNIST. But for PACRR-DRMM the 'OOV' embedding vector is the average of all the embedding vectors in the vocabulary.

IDF *lowest* The reference document is constructed by sampling words with low IDF scores. These words are generally stop-words or words that are similar to stop-words so they should, in general, be irrelevant to the query.

QL *lowest* The reference document comprises of sampled words with low *query-likelihood* scores that are derived from a language model of the *top-1000* documents. As this would have words that are irrelevant to the query, SHAP should be able to pick more relevant terms from the document to be explained.

COLLECTION *rand doc* The reference document is randomly sampled from the rest of the collection minus the *top-1000* documents retrieved for the query.

TOPK LIST *rand doc from bottom* The reference document is randomly sampled from the bottom of the *top-1000* documents retrieved.

These variants were designed based on the intuition that the reference input document would comprise of words that are irrelevant to the query and thus DeepSHAP should be able to pick the most important terms from the input document that explain relevance to the query.

5.3 Experimental Setup

In our experiments, we aim to answer the following research questions:

- **RQ1** Are DeepSHAP explanations sensitive to the type of reference input in the case of NRMs?
- **RQ2** Can we determine which reference input produces the most accurate local explanation?
- **RQ3** Does modeling the reference distribution for query input also affect the DeepSHAP explanations?

To this end, we describe the experimental setup we used to address these questions. The 3 neural ranking models that we considered are–DRMM, MatchPyramid (MP-COS), and PACRR-DRMM. All the models were trained using the setup as described in chapter 4. Once trained all the models were deployed using the Flask web framework as detailed in section 4.7.

To conduct the experiments, we used the Robust04 test collection from TREC. We used Lucene to index and retrieve documents. We chose to study explanations for the distinguished set of hard topics[1] (50) from the TREC Robust Track 2004 (model retrieval effectiveness highlighted in Tab 5.1). We generate the explanations from LIME and SHAP for the top-3 documents retrieved for each query and use only these for our quantitative experiments.

[1] https://trec.nist.gov/data/robust/04.guidelines.html

	MAP	MRR	P@10	P@20	NDCG@10	NDCG@20
BM25	0.0890	0.4786	0.2280	0.2030	0.2462	0.2289
DRMM	0.1202	0.5444	0.2900	0.2430	0.3103	0.2781
MatchPyramid	0.0855	0.4466	0.2420	0.2140	0.2512	0.2340
PACRR-DRMM	0.1036	0.4704	0.2720	0.2310	0.2780	0.2545

Table 5.1: Overview of models retrieval effectiveness for the ROBUST04 hard queries

Evaluating explanations Since no ground truth explanations are available for a neural model, we use LIME based explanations as a proxy. Although LIME is model agnostic, we found that it can accurately model relevance for a given query document pair using a simple linear model over words. Table 5.2 shows this observation, where the number of perturbed samples for every query-document pair is divided into train (90%) and test (10%) splits. To produce the explanations from LIME we used the implementation found in [2] along with the score-based modification described in section 5.1. The primary parameters for training a LIME explanation model are the number of perturbed samples to be considered and the number of words for the explanation. The number of perturbed samples is set to 5000 and the number of words is varied based on the experiment. We used the DeepSHAP implementation provided here [3]. Note that we ignore the polarity of the explanation terms provided by both methods in our comparison since the semantics behind the polarities in LIME and DeepSHAP are different. We are more interested in the terms chosen as the explanations in both cases.

For the SHAP explanations we used the methods as described in Section 5.2. Since the model inputs for MatchPyramid and PACRR-DRMM directly accept document tokens it is a one-to-one mapping from the tokens to the corresponding SHAP value. But for DRMM, the model inputs are query-document histograms and thus we obtain SHAP values for these histogram buckets. So in order to map these SHAP values back to the document tokens we store a mapping from the tokens to their respective histogram buckets.

[2] https://github.com/marcotcr/lime
[3] https://github.com/slundberg/shap

	Linear Regression (Ridge)			
NRM	TRAIN MSE	TEST MSE	TRAIN ACC	TEST ACC
DRMM	0.00631	0.00633	0.92662	0.92654
MatchPyramid	0.01827	0.01839	0.90367	0.90387
PACRR-DRMM	0.00165	0.00160	0.98857	0.98980

Table 5.2: Comparison of mean squared error (MSE) and accuracy (ACC) of LIME's linear model across various NRMs. Low MSE and high accuracy shows that it is able to fit and generalize in the query-document locality.

5.4 Results and Discussion

5.4.1 RQ1. Effect of reference input document

The top row in Figure 5.2 illustrates the overlap in terms of jaccard similarity between the explanation terms produced when varying the reference input. Immediately we observe that the overlap between explanations produced is low; below 50% in most cases and consistently across all NRMs. Each reference input method has its own distinct semantics and this is reflected by the low overlap scores. This is also shown by the high Jensen-Shannon divergence scores (2nd row in Fig. 5.2) between the probability distributions of the term coefficients returned by each reference input method. We also find that there is no consistent trend across NRMs. For MatchPyramid, OOV and QL have highest overlap whereas for PACRR-DRMM its OOV and COL that have highest overlap even though both models have the same input representation and parts of the model architecture are similar (convolutional and max-pooling layers). Table 5.3 shows explanations for DRMM across all variants. Once again we see how the explanations can differ significantly if we are not careful in selecting the reference input. For IR, finding the background image seems to be a much harder question.

Our results show how explanations are highly sensitive to the reference input for NRMs chosen in our experiments. This is also indication that a single reference input method may not be the best for every NRM.

5.4.2 RQ2. Accuracy of reference input methods

To help identify which reference input method is most accurate in explaining a given query-document pair, we compared the LIME explanations for the same against it's

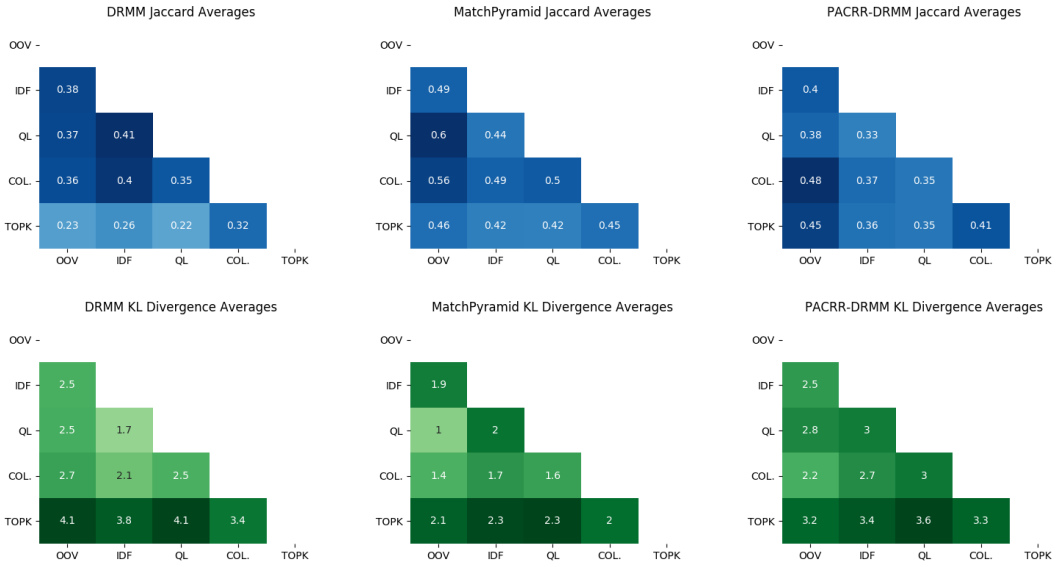

Figure 5.2: Confusion matrices of various DeepSHAP background document methods comparing the overlap in terms of both Jaccard similarities and Jensen-Shannon divergence.

LIME	OOV	IDF	QL	COL.	TOPK
cult	cult	cult	cult	cult	cult
style	followers	style	style	black	**numbers**
followers	black	followers	elite	fraternities	**english**
elite	fraternities	**suspects**	saloon	degenerate	**college**
saloon	degenerate	**belong**	**final**	sons	**university**
student	sons	**reappearing**	march	followers	**fallouts**
home	**academic**	household	**friday**	style	**buccaneers**
members	**american**	black	september	home	**feudings**
march	**tried**	fraternities	**arms**	household	**activists**
september	household	degenerate	**closed**	**avoid**	**troubles**

Table 5.3: An example of words selected by LIME and various DeepSHAP methods with the DRMM model for the query **'cult lifestyles'** and document **'FBIS3-843'** which talks about clashes between cult members and student union's activists at a university in Nigeria. *Words unique to a particular explanation method are highlighted in bold.*

corresponding DeepSHAP explanation methods. In general we found that DeepSHAP produces more explanation terms (see Table 5.4) whereas LIME's L1 regularizer constrains the explanations to only the most important terms (10, 20, 30). Additionally, the discrepancy between the explanations can be attributed to LIME being purely *local*, whereas DeepSHAP has some *global* context since it looks at activations for the whole network which may have captured some global patterns. Hence to estimate which reference input surfaces the most 'ground truth' explanation terms we only computed

	DRMM	MatchPyramid	PACRR-DRMM
OOV	1.000 ± 0.000	0.498 ± 0.087	0.125 ± 0.077
IDF	0.998 ± 0.021	0.500 ± 0.087	0.196 ± 0.113
QL std. form.	0.998 ± 0.012	0.517 ± 0.083	0.205 ± 0.114
COLLECTION	0.999 ± 0.004	0.500 ± 0.087	0.164 ± 0.083
TOPK LIST.	0.999 ± 0.006	0.518 ± 0.095	0.162 ± 0.078

Table 5.4: Comparison of mean and standard deviation of the fraction of terms returned from LIME and DeepSHAP for ROBUST04 hard queries

recall at top 50 and 100 (by shapley value magnitude) DeepSHAP explanation terms (in Table 5.5).

	DRMM						MatchPyramid						PACRR-DRMM					
	top-10		top-20		top-30		top-10		top-20		top-30		top-10		top-20		top-30	
SHAP variants	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100
OOV	0.789	0.905	0.672	0.845	0.615	0.812	**0.793**	**0.843**	**0.656**	**0.726**	**0.566**	**0.640**	0.582	0.582	0.388	0.388	0.299	0.299
IDF	0.830	0.917	0.723	0.871	0.658	0.841	0.795	0.832	0.653	0.711	0.565	0.633	0.633	0.633	0.446	0.446	0.362	0.362
QL	**0.894**	**0.955**	**0.754**	**0.892**	**0.670**	**0.856**	0.765	0.821	0.638	0.711	0.556	0.636	**0.643**	**0.643**	**0.462**	**0.462**	**0.367**	**0.367**
COLLECTION	0.760	0.881	0.673	0.841	0.620	0.815	0.783	0.824	0.639	0.709	0.552	0.630	0.621	0.621	0.429	0.429	0.343	0.343
TOPK LIST.	0.639	0.821	0.606	0.794	0.578	0.788	0.759	0.811	0.624	0.702	0.545	0.627	0.625	0.625	0.425	0.425	0.340	0.340

Table 5.5: Comparison of recall measures at *top-k* (50, 100) terms from DeepSHAP against the *top-k* (10, 20, 30) ground-truth terms from LIME for ROBUST04 hard queries

The first interesting insight is that some NRMs are easier to explain whereas others are more difficult. PACRR-DRMM consistently has a recall less than 0.7 whereas the DeepSHAP explanations of DRMM effectively capture almost all of the LIME explanation terms. When comparing reference input variants within each NRM we find that there is no consistent winner. For DRMM, QL is the best which indicates that sampling terms which are relatively generic for this query in particular is a better 'background image' than sampling generic words from the collection (IDF).

In the case of MatchPyramid, TOPK LIST is the worst performing but it is more difficult to distinguish between the approaches here. The best approach surprisingly is OOV. This can be attributed to how MatchPyramid treats OOV terms. The OOV token is represented by an all-zeros embedding vector that is used for padding the input interaction matrix whereas in DRMM, OOV tokens are filtered out. These preprocessing considerations prove to be crucial when determining the right input reference. Moving on to PACRR-DRMM, we once again find that QL is the best method even though DeepSHAP struggles to find all the LIME terms. It struggles to find more LIME terms because the number of explanation terms returned from DeepSHAP for PACRR-DRMM

is only about a small fraction (0.2) of the document. We have left the investigation of why the number of explanation terms returned by DeepSHAP varies based on different NRMs for future work. In Table 5.6, we show that we get similar insights as discussed above even if we filter out the query terms from the explanation terms returned from both LIME and SHAP.

	DRMM						MatchPyramid						PACRR-DRMM					
	top-10		top-20		top-30		top-10		top-20		top-30		top-10		top-20		top-30	
SHAP variants	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100
OOV	0.762	0.888	0.654	0.836	0.603	0.805	**0.762**	**0.812**	**0.633**	**0.702**	**0.548**	**0.619**	0.472	0.472	0.315	0.315	0.245	0.245
IDF	0.796	0.902	0.702	0.863	0.645	0.836	0.759	0.800	0.628	0.685	0.545	0.612	0.538	0.538	0.380	0.380	0.314	0.314
QL std. form.	**0.874**	**0.945**	**0.737**	**0.884**	**0.657**	**0.851**	0.719	0.782	0.609	0.684	0.536	0.614	**0.552**	**0.552**	**0.399**	**0.399**	**0.319**	**0.319**
COLLECTION	0.721	0.864	0.654	0.832	0.609	0.811	0.741	0.788	0.611	0.682	0.532	0.607	0.524	0.524	0.362	0.362	0.292	0.292
TOPK LIST.	0.601	0.804	0.588	0.786	0.568	0.783	0.715	0.776	0.596	0.676	0.524	0.606	0.529	0.529	0.357	0.357	0.289	0.289

Table 5.6: Comparison of recall measures at *top-k* (50, 100) terms from DeepSHAP against the *top-k* (10, 20, 30) ground-truth terms from LIME for ROBUST04 hard queries; **From both sets of terms we ignore the query terms**

5.4.3 RQ3. Effect of reference query distribution

In order to understand the effect of using a reference query input distribution along with the different reference document methods, we construct a reference query input by sampling words that have low IDF scores from the collection. We evaluate the DeepSHAP explanations by computing recall at top 50 and 100 with respect to 'ground truth' terms (in Table 5.7) as in section 5.4.2.

	DRMM						MatchPyramid						PACRR-DRMM					
	top-10		top-20		top-30		top-10		top-20		top-30		top-10		top-20		top-30	
SHAP variants	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100	recall @50	recall @100
OOV	0.783	0.903	0.662	0.846	0.604	0.815	**0.789**	**0.840**	**0.658**	**0.730**	**0.568**	**0.648**	0.565	0.565	0.378	0.378	0.293	0.293
IDF	0.885	0.948	0.751	0.897	0.664	0.858	0.754	0.813	0.627	0.712	0.553	0.643	0.646	0.646	0.459	0.459	0.372	0.372
QL std. form.	**0.891**	**0.954**	**0.760**	**0.892**	**0.676**	**0.861**	0.746	0.811	0.621	0.707	0.541	0.637	**0.645**	**0.645**	**0.463**	**0.463**	**0.380**	**0.380**
COLLECTION	0.824	0.901	0.699	0.850	0.640	0.821	0.756	0.815	0.626	0.707	0.545	0.633	0.613	0.613	0.422	0.422	0.335	0.335
TOPK LIST.	0.818	0.907	0.698	0.857	0.634	0.827	0.746	0.803	0.619	0.695	0.540	0.622	0.610	0.610	0.418	0.418	0.334	0.334

Table 5.7: Comparison of recall measures at *top-k* (50, 100) terms from DeepSHAP using also query distribution sampled from low IDF scores against the *top-k* (10, 20, 30) ground-truth terms from LIME for ROBUST04 hard queries

Similar to the insights described in section 5.4.2, we see that the best performing approaches across each of the models are the same–DRMM (QL), MatchPyramid (OOV)

and PACRR-DRMM (QL). Thus, using a reference query input distribution doesn't effect the best performing reference document input variants within each NRM. However, for DRMM we observe that it is now hard to distinguish between various approaches which is different from the insights in Table 5.5. This could be because of the low activations through the attention network in DRMM from the query reference input due to the use of irrelevant words sampled from IDF. But we don't see such differences in MatchPyramid and PACRR-DRMM, as it was already difficult to distinguish between the approaches when the reference query input was fixed and also these two models don't have an attention network.

Chapter 6

Conclusion

In this thesis, we successfully trained various *interaction-based* neural ranking models on the standard TREC newswire collection (Robust04) and reproduced results that had a similar retrieval effectiveness or a retrieval performance close to what was reported in the papers across all the evaluation metrics. The few differences that we observed in our experiments were mainly due to the lack of information regarding the exact splits of the data that were used for cross-validation, or the exact hyper-parameters used to tune the model along with the amount of data that was sampled (number of iterations or epochs for which the model was trained, number of query-document pairs sampled) or how the ranking pairs were sampled from the relevance judgements. The two neural ranking models–PACRR-DRMM and NPRF-DRMM for which the above data was reported or shared, helped us train models that had retrieval effectiveness exactly as reported in the paper. We also investigated the reproducibility of DSSM, a *representation-based* neural ranking model which has been shown to perform well only on large-scale click-through proprietary datasets. We had tried various approaches that included pre-training the triletter representations using the target corpus, or weak supervision using AOL queries to generate a large set of training data and a combination of both, but couldn't train an effective model which was also highlighted in a parallel study [Nie et al., 2018b]. In the future, we would like to investigate in detail as to why we were not able train an effective model on the Robust04 newswire collection, and if given a large public dataset (MS-MARCO) can we train an effective DSSM model. This thesis covers an extensive survey of the existing neural ranking approaches in the literature as well (Chapter 2).

In this thesis, we also investigated the applicability of existing *model-introspective* interpretability approaches in ML, such as, DeepSHAP for explaining the decisions of complex neural ranking models–*why* a particular document was considered relevant for the query. We suggested several reference input methods for DeepSHAP that take into account the unique semantics of document ranking and relevance in IR. Through quantitative experiments we found that it is indeed sensitive to the reference input. The distinct lack of overlap in most cases was surprising but in line with recent works on the lack of robustness in interpretability approaches [Ghorbani et al., 2019]. We also tried to evaluate which reference method is more accurate by comparing against a *model-agnostic* approach LIME. Here we found that the reference input method selection is highly dependent on the neural ranking model at hand. We believe that this work exposes new problems when dealing with model introspective interpretability for NRMs.

A worthwhile endeavor will be to investigate new approaches that explicitly take into account the discreteness of text and the model's preprocessing choices when generating explanations. It would also be interesting to consider how we can answer the following ranking questions to help the user understand the underlying neural ranking model better:

- What is the intent of the query encoded by the neural ranking model? – Could we simply aggregate the term weights returned by DeepSHAP from top-k results and pick the top-n words to indicate intent of the query.
- Why a particular document d_A is ranked higher than another document d_B in the ranked list? – Could we construct a 'reference' input from the distribution of terms in the lower ranked document d_B that would help pick terms which are different or unique that explain the higher relevance of d_A.

Appendix A

LIME vs SHAP Examples

In this appendix, we include qualitative examples of the top-10 explanation terms selected by LIME and the various SHAP methods for the query '*cult lifestyles*' and the document '*FBIS3-843*' which talks about clashes between cult members and student union's activists at a university in Nigeria. Examples are included for all the 3 neural ranking models–DRMM (Tab. A.1), MatchPyramid (Tab. A.2) and PACRR-DRMM (Tab. A.3). In all the tables, words unique to a particular explanation method are highlighted in bold.

This example query-document pair reflects how the different reference input methods encode its own distinct semantics and why DeepSHAP is highly sensitive to the reference input distribution. Thus, explanation terms can be quite different if we are not careful in selecting the reference input especially when comparing it with *model-agnostic* approaches like LIME.

LIME	OOV	IDF	QL	COL.	TOPK
cult	cult	cult	cult	cult	cult
style	followers	style	style	black	**numbers**
followers	black	followers	elite	fraternities	**english**
elite	fraternities	**suspects**	saloon	degenerate	**college**
saloon	degenerate	**belong**	**final**	sons	**university**
student	sons	**reappearing**	march	followers	**fallouts**
home	**academic**	household	**friday**	style	**buccaneers**
members	**american**	black	september	home	**feudings**
march	**tried**	fraternities	**arms**	household	**activists**
september	household	degenerate	**closed**	**avoid**	**troubles**

Table A.1: Explanation words selected by LIME and DeepSHAP methods for DRMM

LIME	OOV	IDF	QL	COL.	TOPK
cult	cult	cult	cult	cult	cult
focus	activities	**bloody**	activities	end	end
style	focus	**complain**	violent	call	call
type	type	**suspects**	article	action	style
gang	article	secret	type	style	**injected**
activities	bfn	members	**recording**	bfn	action
free	**taken**	campus	style	activities	**tried**
described	described	**hired**	focus	**today**	bfn
violent	style	followers	campus	**leader**	described
members	end	activities	action	secret	followers

Table A.2: Explanation words selected by LIME and DeepSHAP methods for MatchPyramid

LIME	OOV	IDF	QL	COL.	TOPK
cult	cult	cult	cult	cult	cult
style	**language**	**60**	**peugeot**	**article**	to
cults	their	**described**	to	english	their
their	english	**student**	**as**	their	**expel**
activities	activities	activities	activities	on	**with**
the	style	**union**	the	**university**	members
fallouts	suspects	their	their	activities	activities
themselves	members	**nigeria**	**academic**	style	style
by	the	suspects	**a**	**lagos**	the
suspects	and	on	and	members	**naked**

Table A.3: Explanation words selected by LIME and DeepSHAP methods for PACRR-DRMM

Appendix B

LIME vs SHAP experiments (precision-recall)

In this appendix, we compare on an average at what rank (based on shapley value magnitude) across the various DeepSHAP variants are the LIME 'ground truth' explanation terms at different recall percentages found.

	DRMM					MatchPyramid					PACRR-DRMM				
Recall %	OOV	IDF	QL	COL.	TOPK	OOV	IDF	QL	COL.	TOPK	OOV	IDF	QL	COL.	TOPK
0.1	0.975	0.997	1.000	0.981	0.972	0.987	0.992	0.981	0.985	0.969	0.979	0.982	0.978	0.977	0.912
0.2	0.931	0.965	0.979	0.889	0.730	0.918	0.917	0.917	0.914	0.810	0.939	0.913	0.908	0.921	0.774
0.3	0.835	0.901	0.939	0.823	0.540	0.839	0.838	0.803	0.807	0.679	0.809	0.720	0.731	0.755	0.637
0.4	0.710	0.813	0.881	0.695	0.367	0.673	0.689	0.630	0.645	0.554	0.679	0.536	0.531	0.613	0.553
0.5	0.635	0.729	0.800	0.586	0.292	0.554	0.581	0.522	0.534	0.488	0.603	0.446	0.443	0.517	0.508
0.6	0.547	0.648	0.712	0.507	0.237	0.456	0.488	0.429	0.451	0.401	0.576	0.414	0.391	0.477	0.479
0.7	0.449	0.543	0.603	0.413	0.192	0.377	0.411	0.362	0.367	0.319	0.580	0.389	0.385	0.436	0.467
0.8	0.346	0.457	0.496	0.333	0.157	0.317	0.341	0.291	0.309	0.272	0.595	0.381	0.375	0.371	0.417
0.9	0.261	0.334	0.380	0.242	0.137	0.274	0.305	0.257	0.283	0.244	0.583	0.426	0.394	0.425	0.405
1.0	0.189	0.214	0.246	0.172	0.128	0.216	0.226	0.199	0.248	0.210	0.000	0.411	0.442	0.476	0.000

Table B.1: Comparison of mean precision of DeepSHAP variants at various recall percentages of the *top-10* ground-truth terms from LIME for ROBUST04 hard queries (50)

	DRMM					MatchPyramid					PACRR-DRMM				
Recall %	OOV	IDF	QL	COL.	TOPK	OOV	IDF	QL	COL.	TOPK	OOV	IDF	QL	COL.	TOPK
0.1	0.913	0.955	1.000	1.000	1.000	0.964	0.952	0.929	0.946	0.899	0.767	0.700	0.667	0.700	0.840
0.2	0.824	0.692	0.918	0.694	0.590	0.732	0.829	0.749	0.742	0.734	0.783	0.683	0.567	0.767	0.700
0.3	0.841	0.642	0.789	0.589	0.391	0.665	0.794	0.683	0.601	0.491	0.857	0.616	0.467	0.725	0.570
0.4	0.628	0.646	0.795	0.501	0.238	0.536	0.655	0.608	0.525	0.420	0.750	0.553	0.482	0.718	0.536
0.5	0.495	0.570	0.650	0.451	0.248	0.381	0.417	0.381	0.378	0.330	0.463	0.321	0.282	0.361	0.372
0.6	0.568	0.489	0.594	0.340	0.129	0.413	0.440	0.454	0.351	0.293	0.929	0.600	0.633	0.625	0.508
0.7	0.375	0.436	0.473	0.302	0.116	0.351	0.380	0.360	0.306	0.267	0.818	0.525	0.510	0.561	0.500
0.8	0.332	0.341	0.394	0.299	0.120	0.300	0.321	0.312	0.323	0.219	0.786	0.461	0.508	0.472	0.401
0.9	0.254	0.303	0.326	0.275	0.115	0.307	0.365	0.347	0.293	0.189	0.900	0.500	0.643	0.500	0.000
1.0	0.173	0.190	0.217	0.150	0.111	0.187	0.193	0.174	0.214	0.178	0.000	0.345	0.373	0.389	0.000

Table B.2: Comparison of mean precision of DeepSHAP at various recall percentages of the *top-10* ground-truth terms from LIME for ROBUST04 difficult queries (50); **From both sets of terms we ignore the query terms**

Appendix C

Public Implementations of Neural IR models

We include the link to all the publicly available implementations of existing neural IR models in the table below.

Paper	Implementation Link
HiNT	https://github.com/faneshion/HiNT
DeepRank	https://github.com/pl8787/textnet-release
DRMM	https://github.com/faneshion/DRMM, MatchZoo[a]
NPRF	https://github.com/ucasir/NPRF
PACRR-DRMM	https://github.com/nlpaueb/deep-relevance-ranking
MatchPyramid	MatchZoo[a]
DeepTileBars	https://github.com/smt-HS/DeepTileBars-release
SNRM Zamani et al. [2018]	https://github.com/hamed-zamani/snrm
PACRR Co-PACRR	https://github.com/khui/copacrr
KNRM	https://github.com/AdeDZY/K-NRM, MatchZoo[a]
DUET	https://github.com/bmitra-msft/NDRM, MatchZoo[a]
DSSM	https://github.com/NTMC-Community/MatchZoo
C-DSSM	https://github.com/airalcorn2/Deep-Semantic-Similarity-Model, MatchZoo[a]

[a] https://github.com/NTMC-Community/MatchZoo

Table C.1: Link to public repositories of neural IR models.

Bibliography

[1] Ai, Q., Yang, L., Guo, J., and Croft, W. B. (2016a). Analysis of the paragraph vector model for information retrieval. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 133–142. ACM.

[2] Ai, Q., Yang, L., Guo, J., and Croft, W. B. (2016b). Improving language estimation with the paragraph vector model for ad-hoc retrieval. In *Proceedings of the 39th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '16, pages 869–872. ACM.

[3] Arras, L., Horn, F., Montavon, G., Müller, K.-R., and Samek, W. (2017). "what is relevant in a text document?": An interpretable machine learning approach. *PLOS ONE*, 12:1–23.

[4] Bach, S., Binder, A., Montavon, G., Klauschen, F., Müller, K.-R., and Samek, W. (2015). On pixel-wise explanations for non-linear classifier decisions by layer-wise relevance propagation. *PLOS ONE*, 10:1–46.

[5] Berger, A. and Lafferty, J. (1999). Information retrieval as statistical translation. In *Proceedings of the 22nd Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '99, pages 222–229. ACM.

[6] Bojanowski, P., Grave, E., Joulin, A., and Mikolov, T. (2017). Enriching word vectors with subword information. *Transactions of the Association for Computational Linguistics*, 5:135–146.

[7] Cohen, D., O'Connor, B., and Croft, W. B. (2018). Understanding the representational power of neural retrieval models using nlp tasks. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '18, pages 67–74. ACM.

[8] Craswell, N., Croft, W. B., de Rijke, M., Guo, J., and Mitra, B. (2017). Sigir 2017 workshop on neural information retrieval (Neu-IR'17). In *Proceedings of 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1431–1432. ACM.

[9] Craswell, N., Croft, W. B., de Rijke, M., Guo, J., and Mitra, B. (2018). Report on the second sigir workshop on neural information retrieval (Neu-IR'17). *SIGIR Forum*, 51(3):152–158.

[10] Dai, Z., Xiong, C., Callan, J., and Liu, Z. (2018). Convolutional neural networks for soft-matching n-grams in ad-hoc search. In *Proceedings of the 11th ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 126–134. ACM.

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

[12] Dehghani, M., Mehrjou, A., Gouws, S., Kamps, J., and Schölkopf, B. (2018a). Fidelity-weighted learning. In *International Conference on Learning Representations*, ICLR '18.

[13] Dehghani, M., Severyn, A., Rothe, S., and Kamps, J. (2018b). Learning to learn from weak supervision by full supervision. In *NIPS'17 workshop on Meta-Learning*, MetaLearn '17.

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

[15] Diaz, F., Mitra, B., and Craswell, N. (2016). Query expansion with locally-trained word embeddings. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 367–377.

[16] Fan, Y., Guo, J., Lan, Y., Xu, J., Zhai, C., and Cheng, X. (2018). Modeling diverse relevance patterns in ad-hoc retrieval. In *The 41st International ACM SIGIR Conference on Research & Development in Information Retrieval*, SIGIR '18, pages 375–384. ACM.

[17] Ganguly, D., Roy, D., Mitra, M., and Jones, G. J. (2015). 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. ACM.

[18] Ghorbani, A., Abid, A., and Zou, J. Y. (2019). Interpretation of neural networks is fragile. *AAAI'19*.

[19] Guo, J., Fan, Y., Ai, Q., and Croft, W. B. (2016). 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. ACM.

[20] Hearst, M. A. (1994). Multi-paragraph segmentation of expository text. In *Proceedings of the 32nd Annual Meeting on Association for Computational Linguistics*, ACL '94, pages 9–16. Association for Computational Linguistics.

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

[22] Hui, K., Yates, A., Berberich, K., and de Melo, G. (2017). PACRR: A position-aware neural ir model for relevance matching. In *Proceedings of the 2017 Conference on Empirical Methods in Natural Language Processing*, pages 1049–1058, Copenhagen, Denmark.

[23] Hui, K., Yates, A., Berberich, K., and de Melo, G. (2018). Co-PACRR: A context-aware neural ir model for ad-hoc retrieval. In *Proceedings of the Eleventh ACM International Conference on Web Search and Data Mining*, WSDM '18, pages 279–287. ACM.

[24] Lavrenko, V. and Croft, W. B. (2001). 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 York, NY, USA. ACM.

[25] Le, Q. and Mikolov, T. (2014). Distributed representations of sentences and documents. In *Proceedings of the 31st International Conference on International Conference on Machine Learning - Volume 32*, ICML'14, pages II–1188–II–1196. JMLR.org.

[26] LeCun, Y., Bengio, Y., and Hinton, G. (2015). Deep learning. *nature*, 521(7553):436.

[27] Letham, B., Rudin, C., McCormick, T. H., and Madigan, D. (2015). Interpretable classifiers using rules and bayesian analysis: Building a better stroke prediction model. *The Annals of Applied Statistics*, 9(3):1350–1371.

[28] Li, B., Cheng, P., and Jia, L. (2018a). Joint learning from labeled and unlabeled data for information retrieval. In *Proceedings of the 27th International Conference on Computational Linguistics*, pages 293–302. Association for Computational Linguistics.

[29] Li, C., Sun, Y., He, B., Wang, L., Hui, K., Yates, A., Sun, L., and Xu, J. (2018b). NPRF: A neural pseudo relevance feedback framework for ad-hoc information retrieval. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*.

[30] Liu, T.-Y. et al. (2009). Learning to rank for information retrieval. *Foundations and Trends® in Information Retrieval*, 3(3):225–331.

[31] Lundberg, S. M. and Lee, S.-I. (2017). A unified approach to interpreting model predictions. In *Advances in Neural Information Processing Systems 30*, pages 4765–4774.

[32] Matveeva, I., Burges, C., Burkard, T., Laucius, A., and Wong, L. (2006). High accuracy retrieval with multiple nested ranker. In *Proceedings of the 29th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '06, pages 437–444. ACM.

[33] McDonald, R., Brokos, G., and Androutsopoulos, I. (2018). Deep relevance ranking using enhanced document-query interactions. In *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, pages 1849–1860. ACL.

[34] Mikolov, T., Sutskever, I., Chen, K., Corrado, G., and Dean, J. (2013). Distributed representations of words and phrases and their compositionality. In *Proceedings of the 26th International Conference on Neural Information Processing Systems - Volume 2*, NIPS'13, pages 3111–3119.

[35] Mitra, B. and Craswell, N. (2018). An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126.

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

[37] Mitra, B., Nalisnick, E. T., Craswell, N., and Caruana, R. (2016). A dual embedding space model for document ranking. *arXiv preprint*, arXiv:1602.01137.

[38] Nalisnick, E., Mitra, B., Craswell, N., and Caruana, R. (2016). Improving document ranking with dual word embeddings. In *Proceedings of the 25th International Conference Companion on World Wide Web*, WWW '16 Companion, pages 83–84.

[39] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). MS MARCO: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

[40] Nie, Y., Li, Y., and Nie, J.-Y. (2018a). Empirical study of multi-level convolution models for ir based on representations and interactions. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '18, pages 59–66. ACM.

[41] Nie, Y., Li, Y., and Nie, J.-Y. (2018b). Empirical study of multi-level convolution models for ir based on representations and interactions. In *Proceedings of the 2018 ACM SIGIR International Conference on Theory of Information Retrieval*, ICTIR '18, pages 59–66, New York, NY, USA. ACM.

[42] Nie, Y., Sordoni, A., and Nie, J.-Y. (2018c). Multi-level abstraction convolutional model with weak supervision for information retrieval. In *the 41st International ACM SIGIR Conference*, SIGIR '18, pages 985–988. ACM.

[43] Onal, K. D., Zhang, Y., Altingovde, I. S., Rahman, M. M., Karagoz, P., Braylan, A., Dang, B., Chang, H.-L., Kim, H., Mcnamara, Q., Angert, A., Banner, E., Khetan, V., Mcdonnell, T., Nguyen, A. T., Xu, D., Wallace, B. C., Rijke, M., and Lease, M. (2018). Neural information retrieval: At the end of the early years. *Information Retrieval Journal*, 21(2-3):111–182.

[44] Pang, L., Lan, Y., Guo, J., Xu, J., and Cheng, X. (2016). A study of MatchPyramid models on ad-hoc retrieval. *SIGIR workshop on Neural Information Retrieval (NeuIR-16)*, arXiv:1606.04648.

[45] Pang, L., Lan, Y., Guo, J., Xu, J., and Cheng, X. (2017a). A deep investigation of deep IR models. *SIGIR workshop on Neural Information Retrieval (NeuIR-17)*, arXiv:1707.07700.

[46] Pang, L., Lan, Y., Guo, J., Xu, J., Xu, J., and Cheng, X. (2017b). DeepRank: A new deep architecture for relevance ranking in information retrieval. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, CIKM '17, pages 257–266. ACM.

[47] Pennington, J., Socher, R., and Manning, C. D. (2014). Glove: Global vectors for word representation. In *Empirical Methods in Natural Language Processing*, EMNLP '14, pages 1532–1543.

[48] Rennings, D., Moraes, F., and Hauff, C. (2019). An axiomatic approach to diagnosing neural ir models. In *European Conference on Information Retrieval '19*.

[49] Ribeiro, M. T., Singh, S., and Guestrin, C. (2016). "why should i trust you?": Explaining the predictions of any classifier. In *Proceedings of the 22nd ACM SIGKDD International Conference on Knowledge Discovery and Data Mining*, KDD '16, pages 1135–1144. ACM.

[50] Ribeiro, M. T., Singh, S., and Guestrin, C. (2018). Anchors: High-precision model-agnostic explanations. In *AAAI '18*.

[51] Roy, D., Ganguly, D., Mitra, M., and Jones, G. J. F. (2016). Representing documents and queries as sets of word embedded vectors for information retrieval. *SIGIR workshop on Neural Information Retrieval (NeuIR-16)*.

[52] Salakhutdinov, R. and Hinton, G. (2009). Semantic hashing. *International Journal of Approximate Reasoning*, 50(7):969 – 978. Special Section on Graphical Models and IR.

[53] Shapley, L. S. (1953). A value for n-person games. *Contributions to the Theory of Games*, 2(28):307–317.

[54] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014a). A latent semantic model with convolutional-pooling structure for information retrieval. In *Proceedings of the 23rd ACM International Conference on Conference on Information and Knowledge Management*, CIKM '14, pages 101–110. ACM.

[55] Shen, Y., He, X., Gao, J., Deng, L., and Mesnil, G. (2014b). 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. ACM.

[56] Shrikumar, A., Greenside, P., and Kundaje, A. (2017). Learning important features through propagating activation differences. *arXiv preprint*, abs/1704.02685.

[57] Simonyan, K., Vedaldi, A., and Zisserman, A. (2014). Deep inside convolutional networks: Visualising image classification models and saliency maps. *ICLR'14 Workshop.*

[58] Singh, J. and Anand, A. (2019). Exs: Explainable search using local model agnostic interpretability. In *Proceedings of the 12th ACM International Conference on Web Search and Data Mining*, WSDM '19, pages 770–773. ACM.

[59] Tang, Z. and Yang, G. H. (2019). Deeptilebars: Visualizing term distribution for neural information retrieval. *AAAI conference on Artificial Intelligence*, AAAI '19.

[60] Ustun, B. and Rudin, C. (2016). Supersparse linear integer models for optimized medical scoring systems. *Machine Learning*, 102(3):349–391.

[61] Voorhees, E. M. (2005). Overview of the TREC 2004 robust track. In *TREC, volume Special Publication 500-261*. National Institute of Standards and Technology (NIST).

[62] Xiong, C., Dai, Z., Callan, J., Liu, Z., and Power, R. (2017). 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. ACM.

[63] Xu, K., Ba, J. L., Kiros, R., Cho, K., Courville, A., Salakhutdinov, R., Zemel, R. S., and Bengio, Y. (2015). Show, attend and tell: Neural image caption generation with visual attention. In *International Conference on Machine Learning - Volume 37*, ICML'15, pages 2048–2057.

[64] Yates, A. and Hui, K. (2017). DE-PACRR: exploring layers inside the PACRR model. *CoRR*, abs/1706.08746.

[65] Zamani, H. and Croft, W. B. (2016). Embedding-based query language models. In *Proceedings of the 2016 ACM International Conference on the Theory of Information Retrieval*, ICTIR '16, pages 147–156. ACM.

[66] Zamani, H. and Croft, W. B. (2018). 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. ACM.

[67] Zamani, H., Dehghani, M., Croft, W. B., Learned-Miller, E., and Kamps, J. (2018). 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. ACM.

[68] Zuccon, G., Koopman, B., Bruza, P., and Azzopardi, L. (2015). 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. ACM.