End-to-end Neural Information Retrieval

by

Wei Yang

A thesis
presented to the University of Waterloo
in fulfillment of the
thesis requirement for the degree of
Master
in
Computer Science

Waterloo, Ontario, Canada, 2019

I hereby declare that I am the sole author of this thesis. This is a true copy of the thesis, including any required final revisions, as accepted by my examiners.

I understand that my thesis may be made electronically available to the public.

Abstract

In recent years we have witnessed many successes of neural networks in the information retrieval community with lots of labeled data. Yet it remains unknown whether the same techniques can be easily adapted to search social media posts where the text is much shorter. In addition, we find that most neural information retrieval models are compared against weak baselines. In this thesis, we build an end-to-end neural information retrieval system using two toolkits: Anserini and MatchZoo. In addition, we also propose a novel neural model to capture the relevance of short and varied tweet text, named MP-HCNN.

With the information retrieval toolkit Anserini, we build a reranking architecture based on various traditional information retrieval models (QL, QL+RM3, BM25, BM25+RM3), including a strong pseudo-relevance feedback baseline: RM3. With the neural network toolkit MatchZoo, we offer an empirical study of a number of popular neural network ranking models (DSSM, CDSSM, KNRM, DUET, DRMM). Experiments on datasets from the TREC Microblog Tracks and the TREC Robust Retrieval Track show that *most* existing neural network models cannot beat a simple language model baseline. However, DRMM provides significant improvement over the pseudo-relevance feedback baseline (BM25+RM3) on the Robust04 dataset and DUET, DRMM and MP-HCNN can provide significant improvements over the baseline (QL+RM3) on the microblog datasets. Further detailed analyses suggest that searching social media and searching news articles exhibit several different characteristics that require customized model design, shedding light on future directions.

Acknowledgements

I would like to thank Professor Jimmy Lin for his great support and help during my study and research in University of Waterloo. I would like to thank Jinfeng Rao, Peilin Yang, Kuang Lu for their great help in my thesis experiments.

I would like to thank the readers of my thesis, Professor Ming Li and Professor Charles Clarke, for reviewing my work.

Finally, I would like to thank my family for the encouragement during my study abroad.

Dedication

This is dedicated to my family.

Table of Contents

List of Tables

List of Figures

Chapter 1

Introduction

Text matching is one of the most important problems in natural language processing. Given two texts (which could be sentences, paragraphs or passages), the target of text matching is to compute the similarity of these two texts. Many natural language processing problems can be modeled as or transformed into a text matching problem such as information retrieval, question answering and so on. The main challenge of text matching comes from:

- **Hierarchical matching signals at different levels**: Word matching, phrase matching, and sentence matching are all important in query-document matching and it is difficult to define the weights of each matching signal. Document matching is the highest level matching, which requires the matching algorithm to summarize the main idea of the whole paragraph or passage. Although it has been researched for many decades, it still remains unclear how much matching signal in the lower granularities (word matching, phrase matching, and sentence matching) document matching requires.
- **The trade-off between exact matching and semantic matching**: exact matching requires matching algorithms to find the text pairs that share the most common words, while semantic matching considers the semantics of the words, phrases, and sentences. The target of these two requirements are similar, but not exactly the same and sometimes we need to sacrifice one to maximize the other. For example, it has been shown in previous work [21] that information retrieval prefers exact matching and question answering needs more semantic matching.

The traditional text matching problem is solved by a large amount of hard-coded feature engineering based on bag-of-words and bag-of-phrases models. Most feature-based methods

have few learnable parameters. For example, Bing or Google has thousands of features, which means we need to design domain-specific features for each task and dataset. Then, applying these domain-specific features to a new or general domain is difficult. However, these bag-of-words based methods cannot leverage all information required by the text matching task. For example, rich text semantics at the sentence level is ignored by the traditional bag-of-words and bag-of-phrases methods. On the other hand, in short text matching, the length is relatively small (<100) compared to long documents such as news. Since there are very few terms after tokenization and stop word removal, bag-of-words based models are not satisfying because they cannot leverage the meaning behind the terms. To address this issue and extract more matching signals, latent semantic models were proposed such as LSA, PLAS, LDA, which largely attracted NLP researchers' attention. Recently, neural network models (especially deep learning models) have been applied to learn the semantic representation and matching of two texts.

With the popularity of neural network models and their application in natural language processing, a number of neural text matching models have been proposed. These neural text matching methods can automatically learn features from the text and the target. Furthermore, most deep neural text matching methods start encoding the sentence using word embeddings, which can be pretrained by unsupervised methods such as word2vec [53] or Glove [63]. These word embeddings (vectors) are a low-dimension representation of words and often contain rich semantic information. In addition, there is representation learning at multiple levels using deep neural network models, which successfully leverages hierarchical matching signals in text matching.

In comparison, the breakthrough of neural networks is not as great in the information retrieval (IR) community. Although distributed word representations offer an opportunity to overcome the classic vocabulary mismatch problem in IR [16], there are still some fundamental challenges to be solved. Guo et al. [21] first pointed out three key differences between a *relevance matching* problem in IR and *semantic matching* in NLP: exact matching signals, term importance weighting, and diverse matching requirements. Essentially, IR problems still rely on exact matching signals between query and document as a strong relevance indicator. In addition, because queries are short, it is ineffective to treat each query word/phrase equally, and hence, query term weighting can be important. To overcome these problems, many neural network models [89, 55] have been proposed recently that perform well on *ad hoc* retrieval tasks on Web documents.

1.1 Problem Definition

A typical search-based question answering system follows the pipeline: (1) build the index on all candidates offline, (2) receive queries online and search the index to get the first batch of candidate replies, (3) apply matching models to rerank the candidates and return the top K. To bridge the gap between information retrieval and text matching, the target of our thesis work is to build an end-to-end retrieval system based on four popular open source projects: Anserini, MatchZoo, MP-HCNN and BERT. In this work, we integrate Anserini (a state-of-the-art IR toolkit) with two different neural retrieval frameworks for end-to-end neural IR: (1) MatchZoo, using existing neural models, and (2) MP-HCNN, a model we developed.

Anserini is an open source information retrieval toolkit for replicable information retrieval research built on Lucene.[1] Anserini helps simplify ad hoc experimentation and allow researchers to easily reproduce results with modern bag-of-words ranking models on diverse test collections. It contains experiments on 13 test collections with regression tests in version 0.3.0, which is built on Lucene 7.6.

MatchZoo is a toolkit for text matching. It implements 11 popular deep text matching models in version 1.0. Although some of these models are not designed for information retrieval, we can apply them to estimate the relevance between the query and document. For example, ARC-I, ARC-II, and MV-LSTM are targeted for short text semantic matching. The model structure does not limit the application on the short text pairs only, and similarity scores in semantic matching can be treated as relevance scores in information retrieval.

MP-HCNN is a neural text matching model co-developed by the author of this thesis and Jinfeng Rao, and is specifically designed for ranking short social media posts. Experiments in the MP-HCNN paper on the Microblog dataset show that our model significantly outperforms prior feature-based methods, as well, and existing neural ranking models.

Through combining Anserini and MatchZoo/MP-HCNN, the goal of our thesis is to build an open source end-to-end system for neural information retrieval and share the baseline' results. Beyond that, we also try to survey recent work on end-to-end neural information retrieval. By comparing previous results with our results, we show the importance of robust engineering work, which can enhance the effectiveness of the whole system.

[1] http://anserini.io

1.2 Contributions

We summarize our contributions in this thesis as follows:

- We build an end-to-end retrieval and reranking system to allow the researchers to select different retrieval models and neural reranking models on a specific dataset. This work bridges the gap between information retrieval and text matching. We report the hyper-parameters in both steps to make sure the results are replicable. In addition, neural IR baselines provided throughout this work will be publicly available and provide the standard for future neural IR research.
- By combining traditional IR methods (QL, QL+RM3, BM25, BM25+RM3) and neural text matching models (DSSM, CDSSM, DUET, KNRM, DRMM, MP-HCNN, and BERT), we present detailed experimental results on datasets from two domains: newswire articles and microblog, representing two types of documents in IR. We build strong baselines using our system and show the importance of a strong baseline that a neural network model should build on.
- We achieve state-of-the-art effectiveness on two benchmark datasets using existing methods. This results from employing effective term-based IR methods in neural IR research and a novel neural network model we propose. Furthermore, we provide analyses based on these results comparing the performance of different models on different datasets.

1.3 Thesis Organization

Chapter 2 introduces related work in neural information retrieval and text matching models, including most models used in this thesis. Chapter 3 introduces all toolkits used and describes how we integrate them into a pipeline toolkit. We will also introduce a simple but useful interpolation method to combine the scores from both retrieval models and neural text matching models. Chapter 4 describe how we conduct the experiments and shows all of our experimental results on two datasets: Robust04 and Microblog. Chapter 5 concludes and summarizes our work and discusses some interesting future research directions.

Chapter 2

Related Work

2.1 Unsupervised Retrieval Methods

As introduced in Chapter 1, the information toolkit Anserini provides basic and traditional retrieval models such as Query likelihood (QL), BM25, and RM3, where RM3 is a query expansion method and can be combined with QL or BM25. We will use QL+RM3 and BM25+RM3 to represent the combination in this paper.

2.1.1 Query likelihood with Dirichlet smoothing

QL (short for Query likelihood) is a simple model in information retrieval, which maximizes the likelihood of the document given the query terms. Dirichlet smoothing method is typically used to handle the unseen term during the computation of scores. [94]

2.1.2 Okapi BM25

BM25 (short for Okapi BM25) is a ranking function to estimate the query-document relevance. This algorithm is not affected by word semantic types such as whether the word is a noun or a verb, and the meaning of each word. It is only sensitive to which are common words and which are rare words, and the document length. If one query contains both common words and rare words, this method puts more weight on the rare words and returns documents with more rare words in the query. Besides, a term saturation mechanism

is applied to descrease the matching signal when a matched word appears too frequently in the document.

There are two parameters in BM25, k_1 and b, representing term frequency saturation and field-length normalization respectively. However, neural IR researchers often forget to report their parameters of BM25 when they use it.

Note that both QL and BM25 are exact match based approaches. Word matching between queries and documents is simply counted. This might not be the most effective choice in information retrieval. Thus, some word embedding based methods have been introduced to enhance the two baselines in recent years. [93, 16]

2.1.3 RM3

RM3 [37] is a query expansion based model combining the QL score with a relevance model using pseudo-relevance feedback. RM3 is an effective method in traditional IR methods. However, neural IR researchers in recent years often ignore this method as a baseline. Instead, weaker retrieval models like QL and BM25 are commonly used. We will show RM3 can effectively boost the performance of neural text matching models in the experiments in Section 4.

2.2 Learning to Rank Methods

Before the surge of neural networks, learning to rank (L2R) was a field that took advantage of recent advances in machine learning to improve ranking effectiveness. Existing work on L2R can be summarized into three main categories: pointwise, pairwise and listwise. The main difference lies in the problem formulations with different assumptions, input/output spaces, and loss functions. Pointwise methods, focus on learning a relevance score for each query-document pair represented in a feature space, while pairwise approaches, such as LambdaMART [5] and RankSVM [32], aim to learn the preference between a pair of documents to a query. Listwise approaches, such as ListNet [6], directly optimize the input list of documents to a query to find the best-ranked list. The major drawback of L2R is that it requires manual feature engineering, which can be time-consuming, incomplete, and difficult to generalize to other problems.

2.3 Neural Information Retrieval Methods

2.3.1 Neural Network Basics

Convolution Neural Networks (CNN) can efficiently capture local features. Ever since LeCun et al. [39] first applied backpropagation and gradient-based learning to train CNNs and succeeded in document recognition, CNNs have become one of the most widely used neural networks in various areas such as image recognition [35], speech recognition [1] and natural language processing [15]. In a traditional feedforward neural network, we connect each input neuron to each output neuron in the next layer. That is also called a fully connected layer, or affine layer. In CNNs, we do not do that. Instead, we use convolutions over the input layer to compute the output. This results in local connections, where each region of the input is connected to a neuron in the output. Thus, CNN can preserve the spatial structure of the input matrix (e.g., image) or sequence (e.g., text).

Recurrent Neural Networks (RNN) have shown promising effectiveness on processing sequencial data. RNNs have been applied to image generation [20], sequence-to-sequence learning [82], language modeling [81] and many NLP tasks [87]. The key to why RNNs can learn a good representation for sequential data is that every time an input is fed to the RNN cell, the RNN cell will compute and update its hidden state which will be fed back into the model the next time a new input is fed. Of all RNN structures proposed, long short-term memory networks (LSTM) are one of the most popular types introduced by Hochreiter et al. [27]. In LSTM, a sigmoid layer called the "forget gate" is employed to avoid useless history information, which is the origin of its name.

Recently, as neural network models have achieved success in multiple areas such as computer vision and speech recognition, they have been widely applied in natural language processing, too. The advantages of applying neural network models to natural language processing problems are:

- Neural network models can represent words in low-dimensional semantic space. In this semantic space, the distance of word vectors can represent the similarity between words.
- There are thousands of variations in neural networks, which can model complex natural language knowledge flexibly.
- The neural network itself is hierarchical, making it easy to leverage hierarchical semantic information in a sentence or document.

Tasks	Text 1	Text 2	Objective
Paraphrase Indentification	string 1	string 2	C
Textual Entailment	text	hypothesis	C
Question Answering	question	answer	C/R
Conversation	dialog	response	C/R
Information Retrieval	query	document	R

Table 2.1: Typical text matching tasks (C: classification; R: ranking)

- Hardware developments, especially the development of GPUs for parallel training of neural network models, greatly boost the ability to learn from a huge amount of data in real life and make it possible to train complex models within days or even hours.

In summary, according to previous work on neural network models on natural language processing tasks such as POS tagging, semantic analysis, sentiment analysis, and relation classification, CNN-based networks tend to learn local semantics through the convolutional layer, making it easier to model hierarchical semantic information. In contrast, RNN based networks tend to learn sequential information by compressing the history text into a vector.

2.3.2 Text Matching and Information Retrieval

Natural language processing (NLP) tasks can be divided into several groups according to the input and output type: sentence classification, sentence pair classification (ranking), sequence labeling, and sequence-to-sequence generation [82]. Each group has various sub-problems. For example, in sentence pair classification (ranking), which is also known as text matching, we have subtasks such as question answering [66, 75], paraphrase identification [80], and sentence similarity [24]. Correspondingly, there have been a series of models for modeling sentence pair similarity or relevance [75]. Detailed information for text matching tasks is shown in Table 2.1.

According to Lan et al. [36], there are two kinds of neural text matching models in terms of the model architecture: sentence encoding models (SEM) and sentence pair interaction models (SPIM). SEM models come from Siamese architecture [9] and vector representations of individual sentences are learned, and then the semantic relationship between sentences based on vector distance is calculated. SPIM uses some types of word alignment mechanisms (e.g., attention) and then aggregates inter-sentence interactions. Compared to SEM, SPIM directly models the matching signals and intersects the two texts

as early as possible. This has been shown to be a stable advantage and has become more popular in text matching. Currently, the published popular SEMs are DSSM, CDSSM, ARC-I, CNTN, and so on. The published popular SPIMs are MVLSTM, DeepMatch, ARC-II, MatchPyramid, Match-SRNN, DRMM, KNRM, among others. In our experiments, we try both kinds of network and show the performance difference.

As a sub-problem of text matching, the current neural approaches for information retrieval can also be divided into two types: representation-based approaches [28, 77, 75] and interaction-based approaches [21, 89, 55]. The early attempts on neural IR mainly focus on representation-based modeling between query and document, such as DSSM [28] and C-DSSM [77]. DSSM [28] is early popular NN architecture for Web search that maps word sequence to character-level trigrams by using a word hashing layer, and then feeds the dense hashed features to a multi-layer perceptron (MLP) for similarity learning. C-DSSM [77] extends this idea by replacing the MLP in DSSM with a convolutional neural network-based (CNN) layer to capture local contextual signals from neighboring character trigrams. SM-CNN can be viewed as a hybrid approach with the main component of a convolutional layer for learning discriminative representations of query and document and a feature layer that exploits hand-crafted features.

In contrast, interaction-based approaches [21, 89, 55] model the similarity matrix of word pairs from the query and document directly. The similarity matrix is usually computed through word embeddings, such as *word2vec* [53], which solves the sparsity issue of count-based approaches. The DRMM approach [21] introduces a pyramid pooling technique to convert the similarity matrix to histogram representations, on top of which a term gating network aggregates weighted matching signals from different query terms. Inspired by DRMM, Xiong et al. [89] propose K-NRM that introduces a differentiable kernel-based pooling technique to capture matching signals at different strength levels. The DUET model [55] combines the representation-based and interaction-based models and proposes an idea with both a global component for the semantic match and a distributed component for the exact match. However, there are still problems to be solved in current neural IR research.

- Various lengths [22]: the matched document will be either very long or very short, while most deep learning based models are good at dealing with short documents. For a long document, deep learning based methods fail because: 1) it takes a long time and a large amount of resources for deep learning based methods to train and test; 2) exact matching is still the most effective factor in long document retrieval, but most deep learning based methods put too much weight on semantic matching.

- External knowledge: in recent NLP research, it has been shown that the pretrained language models, especially the contextual embeddings [64], greatly boost the effectiveness of the down-stream NLP task. When pretrained embeddings or external knowledge are employed into the information retrieval area, care should be taken to make sure the representations are suitable for transferring into this domain [91].

2.3.3 Neural Information Retrieval Models

As introduced above, MatchZoo contains the implementations of 11 neural network models. The basic information for each model in MatchZoo is listed in Table 2.2. As mentioned in Section 2, existing deep text matching models can be divided into composition-focused methods and interaction-focused methods. Composition-focused methods compose each sentence into one embedding and then measure the similarity between the two embeddings. In interaction-focused methods, two sentences meet before their own high-level representations mature, which can capture complex matching patterns. In our experiments, DSSM and CDSSM are representives of composition-focused methods, while DUET, KNRM and DRMM are representives of interaction-focused methods. Due to space constraints, for more details on each NN model, we refer the readers to the original papers. We will present a brief introduction to each model's main features and give some analysis in this section.

DSSM

DSSM is the first neural semantic matching model applied to Web search. It splits a sentence into letter-trigrams and combines the query and document representation into a similarity matrix through dot product and softmax layers. It claims state-of-the-art performance on the search log dataset. However, as pointed out by later researchers such as Guo et al. [21] and Pang et al. [61], DSSM suffers from the following problems:

- DSSM needs a huge amount of data for training. So, it is difficult to obtain a well-trained model on a small dataset in another domain. Thus, most researchers have to directly use the released models (trained on raw large click-through dataset described in the paper). However, the weakness of leveraging domain-specific matching signals without fine-tuning the model will be magnified as a result.
- A character-based model can address the matching of words that share similar subwords (e.g., architecture and architect). However, it still cannot leverage the matching of words that share the meaning, but have totally different character composition

Model	Task	Dataset	NN Architecture		
			Encoding	**Hidden**	**Combination**
DSSM (2013)	Web search	clickthrough data	Word Hashing of Letter-Trigram	MLP	Dot + softmax
CDSSM (2014)	Web search	clickthrough data	Word Hashing of Letter-Trigram	Conv1D + MLP	Dot + softmax + Max Pooling
ARC-I (2015)	Semantic matching	Reuters, Weibo	Word Embedding	Conv1D	Concatenation
ARC-II (2015)	Semantic matching	Reuters, Weibo	Word Embedding	Conv1D + Conv2D + MaxPooling2D	Concatenation
MV-LSTM (2015)	Semantic matching	Yahoo! Answers	Word Embedding	BiLSTM	Cosine / Bilinear / Tensor + k-Max Pooling
DRMM (2016)	Ad-hoc Retrieval	Robust04 and ClueWeb09B	Query: Word Embedding; Doc: local interaction+matching histogram	MLP	Dot
DRMM$_{TKS}$ (2016)	Ad-hoc Retrieval	Robust04 and ClueWeb09B	Word Embedding	MLP	Cosine+k-Max Pooling
aNMM (2016)	QA	TREC QA	Query: Word Embedding; Doc: local interaction+matching bin sums	MLP	Dot
Match-Pyramid (2016)	Paraphrase Identification, Paper Citation Matching	MSRP, a large academic dataset (commercial)	Word embedding	Conv2D	Indicator / Dot / Cosine + k-max pooling
DUET (2017)	Web search	Bings search logs	LM: one-hot vector; DM: word embedding	Conv1D	LM: intersection; DM: entrywise product
K-NRM (2017)	Ad-hoc Retrieval	search logs from Sogou.com	Word embedding	Kernel pooling	Cosine

Table 2.2: Model details

(e.g., architecture and construction). It also cannot distinguish different words that share similar character composition, but express different (sometimes even opposite) meanings (e.g., angle and angel).

CDSSM

CDSSM is a CNN version of DSSM. There is one more Conv1D layer to leverage the information of letter-trigram matching, which represents local contextual features as described in Shen et al. [77]. In addition, a max pooling layer is added to collect these local contextual features and form a global feature vector. Both local and global contextual features have been shown to be effective in semantic matching for Web search.

CDSSM improves over DSSM by replacing the fully-connected layer in DSSM with a convolutional neural network (CNN) to capture local context signals from neighboring trigrams. However, CDSSM still cannot avoid the two weaknesses of DSSM discussed above.

DUET

DUET is a combined model for local and distributed representations proposed by Mitra et al. [55]. DUET is also a model for Web search. However, unlike DSSM and CDSSM, DUET applies a one-hot representation and word embeddings to leverage both exact matching and semantic match signals.

According to Mitra et al. [55], for exact matching by local models, query term matches in relevant documents are observed to be more clustered, and more localized near the beginning of the document. This also matches our intuition for query-document matching in information retrieval. As for semantic matching by the distributed model, after learning the word and phrase representations of the query and the document, the distributed model performs query-document matching by computing element-wise or Hadamard product between the embedded document matrix and the extended or broadcasted query embedding. This matching method has been shown to provide outstanding results on Web searching, with training data from clickthrough logs. However, little work after DUET follows DUET and applies the element-wise product as the intersection method. So, it is unknown whether this module has an effective contribution to document matching.

Interestingly, the distributed model in DUET shows better effectiveness than the local model, which means the distributed model can somewhat leverage the exact matching

signals. On the other hand, the local model can reach the effectiveness of the previous non-neural baselines and neural baselines, meaning exact matching is still the most important matching signal in Web search.

DRMM

The Deep Relevance Matching Model (DRMM) [21] is an interaction-based approach that converts the query-document similarity matrix to a histogram representation, on top of which a term gating network is introduced to aggregate matching signals from different query terms with term weightings being incorporated. In other words, DRMM contains three major components, the matching histogram mapping, a feed forward matching network, and a term gating network.

DRMM was shown to outperform strong IR baselines and other recent deep learning methods. In the DRMM paper, the authors pointed out that many existing deep matching models are designed for semantic matching, and thus may not be appropriate for ad-hoc retrieval. The authors argued that DRMM was designed for ad-hoc retrieval and could significantly outperform traditional retrieval models (BM25, QL), as well as state-of-the-art deep matching models.

KNRM

KNRM [89] is an end-to-end neural ranking model that introduces a differentiable kernel-based layer to capture multi-level granularities of soft-matching signals from the input similarity matrix. The novel part of KNRM is its kernel pooling method. First, soft matches from query-document word interactions are used to rank the documents and the relevance preference encoding are achieved by the kernels. Kernel pooling is a novel method to convert word-word interactions in the similarity matrix to query-document ranking features. Then k-max pooling is applied to transform the features into a K-dimensional feature vector for each word in the query. According to Xiong et al. [89], KNRM significantly beats the effectiveness of traditional ranking methods and neural ranking methods like DSSM, CDSSM, and DRMM. However, this advantage is only shown using the search logs from Sogou.com, which is not a publicly available dataset. To the best of this thesis's author's knowledge, the performance of KNRM on other datasets, specially for long documents, is still unknown.

BERT

BERT is a pretrained language model proposed by Devlin et al. [11] that has been proven effective in many natural language processing tasks [92, 57], ranging from sentence classification to question answering to sequence labeling. The effectiveness of BERT mainly results from the heavy use of pretraining for the language model. BERT is pretrained on the concatenation of the BooksCorpus (800M words) and English Wikipedia (2,500M words). Another major contribution of BERT from the model perspective is that BERT is novel deep bidirectional architecture, allowing the same pre-trained model to successfully tackle a broad set of natural language processing tasks.

2.4 Evaluation Metrics

There are two evaluation perspectives for text matching tasks: classification and ranking. The difference is how we build the datasets and what the final target is. So, in general, all text matching models can be treated as both a classification model and a ranking model. The datasets are built for specific purposes so they have their own evaluation metrics. Even the models themselves can be used for both classification or ranking, standard benchmark datasets define a specific evaluation methodology. Table 2.3 shows the popular datasets in the text matching task and their evaluation metrics:

Dataset	Type	Task	Evaluation Metrics
SNLI	C	Natural Language Inference	Accuracy
Multi-NLI	C	Natural Language Inference	Accuracy
Quora	C	Paraphrase Identification	Accuracy
Twitter-URL	C	Paraphrase Identification	Precision, Recall, F1
PIT-2015	C	Paraphrase Identification	Precision, Recall, F1
STS-2014	C	Semantic Textual Similarity	Pearson's correlation
TrecQA	R	Question Answering	MAP, MRR
WikiQA	R	Question Answering	MAP, MRR
Robust04	R	Information retrieval	MAP, P@20, NDCG@20
Microblog	R	Information retrieval	MAP, P@30

Table 2.3: Datasets and their evaluation metrics (C: classification; R: ranking)

For the information retrieval task, we usually treat it as a ranking problem because we care about the entire rank list returned by the retrieval model or system. Thus, MAP

(short for Mean Average Precision), P@K (short for precision at K), and NDCG@K (short for Normalized Discounted Cumulative Gain at K [31]) are used for evaluation.

Precision compute the fraction of relevant documents retrieved for a query q with respect to the total number of documents in the retrieved set R_q:

$$Precision_q = \frac{\sum_{\langle i,d\rangle \in R_q} rel_q(d)}{|R_q|} \tag{2.1}$$

The definition of average precision is as follows:

$$AP_q = \frac{\sum_{\langle i,d\rangle \in R_q} Precision_{q,i} \times rel_q(d)}{\sum_{d \in D} rel_q(d)} \tag{2.2}$$

where, $Precision_{q,i}$ is the precision computed at rank i for the query q.

NDCG [31] calculates the Discounted Cumulative Gain (DCG) over the relevance labels $rel(\mathbf{r}[i])$ for each document in the top K of a ranking and then is normalized by the maximal DCG possible for a query (iDCG). The definition of NDCG is shown as follows:

$$\mathrm{N}DCG_q = \frac{DCG_q}{IDCG_q} \tag{2.3}$$

where $IDCG_q$ is the ideal DCG (iDCG), which is the DCG for the perfect ranking and the DCG_q is defined as follows:

$$\mathrm{D}CG_q = \sum_{\langle i,d\rangle \in R_q} \frac{2^{rel_q(d)} - 1}{\log_2(i+1)} \tag{2.4}$$

The ideal DCG ($IDCG_q$) is computed the same way but by assuming an ideal rank order for the documents up to rank K. Note both P@K and MAP metrics assume that the relevance labels are binaIniformationry but NDCG@K does not have such an assumption.

In summary, IR metrics focus on rank-based evaluation of retrieved results using ground truth information, as determined by manual judgments or implicit feedback from user behaviour data. However, this is also one of the drawbacks of IR evaluation. Humans cannot manually label all possible relevant documents, and all documents not labeled are treated as irrelevant, which might not be always true.

Chapter 3

End-to-End Information Retrieval Architecture

3.1 Architecture

The integration between traditional information retrieval methods and reranking models is straightforward. After building indexes with a search engine, we search the candidates given a query with traditional IR methods. Different from the Java native implementation, we implement this retrieval-rerank pipeline using Python API with Pyjnius,[1] a Python library for accessing Java classes. The candidate output are connected with the input of reranking models directly. The framework of the retrieval-rerank system is shown in Figure 3.1. First, we select K1 candidate documents using a traditional IR baseline. Then we rerank the K1 documents using the reranking models and aggreate the matching scores from the neural models and the relevance scores from the IR baselines using a linear interpolation (See Section 3.6 for details). Finally, we select K2 candidate with the interpolated scores as the output. In our experiments, we choose K1=K2=1000 for both the Microblog and the Robust04 datasets.

For the retrieval models, we use the implementation from the Anserini toolkit introduced in the Chapter 1. We use four IR methods (QL, QL+RM3, BM25, BM25+RM3) and apply neural reranking models in the MatchZoo toolkit introduced in the Chapter 1. In addition, in this thesis, we also integrate MP-HCNN and BERT with the open-source Anserini toolkit to create an end-to-end information retrieval system. We will introduce the details of Anserini, MatchZoo, MP-HCNN and BERT in the following sections.

[1] https://pyjnius.readthedocs.io/en/latest/

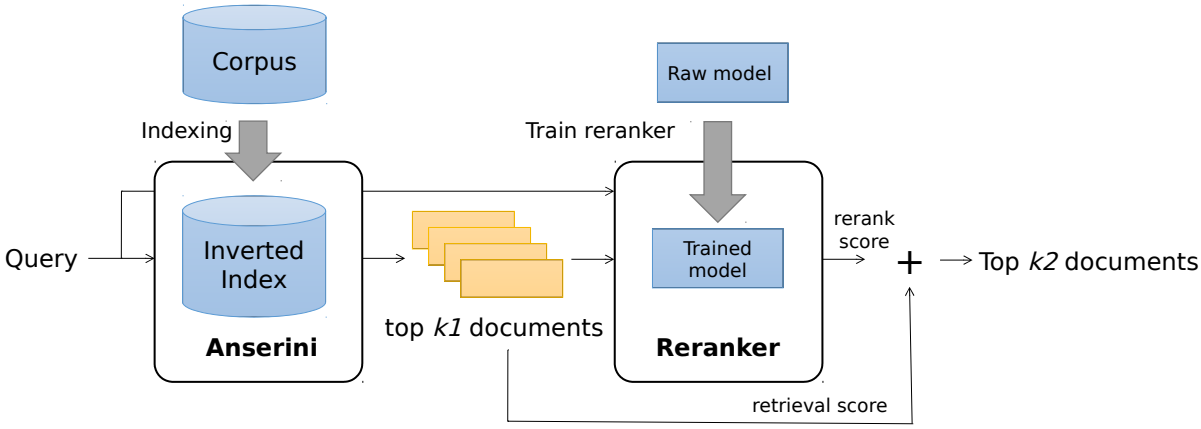

Figure 3.1: The Architecture of the Retrieval-rerank framework.

Note that in our framework, only the indexing part is written and executed using Java. Once after indexing, all other modules are written in Python and executed in a list of Python scripts, so we achieve retrieval and reranking in one shell script. All neural network models in MatchZoo are implemented using Keras.[2] Similar to MatchZoo, MP-HCNN is implemented in Keras. We use almost the same interface as the Anserini+MatchZoo and tweak the data file format to match the input of MP-HCNN and BERT model.

3.2 Anserini

It is known that Lucene can handle heterogeneous web collections at scale. However, Lucene lacks systematic support for evaluation over standard test collections, which is exactly the intuition behind the development Anserini,[3] an open source information retrieval system for reproducible information retrieval research built on Lucene [90]. In addition, Anserini is also aimed at efficiently indexing large web collections, and supporting low-latency query evaluation to facilitate rapid experimentation.

Anserini provides wrappers and extensions on top of core Lucene libraries that allow researchers to use more intuitive APIs to accomplish common research tasks. As of January 1st, 2019, Anserini has included 13 standard IR experiments with regression tests for

[2] https://keras.io/
[3] http://anserini.io/

replicability across various domains such as news, microblogs, and Wikipedia. On the other hand, Anserini also provides some features such as multi-threaded indexing and relevance feedback algorithms like RM3 and axiomatic reranking. These reranking methods actually work well in most IR retrieval tasks.

3.3 MatchZoo

MatchZoo[4] is a text matching toolkit that aims at facilitating the design, comparison and sharing of deep text matching models. Figure 3.2 shows the three components of MatchZoo toolkit:

- Data processing module: this module contains general text processing functions such as word tokenization, low-frequency word removal, word stemming, and so on. Then, this module transfers different types of text matching task into the same format. In addition, this module provides different data generator for training including the single document generator, document pair generator, and document list generator. Different data generators are suitable for different text processing tasks such as question answering, dialogue system, and text ranking.
- Model construction module: this module contains widely used layers in deep learning such as a convolutional layer, pooling layer, and fully connection layer. Also, this module designs specific layer for text matching task such as dynamic k-max pooling layer and term gating layer.
- Train and evaluation module: this module provides target functions and evaluation metrics for classification, regression and ranking problems. Users can choose the target function according to the task.

MatchZoo has been under active development since it was built in 2017. In the released version 1.0, MatchZoo contains 11 deep matching methods for text matching: DRMM, MatchPyramid, MV-LSTM, aNMM, DUET, ARC-I, ARC-II, DSSM, CDSSM, CONV-KNRM, and DUET. There are models under development such as Match-SRNN and DeepRank, which might be released in the future version. Initially, these models were proposed for different subtasks of text matching: paraphrase identification, textual entailment, question answering, information retrieval.

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

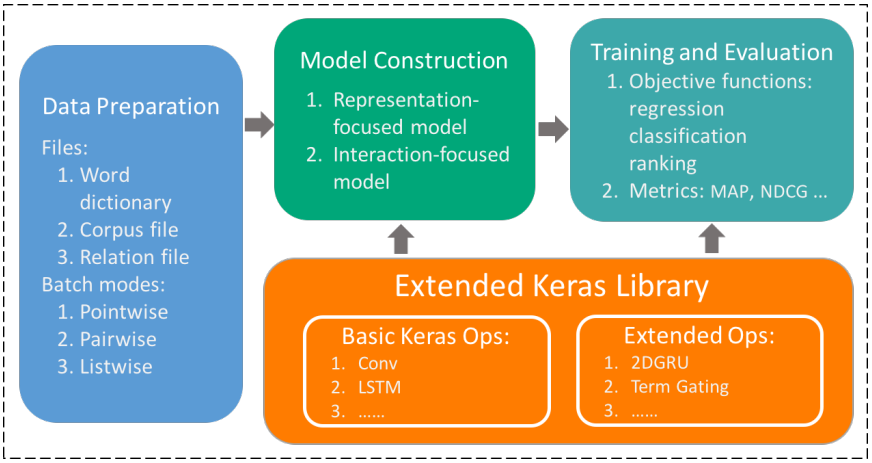

Figure 3.2: The Architecture of the MatchZoo toolkit.

3.4 MP-HCNN

MP-HCNN (Multi-Perspective Hierarchical Convolutional Neural Network) [71] is a novel neural ranking model specifically designed for *ad hoc* retrieval over short social media posts, which is co-authored by the author of this thesis. Basically MP-HCNN is composed of two parts: hierarchical convolutional layers and similarity aggregation.

3.4.1 Hierarchical Convolutional Layers

Given a query q and a document d, the textual matching component aims to learn a relevance score $f(q, d)$ using the query terms $\{w_1^q, w_2^q, ..., w_n^q\}$ and document terms $\{w_1^d, w_2^d, ..., w_m^d\}$, where n and m are the number of terms in q and d, respectively. To be clear, "document" can either refer to a social media post or an URL, and "term" refers to either words or character trigrams. One important novel aspect of our model is relevance modeling from multiple perspectives, and our architecture exhibits symmetry in the word- and character-level modeling. Thus, for expository convenience, we use "document" and "term" in the generic sense above. We first employ an embedding layer to convert each term into a L-dimensional vector representation, generating a matrix representation for the query $\mathbf{Q}$ and document $\mathbf{D}$, where $\mathbf{Q} \in \mathbb{R}^{n \times L}$ and $\mathbf{D} \in \mathbb{R}^{m \times L}$. In the following, we introduce our representation learning method with hierarchical convolutional neural networks.

A convolutional layer applies convolutional filters to the text, which is represented by an embedding matrix $\mathbf{M}$ ($\mathbf{Q}$ or $\mathbf{D}$). Each filter is moved through the input embedding

incrementally as a sliding window (with window size k) to capture the compositional representation of k neighboring terms. Assuming a convolution layer has F filters, then this CNN layer produces output matrix $\mathbf{M}^o \in \mathbb{R}^{\|M\| \times F}$ with $O(F \times k \times L)$ parameters.

We then stack multiple convolutional layers in a hierarchical manner to obtain higher-level k-gram representations. For notational simplicity, we drop the superscript o from all output matrices and add a superscript h to denote the output of the h-th convolutional layer. Stacking N CNN layers therefore corresponds to obtaining the output matrix of the h-th layer $\mathbf{M}^h \in \mathbb{R}^{\|M\| \times F^h}$ via:

$$\mathbf{M}^h = \text{CNN}^h(\mathbf{M}^{h-1}), h = 1, \ldots, N,$$

where $\mathbf{M}^{h-1}$ is the output matrix of the $(h-1)$-th convolutional layer. Note that $\mathbf{M}^0 = \mathbf{M}$ denotes the input matrix ($\mathbf{Q}$ or $\mathbf{D}$) obtained directly from the word embedding layer, and the parameters of each CNN layer are shared by the query and document inputs.

Intuitively, consecutive convolutional layers allow us to obtain higher-level abstractions of the text, starting from character-level or word-level to phrase-level and eventually to sentence-level. A single CNN layer is able to capture the k-gram semantics from the input embeddings, and two CNN layers together allow us to expand the context window to up to $2k - 1$ terms. Generally speaking, the deeper the convolutional layers, the wider the context considered for relevance matching. Empirically, we found that a filter size $k = 2$ for word-level inputs and $k = 4$ for character-level inputs work well. The number of convolutional layers N was set to 4. This setting is reasonable as it enables us to gradually learn the representations of word-level and character-level n-grams of up to $O(N \times k)$ length. Since most queries and documents in the social media domain are either shorter or not much longer than this length, we can regard the output from the last CNN layer as an approximation of sentence representations.

An alternative to our deep hierarchical design is a *wide* architecture, which reduces the depth but expands the width of the network by concatenating multiple convolutional layers with different filter sizes k in parallel to learn variable-sized phrase representations. However, such a design will require quadratically more parameters and be more difficult to learn than our approach. More specifically, our deep model comprises $O(N \times F \times kL)$ parameters with N CNN layers, while a wide architecture with the same representation window will need $O(F \times (kL + 2kL + ... + NkL)) = O(N^2 \times F \times kL)$ parameters. The reduced parameters in our approach mainly come from representation reuse at each CNN layer, which also generalizes the learning process by sharing representations between successive layers.

3.4.2 Similarity Aggregation

To measure the similarity between the query and the document, we match the query with the document at each convolutional layer by taking the dot product between the query representation matrix $\mathbf{M}_q$ and the document representation matrix $\mathbf{M}_d$:

$$\mathbf{S} = \mathbf{M}_q {\mathbf{M}_d}^T, \mathbf{S} \in \mathbb{R}^{n \times m},$$
$$\tilde{\mathbf{S}}_{i,j} = softmax(\mathbf{S}_{i,j}) = \frac{e^{\mathbf{S}_{i,j}}}{\sum_{k=1}^{m} e^{\mathbf{S}_{i,k}}}$$

where $\mathbf{S}_{i,j}$ can be considered the similarity score by matching the query phrase vector $\mathbf{M}_q[i]$ with the document phrase vector $\mathbf{M}_d[j]$. Since the query and document share the same convolutional layers, similar phrases will be placed closer together in a high-dimensional embedding space and their product will produce larger scores. The similarity matrix $\mathbf{S}$ is further normalized to $\tilde{\mathbf{S}}$ with range $[0, 1]$ through a document-side *softmax* function.

We then apply *max* and *mean* pooling to the similarity matrix to obtain discriminative feature vectors:

$$Max(\tilde{\mathbf{S}}) = [max(\tilde{\mathbf{S}}_{1,:}), ..., max(\tilde{\mathbf{S}}_{n,:})], Max(\tilde{\mathbf{S}}) \in \mathbb{R}^n;$$
$$Mean(\tilde{\mathbf{S}}) = [mean(\tilde{\mathbf{S}}_{1,:}), ..., mean(\tilde{\mathbf{S}}_{n,:})], Mean(\tilde{\mathbf{S}}) \in \mathbb{R}^n;$$

Each score generated from pooling can be viewed as one piece of matching evidence for a specific query term or phrase to the document, and its value denotes the importance of the relevance signal.

To measure the relative importance of different query terms and phrases, we inject external weights as prior information by multiplying the score after pooling with the weight of that specific query term or phrase. These are provided as feature inputs to the subsequent learning-to-rank layer, denoted by Φ:

$$\Phi = \{weights(q) \odot Max(\tilde{\mathbf{S}}), weights(q) \odot Mean(\tilde{\mathbf{S}})\},$$

where $\odot$ is an element-wise product between the weights of query terms or phrases with the pooling scores and $weights(q)[i]$ denotes the weight of the i-th term or phrase in the query. We choose inverse document frequency (IDF) as our weighting measure; a higher IDF weight implies rarer occurrence in the collection and thus larger discriminative power. This weighting method also reduces the impact of high matching scores from common words like stopwords.

Our similarity measurement layer has two important properties. First, all the layers, including matching, softmax, pooling, and weights, have no learnable parameters. Second, the parameter-free nature enables our model to be more interpretable and to be more robust from overfitting. By matching query phrases with document phrases in a joint manner, we can easily track which matches contribute more to the final prediction. This greatly increases the interpretability of our model, an important benefit as this issue has become a prevalent concern given the complexity of neural models for IR and NLP applications [41].

Given similarity features learned from word-level Φ^w and character-level Φ^c, we employ a multi-layer perceptron (MLP) with a ReLU activation in between as our learning-to-rank module:

$$o = \mathit{softmax}(\mathbf{MLP}(\Phi^w \sqcup \Phi^c))$$

where $\sqcup$ is a concatenation operation and the *softmax* function normalizes the final prediction to a similarity vector o with its values between 0 and 1. The training goal is to minimize the negative log likelihood loss L summed over all samples (o_i, y_i): $L = -\sum_{(o_i,y_i)} \log o_i[y_i]$, where y_i is the annotation label of sample i.

Since MP-HCNN is claimed to be able to leverage the domain-specific information in the microblog matching task, it has been shown to reach state-of-the-art performce on the Microblog dataset. We will apply MP-HCNN to experiments on the Microblog dataset. Instead of trying it on a single QL baseline, we will show we can achieve even better performance with the stronger RM3 baseline.

3.5 BERT

We use the base version of the pretrain BERT model of the PyTorch implementation.[5] We use BERT as a sentence pair classification model as demostrated in Figure 3.4. In our experiments, sentence 1 is the query and sentence 2 is the document. Then we fine tune the BERT model in our datasets and we extract the probability of the class label as the relevance of the two sentences during the inference stage.

[5] https://github.com/huggingface/pytorch-pretrained-BERT

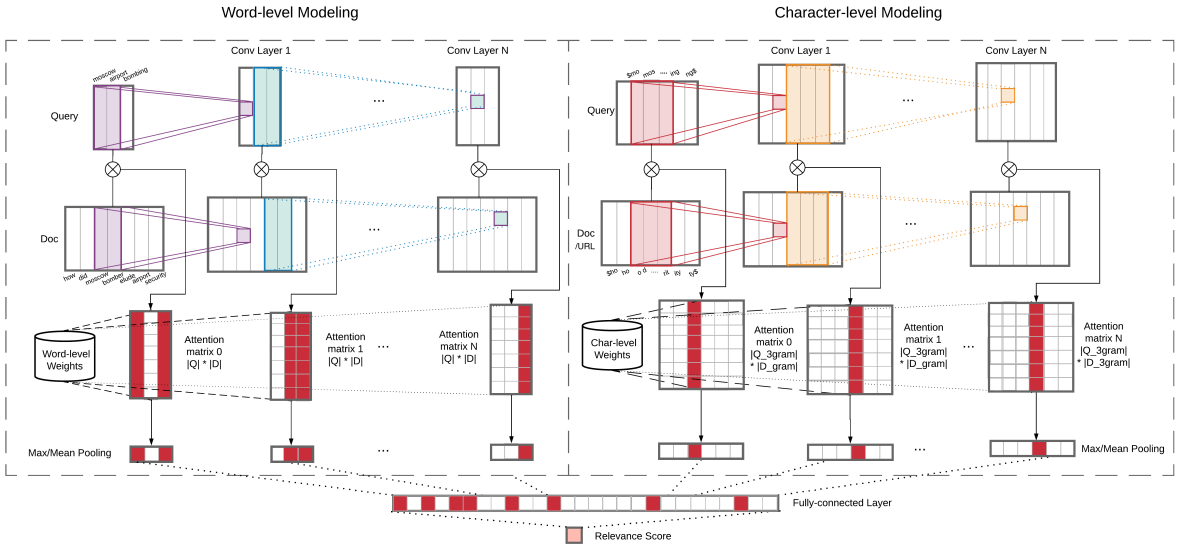

Figure 3.3: Overview of our Multi-Perspective Hierarchical Convolutional Neural Network (MP-HCNN), which consists of two parallel components for word-level and character-level modeling between queries, social media posts, and URLs. The two parallel components share the same architecture (with different parameters), which comprises hierarchical convolutional layers for representation learning and a semantic similarity layer for multi-level matching. Finally, all relevance signals are integrated using a fully-connected layer to produce the final relevance score.

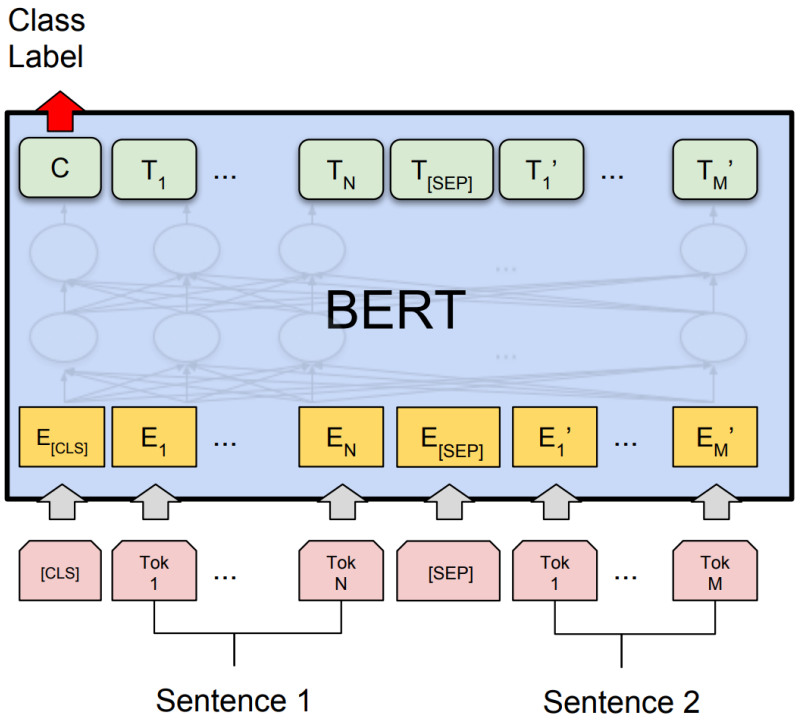

Figure 3.4: The Architecture of the BERT model for sentence pair classification.

3.6 Interpolation between Retrieval Models and Reranking Models

The above described NN models are good at modeling semantic similarities between queries and documents, yet we know exact matching signals can also substantially contribute to relevance ranking [21, 55]. Thus, we adopt a simple linear interpolation method to combine the evidence from above NN models with the classical retrieval model (QL, BM25, RM3) method below:

$$\text{rel}(q, d) = \lambda * \text{Reranker}(q, d) + (1 - \lambda) * \text{Retriever}(q, d) \tag{3.1}$$

The best hyper-parameter α is tuned on the validation set considering all values in tenth increments from 0 to 1, and applied to the test set.

Chapter 4

Evaluation

4.1 Experimental Setup

We design our experiments as follows. All experiments are performed using Anserini (the backend Lucene version: 7.6), MatchZoo of the released version 1.0 and the released code of MP-HCNN and BERT. Word vectors are pre-trained on the Google news corpus (3 billion running words).[1] All the hyper-parameters in the experiments can be viewed in Table 4.1 and Table 4.4. In our experiments, we select four information retrieval methods from Anserini (QL, QL+RM3, BM25, BM25+RM3), five models for information retrieval only in MatchZoo (DSSM, CDSSM, DUET, KNRM, DRMM), MP-HCNN and BERT, and apply them into two datasets: Microblog and Robust04. In order to make our results comparable and consistent to the previous results, we report the MAP, MRR, P@20, P@30, and NDCG@20 scores as the evaluation metrics for the IR baselines. For the neural reranking results, we report MAP, P@20 and NDCG@20 scores for the Robust04 dataset and report MAP and P@30 scores for the Microblog dataset.

Generally, there are two ways to generate the training data. First, we can train the neural network models using relevant judgment from 'qrels' files, which are the results submitted by all participants in TREC Robust IR 2004. We test the neural network models on the candidates generated by different traditional information retrieval methods on the test topics. The second way is to use the base run from the retrieval models as the training set, which is the same process of test set generation. We did not find a significant difference between these two methods in our experiments. So we applied the

[1] https://github.com/mmihaltz/word2vec-GoogleNews-vectors

second method, which we believe is the more common method in the neural IR community, to generate the training set.

Model	Parameter	Value
Unsupervised methods		
QL	mu	1000
BM25	k1	0.9
	b	0.4
Shared MatchZoo parameters		
MatchZoo	Embedding size	300
	text1_maxlen	10
	text2_maxlen	500/68
	Optimizer	Adam
	Learning rate	0.001
	Batch size	100
Specific models		
CDSSM	kernel_count	50
	kernel_size	3
	hidden_sizes	[10]
	dropout_rate	0.1
DSSM	hidden_sizes	[300]
	dropout_rate	0.1
DUET	lm_kernel_count	32
	lm_hidden_sizes	[30]
	dm_kernel_count	32
	dm_kernel_size	3
KNRM	kernel_num	21
	sigma	0.1
	exact_sigma	0.001
DRMM	Histogram size	20
	hidden_sizes	[30, 10, 1]
MP-HCNN	# of convolutional layers	5
BERT	Learning rate	1e-5
	Epochs	5
	Batch size	16

Table 4.1: Hyper-parameters of all models except RM3 in the experiments

4.2 Datasets

4.2.1 Microblog

We conduct experiments on two corpora: Microblog and Robust04. The Microblog corpus is provided by TREC Microblog Track in 2011 and 2013,[2] which is the live stream of tweets. The corpus statistics are shown in Table 4.2. We train the NN models using the golden labeled data from 'run' files from MB2011, MB2012, and MB2013 and test the NN models on the 'run' file generated by QL on MB2014. Each year' data contains the queries from the users and the target to find the most recent and relavant tweets in the corpus [58, 79, 46, 48]. A sample query is shown belew:

- ID: MB051
- Query: British Government cuts
- Querytime: Tue Feb 08 23:56:46 +0000 2011
- Querytweettime: 35124912364457984

We sample 10% of the training queries to form the validation set. In practice, we did not observe significant differences with different validation set selection. The relevance judgments are made in a three-point scale ("not relevant", "relevant", "highly relevant"), but in this work, we treat both higher grades as relevant. For the RM3 baseline, we use the hyperparameters fbTerms=10, fbDocs=10, and originalQueryWeight=0.5.

To enable fair comparisons across the NN models, we adopt the same tuning strategies and report the best performance. we use the 300-dimension word vectors[3] pre-trained from the Common Crawl Dataset with 840B tokens for word-based models (e.g., DRMM) with GloVe [63] and standard one-hot vector representations for character-based models (e.g., DSSM). The statistics of vocabulary sizes for each dataset and out-of-vocabulary (OOV) words can be found in Table 4.2. Out-of-vocabulary words' vectors are sampled from the uniform distribution $U(-0.25, 0.25)$. The same padding strategy is used across the four datasets by setting to the largest query/document length, where each query is padded to 10 words and 51 characters and each document is padded to 68 words and 140 characters, respectively. For each model, we select the best optimizer from the set [SGD, Adam, Adadelta] and the best initial learning rate from the set [0.1, 0.01, 0.001, 0.0001].

[2] https://trec.nist.gov/data/microblog.html
[3] https://nlp.stanford.edu/projects/glove/

Test Set	2011	2012	2013	2014
# query topics	49	60	60	55
# query-doc pairs	49,000	60,000	60,000	55,000
% relevant docs	4.9	8.6	7.4	16.4
% URLs	51.2	50.9	50.0	50.2
% hashtags	17.1	16.1	17.0	17.7

Table 4.2: Statistics of the TREC Microblog Track datasets.

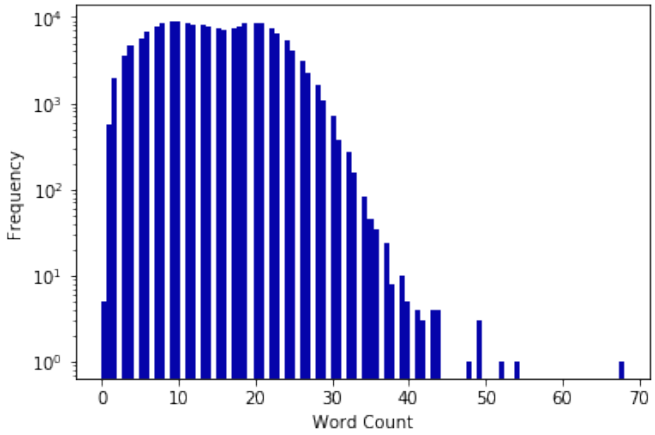

Figure 4.1: Distribution of documents' length in Microblog dataset

Figure 4.1 shows the documents' length distribution in tokens of the Microblog corpus. We can see that the lengths of most tweets are in the range [0, 40], making it easy for neural text matching models to handle the varied length problem in the Microblog dataset.

4.2.2 Robust04

The Robust04 corpus is provided by TREC Robust Track in 2004, which is the set of documents on both TREC Disks 4 and 5 minus the Congressional Record on disk 4. The corpus statistics are shown in Table 4.3. There are 250 topics in the query set and one of the queries was subsequently dropped due to having no relevant documents.

Figure 4.2 shows the document length distribution in tokens of the Robust04 corpus. We can see that the lengths of news documents are highly biased. It makes it hard for neural text matching models to headle the varied length problem in Robust04 dataset. To

Source	# Docs	Size (MB)
Financial Times	210,158	564
Federal Register 94	55,630	395
FBIS, disk 5	130,471	470
LA Times	131,896	475
Total	528,155	1904

Table 4.3: Statistics of the Robust04 corpus

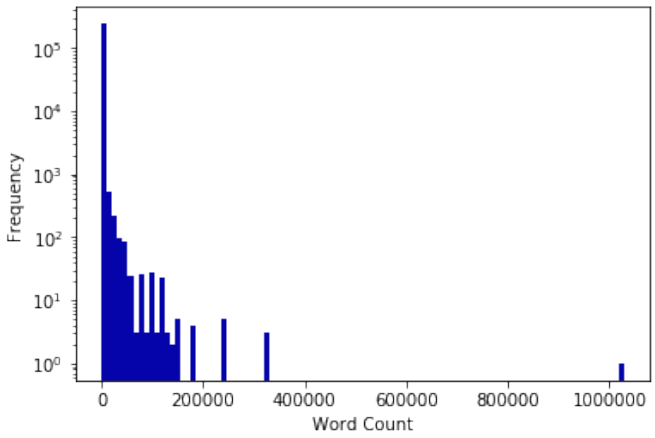

Figure 4.2: Distribution of documents' length in the Robust04 dataset

Parameter	1	2	3	4	5
fbTerms	45	45	50	50	45
fbDocs	8	6	10	12	8
originalQueryWeight	0.2	0.3	0.2	0.2	0.2

Table 4.4: Hyper-parameters of RM3 method on Microblog dataset and five data splits of Robust04 in the experiments

deal with the long text problem, most text matching models in our experiments (except DRMM) truncates the first K (text2_maxlen) words and take it as input. This truncation method often loses information in a global view and also might cause mismatch if the matching terms do not appear in the first K words. In our experiments, to ensure that neural text matching models can capture information and are trained within a reasonable time cost, we set K=500 for all models. Note DRMM computes the similarity between each term in the query and each term in the documents. Then DRMM compresses this large similarity matrix in the small matrix using a histogram mapping trick. Each similarity score becomes a count in the histogram vector. This might be the reason that DRMM performs well on the long document text matching with varied lengths.

Following recent work on the Robust04 dataset [85, 21, 51], we used 5-fold cross-validation on 250 topics in the Robust04 corpus and split them into train/dev/test sets by the ratio 3:1:1 following Lin [45] (3 folds for training, one fold for dev and one fold for test).[4] The tuned hyper-parameters for each split in the Robust04 dataset are shown in Table 4.4. We use the pretrain embedding from the Google news corpus.[5] We also try to use the embedding from other two sources: Glove[6] and pretrained from the Robust04 raw corpus. However, in both ways in our experiements, the performance cannot beat those with the pretrain embedding from the Google news corpus. The results of the reranking are in Section 4.3.

	QL	QL+RM3	BM25	BM25+RM3
AP	0.4181	0.4676^{+}	0.3931	0.4374^{+}
MRR	0.8581	0.8181	0.8366	0.8282
P@20	0.6709	0.6927^{+}	0.6682	0.6827^{+}
P@30	0.6430	0.6533	0.6212	0.6442^{+}
NDCG@20	0.6385	0.6479	0.6307	0.6407

Table 4.5: Results of retrieval models on Microblog datasets. Significant improvement of RM3 with respect to the baseline retrieval model (QL/BM25) is indicated (+) (p-value $\leq$ 0.05).

	QL	QL+RM3	BM25	BM25+RM3
AP	0.2465	0.2743^{+}	0.2515	0.3033^{+}
MRR	0.6817	0.6405	0.6842	0.6523
P@20	0.3508	0.3639^{+}	0.3612	0.3973^{+}
P@30	0.3076	0.3229^{+}	0.3112	0.3493^{+}
NDCG@20	0.4109	0.4172	0.4225	0.4514^{+}

Table 4.6: Results of retrieval models on the Robust04 dataset. Significant improvement of RM3 with respect to the baseline retrieval model (QL/BM25) is indicated (+) (p-value $\leq$ 0.05).

4.3 Results

4.3.1 Retrieval Baselines

Results of the retrieval models on the Robust04 dataset are shown in Table 4.6 and results of the retrieval models on the Microblog dataset are shown in Table 4.5. Lin et al. [45], point out most neural network models for information retrieval compared with weak IR baselines. Thus, we tried RM3 in our experiments, and we can see it provides a significant improvement on the effectiveness.

Among the five evaluation metrics, MRR metric is only affected by the single highest-ranked relevant item. So MRR is appropriate to judge a system where either (a) there is

[4]The query IDs of each split can be found in https://github.com/castorini/Anserini/tree/master/src/main/resources/fine_tuning/drr_folds.

[5]https://github.com/mmihaltz/word2vec-GoogleNews-vectors

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

only one relevant result, or (b) researcher only really care about the highest-ranked one. In the two datasets in our experiments, we can see that the MRR metric may not follow the same increasing or decreasing trend as other metrics because other metrics are affected by the rank of most relevant items.

From the results in Table 4.6 and Table 4.5, we can see that QL performs better than BM25 on the Microblog dataset but worse on the Robust04 dataset. RM3 reranking achieves consistent improvement over the baseline ranking models, demonstrating its superior effectiveness. Note that RM3 requires an additional round of retrieval to select terms for query expansion, and thus is substantially slower. However, if we rerank the candidates from RM3 method using neural text matching models, the time complexity of RM3 method can be ignored since the time cost of the neural network models becomes the bottleneck.

4.3.2 Reranking

Microblog

The baseline results on Microblog datasets are shown in Table 4.7 and all of them are from Rao et al. [71] including QL and QL+RM3. Note that these results are different from Anserinit results in Table 4.7. Row 3, 4, 5, 6 are from a competitive ranking algorithm, LambdaMART [5], that won the Yahoo! Learning to Rank Challenge [4] with different features. The details of L2R and MP-HCNN are introduced by Rao et al. [71]. We can see that our QL and QL+RM3 results in Table 4.5 are higher than the QL and QL+RM3 results in Table 4.7, as well as other baselines' results. This provides a good candidate set for neural text matching models.

The results on the Microblog dataset are shown in Table 4.8. Note NN+ (NN = [DSSM, CDSSM, DUET, KNRM, DRMM]) means the interpolation result between the NN model and QL baseline. From this table, we can see that if we do not consider the BERT model which learn knowledge from external data, we can achieve the state-of-the-art results using QL+RM3 retrieval model and MP-HCNN text matching model with the interpolation method. Among all neural text matching methods in the MatchZoo toolkit, DRMM performs the best. But after interpolation, KNRM+, DRMM+, DUET+, and MP-HCNN perform at a similar level. This means that these neural methods capture similar matching information that contributes to the base retrieval model.

Comparing the results of original NN models without interpolation, we find that character-based approaches (DSSM, C-DSSM, DUET) tend to perform worse than word-based ap-

ID	Method	AP	P@30
1	**QL**	0.3924	0.6182
2	**QL+RM3**	0.4480	0.6339
3	**L2R (all)**	0.3943	0.6200
4	**L2R (text)**	0.3824	0.6091
5	**L2R (text+URL)**	0.3974	0.6206
6	**L2R (text+hashtag)**	0.3815	0.5939
7	**MP-HCNN**	0.4304	0.6297
8	**Interpolated MP-HCNN**	0.4420	0.6394

Table 4.7: Previous results on Microblog datasets by Rao et al. [71]

proaches (MatchPyramid, DRMM, K-NRM) in general. This is likely basecause that all word-based NN models use pre-trained word vectors that encode more semantic meaning than one-hot vector representations as used in character-based approaches, suggesting that the Twitter datasets are not sufficient to support the learning of character-based representations from scratch. C-DSSM suffers more than DSSM, showing that a more complicated model tend to have worse effectiveness in a data-poor setting. DUET is generally comparable to DSSM in all metrics, even though it has an additional local component to capture exact matching signals.

Comparing the methods with and without interpolation, we observe that simple interpolation with QL boosts the effectiveness of all neural baselines dramatically, showing that exact match signals are complementary to the soft match signals captured by NN methods. This observation also holds for the proposed model: interpolated MP-HCNN, although the margin of improvement is smaller due to the effectiveness of MP-HCNN alone.

Comparing all word-based NN models with interpolation, we find that MP-HCNN seems to be the most effective approach while CDSSM is the worst. Considering that the three models share many similarities at the input level (all based on an embedding-level similarity matrix), the large performance difference between DRMM/K-NRM and MatchPyramid suggests that term weighting is crucial for tweet search. Moreover, we find that the kernel-based manipulations of the input similarity matrix in KNRM do not lead to any improvement over simple histogram representations in DRMM. Overall, all the NN models perform much worse than a simple QL baseline, suggesting that these well-performing Web ranking models fail to adapt to tweet search.

Comparing the interpolation results, we observe that the performance difference between NN models still holds in most settings. For example, word-based approaches remain

RM	NRM	AP	P@30
QL	-	0.4184	0.6424
	DRMM	0.3778−	0.5588−
	KNRM	0.2984−	0.4570−
	DUET	0.2658−	0.3879−
	DSSM	0.2481−	0.3921−
	CDSSM	0.1914−	0.2655−
	MP-HCNN	0.4310	0.6362
	BERT	0.4314	0.6376
	Interpolated DRMM	0.4475+	0.6448
	Interpolated KNRM	0.4389+	0.6564
	Interpolated DUET	0.4476+	0.6606
	Interpolated CDSSM	0.4229	0.6442
	Interpolated DSSM	0.4184	0.6424
	Interpolated MP-HCNN	0.4533+	0.6610+
	Interpolated BERT	**0.4748**+	**0.6800**+
QL +RM3	-	0.4676	0.6533
	DRMM	0.4477−	0.6127−
	KNRM	0.3432−	0.5121−
	DUET	0.2713−	0.3533−
	DSSM	0.2634−	0.3836−
	CDSSM	0.1936−	0.2636−
	MP-HCNN	0.4497	0.6219
	BERT	0.4646	0.6509
	Interpolated DRMM	0.4862+	0.6703
	Interpolated KNRM	0.4848+	0.6624
	Interpolated DUET	0.4844+	0.6594
	Interpolated DSSM	0.4666	0.6539
	Interpolated CDSSM	0.4703	0.6624
	Interpolated MP-HCNN	0.4902+	0.6712
	Interpolated BERT	**0.5011**+	**0.6842**+

Table 4.8: Result of retrieval and text matching on Microblog datasets. RM: retrieval model. NRM: neural re-ranking model. Significant improvement or degradation with respect to the retrieval model is indicated (+/-) (p-value ≤ 0.05).

Model	AP	P@20	NDCG@20
QL (Guo et al.)[21]	0.253	0.369	0.415
BM25 (Guo et al.)[21]	0.255	0.370	0.418
DRMM (Guo et al.)[21]	0.279	0.382	0.431
QL (Pang et al.)[60]	0.253	0.369	0.415
BM25 (Pang et al.)[60]	0.255	0.370	0.418
MatchPyramid (Pang et al.)[60]	0.232	0.327	0.411
BM25 (Mcdonald et al.)[51]	0.238	0.354	0.425
PACRR (Mcdonald et al.)[51]	0.258	0.372	0.443
POSIT-DRMM+MV (Mcdonald et al.)[51]	0.272	0.386	0.461

Table 4.9: Previous results on the Robust04 dataset

to be slightly better than character-based models after interpolation. Note that interpolation scores can be lower than the QL-only scores because the model can learn a non-zero weight for the NN component from the training data, which is less effective than QL on the test data. The most robust models are DUET and KNRM with interpolations, which consistently or even significantly outperform the QL baseline in some datasets. This suggests that these two methods do capture relevance signals that are complementary to an exact matching method like QL. It is also worth noting that we cannot conclude one method is better than another by only comparing their interpolation performance, such as DSSM and C-DSSM. The better interpolation performance of C-DSSM merely is because its prediction scores are mostly dominated by QL scores (i.e., its interpolation weights are close to zero across all datasets). Overall, our finding is consistent in the original and interpolation setup – all existing NN models suffer in tweet search, while some are marginally effective when interpolated with the QL baseline results.

Robust04

There are many researchers working on the Robust04 dataset, and these previous results are shown in Table 4.9. Similar to the Microblog dataset, our QL and BM25 results in Table 4.6 are higher than the QL and BM25 results in Table 4.9 due to better implementation. As far as our knowledge goes, there are no known published results of neural information retrieval to rerank the results of the RM3 retrieval method. We can see that BM25+RM3 in Table 4.6 can beat all neural information retrieval methods in recent years.

The results in our experiments on the Robust04 corpus is shown in Table 4.10 using

RM	NRM	AP	P@20	NDCG@20
BM25	-	0.2515	0.3611	0.4226
	DRMM	0.2508	0.3453	0.4078
	KNRM	0.1016^{-}	0.1425^{-}	0.1461^{-}
	DUET	0.1313^{-}	0.1531^{-}	0.1841^{-}
	DSSM	0.0978^{-}	0.1318^{-}	0.1421^{-}
	CDSSM	0.0625^{-}	0.0841^{-}	0.0781^{-}
	Interpolated DRMM	$\mathbf{0.2775}^{+}$	$\mathbf{0.3886}^{+}$	$\mathbf{0.4517}^{+}$
	Interpolated KNRM	0.2546	0.3681	0.4320
	Interpolated DUET	0.2556	0.3653	0.4221
	Interpolated DSSM	0.2483	0.3922	0.4189
	Interpolated CDSSM	0.2482	0.3918	0.4198
BM25 +RM3	-	0.3493	0.3973	0.4514
	DRMM	0.2543^{-}	0.3405^{-}	0.4025^{-}
	KNRM	0.1145^{-}	0.1480^{-}	0.1512^{-}
	DUET	0.1426^{-}	0.1561^{-}	0.1946^{-}
	DSSM	0.0982^{-}	0.1331^{-}	0.1551^{-}
	CDSSM	0.0641^{-}	0.0842^{-}	0.0772^{-}
	Interpolated DRMM	$\mathbf{0.3151}^{+}$	$\mathbf{0.4147}^{+}$	$\mathbf{0.4717}^{+}$
	Interpolated KNRM	0.3036	0.3928	0.4441
	Interpolated DUET	0.3051	0.3986	0.4502
	Interpolated DSSM	0.3026	0.3946	0.4491
	Interpolated CDSSM	0.2995	0.3944	0.4468

Table 4.10: Result of retrieval and reranking on the Robust04 dataset. RM: retrieval model. NRM: neural re-ranking model. Significant improvement or degradation with respect to the retrieval model is indicated (+/-) (p-value ≤ 0.05).

the base retrieval method: QL, BM25, BM25+RM3 respectively. From the results, we can see that DRMM performs significantly better on the Robust04 dataset compared to the Microblog dataset. The reason has been analyzed by many researchers [30, 40] and the consensus is the DRMM captures more exact term matching instead of semantic matching and this property is similar to a retrieval model like BM25. The visualization figure by Bhaskar et al. [54] shows that DRMM and BM25 are both lexical matching models. This might also explain DRMM is the only method in our experiments that achieves close performance to BM25.

From Table 4.10 we can see that we achieve state-of-the-art effectiveness on the Robust04 dataset using interpolated DRMM. Note all these methods and tricks have been proposed for at least two years (RM3, DRMM, and interpolation). The performance is much better DRMM when it was proposed in Table 4.9.

Another interesting finding is that individual neural network models especially DRMM cannot achieve much improvement based on the good performance of the base retrieval model BM25+RM3. Take the MAP score as an example, compared to the base model BM25 and QL, DRMM alone achieves comparable performance. After interpolation, the AP result increases around 3 points. However, in Table 4.10, single DRMM's performance is lower than the base model by 4.4 points. After interpolation, the increase based on BM25+RM3 is just 1.7 points. This might due to:

- RM3 rerank method does not contribute a lot to the recall of returned documents but increases the precision score. The recall rate is actually the key point that affects the performance of neural text matching models.
- RM3 captures similar information to DRMM model, thus the combination does not contribute a lot.

In addition, we find DUET and KNRM are good at modeling short text such as the microblog text. However, they perform terribly on the long documents such as the news corpus (Robust04). We think the reason is these models cannot deal variant length problems in long documents. For short text the length variance is short, and it is easy for neural network models to learn a pattern for matching at different length scales. But for a long document, it is hard for CNN or RNN based methods to capture the information in the similarity matrix generated from the query and document directly. DRMM avoids this by transforming this similarity matrix into histograms, thus mostly relieving the dimension problem without losing much information. This allows DRMM captures more exact

matching information, which is more important in long document retrieval than short text retrieval.

From the results, we can see that without interpolation, all NN methods in this experiment cannot beat the retrieval model baseline (BM25 and BM25+RM3) significantly, and even worse compared to the BM25+RM3 baseline. This shows that although lots of complex supervised neural network models are proposed in recent years, traditional unsupervised retrieval methods are still one of the most stable and robust ways to solve the IR problems. In our experiments, traditional retrieval methods are much faster even the input (the whole corpus) is larger, since we build an index to accelerate searching before retrieval. This is another point we need to consider when we choose IR and text matching algorithms, especially in the industry environment.

I: irrelevant; R: relevant

ID	**Query**	**Sample Tweet**	**Label**	**Rank**		
				QL	**DUET**	**KNRM**
1	164: social media as educational tool	without social media the internet is an education tool	I	1	475	254
2		social media what use is social media in education : in view of the growing demand for social media skills st http://punchng.com/i-punch/what-use-is-social-media-in-education	R	11	3	1
3		should law schools be making better use of social media as a teaching tool http://canadianlawyermag.com/4537/Time-to-tweet.html #education #law	R	110	419	77
4	159: circular econoour initiatives	mof issues financial circular for the general federal budget http://wam.ae/ar/servlet/Satellite	I	256	7	14
5		what is a circular econoour https://www.ellenmacarthurfoundation.org/circular-econoour via #circulareconoour	R	3	17	16

Table 4.11: Sample Analysis

4.3.3 Error Analysis

So far, we have shown the performances of different neural information retrieval models on different datasets. However, we still lack knowledge about the following two questions: (1) What are the common characteristics of well-performing topics, and how do the different components contribute to overall effectiveness? (2) When does our model fail, and how can we further improve the model? To answer these questions, we provide additional qualitative and quantitative analyses over the per-query performance and sample tweets from poor-performing topics and models.

Microblog

To gain further insight into how NN models perform across different query topics, we conduct a per-topic analysis on the Microblog dataset as shown in Figure 4.4, and Figure 4.5. Each bar in the figure means the difference of the MAP scores between a neural reranking model and a retrieval baseline on a query. we select the two best-performing NN models (DUET and K-NRM), and show their MAP scores w.r.t QL baseline on the TREC 2014 dataset. Other methods without interpolation exhibit a similar pattern here.

First, form we find that both DUET and K-NRM suffer in most topics, affirming the ineffectiveness of existing NN models on tweet search. It is also interesting to see that the two models tend to perform well on a range of similar topics, suggesting that NN models are learning some shared representations that help with the search. To further understand this, we select some sample tweets for the best-performing topic 164 ("social media as educational tool") and the worst-performing topic 159 ("circular econoour initiatives") and present them in Table 4.11. The column "Rank" denotes the ranked position of the sample tweet in the returned list. Comparing sample 1 and 2, we can clearly see the benefits of NN models over QL baseline: NN methods tend to be less sensitive to exact matching signals and more semantic-oriented. Also, K-NRM produces a higher rank for sample 3 likely due to the semantic matches between the query word "educational" and the document words "schools", "teaching". In addition, we observe that the hashtags and URLs in sample 2, 3 and 5 contain strong relevance signals, which represent a unique characteristic of Microblog text that can be leveraged by character-level modeling.

Furthermore, from Figure 4.3, we can see that MP-HCNN models can achieve much better performance with interpolation. Combining MP-HCNN and the QL+RM3 baseline largely helps the topics on which the single MP-HCNN model performs relatively poor. From Figure 4.5, we can see that the base BERT model's performance on each topic has more difference to the base retrieval methods than other neural network methods. Thus,

with interpolation, BERT can achieve higher performance gain by considering two models capturing different match information.

Robust04

The per-topic analysis on the Robust04 dataset is shown in Figure 4.6 . Similar to the effect of interpolation in the Microblog dataset, we can see interpolation between DRMM and the BM25+RM3 baseline largely helps the topics on which the single DRMM model performs relatively poor. Although we can notice that the MAP scores decrease after interpolation for some topics on whic the single DRMM model performs well, the advantage of interpolation is still much larger than the disadvantage. This shows the effectiveness of the simple interpolation method in end-to-end information retrieval.

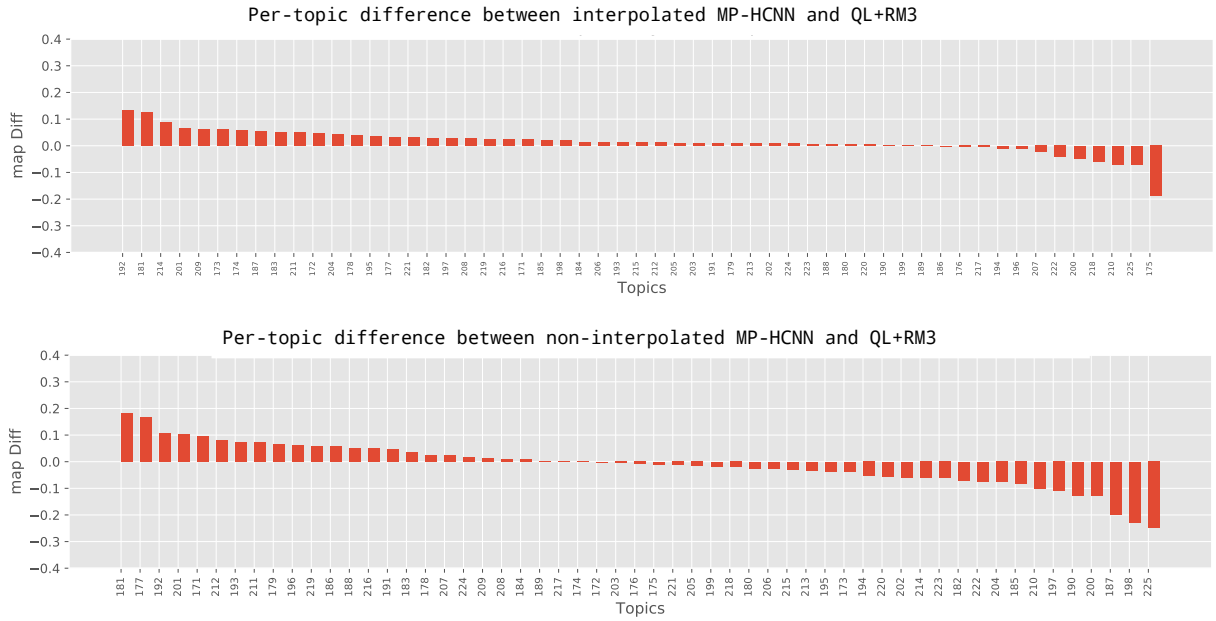

Figure 4.3: Per-topic differences of MAP scores between MP-HCNN and QL+RM3 on the Microblog dataset

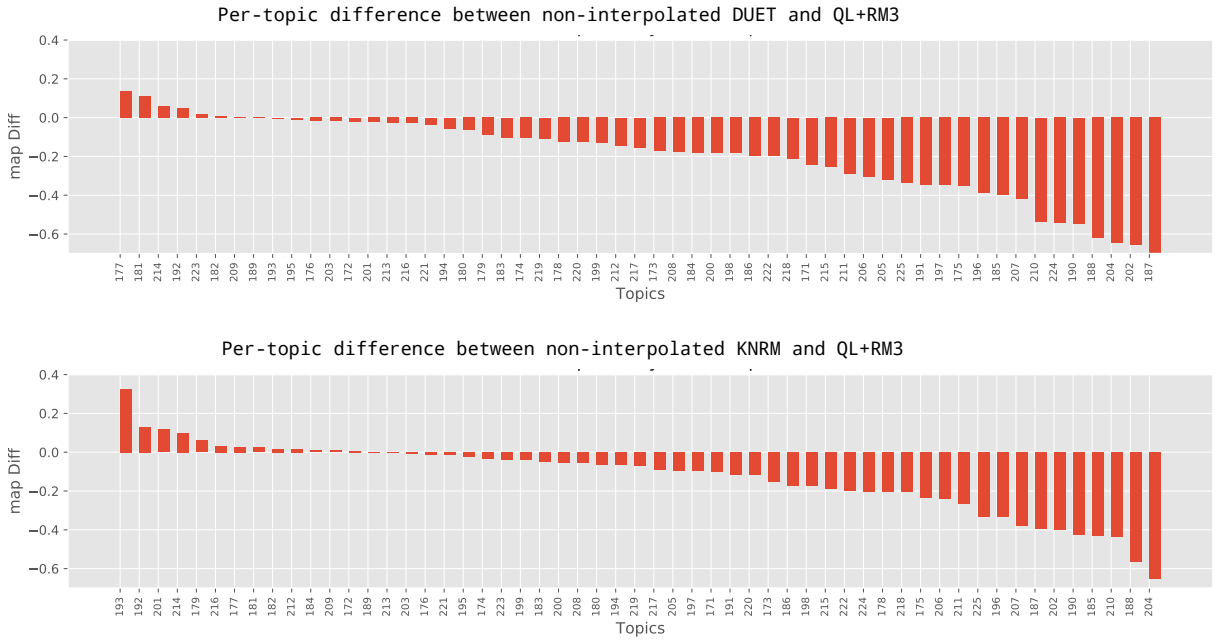

Figure 4.4: Per-topic differences of MAP scores between DUET/KNRM and QL+RM3 on the Microblog dataset

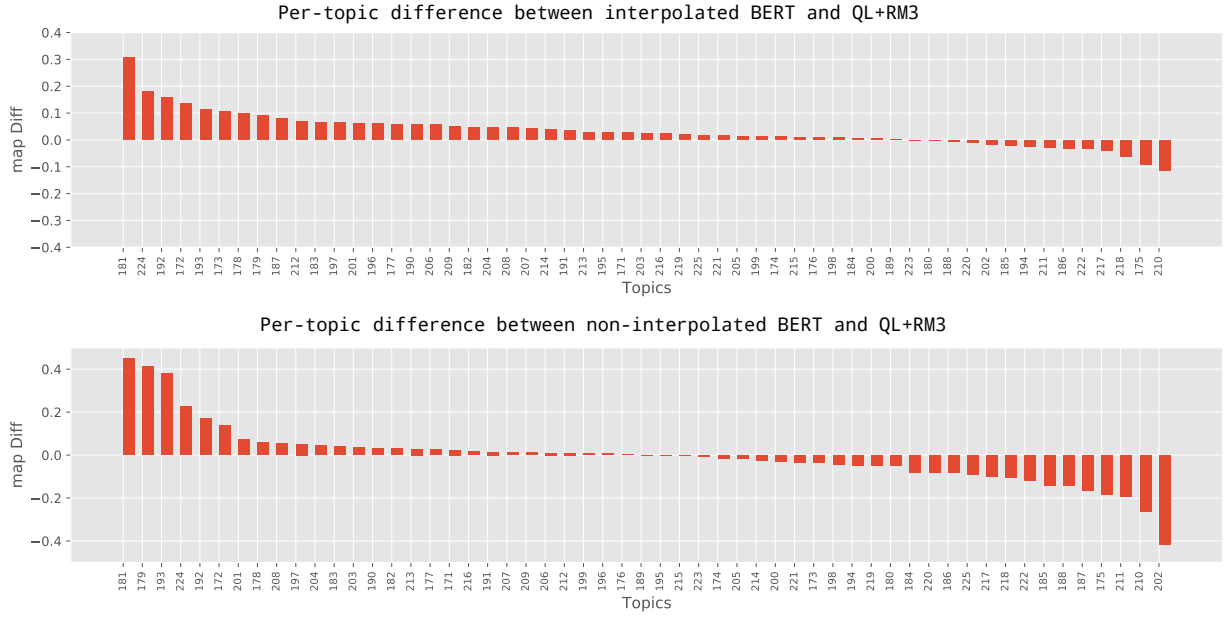

Figure 4.5: Per-topic differences of MAP scores between BERT and QL+RM3 on the Microblog dataset

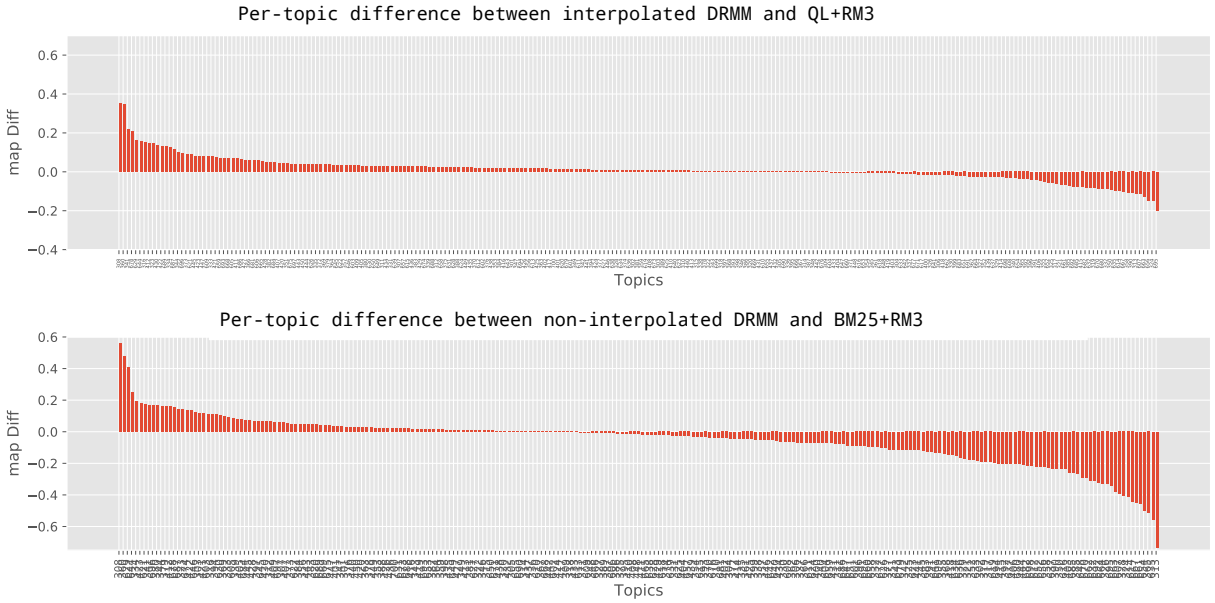

Figure 4.6: Per-topic differences of MAP scores between DRMM and BM25+RM3 on the Robust04 dataset

Chapter 5

Conclusions and Future Work

In our thesis, we work on the end-to-end neural information retrieval problem, which includes the retrieval step and the text matching step. For efficiency purposes, traditional methods like QL and BM25 retrieve candidate documents given a query in a rough but hard to beat way. Then complicated neural information retrieval models are applied to compute the similarity between the query and each candidate document. Then these candidate documents are ranked again by neural models, and the top K documents are returned to the user. The work in this thesis is based on four open-source toolkits and models: Anserini, MatchZoo, MP-HCNN, and BERT, which provides the models for traditional IR methods and the neural text matching. Through a straightforward combination, we construct an end-to-end system for neural information retrieval.

In our work we find researchers working on neural IR report different results of traditional IR methods (QL, BM25) on a single dataset. This is because they did not report the hyper-parameters of traditional IR methods in detail but just briefly mentioned 'we apply QL/BM25 in our work...'. Furthermore, RM3 is also an important relevance feedback based traditional IR method, and it performs better than QL and BM25. However, few neural IR researchers nowadays report the RM3 score in their work. In addition, we introduce a simple but effective interpolation method to combine the relevance score from traditional IR methods and the similarity score from the neural text matching models, and this method is rarely used in the neural IR papers in recent years, either. By fixing the problems mentioned above and make further improvements, we can obtain state-of-the-art performance on two datasets (Robust04 and Microblog) for end-to-end information retrieval using existed models.

However, since deep learning models have many parameters to learn, it is likely that

deep text matching models end up with overfitting the data. This means that most deep text matching models are purely data-driven. In the future, we hope to add prior information (e.g., large scale knowledge base and other information) which should be generally helpful for all downstream NLP tasks. On the other hand, for a specific downstream application, domain-specific matching models are required. Current matching models are designed for a general goal (relevance). However, different applications have a different matching goals. For example, in information retrieval, relevance does not strictly equal to the similarity. How to balance between exact matching and semantic matching is still an unsolved problem.

References

[1] Ossama Abdel-Hamid, Abdel-rahman Mohamed, Hui Jiang, Li Deng, Gerald Penn, and Dong Yu. Convolutional neural networks for speech recognition. *IEEE/ACM Transactions on audio, speech, and language processing*, 22(10):1533–1545, 2014.

[2] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014.

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

[4] Christopher Burges, Krysta Svore, Paul Bennett, Andrzej Pastusiak, and Qiang Wu. Learning to rank using an ensemble of lambda-gradient models. In *Proceedings of the Learning to Rank Challenge*, pages 25–35, 2011.

[5] Christopher JC Burges. From ranknet to lambdarank to lambdamart: An overview. *Learning*, 11(23-581):81, 2010.

[6] Zhe Cao, Tao Qin, Tie-Yan Liu, Ming-Feng Tsai, and Hang Li. Learning to rank: from pairwise approach to listwise approach. In *Proceedings of the 24th international conference on Machine learning*, pages 129–136. ACM, 2007.

[7] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. *arXiv preprint arXiv:1704.00051*, 2017.

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

[9] Sumit Chopra, Raia Hadsell, and Yann LeCun. Learning a similarity metric discriminatively, with application to face verification. In *Proceedings of the Computer Vision and Pattern Recognition, 2005*, volume 1, pages 539–546. IEEE, 2005.

[10] Yann N Dauphin, Angela Fan, Michael Auli, and David Grangier. Language modeling with gated convolutional networks. *arXiv preprint arXiv:1612.08083*, 2016.

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

[12] Jeffrey Donahue, Lisa Anne Hendricks, Sergio Guadarrama, Marcus Rohrbach, Subhashini Venugopalan, Kate Saenko, and Trevor Darrell. Long-term recurrent convolutional networks for visual recognition and description. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 2625–2634, 2015.

[13] Yixing Fan, Liang Pang, JianPeng Hou, Jiafeng Guo, Yanyan Lan, and Xueqi Cheng. Matchzoo: A toolkit for deep text matching. *arXiv:1707.07270*, 2017.

[14] Jenny Rose Finkel, Trond Grenager, and Christopher Manning. Incorporating non-local information into information extraction systems by gibbs sampling. In *Proceedings of the 43rd annual meeting on association for computational linguistics*, pages 363–370. Association for Computational Linguistics, 2005.

[15] Jenny Rose Finkel and Christopher D Manning. Joint parsing and named entity recognition. In *Proceedings of Human Language Technologies: The 2009 Annual Conference of the North American Chapter of the Association for Computational Linguistics*, pages 326–334. Association for Computational Linguistics, 2009.

[16] Debasis Ganguly, Dwaipayan Roy, Mandar Mitra, and Gareth JF Jones. Word embedding based generalized language model for information retrieval. In *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, pages 795–798. ACM, 2015.

[17] Jianfeng Gao, Mu Li, Andi Wu, and Chang-Ning Huang. Chinese word segmentation and named entity recognition: A pragmatic approach. *Computational Linguistics*, 31(4):531–574, 2005.

[18] Jonas Gehring, Michael Auli, David Grangier, Denis Yarats, and Yann N Dauphin. Convolutional sequence to sequence learning. *arXiv preprint arXiv:1705.03122*, 2017.

[19] Alex Graves and Jürgen Schmidhuber. Framewise phoneme classification with bidirectional lstm and other neural network architectures. *Neural Networks*, 18(5):602–610, 2005.

[20] Karol Gregor, Ivo Danihelka, Alex Graves, Danilo Jimenez Rezende, and Daan Wierstra. Draw: A recurrent neural network for image generation. *arXiv preprint arXiv:1502.04623*, 2015.

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

[22] Jiafeng Guo, Yixing Fan, Liang Pang, Liu Yang, Qingyao Ai, Hamed Zamani, Chen Wu, W Bruce Croft, and Xueqi Cheng. A deep look into neural ranking models for information retrieval. *arXiv preprint arXiv:1903.06902*, 2019.

[23] Jiafeng Guo, Gu Xu, Xueqi Cheng, and Hang Li. Named entity recognition in query. In *Proceedings of the 32nd international ACM SIGIR conference on Research and development in information retrieval*, pages 267–274. ACM, 2009.

[24] Hua He and Jimmy Lin. Pairwise word interaction modeling with deep neural networks for semantic similarity measurement. In *Proceedings of the 2016 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 937–948, 2016.

[25] Hua He, John Wieting, Kevin Gimpel, Jinfeng Rao, and Jimmy Lin. Umd-ttic-uw at semeval-2016 task 1: Attention-based multi-perspective convolutional neural networks for textual similarity measurement. In *SemEval*, pages 1103–1108, 2016.

[26] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016.

[27] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997.

[28] Po-Sen Huang, Xiaodong He, Jianfeng Gao, Li Deng, Alex Acero, and Larry Heck. Learning deep structured semantic models for web search using clickthrough data. In *Proceedings of the 22nd ACM international conference on Information & Knowledge Management*, pages 2333–2338. ACM, 2013.

[29] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Pacrr: A position-aware neural ir model for relevance matching. *arXiv preprint arXiv:1704.03940*, 2017.

[30] Kai Hui, Andrew Yates, Klaus Berberich, and Gerard de Melo. Position-aware representations for relevance matching in neural information retrieval. In *Proceedings of the 26th International Conference on World Wide Web Companion*, pages 799–800. International World Wide Web Conferences Steering Committee, 2017.

[31] Kalervo Järvelin and Jaana Kekäläinen. Cumulated gain-based evaluation of ir techniques. *ACM Transactions on Information Systems (TOIS)*, 20(4):422–446, 2002.

[32] Thorsten Joachims. Optimizing search engines using clickthrough data. In *Proceedings of the eighth ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 133–142. ACM, 2002.

[33] Yoon Kim. Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*, 2014.

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

[35] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[36] Wuwei Lan and Wei Xu. Neural network models for paraphrase identification, semantic textual similarity, natural language inference, and question answering. *arXiv preprint arXiv:1806.04330*, 2018.

[37] Victor Lavrenko and W Bruce Croft. Relevance based language models. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 120–127, 2001.

[38] Steve Lawrence, C Lee Giles, Ah Chung Tsoi, and Andrew D Back. Face recognition: A convolutional neural-network approach. *IEEE transactions on neural networks*, 8(1):98–113, 1997.

[39] Yann LeCun, Léon Bottou, Yoshua Bengio, and Patrick Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998.

[40] Canjia Li, Yingfei Sun, Ben He, Le Wang, Kai Hui, Andrew Yates, Le Sun, and Jungang Xu. Nprf: A neural pseudo relevance feedback framework for ad-hoc information retrieval. *arXiv preprint arXiv:1810.12936*, 2018.

[41] J. Li, X. Chen, E. Hovy, and D. Jurafsky. Visualizing and understanding neural models in NLP. *arXiv:1506.01066*, 2015.

[42] Jiwei Li, Will Monroe, Alan Ritter, Michel Galley, Jianfeng Gao, and Dan Jurafsky. Deep reinforcement learning for dialogue generation. *arXiv preprint arXiv:1606.01541*, 2016.

[43] Jiwei Li, Will Monroe, Tianlin Shi, Alan Ritter, and Dan Jurafsky. Adversarial learning for neural dialogue generation. *arXiv preprint arXiv:1701.06547*, 2017.

[44] Jun Li, Jinxian Pan, Chen Ye, Yong Huang, Danlu Wen, and Zhichun Wang. Linking entities in chinese queries to knowledge graph. In *Proceedings of the National CCF Conference on Natural Language Processing and Chinese Computing*, pages 590–597. Springer, 2015.

[45] Jimmy Lin. The neural hype and comparisons against weak baselines. In *Proceedings of the SIGIR Forum*, volume 52(2), pages 40–51. ACM, 2018.

[46] Jimmy Lin and Miles Efron. Overview of the trec-2013 microblog track. In *Proceedings of the TREC conference*, 2013.

[47] Jimmy Lin and Miles Efron. Overview of the TREC-2013 microblog track. In *Proceedings of the TREC conference*, 2013.

[48] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. Overview of the trec-2014 microblog track. In *Proceedings of the TREC conference*, 2014.

[49] Jimmy Lin, Miles Efron, Yulu Wang, and Garrick Sherman. Overview of the trec-2014 microblog track. Technical report, MARYLAND UNIV COLLEGE PARK, 2014.

[50] Yuanhua Lv and ChengXiang Zhai. Positional relevance model for pseudo-relevance feedback. In *Proceedings of the 33rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 579–586, 2010.

[51] Ryan McDonald, Georgios-Ioannis Brokos, and Ion Androutsopoulos. Deep relevance ranking using enhanced document-query interactions. *arXiv preprint arXiv:1809.01682*, 2018.

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

[53] Tomas Mikolov, Ilya Sutskever, Kai Chen, Greg S Corrado, and Jeff Dean. Distributed representations of words and phrases and their compositionality. In *Proceedings of the Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[54] Bhaskar Mitra and Nick Craswell. Neural models for information retrieval. *CoRR*, abs/1705.01509, 2017.

[55] Bhaskar Mitra, Fernando Diaz, and Nick Craswell. Learning to match using local and distributed representations of text for web search. In *Proceedings of the Web conference*, pages 1291–1299, 2017.

[56] Vinod Nair and Geoffrey E Hinton. Rectified linear units improve restricted boltzmann machines. In *Proceedings of the 27th international conference on machine learning*, pages 807–814, 2010.

[57] Rodrigo Nogueira and Kyunghyun Cho. Passage re-ranking with bert. *arXiv preprint arXiv:1901.04085*, 2019.

[58] Iadh Ounis, Craig MacDonald, Jimmy Lin, and Ian Soboroff. Overview of the trec-2011 microblog track. In *Proceedings of the TREC conference*, 2011.

[59] Iadh Ounis, Craig Macdonald, Jimmy Lin, and Ian Soboroff. Overview of the trec-2011 microblog track. In *Proceedings of the TREC conference*, volume 32, 2011.

[60] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. A study of match-pyramid models on ad-hoc retrieval. *arXiv preprint arXiv:1606.04648*, 2016.

[61] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Shengxian Wan, and Xueqi Cheng. Text matching as image recognition. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[62] Nanyun Peng and Mark Dredze. Named entity recognition for chinese social media with jointly trained embeddings. In *Proceedings of the Empirical Methods in Natural Language Processing conference*, pages 548–554, 2015.

[63] Jeffrey Pennington, Richard Socher, and Christopher D. Manning. Glove: Global vectors for word representation. In *Proceedings of the Empirical Methods in Natural Language Processing conference*, pages 1532–1543, 2014.

[64] Matthew E Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. *arXiv preprint arXiv:1802.05365*, 2018.

[65] Qiao Qian, Minlie Huang, and Xiaoyan Zhu. Assigning personality/identity to a chatting machine for coherent conversation generation. *arXiv preprint arXiv:1706.02861*, 2017.

[66] Jinfeng Rao, Hua He, and Jimmy Lin. Noise-contrastive estimation for answer selection with deep neural networks. In *Proceedings of the Conference on Information and Knowledge Management conference*, pages 1913–1916. ACM, 2016.

[67] Jinfeng Rao, Hua He, and Jimmy Lin. Experiments with convolutional neural network models for answer selection. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1217–1220. ACM, 2017.

[68] Jinfeng Rao, Hua He, Haotian Zhang, Ferhan Ture, Royal Sequiera, Salman Mohammed, and Jimmy Lin. Integrating lexical and temporal signals in neural ranking models for social media search. In *Proceedings of SIGIR Workshop on Neural Information Retrieval (Neu-IR)*, 2017.

[69] Jinfeng Rao, Ferhan Ture, Hua He, Oliver Jojic, and Jimmy Lin. Talking to your tv: Context-aware voice search with hierarchical recurrent neural networks. In *Proceedings of the 2017 ACM on Conference on Information and Knowledge Management*, pages 557–566. ACM, 2017.

[70] Jinfeng Rao, Ferhan Ture, Xing Niu, and Jimmy Lin. Mining the temporal statistics of query terms for searching social media posts. In *Proceedings of the ACM SIGIR International Conference on Theory of Information Retrieval*, pages 133–140. ACM, 2017.

[71] Jinfeng Rao, Wei Yang, Yuhao Zhang, Ferhan Ture, and Jimmy Lin. Multi-perspective relevance matching with hierarchical convnets for social media search. *Proceedings of the Thirty-Third AAAI Conference on Artificial Intelligence (AAAI)*, 2019.

[72] Alan Ritter, Sam Clark, Oren Etzioni, et al. Named entity recognition in tweets: an experimental study. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing*, pages 1524–1534. Association for Computational Linguistics, 2011.

[73] Royal Sequiera, Gaurav Baruah, Zhucheng Tu, Salman Mohammed, Jinfeng Rao, Haotian Zhang, and Jimmy Lin. Exploring the effectiveness of convolutional neural networks for answer selection in end-to-end question answering. *arXiv preprint arXiv:1707.07804*, 2017.

[74] Iulian Vlad Serban, Alessandro Sordoni, Yoshua Bengio, Aaron C Courville, and Joelle Pineau. Building end-to-end dialogue systems using generative hierarchical neural network models. In *Proceedings of the AAAI conference*, pages 3776–3784, 2016.

[75] A. Severyn and A. Moschitti. Learning to rank short text pairs with convolutional deep neural networks. In *Proceedings of the International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 373–382, 2015.

[76] Lifeng Shang, Zhengdong Lu, and Hang Li. Neural responding machine for short-text conversation. *arXiv preprint arXiv:1503.02364*, 2015.

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

[78] Mark D. Smucker, James Allan, and Ben Carterette. A comparison of statistical significance tests for information retrieval evaluation. In *Proceedings of the Conference on Information and Knowledge Management conference*, pages 623–632, 2007.

[79] Ian Soboroff, Iadh Ounis, Craig Macdonald, and Jimmy J Lin. Overview of the trec-2012 microblog track. In *Proceedings of the TREC conference*, volume 2012, page 20. Citeseer, 2012.

[80] Richard Socher, Eric Huang, Jeffery Pennin, Christopher Manning, and Andrew Ng. Dynamic pooling and unfolding recursive autoencoders for paraphrase detection. In *NIPS*, pages 801–809, 2011.

[81] Martin Sundermeyer, Ralf Schlüter, and Hermann Ney. Lstm neural networks for language modeling. In *Thirteenth Annual Conference of the International Speech Communication Association*, 2012.

[82] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. In *Proceedings of Advances in Neural Information Processing Systems*, pages 3104–3112, 2014.

[83] Duyu Tang, Furu Wei, Nan Yang, Ming Zhou, Ting Liu, and Bing Qin. Learning sentiment-specific word embedding for twitter sentiment classification. In *Proceedings of the ACL conference*, pages 1555–1565, 2014.

[84] Kristina Toutanova, Dan Klein, Christopher D Manning, and Yoram Singer. Feature-rich part-of-speech tagging with a cyclic dependency network. In *Proceedings of the 2003 Conference of the North American Chapter of the Association for Computational Linguistics on Human Language Technology-Volume 1*, pages 173–180. Association for Computational Linguistics, 2003.

[85] Ellen M Voorhees et al. Overview of the trec 2005 robust retrieval track. In *Proceedings of the TREC conference*, 2005.

[86] Shengxian Wan, Yanyan Lan, Jiafeng Guo, Jun Xu, Liang Pang, and Xueqi Cheng. A deep architecture for semantic matching with multiple positional sentence representations. In *Proceedings of the AAAI conference*, volume 16, pages 2835–2841, 2016.

[87] Chunwei Lu Anqi Cui Han Li Xi Chen Kun Xiong Muzi Wang Ming Li Jian Pei Wei Yang, Luchen Tan and Jimmy Lin. Detecting customer complaint escalation with recurrent neural networks and manually-engineered features. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, 2019.

[88] Qing Xia, Xin Yan, Zhengtao Yu, and Shengxiang Gao. Research on the extraction of wikipedia-based chinese-khmer named entity equivalents. In *Proceedings of the National CCF Conference on Natural Language Processing and Chinese Computing*, pages 372–379. Springer, 2015.

[89] Chenyan Xiong, Zhuyun Dai, Jamie Callan, Zhiyuan Liu, and Russell Power. End-to-end neural ad-hoc ranking with kernel pooling. In *Proceedings of the 40th International ACM SIGIR conference on research and development in information retrieval*, pages 55–64. ACM, 2017.

[90] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Enabling the use of lucene for information retrieval research. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1253–1256. ACM, 2017.

[91] Wei Yang, Wei Lu, and Vincent Zheng. A simple regularization-based algorithm for learning cross-domain word embeddings. In *Proceedings of the 2017 Conference on*

Empirical Methods in Natural Language Processing, pages 2898–2904. Association for Computational Linguistics, 2017.

[92] Wei Yang, Yuqing Xie, Aileen Lin, Xingyu Li, Luchen Tan, Kun Xiong, Ming Li, and Jimmy Lin. End-to-end open-domain question answering with bertserini. *arXiv preprint arXiv:1902.01718*, 2019.

[93] Hamed Zamani and W Bruce Croft. Relevance-based word embedding. In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 505–514. ACM, 2017.

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

[95] GuoDong Zhou and Jian Su. Named entity recognition using an hmm-based chunk tagger. In *proceedings of the 40th Annual Meeting on Association for Computational Linguistics*, pages 473–480. Association for Computational Linguistics, 2002.