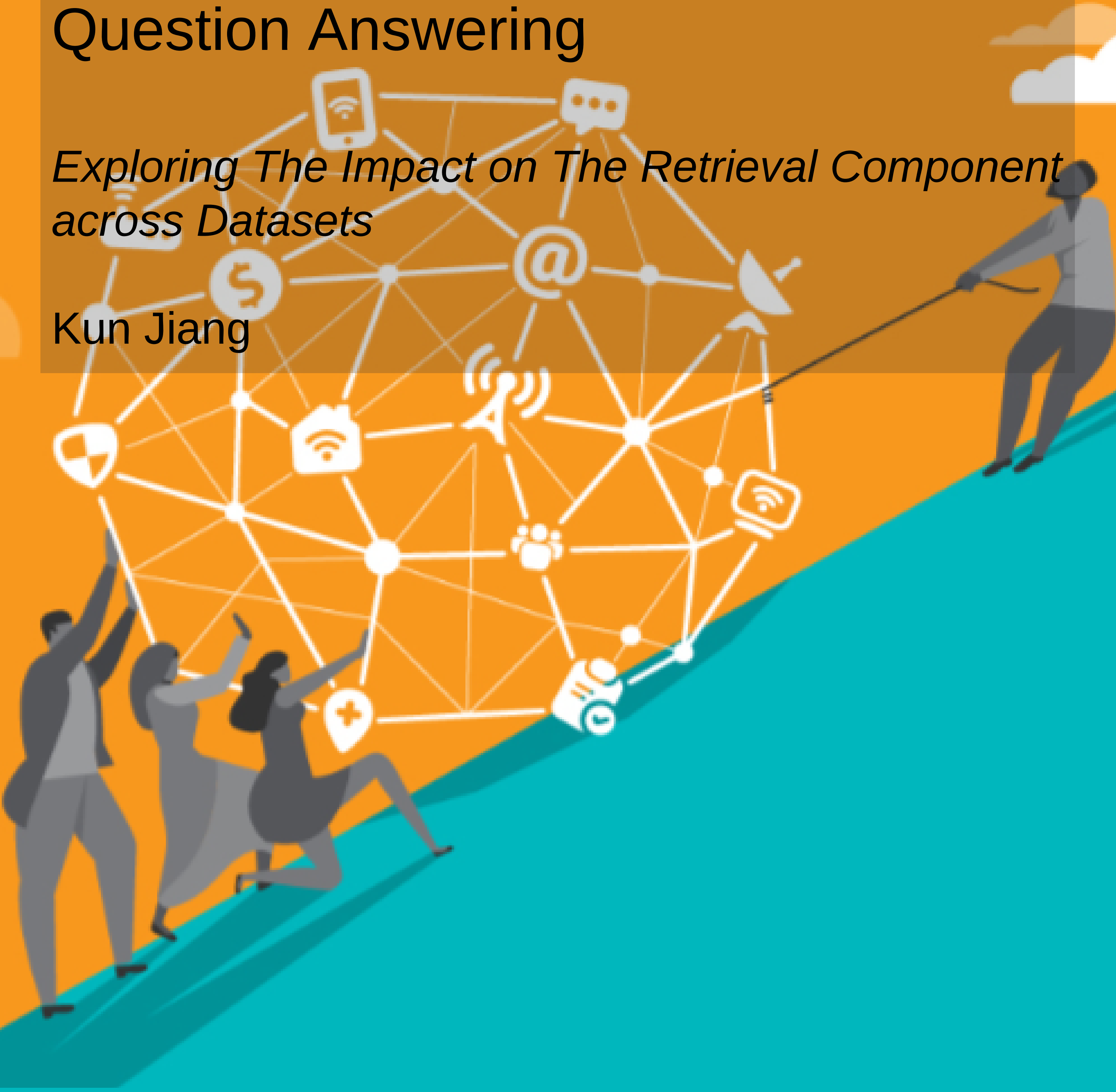
Retrieval-Based Open-Domain Question Answering

Exploring The Impact on The Retrieval Component across Datasets

Kun Jiang



Retrieval-Based Open-Domain Question Answering

Exploring The Impact on The Retrieval Component across Datasets

MASTER THESIS

submitted in partial fulfillment of the
requirements for the degree of

MASTER OF SCIENCE

in

EMBEDDED SYSTEMS

by

KUN JIANG

Faculty EEMCS, Delft University of Technology
Delft, the Netherlands

Preface

The past two years of being a Master student at TU Delft have been an unforgettable and invaluable experience to me. When I first came to the Netherlands for my Master study, I knew little about this country and was a bit nervous about being a freshman in a new environment. It is unbelievable that over the following two years I have succeeded in obtaining 80 credits for all the courses of my Master journey and currently I am almost there to finish my Master thesis. I would not be able to make this journey without the help and support from many people and I am thankful to them.

First and foremost, my greatest thanks go to my supervisor Claudia Hauff for her extensive efforts to guide me throughout my entire work. I was initially confused and worried about my thesis and did not know what to do. Thanks to the guidance from Claudia Hauff, I can perform my Master thesis step by step. Also, she has contributed enormously to helping me improve my thesis, including the thesis's structure, the validity of writing, conciseness of statements, etc. She is always insightful and thoughtful and gives me significant comments and instructions. Moreover, she arranges a *Paper Reading* meeting every Monday for her Master and PhD students. During the meeting, we can share our thoughts about a specific published paper and this helps me improve my reading and speaking skills as well as my critical thinking. I am forever grateful to her.

I would like to thank professor Odette Scharenborg and professor Fernando Kuipers for being my thesis committee. Also, I would like to appreciate the help from Dimitrios Bountouridis, who gave me some useful advice on my thesis and Nirmal Roy, who has been helping me go through the journey of my thesis.

Next, I am grateful to the Master students and PhD students in our lab, including Wanning Yang, Gustavo Penha, Felipe Moraes, Arthur Camara, Francisco Morales, Alex Balan, and Daan Rennings. We have been together for many paper reading meetings and I have learned a lot from them.

Last but not least, I would thank my mom and dad for giving me a great deal of support during my Master study.

KUN JIANG
Delft, the Netherlands
July 29, 2019

Abstract

Open-domain question answering (QA) is an important step in Artificial Intelligence and its ultimate goal is to build a QA system that can answer any question posed by humans. The majority of the open-domain QA system is the retrieval-based open-domain QA system, which enables the retrieval component to retrieve relevant documents from a large-scale knowledge source to a question and the answer extraction component to extract the answer to this question based on retrieved documents. As the techniques of Deep Learning progressing significantly, many researchers tried to apply the neural reading comprehension (RC) model to serve the answer extraction component of the open-domain QA system. However, the performance of the neural RC model in open-domain QA is considerably worse than the performance of it in RC-style QA. Therefore, many works have focused on the neural RC model for addressing the performance gap, whereas the retrieval component of the open-domain QA system lacks equivalent attention.

Some researchers have built the neural network based information retrieval (IR) models, but currently, it is still difficult for these neural IR models to directly retrieve documents from a large-scale knowledge source in open-domain QA. Hence, many works attempted to use neural IR models for re-ranking documents retrieved by the traditional but efficient IR models (e.g., TF-IDF, BM25) in open-domain QA. However, these works did not analyze the impact of different questions of QA datasets on traditional IR models. Thus, this research gap is the focus in this thesis.

We conduct error analyses of questions of different QA datasets to figure out the error types of questions that have a negative impact on the traditional IR models. From the error analysis, we learn that different QA datasets have different impacts on the traditional IR models and are differently hard to be dealt with by the traditional IR models. Therefore, we propose hypotheses that might mitigate the negative impact of the error types of questions that are relatively harder to be handled by the traditional IR models. Furthermore, we perform experiments based on the methodologies that implement our hypotheses for figuring out the validity of these hypotheses.

In conclusion, we believe that our work is a step forward to obtaining more insights into the retrieval component of the open-domain QA system and will contribute to the development of the retrieval component for a better open-domain QA system. Moreover, our work can give our users guidance on how to issue a more suitable question that can be processed by the open-domain QA system for giving a more accurate and better answer.

Contents

Preface i

Abstract iii

Contents v

List of Figures vii

List of Tables ix

1 Introduction 1
- 1.1 Motivation . . . 3
- 1.2 Aim of Research . . . 6
- 1.3 Thesis Outline . . . 6

2 Related Work 7
- 2.1 Question Answering . . . 7
  - 2.1.1 Reading Comprehension-Style QA . . . 7
  - 2.1.2 Open-Domain QA . . . 12
- 2.2 Information Retrieval . . . 13
  - 2.2.1 Preliminaries . . . 14
  - 2.2.2 Traditional IR Models . . . 16
  - 2.2.3 Neural IR Models . . . 20

3 Exploratory Analysis of SQuAD, HotpotQA and TriviaQA 25
- 3.1 Datasets Selection . . . 25
  - 3.1.1 SQuAD . . . 26
  - 3.1.2 HotpotQA . . . 28
  - 3.1.3 TriviaQA . . . 30
- 3.2 Retrieval Test on Datasets . . . 31
  - 3.2.1 Retrieval Test Setup . . . 31
  - 3.2.2 Retrieval Results . . . 34
- 3.3 Analysis of Retrieval Test . . . 35
  - 3.3.1 Error Analysis . . . 35

3.3.2 Conclusion 43
3.4 Hypothesis 45
3.4.1 Hypothesis I (H1) 45
3.4.2 Hypothesis II (H2) 46
3.4.3 Hypothesis III (H3) 47
3.5 Methodology 47
3.5.1 Paragraph and Sentence Based Retrieval Document Unit (H1) 47
3.5.2 Query Expansion by Pseudo Relevance Feedback (H2) 49
3.5.3 Neural Paragraph Ranker (H3) 49
3.5.4 RC Models 51

4 Experiments for Validating Hypotheses **55**
4.1 Experimental Setup 55
4.1.1 Datasets and The Knowledge Source 56
4.1.2 Evaluation Metrics 56
4.1.3 Traditional IR Models 56
4.1.4 Hyperparameter Tuning 56
4.2 Evaluation 59
4.2.1 Paragraph and Sentence Based Retrieval Document Unit (H1) 59
4.2.2 Query Expansion by Pseudo Relevance Feedback (H2) 59
4.2.3 Neural Paragraph Ranker (H3) 59
4.3 Results and Analysis 60
4.3.1 Results and Analysis Based on H1 60
4.3.2 Results and Analysis Based on H2 63
4.3.3 Results and Analysis Based on H3 64
4.4 Conclusion 69

5 Conclusion and Future Work **71**
5.1 Conclusion 71
5.2 Future Work 72
5.2.1 Performance Improvement on A Specific Error Type 72
5.2.2 Integration of Our Methodologies 72
5.2.3 Impact of Different Aspects of Questions 73

Bibliography **75**

A Hyperparameter Tuning **89**

List of Figures

1.1 The two-stage working pipeline of the open-domain QA system. 1

2.1 The working process of RNNs. Each input word is processed by the hidden layer and the hidden layer maintains previous hidden values to output a text embedding. Input words are generally represented by pre-trained word embeddings (e.g., Glove [91] or Fasttext [50]). 9

2.2 The overview of an open-domain QA system DrQA. Figure is copied from [17]. 13

2.3 The overview of the IR process. 14

2.4 A term-document incidence matrix and each entry (t,d) is 1 if the document *d* in column contains the word *t* in row, and is 0 otherwise. The figure is copied from [72]. 14

2.5 The inverted index. The figure is copied from [72]. 15

2.6 Two categories of Neural IR models. 22

2.7 The architecture of Deep Structured Semantic Model (DSSM) [45], which belongs to the representation-focused model. Figure is copied from [45]. . 22

2.8 The architecture of ARC-II [43], which belongs to interaction-focused models. The figure is copied from [43]. 23

2.9 The overview of the retrieval process with the neural IR model as the ranker for re-ranking top-ranked documents retrieved by the traditional IR model. 23

3.1 A conversation from the CoQA dataset. The figure is copied from [96]. . 26

3.2 The hyperlink in the Wikipedia article is used to generate QA pairs of HotpotQA, so each question was generated based on two paragraphs of two Wikipedia articles. 28

3.3 The reasoning process for one question of HotpotQA with bridge entity. . 29

3.4 Some paragraphs of the Wikipedia article "*Super Bowl 50*". 48

3.5 Some sentences of the Wikipedia article "*Super Bowl 50*". 48

3.6 The overview of the retrieval process of the traditional IR model with the index based on each paragraph or each sentence of Wikipedia articles as the retrieval document unit. 48

3.7 The overview of the retrieval process of the traditional IR model with RM3 model. 49
3.8 The overview of the retrieval process of the traditional IR model with Paragraph Ranker. 50
3.9 The overview of the question encoder and paragraph encoder. The input of the question encoder is word embeddings of question terms and the input of the paragraph encoder is word embeddings of paragraphs terms. 50
3.10 The architecture of Document Reader of DrQA. 53
3.11 The architecture of the neural RC model of HotpotQA. 54

4.1 The pipeline of our experiments. 55
4.2 The performance of traditional IR models on SQuAD in the open-domain setting with Paragraph Ranker re-ranking the increased number of documents. The metrics with "Ori-" indicate that these performances are under the traditional IR model without Paragraph Ranker. 66
4.3 The performance of traditional IR models on HotpotQA in the open-domain setting with Paragraph Ranker re-ranking increased number of documents. The metrics with "Ori-" indicate that these performances are under the traditional IR model without Paragraph Ranker. 68

List of Tables

1.1 Examples of RC-style QA and open-domain QA. The paragraph containing the answer is in green and not containing the answer is in blue. Answers are in yellow. 2
1.2 An example of HotpotQA [136]. Reasoning facts are in underlineand the answer is in yellow. 4

2.1 Examples from each category of RC-style QA based on the answer type. The answer in the text is in yellow. 11
2.2 Some fabricated examples of document vectors using TF-IDF weighting. 17

3.1 The comparison of some popular QA datasets focusing on the task of RC-style QA and open-domain QA. Note that in the column of *Ques form (Nature)*, "✓" indicates that the question form of the dataset is natural language and "✗" otherwise. 27
3.2 Two examples of dataset SQuAD. Each QA pair was generated from the provided paragraph, which is one of the paragraphs of a Wikipedia article. The ground truth answer is in yellow. 27
3.3 Two examples of dataset HotpotQA. Each QA pair was generated from two paragraphs of two different Wikipedia articles. The ground truth answer only appears in one of the paragraphs, which is in yellow. 29
3.4 Two examples of dataset TriviaQA. The document for each QA pair was gathered from Wikipedia. The ground truth answer is in yellow. 31
3.5 Statistics of datasets used in our retrieval test. 32
3.6 Fine-tuned hyperparameters are derived from hyperparameter tuning on each training set of SQuAD, HotpotQA and TriviaQA in the open-domain setting for optimizing the metric TOP-10. 34
3.7 Retrieval results of our retrieval test on development set of SQuAD, HotpotQA and TriviaQA in the open-domain setting. TOP-5(10) indicates % of questions whose top-ranked 5(10) documents retrieved by the traditional IR model from Wikipedia contain the answer. The best retrieval performance of TOP-5(10) for each dataset is in **bold**. 34

3.8 The overview of the error analysis. We randomly sample 100 questions from each dataset respectively and these questions are ones whose top-ranked 10 documents retrieved by BM25 from Wikipedia do not contain the answer. Error types with * indicate that the reason for these questions whose top-ranked 10 documents do not contain the answer is not due to the questions or the retrieval model BM25. The error type of each dataset in the open-domain setting which has the largest number of questions is in **bold** (except for error types with *). 36
3.9 The number of questions with ambiguous, unambiguous and bridge entities out of 100 questions sampled from SQuAD, HotpotQA and TriviaQA respectively. The number of questions is analyzed in the open-domain setting. Note that HotpotQA$^*_{fullwiki}$ has some questions with ambiguous and bridge entities simultaneously, so these questions not only belong to questions with ambiguous entities but also questions with bridge entities. The largest number of questions with ambiguous, unambiguous and bridge entities among all datasets is in **bold**. 44

4.1 The fine-tuned hyperparameters of BM25 and QL based on the indexes which consider each paragraph (+Para) and each sentence (+Sent) of Wikipedia articles as the retrieval document unit. Hyperparameters are tuned on the training set of each dataset in the open-domain setting. 57
4.2 The fine-tuned hyperparameters of traditional IR models with RM3. Hyperparameters are tuned on the training set of each dataset in the open-domain setting. 58
4.3 The experimental results of the methodology on SQuAD and HotpotQA in the open-domain setting, which is based on Wikipedia considering paragraph-based (+Para) and sentence-based (+Sent) retrieval document units. For each traditional IR model, the best performance on each metric is in bold. Note that the performances of "TF-IDF", "BM25" and "QL" are ones with article-based retrieval document unit from the retrieval test in Table 3.7. . 61
4.4 The experimental results of the methodology on SQuAD and HotpotQA in the open-domain setting, which employs RM3 (+RM3) to do query (i.e., question) expansion. For each traditional IR model, the best performance on each metric is in bold. Note that the performances of "TF-IDF", "BM25" and "QL" are ones with article-based retrieval document unit from the retrieval test in Table 3.7. 64
4.5 The experimental results of the methodology on SQuAD and HotpotQA in the open-domain setting, which employs Paragraph Ranker (+Ranker) to re-rank all paragraphs of top-ranked 20 Wikipedia articles retrieved by the traditional IR model. For each traditional IR model, the best performance on each metric is in bold. Note that the performances of "TF-IDF", "BM25" and "QL" are ones with article-based retrieval document unit from the retrieval test in Table 3.7. 65

A.1 Values to be tested by grid search for hyperparameter tuning of traditional IR models used in our work. All hyperparameters are tuned on the training set of each dataset in the open-domain setting. 89

Chapter 1

Introduction

Open-domain question answering (QA) is an important step approaching the ultimate goal of QA, that is a general QA system which can answer a question regarding any domain posed by humans, no matter what sort of resources it depends on [16, 32]. Open-domain QA systems aim to answer questions posed by humans in the form of natural language [28, 111, 112] and the majority[1] of these systems have a two-stage working pipeline: *retrieval stage* and *answer extraction stage*, which can be found in Figure 1.1. This retrieval-based open-domain QA system[2] first retrieves relevant documents to the question from a large-scale knowledge source (e.g., the web and the full Wikipedia) by information retrieval (IR) techniques as the retrieval component in the retrieval stage and then it extracts the answer to the question based on the retrieved documents by the answer retrieval component in the answer extraction stage [17, 28].

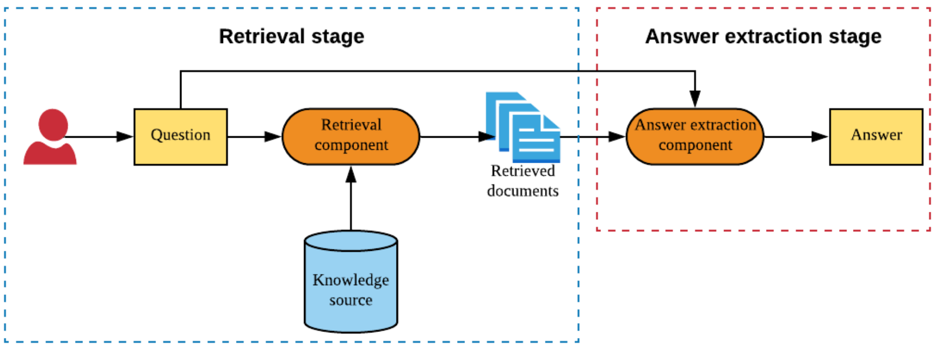


Figure 1.1: The two-stage working pipeline of the open-domain QA system.

In the QA community, *open-domain* can be interpreted either as the scope of question topics or the capacity and breadth of the knowledge source used to answer questions or both of them [17, 125]. In contrast, *closed-domain* QA is a task in which questions are domain specific (e.g., science or medicine) [2], topics are limited in the

[1]Some recent open-domain QA systems have been working with approach of answer generation instead of this two-stage pipeline [15].

[2]For convenience, open-domain QA in this thesis only refers to the retrieval-based open-domain QA, which has a two-stage pipeline.

knowledge source, question types are limited or even a specific text is provided for a QA system to answer the question based on this text. In particular, the task in which a QA system tries to answer questions based on the provided text is called reading comprehension (RC)-style QA [32, 42]. In the RC-style QA task, the question is created based on the text that contains all the reasoning facts as well as the answer and the QA system extracts the answer to the question from this text [40, 94]. This is because the RC-style QA task aims at improving the ability of the QA system on the linguistic analysis of the text, so generating a question from the text enables that the question can be answered successfully by the QA system only based on the source text. Some examples of RC-style QA and open-domain QA are shown in Table 1.1. As we can see from the examples in Table 1.1, RC-style QA is given a paragraph that contains the answer span[3] "*Association of American Universities*", but this is not the case for open-domain QA, since the paragraphs retrieved by the retrieval component might not contain the answer (e.g., paragraph2 in blue in Table 1.1).

Reading comprehension-style question answering
Question: What organization did Harvard found in 1900?
Paragraph: Following the American Civil War, President Charles W. Eliot's long tenure (1869–1909) transformed the college and affiliated professional schools into a modern research university; Harvard was a founding member of the Association of American Universities in 1900.
Answer: Association of American Universities
Open-domain question answering
Question: What does a camel store in its hump?
Paragraph1: The humps are reservoirs of fatty tissue: concentrating body fat in their humps minimizes the insulating effect fat would have if distributed over the rest of their bodies, helping camels survive in hot climates.
Paragraph2: Camels with one hump are called Arabian camels, or Dromedaries, and come from North Africa. Camels with two humps are from Asia, and are called Bactrian camels.
Answer: fat

Table 1.1: Examples of RC-style QA and open-domain QA. The paragraph containing the answer is in green and not containing the answer is in blue. Answers are in yellow.

Question answering is a challenging and one of the earliest tasks in Natural Language Processing (NLP) which can date back to the 1960s [16, 47, 57]. PROTO-SYNTHEX system proposed by Simmons et al. [107] employed some rules based on lexical and semantic heuristics to analyze the dependency relations of words, which lays a foundation to the rule-based QA systems. MURAX is another notable QA system from [58] which used some linguistic approaches (e.g., part-of-speech tagger and a lexico-syntactic pattern matcher) to answer general-knowledge questions based on

[3]The *answer string* is a piece of text (e.g., "*fat*" in Table 1.1) and if the answer is a consecutive text, it is called the *answer span* [88].

the online encyclopedia. Open-domain QA has received much attention since 1999, when the QA track was first introduced in Text Retrieval Conference (TREC) competitions by the National Institute of Standards and Technology (NIST) [16, 77]. In the QA track of TREC, QA was defined such that the systems were to retrieve small snippets of text that contain the answer to open-domain questions[4]. It has motivated a surge of QA systems [79, 120] and most of them consist of the two-stage working pipeline as shown in Figure 1.1.

After the QA track was introduced in TREC, open-domain QA had generally been built upon knowledge-bases (KBs) as the knowledge source, such as Freebase [9] and DBpedia [4]. However, the drawback of KBs is that these knowledge bases are not complete to be used for answering questions that cover a range of topics [42]. In addition, these KBs are not easy to construct and maintain [130]. Therefore, recent works on open-domain QA have focused on using unstructured text (i.e., raw text), such as Wikipedia articles, newswire articles [17, 27, 125], as there is a large amount of raw text available and generated every day, which can cover different kinds of topics [42]. Furthermore, the unstructured text does not need much human intervention to collect [92], we thus consider the unstructured text as our knowledge source in our work. In addition to KBs and unstructured text, open-domain QA can also retrieve relevant documents based on the whole web [34] and other modalities of knowledge sources, including tables [90], images [3], diagrams [52] and even videos [115], but we only consider unstructured text as our knowledge source as we explained above.

1.1 Motivation

RC-style QA has achieved significant advances in recent years due to the introduction of large-scale QA datasets and the promising learning technologies (e.g., Machine Learning, Deep Learning) [26, 104, 139]. In previous years, the size of the QA datasets was small. For example, QA datasets of TREC 8-12 only contains almost 2.4K QA pairs [124]. However, there have been many large-scale datasets proposed in recent years. For example, Stanford Question Answering Dataset (SQuAD) proposed by Rajpurkar et al. [94] contains 107K QA pairs and Yang et al. [136] proposed a dataset named HotpotQA, which includes 113K QA pairs. These datasets are also called RC datasets [28, 42] since these datasets provide one or more paragraphs containing the reasoning facts and answers to RC models for extracting the answer to the question. The goal of these datasets is to motivate the development of high-performing RC models. Hence, these large-scale datasets can be used to train neural network based RC models and motivate the potential and development of RC models. As a result, RC models have achieved significant success [26, 44, 68] and some of them have outperformed human level accuracy[5].

The significant success of RC models in RC-style QA have motivated NLP researchers to combine the retrieval component with RC models that are used as the answer extraction component to challenge the task of open-domain QA [17, 42, 88, 136]. The QA pipeline DrQA proposed by Chen et al. [17] is the earliest work that applied the neural network based RC model with the retrieval component in the open-domain

[4]QA track in TREC: `https://trec.nist.gov/data/qa.html`.

[5]SQuAD leaderboard: `https://rajpurkar.github.io/SQuAD-explorer/`.

setting [88]. They compared the QA performance by applying DrQA in the task of RC-style QA and the task of open-domain QA. In RC-style QA, DrQA was given a question as well as the paragraph that contains all reasoning facts and the answer to this question. Nevertheless, in open-domain QA, DrQA was only given a question and required to retrieve relevant documents from the large-scale knowledge source Wikipedia using an IR system and then extracted the answer to this question from retrieved documents. DrQA achieved the answer accuracy of 69.5 in RC-style QA and its accuracy decreased to 27.1 in open-domain QA. The work in HotpotQA [136] also showed the answer accuracy drop from 48.38 to 24.68 when the paragraph containing all the reasoning facts and the answer was not given to the RC model directly. In order to extract the correct answer, the open-domain QA system should have an effective retrieval component that can retrieve highly relevant documents. Therefore, the retrieval component plays an important role in open-domain QA.

Further works have been conducted on RC models to narrow down the performance gap between open-domain QA and RC-style QA. Lin et al. [66] proposed DS-QA to filter out those noisy documents retrieved by the retrieval component which do not contain the answer. Dehghani et al. [25] and Wang et al. [126] tried to take advantage of multiple documents retrieved by the retrieval component to combine and reason facts from them for answering the question. Table 1.2 shows an example from the QA dataset HotpotQA. As we can see, the answer can be extracted based on the multiple reasoning facts from two different documents. Furthermore, the dataset SQuAD2.0 [95] motivates an emergence of RC models that are able to distinguish whether a paragraph can be used to answer the question correctly or not [110, 127].

Question: The birthplace of George McCall Theal is a port city of what bay?
Doc1: George McCall Theal (11 April 1837, Saint John, New Brunswick-17 April 1919, Wynberg, Cape Town), was the most prolific and influential South African historian, archivist and genealogist of the late nineteenth and early twentieth century.
Doc2: Saint John is the port city of the Bay of Fundy in the Canadian province of New Brunswick.
Answer: Bay of Fundy

Table 1.2: An example of HotpotQA [136]. Reasoning facts are in underline and the answer is in yellow.

However, IR techniques in the retrieval stage have not received much attention compared to RC models in the answer extraction stage in the pipeline of open-domain QA. Many NLP researchers have placed much more emphasis on the answer extraction stage of open-domain QA to emphasize various aspects of linguistic analysis [135]. Despite there have been an emergence of neural network based IR models in recent years [43, 45, 86, 105], it is difficult for these neural IR models to retrieve relevant documents from a large-scale knowledge source directly[6] since learning features of

[6]Some works have attempted to employ neural IR models in a large-scale knowledge source directly by learning sparse representations of each question and documents [142] or compressing the neural models [39].

all documents exhaustively in a large-scale knowledge source by neural IR models is computationally expensive and time-consuming (e.g., Wikipedia contains more than 5.7M articles) [84, 135]. Therefore, some works [42, 63, 84, 125] attempted to apply neural IR models as a ranker in open-domain QA to re-rank the documents retrieved by the traditional but efficient IR models. For example, Lee et al. [63] used a traditional IR model called TF-IDF (Section 2.2.2), which is a term-based (i.e., lexical matching) IR model. They used TF-IDF to retrieve the initial round of documents and then used a neural ranker to re-rank top-ranked documents retrieved by TF-IDF. This term-based IR model is efficient because it builds a retrieval index (Section 2.2) before conducting the retrieval process and mainly relies on the feature of terms appearing in the question and documents. Hence, the ability of the efficiency of traditional IR models makes them popular for the task of full-text search [109]. The assumption behind the neural ranker is that the top-ranked documents after re-ranking are more relevant to the question than low-ranked documents. The combination of traditional but efficient IR models and neural rankers helps the retrieval component in open-domain QA improve the retrieval performance without losing too much efficiency when retrieving documents from the large-scale knowledge source like Wikipedia [84].

While adding the neural ranker after traditional IR models in open-domain QA has shown some retrieval improvements over the traditional IR models without the neural ranker [42, 63, 84, 125], these previous works did not conduct fine-grained error analysis of traditional IR models based on different questions of QA datasets. More specifically, there are three aspects that previous works did not explore, which are shown in the following:

1. Previous works did not explore whether some different existing QA datasets in the open-domain setting have the same impact on the traditional IR models in the retrieval stage. Some existing datasets contain different questions from each other. For example, a question of dataset SQuAD "*Who was the Super Bowl 50 MVP?*" has a different syntactic structure from a question of dataset HotpotQA "*Where is the company, which designed the Glomar Challenger, based?*". The latter question has an attributive clause "*which designed the Glomar Challenger*".

2. Previous works did not explore the error types of questions of these datasets that have a negative impact on the traditional IR models, namely the reasons for retrieval ineffectiveness of traditional IR models.

3. Previous works did not explore whether the neural rankers they used can alleviate the negative impact on the traditional IR models resulting from the error types of questions of datasets. In addition to the approach of neural rankers, previous works also did not explore whether there are other approaches based on the traditional IR models that can alleviate the negative impact of the error types of questions of datasets.

To develop a better retrieval component of the open-domain QA system, we need to figure out the insights of these three aspects described above. If we attempt to improve the overall performance of an open-domain QA system without knowing what aspects make the pipeline of the open-domain QA system get wrong, we are more likely to fix the trivial things of the pipeline and our efforts might be in vain. Therefore, these three research gaps above inspire our work to do an exploration of them.

1.2 Aim of Research

There have been many works already that focused on the RC models in open-domain QA [25, 66, 126], whereas the retrieval component in open-domain QA is usually treated as a black box. Hence, we will pay our attention on the retrieval component in this thesis, as the retrieval component plays an important role in open-domain QA and it does influence the overall performance of an open-domain QA system (see the retrieval stage in Figure 1.1). The research questions of this thesis are the following:

RQ1: Whether different QA datasets in the open-domain setting have the same impact on traditional IR models that are used as the retrieval component in the retrieval stage?

RQ2: What are the error types of questions of QA datasets in the open-domain setting that have a negative impact on traditional IR models?

RQ3: What are the approaches that can be used to alleviate the negative impact of the error types of questions of QA datasets in the open-domain setting on traditional IR models?

1.3 Thesis Outline

Following the research questions we proposed above, the remainder of this thesis is organized as follows. In Chapter 2, we provide readers with some background knowledge of QA and IR and we also discuss some related works. Subsequently, in Chapter 3, we focus on the retrieval stage of the open-domain QA system and describe the exploratory analysis of datasets that are used in this thesis. In addition, we also present the hypotheses with corresponding methodologies for alleviating the negative impact of QA datasets. In Chapter 4, we then discuss experiments that are conducted to validate the hypotheses we proposed in Chapter 3 and provide the analysis of results based on the performance of the retrieval stage and the answer extraction stage in the pipeline of the open-domain QA system. Finally, in Chapter 5, we give a summary of this thesis and introduce some related future work.

Chapter 2

Related Work

This chapter aims to provide readers with some background knowledge of question answering (QA) and information retrieval (IR), which can facilitate readers to have some initial insights into the work in this thesis. Firstly, we begin with the definition and composition of QA in natural language processing (NLP) (Section 2.1) and we elaborate the reading comprehension (RC)-style QA and open-domain QA. Then, we provide the introduction of different IR techniques (Section 2.2), including traditional IR models and neural IR models.

2.1 Question Answering

Question answering is a discipline of computer science whose goal is to build an automated mechanism to answer questions posed by humans in natural language [16, 47]. QA is a heated research field in NLP which motivates the development of QA datasets and learning technologies [20, 68, 94, 95, 136]. QA can be divided into two different types [30, 35]:

- **Closed-domain QA**: It aims to solve the questions under limited domains (e.g., science or medicine) based on domain-specific knowledge sources or a specific context. In particular, the task of QA based on a specific context is called reading comprehension-style QA [28, 42], which requires an RC system to read the context and answer the question. RC-style QA has gained much attention in recent years [26, 44, 94, 136] and the models used in RC-style QA are also important to the open-domain QA which is another type of QA. Hence, we will give more detailed introduction of RC-style QA in this chapter.

- **Open-domain QA**: It is not restricted to any specific domain and aims to answer questions regarding almost everything based on larger knowledge sources than closed-domain QA (e.g., the web and Wikipedia) [135]. This task is the focus of our work, we thus will elaborate it in this chapter as well.

2.1.1 Reading Comprehension-Style QA

Reading comprehension (RC)-style QA emulates how our humans answer questions based on a text and it requires QA systems to *understand* the question and text from which the question is generated [47]. RC-style QA puts emphasis on text understanding by answering questions as an evaluation of language understanding [16].

Early RC-style QA systems used rule-based approaches to answer questions based on some hand-designed syntactic and semantic rules [99]. Then, this field moved to the machine learning approaches due to the availability of large text material [12, 31]. Machine learning approaches attempted to formulate this task as a *supervised learning problem* based on some feature engineering works [82]. Nowadays, more and more researchers have applied the learning approach based on neural networks (i.e. Deep Learning) in RC-style QA, because this learning technology is able to learn the features of data automatically without hand-designed rules or feature engineering works by domain experts which are time-consuming and expensive [26, 104, 139].

Rule-Based Approach

Most of early RC-style QA systems are logical representations of decision trees. The decision trees are linguistic structures (e.g., grammatical rules) that mimic the way humans understand text [47, 71]. Through decision trees, QA systems are able to find syntactic and semantic clues in the question and text. The performance of these QA systems highly depends on the constant extension of rules in decision trees and all these rules at the very beginning were written by hand [98]. These QA systems require different rule sets that define paths in decision trees to answer different types of questions. For example, the question "*where is the city Delft*" has a different path from the question "*how large is the city Delft*". The former question asks about the *location*, while the latter one asks about the *quantity*. The performance of early RC-style QA systems can be improved with some shallow linguistic processing methods like stemming[1] and part-of-speech tagging[2] [47]. Despite rule-based systems were successful for QA at the beginning, these systems are not applicable to highly volatile questions and extensions to these systems cost too much effort [37, 47].

Machine Learning Approach

Due to the availability of a large amount of data in text material, researchers have put their efforts on machine learning technology to tackle the task of RC-style QA [82]. A large amount of data can be used to train statistical models that learn a function which maps a piece of text and a question into the corresponding answer [16]. Two notable datasets MCTEST [97] and PROCESSBANK [8] were proposed which inspired a surge of machine learning models [81, 102, 122] and most of these models were built over a number of hand-designed linguistic features such as syntactic dependencies, coreference resolution, semantic frames and discourse relations [16]. The performance on QA based on the machine learning approach has been significantly improved over rule-based heuristic approaches [16]. Nevertheless, the performance of the machine learning approach is limited because of some drawbacks of the hand-designed linguistic features. Machine learning models highly depend on existing linguistic tools (e.g., dependency parsers and part-of-speech taggers) to obtain features, whereas some

[1]The process of reducing a word to its root form (e.g., "*playing*" to "*play*").

[2]The process of identifying a word as noun, verb, adverb, etc.

noise exists in this process. In addition, these hand-designed features are difficult and insufficient to construct an effective QA system to simulate human-level performance.

Deep Learning Approach

Thanks to the creation of many large-scale supervised training data [40, 94], deep learning has boosted the recent development of NLP since its ability of learning underlying features of data automatically using neural networks [47, 62]. Unlike rule-based and machine learning approaches, deep learning models do not rely on hand-designed linguistic features and all the features are learned in the end-to-end neural network by deep learning models themselves [40]. This can help models avoid the noise produced by linguistic tools and enlarge the scope of useful features [16]. RC-style QA systems have benefited significantly from deep learning models and some of them have outperformed human level performance [26]. Neural networks are applied in text understanding to learn linguistic features and latent semantic (i.e., the meaning of the text) of text based on the training data. Then, trained neural networks encode the textual text (e.g., word, sentence or paragraph) into a distributed representation which is a vector called *embedding*. Recurrent neural networks (RNNs) [41] have shown promising results on language processing [138] and the process of RNNs can be found in Figure 2.1. The hidden layers of RNNs retain the previous values and this process is similar to the process of human processing and comprehending text from left to right. We memorize the previous meaning of words to cumulatively understand the entire meaning of text [74]. This special property of RNNs has the potential to model the dependencies of words in a long distance (i.e., long sentences) [74, 143], but it also brings a problem of gradient explosion or vanishing resulting in the difficulty of optimization of weights in hidden layers [89]. Hence, some other neural networks based on RNNs were proposed to alleviate this problem such as long short-term memory networks (LSTMs) [41] and gated recurrent units (GRUs) [18].

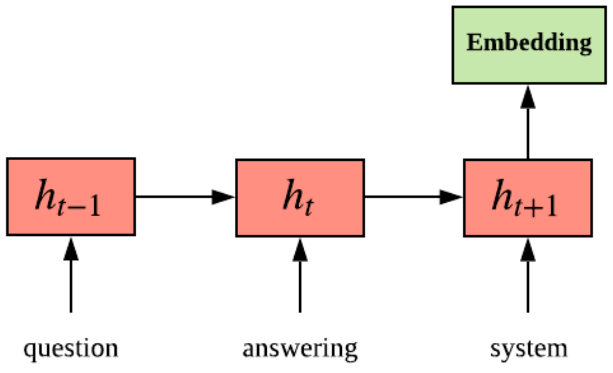


Figure 2.1: The working process of RNNs. Each input word is processed by the hidden layer and the hidden layer maintains previous hidden values to output a text embedding. Input words are generally represented by pre-trained word embeddings (e.g., Glove [91] or Fasttext [50]).

Categories of RC-Style QA
Based on the answer type of the existing RC-style QA datasets, the task of RC-style QA can be divided into four categories [16] and the examples of each category can be found in Table 2.1:

- **Cloze style**: In the cloze-style dataset, some words in the text have been masked and missing and QA systems need to predict them based on the context. Representative datasets are CNN/Daily Mail [40] and Story Cloze Test [80].
- **Multiple choice**: In this case, each question has multiple choices to be chosen and one of them is correct. QA systems need to distinguish the correct one from all the hypothesized answers. Representative dataset is ARC [21].
- **Text span**: In this case, the answer to each question is a span of text (i.e., a word or multiple continuous words) from the corresponding provided passage and this is also called *extractive QA*. QA systems require to predict the starting word and ending word respectively thereby forming an answer span. Representative datasets are SQuAD [94] and HotpotQA [136].
- **Free-form answer**: The answer belonging to this category can be a text without length limitation. Representative dataset is Natural Questions [59].

Evaluation Metrics
In terms of cloze-style questions and multiple-choice questions, the evaluation is definitely straightforward to be done by measuring the accuracy of predicted answer [27, 113] since the answer is chosen from candidate answers. The question whose predicted answer is correct gets credit 1.0 and 0.0 otherwise. Hence, the metric **Accuracy** of all questions can be derived from Equation 2.1.

$$Accuracy = \frac{\#\ ques\ correct\ predicted\ answer}{\#\ total\ ques} \tag{2.1}$$

For text-span questions whose answer is string(s), we need to compare the predicted string(s) with the ground truth answer string(s) (i.e., the correct answer). RC-style QA task generally uses evaluation metrics **Exact Match (EM)** and **F1 score (F1)** proposed by Rajpurkar et al. [94] for text-span questions [104, 116]. EM assigns credit 1.0 to questions whose predicted answer is exactly the same as the ground truth answer and 0.0 otherwise, so the computation of EM is the same as the metric Accuracy but for different categories of RC-style QA. F1 measures the average word overlap between the predicted answer and the ground truth answer. These two answers are both considered as bag of words with lower cases and ignored the punctuation and articles "*a*", "*an*" and "*the*". For example, the answer "*The Question Answering System*" is treated as a set of words $\{question, answering, system\}$. Therefore, F1 of each text-span question can be computed at word-level by Equation 2.2.

$$F1 = \frac{2 \times Precision \times Recall}{Precision + Recall} \tag{2.2}$$

Cloze style	**Passage**: The ent381 producer allegedly struck by ent212 will not press charges against the " ent153 " host, his lawyer said friday. ent212, who hosted one of the most-watched television shows in the world, was dropped by the ent381 wednesday after an internal investigation by the ent180 broadcaster found he had subjected producer ent193 "to an unprovoked physical and verbal attack." **Question**: producer X will not press charges against ent212, his lawyer says. **Answer**: ent193
Multiple choice	**Question**: What is a worldwide increase in temperature called? **Candidate answers**: (A) greenhouse effect (B) global warming (C) ozone depletion (D) solar heating **Answer**: (B)
Text span	**Passage**: Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champions Denver Broncos defeated the National Football Conference (NFC) champions Carolina Panthers, 24–10. **Question**: Which NFL team represented the AFC at Super Bowl 50? **Answer**: Denver Broncos
Free-form answer	**Question**: Where is the bowling hall of fame located? **Long answer**: The World Bowling Writers (WBW) International Bowling Hall of Fame was established in 1993 and is located in the International Bowling Museum and Hall of Fame , on the International Bowling Campus in Arlington, Texas. **Short answer**: Arlington, Texas

Table 2.1: Examples from each category of RC-style QA based on the answer type. The answer in the text is in yellow .

where *Precision* and *Recall* are computed by Equation 2.3 and Equation 2.4 respectively.

$$Precision = \frac{\#\,\{predicted\ words\} \bigcap \{ground\ truth\ words\}}{\#\ predicted\ words} \tag{2.3}$$

$$Recall = \frac{\#\,\{predicted\ words\} \bigcap \{ground\ truth\ words\}}{\#\ ground\ truth\ words} \tag{2.4}$$

In terms of free-form questions whose answer has no length limitation, most of works [65, 104, 116] use standard evaluation metrics that are employed in natural language generation (NLG) tasks, including BLEU, METEOR and ROUGE[3].

[3]See `https://github.com/sebastianruder/NLP-progress/blob/master/english/question_answering.md`.

2.1.2 Open-Domain QA

Open-domain QA systems are not restricted to any specific domain or any specific question type and they aim to answer questions based on large knowledge sources such as the World Wide Web, which covers a large number of different topics [135]. Open-domain QA combines the researches from different fields like IR, information extraction and NLP [16] and it mainly consists of two stages as we introduced in Chapter 1 [17, 28] . The two-stage pipeline of the open-domain QA system can be found in Figure 1.1.

The research in open-domain QA has increased since 1999 when the QA track was first introduced in Text Retrieval Conference (TREC) competitions by the National Institute of Standards and Technology (NIST) [16, 77]. The first TREC competition on QA track[4] provided 200 questions and a set of documents where the answer to the question exists. At that time, QA systems were expected to retrieve small snippets of a text containing the answer to the question and it inspired some developments of QA systems [79, 120]. In the QA track of next TREC competition (i.e., TREC-9) held in 2000, the number of questions and the size of document collection were increased (the number of questions increased by 693). In TREC-11 held in 2002, the QA systems were asked to give an exact short answer to the question and the later TREC competitions of QA track introduced different types of questions (e.g., list type questions and factoid questions) and increased the diversity of question topics. Hence, TREC competitions on QA track progressed with increasing size of document collection, the complexity of questions and strictness of evaluation strategies.

With the rise of information online, there is a large amount of data that may provide useful information to the user and it has motivated the development of knowledge bases (KBs) in recent years such as Freebase [9] and DBpedia [4]. Some QA datasets have been proposed based on these KBs such as WebQuestions [7] and SimpleQuestions [11] and most of the models on these datasets are either based on semantic parsing or information extraction techniques [137]. Nevertheless, KBs based open-domain QA systems have inherent drawbacks: (1) KBs are not complete, so they cannot cover a large range of knowledge sources [130]; (2) the knowledge in KBs is arranged in fixed schemas for information extraction, so it is expensive to construct and maintain [42]. Therefore, the limitations of KBs have motivated researchers to return to the original setting of open-domain QA which is based on raw text [16].

Open-domain QA based on raw text is also called corpus-based[5] approach, where QA systems look for the answer in the unstructured text corpus (i.e., raw text) [34]. This approach alleviates the cost to build and maintain KBs by taking advantage of the availability of a large amount of text on the web [42].

There are also some open-domain QA systems using hybrid resources to look for the answer, including both unstructured text corpus and structured knowledge bases such as AskMSR from Microsoft [13] and DeepQA from IBM [34]. IBM's DeepQA has received much attention after its victory at the TV game-show *Jeopardy!* in 2011 and it is a sophisticated and complicated QA system that relies on unstructured information and structured information to get the answer.

[4]See `https://trec.nist.gov/data/qa/t8_qadata.html`

[5]Corpus refers to a body of raw text from a group of documents [72].

There are other modalities of knowledge sources, including tables [90], images [3], diagrams [52] or even videos [115], but we only focus on unstructured text-based knowledge source in this thesis as we described in Chapter 1.

As the significant success has been achieved by neural network based RC models in RC-style QA, researchers have attempted to apply neural network based RC models in open-domain QA to implement the answer extraction stage of open-domain QA (see Figure 1.1) [17, 42, 88, 136]. For instance, Chen et al. [17] proposed an open-domain QA system named *DrQA* and the working pipeline is shown in Figure 2.2. It consists of a *Document Retriever* and a *Document Reader*. The Document Retriever is based on the traditional IR model *TF-IDF* (will be described in Section 2.2.2), upon the unstructured text corpus *Wikipedia*. The Document Reader is a neural network based RC model and it will be used in our experiments (will be described). Despite the Document Reader of DrQA achieves EM of 69.5 on dataset SQuAD, this performance drops to 27.1 in open-domain setting when the relevant paragraph containing the answer is not given to the Document Reader.

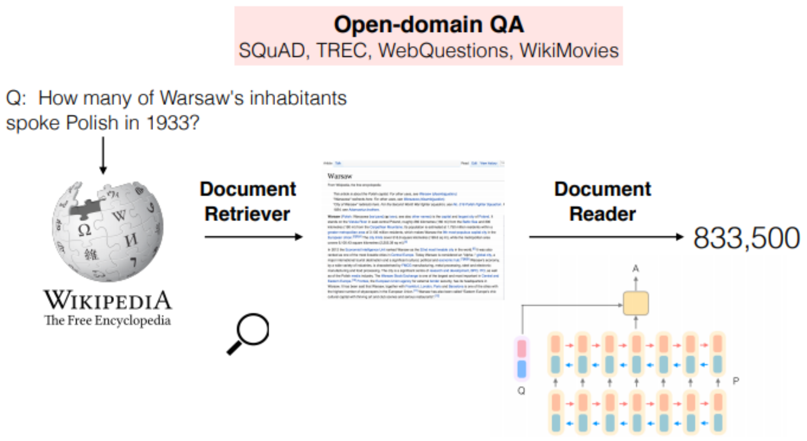

Open-domain QA
SQuAD, TREC, WebQuestions, WikiMovies
Q: How many of Warsaw's inhabitants spoke Polish in 1933?
WIKIPEDIA
The Free Encyclopedia
Document Retriever
Document Reader
833,500
A
Q
P

Figure 2.2: The overview of an open-domain QA system DrQA. Figure is copied from [17].

2.2 Information Retrieval

The definition of IR in academia is *finding material (usually documents) of an unstructured nature (usually text) that satisfies an information need from within large collections (usually stored on computers)* [72]. A common practical example of IR in our daily lives is using web commercial search engines (e.g., Google and Bing) or search the emails. When engaging in the web search engine, the user initiates a search with a set of keywords as a query to convey the information need and then the query is executed by the IR system built in the search engine. Finally, the IR system returns ranked documents (i.e., web pages) to the user according to the relevance[6]. The overview of the retrieval process is displayed in Figure 2.3.

[6]A document is relevant if it contains the information that the user is looking for.

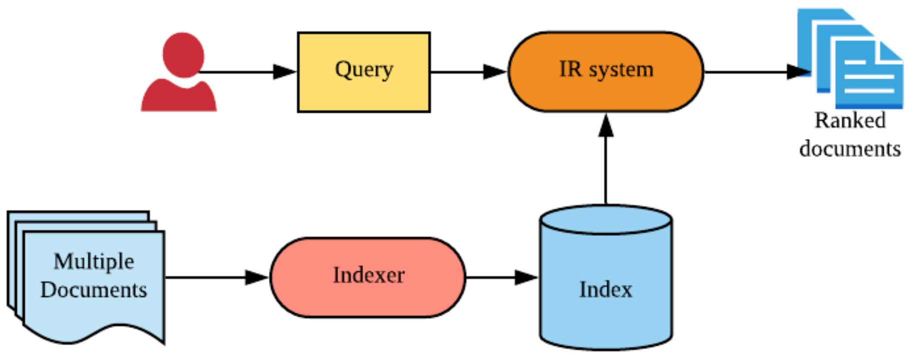


Figure 2.3: The overview of the IR process.

2.2.1 Preliminaries

This part introduces some background knowledge of IR which is fundamental to traditional IR models. There is a large number of documents on the web that can be used for retrieval and a way to avoid linearly scanning all the documents for a query is to **index** these documents in advance. Manning et al. [72] gives a detailed description of how to index documents and what the purpose of the index is. They consider a play of Shakespeare's as a document and judge whether it contains each word out of all words in all plays (about 32,000 words). As a result, a binary term-document *incidence matrix* can be derived as in Figure 2.4.

	Antony and Cleopatra	Julius Caesar	The Tempest	Hamlet	Othello	Macbeth	...
Antony	1	1	0	0	0	1	
Brutus	1	1	0	1	0	0	
Caesar	1	1	0	1	1	1	
Calpurnia	0	1	0	0	0	0	
Cleopatra	1	0	0	0	0	0	
mercy	1	0	1	1	1	1	
worser	1	0	1	1	1	0	
...							

Figure 2.4: A term-document incidence matrix and each entry (t,d) is 1 if the document *d* in column contains the word *t* in row, and is 0 otherwise. The figure is copied from [72].

The terminology **term** is the indexed unit and it is usually a word[7]. **Document** is the indexed retrieval unit over which we can decide to build our retrieval system and it can be based on each paragraph, chapter or even the whole content of a book.

[7]Sometimes it is not a word such as "*Hong Kong*", so using *term* to indicate the indexed unit instead of *word*.

Collection indicates a group of documents over which the retrieval system performs and it is sometimes referred to a **corpus** (a body of raw text) [72].

As we can see from the term-document incidence matrix, if a document only contains 1,000 unique words, its column is filled with a large number of entries 0 since the number of dimensions of each column equals to the number of unique words in all documents. Hence, the matrix is too sparse to fit the memory of computers and the **inverted index** was introduced to solve this problem, which can be found in Figure 2.5. The inverted index is an index that maps a term to documents that contain it.

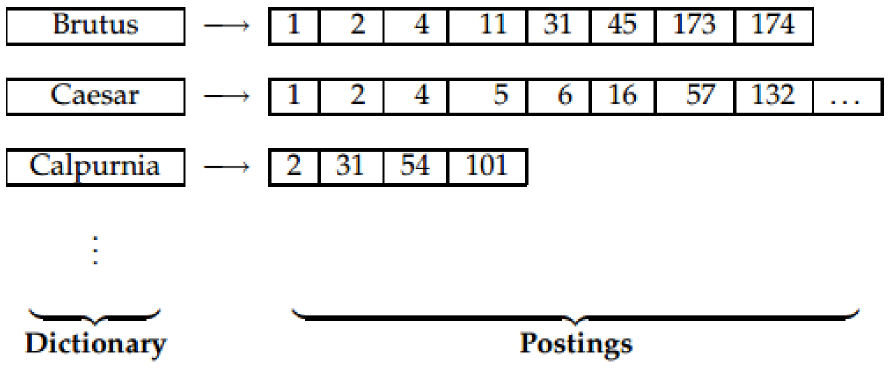


Figure 2.5: The inverted index. The figure is copied from [72].

All terms are kept in *dictionary* which is a type of data structure and each item in the *postings* indicates the ID of the document that contains the term. The inverted index can be built based on the major steps below [72]:

1. Choose the retrieval document unit (e.g., each paragraph or chapter of a book) and collect documents to be indexed.

2. Tokenize the text and transform the text of each document into a list of tokens. For example, a piece of text "*Lily went to the park with her friends*" is transformed into a set of tokens {*Lily, went, to, the, park, with, her, friends*}.

3. Implement linguistic preprocessing (e.g., lower case, stopwords[8] removal and stemming) so that the tokens are normalized before becoming indexed terms. In terms of the example in step 2, a set of tokens {*Lily, went, to, the, park, with, her, friends*} can be preprocessed into tokens {*lily, go, park, friend*}.

4. Assign a unique document number (i.e., document ID) to each document and then index documents that each term appears in by creating an inverted index with a *dictionary* and *postings* as shown in Figure 2.5. For an efficient retrieval, terms in the dictionary are sorted alphabetically and document IDs in postings are sorted from smallest number to largest number.

The inverted index described above is a binary index which means we only can know whether a document contains the term or not. There are other methods that can

[8]Some common words are of little value in helping matching documents to the user's query (e.g., *of*, *to*, *in*).

construct different indexes with more statistics for different IR models, such as term frequency, document frequency and positions the term appears in [24].

2.2.2 Traditional IR Models

Traditional IR models mainly contain Boolean retrieval models, vector space models, probabilistic retrieval models, language modeling and relevance models [72]. As we can see the *term-document incidence matrix* in Figure 2.4, it gives us an intuition about how Boolean retrieval models work, which just indicates that a document either contains or does not contain a query. However, the Boolean retrieval model is hardly used in open-domain QA since it returns a set of documents just satisfying the Boolean queries[9]. For example, a Boolean query "*Brutus and Caesar not Calpurnia*" can be executed by the Boolean retrieval model and this model returns a collection of documents that contain keywords "*Brutus*" and "*Caesar*" but not "*Calpurnia*". The Boolean query is not in the form of natural language and documents returned are not ranked by relevance. This makes it hard for RC models to extract a correct answer to the question in the open-domain setting, as RC models tend to extract the answer from top-ranked documents retrieved by the retrieval component [17]. We thus introduce other traditional IR models that can be deployed in open-domain QA.

Vector Space Models

It is important for a retrieval model that can rank matching documents to a query according to their relevance [72]. A vector space retrieval model is an IR model that can implement the ranking based on the relevance score computed for each document. It treats each document as a vector of term weights and computes a score between the query and each document. Before introducing the process of computing the relevance score by the vector space model, we give a description of how to construct a vector for each document and what each entry in the vector is.

As we described above, the Boolean retrieval model only cares about whether or not a query term is present in a document, but a document that contains a term several times should be more important and have a higher relevance score compared to the one that only has this term once [69]. Toward this, a *weight* can be assigned to each term in a document that depends on the number of occurrences of the term in the document and this weighting scheme is called **term frequency (TF)**, denoted by $tf_{t,d}$ [72]. The vector space model considers the text in the query as free text query, without any Boolean retrieval operators, so the query and the document can be treated by vector space model as **bag of words**. In this way, the vector space model ignores the ordering of terms in the query and the document, but the number of occurrences of each term is retained. However, not all terms are equally important when we evaluate the relevance between a query and documents. For example, the article "*the*" may appear in every document many times and it has little value to help evaluate the relevance of a document to a query. This problem can be solved by *stopwords removal*, which we described in the construction of the inverted index. Another scenario is that a collection of documents on the topic "*information retrieval*" is likely to have the term *retrieve* in every document, but it is not a stopword. Hence, the idea to scale down the term

[9]Combining keywords with relational operators (e.g., *and*, *or*).

weights of this type of terms is by **inverse document frequency (IDF)** and **document frequency** or **collection frequency** is referred to the number of occurrences of a term in the collection, denoted by df_t. Inverse document frequency is defined as Equation 2.5, where *N* indicates the total number of documents in the collection.

$$idf_t = log\frac{N}{df_t} \tag{2.5}$$

Therefore, a term *t* can be assigned a weight in a document *d* by **TF-IDF** weighting scheme using Equation 2.6. In fact, this formula is a general one and has different variants of tf weight and idf weight, such as sublinear TF scaling and maximum TF normalization [1, 72].

$$tf - idf_{t,d} = tf_{t,d} \times idf_t \tag{2.6}$$

After introducing the TF-IDF weighting scheme, we can consider each document as a **document vector** in a vector space, in which each entry is a TF-IDF weight of each term in the dictionary of the inverted index. For terms in the dictionary that do not appear in a document, the TF-IDF weights of these terms in this document vector are 0. Some fabricated examples of document vectors can be found in Table 2.2. As we can see from the examples, the document vectors of *Doc1* and *Doc3* are $\overrightarrow{V}(Doc1) = (16, 8, 25, 6, ...)$ and $\overrightarrow{V}(Doc1) = (20, 14, 10, 0, ...)$ respectively and *Doc3* does not contain the term "*document*". The query can be considered as a vector as well by considering the query as a short document and we thus can build a **query vector**. Consequently, we can compute the relevance score for each document to the query based on the relation between the document vector and query vector and the relevance score can be used for ranking. A standard way for quantifying the relation between two vectors is to compute their **cosine similarity** [108], which is given in Equation 2.7, where *q* is a query and *d* is a document.

$$sim(q,d) = \frac{\overrightarrow{V}(q) \cdot \overrightarrow{V}(d)}{\left|\overrightarrow{V}(q)\right| \left|\overrightarrow{V}(d)\right|} \tag{2.7}$$

	Doc1	Doc2	Doc3	...
information	16	12	20	...
retrieval	8	10	14	...
relevance	25	16	10	...
document	6	8	0	...
...	...	...	...	...

Table 2.2: Some fabricated examples of document vectors using TF-IDF weighting.

Probabilistic Models

The probabilistic approach was proposed to use probability theory to reason about the uncertainty of the relevance of a document to a query [72] since no retrieval system can predict with certainty that a document is either relevant or irrelevant to a query [73, 101]. Intuitively, probabilistic models can rank documents by their estimated probabilities of relevance with respect to a query, so it models the probability $P(d|q)$ of the relevance of a document d with respect to a query q [72].

Probabilistic models originally did not show superiority than other traditional IR models (e.g., TF-IDF) and this situation changed when the **BM25** weighting scheme was proposed in 1990s [72]. The original probabilistic models such as *Binary Independence Model (BIM)* did not take term frequency and document length into consideration [131], while the BM25 weighting scheme, also called **Okapi BM25 weighting**, was implemented to focus on these aspects in text [48]. BM25 can be computed in Equation 2.8, where $tf_{t,d}$ is term frequency of term t in the document d, L_d is the length of document d and L_{ave} is the average document length of whole documents in the collection. k_1 and b are hyperparameters which need to be tuned manually. Specifically, k_1 ($k_1 \geq 0$) indicates the scaling to term frequency and BM25 is a binary model with ignoring term frequency if k_1 is 0, and k_1 has a large value which indicates to use almost raw term frequency. b ($0 \leq b \leq 1$) determines the scaling by document length and if it is 1 which indicates fully scaling down the term weight by document length and 0 means no scaling by document length.

$$BM25(q,d) = \sum_{t \in q} log \left[\frac{N}{df_t} \right] \cdot \frac{(k_1 + 1)tf_{t,d}}{k_1((1-b) + b \times (L_d/L_{ave})) + tf_{t,d}} \tag{2.8}$$

Language Modeling

Unlike probability models that model the probability $P(d|q)$ of relevance of a document d with respect to a query q, language modeling approach aims to build a **language model** M_d from each document d and ranks documents based on the probability of generating the query $P(q|M_d)$ with their own language models [72, 78]. Query terms have a probability distribution of generation in a document and the likelihood of generating a query term by a document can be estimated by the probability of successfully sampling this query term from all terms in this document, which is the *maximum likelihood estimation* (MLE) of this query term in this document [78]. The language model M_d of a document d stores the probability distribution of query terms and it can be used to estimate the probability of query generation $P(q|M_d)$ by the product of each query term's generation probability.

The basic approach to use language models in IR is **query likelihood (QL)** model [72] and it computes the probability of relevance of a document $P(d|q)$ based on the *Bayes rule* in Equation 2.9. $P(q)$ indicates the probability of a query proposed by the user, so it is the same for all documents. The prior probability of a document $P(d)$ is often considered as uniform distribution across all documents [72], hence the probability of relevance of a document $P(d|q)$ only depends on the value of $P(q|d)$, which is equivalent to the query generation probability based on the document language model $P(q|M_d)$.

$$P(d|q) = \frac{P(q|d)P(d)}{P(q)} \tag{2.9}$$

Query generation probability based on the document language model can be estimated in Equation 2.10 ($\hat{P}$ means estimated probability), where M_d is the language model of a document d, $tf_{t,d}$ is the term frequency of a query term t in the document d and L_d indicates the number of all tokens in the document d. This equation is the product of all query terms' generation probability based on the document language model and the value of it is positive only when the document contains all query terms. If the document does not have one of the query terms, the value of the equation becomes 0, which means the document has nothing to do with the query. However, some terms do not appear in the document, but they are possible terms for the user's information need [72]. To avoid this problem, we should not only consider a query term that is generated by a document, but also by all documents in the collection. One of this *smoothing* approaches called **Jelinek-Mercer smoothing** can be found in Equation 2.11. In this smoothing approach, $0 < \lambda < 1$ and M_c is a language model built on the entire documents in the collection.

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{MLE}(t|M_d) = \prod_{t \in q} \frac{tf_{t,d}}{L_d} \tag{2.10}$$

$$\hat{P}(t|d) = (1-\lambda)\hat{P}_{MLE}(t|M_d) + \lambda \hat{P}_{MLE}(t|M_c) \tag{2.11}$$

Pseudo Relevance Feedback

In IR community, a user's query expression sometimes might be too short or simple to indicate the information need of the user, so a user's original query might not be sufficient to retrieve the information that the user is looking for [5, 36]. As a result, researchers have been motivated to focus on the query formulation and the refinement mechanism such as query suggestion and query expansion [5, 36, 51].

Based on users' relevance feedback is one of the approaches[10] to implement the mechanism of query expansion [51, 103]. Relevance feedback is to build interactions between users and the retrieval system to improve the final retrieval results [72]. A user firstly issues a query and the retrieval system returns the initial round of retrieved documents. Then, the user requires to pick out some relevant or non-relevant documents from the initial round of retrieved documents which can help this retrieval system do a query refinement. Subsequently, the second round of retrieved documents after revising the original query are returned. The process of relevance feedback can go through one or more iterations like this.

Pseudo relevance feedback (PRF) is one of the types of relevance feedback[11] [108] and it replaces the user's intervention by an automatic process. PRF firstly employs retrieval models (e.g., BM25 and QL) to retrieve a set of initial documents and then assumes $top-k$ ranked documents are relevant, so it does not require users to pick out relevant documents. Relevance-based language model (usually called **Relevance Model** or **RM**) is an approach based on PRF to do query expansion which was proposed by Lavrenko and Croft [61] and it has four variants: RM1, RM2, RM3 and RM4. In particular, RM3 has been used in many IR related researches and it is the most popular one [14, 64, 100].

RM3 is built on RM1, so we will introduce RM1 before RM3. The process of RM1 consists of following steps[12]:

1. RM1 assigns a weight to $top-k$ relevant documents based on their query generation probabilities in Equation 2.12, which is the same as the formula of *query*

[10]Other automatic query expansion approaches are re-weighting query terms and using external sources (e.g., thesaurus).

[11]Explicit feedback and implicit feedback are other two types.

[12]See `http://people.cs.vt.edu/~jiepu/cs5604_fall2018/10_qm.pdf`.

likelihood model in Equation 2.10.

$$\hat{P}(q|M_d) = \prod_{t \in q} \hat{P}_{MLE}(t|M_d) \tag{2.12}$$

In this equation, the query generation probability is the product of generation probability of each query term t based on the relevant document language model M_d.

2. Compute the probability of a term w that is expanded to the original query based on the language model of the relevant document M_d by $\hat{P}_{MLE}(w|M_d) \cdot \hat{P}(q|M_d)$.

3. Normalize the probability of all terms that are expanded to the original query in relevant documents in Equation 2.13, where D_R is the collection of $top-k$ relevant documents and W_E are the set of terms in relevant documents that are expanded to the query.

$$\hat{P}_{RM1}(w|q,D_R) = \frac{\sum_{d \in D_R}(\hat{P}_{MLE}(w|M_d)\hat{P}(q|M_d))}{\sum_{w \in W_E}\sum_{d \in D_R}(\hat{P}_{MLE}(w|M_d)\hat{P}(q|M_d))} \tag{2.13}$$

In terms of RM1, the term which has a larger value of $\hat{P}_{RM1}(w|q,D_R)$ is expanded to the original query and then the retrieval system re-runs the revised query to retrieve new documents. Sometimes, the term appears in the original query does not have large value of $\hat{P}_{RM1}(w|q,D_R)$ which seems risky and problematic, because query terms are supposed to be more important than some other terms appearing in relevance feedback documents to the user's information need [6, 72]. Hence, RM3 was proposed to alleviate this *query drift* problem with the linear interpolation of the language model of original query q [6]. RM3 can be found in Equation 2.14, where M_q is the language model of the original query and λ $(0 < \lambda < 1)$ is a hyperparameter to indicate how much weight is assigned to the query language model that influences the final probability of the expanded term w.

$$\hat{P}_{RM3}(w|q,D_R) = \lambda \cdot \hat{P}_{MLE}(w|M_q) + (1-\lambda) \cdot \hat{P}_{RM1}(w|q,D_R) \tag{2.14}$$

In addition to the hyperparameter λ, two more hyperparameters also can influence the performance of RM3 in retrieval effectiveness and they are:

- *fbDocs*: the number of relevance feedback documents.
- *fbTerms*: the number of terms that can be expanded to the original query from relevance feedback documents.

2.2.3 Neural IR Models

As we described in Section 2.1.1, Deep Learning has boosted the development of neural RC models which achieve significant success in RC-style QA, as deep neural networks can learn features of input data automatically [47, 62]. This capability has inspired the development of deep neural IR models as well [23] and with this capability, neural IR models can learn features from raw text of the query and document automatically. Hence, many neural IR models have been proposed to deal with the relevance ranking problem of documents by only taking the textual data of the query and document into account [38, 45, 86].

Neural IR models aim to determine the relevance score of a document to a query. According to Hu et al. [43], Pang et al. [86], this process can be formulated in Equation 2.15, where the relevance $Rel(q,d)$ relies on the mapping functions Ψ and Φ of a query q and a document d respectively as well as the scoring function F. The mapping functions of the query and document can map the text of them into a query vector and a document vector (i.e., distributed representation described in Section 2.1.1) and the scoring function can model the relation between these two vectors. Existing neural IR models thus can be categorized into two types that focus on mapping functions and the scoring function respectively and they are: **representation-focused models** and **interaction-focused models** [87].

$$Rel(q,d) = F(\Psi(q), \Phi(d)) \tag{2.15}$$

Representation-Focused Models

Representation-focused models concentrate on building a good representation of the query and document by a deep neural network, so mapping functions Ψ and Φ are paid more attention to and relatively complex than the scoring function F [87]. In general, the model architecture of representation-focused models can be viewed as a Siamese (i.e., symmetric) architecture over the text inputs (i.e., input query and input document) [10, 87], as displayed in Figure 2.6a. The representative neural IR models belonging to the representation-focused model are Deep Structured Semantic Model (DSSM) [45], Convolutional Deep Structured Semantic Model (C-DSSM) [105] and ARC-I [43]. In DSSM, as shown in Figure 2.7, the mapping functions Ψ and Φ are built on feed forward networks (i.e., multi-layer non-linear projection) for learning to map the query and document into representations and the scoring function F uses cosine similarity (Equation 2.7) to compute the relevance score over the query representation and document representation.

Interaction-Focused Models

The architecture of interaction-focused models can be found in Figure 2.6b. As we can see from this architecture, interaction-focused models primarily build the local interactions between the query representation and document representation which are generated based on relatively simple mapping functions. Some interaction-focused models [43, 87] use pre-trained word embeddings (e.g., Glove [91] or Fasttext [50]) instead of using deep neural networks to generate high-level representations of the query and document based on the input text . Then, interaction-focused models make use of deep neural networks to learn complex matching signals based on their local interactions and finally aggregate these matching signals to derive the relevance score between the query and document [87, 145]. The representative neural IR models belonging to the interaction-focused model are MatchPyramid [86], ARC-II [43] and MV-LSTM [121]. For example, in ARC-II, as displayed in Figure 2.8, mapping functions Ψ and Φ map two sentences into representations which are the sum of word embeddings of the words they contain. Then, local interactions are performed by a 1D convolution over each pair of words from these two sentences, which is followed by 2D convolution and 2D max-pooling to operate more complex matching signals. Consequently,

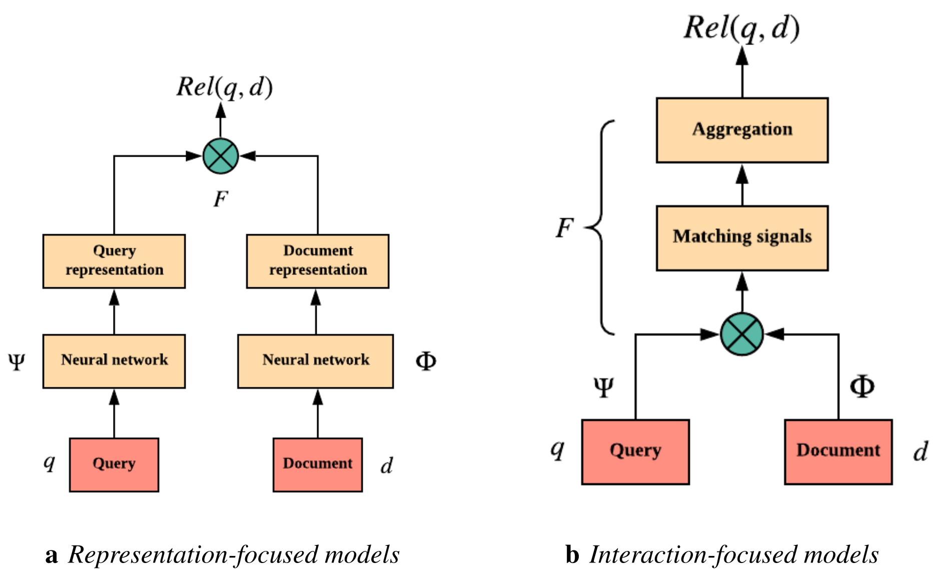


Figure 2.6: Two categories of Neural IR models.

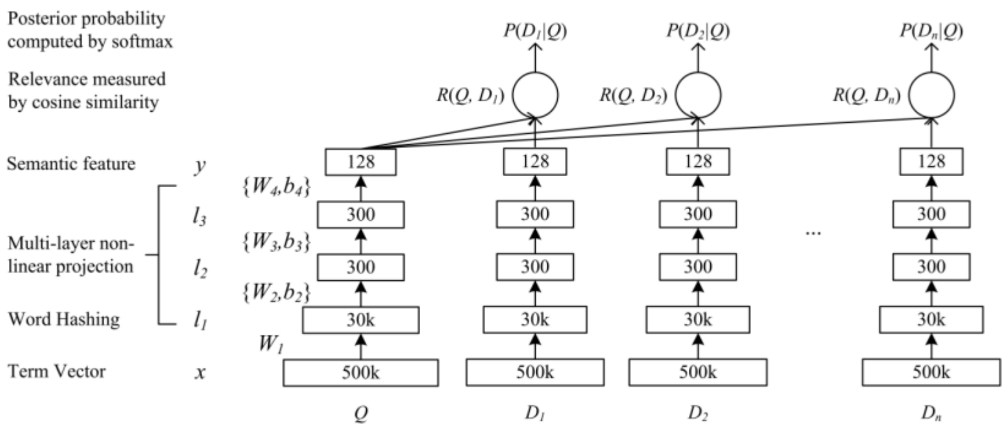


Figure 2.7: The architecture of Deep Structured Semantic Model (DSSM) [45], which belongs to the representation-focused model. Figure is copied from [45].

all matching signals go through a multi-layer perceptron (MLP) to obtain the matching degree between input sentences.

Neural IR Models as A Ranker

Traditional IR models discussed in Section 2.2.2 can retrieve documents to a query from a large document collection in an efficient way based on the index they build in advance [72, 109]. Traditional IR models construct an index with indexing each term and each document, so traditional IR models are also considered as *term-based IR models* which are widely used in full text search [109].

Many works [42, 63, 84, 125] deploy neural IR models to subsequently re-rank a number of top-ranked documents retrieved from a large document collection (i.e., the

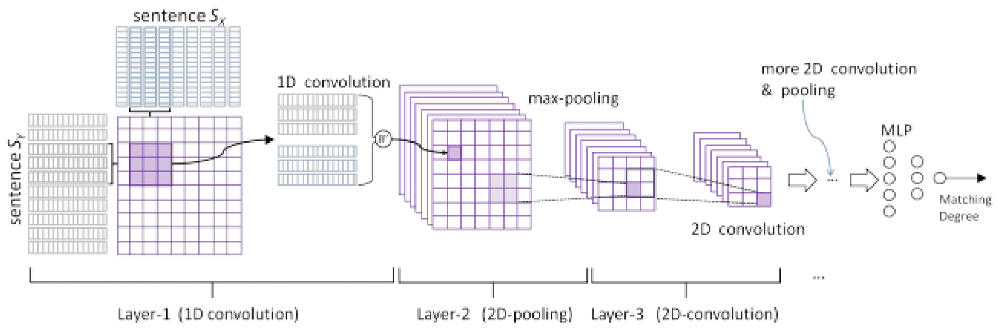


Figure 2.8: The architecture of ARC-II [43], which belongs to interaction-focused models. The figure is copied from [43].

large-scale knowledge source) by traditional but efficient IR models. Therefore, neural IR models can be considered as an *extension* of the original ranking process and this process is composed of two steps as shown in Figure 2.9:

1. Traditional IR models (e.g., TF-IDF, BM25) retrieve a number of top-ranked documents from a large document collection.

2. Neural IR models as the ranker to re-rank these top-ranked documents with respect to the input query to obtain the final ranked documents.

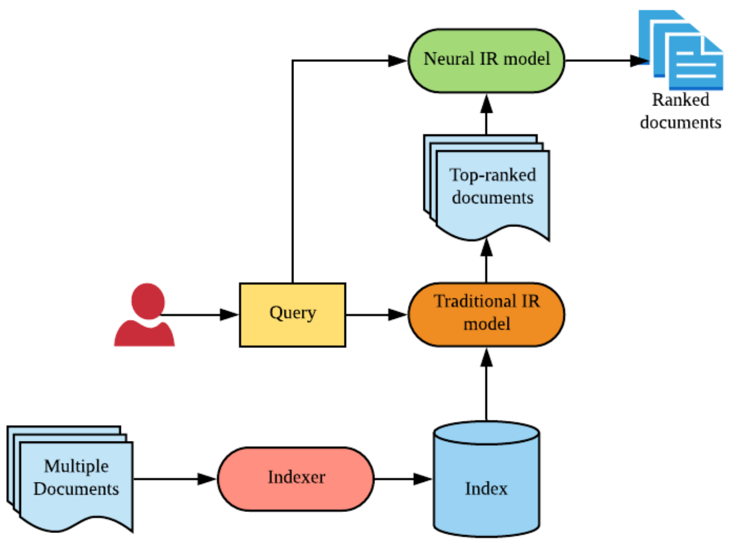


Figure 2.9: The overview of the retrieval process with the neural IR model as the ranker for re-ranking top-ranked documents retrieved by the traditional IR model.

Chapter 3

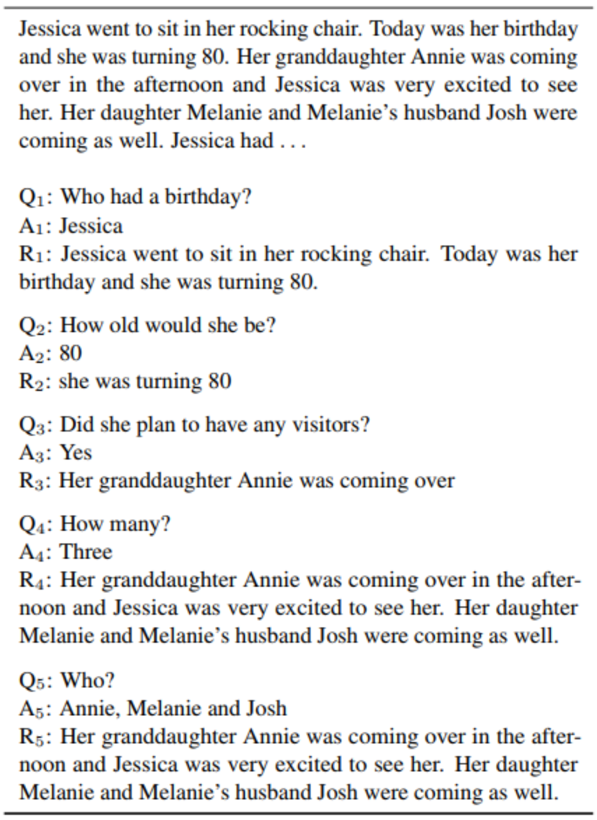
Exploratory Analysis of SQuAD, HotpotQA and TriviaQA

The goal of this chapter is to conduct an analysis of three QA datasets SQuAD [94], HotpotQA [136] and TriviaQA [49] in the task of open-domain QA based on traditional IR models. More specifically, we present the comparison of existing QA datasets and select three datasets that are suitable in our work (Section 3.1). We then describe the retrieval test on these three datasets (Section 3.2), which is followed by the analysis and conclusion of retrieval results as well as our answers to **RQ1** and **RQ2** (Section 3.3). Subsequently, we propose our hypotheses for alleviating the negative impact of error types of questions on traditional IR models (Section 3.4) and we finally introduce specific methodologies that are used to implement hypotheses (Section 3.5).

3.1 Datasets Selection

In NLP community, there have been a large range of QA datasets covering different aspects of QA, such as QA datasets for reading comprehension [40, 94], conversational QA [96], multimodal comprehension [132], and open-domain QA [28, 32]. Nevertheless, not all types of the QA datasets are applicable in our work. For example, the conversational QA dataset CoQA proposed by Reddy et al. [96] focuses on the conversational scenario and some questions are related to the information in history, so this type of QA dataset is different from datasets of RC-style QA and open-domain QA. An example of CoQA can be found in Figure 3.1. As we can see from Figure 3.1, the question Q_2 uses the pronoun "*she*" to refer to "*Jessica*", which is the answer to the question Q_1.

Therefore, we compare some popular QA datasets focusing on the task of RC-style QA and the task of open-domain QA based on some prominent characteristics. The comparison can be seen in detail in Table 3.1. As we can see from Table 3.1, we compare some QA datasets based on the number of questions, the source of questions, the source of retrieved documents (i.e., the knowledge source), the form of questions and the answer type. Considering the definition of open-domain QA (Section 2.1.2) and the knowledge source (i.e., unstructured text) we focus on in our work, we choose QA datasets SQuAD, HotpotQA and TriviaQA in our work and the reasons are the following:



Jessica went to sit in her rocking chair. Today was her birthday and she was turning 80. Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well. Jessica had . . .

Q_1: Who had a birthday?
A_1: Jessica
R_1: Jessica went to sit in her rocking chair. Today was her birthday and she was turning 80.

Q_2: How old would she be?
A_2: 80
R_2: she was turning 80

Q_3: Did she plan to have any visitors?
A_3: Yes
R_3: Her granddaughter Annie was coming over

Q_4: How many?
A_4: Three
R_4: Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well.

Q_5: Who?
A_5: Annie, Melanie and Josh
R_5: Her granddaughter Annie was coming over in the afternoon and Jessica was very excited to see her. Her daughter Melanie and Melanie's husband Josh were coming as well.

Figure 3.1: A conversation from the CoQA dataset. The figure is copied from [96].

- Questions of these three QA datasets are in the form of natural language.
- These three QA datasets have the same knowledge source (i.e., Wikipedia) which enables traditional IR models to retrieve documents to questions of these QA datasets from the same resource.
- The answer type of these three QA datasets are all text span (answer type are described in Table 2.1).

3.1.1 SQuAD

Stanford Question Answering Dataset (SQuAD) [94] is a reading comprehension dataset including 107K question-answer pairs and each QA pair is provided one ground truth paragraph that can be used to answer the question. QA pairs of SQuAD were generated in a way that the provided paragraph for each question is one of the paragraphs of a Wikipedia article and this paragraph was guaranteed to contain all reasoning facts and the answer to the question, because each QA pair was constructed and extracted based on this paragraph [32, 125]. Specifically, crowd-workers were provided a single paragraph firstly, they then generated QA pairs from this paragraph. Two examples of SQuAD can be found in Table 3.2.

In the open-domain setting, the ground truth paragraph of each QA pair is not provided to the open-domain QA system. Hence, the open-domain QA system requires to

Dataset	# Ques	Source of Ques	Source of Docs	Ques form (Nature)	Answer Type	Used
RACE [60]	870K	English exam	English exam	✓	Mutiple choice	
QAngaroo [129]	53K	Wikipedia/PubMed	Wikipedia/PubMed	✗	Text span	
NewsQA [118]	100K	CNN	CNN	✓	Text span	
MultiRC [53]	6.5k	Science, News, stories, travel guides	Science, News, stories, travel guides	✓	Mutiple choice	
MS MARCO [83]	100K	User logs	Web search (Bing)	✓	Free-form	
NarrativeQA [56]	46K	Movie Scripts, Literature	Movie Scripts, Literature	✓	Free-form	
QUASAR-T [28]	43K	Trivia websites	ClueWeb09	✓	Text span	
SearchQA [32]	140K	Jeopardy	Web search (Google)	✓	Text span	
TriviaQA [49]	95K	Trivia websites	Wikipedia/Web search (Bing)	✓	Text span	✓
SQuAD [94]	107K	Wikipedia	Wikipedia	✓	Text span	✓
HotpotQA [136]	113K	Wikipedia	Wikipedia	✓	Text span	✓

Table 3.1: The comparison of some popular QA datasets focusing on the task of RC-style QA and open-domain QA. Note that in the column of *Ques form (Nature)*, "✓" indicates that the question form of the dataset is natural language and "✗" otherwise.

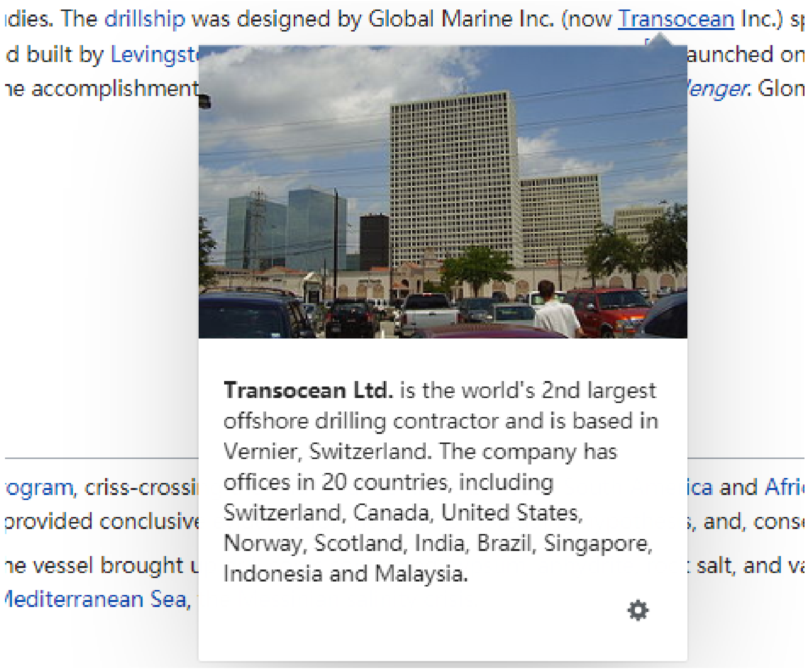
SQuAD
Paragraph: The Broncos took an early lead in Super Bowl 50 and never trailed. Newton was limited by Denver's defense, which sacked him seven times and forced him into three turnovers, including a fumble which they recovered for a touchdown. Denver linebacker Von Miller was named Super Bowl MVP, recording five solo tackles, 2 sacks, and two forced fumbles. **Question**: Who was the Super Bowl 50 MVP? **Answer**: Von Miller
Paragraph: Harvard is a large, highly residential research university. The nominal cost of attendance is high, but the University's large endowment allows it to offer generous financial aid packages. It operates several arts, cultural, and scientific museums, alongside the Harvard Library, which is the world's largest academic and private library system, comprising 79 individual libraries with over 18 million volumes... **Question**: How many volumes are contained in the library? **Answer**: 18 million

Table 3.2: Two examples of dataset SQuAD. Each QA pair was generated from the provided paragraph, which is one of the paragraphs of a Wikipedia article. The ground truth answer is in yellow.

retrieve relevant documents to the question from Wikipedia by the retrieval component, and the RC model extracts the answer to the question based on retrieved documents. We use the term **SQuAD**$_{open}$ to indicate the open-domain setting of SQuAD, so the term SQuAD indicates the task of RC-style QA on SQuAD which provides the ground truth paragraph of each QA pair to the RC model directly, while the term SQuAD$_{open}$ indicates the task of open-domain QA on SQuAD which needs the retrieval component to retrieve relevant documents from Wikipedia to the RC model. The notation of SQuAD$_{open}$ is the same as that in [63, 125].

3.1.2 HotpotQA

HotpotQA [136] is a reading comprehension dataset containing 113K QA pairs. Despite QA pairs in HotpotQA were also constructed based on the paragraphs of Wikipedia articles which is similar to SQuAD, there are still some differences between these two QA datasets. On one hand, each question of SQuAD was generated based on a single paragraph of a Wikipedia article, while each question of HotpotQA was generated based on two paragraphs[1] of two different Wikipedia articles. HotpotQA connected two Wikipedia articles by the hyperlink appearing in the first paragraph of one of the Wikipedia articles. The hyperlink can be seen in Figure 3.2. On the other hand, the ground truth answer to the question of SQuAD appears in the provided single ground truth paragraph, whereas the ground truth answer to the questions of HotpotQA only appears in one of two paragraphs instead of both paragraphs. Two examples of HotpotQA are shown in Table 3.3.


ıdies. The drillship was designed by Global Marine Inc. (now Transocean Inc.) sı
d built by Levingst ... aunched on
ıe accomplishment ... lenger. Glon

Transocean Ltd. is the world's 2nd largest offshore drilling contractor and is based in Vernier, Switzerland. The company has offices in 20 countries, including Switzerland, Canada, United States, Norway, Scotland, India, Brazil, Singapore, Indonesia and Malaysia.

ogram, criss-crossi ... ica and Afri
provided conclusive ... , and, cons
ıe vessel brought ı ... salt, and vá
lediterranean Sea,

Figure 3.2: The hyperlink in the Wikipedia article is used to generate QA pairs of HotpotQA, so each question was generated based on two paragraphs of two Wikipedia articles.

HotpotQA is composed of three categories of questions and they are described as follows.

(1) **Questions with the bridge entity**. The bridge entity connects the answer and other reasoning facts. As we can see from Table 3.3, the question at the top "*Where is the company, which designed the Glomar Challenger, based?*" contains the bridge entity "*the company*" that is a descriptive phrase. If we want

[1]Each paragraph is the first paragraph of a Wikipedia article, since it contains the main information of this article [136].

HotpotQA
Paragraph A, Glomar Challenger: Glomar Challenger was a deep sea research and scientific drilling vessel for oceanography and marine geology studies. The drillship was designed by Global Marine Inc. (now Transocean Inc.) specifically for a long term contract with the American National Science Foundation and University of California Scripps Institution of Oceanography and built by Levingston Shipbuilding Company in Orange, Texas... **Paragraph B, Transocean**: Transocean Ltd. is the world's 2nd largest offshore drilling contractor and is based in Vernier, Switzerland. The company has offices in 20 countries, including Switzerland, Canada, United States, Norway, Scotland, India, Brazil, Singapore, Indonesia and Malaysia. **Question**: Where is the company, which designed the Glomar Challenger, based? **Answer**: Vernier
Paragraph A, Emma Bull: Emma Bull (born December 13, 1954) is an American science fiction and fantasy author. Her novels include the Hugo- and Nebula-nominated Bone Dance and the urban fantasy War for the Oaks... **Paragraph B, Virginia Woolf**: Adeline Virginia Woolf (25 January 1882 – 28 March 1941) was an English writer, considered one of the most important modernist 20th-century authors and also a pioneer in the use of stream of consciousness as a narrative device. **Question**: Who was born earlier, Emma Bull or Virginia Woolf? **Answer**: Adeline Virginia Woolf

Table 3.3: Two examples of dataset HotpotQA. Each QA pair was generated from two paragraphs of two different Wikipedia articles. The ground truth answer only appears in one of the paragraphs, which is in yellow.

to obtain the answer to this question, we need to figure out the real name of this bridge entity based on other reasoning facts in this question (i.e., "*which designed the Glomar Challenger*"). Then, we can check the Wikipedia article of "*Glomar Challenger*", in which we can find out the real name of this bridge entity namely "*Transocean*". Consequently, by retrieving the Wikipedia article of "*Transocean*", we can get the answer "*Vernier*". The reasoning process can be seen in Figure 3.3.

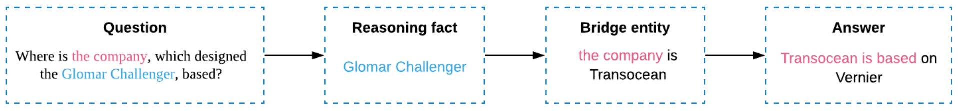

Question
Where is the company, which designed the Glomar Challenger, based?
Reasoning fact
Glomar Challenger
Bridge entity
the company is Transocean
Answer
Transocean is based on Vernier

Figure 3.3: The reasoning process for one question of HotpotQA with bridge entity.

(2) **Comparison questions**. The comparison questions compare two entities in the question. As we can see from Table 3.3, the question at the bottom "*Who*

was born earlier, Emma Bull or Virginia Woolf?" compares two named entities "*Emma Bull*" and "*Virginia*" on the date of birth.

(3) **"yes/no" questions**. This type of questions is not included in our work, as the answer "*yes*" or "*no*" is not extracted from text. For example, the question "*Are Giuseppe Verdi and Ambroise Thomas both Opera composers*" has the answer "*yes*", but the answer is not extracted from this question's ground truth Wikipedia articles of entities "*Giuseppe Verdi*" and "*Ambroise Thomas*".

Furthermore, HotpotQA consists of two benchmark settings:

(1) **Distractor setting**. This setting aims to challenge the RC model to find out the ground truth paragraphs in the presence of noisy paragraphs. The authors of HotpotQA employed the traditional IR model TF-IDF to retrieve eight paragraphs from Wikipedia as *distractors* and mix them with two ground truth paragraphs. Hence, this setting concentrates on the development of RC models.

(2) **Fullwiki setting** (i.e., the open-domain setting). This setting fully tests the performance of the open-domain QA system on HotpotQA, whose retrieval component needs to retrieve relevant documents from Wikipedia to the RC model. Hence, fullwiki setting without providing ground truth paragraphs for the RC model is the setting we employ in our work and we use the term **HotpotQA**$_{fullwiki}$ to denote the task of open-domain QA on HotpotQA.

3.1.3 TriviaQA

TriviaQA [49] is a QA dataset for reading comprehension including 95K QA pairs. As we can see from the comparison of datasets (Table 3.1), questions of TriviaQA were collected from online trivia websites[2], which are different from HotpotQA and SQuAD whose questions were generated from paragraph(s) of Wikipedia articles. Hence, relevant documents of questions of TriviaQA were independently gathered from knowledge sources and these documents are called *distant supervision*.

Distant supervision is commonly used in some QA datasets for reading comprehension [17, 114, 129]. These QA datasets only contain QA pairs without associated documents that can be used to answer the question, so they cannot be used for training neural network based RC models. Thus, some works [32, 83] either deployed commercial search engines (e.g., Google, Bing) to obtain relevant documents or used a procedure of relation extraction [17, 76] to automatically obtain relevant documents from knowledge sources to these QA pairs. These retrieved documents are called distant supervision.

TriviaQA gathered distant supervision from two knowledge sources: Wikipedia and the Web[3]. We include QA pairs whose distant supervision were gathered from Wikipedia in our work for being comparable to SQuAD and HotpotQA which both use Wikipedia as the knowledge source. Retrieved documents from Wikipedia have been demonstrated that they contain reasoning facts and answers to questions of TriviaQA

[2]See `https://www.reddit.com/r/trivia/comments/3wzpvt/free_database_of_50000_trivia_questions/`.

[3]Using commercial search engine Bing for retrieval from the Web.

[49]. We term the task of open-domain QA on TriviaQA as **TriviaQA**$_{wiki}$, which requires the retrieval component to retrieve relevant documents from Wikipedia to the RC model. Two examples of TriviaQA can be found in Table 3.4.

TriviaQA
Document: O'Hare International Airport, typically referred to as O'Hare Airport, Chicago O'Hare, or simply O'Hare, is an international airport located on the far Northwest Side of Chicago, Illinois, 14 miles (23 km) northwest of the Loop business district, operated by the Chicago Department of Aviation and covering 7,627 acres (3,087 ha)... **Question**: In which city would you find O'Hare International Airport? **Answer**: Chicago, Illinois
Document: Anthony Patrick Hadley (born 2 June 1960) is an English singer-songwriter, occasional stage actor and radio presenter. He rose to fame in the 1980s as the lead singer of the New Romantic band Spandau Balletand launched a solo career following the group's split in 1990... **Question**: Tony Hadley was the lead singer with which 1980s new romantic band? **Answer**: Spandau Balletand

Table 3.4: Two examples of dataset TriviaQA. The document for each QA pair was gathered from Wikipedia. The ground truth answer is in yellow.

3.2 Retrieval Test on Datasets

This Section focuses on the retrieval stage of the open-domain QA system on SQuAD, HotpotQA and TriviaQA. We perform a retrieval test on these datasets using traditional IR models TF-IDF, BM25 and QL as the retrieval component to retrieve documents from the knowledge source Wikipedia. The goal of the retrieval test is to analyze the impact on traditional IR models across different QA datasets in the open-domain setting, thereby figuring out answers to **RQ1** and **RQ2** in Section 1.2.

3.2.1 Retrieval Test Setup

Datasets Statistics

The statistics of datasets that we use in the retrieval test can be found Table 3.5. Note that the test sets of SQuAD[4] and HotpotQA[5] are not public, so we use the training set and the development set of these three datasets in our work.

Knowledge source of Datasets

Datasets SQuAD, HotpotQA and TriviaQA all consider Wikipedia as the knowledge source over which the retrieval component performs the retrieval. In our work, we use the same version of Wikipedia in [17, 125, 135] and it is the 2016-12-21 dump

[4]See `https://rajpurkar.github.io/SQuAD-explorer/`.

[5]See `https://hotpotqa.github.io/`.

Dataset	# Train	# Dev	# Test
SQuAD	87,599	10,570	-
HotpotQA	90,564	6,947	-
TriviaQA	61,888	7,993	7,701

Table 3.5: Statistics of datasets used in our retrieval test.

of English Wikipedia[6] which contains 5,075,182 Wikipedia articles with 9,008,962 unique lowercase words. This Wikipedia retains the raw text of each Wikipedia article and abandons all structured and semi-structured data such as lists, tables and figures, as these works [17, 125, 135] and our work concentrate on the task of open-domain QA over the knowledge source of unstructured text.

We note that the Wikipedia used in the original work of HotpotQA only retains the first paragraph instead of the entire raw text of each Wikipedia article. Hence, the size of Wikipedia in our work is much larger than the one used in the original work of HotpotQA.

Evaluation Metric

A high-performing retrieval component of the open-domain QA system can retrieve highly relevant documents containing all reasoning facts and answers to the question. As we described above (see Table 3.1), datasets SQuAD, HotpotQA and TriviaQA all belong to the answer type *text span* whose answers are the span of text extracted from retrieved documents by the RC model. Therefore, if retrieved documents do not contain the answer, the RC model[7] cannot extract the correct answer from these retrieved documents. In terms of datasets whose answer type is text span, many previous works [17, 42, 63, 125, 135] used the evaluation metric **top-k recall (TOP-k)** to evaluate the retrieval performance of the retrieval component of the open-domain QA system and this metric indicates that whether the ground truth answer to the question appears in top-ranked *k* documents retrieved by the retrieval component. The computation of TOP-k can be found in Equation 3.1. The metric TOP-k can also indicate the probability of retrieved top-ranked *k* documents containing the ground truth answer to a question. If an IR model has a higher value of TOP-k than another IR model on the same dataset, it indicates that questions of this dataset are more likely to be answered successfully based on documents retrieved by the IR model with the higher TOP-k. Hence, TOP-k showcases the retrieval effectiveness of an IR model to some extent.

$$top-k\ recall = \frac{\#\ ques\ TOP-k\ docs\ have\ ans}{\#\ total\ ques} \tag{3.1}$$

Traditional IR Models for Retrieval Test

Some previous works in the task of open-domain QA [17, 63, 92, 125, 135] used traditional IR models TF-IDF and BM25 as the retrieval component of the open-domain QA system to retrieve relevant documents from the large-scale knowledge source. We

[6]Obtained from `https://dumps.wikimedia.org/enwiki/`.

[7]Here we do not include the RC model that has the ability of answer generation.

thus employ TF-IDF and BM25 for our retrieval test on datasets SQuAD, HotpotQA and TriviaQA in the open-domain setting. Besides, we also include the traditional IR model query likelihood model (QL) with Jelinek-Mercer smoothing in our retrieval test, as it is also a term-based and efficient IR model that can be used in open-domain QA. Also, QL uses a different retrieval principle from the ones used in TF-IDF and BM25, so we can compare the retrieval performance of QL with that of TF-IDF and BM25 in the open-domain setting across different datasets for fully exploring the impact on the retrieval component across datasets. Detailed introductions of these traditional IR models are described in Section 2.2.2. For reproducibility of our retrieval test and providing clear insights for readers about the difference between the retrieval component in the retrieval test and the modification of the retrieval component in our hypotheses later (Section 3.4), we will describe some implementation details of our retrieval test in this section.

In terms of traditional IR models TF-IDF, BM25 and QL, we employ an open-source IR toolkit **Anserini**[8] to implement them. Anserini was built on **Lucene**[9] project and it aims to bridge the gap between the research on IR and the practice of building real-world search applications [133], many works [134, 135, 140] thus have used Anserini to do explorations on IR for academic research.

In terms of Wikipedia as the knowledge source for traditional IR models in the open-domain setting, we also take advantage of Anserini to index all articles of Wikipedia. We consider each Wikipedia article as the retrieval document unit of traditional IR models. Furthermore, stopwords appearing in each article are not kept and *Porter stemmer* [93] is used to do word stemming in the construction of the index.

Hyperparameter Tuning

As described in Section 2.2.2, traditional IR models BM25 and QL with Jelinek-Mercer smoothing have two hyperparameters (k_1, b) and one hyperparameter (λ) to be tuned respectively. Correctly setting these hyperparameters is important to the good retrieval performance of BM25 and QL [72]. We employ the method *grid search* to tune hyperparameters and it is a traditional way of performing hyperparameter tuning [19, 33].

Grid search tests the performance of all hyperparameters' candidate values and then selects the hyperparameters' values achieving the best performance. For BM25, most experiments [67, 70, 117, 119] have shown that the optimal b is in the range of 0.3-0.9 and k_1 is in the range of 0.5-2.0. In terms of QL with Jelinek-Mercer smoothing, Zhai and Lafferty [144] indicated that the optimal λ depends on the query and the text collection for retrieval and it is around 0.1 for short queries and 0.7 for long queries. Although these experiments showed an empirical analysis of setting a suitable range of candidate values of hyperparameters, the best way to tune hyperparameters satisfying our work is tuning hyperparameters on datasets we use. In the process of hyperparameter tuning, BM25 and QL are both optimized for the evaluation metric **top-10 recall (TOP-10)** and we tune hyperparameters on each training set of SQuAD, HotpotQA and TriviaQA. The overview of range of hyperparameters' values can be

[8]See `https://github.com/castorini/anserini`.

[9]See `https://lucene.apache.org/`.

found in Appendix A. Best-performing hyperparameters are adopted in our retrieval test and they can be found in Table 3.6.

Model	Hyperparameter	SQuAD$_{open}$ (tuned)	HotpotQA$_{fullwiki}$ (tuned)	TriviaQA$_{wiki}$ (tuned)
BM25	k_1	0.1	0.3	0.4
	b	0.1	0.4	0.3
QL	λ	0.1	0.1	0.1

Table 3.6: Fine-tuned hyperparameters are derived from hyperparameter tuning on each training set of SQuAD, HotpotQA and TriviaQA in the open-domain setting for optimizing the metric TOP-10.

3.2.2 Retrieval Results

Many previous works [22, 72, 85, 106] have demonstrated that BM25 and QL have similar retrieval performance in different tasks of information retrieval and their performances are both better than that of TF-IDF. Therefore, we expect that our retrieval results also can showcase similarities to previous works, which can be considered as a sanity check of our implementation of traditional IR models used in our retrieval test.

We perform our retrieval test on datasets SQuAD, HotpotQA and TriviaQA in the open-domain setting, in which we use traditional IR models TF-IDF, BM25 and QL to retrieve top-ranked 5 and 10 documents from Wikipedia to questions of development set of each dataset. We then compute the metric **TOP-5** and **TOP-10** to indicate the retrieval performance of three traditional IR models across datasets. The retrieval results of our retrieval test can be found in Table 3.7.

Model	SQuAD$_{open}$		HotpotQA$_{fullwiki}$		TriviaQA$_{wiki}$	
	TOP-5	TOP-10	TOP-5	TOP-10	TOP-5	TOP-10
TF-IDF	10.45	17.21	21.48	27.75	33.55	44.53
BM25	**68.06**	**74.33**	**60.92**	**67.68**	**89.64**	**92.82**
QL	56.86	66.04	58.76	65.51	85.70	90.48

Table 3.7: Retrieval results of our retrieval test on development set of SQuAD, HotpotQA and TriviaQA in the open-domain setting. TOP-5(10) indicates % of questions whose top-ranked 5(10) documents retrieved by the traditional IR model from Wikipedia contain the answer. The best retrieval performance of TOP-5(10) for each dataset is in **bold**.

As we can see from Table 3.7, traditional IR models BM25 and QL have similar retrieval performance across datasets and they are both have a better retrieval performance than TF-IDF, which is similar to the results of these traditional IR models in previous works.

3.3 Analysis of Retrieval Test

After the sanity check of the implementation of TF-IDF, BM25 and QL in our retrieval test, we will give more detailed analysis of retrieval results in Table 3.7 in this section.

As we can see from Table 3.7, BM25 achieves the best retrieval performance across three different datasets in the open-domain setting based on metrics TOP-5 and TOP-10. The results of BM25 are in **bold** in Table 3.7. In contrast, TF-IDF is the worst retrieval model in our retrieval test across datasets in the open-domain setting. In terms of each traditional IR model, no matter what traditional IR model is, it achieves relatively worse retrieval performance on SQuAD and HotpotQA in the open-domain setting compared to the performance achieved on TriviaQA in the open-domain setting. To figure out the performance gap across datasets SQuAD, HotpotQA and TriviaQA in the open-domain setting, we perform an error analysis of exploring error types of questions of these datasets.

3.3.1 Error Analysis

We randomly sample 100 questions from the development set of SQuAD, HotpotQA and TriviaQA respectively and these questions are ones whose top-ranked 10 documents retrieved by BM25 from Wikipedia do not contain the answer to them. The reason we select top-ranked documents retrieved by BM25 is due to the reason that BM25 achieves the best retrieval performance on all datasets in our retrieval test, so questions to which BM25 cannot retrieve top-ranked 10 documents containing the answer are relatively more difficult for traditional IR models TF-IDF and QL to some extent. We manually analyze these sampled 100 questions of each dataset based on questions and their top-ranked 10 retrieved documents for figuring out reasons (i.e., error types) why BM25 cannot retrieve top-ranked 10 documents that contain answers to these questions. The overview of the error analysis can be found in Table 3.8, where we denote error types of SQuAD, HotpotQA and TriviaQA using the Arabic number with the character "A", "B" and "C" respectively.

The detailed analysis of figuring out the error types of questions is described below with a representative example of each error type.

Error Analysis of SQuAD in the open-domain setting

A1
- **Question**: "*which French kind issued this declaration?*"
- **Analysis**: This question contains two entities "*French kind*" and "*this declaration*". The former one has a spelling error and it should be "*French king*". The latter one becomes an ambiguous entity in the open-domain setting, as BM25 does not know what exactly "*this declaration*" refers to. BM25 might retrieve Wikipedia articles of "*Anglo-French Declaration*" and "*Balfour Declaration*" which contain "*French*" and "*declaration*" but not the answer "*Louis XIV*".
- **Error type**: The question contains an ambiguous entity and an entity with a spelling error.

Dataset	Error type	# ques
SQuAD$_{open}$	A1: Spelling error (ambiguous)	2
	A2*: No answer in ground truth article (ambiguous)	1
	A3: No topical entity (ambiguous)	**52**
	A4: Incomplete name of entity (ambiguous)	5
	A5: Acronym of entity (ambiguous)	5
	A6: Long name of entity (unambiguous)	9
	A7: Lexical variation (unambiguous)	4
	A8: Low term frequency (unambiguous)	13
	A9*: No answer in ground truth article (unambiguous)	7
	A10: Spelling error (unambiguous)	2
Total		100
HotpotQA$_{fullwiki}$	**B1: No name of bridge entity (bridge)**	**57**
	B2: Low term frequency (bridge)	19
	B3: Stopwords in name of entity (bridge)	3
	B4*: No answer in ground truth article (bridge)	2
	B5: No topical entity (bridge, ambiguous)	8
	B6: Acronym of entity (bridge, ambiguous)	2
	B7: Long name of entity (bridge)	2
	B8: Stopwords in name of entity (unambiguous)	3
	B9*: Answer partially correct (unambiguous)	4
Total		100
TriviaQA$_{wiki}$	C1*: Answer partially correct (unambiguous)	41
	C2: Spelling error (unambiguous)	1
	C3: Low term frequency (unambiguous)	**23**
	C4: Stopwords in name of entity (unambiguous)	10
	C5*: No Wikipedia article of entity (unambiguous)	22
	C6: No topical entity (ambiguous)	3
Total		100

Table 3.8: The overview of the error analysis. We randomly sample 100 questions from each dataset respectively and these questions are ones whose top-ranked 10 documents retrieved by BM25 from Wikipedia do not contain the answer. Error types with * indicate that the reason for these questions whose top-ranked 10 documents do not contain the answer is not due to the questions or the retrieval model BM25. The error type of each dataset in the open-domain setting which has the largest number of questions is in **bold** (except for error types with *).

A2
- **Question**: "*when do the stated Treaties apply?*"
- **Analysis**: This question contains an entity "*Treaties*" and this entity becomes an ambiguous entity in the open-domain setting, as BM25 does not know what exactly "*Treaties*" refers to. BM25 might retrieve many Wikipedia articles contain "*Treaties*" but without the answer. However, in this scenario, BM25 has retrieved the ground truth article of this question

successfully which is "*European Union law*", but this Wikipedia article does not contain the answer to this question. This is because the version of Wikipedia in our retrieval test is different from that used for the production of dataset SQuAD, so some Wikipedia articles have a few changes.

– **Error type**: The question contains an ambiguous entity, but BM25 has retrieved the ground truth article. However, this article does not have the ground truth answer.

A3
– **Question**: "*what day was the game played on?*"
– **Analysis**: This question contains an entity "*the game*" and this entity becomes an ambiguous entity in the open-domain setting, as BM25 does not know what exactly "*the game*" refers to. BM25 might retrieve Wikipedia articles of "*Video games*" and "*Olympic Games*" that contain "*game*" many times, but without the answer. In this scenario, the ground truth article of this question is not retrieved successfully in top-ranked 10 documents retrieved by BM25 from Wikipedia.
– **Error type**: The question contains an ambiguous entity since it does not have the topical entity to specify what the ambiguous entity refers to.

A4
– **Question**: "*the Yuan was the first time all of China was ruled by whom?*"
– **Analysis**: This question contains an entity "*the Yuan*" and this entity becomes an ambiguous entity in the open-domain setting. The name of this ambiguous entity incomplete and it is supposed to be "*the Yuan Dynasty*". In this scenario, BM25 has retrieved many Wikipedia articles containing "*China*" and "*Yuan*" such as the article of "*Law of the Republic of China*", but none of them contains the answer.
– **Error type**: The question contains an ambiguous entity with an incomplete name.

A5
– **Question**: "*what are the only states where ABC doesn't have a licensed affiliate?*"
– **Analysis**: This question contains an entity "*ABC*" and this entity becomes an ambiguous entity in the open-domain setting, since it is an acronym of the entity. The specific entity "*ABC*" refers to is "*American Broadcasting Company*", but there are some other entities with the same acronym in Wikipedia. For example, BM25 might retrieve Wikipedia articles of "*Australian Broadcasting Corporation*" and "*Another Bad Creation*", but they do not contain the answer.
– **Error type**: The question contains an ambiguous entity and the name of this entity is an acronym.

A6
– **Question**: "*where did Super Bowl 50 take place?*"
– **Analysis**: This question contains an entity "*Super Bowl 50*" and this entity is an unambiguous entity[10] in the open-domain setting. "*Super Bowl 50*"

[10]The unambiguous entity refers to the entity whose name is not a pronoun, an acronym or incomplete.

has its own Wikipedia article which is the ground truth article of this question, but BM25 fails to retrieve it. This is due to the long name of "*Super Bowl 50*" and this name is considered at word-level[11] independently instead of as a connective phrase in the process of retrieval. Therefore, BM25 has retrieved Wikipedia articles of "*Peyton Manning*" and "*Super Bowl XXXVIII halftime-show controversy*" in which words "*Super*", "*Bowl*" and "*50*" appear independently.

– **Error type**: The question contains an unambiguous entity, but the name of this entity is long.

A7 – **Question**: "*what is the enrollment of undergraduates at Harvard?*"

– **Analysis**: This question contains an entity "*Harvard*" and this entity is an unambiguous entity in the open-domain setting. "*Harvard*" has its own Wikipedia article which is the ground truth article of this question, but BM25 fails to retrieve it. We check this ground truth article and find that there is a lexical variation[12] between the text of question and the ground truth article. Specifically, the text of the ground truth article is "*there are 16,000 staff and faculty, including 2,400 professors, lecturers, and instructors teaching 7,200 undergraduates and 14,000 graduate students*" which contains the answer "*7,200*". However, the text of the question "enrollment of undergraduates" is different from the text "*instructors teaching 7,200 undergraduates*" of the article, but they indicate the same meaning.

– **Error type**: The question contains an unambiguous entity and there is a lexical variation between the text of the question and the ground truth article.

A8 – **Question**: "*what is the Dutch word for the Amazon rainforest?*"

– **Analysis**: This question contains an entity "*Amazon rainforest*" and this entity is an unambiguous entity in the open-domain setting. "*Amazon rainforest*" has its own Wikipedia article which is the ground truth article of this question, but BM25 fails to retrieve it. We check this ground truth article and find that the word in the question "*Dutch*" only appears in the ground truth article once. Instead, BM25 has retrieved Wikipedia articles of "*Belem*" and "*White Brazilians*" which contain many times of word "*Dutch*" but without the answer.

– **Error type**: The question contains an unambiguous entity but with some words that have low term frequency in the ground truth article.

A9 – **Question**: "*what is European Union Law?*"

– **Analysis**: This question contains an entity "*European Union Law*" and this entity is an unambiguous entity in the open-domain setting. "*European Union Law*" has its own Wikipedia article which is the ground truth article of this question and BM25 has retrieved the ground truth article of this

[11]As we described in Section 2.2, the indexed unit of text is usually a word.

[12]Two pieces of text indicate the same information need but with different words (e.g., synonym).

question successfully. However, this ground truth article does not contain the answer to this question. This is because the version of Wikipedia in our retrieval test is different from that used for the production of dataset SQuAD, so some Wikipedia articles have a few changes.

- **Error type**: The question contains an unambiguous entity and BM25 has retrieved the ground truth article. However, this article does not have the answer.

A10
- **Question**: "*What was the percentage of whit people in Fresno in 2010?*"
- **Analysis**: This question contains an entity "*Fresno*" and this entity is an unambiguous entity in the open-domain setting. "*Fresno*" has its own Wikipedia article which is the ground truth article of this question, but BM25 fails to retrieve this article. The word "*whit*" in the question should be "*white*".
- **Error type**: The question contains an unambiguous entity but with a spelling error of words.

Error Analysis of HotpotQA in the open-domain setting

B1
- **Question**: "*the birthplace of George McCall Theal is a port city of what bay?*"
- **Analysis**: This question contains an unambiguous entity "*George McCall Theal*" which has its own Wikipedia article and a bridge entity "*a port city*". Based on top-ranked 10 documents retrieved by BM25, one of the ground truth articles[13] of this question "*George McCall Theal*" has been retrieved by BM25 in top-ranked 10 documents. However, BM25 fails to retrieve the other ground truth article "*Saint John, New Brunswick*" which contains the answer to this question and "*Saint John, New Brunswick*" is also the name of the bridge entity "*a port city*". This is due to the reason that the real name of the bridge entity "*Saint John, New Brunswick*" does not appear in the question and it is replaced by a descriptive phrase "*a port city*". BM25 thus cannot retrieve the Wikipedia article of "*Saint John, New Brunswick*" based on the bridge entity "*a port city*" in the question.
- **Error type**: The question contains an unambiguous entity and a bridge entity. BM25 has retrieved the ground truth article of the unambiguous entity but fails to retrieve the ground truth article of the bridge entity, as the real name of the bridge entity does not appear in the question.

B2
- **Question**: "*when did the baseball draft with which Alex Lange was the 30th pick began?*"

[13]As we described in Section 3.1, each question of HotpotQA was generated over two paragraphs from two different Wikipedia articles. Hence, there are two ground truth Wikipedia articles for each question of HotpotQA.

- **Analysis**: This question contains an unambiguous entity "*Alex Lange*" which has its own Wikipedia article and a bridge entity "*the baseball draft*". Based on top-ranked 10 documents retrieved by BM25, none of the ground truth articles have been retrieved in top-ranked 10 documents. The unambiguous entity "*Alex Lange*" only appears in the ground truth article few times and the bridge entity "*the baseball draft*" does not have the real name in the question.
- **Error type**: The question contains an unambiguous entity and a bridge entity. BM25 does not retrieve any of the ground truth articles in top-ranked 10 documents, as the name of the unambiguous entity has low term frequency in the ground truth article and the real name of the bridge entity does not appear in the question.

B3
- **Question**: "*show 'Em (What You're Made Of) is a song written by a group of people including an American singer/songwriter who was inducted into what on April 10, 2015?*"
- **Analysis**: This question contains an unambiguous entity "*What You're Made Of*" which has its own Wikipedia article and a bridge entity "*a group of people*". Based on top-ranked 10 documents retrieved by BM25, none of the ground truth articles have been retrieved in top-ranked 10 documents. The unambiguous entity "*What You're Made Of*" contains many stopwords (e.g., "*what*", "*you*" and "*of*") and the bridge entity "*a group of people*" does not have the real name in the question.
- **Error type**: The question contains an unambiguous entity and a bridge entity. BM25 does not retrieve any of the ground truth articles in top-ranked 10 documents, as the name of the unambiguous entity contains many stopwords and the real name of the bridge entity does not appear in the question.

B4
- **Question**: "*how far from Sacramento is the flight school in Atwater?*"
- **Analysis**: This question contains a bridge entity "*the flight school*". However, in this scenario, BM25 has retrieved two ground truth articles of this question in top-ranked 10 documents, but they do not have the answer in the text. This is because the version of Wikipedia in our retrieval test is different from that used for the production of dataset HotpotQA, so some Wikipedia articles have a few changes.
- **Error type**: The question contains a bridge entity and BM25 has retrieved all ground truth article. However, these articles do not have the answer.

B5
- **Question**: "*the most popular temple in terms of attractions is located in what part of the city?*"
- **Analysis**: This question contains a bridge entity "*the city*". Based on top-ranked 10 documents retrieved by BM25, none of the ground truth articles have been retrieved in top-ranked 10 documents. The bridge entity "*the city*" does not have the real name in the question and the description "*the*

most popular temple in terms of attractions" is not specific and lacks topical entity in the question which is ambiguous in the open-domain setting for BM25.

- **Error type**: The question contains a bridge entity and an ambiguous description. BM25 does not retrieve any of the ground truth articles in top-ranked 10 documents, as the real name of the bridge entity does not appear in the question and the description lacks topical entity.

B6
- **Question**: "*M.F.A. starred the actress known for starring in what E! reality series?*"
- **Analysis**: This question contains an ambiguous entity "*M.F.A.*" and a bridge entity "*the actress*". Based on top-ranked 10 documents retrieved by BM25, none of the ground truth articles have been retrieved in top-ranked 10 documents. The bridge entity "*the actress*" does not have the real name in the question and the name of the entity "*M.F.A.*" is an acronym, which is ambiguous in the open-domain setting for BM25.
- **Error type**: The question contains an ambiguous entity and a bridge entity. BM25 does not retrieve any of the ground truth articles in top-ranked 10 documents, as the real name of the bridge entity does not appear in the question and the name of the other entity in the question is an acronym which is ambiguous in the open-domain setting.

B7
- **Question**: "*what crossroads is the town located 10 km north of the Charcoal Tank Nature Reserve located?*"
- **Analysis**: This question contains an unambiguous entity "*Charcoal Tank Nature Reserve*" and a bridge entity "*the town*". Based on top-ranked 10 documents retrieved by BM25, none of the ground truth articles have been retrieved in top-ranked 10 documents. The unambiguous entity has a long name which is not considered as a connective phrase in the process of retrieval. The bridge entity "*the town*" does not have the real name in the question.
- **Error type**: The question contains an unambiguous entity and a bridge entity. BM25 does not retrieve any of the ground truth articles in top-ranked 10 documents, as the real name of the bridge entity does not appear in the question and the name of the unambiguous entity is long.

B8
- **Question**: "*who was born first Am Rong or Ava DuVernay?*"
- **Analysis**: This question contains two unambiguous entities "*Am Rong*" and "*Ava DuVernay*" in comparison, which both have their own Wikipedia articles. The Wikipedia articles of these two unambiguous entities are ground truth articles of this question. Based on top-ranked 10 documents retrieved by BM25, BM25 only has retrieved the article of "*Ava DuVernay*" without the article of "*Am Rong*", as the word "*Am*" is a stopword.
- **Error type**: The question contains two unambiguous entities in comparison. BM25 only retrieves one of the ground truth articles of one entity, as the other entity whose name contains a stopword.

B9
- **Question**: "*who died last Vladimir Arnold or George Cantor?*"
- **Analysis**: This question contains two unambiguous entities "*Vladimir Arnold*" and "*George Cantor*" in comparison, which both have their own Wikipedia articles. The Wikipedia articles of these two unambiguous entities are ground truth articles of this question. Based on top-ranked 10 documents retrieved by BM25, BM25 has retrieved ground truth articles for both entities, but the answer in one of the ground truth articles is partially correct compared to the ground truth answer ("*Vladimir Arnold*" vs "*Vladimir Igorevich Arnold*").
- **Error type**: The question contains two unambiguous entities in comparison. BM25 has retrieved ground truth articles for both entities, but the answer is partially correct.

Error Analysis of TriviaQA in the open-domain setting

C1
- **Question**: "*Who was the male star of the movie The Man of La Mancha?*"
- **Analysis**: This question contains an unambiguous entity "*The Man of La Mancha*" which has its own Wikipedia article. Based on top-ranked 10 documents retrieved by BM25, the article of this entity is retrieved successfully, but the answer appearing in this article is partially correct (i.e., not the same) (*"peter o'toole*" vs "*peter otoole*").
- **Error type**: The question contains an unambiguous entity. BM25 has retrieved the Wikipedia article of this unambiguous entity which contains the answer, but the answer is not the same as the ground truth answer.

C2
- **Question**: "*Long An Provence is in which Asian country?*"
- **Analysis**: This question contains an unambiguous entity "*Long An Provence*" which has its own Wikipedia article. Based on top-ranked 10 documents retrieved by BM25, the article of this entity is not retrieved. The word "*Provence*" has a spelling error and it should be "*Province*".
- **Error type**: The question contains an unambiguous entity with a spelling error.

C3
- **Question**: "*What was Warren Beatty's first movie?*"
- **Analysis**: This question contains an unambiguous entity "*Warren Beatty*" which has its own Wikipedia article. Based on top-ranked 10 documents retrieved by BM25, the article of this entity is not retrieved, as the name of this entity appears in this article a few times. Therefore, BM25 has retrieved Wikipedia articles that contain these two words more times such as Wikipedia article of "*Dick Tracy (1990 film)*".
- **Error type**: The question contains an unambiguous entity but with some words that have low term frequency in the article.

C4
- **Question**: "*Which musical featured the song The Street Where You Live?*"

- **Analysis**: This question contains an unambiguous entity "*The Street Where You Live*" which has its own Wikipedia article. Based on top-ranked 10 documents retrieved by BM25, the article of this entity is not retrieved, as the name of this entity contains some stopwords such as "*The*", "*Where*" and "*You*".
- **Error type**: The question contains an unambiguous entity but the name of this entity contains some stopwords.

C5
- **Question**: "*Mr Worldly Wisemen appears in which 17th Century book?*"
- **Analysis**: This question contains an unambiguous entity "*Mr Worldly Wisemen*", but Wikipedia has no article of this entity. This is because questions of TriviaQA were not generated based on Wikipedia articles as we described in Section 3.1. Therefore, some entities in the questions of TriviaQA do not have their corresponding Wikipedia articles.
- **Error type**: The question contains an unambiguous entity but this entity does not have the corresponding Wikipedia article.

C6
- **Question**: "*In which year did St George die?*"
- **Analysis**: This question contains an ambiguous entity "*St George*". in the open-domain setting, Wikipedia has many articles of this entity, so BM25 cannot retrieve accurate articles containing the answer to this question. In addition, the topical entity is not provided in the question to help specify what the entity "*St George*" is related to.
- **Error type**: The question contains an ambiguous entity, as it does not have the topical entity to specify what the ambiguous entity is related to.

3.3.2 Conclusion

As we can see from the overview of error analysis (Table 3.8), SQuAD, HotpotQA and TriviaQA in the open-domain setting have different main error types of questions respectively. We give an overall description of error types of questions of datasets following.

- The majority of error types of questions which were sampled from dataset SQuAD are based on questions with *ambiguous* entities and the number of these questions is 64[14] out of all 100 sampled questions. The error type *A3: No topical entity (ambiguous)* is the most prominent one, which has 52 questions. Therefore, we can conclude that **questions with ambiguous entities of SQuAD are the main questions that have a negative impact on the retrieval performance of traditional IR models TF-IDF, BM25 and QL**. The ambiguity of entities in the open-domain setting existing in SQuAD is also referred to in some previous works [17, 136]. Furthermore, we randomly sample 100 questions from the development set of SQuAD to manually analyze the number of questions with a certain type of entity out of 100 sampled questions in the open-domain setting, which can be found in Table 3.9.

[14]This number does not include the error type *A2: No answer in ground truth article (ambiguous)*.

- The majority of error types of questions which were sampled from dataset HotpotQA are based on questions with *bridge* entities and the number of these questions is 91[15] out of all 100 sampled questions. The error type *B1: No name of bridge entity (bridge)* is the most prominent one, which has 57 questions. Therefore, we can conclude that **questions with bridge entities of HotpotQA are the main questions that have a negative impact on the retrieval performance of traditional IR models TF-IDF, BM25 and QL**. Furthermore, we randomly sample 100 questions from the development set of HotpotQA to manually analyze the number of questions with a certain type of entity out of 100 sampled questions in the open-domain setting, which can be found in Table 3.9.

- The majority of error types of questions which were sampled from dataset TriviaQA are based on questions with *unambiguous* entities and the number of these questions is 34[16] out of all 100 sampled questions. The error type *C3: Low term frequency (unambiguous)* is the most prominent one, which has 23 questions. Therefore, we can conclude that **questions with unambiguous entities of TriviaQA are the main questions that have a negative impact on the retrieval performance of traditional IR models TF-IDF, BM25 and QL**. Furthermore, we randomly sample 100 questions from the development set of TriviaQA to manually analyze the number of questions with a certain type of entity out of 100 sampled questions in the open-domain setting, which can be found in Table 3.9.

Dataset	# ques (ambiguous)	# ques (unambiguous)	# ques (bridge)
SQuAD$_{open}$	**42**	58	0
HotpotQA$^*_{fullwiki}$	4	15	**85**
TriviaQA$_{wiki}$	4	**96**	0

Table 3.9: The number of questions with ambiguous, unambiguous and bridge entities out of 100 questions sampled from SQuAD, HotpotQA and TriviaQA respectively. The number of questions is analyzed in the open-domain setting. Note that HotpotQA$^*_{fullwiki}$ has some questions with ambiguous and bridge entities simultaneously, so these questions not only belong to questions with ambiguous entities but also questions with bridge entities. The largest number of questions with ambiguous, unambiguous and bridge entities among all datasets is in **bold**.

In conclusion, the result and analysis of the retrieval test above can give us some insights that can help us answer **RQ1** and **RQ2** as we described in Section 1.2.

RQ1: In the task of open-domain QA, different QA datasets have different impacts on the retrieval performance of traditional IR models that are used in the retrieval stage of open-domain QA. This is due to the reason that different QA datasets contain questions with different types of entities (i.e., ambiguous, unambiguous

[15]This number does not include the error type *B4: No answer in ground truth article (bridge)*.

[16]This number does not include the error type *C1: Answer partially correct (unambiguous) and C5: No Wikipedia article of entity (unambiguous)*.

and bridge entities) and they differ in difficulty of being dealt with by traditional IR models.

RQ2: As we can see from Table 3.9, compared to TriviaQA, SQuAD and HotpotQA contain a much larger number of questions with ambiguous entities and bridge entities respectively (42% and 85%). In contrast, TriviaQA contains a much larger number of questions with unambiguous entities (96%). Therefore, the performance gap shown in Table 3.7 is due to the reason that SQuAD and HotpotQA contain a large number of questions with ambiguous entities and bridge entities respectively and these questions are much harder to be handled by traditional IR models than questions with unambiguous entities. Error types based on questions with ambiguous entities and bridge entities are prominent error types of questions of SQuAD and HotpotQA respectively that have a negative impact on traditional IR models.

3.4 Hypothesis

In this section, we propose three hypotheses that we think might alleviate the negative impact on the retrieval performance of traditional IR models resulting from error types of questions of SQuAD and HotpotQA. There are some reasons for focusing on error types of questions of SQuAD and HotpotQA, which are described following.

- In the open-domain setting, the retrieval performance of traditional IR models on SQuAD and HotpotQA are much worse than that on TriviaQA (Table 3.7). Hence, it is more significant to improve the retrieval performance of traditional IR models on datasets SQuAD and HotpotQA, as these two QA datasets in the open-domain setting are much harder than the dataset TriviaQA for traditional IR models.

- In the open-domain setting, the error types of questions of TriviaQA are included in the error types of questions of SQuAD and HotpotQA (Table 3.8). Hence, if the retrieval performance[17] of traditional IR models is improved on SQuAD and HotpotQA under our hypotheses, it indicates that error types of questions of TriviaQA can be handled by our hypotheses to some extent.

3.4.1 Hypothesis I (H1)

In the retrieval test as we described in Section 3.2, we employ traditional IR models TF-IDF, BM25 and QL to retrieve documents from Wikipedia to questions of different datasets. The knowledge source Wikipedia used in the retrieval test is indexed based on each Wikipedia article as the retrieval document unit.

However, we think that Wikipedia with smaller retrieval document units can be beneficial to the retrieval performance of traditional IR models on dataset SQuAD and HotpotQA. The reasons for this are analyzed as follows:

[17]Especially the retrieval performance of traditional IR models on questions with unambiguous entities in SQuAD and HotpotQA is increased.

- As we described in the selection of datasets (Section 3.1), questions of SQuAD were generated based on a single paragraph of a Wikipedia article and questions of HotpotQA were generated based on two paragraphs of two different Wikipedia articles. Hence, traditional IR models retrieving Wikipedia articles that are relevant to questions that were generated based on paragraph-level is harder than IR models retrieving smaller pieces of text (e.g., paragraphs or sentences of Wikipedia articles) that are relevant to questions. A Wikipedia article generally describes a topic about an entity, but there are various sub-topics about this entity in this article. For example, the Wikipedia article of the entity "*Super Bowl 50*" contains sub-topics about the background, broadcasting, entertainment, game summary, etc. Therefore, smaller retrieval document units can be more topic specific than each Wikipedia article as the retrieval document unit, which we think can help improve the retrieval performance of traditional IR models on error types of questions of SQuAD and HotpotQA.
- One of the error types of questions of SQuAD and HotpotQA is related to the query term which has the low term frequency in the ground truth article. Hence, smaller retrieval document units can increase the term frequency since the total number of words of a paragraph or a sentence is much smaller than that of a Wikipedia article (e.g., the Wikipedia article "*Super Bowl 50*" has 8,008 words).

In conclusion, our first hypothesis is that **if the retrieval document unit of traditional IR models for Wikipedia is smaller than the retrieval document unit that is based on each Wikipedia article, the retrieval performance of traditional IR models on SQuAD and HotpotQA in the open-domain setting can be improved**.

3.4.2 Hypothesis II (H2)

As we can see from the overview of error analysis (Table 3.8), the error type of questions *A3: No topical entity (ambiguous)* and the one *B1: No name of bridge entity (bridge)* are the error type of questions with the largest number of questions of SQuAD and HotpotQA respectively. If the topical entity and the name of the bridge entity are expanded into the original question, traditional IR models are likely to retrieve documents that contain the answer based on the revised question. The reasons for this are analyzed as follows:

- In terms of the example of the error type *A3* of SQuAD, the question "*what day was the game played on?*" contains the ambiguous entity "*the game*" in the open-domain setting. If some relevant terms, that are related to the topical entity "*Super Bowl 50*" and this question, are provided in this question, these relevant terms then can help specify what the ambiguous entity "*the game*" is related to. As a result, traditional IR models are more likely to retrieve the ground truth article of "*Super Bowl 50*" from Wikipedia that contains the answer.
- In terms of the example of the error type *B1* of HotpotQA, the question "*the birthplace of George McCall Theal is a port city of what bay?*" contains the bridge entity "*a port city*" which is a descriptive phrase. The real name of this bridge entity is not provided in this question, which is "*Saint John, New-Brunswick*". If some relevant terms, that are related to the name of this bridge

entity and this question, are provided in this question, traditional IR models are more likely to retrieve one of the ground truth articles[18] of "*Saint John, New-Brunswick*" from Wikipedia that contains the answer.

In conclusion, our second hypothesis is that **if some relevant terms, that are related to questions with ambiguous entities and questions with bridge entities, are provided into the original question, the retrieval performance of traditional IR models on SQuAD and HotpotQA in the open-domain setting can be improved by retrieving documents based on the revised question**.

3.4.3 Hypothesis III (H3)

As we described in Section 1.1, some previous [42, 63, 84, 125] works have demonstrated that adding the neural ranker after traditional IR models in open-domain QA has shown some retrieval improvements over the traditional IR models without the neural ranker, but they did not conduct fine-grained error analyses of traditional IR models like we did in Section 3.3. Therefore, it is still unknown whether the neural ranker can be valid to error types of questions of SQuAD and HotpotQA. We thus attempt to add a neural ranker after traditional IR models to see whether the neural ranker can handle error types of questions of SQuAD and HotpotQA, thereby improving the retrieval performance of traditional IR models on these two datasets.

In conclusion, our final hypothesis is that **adding a neural ranker after traditional IR models can improve the retrieval performance of traditional IR models on SQuAD and HotpotQA in the open-domain setting**.

3.5 Methodology

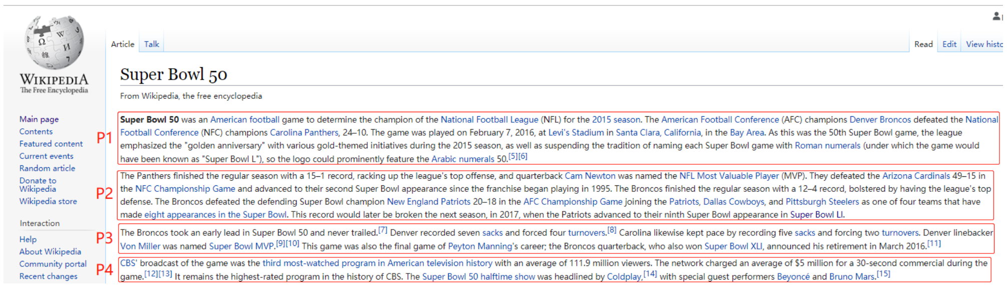
This section aims to describe methodologies that we adopt for implementing **H1**, **H2** and **H3** that we proposed in Section 3.4.

3.5.1 Paragraph and Sentence Based Retrieval Document Unit (H1)

We attempt to index the knowledge source Wikipedia by considering each paragraph and each sentence of Wikipedia articles as the retrieval document unit of traditional IR models for implementing **H1**. In the retrieval test (Section 3.2), we used Anserini to index Wikipedia by considering each Wikipedia article as the retrieval document unit of traditional IR models. To make the retrieval document unit smaller, we use Anserini to index Wikipedia in two ways: (1) by considering each paragraph of Wikipedia articles as the retrieval document unit; (2) by considering each sentence of Wikipedia articles as the retrieval document unit. We define the paragraph and the sentence as follows:

- **Paragraph**: The piece of text ends up with one or more line breaks in Wikipedia articles. Some paragraphs of the Wikipedia article "*Super Bowl 50*" can be seen in Figure 3.4.

[18]The other one is the Wikipedia article of the unambiguous entity "*George McCall Theal*" and the real name of the bridge entity "*a port city*" appears in this Wikipedia article.



Super Bowl 50

From Wikipedia, the free encyclopedia

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champions Denver Broncos defeated the National Football Conference (NFC) champions Carolina Panthers, 24–10. The game was played on February 7, 2016, at Levi's Stadium in Santa Clara, California, in the Bay Area. As this was the 50th Super Bowl game, the league emphasized the "golden anniversary" with various gold-themed initiatives during the 2015 season, as well as suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so the logo could prominently feature the Arabic numerals 50.[5][6]

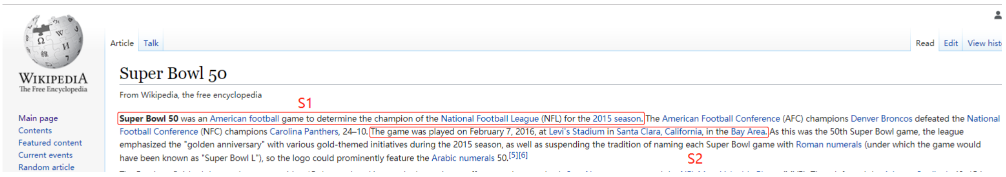
The Panthers finished the regular season with a 15–1 record, racking up the league's top offense, and quarterback Cam Newton was named the NFL Most Valuable Player (MVP). They defeated the Arizona Cardinals 49–15 in the NFC Championship Game and advanced to their second Super Bowl appearance since the franchise began playing in 1995. The Broncos finished the regular season with a 12–4 record, bolstered by having the league's top defense. The Broncos defeated the defending Super Bowl champion New England Patriots 20–18 in the AFC Championship Game joining the Patriots, Dallas Cowboys, and Pittsburgh Steelers as one of four teams that have made eight appearances in the Super Bowl. This record would later be broken the next season, in 2017, when the Patriots advanced to their ninth Super Bowl appearance in Super Bowl LI.

The Broncos took an early lead in Super Bowl 50 and never trailed.[7] Denver recorded seven sacks and forced four turnovers.[8] Carolina likewise kept pace by recording five sacks and forcing two turnovers. Denver linebacker Von Miller was named Super Bowl MVP.[9][10] This game was also the final game of Peyton Manning's career; the Broncos quarterback, who also won Super Bowl XLI, announced his retirement in March 2016.[11]

CBS' broadcast of the game was the third most-watched program in American television history with an average of 111.9 million viewers. The network charged an average of $5 million for a 30-second commercial during the game.[12][13] It remains the highest-rated program in the history of CBS. The Super Bowl 50 halftime show was headlined by Coldplay,[14] with special guest performers Beyoncé and Bruno Mars.[15]

Figure 3.4: Some paragraphs of the Wikipedia article "*Super Bowl 50*".

- **Sentence**: The piece of text starts with capital letters (i.e., "A-Z") and ends up with punctuation ".", "!" and "?" in Wikipedia articles. Some sentences of the Wikipedia article "*Super Bowl 50*" can be seen in Figure 3.5.



Super Bowl 50

From Wikipedia, the free encyclopedia

Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season. The American Football Conference (AFC) champions Denver Broncos defeated the National Football Conference (NFC) champions Carolina Panthers, 24–10. The game was played on February 7, 2016, at Levi's Stadium in Santa Clara, California, in the Bay Area. As this was the 50th Super Bowl game, the league emphasized the "golden anniversary" with various gold-themed initiatives during the 2015 season, as well as suspending the tradition of naming each Super Bowl game with Roman numerals (under which the game would have been known as "Super Bowl L"), so the logo could prominently feature the Arabic numerals 50.[5][6]

Figure 3.5: Some sentences of the Wikipedia article "*Super Bowl 50*".

The overview of the retrieval process of the traditional IR model with the index based on each paragraph or each sentence of Wikipedia articles as the retrieval document unit is depicted in Figure 3.6.

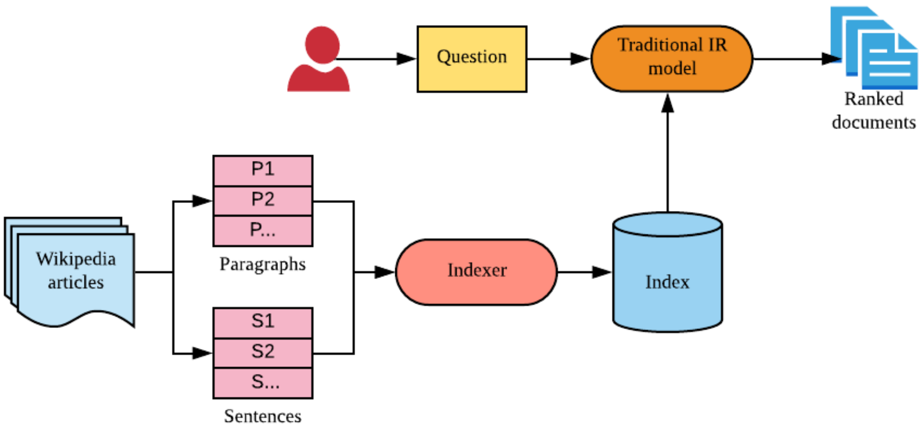


Figure 3.6: The overview of the retrieval process of the traditional IR model with the index based on each paragraph or each sentence of Wikipedia articles as the retrieval document unit.

3.5.2 Query Expansion by Pseudo Relevance Feedback (H2)

We attempt to use the relevance model RM3 based on pseudo relevance feedback (PRF) for implementing **H2**. As we described in Section 2.2.2, RM3 is an approach based on PRF to do query expansion. In terms of a question (i.e., query), the traditional IR model firstly retrieves initial round of documents from Wikipedia and top-ranked k documents are assumed to be relevant to this question. Subsequently, RM3 builds a document language model for each document in top-ranked k documents and expands a number of relevant terms into the question based on these document language models. Finally, the traditional IR model retrieves the second round of documents from Wikipedia based on the new revised question. There are three hyperparameters of RM3 that need to be tuned for the best retrieval performance: (1) *fbDocs* indicates the number of relevance feedback documents; (2) *fbTerms* indicates the number of terms that are expanded into the original question based on relevance feedback documents; (3) λ indicates the weight assigned to the original question language model, which can be seen in Equation 2.14.

The overview of the retrieval process of the traditional IR model with RM3 model is depicted in Figure 3.7.

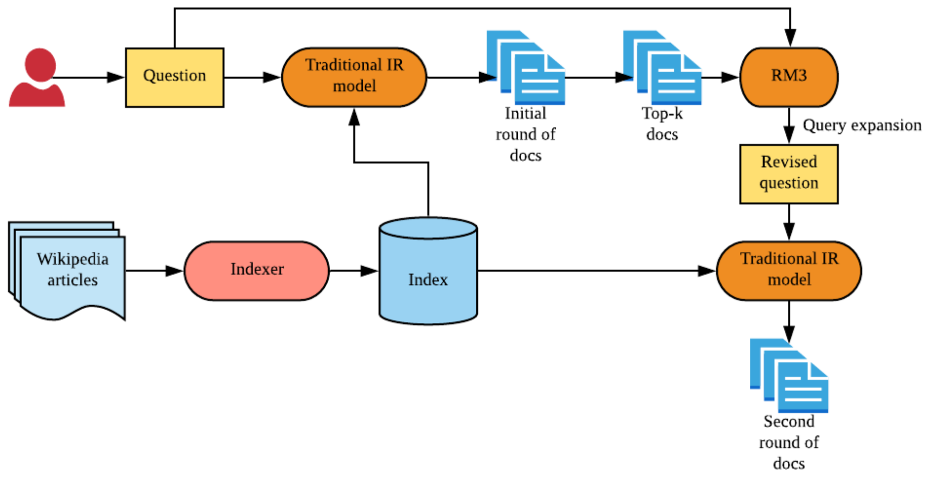


Figure 3.7: The overview of the retrieval process of the traditional IR model with RM3 model.

3.5.3 Neural Paragraph Ranker (H3)

We attempt to adopt a neural ranker named *Paragraph Ranker* proposed by Lee et al. [63] for implementing **H3**. The overview of the retrieval process of traditional IR models with Paragraph Ranker can be found in Figure 3.8. As we can see from the pipeline in Figure 3.8, the goal of Paragraph Ranker is to re-rank all paragraphs of top-ranked N documents retrieved by the traditional IR model and then select top-ranked M paragraphs as the input to the RC model[19].

[19]The RC model is not depicted in the pipeline.

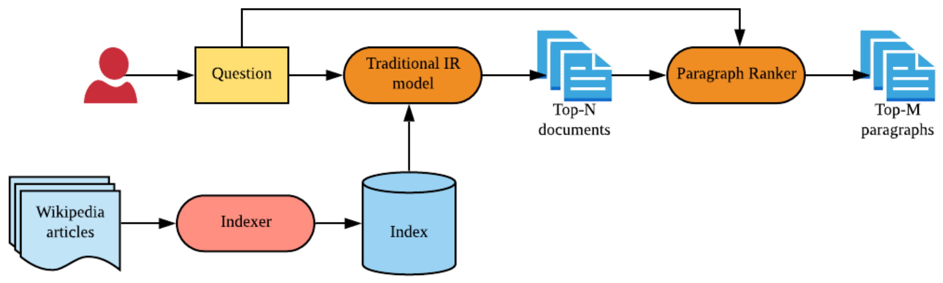


Figure 3.8: The overview of the retrieval process of the traditional IR model with Paragraph Ranker.

Paragraph Ranker is a representation-focused neural IR model [87] as we described in Section 2.2.3 and it consists of a question encoder and a paragraph encoder. The question encoder and paragraph encoder are both based on Bidirectional Long-Short Term Memory networks (Bi-LSTMs) [46], which is a variant of recurrent neural networks (RNNs) [41]. The overview of the question encoder and paragraph encoder is depicted in Figure 3.9.

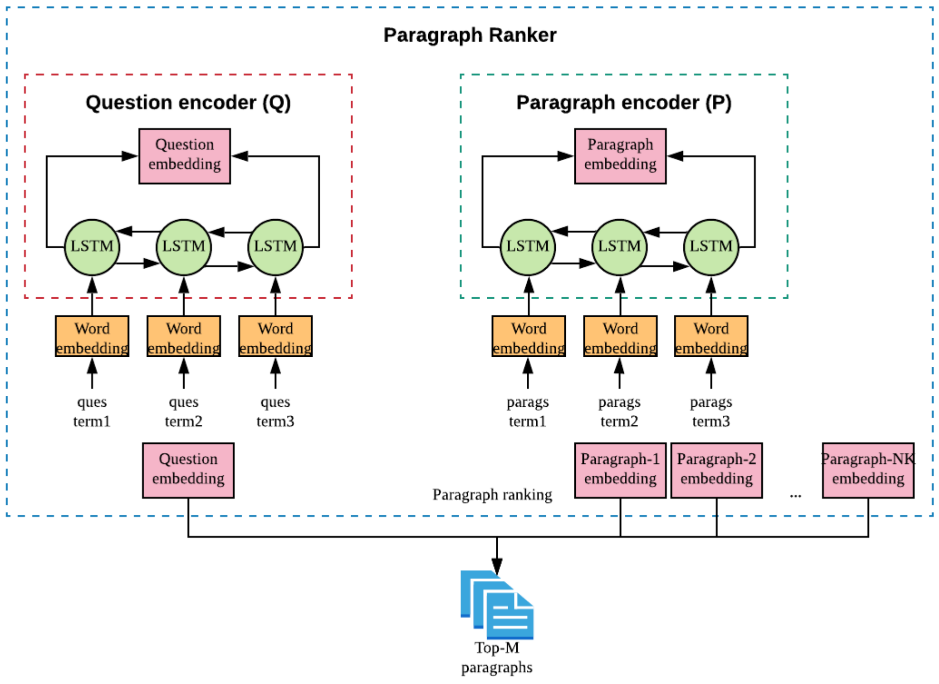


Figure 3.9: The overview of the question encoder and paragraph encoder. The input of the question encoder is word embeddings of question terms and the input of the paragraph encoder is word embeddings of paragraphs terms.

In terms of a question q, the traditional IR model retrieves top-ranked N documents and each document contains T paragraphs on average. The question q and each paragraph p_i where i ranges from 1 to NT are encoded into a question embedding and NT paragraph embeddings by the question encoder (Q) and paragraph encoder (P) respectively. The computation of encoding can be seen in Equation 3.2, where $\hat{q}$ is the question embedding, $\hat{p}_i$ is the paragraph embedding and $E(\cdot)$ converts each question term or paragraph term into a pre-trained word embedding of Glove [91]. Finally, Paragraph Ranker can rank all NT paragraphs based on the probability of relevance of each paragraph with respect to the question q and this probability is computed in Equation 3.3, where the similarity function $s(\cdot,\cdot)$ is the dot product of $\hat{q}$ and $\hat{p}_i$.

$$\hat{q} = BiLSTM_Q(E(q)) \quad \hat{p}_i = BiLSTM_P(E(p_i)) \tag{3.2}$$

$$P(p_i|q) = \frac{1}{1+e^{-s(\hat{p}_i,\hat{q})}} \tag{3.3}$$

The question encoder and paragraph encoder are trained using a negative sampling of irrelevant paragraphs and the loss function which is a binary cross-entropy loss can be found in Equation 3.4, where $P(p|q)$ is the probability of relevance of the paragraph p with respect to the question q and y indicates the actual label of paragraph (relevant: $y = 1$, non-relevant: $y = 0$). Θ indicates parameters to be trained to minimize the loss function. Therefore, Paragraph Ranker which consists of the question encoder and paragraph encoder outputs the probability of relevance of a paragraph to a question and then Paragraph Ranker can rank paragraphs based on their probabilities. The reason for using negative sampling is that it can help accelerate the convergence of loss function and balance positive training samples and negative training samples

$$L(\Theta) = -(y \cdot log(P(p|q)) + (1-y) \cdot log(1-P(p|q)) \tag{3.4}$$

3.5.4 RC Models

In Chapter 1, we depicted the two-stage working pipeline of the open-domain QA system, which can be found in Figure 1.1. In the retrieval test (Section 3.2), we only focused on the first stage (i.e., retrieval stage) of the open-domain QA system to evaluate the retrieval performance of traditional IR models across datasets. However, the final goal of improving the retrieval component is to improve the QA performance in the second stage (i.e., answer extraction stage) and a better retrieval component can benefit to the RC model, thereby boosting the QA performance. Therefore, to validate our hypotheses regarding the retrieval component, we not only consider the improvement of retrieval performance but also take the QA performance of the open-domain QA system into consideration.

We adopt two RC models for dataset SQuAD and HotpotQA respectively. Specifically, we use the neural network based RC model of DrQA [17] named *Document Reader* for dataset SQuAD and the neural network based RC model used in the work of HotpotQA [136] for dataset HotpotQA. The reasons are described as follows:

1. In terms of the dataset SQuAD in the open-domain setting, DrQA is the baseline model of combining the IR technique and the neural RC model in the task of open-domain QA. Some previous works [63, 125, 135] all compared their works with the open-domain QA system DrQA on SQuAD in the open-domain setting.

Therefore, we attempt to use the same neural RC model of DrQA to validate our hypotheses about the retrieval component of the open-domain QA system and compare our QA performance with that of the baseline model DrQA.

2. In terms of the dataset HotpotQA in the open-domain setting, the original work of HotpotQA proposed an open-domain QA system that can be used to evaluate the performance of open-domain QA on HotpotQA. This system is also considered as a baseline model on HotpotQA [29, 75]. Hence, we use the same neural RC model of this baseline model on HotpotQA to validate our hypotheses about the retrieval component of the open-domain QA system.

Document Reader of DrQA

We have described the overview of DrQA in Section 2.1.2 and it is in Figure 2.2. We thus pay our attention on the neural RC model of DrQA and the architecture of Document Reader can be found in Figure 3.10. Given a question q containing l terms $\{q^1,...,q^l\}$ and a paragraph p containing m terms $\{p^1,...,p^m\}$, the embedding of each paragraph term can be computed in Equation 3.5, where $E(\cdot)$ converts each paragraph term into a pre-trained word embedding of Glove [91]. Question encoding is similar to paragraph encoding, but the final question embedding is the combination of each question term embedding: $\hat{q} = \sum_j b_j \hat{q^j}$ where b_j encodes the importance of each question term. The computation of b_j can be found in Equation 3.6, where w is a weight vector to learn. In terms of the prediction, Document Reader uses a bilinear term to capture the similarity between $\hat{p^i}$ and $\hat{q}$ and computes the probabilities of each paragraph term being start of the answer and end of the answer in Equation 3.7. Therefore, the text span of the answer is the maximum of $P^i_{start} \times P^{i'}_{end}$ and $i \leq i' \leq i+15$.

$$\{\hat{p^1},...,\hat{p^m}\} = BiLSTM(\{E(p^1),...,E(p^m)\}) \tag{3.5}$$

$$b_j = \frac{exp(w \cdot \hat{q_j})}{\sum_{j'} exp(w \cdot \hat{q_{j'}})} \tag{3.6}$$

$$P^i_{start} \propto exp(\hat{p^i} W_s \hat{q}) \quad P^i_{end} \propto exp(\hat{p^i} W_e \hat{q}) \tag{3.7}$$

Neural RC model of HotpotQA

The architecture of the RC model of HotpotQA can be found in Figure 3.11. This neural RC model is based on the RC model proposed by Clark and Gardner [20] with additional technical advances, including character-level models[20] [54], self-attention [128] and bi-attention [104].

[20]For RNN or LSTM, the input is each character of a word instead of the entire word.

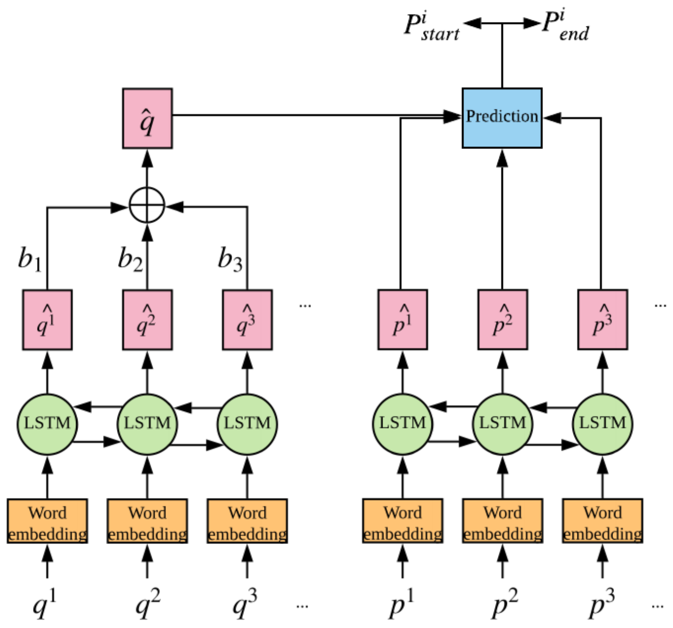


Figure 3.10: The architecture of Document Reader of DrQA.

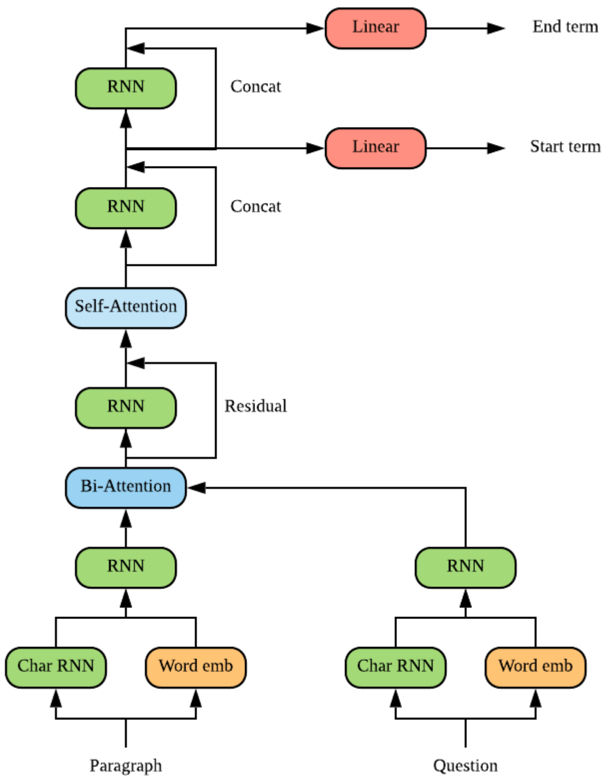


Figure 3.11: The architecture of the neural RC model of HotpotQA.

Chapter 4

Experiments for Validating Hypotheses

The goal of this chapter is to conduct experiments using methodologies we described in Section 3.5 for validating our hypotheses proposed in Section 3.4. The pipeline of our experiments is depicted in Figure 4.1.

We begin this chapter with the experimental setup, in which the hyperparameter tuning and training[1] for our methodologies are performed (Section 4.1). Then, we describe the evaluation of our methodologies on the development set of each dataset (Section 4.2) and we end up with this chapter by presenting the results and analysis of experiments for answering **RQ3** (Section 4.3).

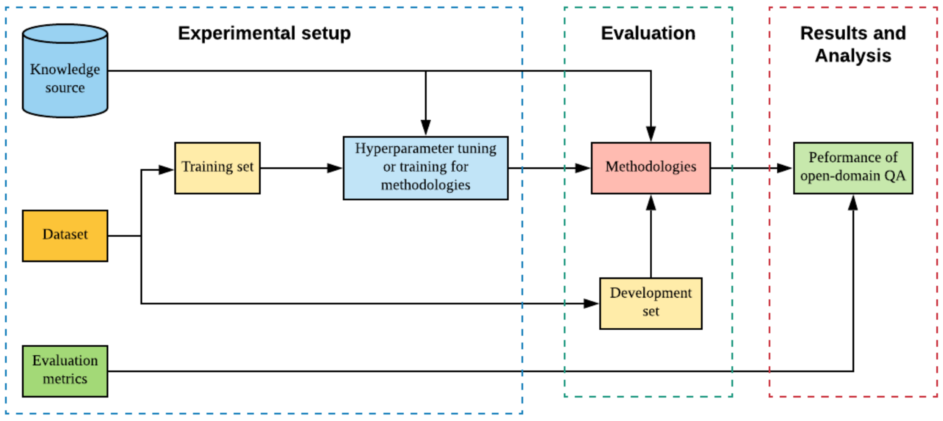


Figure 4.1: The pipeline of our experiments.

4.1 Experimental Setup

In this section, we describe the experimental setup for our experiments, including the datasets, evaluation metrics and traditional IR models we use in experiments. In ad-

[1]We only need to train neural Paragraph Ranker for the methodology that implements **H3**.

dition, this section also includes the hyperparameter tuning for methodologies we described in Section 3.5.

4.1.1 Datasets and The Knowledge Source

As we described in the retrieval test (Section 3.2), we used three different QA datasets SQuAD, HotpotQA and TriviaQA with the knowledge source Wikipedia to conduct the retrieval test in the open-domain setting. In this chapter, we attempt to use almost the same setup as that of the retrieval test in our experiments except that we do not conduct experiments on the dataset TriviaQA. This is due to the reason that our hypotheses proposed in Section 3.4 all concentrate on improving the retrieval performance of traditional IR models on datasets SQuAD and HotpotQA in the open-domain setting and we gave detailed explanations about why our hypotheses only focus on SQuAD and HotpotQA at the beginning of Section 3.4.

Therefore, we use datasets SQuAD and HotpotQA with the knowledge source Wikipedia in our experiments for validating our hypotheses. The statistics of datasets SQuAD and HotpotQA, as well as the description of Wikipedia, can be found in the retrieval test setup of Section 3.2.

4.1.2 Evaluation Metrics

As we described in the retrieval test (Section 3.2), we used the metric *top-k recall* (TOP-k) to evaluate the retrieval performance of traditional IR models across datasets in the open-domain setting, which indicates whether the ground truth answer to the question appears in the top-ranked *k* documents retrieved by the retrieval component. We employ TOP-k in our experiments to evaluate the retrieval performance of our methodologies in the task of open-domain QA.

In addition to the metric TOP-k, we make use of evaluation metrics *Exact Match* (EM) and *F1 score* (F1) to evaluate the QA performance of the RC model in the answer extraction stage of the open-domain QA system. EM and F1 can help us validate whether our hypotheses can benefit the RC model for achieving a better QA performance in the task of open-domain QA.

4.1.3 Traditional IR Models

In our experiments, we employ the same traditional IR models that were used in the retrieval test (Section 3.2) and they are TF-IDF, BM25 and QL. In addition, the relevance model RM3 is used for query expansion to implement **H2**. These traditional IR models are implemented using the open-source toolkit Anserini, which was also used in the retrieval test for implementing TF-IDF, BM25 and QL.

4.1.4 Hyperparameter Tuning

Traditional IR models BM25, QL and RM3 contain hyperparameters to be tuned for the best retrieval performance. We will describe the process of hyperparameter tuning based on each methodology we use in our experiments. Note that all hyperparameters are tuned on the training set of each dataset in the open-domain setting and then traditional IR models with the best-performing hyperparameters are evaluated on the development set of each dataset.

Paragraph and Sentence Based Retrieval Document Unit (H1)
According to the definitions of *paragraph* and *sentence* we described in the methodology for implementing **H1** (Section 3.5.1), we split all 5,075,182 Wikipedia articles into 37,281,878 paragraphs and 102,188,601 sentences respectively. Hence, a Wikipedia article contains 7.4 paragraphs and 20 sentences on average respectively. We then use Anserini to build two indexes for Wikipedia by considering each paragraph and each sentence as the retrieval document unit respectively. Finally, we perform hyperparameter tuning for traditional IR models BM25 and QL based on these two indexes.

In the retrieval test (Section 3.2), we performed hyperparameter tuning for BM25 and QL based on the index which considered each Wikipedia article as the retrieval document unit and hyperparameter tuning was performed on the training set of each dataset for the best retrieval performance of the metric TOP-10. The metric TOP-10 in the retrieval test was based on the top-ranked 10 Wikipedia articles retrieved by BM25 and QL, as the retrieval document unit was each Wikipedia article. Therefore, to do hyperparameter tuning on two indexes which consider each paragraph and each sentence of Wikipedia articles as the retrieval document unit respectively, we perform hyperparameter tuning on paragraph-based retrieval document unit and sentence-based retrieval document unit for the best retrieval performance of the metric TOP-74 and TOP-200 respectively, as 10 Wikipedia articles are estimated to have the same amount of text of 74 (7.4×10) paragraphs and 200 (20×10) sentences.

The overview of the range of hyperparameters' values can be found in Appendix A. The best-performing hyperparameters are adopted in our experiments and they can be found in Table 4.1. We denote a traditional IR model retrieving documents from Wikipedia considering each paragraph and each sentence of Wikipedia articles as the retrieval document unit by appending "*Para*" and "*Sent*" to its original name respectively (e.g., BM25+Para, BM25+Sent).

Model	Hyperparameter	SQuAD$_{open}$ (tuned)	HotpotQA$_{fullwiki}$ (tuned)
BM25+Para	$k1$	0.4	0.3
	b	0.3	0.5
QL+Para	λ	0.3	0.2
BM25+Sent	$k1$	0.2	0.4
	b	0.5	0.1
QL+Sent	λ	0.7	0.5

Table 4.1: The fine-tuned hyperparameters of BM25 and QL based on the indexes which consider each paragraph (+Para) and each sentence (+Sent) of Wikipedia articles as the retrieval document unit. Hyperparameters are tuned on the training set of each dataset in the open-domain setting.

Query Expansion by Pseudo Relevance Feedback (H2)
The relevance model RM3 contains three hyperparameters *fbDocs*, *fbTerms* and λ which we described in Section 2.2.2 and we make use of RM3 to do query expansion based on relevance feedback documents retrieved by traditional IR models TF-IDF, BM25 and QL. Hence, traditional IR models with query expansion by RM3 have three combinations and they are TF-IDF with RM3 (TF-IDF+RM3), BM25 with RM3 (BM25+RM3) and QL with RM3 (QL+RM3).

Note that the index for Wikipedia in this methodology considers each Wikipedia article as the retrieval document unit, so we perform hyperparameter tuning for RM3 to achieve the best TOP-10 based on fine-tuned hyperparameters of BM25 and QL which were fine-tuned in the retrieval test (Section 3.2). The overview of the range of hyperparameters' values can be found in Appendix A. The best-performing hyperparameters are adopted in our experiments and they can be found in Table 4.2.

Model	Hyperparameter	SQuAD$_{open}$ (tuned)	HotpotQA$_{fullwiki}$ (tuned)
TF-IDF+RM3	$fbDocs$	5	5
	$fbTerms$	200	200
	λ	0.9	0.9
BM25+RM3	k_1	0.1	0.3
	b	0.1	0.4
	$fbDocs$	5	15
	$fbTerms$	150	150
	λ	0.9	0.9
QL+RM3	λ_{QL}	0.1	0.1
	$fbDocs$	5	5
	$fbTerms$	100	100
	λ	0.5	0.7

Table 4.2: The fine-tuned hyperparameters of traditional IR models with RM3. Hyperparameters are tuned on the training set of each dataset in the open-domain setting.

Neural Paragraph Ranker (H3)
As we depicted the pipeline of the traditional IR model with Paragraph Ranker in Figure 3.8, the traditional IR model initially retrieves top-ranked *N* documents from Wikipedia for re-ranking by Paragraph Ranker. The knowledge source Wikipedia is indexed by considering each Wikipedia article as the retrieval document unit. Hence, the hyperparameters of BM25 and QL we adopt in this methodology are those tuned in the retrieval test (Section 3.2), which can be found in Table 3.6.

In addition to the hyperparameters of traditional IR models, Paragraph Ranker contains some hyperparameters to be decided for training, such as the number of layers of Bi-LSTMs, the number of retrieved documents *N* and dropout rate, etc. We adopt the same hyperparameters setting as that of the original work of Paragraph Ranker [63] for training Paragraph Ranker on datasets SQuAD and HotpotQA. Concretely, Paragraph Ranker uses 3-layer Bi-LSTM networks with 128 hidden units and Adamax [55] as the optimization algorithm. Dropout is applied to Bi-LSTMs and word embeddings with the dropout rate of 0.4 and Paragraph Ranker retrieves 20 documents from Wikipedia. We denote the traditional IR model with Paragraph Ranker by appending "*Ranker*" to its original name (e.g., BM25+Ranker).

4.2 Evaluation

In this section, we aim to evaluate the performance of our methodologies in the task of open-domain QA on datasets SQuAD and HotpotQA, including the retrieval performance and the QA performance of each methodology.

4.2.1 Paragraph and Sentence Based Retrieval Document Unit (H1)

We evaluate the performance of traditional IR models TF-IDF, BM25 and QL in the task of open-domain QA on datasets SQuAD and HotpotQA and these IR models have paragraph-based and sentence-based retrieval document units for Wikipedia. Specifically, we evaluate the retrieval performance of traditional IR models by the metric TOP-37 (i.e., top-ranked 5 articles) and TOP-74 (i.e., top-ranked 10 articles) based on Wikipedia considering paragraph-based retrieval document unit and TOP-100 (i.e., top-ranked 5 articles) and TOP-200 (i.e., top-ranked 10 articles) based on Wikipedia considering sentence-based retrieval document unit and this is due to two reasons. On one hand, we need to compare the retrieval performance of this methodology with that of traditional IR models in the retrieval test (Section 3.2). On the other hand, as we described in the experimental setup (Section 4.1), a Wikipedia article contains 7.4 paragraphs and 20 sentences on average respectively.

Subsequently, we evaluate the QA performance by Document Reader of DrQA and the neural RC model of HotpotQA for SQuAD and HotpotQA respectively based on the retrieved documents above. The descriptions of Document Reader and the neural RC model of HotpotQA are described in Section 3.5.4. To fairly compare the QA performance of this methodology with that of original work of DrQA [17] and HotpotQA [136], RC models need to extract the answer from the same amount of text used in the original work of DrQA and HotpotQA[2].

4.2.2 Query Expansion by Pseudo Relevance Feedback (H2)

The index for Wikipedia for this methodology considers each Wikipedia article as the retrieval document unit, hence we evaluate the retrieval performance of traditional IR models by the metric TOP-5 and TOP-10. Also, we use Document Reader of DrQA and the neural RC model of HotpotQA to evaluate the QA performance on SQuAD and HotpotQA respectively.

4.2.3 Neural Paragraph Ranker (H3)

We use traditional IR models to retrieve top-ranked 20 articles from Wikipedia and employ Paragraph Ranker to re-rank all paragraphs of these 20 Wikipedia articles. As

[2]In their works, Document Reader of DrQA extracts the answer from top-ranked 5 articles, while the RC model of HotpotQA extracts the answer from top-ranked 10 paragraphs.

a result of re-ranking by Paragraph Ranker, we select top-ranked 37 paragraphs and top-ranked 74 paragraphs to represent the same amount of text of top-ranked 5 articles and top-ranked 10 articles respectively. We evaluate the retrieval performance and the QA performance of this methodology based on the retrieved paragraphs.

The number of documents retrieved by traditional IR models from Wikipedia for re-ranking is 20 in the original work of Paragraph Ranker [63]. Therefore, to explore the impact of the number of documents retrieved by traditional IR models from Wikipedia for re-ranking on the performance of this methodology, we evaluate the retrieval performance and the QA performance of traditional IR models with Paragraph Ranker based on the increased number of documents retrieved by traditional IR models. However, we cannot increase the number of retrieved documents without considering the computational memory that Paragraph Ranker requires. Hence, we evaluate the performance of this methodology on SQuAD and HotpotQA from retrieving 20 documents to 200 documents with the size of 20 documents increasing.

4.3 Results and Analysis

In this section, we aim to present the experimental results of our methodologies on datasets SQuAD and HotpotQA in the task of open-domain QA. Based on the experimental results, we give an analysis of each methodology for validating whether the corresponding hypothesis is able to improve the retrieval performance of traditional IR models on SQuAD and HotpotQA in the open-domain setting. Consequently, we can give the answer to our **RQ3** based on the results and analyses.

4.3.1 Results and Analysis Based on H1

The experimental results of the methodology that implements **H1** can be found in Table 4.3 and this methodology is based on Wikipedia considering paragraph-based and sentence-based retrieval document units.

Analysis of the Results on SQuAD in the open-domain setting
As we can see from the results on SQuAD in the open-domain setting in Table 4.3, traditional IR models TF-IDF, BM25 and QL have better retrieval performance (i.e., TOP-5 and TOP-10) and QA performance (i.e., EM and F1) with smaller retrieval document units for Wikipedia (i.e., paragraph-based and sentence-based retrieval document unit) over traditional IR models with the retrieval document unit of each Wikipedia article. Specifically, TF-IDF with sentence-based retrieval document unit achieves the best performance over TF-IDF with article-based and paragraph-based retrieval document unit. However, in terms of BM25 and QL, they both achieve the best performance with paragraph-based retrieval document unit instead of the sentence-based retrieval document unit. Overall, BM25 with paragraph-based retrieval document unit achieves not only the best retrieval performance but also the best QA performance compared to other retrieval components in Table 4.3. Furthermore, BM25 with paragraph-based retrieval document unit achieves EM of 28.62 which is better than EM of 27.1 of the original work of DrQA [17].

Model	SQuAD$_{open}$				HotpotQA$_{fullwiki}$			
	TOP-5	TOP-10	EM	F1	TOP-5	TOP-10	EM	F1
TF-IDF	10.45	17.21	2.38	5.54	21.48	27.15	3.76	8.11
TF-IDF+Para	17.84	24.90	8.07	11.91	28.96	34.49	6.16	12.80
TF-IDF+Sent	**43.34**	**49.90**	**20.15**	**25.12**	**33.87**	**39.54**	**7.99**	**15.57**
BM25	68.06	74.33	18.83	24.04	60.92	67.68	11.21	20.73
BM25+Para	**80.49**	**84.63**	**28.62**	**35.38**	**64.60**	**69.70**	**15.23**	**26.86**
BM25+Sent	74.09	77.66	23.93	29.98	55.10	60.10	11.73	22.53
QL	56.86	66.04	14.31	20.05	58.76	65.51	10.95	20.41
QL+Para	**77.22**	**81.97**	**27.13**	**33.70**	**63.19**	**68.85**	**14.97**	**26.10**
QL+Sent	73.23	77.12	23.93	29.96	53.82	58.92	12.16	22.73

Table 4.3: The experimental results of the methodology on SQuAD and HotpotQA in the open-domain setting, which is based on Wikipedia considering paragraph-based (+Para) and sentence-based (+Sent) retrieval document units. For each traditional IR model, the best performance on each metric is in bold. Note that the performances of "TF-IDF", "BM25" and "QL" are ones with article-based retrieval document unit from the retrieval test in Table 3.7.

The reason why TF-IDF with paragraph-based retrieval document unit does not achieve better performance than TF-IDF with sentence-based retrieval document unit is that the performance of TF-IDF is sensitive to the term frequency in documents. As we described TF-IDF in Section 2.2.2, TF-IDF weighting scheme is dependent on the combination of term frequency and inverse document frequency. In our work, a paragraph contains 2.7 sentences on average[3], so a paragraph contains a larger number of terms in text than a sentence. Therefore, TF-IDF with sentence-based retrieval document unit is able to match question terms with relevant documents easier than TF-IDF with paragraph-based or article-based retrieval document unit, as each document to be evaluated is shorter in text and does not have too much distracting text for TF-IDF. This is also the reason why BM25 and QL with sentence-based retrieval document unit can have better performance than they with article-based retrieval document unit.

However, BM25 and QL with paragraph-based retrieval document unit perform better than they with sentence-based retrieval document unit. This is due to the reasons as follows.

(1) Unlike TF-IDF, BM25 is not that sensitive to term frequency in documents, as term frequency of BM25 can be scaled by document length with the hyperparameter b. This can be found in Equation 2.8 which we described in Section 2.2.2.

(2) Also, for BM25 and QL, the sentence-based retrieval document unit has a drawback which is that BM25 and QL are more likely to miss some important documents that might contain the answer to the question. This is due to the document

[3]As we described in Section 4.1, Wikipedia in our work contains 37,281,878 paragraphs or 102,188,601 sentences.

sparseness for the question based on small retrieval document units [72]. For example, a question from SQuAD is "*Where did Super Bowl 50 take place?*" and its supporting reasoning facts are given as follows.

A: "*Super Bowl 50 was an American football game to determine the champion of the National Football League (NFL) for the 2015 season.*"

B: "*The American Football Conference (AFC) champions Denver Broncos defeated the National Football Conference (NFC) champions Carolina Panthers, 24–10.*"

C: "*The game was played on February 7, 2016, at Levi's Stadium in Santa Clara, California, in the Bay Area.*"

Based on the sentence-based retrieval document unit, BM25 and QL are more likely to miss documents B and C since they do not contain any query term, but the answer to this question appears in document C (i.e., "*Levi's Stadium*"). Hence, this drawback decreases the performance of BM25 and QL with sentence-based retrieval document unit.

Analysis of the Results on HotpotQA in the open-domain setting
As we can see from the results on HotpotQA in the open-domain setting in Table 4.3, the case of traditional IR models on HotpotQA is similar to that of traditional IR models on SQuAD. Concretely, TF-IDF with sentence-based retrieval document unit performs better than TF-IDF with paragraph-based and article-based retrieval document, while BM25 and QL with paragraph-based retrieval document unit have better performance than they with sentence-based and article-based retrieval document unit. The reasons for this can be found in the analysis of results on SQuAD above. An example of document sparseness based on sentence-based retrieval document unit for HotpotQA is given as follows. For a question of HotpotQA "*Brown State Fishing Lake is in a county that has a population of how many inhabitants?*", some supporting reasoning facts of this question are the following:

A: "*Brown County (county code BR) is a county located in the northeast portion of the U.S. state of Kansas.*"

B: "*As of the 2010 census, the county population was 9,984.*"

Based on the sentence-based retrieval document unit, BM25 and QL might not be able to retrieve document B which contains the answer "*9,984*", despite document B contains question term "*county*" and "*population*". If BM25 and QL retrieve documents from Wikipedia based on paragraph-based retrieval document unit, these two sentences are more likely to be retrieved by BM25 and QL, as the paragraph containing these two sentences has more question terms.

Furthermore, we find that even though BM25 and QL with sentence-based retrieval document unit have worse retrieval performance than they with article-based retrieval document unit (e.g., for BM25, TOP-5: 55.10 vs 60.92, TOP-10: 60.10 vs 67.68), BM25 and QL with sentence-based retrieval document unit still achieve better QA performance than they with article-based retrieval document unit (e.g., for BM25, EM: 11.73 vs 11.21, F1: 22.53 vs 20.73). This is due to that the amount of text for TOP-5 (i.e., TOP-100 sentences) and TOP-10 (i.e., TOP-200 sentences) based on sentence-based retrieval document unit is less than that based on article-based retrieval document unit. As we described in the experimental setup (Section 4.1), a Wikipedia article is averagely equivalent to 20 sentences in the amount of text. However, for the questions of HotpotQA in the open-domain setting, the average amount of text of a Wikipedia article retrieved by BM25 and QL contains a larger number of sentences than 20. We use BM25 and QL with article-based retrieval document unit to retrieve top-ranked 5 articles for each question of HotpotQA (6947 questions) respectively, so the number of articles retrieved by BM25 and QL is both 34,735. We split these articles into sentences and they end up with 3,609,386 sentences and 2,644,169 sentences for BM25 and QL respectively, so the average number of sentences of a Wikipedia article retrieved by BM25 and QL for questions of HotpotQA is 103.9 and 76.1 respectively, which are both larger than 20. Despite BM25 and QL with sentence-based retrieval document unit retrieve less text for measuring the retrieval performance, they still have higher EM and F1. This indicates that relevant documents of questions of HotpotQA can be retrieved and ranked by BM25 and QL with sentence-based retrieval document unit to the top positions.

Overall, BM25 with paragraph-based retrieval document unit achieves the best retrieval and QA performance and its EM is 15.23 which is lower than the EM of 24.68 in the original work of HotpotQA [136]. This is due to the reason that the knowledge source (i.e., Wikipedia) in our work is broader and contains much more documents than the Wikipedia used in the original work of HotpotQA. As we described the knowledge source in the retrieval test (Section 3.2), the Wikipedia in the original work of HotpotQA only retains the first paragraph of each Wikipedia article for retrieval, so our work is much more difficult than the original work of HotpotQA.

4.3.2 Results and Analysis Based on H2

The experimental results of the methodology that implements **H2** can be found in Table 4.4 and this methodology is based on the relevance model RM3 to expand relevant terms from retrieved documents to questions for refinement.

Analysis of the Results on SQuAD in the open-domain setting

As we can see from the results on SQuAD in the open-domain setting in Table 4.4, the relevance model RM3 is beneficial to the traditional IR models BM25 and QL, whereas TF-IDF with RM3 has the performance degradation. This is due to the evidence found in many previous works [123, 141], which is that the performance of query expansion based on pseudo relevance feedback strongly relies on the quality of the documents retrieved by the retrieval component in the initial round. We depicted the retrieval process of the traditional IR model with RM3 in Figure 3.7. From the results, we can see that the quality of initially retrieved documents by TF-IDF is much worse than that of documents initially retrieved by BM25 and QL (e.g., TOP-5: 10.45 vs 68.06 vs 56.86). Hence, RM3 decreases the retrieval performance of TF-IDF.

Overall, BM25 with RM3 achieves the best retrieval and QA performance than other traditional IR models with RM3, but its EM is still lower than that of the original work of DrQA (19.16 vs 27.1).

Model	SQuAD$_{open}$				HotpotQA$_{fullwiki}$			
	TOP-5	TOP-10	EM	F1	TOP-5	TOP-10	EM	F1
TF-IDF	**10.45**	**17.21**	**2.38**	**5.54**	**21.48**	**27.15**	**3.76**	**8.11**
TF-IDF+RM3	9.28	15.23	2.11	5.04	18.11	24.60	2.81	6.63
BM25	68.06	74.33	18.83	24.04	**60.92**	**67.68**	**11.21**	**20.73**
BM25+RM3	**69.12**	**75.46**	**19.16**	**25.33**	60.46	66.94	10.97	20.54
QL	56.86	66.04	14.31	20.05	**58.76**	**65.51**	**10.95**	**20.41**
QL+RM3	**62.66**	**73.81**	**17.03**	**22.91**	58.10	65.04	10.72	20.15

Table 4.4: The experimental results of the methodology on SQuAD and HotpotQA in the open-domain setting, which employs RM3 (+RM3) to do query (i.e., question) expansion. For each traditional IR model, the best performance on each metric is in bold. Note that the performances of "TF-IDF", "BM25" and "QL" are ones with article-based retrieval document unit from the retrieval test in Table 3.7.

Analysis of the Results on HotpotQA in the open-domain setting

As we can see from the results on HotpotQA in the open-domain setting in Table 4.4, none of the traditional IR models can benefit from RM3 on the retrieval performance or QA performance. As we analyzed in the error analysis in Section 3.3, the overview of the error analysis in Table 3.8 indicates that the error type *B1: No name of bridge entity (bridge)* is the most prominent error type for HotpotQA. In this type, the real name of the bridge entity does not appear in the question and the question only has the descriptive phrase to represent the bridge entity instead. For example, the question of HotpotQA "*the birthplace of George McCall Theal is a port city of what bay?*" belongs to this error type containing the descriptive phrase of the bridge entity "*a port city*". However, in this error type, traditional IR models are able to retrieve the ground truth article of the unambiguous entity "*McCall Theal*", in which the real name of the bridge entity appears. Therefore, we expected to employ RM3 to expand the terms of the real name or some other terms that are related to the bridge entity to the question, then traditional IR models are more likely to retrieve the ground truth article of the bridge entity that contains the answer based on the revised question.

However, the results of traditional IR models with RM3 on HotpotQA demonstrate that this is not the case as we expected above. Therefore, we check the ground truth article of the unambiguous entity "*McCall Theal*". We find that this article is long in the text which contains 666 words with 302 unique words, but the real name of the bridge entity "*Saint John, NewBrunswick*" only appears once in this article. Therefore, it is difficult for RM3 to expand the term in the real name of the bridge entity to the question, as the term has a low term frequency in the document.

4.3.3 Results and Analysis Based on H3

The experimental results of the methodology that implements **H3** can be found in Table 4.5 and this methodology employs a neural ranker to re-rank all paragraphs within the documents retrieved by the traditional IR model.

Model	SQuAD$_{open}$				HotpotQA$_{fullwiki}$			
	TOP-5	TOP-10	EM	F1	TOP-5	TOP-10	EM	F1
TF-IDF	10.45	17.21	2.38	5.54	21.48	27.15	3.76	8.11
TF-IDF+Ranker	**23.99**	**25.15**	**4.82**	**9.10**	**34.27**	**34.81**	**8.45**	**14.65**
BM25	**68.06**	**74.33**	18.83	24.04	**60.92**	**67.68**	**11.21**	**20.73**
BM25+Ranker	63.70	69.01	**22.73**	**28.76**	56.60	63.51	10.05	16.52
QL	56.86	66.04	14.31	20.05	**58.76**	**65.51**	**10.95**	**20.41**
QL+Ranker	**63.16**	**68.13**	**19.73**	**25.90**	57.90	64.16	10.38	16.92

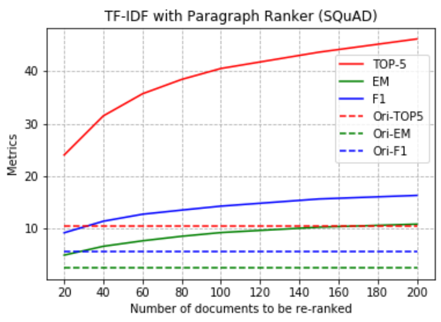
Table 4.5: The experimental results of the methodology on SQuAD and HotpotQA in the open-domain setting, which employs Paragraph Ranker (+Ranker) to re-rank all paragraphs of top-ranked 20 Wikipedia articles retrieved by the traditional IR model. For each traditional IR model, the best performance on each metric is in bold. Note that the performances of "TF-IDF", "BM25" and "QL" are ones with article-based retrieval document unit from the retrieval test in Table 3.7.

Analysis of the Results on SQuAD in the open-domain setting

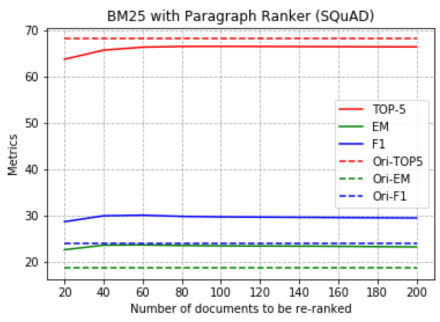
As we can see from the results on SQuAD in the open-domain setting in Table 4.5, Paragraph Ranker help TF-IDF and QL improve the retrieval performance and the QA performance. Also, we can find that even though BM25 with Paragraph Ranker obtains worse retrieval performance than BM25 without Paragraph Ranker (e.g, TOP-5: 63.70 vs 68.06), BM25 with Paragraph Ranker still has a better QA performance than BM25 without Paragraph Ranker (e.g., EM: 22.73 vs 18.83). This is due to the reason that the amount of text retrieved by BM25 with Paragraph Ranker for measuring retrieval performance is not fully equivalent to the amount of text retrieved by BM25 without Paragraph Ranker. Concretely, as we described in the experimental setup (Section 4.1), a Wikipedia article is averagely equivalent to 7.4 paragraphs in the amount of text. However, the average number of paragraphs of a Wikipedia article retrieved by BM25 without Paragraph Ranker for questions of SQuAD is much larger than 7.4. We make use of BM25 without Paragraph Ranker to retrieve 5 Wikipedia articles for each question of SQuAD (10,570 questions) and we split these articles (52,850 articles) into paragraphs. Finally, we end up with retrieving 6,504,151 paragraphs in total, so the average number of paragraphs of a Wikipedia article retrieved for questions of SQuAD is 123.1. Therefore, BM25 with Paragraph Ranker retrieves less text for measuring the retrieval performance than BM25 without Paragraph Ranker, but it achieves a better QA performance. This indicates that the relevant documents of questions of SQuAD can be ranked by Paragraph Ranker to top positions.

Despite Paragraph Ranker only re-ranks paragraphs of top-ranked 20 Wikipedia articles retrieved by TF-IDF, BM25 and QL, these traditional IR models with Paragraph Ranker can still outperform these traditional IR models without Paragraph Ranker. Furthermore, the impact of the number of Wikipedia articles for re-ranking by Paragraph Ranker on the performance of traditional IR models can be found in Figure 4.2.

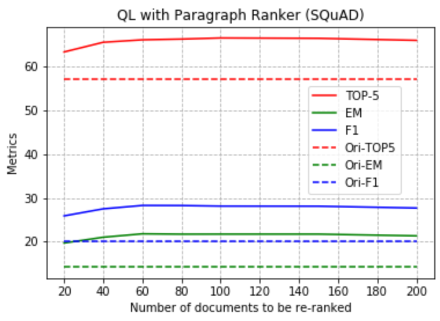
As we can see from Figure 4.2, Figure 4.2a and Figure 4.2c, which indicate TF-IDF and QL with Paragraph Ranker respectively, show that the retrieval performance



a *Performances of TF-IDF w/o Paragraph Ranker on SQuAD in the open-domain setting*



b *Performances of BM25 w/o Paragraph Ranker on SQuAD in the open-domain setting*



c *Performances of QL w/o Paragraph Ranker on SQuAD in the open-domain setting*

Figure 4.2: The performance of traditional IR models on SQuAD in the open-domain setting with Paragraph Ranker re-ranking the increased number of documents. The metrics with "Ori-" indicate that these performances are under the traditional IR model without Paragraph Ranker.

(i.e., TOP-5) and the QA performance (i.e., EM and F1) are consistently better than that of these IR models without Paragraph Ranker based on the increased number of retrieved documents for re-ranking by Paragraph Ranker. Specifically, the performance gap (Figure 4.2a) between TF-IDF with Paragraph Ranker and TF-IDF without Paragraph Ranker increases continuously with respect to the increased number of retrieved documents. In terms of the performance gap (Figure 4.2c) between QL with and without Paragraph Ranker, it is not as significant as that of Figure 4.2a. The performance gap on EM and F1 keeps stable after about 60 retrieved documents and the performance gap on TOP-5 does not increase after around 100 retrieved documents. Therefore, TF-IDF can benefit more significantly from Paragraph Ranker than QL, as TF-IDF without Paragraph Ranker performs worse than QL without Paragraph Ranker (e.g., TOP-5: 10.45 vs 56.86). As a result, TF-IDF can be easier and have a larger space to be improved by Paragraph Ranker than QL. Furthermore, another reason is that the top-ranked documents retrieved by TF-IDF without Paragraph Ranker contain a small amount of text. For example, three top-ranked documents retrieved by TF-IDF without Paragraph Ranker for the question of SQuAD "*Which NFL team rep-*

resented the AFC at Super Bowl 50?" are Wikipedia articles of "*NFL playoff records (team)*", "*NFL Honors*" and "*Johnny Rembert*". The total number of paragraphs of these three articles is 3, which means each article only has one paragraph in the text. However, three top-ranked documents[4] retrieved by QL without Paragraph Ranker for the same question contain 91 paragraphs in total. Hence, with the increased number of documents for re-ranking by Paragraph Ranker, TF-IDF can have more significant performance improvements, as there are more and more paragraphs to be re-ranked by Paragraph Ranker to find more relevant documents for TF-IDF.

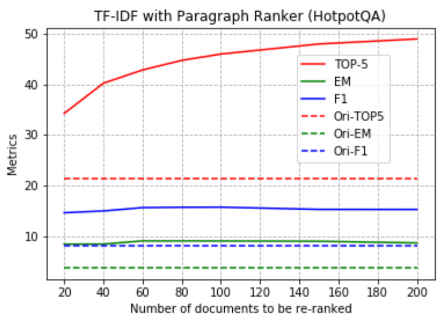
Figure 4.2b shows the performance of BM25 with and without Paragraph Ranker. We can see that the retrieval performance of BM25 with Paragraph Ranker is constantly lower than that of BM25 without Paragraph Ranker, whereas the QA performance of BM25 with Paragraph Ranker is consistently better than that of BM25 without Paragraph Ranker. This is due to the reason that the amount of text retrieved by BM25 with and without Paragraph Ranker for measuring the retrieval performance is not fully equivalent, as we described above. This also indicates that Paragraph Ranker is able to re-rank relevant documents for questions of SuAD to top positions. Furthermore, the performance gap on EM and F1 does not increase after ranking 60 retrieved documents.

Analysis of the Results on HotpotQA in the open-domain setting
As we can see from the results on HotpotQA in the open-domain setting in Table 4.5, when Paragraph Ranker re-ranks all paragraphs of top-ranked 20 Wikipedia articles retrieved by traditional IR models, it only helps TF-IDF improve the retrieval performance and the QA performance. BM25 and QL with Paragraph Ranker have performance degradation compared to BM25 and QL without Paragraph Ranker. To see whether Paragraph Ranker can improve the performance of BM25 and QL based on the different number of retrieved documents for re-ranking, we test the performance of traditional IR models with Paragraph Ranker based on the increased number of retrieved documents for re-ranking and the results can be found in Figure 4.3.

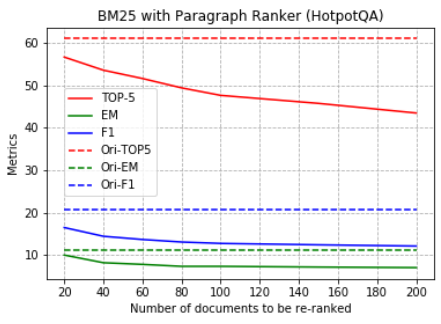
As we can see from Figure 4.3, Paragraph Ranker can still improve the performance of TF-IDF, while it has a negative impact on BM25 and QL. The reasons for the different impacts of Paragraph Ranker between TF-IDF and the other two traditional IR models are described as follows.

(1) As we can see from Figure 4.3a, TF-IDF obtains performance improvements with Paragraph Ranker and the performance gap keeps stable after a certain number of documents. Concretely, the EM and F1 of TF-IDF with Paragraph Ranker do not appear to increase after 80 documents and even appear to decrease after 150 documents. In contrast, the retrieval performance TOP-5 of TF-IDF with Paragraph Ranker keeps increasing from 20 documents to 200 documents. This indicates that many paragraphs re-ranked by Paragraph Ranker at the top positions are *false positive* paragraphs, which only contain the answer to the question, but cannot be used for the RC model to extract the answer correctly. Hence, the QA performance of TF-IDF with Paragraph Ranker does not appear to increase and even drops based on these false positive paragraphs. These false

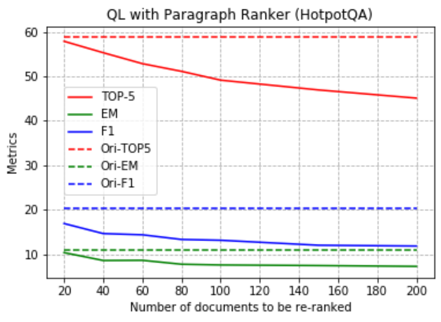
[4]They are Wikipedia articles of "*Super Bowl LI*", "*Super Bowl*" and "*NFL (video game)*".



a *Performance of TF-IDF w/o Paragraph Ranker on HotpotQA in the open-domain setting*



b *Performance of BM25 w/o Paragraph Ranker on HotpotQA in the open-domain setting*



c *Performance of QL w/o Paragraph Ranker on HotpotQA in the open-domain setting*

Figure 4.3: The performance of traditional IR models on HotpotQA in the open-domain setting with Paragraph Ranker re-ranking increased number of documents. The metrics with "Ori-" indicate that these performances are under the traditional IR model without Paragraph Ranker.

positive paragraphs demonstrate that Paragraph Ranker is not robust to some noisy paragraphs that contain the ground truth answer to the question but cannot be used to answer the question successfully.

(2) Paragraph Ranker's inability of robustness to false positive paragraphs becomes more obvious when it needs to re-rank more amount of text. In Figure 4.3a, TF-IDF with Paragraph Ranker has the QA performance degradation after re-ranking around 150 documents. To demonstrate that this inability is severer when re-ranking a larger amount of text, we make use of Paragraph Ranker to re-rank 500 documents retrieved by TF-IDF for each question of HotpotQA. As a result of re-ranking 500 retrieved documents for each question, we find that compared to the performance of TF-IDF with Paragraph Ranker re-ranking 200 documents, not only the QA performance decreases (i.e., EM: 8.69 vs 7.97, F1: 15.30 vs 14.14), but also the retrieval performance decreases (i.e., TOP-5: 48.97 vs 48.41). In this case, Paragraph Ranker even ranks some paragraphs that do not contain the answer to the top-ranked positions. In terms of the amount of text

of a certain number of documents retrieved by BM25 and QL, it is larger than that of the same number of documents retrieved by TF-IDF[5]. Therefore, BM25 and QL have performance degradation with Paragraph Ranker, as the documents retrieved by BM25 and QL have much text to be re-ranked by Paragraph Ranker, so BM25 and QL with Paragraph Ranker suffer from the performance drop easily with just re-ranking a few retrieved documents. For instance, as we can see from the results on HotpotQA in Table 4.5, the performance of BM25 and QL with Paragraph Ranker re-ranking just 20 retrieved documents is worse than that of BM25 and QL without Paragraph Ranker.

4.4 Conclusion

Based on the results and the analysis of each experiment above, we can draw some conclusions of the validity of our hypotheses proposed in Section 3.4 as follows, thereby answering **RQ3** that we described in Section 1.2.

- In the task of open-domain QA on datasets SQuAD and HotpotQA, the knowledge source Wikipedia considers each paragraph of articles or each sentence of articles as the retrieval document unit of traditional IR models TF-IDF, BM25 and QL and these two units can improve the retrieval performance of these traditional IR models. Specifically, TF-IDF can benefit more from sentence-based retrieval document unit, while BM25 and QL can have more retrieval performance improvements based on paragraph-based retrieval document unit. **In conclusion, our H1 is valid on datasets SQuAD and HotpotQA in the open-domain setting**.

- In the task of open-domain QA on datasets SQuAD and HotpotQA, the relevance model RM3 can benefit the traditional IR models BM25 and QL on SQuAD in the open-domain setting. However, traditional IR model TF-IDF with RM3 cannot be improved the retrieval performance on SQuAD or HotpotQA in the open-domain setting. In addition, BM25 and QL do not benefit from RM3 on HotpotQA in the open-domain setting. **In conclusion, our H2 based on the relevance model RM3 for query expansion is only valid to traditional IR models BM25 and QL on dataset SQuAD in the open-domain setting**.

- In the task of open-domain QA on datasets SQuAD and HotpotQA, adding a neural ranker named Paragraph Ranker[6] after traditional IR models for re-ranking can improve the retrieval performance of traditional IR models TF-IDF, BM25 and QL on SQuAD in the open-domain setting. Also, Paragraph Ranker can help TF-IDF improve the retrieval performance on HotpotQA in the open-domain setting when re-ranking a certain number of documents retrieved by TF-IDF. However, Paragraph Ranker following BM25 and QL decreases their retrieval performance on HotpotQA in the open-domain setting. **In conclusion, our H3 based on adding Paragraph Ranker after traditional IR models is**

[5]For example, TF-IDF, BM25 and QL retrieve 40 documents for each question of HotpotQA and they contain 789,792, 7,984,922 and 6,225,160 paragraphs in total respectively.

[6]Paragraph Ranker was proposed by Lee et al. [63].

valid to traditional IR models TF-IDF, BM25 and QL on dataset SQuAD in the open-domain setting as well as TF-IDF on dataset HotpotQA in the open-domain setting, but it is not valid to BM25 and QL on HotpotQA in the open-domain setting. Therefore, we can learn that some previous works, such as the work of Paragraph Ranker [63], proposed the neural ranker, which is not applicable and beneficial to all questions of different QA datasets.

Chapter 5

Conclusion and Future Work

In this chapter, we give a conclusion of our work (Section 5.1) and then we describe some aspects that are involved with future work (Section 5.2).

5.1 Conclusion

Our work focused on the task of retrieval-based open-domain question answering (QA), which enables the retrieval component to retrieve relevant documents from a large-scale knowledge source for a question and the answer extraction component to extract the answer to this question based on the retrieved documents. As many previous works focused on the answer extraction component, our work concentrated on the retrieval component and explored the impact on the retrieval component across different QA datasets, which previous works did not explore. Therefore, we proposed three research questions in our work to explore and mitigate the negative impact on the retrieval component across QA datasets.

We selected QA datasets SQuAD, HotpotQA and TriviaQA in our work that all consider the full Wikipedia as the knowledge source. In terms of the retrieval component in the task of open-domain QA, traditional but efficient information retrieval (IR) models TF-IDF, BM25 and QL were usually considered as the retrieval component to retrieve documents from a large-scale knowledge source in many previous works. We made use of these traditional IR models to conduct the retrieval test on datasets SQuAD, HotpotQA and TriviaQA in the open-domain setting based on the knowledge source Wikipedia. As a result, we found that different QA datasets have different impacts on the traditional IR models and are differently hard to be dealt with by the traditional IR models.

Based on the results of the retrieval test on datasets SQuAD, HotpotQA and TriviaQA in the open-domain setting, we conducted the error analysis of these three datasets for figuring out the error types of questions of each dataset. From error analysis, we learned that the prominent error types of questions of SQuAD, HotpotQA and QL in the open-domain setting are based on questions with *ambiguous* entities, *bridge* entities and *unambiguous* entities respectively. The error types based on ambiguous entities and bridge entities are much more difficult to be handled by traditional IR models than the error types based on unambiguous entities, so the retrieval performance of traditional IR models in our work on TriviaQA in the open-domain setting is much better than the retrieval performance on SQuAD and HotpotQA in the open-domain setting. Therefore, we hypothesized three methodologies that might mitigate the negative impact of the error types of questions of SQuAD and HotpotQA in the open-domain setting, thereby improving the retrieval performance of traditional IR models on SQuAD and HotpotQA in the open-domain setting.

We implemented our methodologies that are involved with our proposed hypotheses and they are based on smaller retrieval document units of traditional IR models, the relevance model RM3 and a neural Ranker *Paragraph Ranker* respectively. We then conducted experiments with these methodologies on datasets SQuAD and HotpotQA in the open-domain setting for validating these hypotheses. From our experiments, we found that not all of the hypotheses we proposed can mitigate the negative impact of the error types of questions of SQuAD and HotpotQA in the open-domain setting. Traditional IR models with smaller retrieval document units (i.e., paragraph-based or sentence-based retrieval document unit) can have performance improvements on both datasets SQuAD and HotpotQA in the open-domain setting, whereas this is not the case for traditional IR models with the relevance model RM3 for query expansion and with Paragraph Ranker for re-ranking documents retrieved by traditional IR models. Traditional IR models based on RM3 and Paragraph Ranker can have performance improvements on SQuAD instead of HotpotQA in the open-domain setting.

In conclusion, we believe that our work is a step forward to obtaining more insights into the retrieval component of the open-domain QA system and will contribute to the development of the retrieval component for a better open-domain QA system. Moreover, our work can give our users guidance on how to issue a more suitable question that can be processed by the open-domain QA system for giving a more accurate and better answer.

5.2 Future Work

For the future work, there are some aspects related to our work can be concentrated on to either improve or extend our research. Hence, we describe these relevant aspects as follows.

5.2.1 Performance Improvement on A Specific Error Type

In our work, we proposed hypotheses that are aimed at mitigating the error types of questions of datasets SQuAD and HotpotQA in the open-domain setting, thereby improving the retrieval performance of traditional IR models in our work on SQuAD and HotptoQA in the open-domain setting. For performance improvements on SQuAD and HotpotQA in the open-domain setting by some methodologies in our work, we can analyze the performance improvement based on a specific error type of questions for future work. As a result of this, we can have a clear insight into whether the methodology we employ can benefit this specific error type.

5.2.2 Integration of Our Methodologies

As we can see from our experimental results (Section 4.3), some traditional IR models in our work with RM3 and Paragraph Ranker do not showcase the performance improvement on datasets HotpotQA in the open-domain setting. For future work, we can integrate traditional IR models based on paragraph-based or sentence-based retrieval document unit with the relevance model RM3 or Paragraph Ranker to see whether the integration of our methodologies can improve the retrieval performance of these traditional IR models on HotpotQA in the open-domain setting.

5.2.3 Impact of Different Aspects of Questions

In our work, we explored the impact on the traditional IR models across datasets SQuAD, HotpotQA and TriviaQA in the open-domain setting. We focused on the characteristics[1] of entities that appear in the questions of these three datasets. For future work, we would like to collect more different questions and attempt to explore the impacts of different aspects of questions on traditional IR models. For example, we can explore the impacts on traditional IR models between short questions and long questions in the text, between cross-lingual questions and single-lingual questions, etc.

[1]Questions with ambiguous, bridge and unambiguous entities in the open-domain setting.

Bibliography

[1] Akiko Aizawa. An information-theoretic perspective of tf–idf measures. *Information Processing & Management*, 39(1):45–65, 2003.

[2] Ali Mohamed Nabil Allam and Mohamed Hassan Haggag. The question answering systems: A survey. *International Journal of Research and Reviews in Information Sciences (IJRRIS)*, 2(3), 2012.

[3] Stanislaw Antol, Aishwarya Agrawal, Jiasen Lu, Margaret Mitchell, Dhruv Batra, C Lawrence Zitnick, and Devi Parikh. Vqa: Visual question answering. In *Proceedings of the IEEE international conference on computer vision*, pages 2425–2433, 2015.

[4] Sören Auer, Christian Bizer, Georgi Kobilarov, Jens Lehmann, Richard Cyganiak, and Zachary Ives. Dbpedia: A nucleus for a web of open data. In *The semantic web*, pages 722–735. Springer, 2007.

[5] Hiteshwar Kumar Azad and Akshay Deepak. Query expansion techniques for information retrieval: a survey. *CoRR*, abs/1708.00247, 2017. URL `http://arxiv.org/abs/1708.00247`.

[6] Rodger Benham, J Shane Culpepper, Luke Gallagher, Xiaolu Lu, and Joel Mackenzie. Towards efficient and effective query variant generation. In *Proceedings of the First Biennial Conference on Design of Experimental Search & Information Retrieval Systems, Bertinoro, Italy*, 2018.

[7] Jonathan Berant, Andrew Chou, Roy Frostig, and Percy Liang. Semantic parsing on freebase from question-answer pairs. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013, 18-21 October 2013, Grand Hyatt Seattle, Seattle, Washington, USA, A meeting of SIGDAT, a Special Interest Group of the ACL*, pages 1533–1544. ACL, 2013. ISBN 978-1-937284-97-8. URL `http://aclweb.org/anthology/D/D13/D13-1160.pdf`.

[8] Jonathan Berant, Vivek Srikumar, Pei-Chun Chen, Abby Vander Linden, Brittany Harding, Brad Huang, Peter Clark, and Christopher D Manning. Modeling biological processes for reading comprehension. In *Proceedings of the 2014*

Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 1499–1510, 2014.

[9] Kurt Bollacker, Colin Evans, Praveen Paritosh, Tim Sturge, and Jamie Taylor. Freebase: a collaboratively created graph database for structuring human knowledge. In *Proceedings of the 2008 ACM SIGMOD international conference on Management of data*, pages 1247–1250. AcM, 2008.

[10] Antoine Bordes, Xavier Glorot, Jason Weston, and Yoshua Bengio. A semantic matching energy function for learning with multi-relational data. *Machine Learning*, 94(2):233–259, 2014.

[11] Antoine Bordes, Nicolas Usunier, Sumit Chopra, and Jason Weston. Large-scale simple question answering with memory networks. *CoRR*, abs/1506.02075, 2015. URL `http://arxiv.org/abs/1506.02075`.

[12] Eric Brill, Jimmy Lin, Michele Banko, Susan Dumais, Andrew Ng, et al. Data-intensive question answering. In *TREC*, volume 56, page 90, 2001.

[13] Eric Brill, Susan Dumais, and Michele Banko. An analysis of the askmsr question-answering system. In *Proceedings of the ACL-02 conference on Empirical methods in natural language processing-Volume 10*, pages 257–264. Association for Computational Linguistics, 2002.

[14] Claudio Carpineto and Giovanni Romano. A survey of automatic query expansion in information retrieval. *ACM Computing Surveys (CSUR)*, 44(1):1, 2012.

[15] Alessandra Cervone, Chandra Khatri, Rahul Goel, Behnam Hedayatnia, Anu Venkatesh, Dilek Hakkani-Tur, and Raefer Gabriel. Natural language generation at scale: A case study for open domain question answering. *arXiv preprint arXiv:1903.08097*, 2019.

[16] Danqi Chen. *Neural Reading Comprehension and Beyond*. PhD thesis, Stanford University, 2018.

[17] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to Answer Open-Domain Questions. *ArXiv e-prints*, art. arXiv:1704.00051, March 2017.

[18] Junyoung Chung, Caglar Gulcehre, KyungHyun Cho, and Yoshua Bengio. Empirical evaluation of gated recurrent neural networks on sequence modeling. *arXiv preprint arXiv:1412.3555*, 2014.

[19] Marc Claesen and Bart De Moor. Hyperparameter search in machine learning. *arXiv preprint arXiv:1502.02127*, 2015.

[20] Christopher Clark and Matt Gardner. Simple and effective multi-paragraph reading comprehension. *CoRR*, abs/1710.10723, 2017. URL `http://arxiv.org/abs/1710.10723`.

[21] Peter Clark, Isaac Cowhey, Oren Etzioni, Tushar Khot, Ashish Sabharwal, Carissa Schoenick, and Oyvind Tafjord. Think you have solved question answering? try arc, the AI2 reasoning challenge. *CoRR*, abs/1803.05457, 2018. URL `http://arxiv.org/abs/1803.05457`.

[22] Daniel Cohen, Liu Yang, and W Bruce Croft. Wikipassageqa: A benchmark collection for research on non-factoid answer passage retrieval. *arXiv preprint arXiv:1805.03797*, 2018.

[23] Nick Craswell, W Bruce Croft, Maarten de Rijke, Jiafeng Guo, and Bhaskar Mitra. Sigir 2017 workshop on neural information retrieval (neu-ir'17). In *Proceedings of the 40th International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 1431–1432. ACM, 2017.

[24] W Bruce Croft, Donald Metzler, and Trevor Strohman. *Search engines: Information retrieval in practice*, volume 520. Addison-Wesley Reading, 2010.

[25] Mostafa Dehghani, Hosein Azarbonyad, Jaap Kamps, and Maarten de Rijke. Learning to transform, combine, and reason in open-domain question answering. In *Proceedings of the Twelfth ACM International Conference on Web Search and Data Mining*, pages 681–689. ACM, 2019.

[26] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018. URL `http://arxiv.org/abs/1810.04805`.

[27] Bhuwan Dhingra, Hanxiao Liu, William W. Cohen, and Ruslan Salakhutdinov. Gated-attention readers for text comprehension. *CoRR*, abs/1606.01549, 2016. URL `http://arxiv.org/abs/1606.01549`.

[28] Bhuwan Dhingra, Kathryn Mazaitis, and William W. Cohen. Quasar: Datasets for question answering by search and reading. *CoRR*, abs/1707.03904, 2017. URL `http://arxiv.org/abs/1707.03904`.

[29] Ming Ding, Chang Zhou, Qibin Chen, Hongxia Yang, and Jie Tang. Cognitive graph for multi-hop reading comprehension at scale. *arXiv preprint arXiv:1905.05460*, 2019.

[30] Hai Doan-Nguyen and Leila Kosseim. Improving the precision of a closed-domain question-answering system with semantic information. In *Coupling approaches, coupling media and coupling languages for information retrieval*, pages 850–859. LE CENTRE DE HAUTES ETUDES INTERNATIONALES D'INFORMATIQUE DOCUMENTAIRE, 2004.

[31] David Dominguez-Sal and Mihai Surdeanu. A machine learning approach for factoid question answering. *Procesamiento del Lenguaje Natural*, 37, 2006. URL `http://journal.sepln.org/sepln/ojs/ojs/index.php/pln/article/view/2742/1260`.

[32] Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179, 2017. URL `http://arxiv.org/abs/1704.05179`.

[33] Katherine Bennett Ensor and Peter W Glynn. Stochastic optimization via grid search. *Lectures in Applied Mathematics-American Mathematical Society*, 33: 89–100, 1997.

[34] David A. Ferrucci, Eric W. Brown, Jennifer Chu-Carroll, James Fan, David Gondek, Aditya Kalyanpur, Adam Lally, J. William Murdock, Eric Nyberg, John M. Prager, Nico Schlaefer, and Christopher A. Welty. Building watson: An overview of the deepqa project. *AI Magazine*, 31(3):59–79, 2010. URL `http://www.aaai.org/ojs/index.php/aimagazine/article/view/2303`.

[35] David A Ferucci, Aditya A Kalyanpur, James W Murdock IV, Christopher A Welty, and Wlodek W Zadrozny. Using ontological information in open domain type coercion, February 14 2017. US Patent 9,569,724.

[36] Morgan Gallant, Haruna Isah, and Farhana H. Zulkernine. Automated query expansion using high dimensional clustering. *CoRR*, abs/1808.09353, 2018. URL `http://arxiv.org/abs/1808.09353`.

[37] Helena Gómez-Adorno, David Pinto, and Darnes Vilariño. A question answering system for reading comprehension tests. In Jesús Ariel Carrasco-Ochoa, José Francisco Martínez-Trinidad, Joaquín Salas Rodríguez, and Gabriella Sanniti di Baja, editors, *Pattern Recognition*, pages 354–363, Berlin, Heidelberg, 2013. Springer Berlin Heidelberg. ISBN 978-3-642-38989-4.

[38] 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.

[39] Song Han, Huizi Mao, and William J Dally. Deep compression: Compressing deep neural networks with pruning, trained quantization and huffman coding. *arXiv preprint arXiv:1510.00149*, 2015.

[40] Karl Moritz Hermann, Tomás Kociský, Edward Grefenstette, Lasse Espeholt, Will Kay, Mustafa Suleyman, and Phil Blunsom. Teaching machines to read and comprehend. *CoRR*, abs/1506.03340, 2015. URL `http://arxiv.org/abs/1506.03340`.

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

[42] Phu Mon Htut, Samuel R. Bowman, and Kyunghyun Cho. Training a ranking function for open-domain question answering. *CoRR*, abs/1804.04264, 2018. URL `http://arxiv.org/abs/1804.04264`.

[43] Baotian Hu, Zhengdong Lu, Hang Li, and Qingcai Chen. Convolutional neural network architectures for matching natural language sentences. In *Advances in neural information processing systems*, pages 2042–2050, 2014.

[44] Minghao Hu, Yuxing Peng, and Xipeng Qiu. Mnemonic reader for machine comprehension. *CoRR*, abs/1705.02798, 2017. URL `http://arxiv.org/abs/1705.02798`.

[45] 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.

[46] Zhiheng Huang, Wei Xu, and Kai Yu. Bidirectional lstm-crf models for sequence tagging. *arXiv preprint arXiv:1508.01991*, 2015.

[47] K. S. D. Ishwari, A. K. R. R. Aneeze, S. Sudheesan, H. J. D. A. Karunaratne, A. Nugaliyadde, and Y. Mallawarachchi. Advances in natural language question answering: A review. *CoRR*, abs/1904.05276, 2019. URL `http://arxiv.org/abs/1904.05276`.

[48] K Sparck Jones, Steve Walker, and Stephen E. Robertson. A probabilistic model of information retrieval: development and comparative experiments: Part 2. *Information processing & management*, 36(6):809–840, 2000.

[49] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer. Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551, 2017. URL `http://arxiv.org/abs/1705.03551`.

[50] Armand Joulin, Edouard Grave, Piotr Bojanowski, Matthijs Douze, Hérve Jégou, and Tomas Mikolov. Fasttext. zip: Compressing text classification models. *arXiv preprint arXiv:1612.03651*, 2016.

[51] Andisheh Keikha, Faezeh Ensan, and Ebrahim Bagheri. Query expansion using pseudo relevance feedback on wikipedia. *J. Intell. Inf. Syst.*, 50(3):455–478, 2018. doi: 10.1007/s10844-017-0466-3. URL `https://doi.org/10.1007/s10844-017-0466-3`.

[52] Aniruddha Kembhavi, Minjoon Seo, Dustin Schwenk, Jonghyun Choi, Ali Farhadi, and Hannaneh Hajishirzi. Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4999–5007, 2017.

[53] Daniel Khashabi, Snigdha Chaturvedi, Michael Roth, Shyam Upadhyay, and Dan Roth. Looking beyond the surface:a challenge set for reading comprehension over multiple sentences. In *Proceedings of North American Chapter of the Association for Computational Linguistics (NAACL)*, 2018.

[54] Yoon Kim, Yacine Jernite, David Sontag, and Alexander M Rush. Character-aware neural language models. In *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

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

[56] Tomás Kociský, Jonathan Schwarz, Phil Blunsom, Chris Dyer, Karl Moritz Hermann, Gábor Melis, and Edward Grefenstette. The narrativeqa reading comprehension challenge. *CoRR*, abs/1712.07040, 2017. URL `http://arxiv.org/abs/1712.07040`.

[57] Lorena Kodra and Elinda Kajo Meçe. Question answering systems: A review on present developments, challenges and trends. *Department of Computer Engineering, Polytechnic University of Tirana, Albania*, 2017.

[58] Julian Kupiec. Murax: A robust linguistic approach for question answering using an on-line encyclopedia. In *Proceedings of the 16th annual international ACM SIGIR conference on Research and development in information retrieval*, pages 181–190. ACM, 1993.

[59] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Matthew Kelcey, Jacob Devlin, Kenton Lee, Kristina N. Toutanova, Llion Jones, Ming-Wei Chang, Andrew Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Transactions of the Association of Computational Linguistics*, 2019.

[60] Guokun Lai, Qizhe Xie, Hanxiao Liu, Yiming Yang, and Eduard H. Hovy. RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683, 2017. URL `http://arxiv.org/abs/1704.04683`.

[61] Victor Lavrenko and W. Bruce Croft. Relevance-based language models. In W. Bruce Croft, David J. Harper, Donald H. Kraft, and Justin Zobel, editors, *SIGIR 2001: Proceedings of the 24th Annual International ACM SIGIR Conference on Research and Development in Information Retrieval, September 9-13, 2001, New Orleans, Louisiana, USA*, pages 120–127. ACM, 2001. ISBN 1-58113-331-6. doi: 10.1145/383952.383972. URL `https://doi.org/10.1145/383952.383972`.

[62] Yann LeCun, Yoshua Bengio, and Geoffrey Hinton. Deep learning. *nature*, 521 (7553):436, 2015.

[63] Jinhyuk Lee, Seongjun Yun, Hyunjae Kim, Miyoung Ko, and Jaewoo Kang. Ranking Paragraphs for Improving Answer Recall in Open-Domain Question Answering. *ArXiv e-prints*, art. arXiv:1810.00494, September 2018.

[64] 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.

[65] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. *Text Summarization Branches Out*, 2004.

[66] Yankai Lin, Haozhe Ji, Zhiyuan Liu, and Maosong Sun. Denoising distantly supervised open-domain question answering. In *Proceedings of the 56th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1736–1745, 2018.

[67] Aldo Lipani, Mihai Lupu, Allan Hanbury, and Akiko Aizawa. Verboseness fission for bm25 document length normalization. In *Proceedings of the 2015 International Conference on The Theory of Information Retrieval*, ICTIR '15, pages 385–388, New York, NY, USA, 2015. ACM. ISBN 978-1-4503-3833-2. doi: 10.1145/2808194.2809486. URL `http://doi.acm.org/10.1145/2808194.2809486`.

[68] Xiaodong Liu, Yelong Shen, Kevin Duh, and Jianfeng Gao. Stochastic answer networks for machine reading comprehension. *CoRR*, abs/1712.03556, 2017. URL `http://arxiv.org/abs/1712.03556`.

[69] H. P. Luhn. A statistical approach to mechanized encoding and searching of literary information. *IBM Journal of Research and Development*, 1(4):309–317, Oct 1957. ISSN 0018-8646. doi: 10.1147/rd.14.0309.

[70] Yuanhua Lv and ChengXiang Zhai. Adaptive term frequency normalization for bm25. In *Proceedings of the 20th ACM International Conference on Information and Knowledge Management*, CIKM '11, pages 1985–1988, New York, NY, USA, 2011. ACM. ISBN 978-1-4503-0717-8. doi: 10.1145/2063576.2063871. URL `http://doi.acm.org/10.1145/2063576.2063871`.

[71] Harish Tayyar Madabushi and Mark Lee. High accuracy rule-based question classification using question syntax and semantics. In *Proceedings of COLING 2016, the 26th International Conference on Computational Linguistics: Technical Papers*, pages 1220–1230, 2016.

[72] Christopher D. Manning, Prabhakar Raghavan, and Hinrich Schütze. *Introduction to Information Retrieval*. Cambridge University Press, New York, NY, USA, 2008. ISBN 0521865719, 9780521865715.

[73] Melvin Earl Maron and John L Kuhns. On relevance, probabilistic indexing and information retrieval. *Journal of the ACM (JACM)*, 7(3):216–244, 1960.

[74] Tomáš Mikolov, Martin Karafiát, Lukáš Burget, Jan Černockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Eleventh annual conference of the international speech communication association*, 2010.

[75] Sewon Min, Victor Zhong, Luke Zettlemoyer, and Hannaneh Hajishirzi. Multi-hop reading comprehension through question decomposition and rescoring. *arXiv preprint arXiv:1906.02916*, 2019.

[76] Mike Mintz, Steven Bills, Rion Snow, and Dan Jurafsky. Distant supervision for relation extraction without labeled data. In *Proceedings of the Joint Conference of the 47th Annual Meeting of the ACL and the 4th International Joint Conference on Natural Language Processing of the AFNLP: Volume 2-Volume 2*, pages 1003–1011. Association for Computational Linguistics, 2009.

[77] Amit Mishra and Sanjay Kumar Jain. A survey on question answering systems with classification. *Journal of King Saud University-Computer and Information Sciences*, 28(3):345–361, 2016.

[78] Bhaskar Mitra, Nick Craswell, et al. An introduction to neural information retrieval. *Foundations and Trends® in Information Retrieval*, 13(1):1–126, 2018.

[79] Dan Moldovan, Sanda Harabagiu, Marius Pasca, Rada Mihalcea, Roxana Girju, Richard Goodrum, and Vasile Rus. The structure and performance of an open-domain question answering system. In *Proceedings of the 38th annual meeting on association for computational linguistics*, pages 563–570. Association for Computational Linguistics, 2000.

[80] Nasrin Mostafazadeh, Michael Roth, Annie Louis, Nathanael Chambers, and James Allen. LSDSem 2017 shared task: The story cloze test. In *Proceedings of the 2nd Workshop on Linking Models of Lexical, Sentential and Discourse-level Semantics*, pages 46–51, Valencia, Spain, April 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-0906. URL `https://www.aclweb.org/anthology/W17-0906`.

[81] Karthik Narasimhan and Regina Barzilay. Machine comprehension with discourse relations. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 1253–1262, 2015.

[82] Hwee Tou Ng, Leonghwee Teo, and Jennifer Lai-Pheng Kwan. A machine learning approach to answering questions for reading comprehension tests. In Hinrich Schütze and Keh-Yih Su, editors, *Joint SIGDAT Conference on Empirical Methods in Natural Language Processing and Very Large Corpora, EMNLP 2000, Hong Kong, October 7-8, 2000*. Association for Computational Linguistics, 2000. URL `https://aclanthology.info/papers/W00-1316/w00-1316`.

[83] Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268, 2016. URL `http://arxiv.org/abs/1611.09268`.

[84] Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. Retrieve-and-read: Multi-task learning of information retrieval and reading comprehension. *CoRR*, abs/1808.10628, 2018. URL `http://arxiv.org/abs/1808.10628`.

[85] Jiaul H Paik. A novel tf-idf weighting scheme for effective ranking. In *Proceedings of the 36th international ACM SIGIR conference on Research and development in information retrieval*, pages 343–352. ACM, 2013.

[86] 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.

[87] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, and Xueqi Cheng. A deep investigation of deep ir models. *arXiv preprint arXiv:1707.07700*, 2017.

[88] Liang Pang, Yanyan Lan, Jiafeng Guo, Jun Xu, Lixin Su, and Xueqi Cheng. HAS-QA: hierarchical answer spans model for open-domain question answering. *CoRR*, abs/1901.03866, 2019. URL `http://arxiv.org/abs/1901.03866`.

[89] Razvan Pascanu, Tomas Mikolov, and Yoshua Bengio. On the difficulty of training recurrent neural networks. In *International conference on machine learning*, pages 1310–1318, 2013.

[90] Panupong Pasupat and Percy Liang. Compositional semantic parsing on semi-structured tables. *CoRR*, abs/1508.00305, 2015. URL `http://arxiv.org/abs/1508.00305`.

[91] Jeffrey Pennington, Richard Socher, and Christopher Manning. Glove: Global vectors for word representation. In *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, pages 1532–1543, 2014.

[92] George-Sebastian Pirtoaca, Traian Rebedea, and Stefan Ruseti. Improving retrieval-based question answering with deep inference models. *CoRR*, abs/1812.02971, 2018. URL `http://arxiv.org/abs/1812.02971`.

[93] Martin F Porter. Snowball: A language for stemming algorithms, 2001.

[94] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250, 2016. URL `http://arxiv.org/abs/1606.05250`.

[95] Pranav Rajpurkar, Robin Jia, and Percy Liang. Know what you don't know: Unanswerable questions for squad. *CoRR*, abs/1806.03822, 2018. URL `http://arxiv.org/abs/1806.03822`.

[96] Siva Reddy, Danqi Chen, and Christopher D. Manning. Coqa: A conversational question answering challenge. *CoRR*, abs/1808.07042, 2018. URL `http://arxiv.org/abs/1808.07042`.

[97] Matthew Richardson, Christopher J.C. Burges, and Erin Renshaw. MCTest: A challenge dataset for the open-domain machine comprehension of text. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 193–203, Seattle, Washington, USA, October 2013.

Association for Computational Linguistics. URL `https://www.aclweb.org/anthology/D13-1020`.

[98] Ellen Riloff and Michael Thelen. Rule-based question answering system for reading comprehension tests. *ANLP/NAACL-2000 Workshop on Reading Comprehension Tests as Evaluation for Computer-Based Language Understanding Systems*, 6, 05 2000. doi: 10.3115/1117595.1117598.

[99] Ellen Riloff and Michael Thelen. A rule-based question answering system for reading comprehension tests. In *Proceedings of the 2000 ANLP/NAACL Workshop on Reading comprehension tests as evaluation for computer-based language understanding sytems-Volume 6*, pages 13–19. Association for Computational Linguistics, 2000.

[100] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009.

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

[102] Mrinmaya Sachan, Kumar Dubey, Eric Xing, and Matthew Richardson. Learning answer-entailing structures for machine comprehension. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, volume 1, pages 239–249, 2015.

[103] Hassan Saneifar, Stéphane Bonniol, Pascal Poncelet, and Mathieu Roche. Enhancing passage retrieval in log files by query expansion based on explicit and pseudo relevance feedback. *Computers in Industry*, 65(6):937–951, 2014. doi: 10.1016/j.compind.2014.02.010. URL `https://doi.org/10.1016/j.compind.2014.02.010`.

[104] Min Joon Seo, Aniruddha Kembhavi, Ali Farhadi, and Hannaneh Hajishirzi. Bidirectional attention flow for machine comprehension. *CoRR*, abs/1611.01603, 2016. URL `http://arxiv.org/abs/1611.01603`.

[105] 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.

[106] Zhendong Shi, Jacky Keung, and Qinbao Song. An empirical study of bm25 and bm25f based feature location techniques. In *Proceedings of the International Workshop on Innovative Software Development Methodologies and Practices*, pages 106–114. ACM, 2014.

[107] Robert F Simmons, Sheldon Klein, and Keren McConlogue. Indexing and dependency logic for answering english questions. *American Documentation*, 15 (3):196–204, 1964.

[108] Amit Singhal et al. Modern information retrieval: A brief overview. *IEEE Data Eng. Bull.*, 24(4):35–43, 2001.

[109] Katarina Stanoevska-Slabeva, Alexis Hombrecher, Siegfried Handschuh, and Beat F Schmid. Efficient information retrieval: Tools for knowledge management. In *PAKM*, volume 98, page 23. Citeseer, 1998.

[110] Fu Sun, Linyang Li, Xipeng Qiu, and Yang Liu. U-net: Machine reading comprehension with unanswerable questions. *arXiv preprint arXiv:1810.06638*, 2018.

[111] Haitian Sun, Bhuwan Dhingra, Manzil Zaheer, Kathryn Mazaitis, Ruslan Salakhutdinov, and William W. Cohen. Open domain question answering using early fusion of knowledge bases and text. *CoRR*, abs/1809.00782, 2018. URL `http://arxiv.org/abs/1809.00782`.

[112] Haitian Sun, Tania Bedrax-Weiss, and William W. Cohen. Pullnet: Open domain question answering with iterative retrieval on knowledge bases and text. *CoRR*, abs/1904.09537, 2019. URL `http://arxiv.org/abs/1904.09537`.

[113] Kai Sun, Dian Yu, Dong Yu, and Claire Cardie. Improving machine reading comprehension with general reading strategies. *CoRR*, abs/1810.13441, 2018. URL `http://arxiv.org/abs/1810.13441`.

[114] Alon Talmor and Jonathan Berant. The web as a knowledge-base for answering complex questions. *CoRR*, abs/1803.06643, 2018. URL `http://arxiv.org/abs/1803.06643`.

[115] Makarand Tapaswi, Yukun Zhu, Rainer Stiefelhagen, Antonio Torralba, Raquel Urtasun, and Sanja Fidler. Movieqa: Understanding stories in movies through question-answering. *CoRR*, abs/1512.02902, 2015. URL `http://arxiv.org/abs/1512.02902`.

[116] Yi Tay, Luu Anh Tuan, Siu Cheung Hui, and Jian Su. Densely connected attention propagation for reading comprehension. *CoRR*, abs/1811.04210, 2018. URL `http://arxiv.org/abs/1811.04210`.

[117] Michael Taylor, Hugo Zaragoza, Nick Craswell, Stephen Robertson, and Chris Burges. Optimisation methods for ranking functions with multiple parameters. In *Proceedings of the 15th ACM International Conference on Information and Knowledge Management*, CIKM '06, pages 585–593, New York, NY, USA, 2006. ACM. ISBN 1-59593-433-2. doi: 10.1145/1183614.1183698. URL `http://doi.acm.org/10.1145/1183614.1183698`.

[118] Adam Trischler, Tong Wang, Xingdi Yuan, Justin Harris, Alessandro Sordoni, Philip Bachman, and Kaheer Suleman. Newsqa: A machine comprehension dataset. *CoRR*, abs/1611.09830, 2016. URL `http://arxiv.org/abs/1611.09830`.

[119] Andrew Trotman, Antti Puurula, and Blake Burgess. Improvements to bm25 and language models examined. In *Proceedings of the 2014 Australasian Document Computing Symposium*, ADCS '14, pages 58:58–58:65, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-3000-8. doi: 10.1145/2682862.2682863. URL `http://doi.acm.org/10.1145/2682862.2682863`.

[120] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82. Citeseer, 1999.

[121] 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 *Thirtieth AAAI Conference on Artificial Intelligence*, 2016.

[122] Hai Wang, Mohit Bansal, Kevin Gimpel, and David McAllester. Machine comprehension with syntax, frames, and semantics. In *Proceedings of the 53rd Annual Meeting of the Association for Computational Linguistics and the 7th International Joint Conference on Natural Language Processing (Volume 2: Short Papers)*, volume 2, pages 700–706, 2015.

[123] Jun Wang, Yu-Gang Jiang, and Shih-Fu Chang. Label diagnosis through self tuning for web image search. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1390–1397. IEEE, 2009.

[124] Mengqiu Wang, Noah A Smith, and Teruko Mitamura. What is the jeopardy model? a quasi-synchronous grammar for qa. In *Proceedings of the 2007 Joint Conference on Empirical Methods in Natural Language Processing and Computational Natural Language Learning (EMNLP-CoNLL)*, 2007.

[125] Shuohang Wang, Mo Yu, Xiaoxiao Guo, Zhiguo Wang, Tim Klinger, Wei Zhang, Shiyu Chang, Gerald Tesauro, Bowen Zhou, and Jing Jiang. R^3: Reinforced reader-ranker for open-domain question answering. *CoRR*, abs/1709.00023, 2017. URL `http://arxiv.org/abs/1709.00023`.

[126] Shuohang Wang, Mo Yu, Jing Jiang, Wei Zhang, Xiaoxiao Guo, Shiyu Chang, Zhiguo Wang, Tim Klinger, Gerald Tesauro, and Murray Campbell. Evidence aggregation for answer re-ranking in open-domain question answering. *CoRR*, abs/1711.05116, 2017. URL `http://arxiv.org/abs/1711.05116`.

[127] Wei Wang, Ming Yan, and Chen Wu. Multi-granularity hierarchical attention fusion networks for reading comprehension and question answering. *arXiv preprint arXiv:1811.11934*, 2018.

[128] Wenhui Wang, Nan Yang, Furu Wei, Baobao Chang, and Ming Zhou. Gated self-matching networks for reading comprehension and question answering. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 189–198, 2017.

[129] Johannes Welbl, Pontus Stenetorp, and Sebastian Riedel. Constructing datasets for multi-hop reading comprehension across documents. *CoRR*, abs/1710.06481, 2017. URL `http://arxiv.org/abs/1710.06481`.

[130] Robert West, Evgeniy Gabrilovich, Kevin Murphy, Shaohua Sun, Rahul Gupta, and Dekang Lin. Knowledge base completion via search-based question answering. In *Proceedings of the 23rd International Conference on World Wide Web*, WWW '14, pages 515–526, New York, NY, USA, 2014. ACM. ISBN 978-1-4503-2744-2. doi: 10.1145/2566486.2568032. URL `http://doi.acm.org/10.1145/2566486.2568032`.

[131] S. K. Michael Wong and YY Yao. A generalized binary probabilistic independence model. *Journal of the American Society for Information Science*, 41(5): 324–329, 1990.

[132] Semih Yagcioglu, Aykut Erdem, Erkut Erdem, and Nazli Ikizler-Cinbis. Recipeqa: A challenge dataset for multimodal comprehension of cooking recipes. *CoRR*, abs/1809.00812, 2018. URL `http://arxiv.org/abs/1809.00812`.

[133] 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*, SIGIR '17, pages 1253–1256, New York, NY, USA, 2017. ACM. ISBN 978-1-4503-5022-8. doi: 10.1145/3077136.3080721. URL `http://doi.acm.org/10.1145/3077136.3080721`.

[134] Peilin Yang, Hui Fang, and Jimmy Lin. Anserini: Reproducible ranking baselines using lucene. *J. Data and Information Quality*, 10(4):16:1–16:20, October 2018. ISSN 1936-1955. doi: 10.1145/3239571. URL `http://doi.acm.org/10.1145/3239571`.

[135] 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. *CoRR*, abs/1902.01718, 2019. URL `http://arxiv.org/abs/1902.01718`.

[136] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600, 2018. URL `http://arxiv.org/abs/1809.09600`.

[137] Xuchen Yao, Jonathan Berant, and Benjamin Van Durme. Freebase qa: Information extraction or semantic parsing? In *ACL 2014*, 2014.

[138] Tom Young, Devamanyu Hazarika, Soujanya Poria, and Erik Cambria. Recent trends in deep learning based natural language processing. *CoRR*, abs/1708.02709, 2017. URL `http://arxiv.org/abs/1708.02709`.

[139] Adams Wei Yu, David Dohan, Minh-Thang Luong, Rui Zhao, Kai Chen, Mohammad Norouzi, and Quoc V. Le. Qanet: Combining local convolution with global self-attention for reading comprehension. *CoRR*, abs/1804.09541, 2018. URL `http://arxiv.org/abs/1804.09541`.

[140] Ruifan Yu, Yuhao Xie, and Jimmy Lin. H2oloo at trec 2018: Cross-collection relevance transfer for the common core track. In *TREC*, 2018.

[141] Shipeng Yu, Deng Cai, Ji-Rong Wen, and Wei-Ying Ma. Improving pseudo-relevance feedback in web information retrieval using web page segmentation. In *Proceedings of the 12th international conference on World Wide Web*, pages 11–18. ACM, 2003.

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

[143] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014.

[144] Chengxiang Zhai and John Lafferty. A study of smoothing methods for language models applied to ad hoc information retrieval. *SIGIR Forum*, 51(2): 268–276, August 2017. ISSN 0163-5840. doi: 10.1145/3130348.3130377. URL `http://doi.acm.org/10.1145/3130348.3130377`.

[145] Ye Zhang, Md Mustafizur Rahman, Alex Braylan, Brandon Dang, Heng-Lu Chang, Henna Kim, Quinten McNamara, Aaron Angert, Edward Banner, Vivek Khetan, et al. Neural information retrieval: A literature review. *arXiv preprint arXiv:1611.06792*, 2016.

Appendix A

Hyperparameter Tuning

This is the overview of the values to be tested for hyperparameter tuning of traditional IR models **BM25** and **QL** as well as the relevance model **RM3**.

Model	Hyperparameter	Values to be tested
BM25	k_1	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0, 1.2, 1.4, 1.6, 1.8, 2.0}
	b	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 1.0}
QL	λ	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}
RM3	$fbDocs$	{5, 10, 15, 20, 25, 30}
	$fbTerms$	{50, 100, 150, 200, 250, 300}
	λ	{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9}

Table A.1: Values to be tested by grid search for hyperparameter tuning of traditional IR models used in our work. All hyperparameters are tuned on the training set of each dataset in the open-domain setting.