Eidgenössische Technische Hochschule Zürich
Swiss Federal Institute of Technology Zurich

Distributed Computing

Towards Multi-Hop Open-Domain Question Answering by Dense Retrieval

Bachelor's Thesis

Jie Ji

jiejijie@student.ethz.ch

Distributed Computing Group
Computer Engineering and Networks Laboratory
ETH Zürich

Supervisors:
Zhao Meng
Prof. Dr. Roger Wattenhofer

August 11, 2022

Acknowledgements

First and foremost, I would like to express my deepest appreciation to my supervisor Zhao Meng for guiding me through this entire journey, always being kind, patient, and supportive when I encountered all kinds of difficulties. Without his assistance and creative ideas, this thesis would have never been accomplished.

I would also like to thank Prof.Dr. Roger Wattenhofer and the Distributed Computing Group for providing me the opportunity and also technical support to complete this work.

Further, I am very grateful to my good friend Jinfan Chen for always being by my side, helping me with the details of this thesis.

Lastly, I would like to thank my cat Aki for all the entertainment and emotional support.

Abstract

One of the ultimate goals of AI is to design a technology that enables intelligent systems to "think" independently like humans. The task of Open-Domain Question Answering (OpenQA) [1] is exactly such an example: Given a question in natural language, the system should return an exact answer to the question. This can be realized through a two-stage process: A retriever will first output a certain amount of relevant passages retrieved from an extensive collection of documents. Then a reader will identify and extract the final answer from the relevant passages. However, the retrieval step is a bottleneck when facing a large-scale resource of contexts and it becomes a particularly challenging task in case of OpenQA. Furthermore, recent success on simple OpenQA tasks has shifted the focus to more complex settings: the Multi-hop QA. Although this has been an active research area, most existing works focused on maximizing model retrieval accuracy with a fixed number of reasoning steps. In real-world scenarios, however, a system will not be informed of the number of reasoning steps required to answer a question. We consider this as a problem that cannot be ignored and intend to develop a uniform end-to-end trainable QA system using the dense retrieval setting. After various attempts, we conclude that such a system is hard to realize by training a dense retrieval model on combined datasets. Nevertheless, we provide an alternative solution based on our success in question hop classification, which can be further adapted for improvements but is generally more complex.

Contents

Chapter 1

Introduction

Question answering (QA) is a computer science topic within the domains of information retrieval and natural language processing (NLP) that focuses on the development of systems, which can provide precise answers in response to questions submitted by humans in natural language.

Compared to a search engine, the QA system tries to explicitly deliver the ultimate answer to a query rather than presenting a list of relevant snippets or hyperlinks, hence providing more user-friendliness and efficiency. Since human knowledge is commonly stored in vast text collections, QA is essential for both human and intelligent systems to access such knowledge. Numerous web search engines nowadays, such as Google and Bing, have also incorporated QA approaches into their search functionality.

With these techniques, search engines may now provide exact responses to certain types of queries. According to the type of information source, QA tasks may be roughly divided into two sectors: Knowledge Base (KB)-QA and Textual QA. Compared to KB-QA, which extracts answers from a predefined structured KB, Textual QA mines answers from unstructured text materials, such as Wikipedia, news articles and science books, etc. In recent years, Textual QA is mainly studied under the task setting of Open-domain Question Answering (OpenQA) [1], which we will be focusing on in this thesis.

1.1 OpenQA

OpenQA aims to answer a given question without any specified context. While the traditional architecture of OpenQA systems is often complex and comprises several components, a much simplified two-stage framework has been developed for the modern OpenQA: 1) A context retriever retrieves the most relevant passages to a question from a large corpus. 2) A machine reader examines the retrieved context and identifies the position of the correct answer. In the recent

decade, deep learning techniques have been successfully applied to OpenQA, which enables the system to be end-to-end trainable. In the following subsections, we would like to explain in detail the retriever and reader stages.

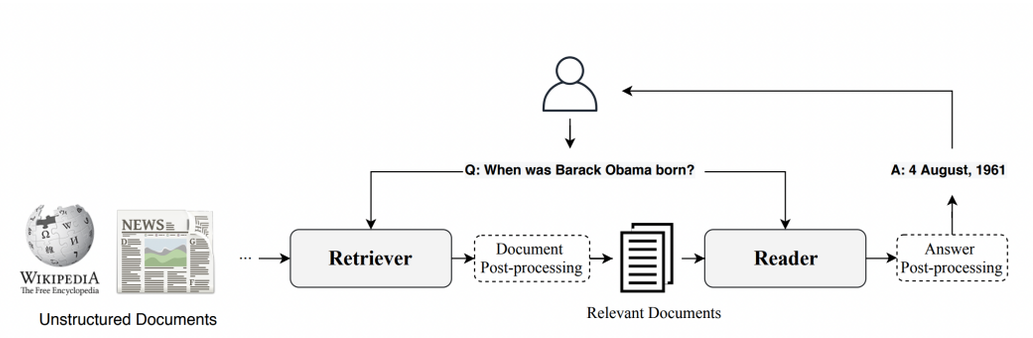

Figure 1.1: An illustration of "Retriever-Reader" architecture of OpenQA systems

1.1.1 Retriever

Current methods of Retriever may be broadly categorized into three groups: Sparse Retriever, Dense Retriever, and Iterative Retriever.[2] Sparse Retriever refers to systems that search for relevant documents using traditional Information Retrieval techniques, such as BM25 [3] and TF-IDF, which will be detailed in a later chapter. However, sparse retriever systems often suffer from "term-mismatch", i.e. synonyms or semantically similar words, consisting of completely different tokens, will not be treated as similar, which results in low retrieval effectiveness. Alongside the success of deep learning in recent years, neural retriever models such as dense retrievers have been developed to perform better encoding for latent semantics, resulting in vastly increased retrieval efficiency and accuracy. An example of the dense retriever we would like to consider in this thesis is the Dense Passage Retriever (DPR) [4], which is a representation-based retriever model consisting of two independent encoders like BERT [5]. However, the two retriever models mentioned above are not capable of solving complex questions like those requiring multi-hop reasoning. Iterative Retriever, also called Multi-step Retriever, aims to tackle this problem while searching for the relevant documents from a large collection in multiple steps given a question. An example related to our work would be the Multi-hop Dense Retriever (MDR) [6], which was designed based on the architecture of DPR and introduced a new strategy to retrieve contexts in multiple steps.

1.1.2 Reader

Apart from the retriever, a reader is also required to build an end-to-end QA system. The reader is commonly implemented in two different types: Extractive readers and generative readers. Extractive readers presume that solutions can be found in the context and attempt to anticipate the start and end tokens based on the context [4]. In contrast, generative readers are not constrained by the input context and may produce responses freely token by token utilizing the complete vocabulary in an autoregressive way. [7] Although generative readers have shown great performance on single-hop OpenQA tasks, extractive readers slightly outperform them on multi-hop QA datasets.

1.2 Multi-hop QA

Recent accomplishments on simple QA tasks have turned the research emphasis to more complicated contexts. Multi-hop QA has been extensively studied in recent years. The notion of "multi-hop question" refers to questions requiring multiple reasoning steps to retrieve the contexts containing the correct answer.

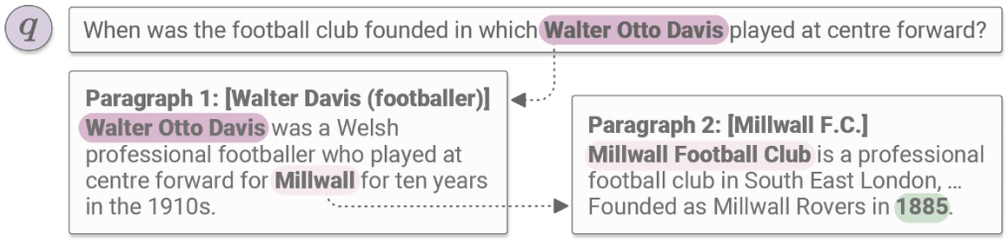

Figure 1.2: An example of open-domain multi-hop question from HotpotQA

It can be surprising that something that seems relatively trivial may be completely confusing for modern AI systems. The ability to respond to multi-hop queries and execute multi-step reasoning may greatly enhance the usefulness of NLP systems. It also has various practical applications in the real world: Current web search engines often need multi-hop reasoning to locate relevant pages in response to queries. Additionally, interactions between people and bots may be more natural and instructive. By using multi-hop reasoning models, user satisfaction may be significantly enhanced. Therefore we argue that solving the task of Multi-hop QA is a crucial challenge, especially in the domain of OpenQA.

1.3 Task Description

Despite the fact that Multi-hop QA has been an active research area in NLP in recent years, most of the existing works only focus on improving the retrieval effectiveness while training and on datasets with a fixed number of reasoning steps. Well-known models such as ORQA [8], DPR [4], PathRetriever [9] and SPARTA [10] mainly focused on single-hop datasets, others such as GoldEn Retriever [11], Graph Recurrent Retriever [9] and DrKIT [12] investigated 2-hop datasets. However, in practical usage, a QA system will normally not be informed of how many reasoning steps it should take to answer a given question. Therefore in this thesis, we aim to tackle this problem by building an uniform retriever model trained on both single- and multi-hop datasets. We decide to build our model based on the dense retrieval method and make modifications to the model MDR, which already achieved an overall good performance in the multi-hop QA setting. Furthermore, we aim to identify the unknown number of question hops by training a classifier based on the output context embeddings generated from our retriever model.

Model	**NQ (1-hop)**
BM25+BERT	26.5
ORQA	33.3
SPARTA	36.8
PathRetriever	32.6
DPR	**41.5**

Table 1.1: Comparison of end-to-end QA(Exact Match) Accuracy of different models evaluated on NQ.

Model	**HotpotQA (2-hop)**
GoldEn Retriever	26.5
Graph Recurrent Retriever	60.0
DrKIT	41.5
Transformer-XH	51.6
MDR	**62.3**

Table 1.2: Comparison of end-to-end QA(Exact Match) Accuracy of different models evaluated on HotpotQA.

Chapter 2

Related Work

2.1 Encoder Models

The retriever model we employ in this thesis relies on third-party libraries for encoder code implementations. Thus we would like to include a brief introduction of the encoder models related to our work in the following subsections.

2.1.1 BERT

BERT (Bidirectional Encoder Representation from Transformers) [5] is a transformer-based model, achieving state-of-the-art results in a wide variety of NLP tasks, including Question Answering, Sentiment Analysis, Natural Language Inference, and others. The key technical innovation of BERT is applying the bidirectional training of Transformer, an attention mechanism that learns contextual relations between words in a text, to language modeling. Unlike general transformers consisting of an encoder and a decoder network, BERT only uses encoders and stacks them on top of each other.

Training a BERT model is usually done in two phases: First, we pretrain BERT to understand language. The second phase is fine-tuning it in a supervised fashion using a relatively small amount of labeled training data. This process enables BERT to learn specific tasks depending on the problem we want to solve. During Pretraining Phase, Bert learns language by training on two unsupervised tasks simultaneously: Masked Language Modeling (MLM) and Next Sentence Prediction (NSP). For MLM, BERT converts a sentence into tokens and uses the token representation as input and output. Then random tokens will be masked during training and the objective is to learn a function that can predict the correct identities of the masked tokens. In the case of NSP, BERT takes in two sentences and determines if the second sentence actually follows the first. This helps BERT understand context across different sentences themselves. Training MLM and NSP objectives simultaneously allow BERT to obtain a good understanding of language.

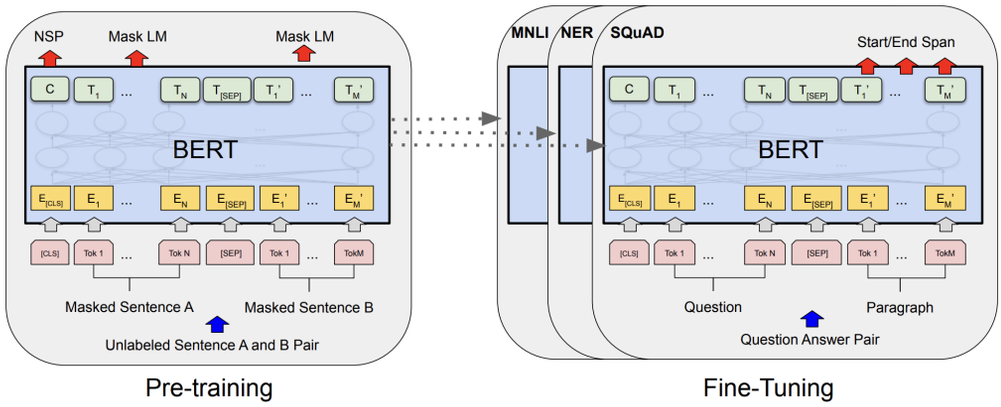

Figure 2.1: An overview of the pre-training and fine-tuning procedures for BERT

2.1.2 RoBERTa

RoBERTa (Robustly Optimized BERT Pretraining Approach) [13] is a retraining of BERT with improved training methodology, 1000% more data and compute power. As stated above, BERT relies on randomly masking and predicting tokens. The masking is only performed once during data preprocessing. RoBERTa claims to improve MLM with a dynamic masking strategy, where a masking pattern will be generated every time when a sequence is fed to the model. Additionally, experiments and results from RoBERTa have shown that removing the NSP task will slightly improve the downstream task performance. The training process was also shown to be more effective with larger batch-training sizes and some modified hyperparameters. Since RoBERTa has shown an overall better performance, also on the dataset SQuAD Open [14] we include in our experiments, we decide to maintain the encoder training in MDR with RoBERTa.

Model	SQuAD 2.0 (SQuAD Open)	
	EM	F1
$BERT_{LARGE}$	79.0	81.8
RoBERTa	**86.5**	**89.4**

Table 2.1: A comparison of results on SQuAD Open using BERT and RoBERTa from [13]

2.2 OpenQA Models

2.2.1 DPR

The retrieval step in the first stage is usually implemented with sparse vector retrievers such as TF-IDF or BM25. In contrast, it would be difficult for these term-based systems to retrieve contexts for questions containing latent semantics. Synonyms or semantically similar words, consisting of completely different tokens, will not be treated as identical. In this case, a dense retrieval system outperforms a normal term-based system in matching synonyms and retrieving the correct contexts. Therefore we would like to introduce the findings in DPR.

DPR [4] is implemented based on a two-stage framework: 1) A context retriever retrieves the most relevant passages to a question from a large corpus. 2) A machine reader examines the retrieved context and identifies the position of the correct answer. Research in DPR mainly focused on improving the retrieval step in the first stage.

For the retriever, DPR uses a dense encoder $E_P(\cdot)$, which maps all text passages p in d-dimensional real-valued vectors. Given a batch of input questions q, DPR applies a different encoder $E_Q(\cdot)$, which also maps the questions to d-dimensional vectors. The similarity of the input question and each passage in the corpus is defined as

$$sim(q, p) = E_Q(q)^{\mathsf{T}} E_P(p) \tag{2.1}$$

Passages with the top-k highest similarity score will be retrieved as results and will be passed to the next stage. DPR relies on third-party libraries for encoder code implementations. In this thesis, we would like to test DPR using two independent BERT networks (base, uncased) and take the representation at the [CLS] token as the output vector, resulting in d = 768.

During inference, the similarity match could normally take hours to days due to the large-scale corpus passages. Therefore we employ FAISS [15] offline for efficient similarity search and dense vector clustering. This helps us to significantly speed up the inference process.

DPR aims to train the encoders by learning a better embedding function, that generates a vector space in which relevant pairs of questions and passages have a lower distance (i.e., higher similarity) than irrelevant pairs. During training, DPR carefully designs the ways to select negative samples to a question. It uses an in-batch-negative training method: Assume that we have n questions q_i in a batch, each associated with a relevant (positive) passage p_i^+. Let $\mathbf{Q}$ and $\mathbf{P}$ be the $(n \times d)$ matrix of question and passage embeddings in batch. $\mathbf{S} = \mathbf{Q}\mathbf{P}^{\mathbf{T}}$ is

then a $(n \times n)$ matrix of similarity scores, whereas only the diagonal elements in this similarity matrix correspond to the similarity score of the question q_i and its positive matching passage p_i^+. Formally speaking, any (q_i, p_j) pair is a positive example when $i = j$, and negative otherwise.

$$\begin{bmatrix} — & q_1 & — \\ — & q_2 & — \\ & \vdots & \\ — & q_n & — \end{bmatrix} \times \begin{bmatrix} | & | & & | \\ p_1^+ & p_2^+ & \cdots & p_n^+ \\ | & | & & | \end{bmatrix} = \begin{bmatrix} s_{11} & s_{12} & \cdots & s_{1n} \\ s_{21} & s_{22} & \cdots & s_{2n} \\ \vdots & \vdots & \ddots & \vdots \\ s_{n1} & s_{n2} & \cdots & s_{nn} \end{bmatrix}$$

For each question, we optimize the loss function as the negative log likelihood of its positive passage:

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n-1}^-) = -log \frac{e^{sim(q_i, p_i^+)}}{e^{sim(q_i, p_i^+)} + \sum_{j=1}^{n-1} e^{sim(q_i, p_{i,j}^-)}} \tag{2.2}$$

This strategy helps us to reuse computation and reduce computation complexity while achieving great performance. DPR suggests that its best model uses the in-batch negatives and one additional hard negative passage: A passage retrieved by BM25 for each question, which functions as a false adversarial passage. Compared to the non-trainable traditional sparse retrievers, DPR can outperform it by fine-tuning the question and passage encoders on existing question-passage pairs from different datasets.

However, the DPR model is often trained and evaluated on exclusively single-hop questions, whereas in practice, we would possibly encounter questions containing multiple hops. Therefore we would like to introduce further findings from MDR in the next subsection.

2.2.2 MDR

Since DPR has only been trained to answer simple questions requiring a single piece of text evidence, MDR [6] suggests a new strategy base on the in-batch-negative training method in DPR to answer multi-hop questions. Since the process of answering such questions may be sequential, single-shot retrieval methods are inadequate. Instead, iterative procedures are required to extract new information recursively at each step.

Akin to DPR, the retrieval in MDR is implemented as the maximum inner product search over the dense representations. Additionally, a new query representation is constructed based on previous results at each retrieval step. At time

step $t > 1$, MDR simply concatenates the question and the retrieved passages as the new question. Then the new question will be fed into the encoder to generate embeddings for the next retrieval step. Instead of using a bi-encoder architecture with independent query and context encoders as in DPR, the results in MDR have shown that a shared encoder for both query and context embedding generation would lead to better performance. Therefore a shared RoBERTa-base [13] encoder is used during training and inference.

During training, MDR also applies the in-batch-negative method with one additional hard negative passage for each question. Instead of using BM25, MDR obtains hard negatives from TF-IDF retrieved passages. For inference, the entire corpus will first be encoded and then indexed by FAISS. The inference is based only on the dense passage index and the query representations. Explicit graph creation with hyperlinks or entity linking is not required.

Another key aspect we can conclude from the experiments in MDR is that the order of the passage chain is important for efficient retriever training. The used dataset, HotpotQA [16], contains two different types of 2-hop questions: 1) bridge questions in which a certain intermediate passage must be retrieved in order to reach the final passage containing the answer, and 2) comparison questions in which two entities will be concurrently referenced and compared. Results show it is necessary to obtain the correct retrieving order for bridge questions. Nevertheless, there is still a gap in retrieval performance between bridge and comparison questions. Since in comparison questions, both entities needed for retrieval are present, it is generally easier to achieve a higher retrieval accuracy.

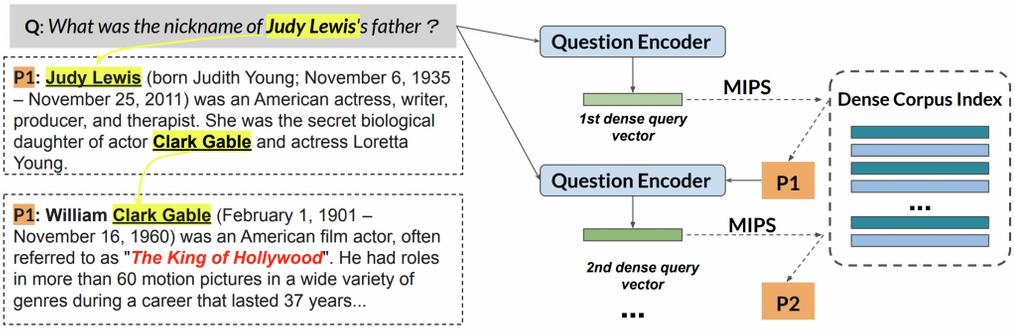

Figure 2.2: An overview of the multi-hop dense retrieval approach

2.3 Datasets

In this section, we would like to introduce some common datasets used in experiments in OpenQA, which we will be focusing on later in our experiments as well. The statistics of the datasets we use in our experiments will be presented in the Methodology chapter since we will modify and create a new dataset consisting of the following datasets.

2.3.1 Natural Questions

Natural Questions (NQ) [17] is a dataset designed for end-to-end question answering. It consists of real anonymized, aggregated questions mined from the Google search engine. The answers were spans in Wikipedia articles identified by annotators. In our experiments we used preprocessed NQ from DPR, containing the positive context, which refers to the Wikipedia article containing the answer, and hard negative contexts generated by BM25 that do not contain the answer. The hard negative contexts were paired with similarity scores and listed in descending order.

2.3.2 SQuAD Open

Stanford Question Answering Dataset Open (SQuAD Open) [14] is a well-known benchmark dataset for reading comprehension. Annotators were asked to write questions that can be answered after reading a given Wikipedia article. Although SQuAD Open was popular and often used in previous OpenQA research, it is not ideal since most questions are relatively trivial and the data was collected from only 500+ Wikipedia articles. The distribution is therefore fairly biased. For a fair comparison to previous work, we still include this dataset in our experiments since it is contained in BeerQA.

2.3.3 HotpotQA

HotpotQA [16] is a question-answering dataset featuring 2-hop questions, which means that for each question, two passages must be retrieved to ensure enough information for the answer. HotpotQA uses documents from Wikipedia and also provides ground truth support passages for each question. This enables us to also evaluate the intermediate retrieval performance.

2.3.4 BeerQA

BeerQA [18] has been designed for both single- and multihop training for OpenQA systems. The training and the development dataset are a combination of SQuAD

Open and HotpotQA. Due to an updated version of Wikipedia and consequently title mismatch, 22,328 examples from SQuAD Open and 18,649 from HotpotQA will be discarded. Furthermore, 530 newly annotated questions requiring at least 3 retrieval steps have been added as the new test set. This can be used to evaluate the generalization capabilities of QA models with respect to unanticipated events.

Chapter 3

Methodology

3.1 Pre-processing

3.1.1 Wikipedia Pre-processing

For passage retrieval during inference, we use the English Wikipedia dump from Aug.1, 2020, as the source documents. Since original Wikipedia pages may include useless links and other distracting information such as tables, info-boxes, and lists, we use the code from WikiExtractor[1] to extract clean text from the entire Wikipedia corpus. Eventually, this results in 15,241,713 passages, each also prepended with the title of the original Wikipedia article.

3.1.2 Generation of hard negatives using TF-IDF

Unlike the dataset HotpotQA, hard negative contexts were not provided in BeerQA. Since we would like to make modifications to MDR, it is necessary to obtain hard negatives using TF-IDF, the same method of hard negatives generation as in MDR.

TF-IDF is an abbreviation for term frequency-inverse document frequency, which is a numerical statistic that is meant to indicate a word's importance to a document in a corpus. The term TF stands for the relative occurrence of term t within document d:

$$TF(t,d) = \frac{f_{t,d}}{\sum_{t' \in d} f_{t',d}} \tag{3.1}$$

And IDF is the inverse of the document frequency which measures the informativeness of term t:

$$IDF(t,D) = log \frac{N}{|\{d \in D : t \in d\}|} \tag{3.2}$$

[1] https://github.com/attardi/wikiextractor

N stands for the total number of documents in the corpus N = $|D|$. On the other hand $|\{d \in D : t \in d\}|$ refers to the number of documents in which the term t appears. It is also common to adjust the denominator to $1 + |\{d \in D : t \in d\}|$ to avoid division-by-zero. Due to the large scale of the Wikipedia corpus, we use the code provided in [9] to efficiently generate hard negatives by TF-IDF.

3.1.3 Dataset construction

In order to maintain the overall training architecture from MDR, we adjusted BeerQA to the same structure as HotpotQA used in MDR. As mentioned in the previous chapter, the retrieving order is important for multi-hop questions, especially for bridge questions. Since the bridge title of the intermediate linking ground-truth passage is not given in BeerQA, we try to match and identify bridge titles from the same question in HotpotQA. Due to an updated version of the Wikipedia corpus, we discarded 200+ and 50+ unidentified samples from the training and development set respectively.

	SQuAD Open (1-hop)	HotpotQA (2-hop)	3+ Hop	Total
Train	59,285	74,542	0	133,827
Dev	8,132	5,971	0	14,103
Test	8,424	5,978	530	14,932
Total	75,841	86,454	530	162,825

Table 3.1: Statistics of the modified BeerQA.

Since the content of the single-hop QA dataset SQuAD contained in BeerQA is relatively biased, we also construct a new dataset, replacing SQuAD with the cleaned, preprocessed NQ, provided in DPR. Same as above, we filter out unidentified samples from the training and development set respectively, which results in the following dataset. For simplicity, we call this dataset NQ-HotpotQA. Note that we did not include 3+ hop questions from both datasets in our experiments, since MDR was not designed to include the training of 3+ hop questions.

	NQ (1-hop)	HotpotQA (2-hop)	Total
Train	58,644	74,542	133,422
Dev	6,515	5,971	12,486
Total	65,196	86,454	145,908

Table 3.2: Statistics of our newly generated dataset.

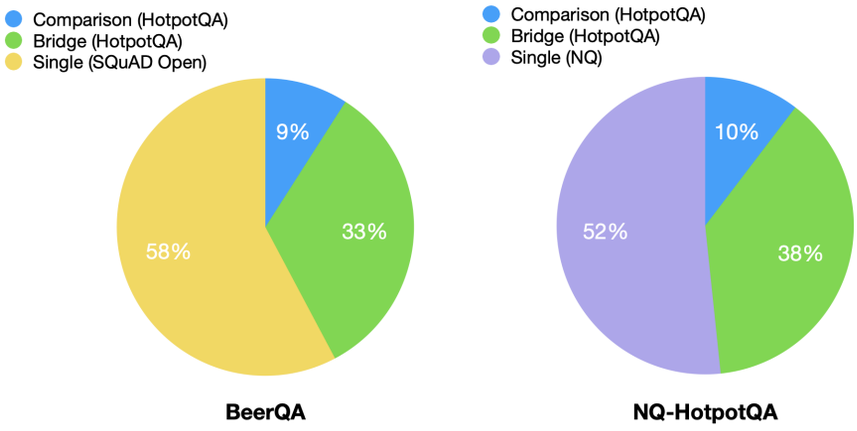

Figure 3.1: Distribution of datasets contained in BeerQA and NQ-HotpotQA

3.2 Training

During training, we used the RoBERTa model provided by Hugging Face[2] as a shared encoder for both questions and passages embedding generation. Instead of feeding the text directly into the model, we first use a matching tokenizer from Hugging Face to preprocess text into a format that is understandable to the model. The tokenizer will split the text into words called tokens. As the next step, it will convert these tokens into numbers in order to construct a tensor as the input to the model. The numerical representation of the tokens can be understood as the model's vocabulary. Taking these as the input, the model returns for each input sample a *(1 × d)* tensor, which is also called embedding. Using the RoBERTa model mentioned above results in $d = 768$.

For the 1st-hop we compute the cross entropy loss of all samples in the batch. Let $\mathbf{Q}$ be a *(n × d)* matrix of question embeddings $\{q_1, q_2, \cdots, q_n\}$, each with shape *(1 × d)*. We construct a *(2n × d)* matrix $\mathbf{P}$, which concatenates the embeddings of 1st-hop and 2nd-hop positive passages paired to each question. We use the notation $p^+_{m,j}$ to denote the embedding of mth-hop positive passage to the jth question. Applying the matrix multiplication $\mathbf{S} = \mathbf{Q}\mathbf{P}^\mathbf{T}$, the result is a *(n × 2n)* matrix of similarity scores. The notation $s_{m,ij}$ is defined as $s_{m,ij} = sim(q_i, p_{m,j})$. Any similarity score $s_{m,ij}$ is a result of a correctly paired example only when i=j. For clarity we also marked these examples with $s^+_{m,ij}$.

[2] https://huggingface.co/roberta-base

$$\begin{bmatrix} — & q_1 & — \\ — & q_2 & — \\ & \vdots & \\ — & q_n & — \end{bmatrix} \times \begin{bmatrix} | & | & & | & | & | & & | \\ p^+_{1,1} & p^+_{1,2} & \cdots & p^+_{1,n} & p^+_{2,1} & p^+_{2,2} & \cdots & p^+_{2,n} \\ | & | & & | & | & | & & | \end{bmatrix} =$$

$$\begin{bmatrix} s^+_{1,11} & s_{1,12} & \cdots & s_{1,1n} & s^+_{2,11} & s_{2,12} & \cdots & s_{2,1n} \\ s_{1,21} & s^+_{1,22} & \cdots & s_{1,2n} & s_{2,21} & s^+_{2,22} & \cdots & s_{2,2n} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots \\ s_{1,n1} & s_{1,n2} & \cdots & s^+_{1,nn} & s_{2,n1} & s_{2,n2} & \cdots & s^+_{2,nn} \end{bmatrix}$$

For each question q_i we also select two hard negative passages $\{p^-_{i,1}, p^-_{i,2}\}$ and compute the similarity score of them. This results in a *(n × 2)* matrix. We denote the similarity score of the ith question and jth hard negative passage with $s^-_{i,j}$. As next step, we concatenate this matrix with the matrix $\mathbf{S}$ mentioned above, resulting in the following matrix representation:

$$\begin{bmatrix} s^+_{1,11} & s_{1,12} & \cdots & s_{1,1n} & s^+_{2,11} & s_{2,12} & \cdots & s_{2,1n} & s^-_{1,1} & s^-_{1,2} \\ s_{1,21} & s^+_{1,22} & \cdots & s_{1,2n} & s_{2,21} & s^+_{2,22} & \cdots & s_{2,2n} & s^-_{2,1} & s^-_{2,2} \\ \vdots & \vdots & \ddots & \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \vdots \\ s_{1,n1} & s_{1,n2} & \cdots & s^+_{1,nn} & s_{2,n1} & s_{2,n2} & \cdots & s^+_{2,nn} & s^-_{n,1} & s^-_{n,2} \end{bmatrix}$$

The loss function can then be written as:

$$L_{CE} = -\sum_{i=1}^{n} t_i \frac{e^{s^+_{1,i=j}}}{\sum_{m=1}^{2}\sum_{j=1}^{n} e^{s_{m,ij}} + e^{s^-_{i,1}} + e^{s^-_{i,2}}} \tag{3.3}$$

where $t_i = \{1, 2, \cdots, n\}$ denotes the truth label.

For the second hop, we filter out single-hop questions and will only consider the multi-hop questions in our batch. In order to reformulate the question representation to account for previously retrieved contexts, we concatenate the original question q_i and the first-hop positive passage $p^+_{1,j=i}$ as the new question representation q'_i for second-hop retrieval and will be fed into the encoder model. Again we construct a similarity matrix based on these new reformulated multi-hop questions only and compute the loss function the same as before. In the end, we add up the first-hop and the second-hop loss together as the final loss. For all our experiments we trained the encoder for up to 10 epochs with a learning rate of

10^{-5} using Adam optimizer, linear scheduling with warm-up and dropout rate 0.1.

Since the retrieval accuracy will be strongly influenced by the quantity and the quality of the negative contexts provided in the same batch, the in-batch-negative setting requires a relatively large batch size for efficient encoder training. Note that gradient accumulation will not result in greater performance in this case. In order to increase the batch size without running out of memory, we came up with the idea to decrease the "max_c_len" parameter, which indicates the maximum length of the context to be tokenized. The part of the context that exceeds this maximum length will then be truncated. However, we should not select this parameter too small since this could possibly cause information loss and affect training performance. Therefore, we took a few experiments using MDR training on HotpotQA. For evaluation, we used the metric Mean Reciprocal Rank (MRR) defined as:

$$MRR = \frac{1}{Q}\sum_{i=1}^{Q}\frac{1}{rank_i} \tag{3.4}$$

For a single query q, the reciprocal rank $\frac{1}{rank}$ refers to the position of the positive passage paired to q among all retrieved passages. If the positive passage does not appear in the retrieved passages, the reciprocal rank is then 0. For multiple queries Q, MRR is the mean of the Q reciprocal ranks.

Based on the result shown in Figure 3.1, we claim that a slightly reduced maximum length of tokenization from 300 to 250 will not have much impact on the overall training performance. Hence we reduced 50 for both context and the reformulated second question maximum tokenization length. The largest batch size we could achieve using our modified MDR is 90 and we decide to conduct all our experiments with this fixed batch size.

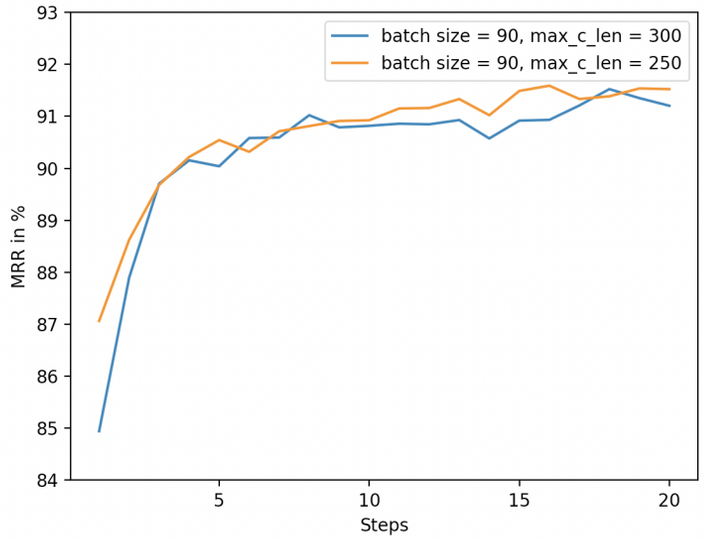

Figure 3.2: MRR evaluation on HotpotQA development set during training

The model will first be trained using 8 Geforce RTX 3090 GPUs. As the next step, we generate embeddings and FAISS index using the trained model for all articles from the preprocessed Wikipedia corpus. Lastly, we will encode each question in the development set and perform the similarity match between questions and contexts by applying the IndexFlatIP from FAISS and evaluate the result using our predefined metrics. The entire process will normally take one or one and a half days.

Chapter 4

Experiments and Results

We conduct our experiments on both BeerQA and our new dataset NQ-HotpotQA. The following metrics are defined for retrieval performance evaluation

P (Average Precision): whether **one** of the supporting passages is included in all retrieved passages;
P-EM (Average Path Exact Match): whether **both** supporting passages are included in all retrieval passages;
1-R (Average 1-Recall): whether **1st-hop** retrieved passage match the ground truth supporting passage;
PR (Average Path Recall): whether any of the top-k retrieved chain extract match the ground-truth supporting passages.

During Inference, we compute the retrieval accuracy of top-k retrieved documents with fixed $k = 2$ in percentage. For efficient similarity search, we apply the exact inner product search index (IndexFlatIP) from FAISS to index the embeddings of the questions and our preprocessed Wikipedia corpus.To better visualize our results, we also provide for each experiment a bar chart including the result of the two most important metrics: The 1-R (Average 1-Recall) and the PR (Average Path-Recall), representing the retrieval accuracy for the 1st-hop and both hops respectively. Note that in this case, we choose PR instead of P-EM, since the order of the retrieved documents is essential for the downstream tasks.

4.1 Experiments on BeerQA

We first keep the original code provided by MDR unmodified. This means, regardless of the different types of questions, we compute the loss as the sum of first-hop and second-hop loss of all the samples. For single-hop questions, we simply concatenate the question and its positive passage as the new question for second-hop retrieval. Since in single-hop samples only one positive passage is provided, we treat this passage as both the first- and the second-hop positive passage paired to the question.

During inference, we evaluate the retrieval performance on the development set according to the question types: comparison, bridge, and single-hop. Compared to the result of MDR trained on HotpotQA only, we observe a slight fall in all metrics evaluated on 2-hop questions. We also notice that the case of comparison questions proves easier in general. This can be explained by the fact that both entities needed for the retrieval are present in comparison questions. On the other hand, MDR achieved a remarkably low accuracy in single-hop question retrieval.

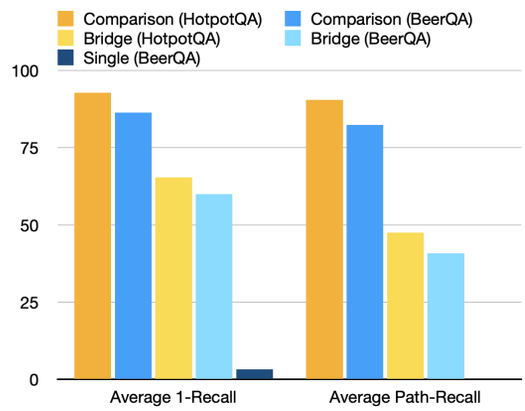

Figure 4.1: Bar chart of results from Table 4.1

	MDR trained on HotpotQA		MDR trained on BeerQA		
Type	Comparison	Bridge	Comparison	Bridge	Single-hop
P	98.8.	73.4	96.9	67.5	–
P-EM	90.9	47.9	82.8	41.1	–
1-R	92.7	65.4	86.3	60.0	3.3
PR	90.5	47.5	82.4	40.8	–

Table 4.1: A comparison of retrieval performance of MDR trained on HotpotQA and BeerQA

The next step is modifying MDR as introduced in section 3.2. During training, single-hop questions will only be accounted for while computing the first-hop loss, whereas multi-hop questions will be considered in both cases as usual. Although the result has slightly improved for all kinds of questions, single-hop retrieval accuracy still remains low. The improvement is far worse than anticipated and almost negligible.

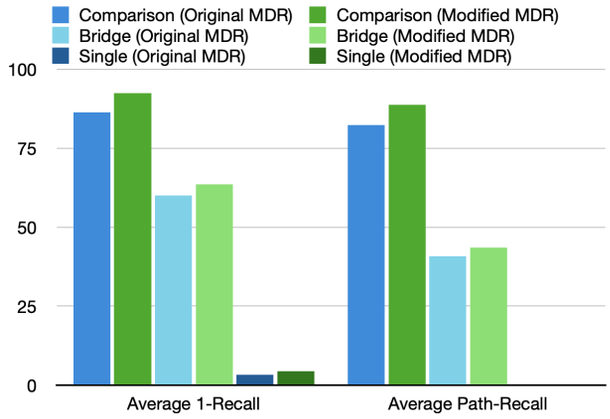

Figure 4.2: Bar chart of results from Table 4.2

	MDR trained on BeerQA			Modified MDR trained on BeerQA		
Type	Comparison	Bridge	Single-hop	Comparison	Bridge	Single-hop
P	96.9	67.5	–	98.4	71.1	–
P-EM	82.8	41.1	–	88.9	44.0	–
1-R	86.3	60.0	3.3	92.4	63.5	4.3
PR	82.4	40.8	–	88.7	43.5	–

Table 4.2: A comparison of retrieval performance of the original MDR and our modified MDR trained on BeerQA

After excluding other potentially relevant factors, we speculate that the choice of the dataset may have contributed to such a result. The single-hop questions in BeerQA were selected from SQuAD Open, in which most questions are comparatively trivial and based on only 500+ Wikipedia articles. This also means that multiple questions are made from the same article, which makes the distribution fairly biased and could greatly influence the final retrieval result. To better simulate real-life scenarios, we conduct further experiments on our new dataset NQ-HotpotQA, which we introduced in the previous chapter.

4.2 Experiments on NQ-HotpotQA

Again, we train the original MDR model and our modified MDR model on the new dataset NQ-HotpotQA and evaluate their performance on the development set. Surprisingly, the Average 1-Recall of single-hop data declines under our modification. In contrast, the Average Path Recall of multi-hop data experienced a slight growth. Nevertheless, the single-hop question retrieval accuracy still remains low.

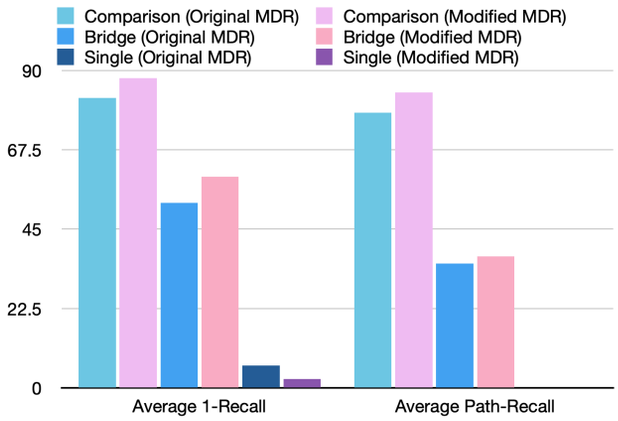

Figure 4.3: Bar chart of results from Table 4.3

	MDR trained on NQ-HotpotQA			Modified MDR trained on NQ-HotpotQA		
Type	Comparison	Bridge	Single-hop	Comparison	Bridge	Single-hop
P	95.6	62.4	–	96.3	67.5	–
P-EM	78.8	35.7	–	84.3	37.9	–
1-R	82.2	52.5	6.3	87.9	59.9	2.5
PR	78.1	35.3	–	83.8	37.3	–

Table 4.3: The retrieval performance of the original MDR trained on our new dataset (NQ-HotpotQA), taking the loss of both retrieval steps into account, compared to taking only the first-hop loss into account.

Based on previous results, we decide to further investigate whether it is caused by our modifications on MDR or if it is simply not feasible to train MDR with a mixed dataset. Hence we designed another experiment to train MDR to retrieve only once, i.e. computing only the first-hop loss in our loss function for all kinds of question samples. We then compare its results with the original MDR. As figure 4.4 illustrated, 1-R increases, and as the consequence of training the model to only retrieve once, PR falls drastically. Even then, single-hop retrieval accuracy

remains low as always.

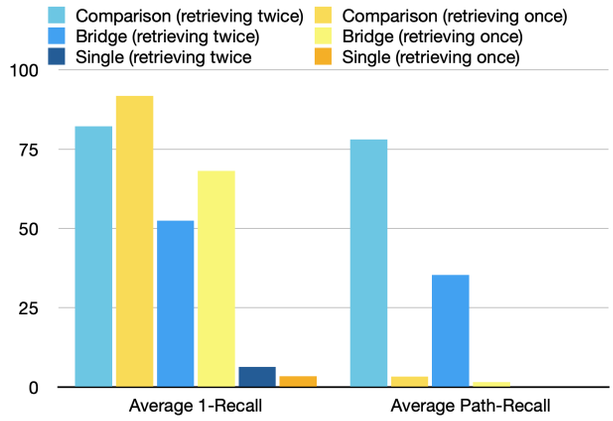

Figure 4.4: Bar chart of results from Table 4.4

	MDR (retrieving twice)			MDR (retrieving once)		
Type	Comparison	Bridge	Single-hop	Comparison	Bridge	Single-hop
P	95.6	62.4	–	92.1	69.0	–
P-EM	78.8	35.7	–	41.5	5.3	–
1-R	82.2	52.5	6.3	91.8	68.2	3.4
PR	78.1	35.3	–	3.3	1.6	–

Table 4.4: The retrieval performance of the original MDR compared to MDR trained to retrieve only once.

To ensure that MDR also works well on single-hop datasets, we trained MDR on NQ only, computing only the first-hop loss as the overall loss. Furthermore, we also trained MDR on NQ-HotpotQA, however this time only taking the first-hop loss into account. We compare the single-hop retrieval results in table 4.5 and discover that the single-hop retrieval performance will strongly deteriorate if we use the embeddings computed from the model trained on a mixed dataset (i.e. containing both single-hop and multi-hop data), no matter in which way we design the training procedure. Therefore we conclude that it is unfeasible to train a uniform model as we presumed.

	MDR trained on NQ-HotpotQA			MDR trained on NQ only
	retrieving twice	retrieving once	modified	all retrieving once
1-R	6.3	3.6	2.5	23.1

Table 4.5: A comparison of single-hop retrieval performance of MDR trained on NQ-HotpotQA under different types of modification, and MDR trained on NQ, taking only the 1st-hop retrieval loss into account. "Modified" represents our modification of MDR introduced in the previous chapter

4.3 Question Hop Classification

During Inference, the system needs to first identify the number of reasoning steps required to answer the question and then adjust the number of its retrieving steps. In this case, we propose to train a classifier that can identify the number of hops in a given question. Since we only focused on datasets consisting of questions up to 2 hops, we decide to design a binary classifier to tackle this problem. We trained our classifier taking the question embeddings generated from our uniform model as inputs, and using binary cross entropy loss as the loss function. After some attempts, we find out that a simple Multi-Layer Perceptron can achieve significantly high classification accuracy using only 1/30 of the data for training.

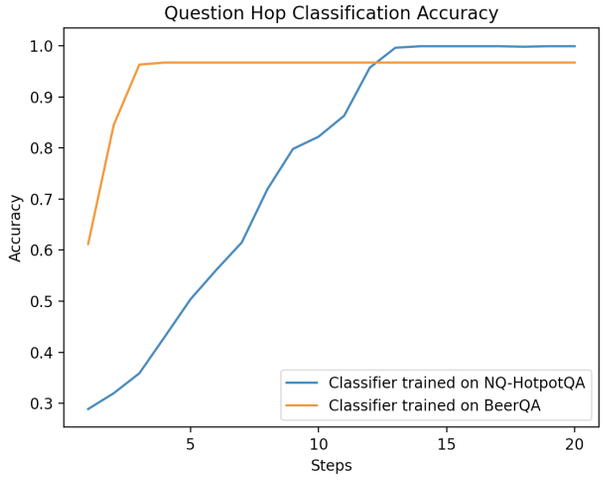

Figure 4.5: Accuracy of our Binary Classifier evaluated on development set within 200 training Steps (1/30 Epoch).

After training one entire epoch, we get the following result evaluated on the development set of BeerQA and NQ-HotpotQA:

	BeerQA	NQ-HotpotQA
Classification Accuracy	96.4	99.8

Table 4.6: Question hop classification accuracy evaluated on BeerQA and NQ-HotpotQA development set after training 1 epoch.

4.4 Pipeline System

Although our attempt of creating an end-to-end QA system by training a uniform model proves unsuccessful, we created an alternative solution based on our previous findings: A pipeline system illustrated in figure 4.6. The system first encodes a given question using our uniform model trained on the "mixed-hop" dataset. Next, the question embedding will be fed into a binary classifier which decides whether the question is single-hop or multi-hop. If the question is classified as single-hop, it will be fed into Model A, which stands for a dense retriever model trained on single-hop datasets only. Otherwise, it will be transferred into Model B, which was trained on 2-hop datasets. Finally, the top-k most matching documents will be retrieved after the similarity match between the encoded question and Wikipedia articles.

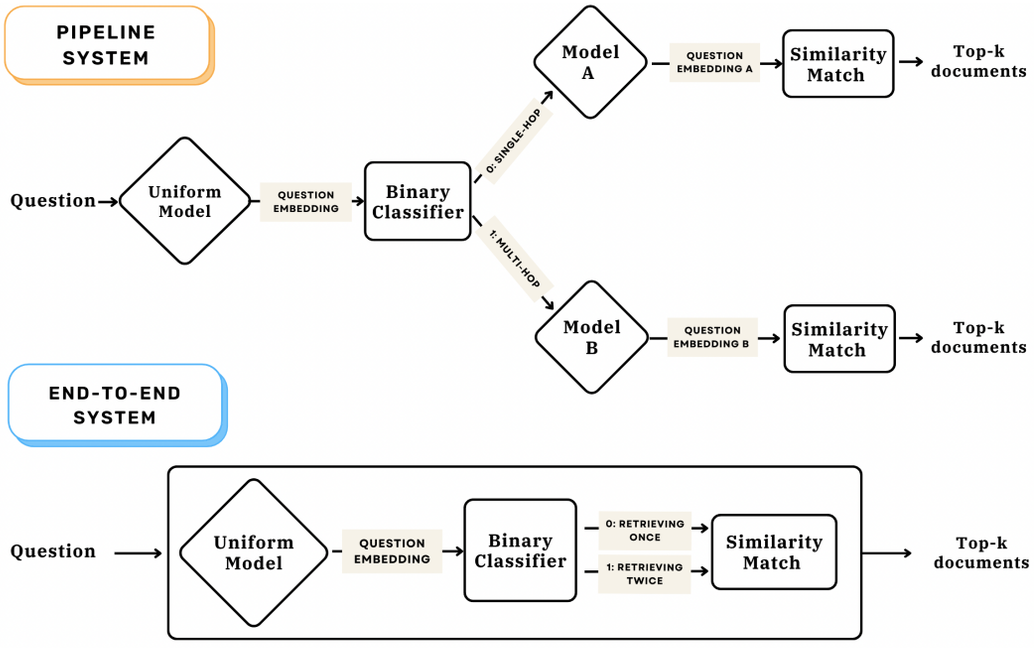

Figure 4.6: An illustration of our pipeline system and the desired end-to-end system. Model A refers to the model trained on single-hop dataset only. Model B refers to the model trained on multi-hop dataset only.

An Advantage of our pipeline system is that we can freely choose model A and model B. Since the performance of dense retrieval rises with increased batch size, we trained model A and B using MDR with batch size 140, which is the largest batch size we can achieve on our machines. The single-hop retrieval may possibly be further improved by using independent encoders for questions and contexts instead of the shared encoder suggested in MDR. Table 4.7 shows our final result compared to the previous result we get using the original MDR.

	Original MDR	Pipeline System
1-R (Single-hop)	6.3	29.1
PR (Multi-hop)	44.5	49.1
Total	24.8	38.8

Table 4.7: A comparison of retrieval performance of the original MDR and our pipeline system, both trained on NQ-HotpotQA, using the metric 1-R (Average 1-Recall) for single-hop questions and PR (Path Recall) for multi-hop questions

Chapter 5

Conclusion and Future Work

5.1 Conclusion

In this thesis, we have made attempts on a topic that has been rarely researched: We aim to develop an end-to-end trainable QA system, which can provide precise answers to both single- and multi-hop questions without predefining a fixed number of reasoning steps. Since the retrieval step in the first stage is crucial for the downstream reader task and requires further improvements, we decide to focus on the retriever part.

The main objective is to train the encoder in our retriever model by learning a better embedding function, that generates a vector space in which relevant pairs of questions and passages have a lower distance. Unfortunately, our experiments have proved that it is impossible to create such a vector space that meets our expectations for both single- and multi-hop questions. Prioritizing any one of them will degrade the performance of the other. Consequently, we argue that it is currently unfeasible for us to develop a uniform model for all kinds of questions using dense retrieval.

On the other hand, we achieved success in identifying the number of required reasoning steps to answer a question. Based on this, we provide an alternative solution: A pipeline system containing a binary classifier to identify the number of question hops and two separate models trained on single-hop and multi-hop datasets respectively. Although this solution greatly outperforms our original end-to-end system, its training and inference procedure is in general much more complicated and time-consuming. Therefore, we would not consider further developments on this system.

5.2 Future Work

For future investigations, we would suggest exploring possibilities in other end-to-end multi-hop reasoning methods. Apart from the Query Reformulation setting we introduced in this thesis, another popular workflow of the Iterative Retriever is the Retrieval Stopping Mechanism: The retriever terminates its retrieval process when it detects that a sufficient amount of relevant documents are already present. We believe further effective models can be designed based on this architecture. As we only conducted experiments on 1-hop or 2-hop datasets, it is also necessary to extend the model functionality to answer 3+ hop questions.

Bibliography

[1] E. M. Voorhees, "The TREC-8 question answering track report," in *TREC, volume 99, pages 77-82*, 1999.

[2] F. Zhu, W. Lei, J. Z. Chao Wang, S. Poria, and T.-S. Chua, "Retrieving and reading: A comprehensive survey on open-domain question answering," in *arXiv:2101.00774*, 2021.

[3] S. Mussmann and S. Ermon, "Learning and inference via maximum inner product search," in *International Conference on Machine Learning (ICML), pages 2587-2596*, 2016.

[4] V. Karpukhin, B. Oğuz, S. Min, P. Lewis, L. Wu, S. Edunov, D. Chen, and W. tau Yih, "Dense passage retrieval for open-domain question answering," in *EMNLP. Association for Computational Linguistics, pages 2587-2596*, 2020.

[5] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," in *North American Association for Computational Linguistics (NAACL)*, 2019.

[6] W. Xiong, X. L. Li, S. Iyer, J. Du, P. Lewis, W. Y. Wang, Y. Mehdad, W. tau Yih, S. Riedel, D. Kiela, and B. Oğuz, "Answering complex open-domain questions with multi-hop dense retrieval," in *arXiv:2009.12756*, 2020.

[7] C. Raffel, N. M. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," in *arXiv:1910.10683*, 2020.

[8] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," in *arXiv:1906.00300*, 2019.

[9] A. Asai, K. Hashimoto, H. Hajishirzi, R. Socher, and C. Xiong, "Learning to retrieve reasoning paths over wikipedia graph for question answering," in *Proceedings of ICLR*, 2020.

[10] Q. Guo, F. Juefei-Xu, C. Zhou, Y. Liu, and S. Wang, "Sparta: Spatially attentive and adversarially robust activation," in *arXiv:2105.08269*, 2021.

[11] P. Qi, X. Lin, L. Mehr, Z. Wang, and C. D. Manning, "Answering complex open-domain questions through iterative query generation," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*

and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 2590–2602., 2019.

[12] B. Dhingra, M. Zaheer, V. Balachandran, G. Neubig, R. Salakhutdinov, and W. W. Cohen, "Differentiable reasoning over a virtual knowledge base," in *Proceedings of ICLR*, 2020.

[13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," in *arXiv:1907.11692*, 2019.

[14] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[15] J. Johnson, M. Douze, and H. Jégou., "Billion-scale similarity search with gpus," in *arXiv:1702.08734*, 2017.

[16] Z. Yang, P. Qi, S. Zhang, Y. Bengio, W. W. Cohen, R. Salakhutdinov, and C. D.Manning, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," in *EMNLP, pages 2369–2380*, 2018.

[17] T. Kwiatkowski, J. Palomaki, O. Redfield, M. Collins, A. Parikh, C. Alberti, D. Epstein, I. Polosukhin, M. Kelcey, J. Devlin, K. Lee, K. N. Toutanova, L. Jones, M.-W. Chang, A. Dai, J. Uszkoreit, Q. Le, and S. Petrov, "Natural questions: A benchmark for question answering research," in *Transactions of the Association of Computational Linguistics (TACL)*, 2019.

[18] P. Qi, H. Lee, T. Sido, and C. Manning, "Answering open-domain questions of varying reasoning steps from text," in *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing, pages 3599–3614*, 2021.