SmallBPR

Parameters Sharing for Binary Passage Retriever

Li, Zhiyao

Award date:
2022

Technische Universiteit
Eindhoven
University of Technology

Department of Mathematics and Computer Science
Data Mining Group

SmallBPR: Parameters Sharing for Binary Passage Retriever

Master Thesis

Zhiyao Li
z.li6@student.tue.nl

Supervisors:
Dr. Meng Fang

Eindhoven, July 2022

Abstract

In open domain question answering systems, Dense Passage Retriever (DPR) has become a popular approach to retrieving relevant passages for finding answers. However, the biggest shortcoming of such a system is it takes too much memory to store the index and model. Though Binary Passage Retriever (BPR) significantly reduces the size of the index, it still suffers from the heavy model size because of large pre-trained models. In this paper, we introduce a new model named SmallBPR, a smaller model extension based on BPR. We approach two ways to reduce the model size without a significant loss in model performance. Firstly, we use the shared encoder for the retriever of SmallBPR to reduce the model size to half of the previous. Second, we use small pre-trained models, and then the encoder is replaced with faster, smaller, and cheaper models to reduce the model size further. Experiments on Natural Questions and TriviaQA show the effectiveness of our models. The performance of the model remains almost the same, with a significant reduction in model size. Surprisingly, SmallBPR-ElectraSmall can achieve 66.1% top 100 on Natural Questions while the model size is only 154MB.

Preface

This thesis is my final assignment at TU/e, and my last assignment as a student. I am delighted that I spent two years as a master student in Eindhoven, which is very interesting and surprising. I am glad I chose Data Science in Engineering two years ago, and I learned a lot in the past two years. After this graduation project, I will transform from a student to a software development engineer. I will apply all I have learned in Tu/e to the new areas.

I want to thank all of you for helping me write my thesis. My deepest gratitude is first to professor Meng Fang, my supervisor, for his constant encouragement and guidance. He gave me selfless advice throughout my graduation project and solved the problems I encountered. Without his usual and enlightening guidance, this paper would not have been in its present form. Secondly, I would like to express my heartfelt thanks to professor Tim A.E. Ophelders, and professor Yulong Pei for accepting to serve in my committee.

Finally, I would like to thank my dear parents for their love and great faith in me for many years. I also want to thank my friends and classmates who have given me help and time to listen to my opinions and help me solve my problems in the complicated process of my thesis.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

1.1 Background

In this section, we introduce the basic knowledge of our research questions, Question-Answering, a sub-field of Natural Language Processing (NLP). Further, we explain the difference and connection between question-answering and open domain question-answering. Meanwhile, we also list some standard methods used in Question-Answering.

1.1.1 Natural Language Processing

Natural language processing applies computers and software to capture the meaning of human language (written or spoken).More specifically, artificial intelligence is used to process and analyze text or voice data to understand and interpret content, classify content, and derive insights from content. NLP is widely used in emotion analysis, speech recognition, question-answering, and machine translation.

1.1.2 Question-Answering

Question-Answering(QA) is one of the most critical tasks in natural language processing, which aims to generate answers automatically based on questions. There are two main challenges in QA tasks, which are the irregularity of language and the use of language depending on background. At the beginning, QA systems require all kinds of manually compiled data. When a user asked a question, the system converted the user's question into a structured query statement. The system searched this statement from the database and return related data to the user. The first question-answering system, named START, was proposed by the MIT Artificial Intelligence Lab in 1993 [21]. The system can return a short and precise answer text directly to the user, which clearly distinguishes itself from search engine systems that return a list of web pages to users.

The processing framework of a general QA system is shown in Fig 1.1, including three functional components: Question Understanding, Information Retrieval, and Answer Generation. [13] Question Understanding module is responsible for processing the user's questions, generating query keywords, determining the type of answer to the question, and other tasks. Information Retrieval module matches documents, paragraphs and sentences based on keywords in the query and sends available messages to the Answer Generation module. In Answer Generation module, it extracts possible answers from the relevant messages retrieved by the Information Retrieval module. After that, all possible answers are scored according to some principles. Then the answer with the highest score is returned to the user.

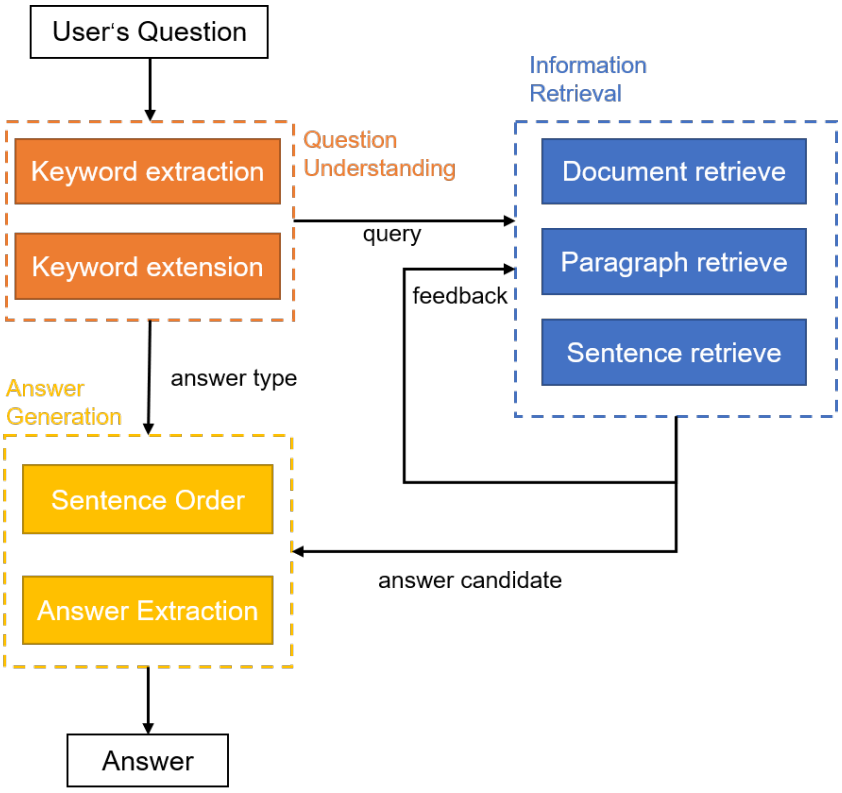

Figure 1.1: The processing framework of a general QA system.

1.1.3 Open Domain Question-Answering

For standard QA systems, they can only answer questions in a specific area. Open Domain QA (ODQA) [41] is a new branch of the QA system to answer questions in different domains from unstructured documents. The ODQA system has a distinctive features that it can answer questions that are not limited to a specific field. Same to standard QA systems, ODQA is based on a document database and can only answer questions whose answers exist in these document databases. The structure of the early ODQA system is so complicated and consists of many components [12, 29]. An ODQA task always consists of many sub-tasks, including query understanding, candidate selection, and answer extraction. Recently, the structure of the ODQA system has been simplified into two parts: **Retriever** and **Reader** [7].

- **Retriever:** its function is to get a few passages that are likely to contain the answer field of the given question from unstructured documents, and to pass these passages to Reader for further processing.
- **Reader:** it is responsible for extracting precise answers from the selected passages.

Wikipedia corpus is one of the common documents used in ODQA [7]. In comparison with knowledge bases, like FreeBase [3], which is structured but too sparsely populated, Wikipedia corpus is designed for human beings to read. The passages in Wikipedia corpus are up-to-date and attractive for human readers.

The conventional method of retrieval in ODQA uses TF-IDF [40] and BM25 [35], which are classic algorithms used in the information retrieval field to calculate query-to-passage similarity scores.

TF-IDF is a statistical method to assess the importance of a word for a document in the document set. The importance of a word increases proportionally with the number of times it

appears in a document, but at the same time decreases inversely with the frequency with which it appears in the corpus.

BM25 is an algorithm used to evaluate the relevance between search terms and documents, and it is an algorithm proposed based on a probabilistic retrieval model. In order to calculate the BM25 score of each query to a document, the query will be cut into several words and calculate the score of each word independently. Eq.1.1 shows how to calculate the BM25 score:

$$\text{Score}(Q, d) = \sum_{i}^{n} W_i R\left(q_i, d\right), \tag{1.1}$$

where Q and d represents one query and one document, q_i represents each word in the query, and W_i is the weight of the word q_i. $R\left(q_i, d\right)$ is the relevance between the word and the document. The BM25 score of one word consists of three main parts: (1) Relevance between the word in the query and the document. (2) Similarity between the word and the query. (3) Weight of each word. Finally we do a summation of the score of each word in the query to get the final score between the query and the document.

In TF-IDF and BM25, they firstly calculate the relevance of each word and passage based on word weight and word frequency. And then calculate the relevance of the query based on each word in this query. This method is efficient but cannot consider word semantics. Synonyms or paraphrases that consist of completely different tokens cannot be recognized by this algorithm.

1.1.4 Dense Representation

In order to resolve the problem in TF-IDF and BM25, the *dense* [20], latent semantic encoding can be used to analyze the semantic meaning. All texts can be translated into continuous vectors containing the semantic by the embedding function. Most efficient retrievers encode the passages in the wikipedia and questions into continuous vectors as index, and then using a nearest neighbor search on the index [20]. And in the retriever, there are two independent encoders processing passages and questions separately [4, 14, 23].

It is well known that learning an excellent dense vector representation needs a large number of labeled pairs of questions and contexts. Using a standard pre-trained model can significantly simplify the training process. Bidirectional Encoder Representations from Transformers (BERT [10]) is an efficient pre-trained model which can be used in a broad set of NLP tasks. BERT can be used in ODQA system after fine-tuned with only pairs of questions and passages.

Many Question-Answering datasets can be used to fine-tune BERT in ODQA system. Natural Questions (NQ) [22] is a question-answering dataset, where all questions are real and aggregated queries collected from the Google search engine. For each data in NQ dataset, it consists of a question, a wikipedia page and a pair of long and short answers. Fig 1.2 shows an example in the NQ dataset. It contains two types of answers, the long answer and the short answer for each question in the NQ dataset.

Trivia QA (TQA) [19] is a QA dataset comprising 650 thousand triples of question-answer-evidence. Table 1.1 shows some examples of Trivia QA. TQA not only provides the pair of question-answer, but it also provides some evidence of how to find to answer.

All these datasets can be used to fine-tune the BERT model as encoders in Retriever. However, a shortcoming of such retrievers is the large memory cost. All indices of passages in wikipedia need to be stored entirely in memory at runtime. For example, the wikipedia corpus with 21 million English passages requires 65 GB for storing index [20]. To solve the memory problem, a method, Binary Passage Retriever (BPR) [44], adds a hash function after the embedding function, converting continuous vectors into the binary code, which take up less memory.

Example
Question: where is the bowling hall of fame located
Wikipedia page: International Bowling Hall of Fame
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

Figure 1.2: An example from NQ dataset, including question, wikipedia page, long answer and short answer.

Q_id	Question	Answer	Evidence
qw_3199	Miami Beach in Florida borders which ocean?	Atlantic	https://nlp.cs.washington.edu/triviaqa/ms/qw_3199.html
bt_1255	What was the occupation of Lovely Rita according to the song by the Beatles	Traffic Warden	https://nlp.cs.washington.edu/triviaqa/ms/bt_1255.html
qg_77	Who was Poopdeck Pappys most famous son?	Popeye	https://nlp.cs.washington.edu/triviaqa/ms/qg_77.html
wh_1026	The Nazi regime was Germany's Third Reich; which was the first Reich?	HOLY ROMAN EMPIRE	https://nlp.cs.washington.edu/triviaqa/ms/wh_1026.html

Table 1.1: Some data examples, question_id, question, answer and corresponding evidence of Trivia QA.

1.2 Research Question

Many state-of-the-art neural models for NLP are heavily parameterized and thus memory inefficient. Given the limitation of previous research on the ODQA system, we hope to decrease the size of Retriever part without a significant loss in retrieval performance. Hereby we raise three research questions:

- The Dual-encoder architecture [4] of the dense embedding model wastes lots of resources because the way of processing questions and passages is similar. How can we develop a method to process questions and passages in a simpler and more efficient way?

- BERT is used as the encoder in several research [11, 24]. Can we find some models with smaller sizes to replace BERT?

- After adjusting the structure of the dense embedding model and replacing BERT with a simpler, cheaper and faster NLP model, can we keep the model's performance without significant degradation?

1.3 Solutions and Contributions

It is difficult to execute pre-trained language models efficiently on resource-restricted devices because they are usually computationally expensive. In this paper, we propose a simplified BPR model, name SmallBPR, via reducing the model size and speeding up the searching time.

First, we use the shared encoder for the retriever of SmallBPR to reduce the model size to half. Thus, the typical dual-encoder architecture can use the same parameters. Second, we use smaller, faster and cheaper pre-trained models to replace the BERT model in the encoder part. Experiments show that compared with BPR, SmallBPR reduces the model size from 2.4GB to less than 1.2GB without a performance loss on two standard ODQA benchmarks: Natural Questions [22] and TriviaQA [19]. It should be mentioned that SmallBPR with ElectraSmall [8] can achieve the accuracy 66.1% while the memory cost is only 154MB. Our main contributions include:

- We propose a new model, SmallBPR, which applies a shared encoder to the retriever part of BPR.
- We explore several small pre-trained models to replace the commonly used BERT model, and reduce the model size further.
- Experimental results show that the proposed model can significantly reduce the model size while the performance is satisfactory to some extent.

1.4 Thesis Structure

Following the introduction, Chapter 2 provides a detailed literature review on some passage retriever models. Chapter 3 introduces knowledge about NLP models, which are always used in question-answering systems and how we adopt these models in this thesis. Chapter 4 introduces key points of our SmallBPR model and detailed steps about how we train the model. Chapter 5 elaborates experimental setups, results and analyses. Finally, we conclude our thesis with limitations and future work in Chapter 6.

Chapter 2

Related work

In this chapter, we provide a detailed literature review on the dense retrieval and open domain question answering. We first introduce how to use wikipedia on ODQA in §2.1. Secondly we introduce Dense Passage Retriever in §2.2 and explain how Efficient Passage Retriever with hash function works to reduce the index size in §2.3.

2.1 Wikipedia on Open-Domain Questions

ODQA was originally defined to find answers in unstructured documents. But most researches are based on structured database, like FreeBase KB [3]. These researches have the same fixed mode. A new approach to answer questions from raw text, Wikipedia, is proposed [7]. The advantage of Wikipedia is flexible and fast updates. The structure of the ODQA system based on Wikipedia is shown in Fig 2.1.

Figure 2.1: An overview of question answering system based on Wikipedia.

There are two main parts in this system: Document Retriever and Document Reader. In Retriever, TF-IDF is used to compare the similarity between passages and questions. Top 5 passages are chosen and sent to the Reader part. In Reader, passages and questions are encoded into vectors by a recurrent neural network (RNN) [46]. The span in the passage with the highest similarity to the question vector is considered as the answer.

2.1.1 Document Retriever

There are three main algorithms used in the Document Retriever: TF-IDF, murmur3, N-gram. We will introduce the detail of each algorithm and how they process the raw text and analyzed each word individually.

TF-IDF TF-IDF is used to measure the correlation between passages and questions. There are three steps to select related passages. TF-IDF is a statistical method to assess the importance of a word for a document in the document set. The importance of a word increases proportionally with the number of times it appears in a document, but at the same time decreases inversely with the frequency with which it appears in the corpus:

$$\mathrm{tfidf_{i,j}} = \mathrm{tf_{i,j}} \times \mathrm{idf_i}. \tag{2.1}$$

TF (Term frequency) refers to the frequency with which a given term appears in the document:

$$\mathrm{tf_{i,j}} = \frac{n_{i,j}}{\sum_k n_{k,j}}, \tag{2.2}$$

where $n_{i,j}$ is the number of occurrences of the word in document d_j. IDF (Inverse document frequency) is a measure of the general importance of a term:

$$\mathrm{idf_i} = \log \frac{|D|}{1 + |\{j : t_i \in d_j\}|}, \tag{2.3}$$

where $|D|$ is the total number of documents in the document set, and $|\{j : t_i \in d_j\}|$ is the number of documents containing the given word.

N-gram N-Gram is an algorithm based on a statistical language model. Its basic idea is to operate a sliding window of size N on the content, forming a sequence of length N. In Document Retriever, it uses a 2-gram operation. Both questions and passages are processed into a sequence of two words. The example of N-gram is shown in Fig 2.2.

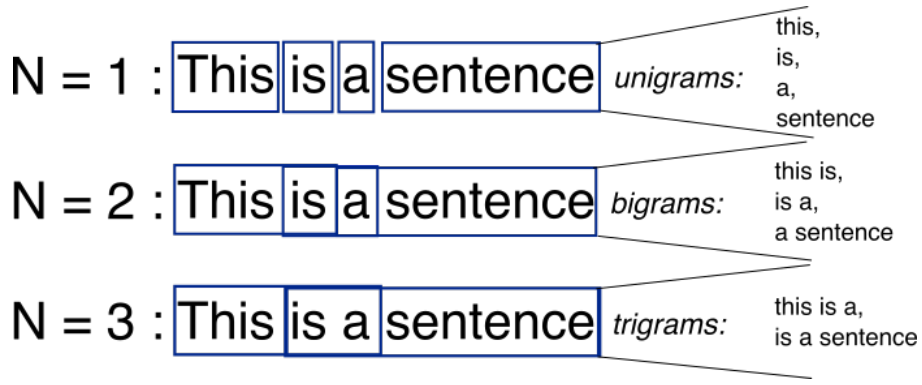

Figure 2.2: The example of unigrams, bigrams and trigrams for a sentence.

murmur3 After getting a processed sequence, these sequences are transferred into hash code by a hash function, murmur3, which can generate 32-bit and 128-bit hash values. Hash values are more evenly distributed, so there are almost no collisions. The algorithm detail of murmur3 is shown in Fig 2.3. In murmur3 algorithm, it can generate some seed c_1, c_2, m and n automatically. Then the input is divided into four sub-parts. For each sub-part, it can multiply by c_1 and do a rotate left operation. After that it can do a *exclusive or* operation and then multiply m, add n. After four times repetitive operations, the final value is the murmur3 hash value. Murmur3 hash

code is well distributed and suitable across several domains: ids generator, checksums, hash tables, bloom filters, hyperlog and machine learning. It almost never has collisions in murmur3-128 bit version.

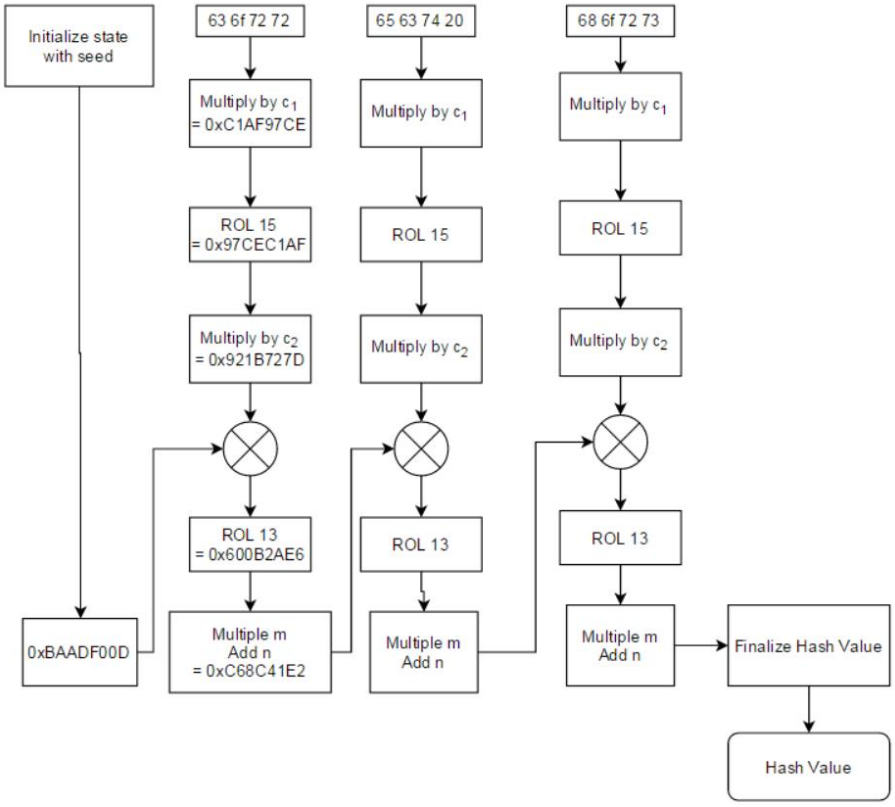

Figure 2.3: The process of generating hash value by murmur3.

2.1.2 Document Reader

In the Document Reader part, the neural network is used to encode paragraphs, encode questions and make predictions. Passages and questions are encoded into vectors separately, and then these vectors are used to predict the answer.

Paragraph encoding For each paragraph, all tokens p_i of each paragraph are represented into a sequence of feature vectors $\tilde{\mathbf{p}}_i \in \mathbf{R}^d$. A recurrent neural network (RNN) [27] is used to convert the sequence into a vector $\mathbf{p}_i$, which contains useful context information:

$$\{\mathbf{p}_1, \ldots, \mathbf{p}_m\} = \mathrm{RNN}\left(\{\tilde{\mathbf{p}}_1, \ldots, \tilde{\mathbf{p}}_m\}\right). \tag{2.4}$$

The internal state of an RNN can demonstrate dynamic temporal behavior. RNN can use its internal memory to process arbitrary temporal sequences of inputs, which makes it easier to handle NLP tasks.

Question Encoding Same as paragraph encoding, RNN is also used in question encoding. Each word in the question corresponds to a word embedding, and the RNN converts each word embedding into a vector $\{\mathbf{q}_1, \ldots, \mathbf{q}_l\}$. All vectors of one question are combined into one vector

$\mathbf{q} = \sum_j b_j \mathbf{q}_j$. b_j represents the importance of one word:

$$b_j = \frac{\exp(\mathbf{w} \cdot \mathbf{q}_j)}{\sum_{j'} \exp(\mathbf{w} \cdot \mathbf{q}_{j'})}, \tag{2.5}$$

where $\mathbf{w}$ is the weight vector to learn.

Prediction After paragraph encoding and question encoding, there are one question vector $\mathbf{q}$ and a set of passage vector $\{\mathbf{p}_1, \ldots, \mathbf{p}_m\}$. They are used to predict a span in the passage which may be the right answer. A bilinear term is used to capture the similarity between passage vector $\mathbf{p}_i$ and question vector $\mathbf{q}$. For each passage vector, the probability of being a start or a end of an answer is computed separately:

$$P_{\text{start}}(i) \propto \exp(\mathbf{p}_i \mathbf{W}_s \mathbf{q}) \tag{2.6}$$

and

$$P_{\text{end}}(i) \propto \exp(\mathbf{p}_i \mathbf{W}_e \mathbf{q}), \tag{2.7}$$

where $\mathbf{W}_s$ and $\mathbf{W}_e$ are two weight vectors to learn. The span from word i to word i' with the highest value of $P_{\text{start}}(i) \times P_{\text{end}}(i')$ can be treated as the right answer. In order to prevent the answer from being too long, another constraint $i \leq i' \leq i + 15$ is set.

2.1.3 Dataset

Several datasets are used to train the Document Retriever and Document Reader. Also, a Wikipedia dataset is needed as a knowledge resource. In this research, 2016-12-21 dump of English Wikipedia is used for all full-scale experiments. Some pre-processing is needed for the Wikipedia corpus because not all pages contain plain text. Lists, figures, and some internal disambiguation are discarded. After the pre-processing, there are 5,075,182 articles left.

The Stanford Question Answering Dataset (SQuAD) [33] is used to train the Document Retriever. SQuAD 1.1 contains 107,785 question-answer pairs on 536 articles. Different from some semi-synthetic datasets, like Daily Mail [38], the question of SQuAD is generated by human beings so that such questions are more in line with human reading habits. Meanwhile, SQuAD made innovations to the form of the answer, which is text span, a small paragraph from the original article as the answer to the question. Compared with free form, the form of text span has clear fixed answers and it is easier to measure the performance of the model. Another advantage of SQuAD is that there are many types of answers. Table 2.1 shows different answer types and their percentage and example in SQuAD 1.1. There are 10 different answer types in SQuAD 1.1 so that the model trained by SQuAD 1.1 can be used to answer various questions in different formats and in different domains.

Answer Type	Percentage	Example
Date	8.9%	19 Oct 1512
Other Numeric	10.9%	12
Person	12.9%	Thomas Coke
Location	4.4%	Germany
Other Entity	15.3%	ABC Sports
Common Noun Phrase	31.8%	property damage
Adjective Phrase	3.9%	second-largest
Verb Phrase	5.5%	returned to Earth
Clause	3.7%	to avoid trivialization
Other	2.7%	quietly

Table 2.1: Different answer types and their percentage and example in SQuAD 1.1 [33]. There are more 9 different types included in SQuAD 1.1.

Besides SQuAD 1.1, some other datasets which are constructed in different ways are also used to train and evaluate the model. The purpose of introducing additional datasets is to train the model to adapt to different types of questions so that the model can produce different types of answers. There are three datasets used in the experiment, CuratedTREC [1], WebQuestions [2] and WikiMovies [28]. The number of train and test examples in each dataset is shown in Table 2.2.

Dataset	Train	Test
SQuAD	71231	10570
CuratedTREC	3464	694
WebQuestions	4602	2032
WikiMovies	36301	9952

Table 2.2: The number of train and test examples used in Document Retriever and Document Reader for each dataset.

2.2 Dense Passage Retriever

Wikipedia is used as a corpus to answer random questions in several research [25, 34]. And using labeled pairs of questions and passages to train dense encoders became popular recently [16, 30]. The shortcoming of most research is training a model with good performance needs too many pairs of questions and answers. In DPR, BERT is used as encoders for passages and questions separately.

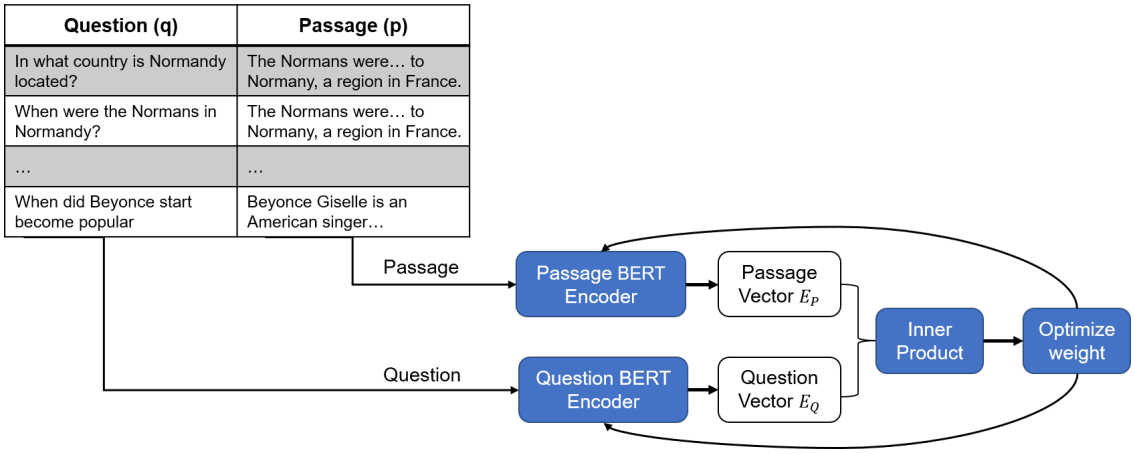

Figure 2.4: A high-level view of the flow of data through a DPR model during training.

DPR uses a dense encoder $E_P(\cdot)$ to translate any text passage into a d-dimensional real-valued vectors. All passages in the Wikipedia corpus are translated into vectors as the index, which will be used in retriever. DPR also applied another encoder $E_Q(\cdot)$ to process questions into a d-dimensional vector. The similarity between questions and passages is defined by an inner product of their vector:

$$\text{sim}(q, p) = E_Q(q)^\top E_P(p). \tag{2.8}$$

FAISS After generating index from all passages in Wikipedia. DPR also adopts Faiss [18], a framework that allows GPU to quickly search for vectors of multimedia documents that are similar to each other. For a input question, DPR generates its corresponding vector $V_q = E_Q(q)$, and retrieves the top-k passages indices with embeddings closest to V_q.

Training Inner product similarity is a good metric to be used in training. The goal of training is to create a metric space where relevant passages and questions could have a higher value in inner product than irrelevant pairs. The training set $\mathcal{D} = \left\{\left\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-\right\rangle\right\}_{i=1}^m$ consists of one question, a positive passage p_i^+ and n negative passages$q_{i,n}^+$, where the positive passage contains the answer to the question. The loss function is designed to minimize the negative log-likelihood of the positive passage:

$$L(q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-) = -\log \frac{e^{\text{sim}\left(q_i, p_i^+\right)}}{e^{\text{sim}\left(q_i, p_i^+\right)} + \sum_{j=1}^n e^{\text{sim}\left(q_i, p_{i,j}^-\right)}}. \tag{2.9}$$

Fig 2.4 illustrates the main process of training BERT model of DPR on some question and passage examples. For question and passage, they are processed by two encoders separately, translating into two vectors. The next step is calculating similarity by the inner product. And the result can also be used to optimize the weight of two encoders.

How to choose positive and negative passages is an essential topic for training, but the way of selecting negative passages is always neglected. Generally, negative passages are randomly selected from a large corpus. Three methods of generating negative passages are proposed in DPR: (1) Randomly select any passage from the corpus. (2) Selecting top passages by BM25 which include most question tokens but do not contain the answer. (3) Taking positive passages paired with other questions as the negative passages in this pair.

2.3 Binary Passage Retriever

The problem in the DPR project is that the size of indices is too large. The size of all indices made from Wikipedia corpus is 65GB. BPR proposed a new way to decrease the size of indices by adding a hash function. In DPR, all passages are converted into continuous vectors by a BERT encoder. In BPR, a hash function is added after each encoder.

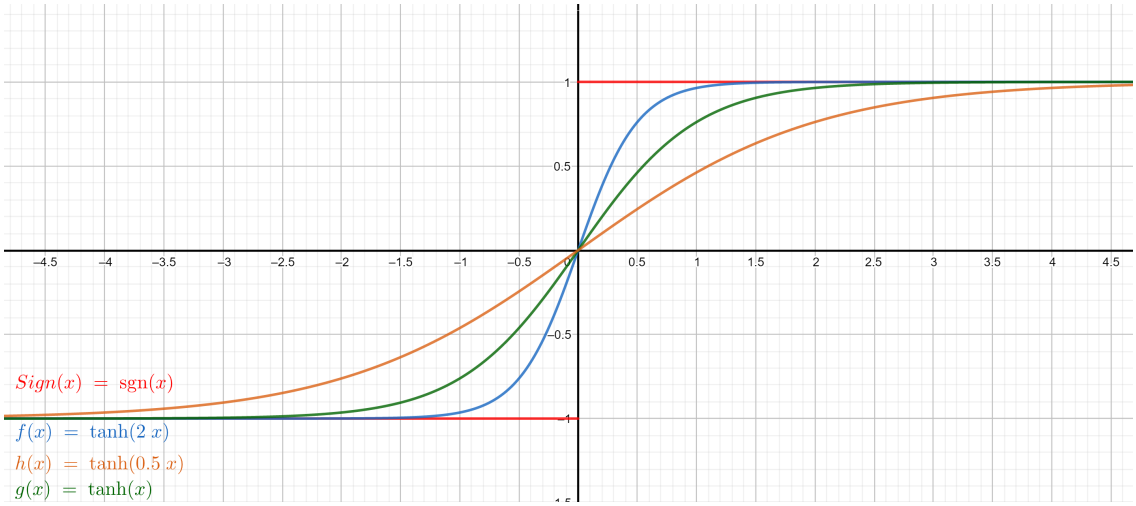

Figure 2.5: The graph of **sign** function and **tanh** function. The red line is sign function. And the blue, green, yellow line are tanh(2x), tanh(x) and tanh(0.5x) separately.

Learning to hash The aim of hash function is to convert continuous vectors into binary vectors. Learning to hash for indexing [43] is a method that rejects the exhaustive comparison and introduces approximate nearest neighbor (ANN) search. BPR uses sign function to generate binary code. Given continuous embedding $\mathbf{e} \in \mathbf{R}^d$ from a encoder, the sign hash function compute its binary hash vector, $\mathbf{h} \in \{-1, 1\}^d$, as:

$$\mathbf{h} = \text{sign}(\mathbf{e}) = \text{sign}(\{e_1, e_2, ..., e_d\}) \tag{2.10}$$

and

$$\text{sign}(e_i) = \begin{cases} 1 & \text{if } e_i > 0 \\ 0 & \text{else} \end{cases}, \tag{2.11}$$

where $i = 1, 2, 3, ..., d$. The problem of sign function is that in deep learning, it cannot be used in back propagation as shown in Fig 2.5. The gradient of sign function is zero when $x \neq 0$ and no gradient when $x = 0$. $Tanh$ function is alternative solutions [6] of sign function when training the model:

$$\mathbf{h} = \tanh(\beta \mathbf{e}), \tag{2.12}$$

where β is scaling parameter. Fig 2.5 shows how β influences the function. If $\beta \to \infty$, it converges to the sign function. During the training in BPR, β increases gradually at each training step:

$$\beta = \sqrt{\gamma \cdot \text{step} + 1}, \tag{2.13}$$

where *step* is the number of finished steps. γ is the coefficient to influence the training result with different value.

Different from DPR, BPR proposed a new loss function for binary vectors. For two binary vectors with the same length, if they are more similar to each other, they can have a larger inner product value. With the same training set as DPR, the loss function of BPR is to minimize the inner product of positive passage binary vector $\mathbf{h}_{p^+}$ and question binary vector $\mathbf{h}_q$:

$$\mathcal{L}_{\text{BPR}} = \sum_{j=1}^{n} \max\left(0, -\left(\langle \mathbf{h}_q, \mathbf{h}_{p^+} \rangle + \left\langle \mathbf{h}_q, \mathbf{h}_{p_j^-} \right\rangle\right) + 2\right), \tag{2.14}$$

where $\langle \cdot, \cdot \rangle$ is the inner product of the vector. $\mathbf{h}_{p_j^-}$ is one negative passage. There are n negative passages in total. BPR use the same way to select positive passages and negative passages as DPR.

Chapter 3

Preliminaries

In this chapter, basic background concepts and models mentioned in this thesis will be introduced and explained. We firstly introduce a popular model, Transformer, in the §3.1. In §3.2, we introduce a famous pre-trained model, BERT, based on the transformer. In order to further decrease the size of the passage retrieval model, we explain some deep learning models with smaller size in §3.3 and §3.4.

3.1 Transformer

Transformer [39] is a new network architecture based on the attention mechanism. Transformer uses self-attention to compute the representation of the input. Transformer is a encoder-decoder structure shown in Fig 3.1. Both encoder and decoder consist of $N = 6$ identical layers.

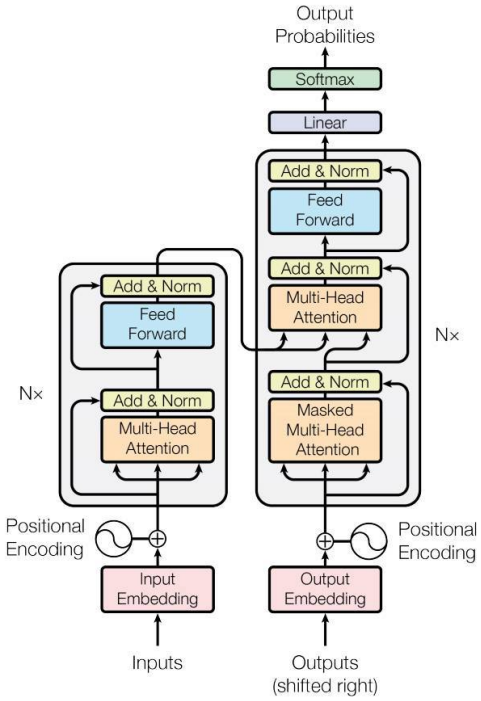

Figure 3.1: The encoder-decoder architecture in Transformer [39], the left part is encoder and the right part is decoder.

The encoder has two main parts: Muti-Head Attention and Feed Forward. In the muti-head attention module, it is the ensemble of several self-attention modules. The output of each self-attention module is concatenated together into a large matrix, and then processed by a fully connected layer. Fig 3.2 shows the relationship between self-attention and multi-head attention. In one multi-head attention module, there are eight different self-attention models with different

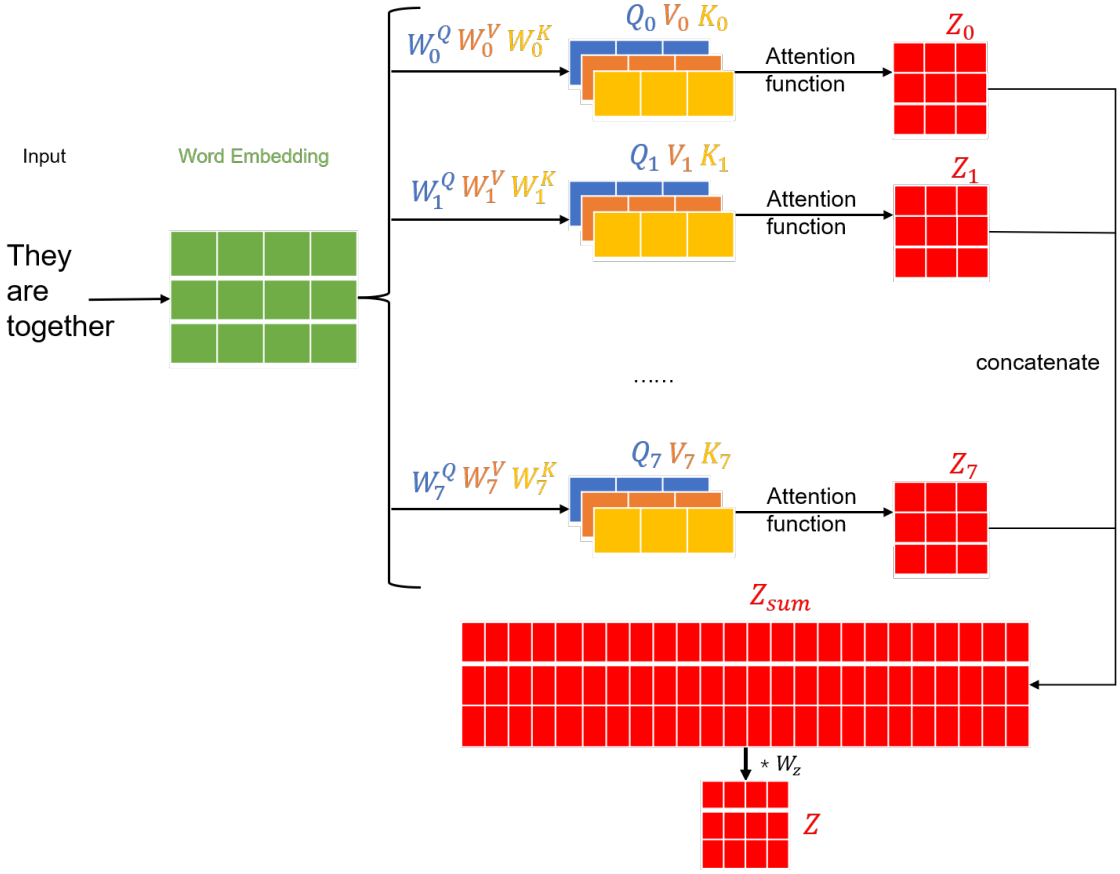

Figure 3.2: The structure of multi-head attention. Each self-attention generates one matrix and is concatenated together.

weight matrices. They can generate different Q, K, V matrices based on the weight. Processed by the attention function, each self-attention module produces one output Z_i. After concatenating all output from each self-attention, the Z_{sum} can be multiplied with the weight matrix W_z. We can get the output of the multi-head attention. In the self-attention module, the input is converted into a vector Z:

$$\mathrm{Z} = \mathrm{Attention}(Q, K, V) = \mathrm{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V, \tag{3.1}$$

where Q, K, and V are the query vector, key vector and value vector generated from the input. For each word in the input sentence, it is converted into a vector, whose length is 512. Meanwhile, there are three matrices, Value matrix, Query matrix and Key matrix. The shape of all these three matrices is $512 * 64$. By matrix multiplication, the input can be converted into three vectors with a length of 64. These three vectors are Value, Query and Key vectors.

After getting the vector Z, it is processed by feed forward module, including a ReLU function and a linear activation function. Same to the encoder, the decoder also includes muti-head attention and feed forward module. Besides that, it also contains an attention layer that helps the decoder focus on relevant parts of the input sentence.

As a seq2seq model, the transformer breaks the limitation that RNN models cannot be computed in parallel. Compared to CNN model, the number of operations to compute the association between two words in a sentence does not grow with the word distance. Self-attention mechanism can produce more interpretable models and each attention head can learn to perform different tasks. Transformer is always used in some pre-trained NLP models, like BERT, and applied in the ODQA domain.

3.2 BERT

Pre-trained models of language representations, like ELMo [31], are always used in downstream tasks. *Fine-tuning* is a strategy to apply pre-trained models in NLP tasks. Generative Pre-trained Transformer (OpenAI GPT) [32] uses fine-tuning approach to introduce minimal task-specific parameters. For downstream tasks, they only need to fine-tune all parameters by a small training dataset. In question-answering tasks, the major limitation is that most pre-trained models can only process tokens in one direction. For example, OpenAI GPT is a left-to-right architecture, in which every token can only get information from previous tokens in the self-attention layers of the Transformer [39]. Unlike these models, BERT enables the representation to fuse the left and the right context.

BERT consists of two parts, *pre-training* and *fine-tuning* as shown in Fig 3.3.

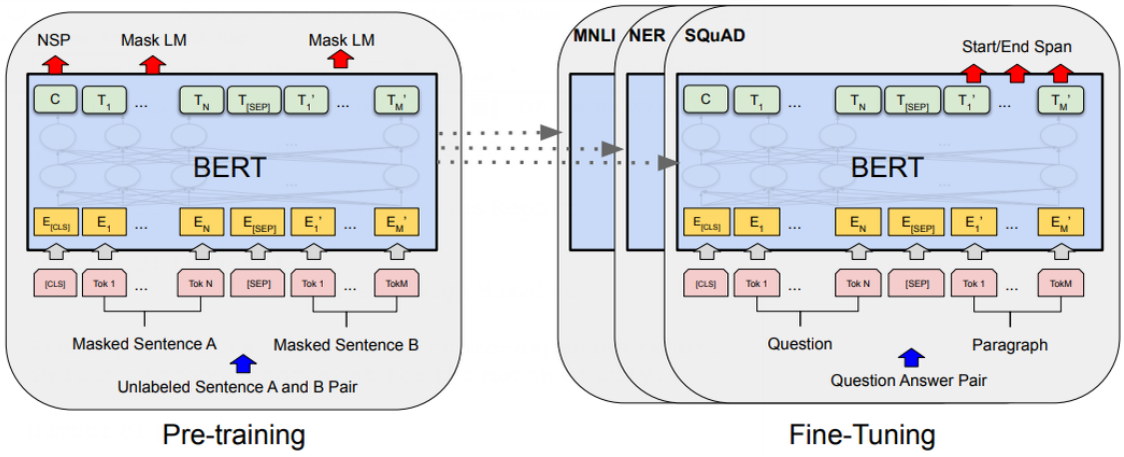

Figure 3.3: Overall pre-training and fine-tuning procedures for BERT [10].

Pre-training In the pre-training part, the model is training on several unlabeled datasets. BERT introduces a "masked language model" (MLM) in pre-training to achieve the bidirectional transformers. 15% words in the input sentences will be replaced with a special token [MASK] at random, and then the BERT model will predict the masked tokens. For better adaptation to downstream tasks, some words will not be replaced by [MASK] but by a random token.

For example, like shown in Fig 3.4, if we want to replace the word *hairy* in the sentence "My dog is hairy". In 80% of the cases the word token is replaced by [MASK]. In 10% of the cases the it is replaced by another token like *apple*. In the rest 10% cases, the token is not replaced. The loss function of MLM is defined in Eq.3.2:

$$L_1(\theta, \theta_1) = -\sum_{i=1}^{M} \log p(m = m_i \mid \theta, \theta_1), m_i \in [1, 2, \ldots, |V|], \tag{3.2}$$

where θ represents parameters in the BERT Encoder. θ_1 is the parameter in the output layer connected to the encoder in the MLM task. M is the word set of all masked words. V represents all words in the pre-training corpus.

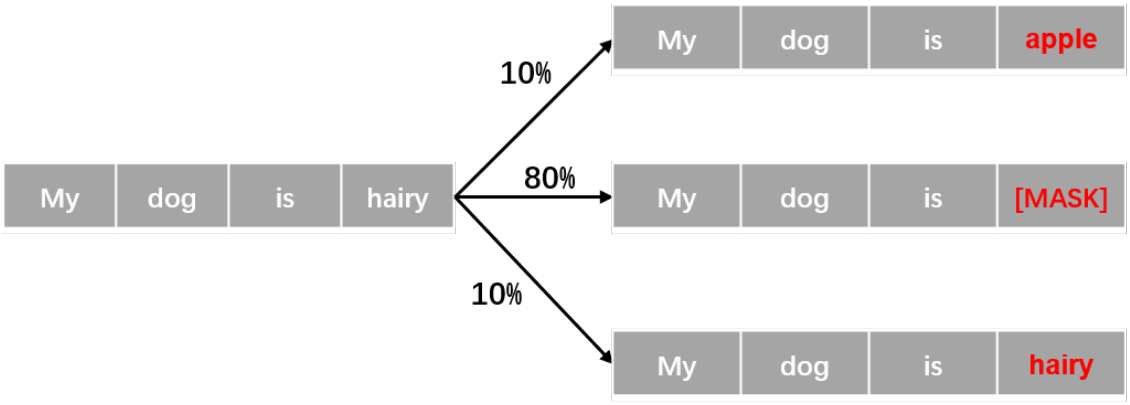

Figure 3.4: Three different ways when mask some words in BERT [10].

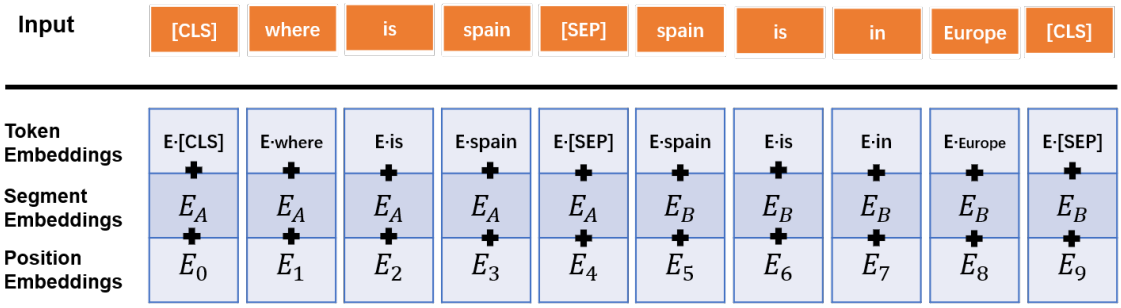

Figure 3.5: The architecture of input in BERT. Input vectors are the summation of token embeddings, segment embeddings and position embeddings [10].

Another task in pre-training is Next Sentence Prediction (NSP), which is closely related to representation-learning objectives [17,26]. For sentences, BERT can combine one or two sentences in one token sequence. The first token of every sequence is [CLS], a special classification token. And token [SEP] is used to separate two sentences. Meanwhile, BERT adds learned embedding for every token to show their position and segment information as shown in Fig 3.5. In pre-training example, 50% of the time sentence B is the actual next sentence following sentence A (labeled as *IsNext*), and the rest is a random sentence from the corpus (labeled as *NotNext*). The loss function of NSP is defined in Eq.3.3:

$$L_2(\theta, \theta_2) = -\sum_{i=1}^{N} \log p(n = n_i \mid \theta, \theta_2), n_i \in [\text{ IsNext, NotNext }], \tag{3.3}$$

where θ represents parameters in the BERT Encoder. θ_2 represents parameters in the output layer connected to the Encoder in the NSP task. N is the set of all sentences with the label between IsNext and NotNext. And the loss function of pre-training stage is the summation of Eq.3.2 and Eq.3.3:

$$L(\theta, \theta_1, \theta_2) = L_1(\theta, \theta_1) + L_2(\theta, \theta_2). \tag{3.4}$$

Fine-tuning In the fine-tuning part, we can use our dataset on the pre-trained BERT model to train a model that works on our own tasks. Non-finetuned baseline embeddings can perform well in a lot of tasks but fine-tuning can boost the performance. The self-attention [39] mechanism allows BERT to carry out several downstream tasks. Compared to pre-training, fine-tuning costs less time and resources.

3.3 DistilBERT

DistilBERT [37] is a small and cheap BERT model based on the Transformer architecture. It reduced the size of original BERT model by 40%. Compared with BERT, some layers, like token-type embedding and pooler are removed. The difference between DistilBERT and BERT is the number of transformer layers. BERT includes 12 layers to process tokenized text while DistilBERT includes 6 layers. In each transformer layer, it uses the self-attention mechanism. It can capture some syntactic features between words in the same sentence, even if the two words are located far apart. Compared with RNN [46] and LSTM [36], transformer is more likely to capture long-distance interdependent features in sentences. Multi-Head Attention computes multiple Attentions simultaneously and finally gets the merged result. It captures the relevant information about different subspaces by computing multiple times.

In DistilBERT, it uses knowledge distillation [5, 15], a compression method to reproduce the performance of a large model based on a compact model. The number of parameters in DistilBERT decreased from 110 million in BERT to 66 million. And the inference time of a full pass of GLUE task STS-B [42] on DistilBERT and BERT is 410 seconds and 668 seconds.

3.4 ElectraSmall

Inspired by Generative adversarial networks [9]. Electra [8] uses a new pre-training task, called Replaced Token Detection (RTD), that is similar to BERT but highly efficient. In BERT, MLM task replaces some words and tries to train a model to predict these masked words. And in Electra, masked words will be replaced by other incorrect, but somewhat plausible, fakes. In Fig 3.6, two words are replaced, and Electra is able to distinguish which words are replaced. Given an input sentence $\mathbf{x} = [x_1, x_2, ..., x_n]$, 15% words are masked out by MLM with a `[MASK]` token.

Model	Layers	Hidden Size	Params
ElectraSmall	12	256	14M
ElectraBase	12	768	110M
ElectraLarge	24	1024	335M

Table 3.1: The details of three pre-trained models.

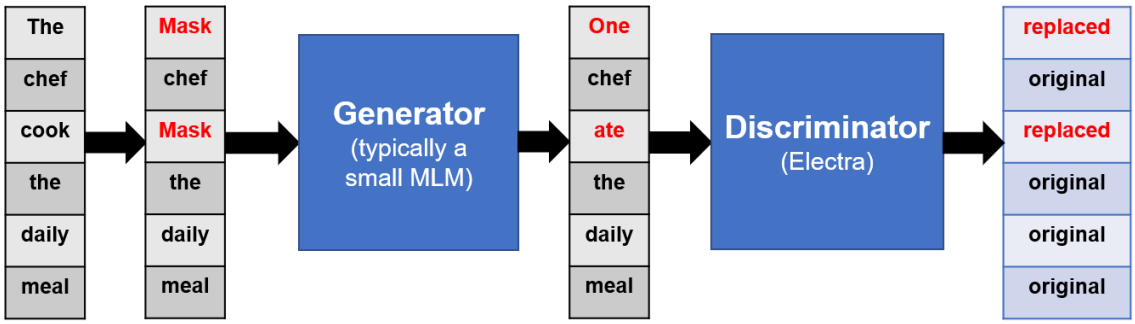

Figure 3.6: Details on the replaced token detection (RTD) task. The fake tokens are sampled from a small masked language model that is trained jointly with Electra.

After pre-training, the Generator will be dropped and the discriminator (the Electra model) is fine-tuned on downstream tasks. This modification can make the pre-processing faster and can be used in downstream tasks. There are three versions of Electra model, Small, Base and Large shown in Table 3.1. Among them, ElectraSmall performs quite well on a small training dataset, even outperforming GPT while requiring only 1/30 as much compute.

Chapter 4

Methodology

In this chapter, the framework of SmallBPR is described in detail. We aim to further reduce the model size of BPR. We proposed two ways to decrease the size of models as shown in Figure 4.1. Firstly, encoders for passages and questions can be shared. In this way, the size of model can be reduced by half. Then we further replace the original BERT model with other smaller, cheaper and faster models. These two methods are introduced separately in §4.1 and §4.2. And §4.3 is about multi-tasks to improve the performance of SmallBPR in the training stage, and we also introduce the detail of two loss functions in this section. §4.4 introduces how we select relative passages in two steps.

Similar to BPR, we have the encoder for translating passages and questions into vectors and then use hash function to translate them into binary vectors. The difference is that we design a shared encoder for both passages and questions. We also use some smaller pre-trained models instead of the BERT model.

4.1 Encoder Sharing

Inspired by previous research [45], combining encoders does not influence the model's performance significantly while the size can decrease a lot. The size of each BERT encoder in BPR is around 1.2GB. Let $E_Q(\cdot)$ and $E_P(\cdot)$ be the question encoder and passage encoder. We share all the parameters of two encoders:

$$E_Q = E_P = E_\theta, \tag{4.1}$$

where E_θ is the shared encoder. The same strategy is used to convert continuous vectors into binary vectors. Both continuous vectors of passages and questions are translate in to binary vector with the same hash function. A parameter *Max_length* is set to limit the maximum length of continuous vectors. If the length of vector is longer than *Max_length*, the part beyond *Max_length* will be cut.

4.2 Using small models

We try to use small pre-trained models as encoders, such as DistilBERT [37] and ElectraSmall [8]. BPR uses BERT [10] as encoders. Table 4.1 shows the details of three models. We can see, compared with BERT, the number of parameters and model size of another two models decreased a lot.

DistilBERT [37] is a small and cheap Transformer model based on the BERT architecture. It reduces the size of original BERT model by 40%. Compared with BERT, some layers, like token-type embedding and pooler are removed.

ElectraSmall is based on Electra [8] that is a method for self-supervised language representation learning. Electra provides several versions in different sizes. ElectraSmall is a small version

of Electra model with fewer parameters. Compared with BERT, ElectraSmall has a shorter sequence length (128), smaller hidden dimension size (256) and smaller token embeddings (128). For ElectraSmall, its size is only one-eighth of that of BERT model.

Model	#parameter	model size
BERT	110 million	1.2 GB
DistilBERT	66 million	760 MB
ElectraSmall	13 million	155 MB

Table 4.1: The number of parameter and the model size of different models.

4.3 Training

Like DPR, we also use a a training dataset including positive and negative passages. Let $\mathcal{D} = \left\{\left\langle q_i, p_i^+, p_{i,1}^-, \cdots, p_{i,n}^-\right\rangle\right\}_{i=1}^m$ be m training instances, where q_i is a question, p_i^+ is a positive passage related to the question q_i. And $p_{i,1}^-$ to $p_{i,n}^-$ are n negative passages irrelevant for the question. Generally, we only need one negative instance in a training set. In practice, a number of questions correspond to several positive passages. We only choose one positive passage with the highest score from the positive passage set.

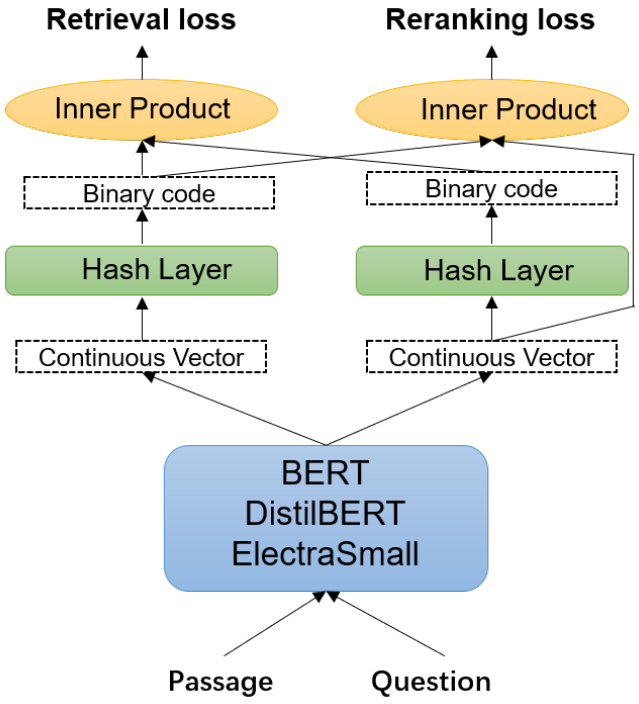

Figure 4.1: The architecture of SmallBPR, the encoder part can be BERT, DistilBERT or ElectraSmall.

When the system processes the input query, the question can be converted into both the continuous vector and binary vector, which can be used to compare with the passage index (only binary vector). In order to improve the model performance, we borrow the method from BPR and DPR and make a multi-tasks with two parts: `Retrieval` and `Reranking`. We design two loss functions for the Retrieval task and the Reranking task separately, and the final loss function is the sum of these two loss functions. Fig 4.1 shows the structure of SmallBPR. Passage and

question are converted by the same encoder. And we use the inner product to calculate Retrieval loss and Reranking loss.

Following the method in BPR, we use the hyperbolic tangent function as the hash function to convert continuous embedding to the embedding $\mathbf{h}_q$. And then embedding $\mathbf{h}_q$ can be transferred into the binary vector. All elements in $\mathbf{h}_q$ are between -1 and 1. By setting a coefficient β, we can control the effect of hyperbolic tangent function:

$$\mathbf{h}_q = tanh\left(\beta \mathbf{e}_q\right) = \{h_1, h_2, ..., h_d\}, \tag{4.2}$$

where $\mathbf{e}_q$ is the continuous embedding of the question. As the coefficient β keeps increasing, the hyperbolic tangent function converges to the sign function. And β can increase with the training step. We only need to set an appropriate value for coefficient γ in the experiment:

$$\beta = \sqrt{\gamma \cdot \text{step} + 1}, \tag{4.3}$$

where *step* means the number of the training step. After that, we convert the embedding $\mathbf{h}_q$ to the approximate binary hash vector $\tilde{\mathbf{h}}_q$:

$$\tilde{\mathbf{h}}_q = \left\{\tilde{h}_1, \tilde{h}_2, ..., \tilde{h}_d\right\} \tag{4.4}$$

and

$$\mathbf{h}_q = \{h_1, h_2, ..., h_d\}, \tag{4.5}$$

where $\tilde{h}_i$ is the element in the approximate binary vector $\tilde{\mathbf{h}}_q$ and h_i is the element in the binary vector $\mathbf{h}_q$. All elements in $\mathbf{h}_q$ which are smaller than 0 are set into -1 and others are set into 1:

$$\tilde{h}_i = \begin{cases} -1 & \text{if } h_i < 0 \\ 1 & \text{else} \end{cases}. \tag{4.6}$$

We use the approximate binary vector $\tilde{\mathbf{h}}_q$ in the Retrieval task and the Reranking task to calculate the loss.

4.3.1 Retrieval

In Retrieval part, the objective of this task is to improve candidate generation using the ranking loss. We try to minimize the loss with the approximated hash vector of question and that of passages:

$$\mathcal{L}_{\text{retrieval}} = \sum_{j=1}^{n} \max\left(0, -\left(\left\langle \tilde{\mathbf{h}}_{q_i}, \tilde{\mathbf{h}}_{p_i^+} \right\rangle + \left\langle \tilde{\mathbf{h}}_{q_i}, \tilde{\mathbf{h}}_{p_{i,j}^-} \right\rangle\right) + 2\right), \tag{4.7}$$

where $\tilde{\mathbf{h}}_{q_i}$ and $\tilde{\mathbf{h}}_{p_i^+}$ are the approximated hash vector of questions and positive passages. $\tilde{\mathbf{h}}_{p_{i,j}^-}$ is one of the approximated hash vector of negative passages. The greater the inner product between two binary vectors represents higher similarity between the question and the passages.

4.3.2 Reranking

In this part, we use the binary vector of passages and the continuous embedding of questions to calculate inner product. The reason we use continuous embedding in Reranking part is that compared with the binary vector, the continuous embedding contains more information of its context. And we keep both the binary vector and the continuous vector the same length of 756, so that it is very easy to perform some operations directly on these two vectors.

For Reranking, we train the model by minimizing the negative log-likelihood of the positive passage. $\mathcal{L}_{reranking}$ models the correlation between passages and questions using binary vectors and continuous embedding:

$$\mathcal{L}_{reranking} = -\log \frac{\exp\left(\left\langle \mathbf{e}_{q_i}, \tilde{\mathbf{h}}_{p_i^+} \right\rangle\right)}{\exp\left(\left\langle \mathbf{e}_{q_i}, \tilde{\mathbf{h}}_{p_i^+} \right\rangle\right) + \sum_{j=1}^{n} \exp\left(\left\langle \mathbf{e}_{q_i}, \tilde{\mathbf{h}}_{p_{i,j}^-} \right\rangle\right)}, \tag{4.8}$$

where $\mathbf{e}_{q_i}$ is the continuous embedding of the question. We combine $\mathcal{L}_{reranking}$ and $\mathcal{L}_{retrieval}$ as the loss function in SmallBPR training process:

$$\mathcal{L} = \mathcal{L}_{\text{retrieval}} + \mathcal{L}_{\text{reranking}}. \tag{4.9}$$

4.4 Passage Selection

After training the model, we use the test data in NQ and TQA to evaluate the performance of our model. For each question, we will select top-L passages with the highest score from each model, and calculate the recall of Top 1, 20, 100. Following muti-tasks, the process of selecting a passage with the right answer requires two steps. Different from the training step, here we no longer use the hyperbolic tangent function to replace the sign function because there is no back-propagation part anymore. Sign function can be used directly to convert the continuous embedding into the binary vector $\mathbf{h} = \text{sign}(\mathbf{e})$, where $\mathbf{e}$ is the continuous embedding and $\mathbf{h}$ is the binary vector processed by hash function.

Retrieval In Retrieval part, we compare the binary vector of passages and questions to select top-L passages with the highest inner product value. The length of continuous embedding and binary vector is the same, we can get a score by inner product. If the passage and the question contain similar content, the product will be higher. The top-L messages with the highest score can be sent to the next stage.

Reranking After get top-L passages, they will be reranked by inner product. In this part, we use the binary vector of passages and the continuous embedding of questions to calculate inner product. The reason we use continuous embedding in Reranking part is that compared with the binary vector, the continuous embedding contains more information about its context. For every bit in the vector, the binary vector can only store information with 0 and 1, but the continuous embedding can store information from 0 to 9. Thus, using continuous embedding to rank candidates is more accurate.

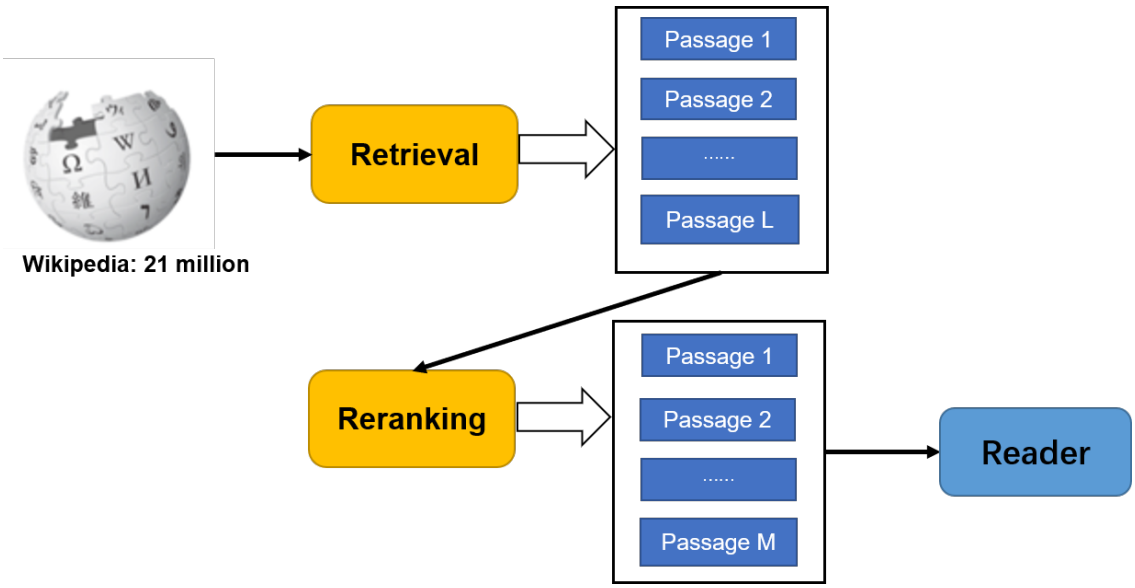

Figure 4.2: The process of SmallBPR selecting top-m passages in two steps and sending passages to the Reader part.

Fig 4.2 shows the way how SmallBPR selects passages from wikipedia corpus and sends top-M passages to the Reader part. Related passages can be filtered in two steps. In the first step, we filter out most irrelevant passages, and in the second step, we will select the best M passages from

the remaining passages. Generally, the number of L is less than 10 thousands, and the number of M is less than 100.

The reason we generate candidates in two stages is to save time and not reduce the accuracy. If we drop the Reranking task, continuous embeddings are not used anymore. Too little information contained in the binary vector can lead to the inaccurate ranking and the accuracy must decrease. If we discard the Retrieval task, choosing top-M messages directly from 21 million passages would cost too much searching time. Retrieval part can filter most passages that are obviously not relevant to the given question. Then we can make a more detailed filter in Reranking part to select Top-M passages.

In this paper, we set $L = 1000$. Theoretically, accuracy improves with L increasing, but the searching time also increases. And after L increases to a certain threshold, the accuracy will increase slowly or stop increasing. We do a series of the same experiments with different value of L, and find that if L is larger than 1000, the accuracy will not keep increasing any more. Thus we set $L = 1000$ in our experiments. The detail wis shown in §5.2.

Selecting 1000 passages from 21 million passages also costs a lot of time. We use Facebook AI Similarity Search (Faiss), a library developed by Facebook AI research, allowing to quickly search for vectors that are similar to each other [18]. It is written in C++ and has wrapper code for python2 and python3. Faiss provide several retrieval methods and can be run only GPU, which can save a lot of time. After generating indices for each question and passage, all passage vector will be stored in the Faiss, and vectors that are very similar to the question vector can be found quickly.

After obtaining potential candidates, the next step is to verify whether these passages can be used to answer the given question. In the test sets, each question is corresponding to one or two answers. If the answer fields are included in the passage candidates, this passage is considered to be correct and can be used in the Reader part to extract answer. Not all 1000 passages need to be used in the next stage, so we only calculate the recall of top-1, 20, 100 passages for a question in the experiments.

Chapter 5

Experiments

In this section, we introduce our experiment steps and results in detail. In §5.1, we introduce two dataset we used and how we do pre-processing on them. In §5.2, some important parameters are listed. The performance of different models and some ablation experiment are explained in §5.3 and §5.4.

5.1 Dataset

We conduct the experiments on Natural Question (NQ) dataset [22] and TriviaQA (TQA) dataset [19]. NQ dataset is about question answering which includes real aggregated queries issued to the Google search engine. TriviaQA is a dataset containing relatively complex combinatorial questions.

5.1.1 Pre-processing

Because only pairs of questions and answers are provided in TriviaQA, we follow the instruction in DPR, using the highest-ranked passage from BM25 [35] that contains the answer as the positive passage. We get top 100 retrieved passages, and if the passage contains the right answer, the passage will be treated as a positive passages. If the whole 100 passages do not contain right answer,the question will be discarded. After pre-processing, we get train, development and test data set of NQ and TQA separately. Table 5.1 shows the number of triple and the file size of different sets. In one triple, there are one question, several right question, positive and negative passages. In the test set, the number of passages is 0 and only questions and answers included in the set.

Dataset	Natural Question		Trivia QA	
	#Triple	size	#Triple	size
train-set	58800	6.86GB	78785	6.13GB
dev-set	8837	777MB	6515	705MB
test-set	3610	290KB	11312	3.9MB

Table 5.1: The number of triples(Question, Answer, Passages) and the size for different datasets.

5.1.2 Wikipedia Corpus

We use the English Wikipedia [23] corpus as the source documents for answering questions. We also do pre-processing on the English Wikipedia dump to delete figures, tables and other non-text parts using the method introduced in previous research [7]. Considering the excessive number

of words in one passages, a passage is split into several sub-passages with the same title. After pre-processing, there are 21,015,324 passages and titles.

5.2 Setup

Our implementation is based on BPR. We use two pretrained models, DistilBERT [37] and ElectraSmall [8], for our shared encoder. The two models have less parameters and smaller size than BERT. Specifically, we consider following models:

- SmallBPR-BERT: it uses BERT as the shared encoder. BERT has 110 million parameters.
- SmallBPR-DistilBERT: it uses DistilBERT [37] as the shared encoder. There are 66 million parameters in DistilBERT, retaining 97% language understanding capabilities of the BERT model.
- SmallBPR-ElectraSmall: it uses a small version ElectraSmall [8] of Electra model as the shared encoder. ElectraSmall has 15 million parameters and good performance on small datasets.

Before we conduct the main experiment, selecting an appropriate value of L is very important. In this research, we first select L passages in Retrieval part and rerank these L passages. If the value of L is too large, it will cost too much time to search and rerank. If the value of L is too small, only a few passages can be used in Reranking part, the precision will decrease. Thus, selecting a fitting value of L is very important. We conduct a preliminary experiments, setting different value of L on NQ and TQA dataset. We use the recall of top-100, search time as the metrics. We set L =100, 200, 500, 800, 1000, 1200 and compare the result. We want to keep the search time as low as possible while maintaining the recall of top-100.

As we can see, when $L \leq 1000$, the number of correct sample keeps increasing, especially when $L \leq 500$. Significant slowdown in increase when $L \geq 800$. In NQ dataset, the correct number increases only one when $L \geq 500$. Same to TQA dataset, the precision only increases 0.04% when $L \geq 1000$. Thus, we set $L = 1000$ to conduct experiments.

	Metrics	L=100	L=200	L=500	L=800	L=1000	L=1200
NQ	#Total sample	3610	3610	3610	3610	3610	3610
	#Correct sample	2969	2992	3002	3003	3004	3004
	Precision	82.24%	82.88%	83.16%	83.19%	83.21%	83.21%
	Search time	988s	990s	991s	995s	1011s	1030s
TQA	#Total sample	11313	11313	11313	11313	11313	11313
	#Correct sample	9280	9365	9405	9409	9411	9416
	Precision	82.03%	82.78%	83.13%	83.17%	83.19%	83.23%
	Search time	3154s	3170s	3197s	3185s	3203s	3218s

Table 5.2: The result of preliminary experiment: precision and search time with different value of L on NQ and TQA dataset with **SmallBPR-BERT** model.

For the Retrieval part, we use binary code for search. The passage index is built using Faiss [18]. For the Reranking part, we use expressive continuous embeddings for $L = 1000$ top candidate passages. All of our experiments are run on a single Tesla V100-SXM2 GPU. More details of hyperparameters used in training SmallBPR is shown in Table 5.3.

Metrics. We evaluate our model based on several metrics: recall, model size, query time of the retrieval system. The top-k recall is the percentage of positive passages in the top-k passages. Positive passages are the ones that include answer fields in the context. Model size shows the size of different NLP models. Query time records the total time when querying 1,000 questions with the Wikipedia corpus.

Hyperparameter	Value
top-l	1000
batch size	16
Max epoch	35
learning rate	2e-5
drop-out	0.1
hash function γ	0.1
max passage length	256
max question length	256

Table 5.3: The important hyperparameters and its value used in training process.

5.3 Main Results and Analysis

Table 5.4 illustrates the performance in terms of recall of top-k ($k \in \{1, 20, 100\}$) and the model size of different models.

Compared with the BPR model, SmallBPR model with the single BERT encoder halved in size and slightly decreased recall. For top 1, 20 and 100 in both NQ and TQA dataset, the decrease of SmallBPR-BERT is less than 3 percent.

SmallBPR-DistilBERT has a much smaller size, 760MB, but its top 100 is almost the same as SmallBPR-BERT. On NQ and TQA dataset the decrease is less than 0.3 percent comparing with SmallBPR-BERT. In practical applications, top 1 recall is the least important because the reader in the QA system usually generate the answer based on more than 20 passage candidates.

We also try ElectraSmall, which is only 154MB. With such a small size, its top 100 recall performance on NQ dataset reaches 66 percent. Meanwhile, as the model size increases, the query time also increases. The query time of SmallBPR-BERT and SmallBPR-DistillBERT is about 65 second faster than BPR.

Model		SmallBPR-ElectraSmall	SmallBPR-DistilBERT	SmallBPR-BERT	BPR
Model Size		154MB	760MB	1.2GB	2.4GB
Query Time		77s	308s	309s	373s
Natural Question	Top 1	15.9%	36.3%	38.3%	41.1%
	Top 20	49.8%	73.8%	74.9%	77.9%
	Top 100	66.1%	83.1%	83.2%	85.7%
Trivia QA	Top 1	10.1%	45.5%	46.6%	49.7%
	Top 20	38.4%	75.3%	75.5%	77.9%
	Top 100	57.5%	82.9%	83.2%	84.5%

Table 5.4: Top 1, 20, 100 recall, query time per 1,000 questions and model size for different models on NQ and TQA.

5.3.1 Top 5 passages

In order to more visually compare the differences between these models. We manually evaluated the selected passages from NQ dataset. Table 5.5, Table 5.6 and Table5.7, illustrate examples of Top 5 passages returned by different models, which is trained by NQ dataset.

We select a question *When is the next deadpool movie being released* and the answer is *May 18, 2018*. Only when the text in the passage completely matches the answer, this passage is considered as a positive passage. There are some important key words in the question.

- **When**: This interrogative adverbs shows that the answer should be about date or time.

- **Deadpool**: Deadpool is the name of the movie, so the passage should be related to this movie.
- **Next**: This is an important key word. It hints that the passage should show the information of deadpool 2 but not deadpool 1.

In positive passages, they contain the right answer to the question. In Top 5 passages from different models, there are 1 positive passage in SmallBPR-ElectraSmall, 4 positive passages in SmallBPR-DistilBERT and 2 positive passages in SmallBPR-BERT. All the passages in Top 5 fully understand the questions and give answers in addition to the ones about Deadpool released information.

However, negative passages have some problems with some details. For example, No.1 in SmallBPR-ElectraSmall is a negative passage and it mainly focuses on "re-release" but not "next release" mentioned in the question. No.2 in SmallBPR-ElectraSmall is also a negative passage, it misses the key word "next" in the question. All the information in this passage is about Deadpool 1. Some passages, like No.4 in SmallBPR-DistilBERT, correctly contain all key information in the passage, but they do not provide an exact answer. It only tells the released month but not the exact date in the passage. And different models can select the same passages from 21,015,324 passages in the Wikipedia corpus. One positive passage is selected by these three models and rank No.3 in SmallBPR-ElectraSmall, No.4 in SmallBPR-DistilBERT and No.3 SmallBPR-BERT.

5.3.2 Semantic Analysis

In order to test the model's ability to understand the semantic meaning of sentences, we provide a pair of questions that contain the same keyword, but with different word meanings in Table 5.8 and 5.9. The first question is *who is the chair man* and the second is *which man invented the chair*. In this two questions, the key word of the question is all about *Chair* but this word has different meanings. In the first question, "chair man" is a noun phrase meaning the leader of a organization. In the second question, "chair" means a kind of furniture. In Table 5.8, all three models understand the question semantic and provide reasonable answers.

In Table 5.8, for the first question, SmallBPR-BERT gives an answer about university president in the United States. SmallBPR-DistilBERT introduces president in Washington D. C. SmallBPR-ElectraSmall introduces the president of Towson State University. We can see all three models know that the key word "chair man" means president and provide relevant passages. In the second question, three models knows that key word "chair" is a kind of furniture. SmallBPR-BERT and SmallBPR-DistilBERT provide the same passage, introducing some information about the first compactly stackable chair, invented by David Rowland in 1924. SmallBPR-ElectraSmall provided information on three of the more popular chairs in England and France.

Not all models provide passages with high relevance yet. In Table 5.9, SmallBPR-ElectraSmall trained by TQA dataset provides a passage about Pistol, which has absolutely no connection to the word *Chair*. Models with more parameters and complex structures can understand the meaning of one questions much better.

Model	SmallBPR-ElectraSmall	
Q&A	When is the next deadpool movie being released? –May 18, 2018	
Rank	Passsage	Has Answer
No.1	another by Reynolds, Reese, and Wernick. On November 7, Fox re-released the film and its special features on Blu-ray for holiday season, as "Deadpool's Holiday Blu-ray package". The film was re-released again in April 2018 in a "Deadpool" Two Year Anniversary Edition Blu-ray package, with collectiblecovers as well as "stickers, car decals, temporary tattoos and a set of paper dolls". A 4K UHD Steelbook version was also released exclusively through Best Buy featuring original artwork. "Deadpool" grossed $363.1 million in the United States and Canada and $420 million in other countries for a worldwide total of $783.1 million	False
No.2	Screen Rant called it possibly "the best film marketing campaign in the history of cinema". HostGator's Jeremy Jensen attributed the campaign's success to Reynolds, and to Fox for embracing the film's R rating. "Deadpool"s world premiere was held at the Grand Rex in Paris on February 8, 2016, before its initial theatrical release in Hong Kong the next day. This was followed by releases in 49 other markets over the next few days, including the United States on February 12. The movie was released in several formats, including IMAX, DLP, premium large formats, and D-Box.	False
No.3	being blocked in China during this initial release, "Deadpool" eventually premiered in the country during the 2018 Beijing International Film Festival, which ran over a week from April 15–22. The original version of the film played at the festival without any edits being made specifically for Chinese censors. "Deadpool" was released for digital download on April 26, 2016, moved up from the physical home media release, which came on May 10. The latter release, for Blu-ray and DVD, included behind-the-scenes featurettes, deleted scenes, a blooper reel, and two audio commentaries: one by Tim Miller and Deadpool co-creator Liefeld	False
No.4	was written by Reese and Wernick and played in front of "Logan". "Deadpool 2" was released on May 18, 2018, with Baccarin, T. J. Miller, Uggams, Hildebrand, and Kapičić all returning. Josh Brolin joined them as Cable. The film explores the team X-Force, which includes Deadpool and Cable. In March 2017, Reese said that a future film focused on that group would be separate from "Deadpool 3", "so I think we'll be able to take two paths. [X-Force] is where we're launching something bigger, but then [Deadpool 3] is where we're contracting and staying personal and small." After the acquisition	True
No.5	May of that year. Several vendors provided visual effects for the film, ranging from the addition of blood and gore to the creation of the CG character Colossus. "Deadpool" was released in the United States on February 12, 2016, after an unconventional marketing campaign. The film achieved both financial and critical success. It earned over $783 million against a $58 million budget, breaking numerous records: it became the highest-grossing R-rated film, the highest-grossing "X-Men" film, and the ninth-highest-grossing 2016 film. Critics praised Reynolds' performance, the film's style and faithfulness to the comics, and its action sequences. Some detractors criticized the	False

Table 5.5: Top 5 passages generated by SmallBPR-ElectraSmall trained by NQ dataset.

Model	SmallBPR-DistilBERT	
Q&A	When is the next deadpool movie being released? –May 18, 2018	
Rank	Passsage	Has Answer
No.1	chimichangas, traditionally Deadpool's favorite food, as well as "Deadpool"-inspired Harder drinks. The campaign also included the chance to win a trip to the film's premiere through Harder, and the Los Angeles pop-up benefitted the nonprofit DTLA Film Festival. "Deadpool 2" premiered at Leicester Square in London on May 10, 2018. It was released in the United States on May 18, 2018, having been previously scheduled for release on June 1 of that year. Leitch's initial cut of the film was around two hours and twelve minutes, with "nips and tucks" done to it to get the run time down to	True
No.2	January, the film's release was moved up to May 18, 2018. In February 2018, Terry Crews was revealed to have a role in the film, the character Shatterstar was confirmed to be appearing, and the production returned to Vancouver for six days of reshoots under a new working title, "Daisy". Some reports emerged by mid-March claiming that these reshoots were due to poor audience responses during test screenings of the film, and consisted of sweeping changes. However, the film was soon confirmed to be testing better than the original did, up to 98 out of 100 over three different tests	True
No.3	was written by Reese and Wernick and played in front of "Logan". "Deadpool 2" was released on May 18, 2018, with Baccarin, T. J. Miller, Uggams, Hildebrand, and Kapičić all returning. Josh Brolin joined them as Cable. The film explores the team X-Force, which includes Deadpool and Cable. In March 2017, Reese said that a future film focused on that group would be separate from "Deadpool 3", "so I think we'll be able to take two paths. [X-Force] is where we're launching something bigger, but then [Deadpool 3]is where we're contracting and staying personal and small." After the acquisition	True
No.4	plot as formulaic as well as the sheer number of jokes in the film. It also received many awards and nominations, including two Critics' Choice Awards and two Golden Globe nominations. A sequel, "Deadpool 2", was released in May 2018. Wade Wilson is a dishonorably discharged special forces operative working as a mercenary when he meets Vanessa, a prostitute. They become romantically involved, and a year later she accepts his marriage proposal. Wilson is diagnosed with terminal cancer, and leaves Vanessa without warning so she will not have to watch him die. A mysterious recruiter approaches Wilson, offering an experimental	False
No.5	is dedicated to her memory. The film's score is the first to receive a parental advisory warning for explicit content, and the soundtrack also includes the original song "Ashes" by Céline Dion. "Deadpool 2" was released in the United States on May 18, 2018. It has grossed over $738 million worldwide, becoming the seventh highest-grossing film of 2018, as well as the third highest-grossing R-rated film and the third highest-grossing "X-Men" film. It received positive reviews from critics, who praised its humor, acting (particularly Reynolds, Brolin, and Beetz's performances), story, and action sequences, with some calling it	True

Table 5.6: Top 5 passages generated by SmallBPR-DistilBERT trained by NQ dataset.

Model	SmallBPR-BERT	
Q&A	When is the next deadpool movie being released? –May 18, 2018	
Rank	Passsage	Has Answer
No.1	May of that year. Several vendors provided visual effects for the film, ranging from the addition of blood and gore to the creation of the CG character Colossus. “Deadpool” was released in the United States on February 12, 2016, after an unconventional marketing campaign. The film achieved both financial and critical success. It earned over $783 million against a $58 million budget, breaking numerous records: it became the highest-grossing R-rated film, the highest-grossing “X-Men” film, and the ninth-highest-grossing 2016 film. Critics praised Reynolds’ performance, the film’s style and faithfulness to the comics, and its action sequences. Some detractors criticized the	False
No.2	is dedicated to her memory. The film’s score is the first to receive a parental advisory warning for explicit content, and the soundtrack also includes the original song “Ashes” by Céline Dion. “Deadpool 2” was released in the United States on May 18, 2018. It has grossed over $738 million worldwide, becoming the seventh highest-grossing film of 2018, as well as the third highest-grossing R-rated film and the third highest-grossing “X-Men” film. It received positive reviews from critics, who praised its humor, acting (particularly Reynolds, Brolin, and Beetz’s performances), story, and action sequences, with some calling it better	True
No.3	release date for “Dark Phoenix” to August 2, 2019; the reshoots required for film were believed to be more extensive than previously considered, with the studio now wanting at least half of the film to be reshot. The studio’s focus was on making the film as distinct from the other entries in the series as “Deadpool” and “Logan”, while avoiding the film becoming a “flop” like “Fantastic Four” which went through similar production issues—industry insiders believed “The New Mutants” would not end up the same way due to Fox not blaming the issues on Boone as the studio did	False
No.4	was written by Reese and Wernick and played in front of “Logan”. “Deadpool 2” was released on May 18, 2018, with Baccarin, T. J. Miller, Uggams, Hildebrand, and Kapičić all returning. Josh Brolin joined them as Cable. The film explores the team X-Force, which includes Deadpool and Cable. In March 2017, Reese said that a future film focused on that group would be separate from “Deadpool 3”, “so I think we’ll be able to take two paths. “X-Force” is where we’re launching something bigger, but then “Deadpool 3” is where we’re contracting and staying personal and small.” After the acquisition	True
No.5	Screen Rant called it possibly “the best film marketing campaign in the history of cinema”. HostGator’s Jeremy Jensen attributed the campaign’s success to Reynolds, and to Fox for embracing the film’s R rating. “Deadpool”s world premiere was held at the Grand Rex in Paris on February 8, 2016, before its initial theatrical release in Hong Kong next day. This was followed by releases in 49 other markets over the next few days, including the United States on February 12. The movie was released in several formats, including IMAX, DLP, premium large formats, and D-Box. Kinberg explained that unlike the	False

Table 5.7: Top 5 passages generated by SmallBPR-BERT trained by NQ dataset.

Question	Who is the chair man?	Which man invented the chair
SmallBPR-BERT	a total compensation of $4,270,665 in his final year of service, the second highest of any university president in the United States. Ground was broken for the new School of International Service building on November 14, 2007, and completed in 2010. A speech was given by Senator Daniel K. Inouye (D-HI). In 2015, American became one of the few universities to offer an accredited, accelerated online MBA program. Neil Kerwin retired as AU's president at the end of May 2017. The current president is Sylvia Mathews Burwell, whose tenure officially began on June 1, 2017.	David Rowland David Lincoln Rowland (February 12, 1924 – August 13, 2010) was an American industrial designer noted for inventing the 40/4 Chair. The chair was the first compactly stackable chair invented, and is able to stack 40 chairs high. David Lincoln Rowland was born on February 12, 1924, in Los Angeles, the only child of Neva Chilberg Rowland, a violinist and W. Earl Rowland, an artist, lecturer and teacher. In 1936 he moved with his parents to Stockton, California where his father became director of the Haggin Museum.
SmallBPR-DistilBERT	look at Washington, D. C. The program is unique in that the courses are not typical lecture courses; instead, speakers from various sectors of a particular field are invited to address the class, often from different perspectives. In the "Chronicle of Higher Education" survey of college presidentsśalaries for 2007–08, President Cornelius M. Kerwin was fifth highest in the nation with a compensation of $1.4 million. On September 24, 2018 AU President Sylvia M. Burwell announced the formation of the Sine Institute of Policy and Politics.	David Rowland David Lincoln Rowland (February 12, 1924 – August 13, 2010) was an American industrial designer noted for inventing the 40/4 Chair. The chair was the first compactly stackable chair invented, and is able to stack 40 chairs high. David Lincoln Rowland was born on February 12, 1924, in Los Angeles, the only child of Neva Chilberg Rowland, a violinist and W. Earl Rowland, an artist, lecturer and teacher. In 1936 he moved with his parents to Stockton, California where his father became director of the Haggin Museum.
SmallBPR-ElectraSmall	James Fisher (president) James Lee Fisher (born June 2, 1931) was the ninth president of Towson State University (now Towson University). Fisher was born and educated in Illinois. He came to Towson with considerable experience at various levels of university administration. Dr. Fisher was a fairly active president and left a significant imprint on the university. Among his accomplishments were the creation of 4 vice-presidential positions, establishment of 5 academic deans, founding of the Academic Council as a legislative and advisory body of faculty and students, creation of the Office of Institutional Development, addition of a winter	Louis XVI, whose name is attached to the first phases of the style. Straight tapering fluted legs joined by a block at the seat rail and architectural mouldings, characterize the style, in which each element is a discrete entity. Louis Delanois, Jean-Claude Sené and Georges Jacob were three leading chairmakers in the 1770s and 80s. The 18th century was indeed the golden age of the chair, especially in France and England (including Colonial America), between which there was considerable give and take of ideas. Even Diderot could not refrain from writing of them in his Encyclopédie.

Table 5.8: The result of two similar questions with the same key word **chair** on models trained by NQ dataset.

Question	Who is the chair man?	Which man invented the chair
SmallBPR-BERT	direct quotations. In World Schools Style debating, male chairs are called "Mr. Chairman" and female chairs are called "Madame Chair". The FranklinCovey Style Guide for Business and Technical Communication, as well as the American Psychological Association style guide, advocate using "chair" or "chairperson", rather than "chairman". The Oxford Dictionary of American Usage and Style suggests that the gender-neutral forms are gaining ground. It advocates using "chair" to refer both to men and to women. The word "chair" can refer to the place from which the holder of the office presides, whether on a chair	Rietveld's intent was to design a piece of furniture that could be cheaply mass-produced. He uses standard beechwood laths and pine planks that intersect and are fixed by wooden pegs. The functions of construction, the seat, the back and armrests are explicitly separated from one another visually. In fact, Rietveld saw the chair as the skeleton of an overstuffed armchair with all the excessive components removed. This modernist creation is perhaps one of the most iconic furniture designs of all time. The Wassily Chair, also known as the Model B3 chair
SmallBPR-DistilBERT	fewer than 100 performances. A largely British cast, including Elaine Paige – making her return to the West End after six years – John Partridge and Summer Strallen joined the show's co-author Bob Martin recreating his Broadway role of "Man in Chair." Joseph Alessi, Anne Rogers, Nickolas Grace, Nick Holder, Selina Chilton, Sean Kingsley, Adam Stafford, Cameron Jack, with Nina French, Paul Iveson, Mark Goldthorpe, Vanessa Barmby, Vivienne Carlyle, Sherrie Perrington and Mark Dickinson as lead understudies. Later during run, TV star, Steve Pemberton took over the role of "Man in Chair"	to the adoption of several features not found on other chairs. One of the earliest known innovators to have created the modern office chair was naturalist Charles Darwin, who put wheels on the chair in his study so he could get to his specimens more quickly. With the advent of rail transport in the mid-19th century, businesses began to expand beyond the traditional model of a family business with little emphasis on administration. Additional administrative staff was required to keep up with orders, bookkeeping, and correspondence as businesses expanded their service areas.
SmallBPR-ElectraSmall	destroyed after the horseś death. Unbeknownst to Tony, Paulie had kept the painting and altered it to change Tonyś clothes from a business suit to a colonial general. When Tony demands to know why he had him painted as a "lawn jockey," Paulie says that he did it out of sincere admiration for Tony as a leader. Tony pauses, but then rips the painting off the wall and puts it in a dumpster. As he throws it away, he stares at the generalś uniform and the sword. When Tony B returns to the farmhouse from an errand, Tony suddenly appears	Protector Palm Pistol The Protector Palm Pistol is a small .32 rimfire revolver designed to be concealed in the palm of the hand. It was unique in that the revolver was clasped in a fist with the barrel protruding between two fingers and the entire handgun was squeezed in order to fire a round. The Protector Palm Pistol was first patented and built in France in 1882 by Jacques Turbiaux and sold as the "Turbiaux Le Protector" or the "Turbiaux Disc Pistol". Later in 1883 it was built in the USA as The Protector by Minneapolis Firearms Co. Peter.

Table 5.9: The result of two similar questions with the same key word **chair** on models trained by TQA dataset.

5.4 Ablations

In the ablation experiment, we try to test how multi-task influences the main experiment result. When evaluating the experiment, we conduct three sets of controlled experiments on SmallBPR-BERT, SmallBPR-DistilBERT, SmallBPR-ElectraSmall separately. We eliminated the Reranking part in the three sets of experiments and compared the results with the complete experiments on different models. Deleting Reranking part means we only use the binary code to calculate the similarity between passages and questions. The results are shown in Fig 5.1. There are six charts indicating the precision for the same experiment with and without Reranking part. The orange bar is the result without Reranking and the blue bar is the result with Reranking.

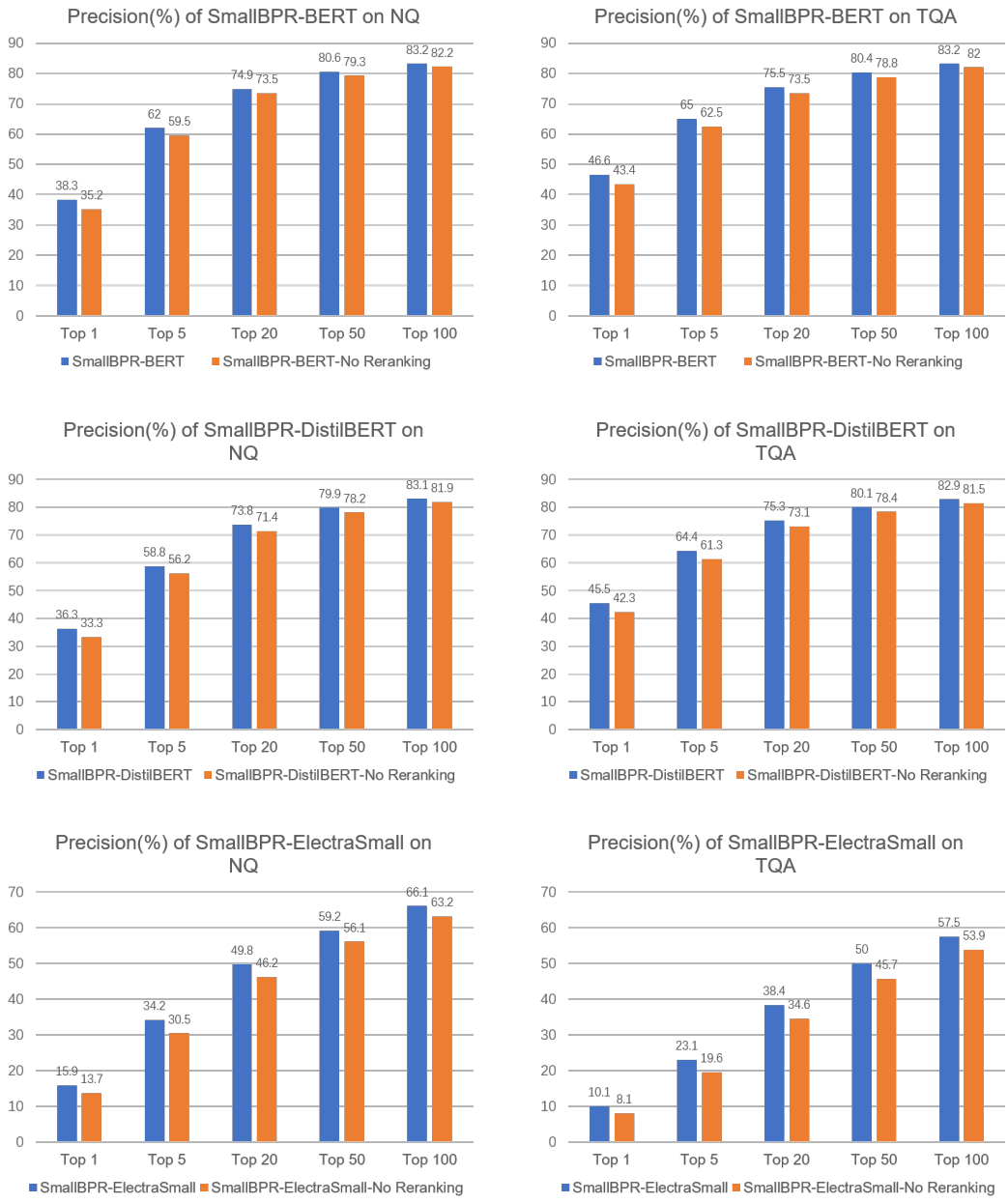

Figure 5.1: The result of controlled experiments.

We can see the result of different models on the NQ and TQA datasets. The result with Reranking is obviously better than those without Reranking. The reason is that in Reranking part, the continuous embedding is used to calculate the similarity, which contains more information than the binary code. In top-m passages, when the value of m is increasing, the gap between them is gradually narrowing. If we only select Top 1 passage, the gap is more than 3% in SmallBPR-BERT and SmallBPR-DistilBERT. And when we select Top 100 passages, the gap is decreasing to less than 1.5% in these two models.

Chapter 6

Conclusion

We have proposed a new model named SmallBPR with much smaller model size than BPR. We design a shared encoder for questions and passages and use small pretrained models to reduce the model size. Our experiments demonstrate that our models are smaller but without a great decrease in recall. For example, SmallBPR-BERT can achieve SmallBPR with ElectraSmall can achieve 83.2% top 100 recall on NQ while the model size is only half of BPR model. Some models with very small size can also perform well. The size of SmallBPR-ElectraSmall is only 154MB. And its top 100 recall on NQ dataset is 66.1%.

6.1 Limitations

Not dominant on well-resourced devices Current performance of our models is worse than the full original models, binary passage retriever, indicating significant room for improvements in retrieval systems. If we have adequate computing resources, the performance of SmallBPR is worse than other models, like BPR and DPR. It would not be suitable for the scenario that systems require high performance.

No test on real resource limited device We do not test our models on real resource limited devices. The main advantage of this project is significantly reducing the model size, so that the whole process can be done on a number of resource-limit devices, even on some mobile devices. It would be more useful to compare different methods on the real resource-limited devices to test the performance of our model.

Too little training data In this experiment, we only extract text passage from each dataset. Some other useful information is abandoned, like chart, formula. If we can consider all types of messages in Natural Questions and Trivia QA dataset, the performance of our model the model will be further improved.

6.2 Future Work

Extensive experiments and interesting findings of SmallBPR leave ample opportunities for further research. In this section we outline two promising research directions.

Applying SmallBPR with Document Reader In this experiment, we only explore the performance of SmallBPR model in the Document Reader part. We work on training the model so that it can find passages that include the correct answer. However, we are not sure if these passages can be used to train a high quality Reader. Proposing an efficient Document Reader

model and combining the Reader with SmallBPR can test whether the SmallBPR can be used in the ODQA system.

Optimized binary coding method After combine two BERT encoders into one, a potential defect of SmallBPR is that it cannot figure out the type of the input, a passage or a question. Though training one encoder to process questions and passages together can improve the generalization capability of SmallBPR, we still want to propose some encoding methods for the model to distinguish the inputs. A possible way to do so is adding a tag in the hash function. If the input is a question, the first bit of binary code is 1. Otherwise, the first bit is 0. This is just a general idea, we need further experiment to prove that our method is feasible.

Bibliography

[1] Petr Baudiš. Yodaqa: a modular question answering system pipeline. In *POSTER 2015-19th International Student Conference on Electrical Engineering*, pages 1156–1165, 2015. 11

[2] 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*, pages 1533–1544, 2013. 11

[3] 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, 2008. 2, 7

[4] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. *Advances in neural information processing systems*, 6, 1993. 3, 4

[5] Cristian Buciluǎ, Rich Caruana, and Alexandru Niculescu-Mizil. Model compression. In *Proceedings of the 12th ACM SIGKDD international conference on Knowledge discovery and data mining*, pages 535–541, 2006. 19

[6] Zhangjie Cao, Mingsheng Long, Jianmin Wang, and Philip S. Yu. Hashnet: Deep learning to hash by continuation. In *2017 IEEE International Conference on Computer Vision (ICCV)*, pages 5609–5618, 2017. 13

[7] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading Wikipedia to answer open-domain questions. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1870–1879, Vancouver, Canada, July 2017. Association for Computational Linguistics. 2, 7, 27

[8] Kevin Clark, Minh-Thang Luong, Quoc V Le, and Christopher D Manning. Electra: Pre-training text encoders as discriminators rather than generators. *arXiv preprint arXiv:2003.10555*, 2020. 5, 19, 21, 28

[9] Antonia Creswell, Tom White, Vincent Dumoulin, Kai Arulkumaran, Biswa Sengupta, and Anil A Bharath. Generative adversarial networks: An overview. *IEEE signal processing magazine*, 35(1):53–65, 2018. 19

[10] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long and Short Papers)*, pages 4171–4186, Minneapolis, Minnesota, June 2019. Association for Computational Linguistics. 3, 17, 18, 21

[11] Bhuwan Dhingra, Danish Danish, and Dheeraj Rajagopal. Simple and effective semi-supervised question answering. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies,*

Volume 2 (Short Papers), pages 582–587, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 4

[12] D. A. Ferrucci. Introduction to "this is watson". *IBM Journal of Research and Development*, 56(3.4):1:1–1:15, 2012. 2

[13] Poonam Gupta and Vishal Gupta. A survey of text question answering techniques. *International Journal of Computer Applications*, 53(4), 2012. 1

[14] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Mingwei Chang. Retrieval augmented language model pre-training. In *International Conference on Machine Learning*, pages 3929–3938. PMLR, 2020. 3

[15] Geoffrey Hinton, Oriol Vinyals, Jeff Dean, et al. Distilling the knowledge in a neural network. *arXiv preprint arXiv:1503.02531*, 2(7), 2015. 19

[16] 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, 2013. 11

[17] Yacine Jernite, Samuel R Bowman, and David Sontag. Discourse-based objectives for fast unsupervised sentence representation learning. *arXiv preprint arXiv:1705.00557*, 2017. 18

[18] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. 11, 25, 28

[19] Mandar Joshi, Eunsol Choi, Daniel Weld, and Luke Zettlemoyer. TriviaQA: A large scale distantly supervised challenge dataset for reading comprehension. In *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1601–1611, Vancouver, Canada, July 2017. Association for Computational Linguistics. 3, 5, 27

[20] Vladimir Karpukhin, Barlas Oğuz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. *arXiv preprint arXiv:2004.04906*, 2020. 3

[21] Boris Katz, Jimmy Lin, and Sue Felshin. Annotating the world wide web. In *In MIT Artificial Intelligence Laboratory Research Abstracts (this volume), September*, 2001. 1

[22] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. 3, 5, 27

[23] Kenton Lee, Ming-Wei Chang, and Kristina Toutanova. Latent retrieval for weakly supervised open domain question answering. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6086–6096, Florence, Italy, July 2019. Association for Computational Linguistics. 3, 27

[24] Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. Unsupervised question answering by cloze translation. *arXiv preprint arXiv:1906.04980*, 2019. 4

[25] Dayiheng Liu, Yeyun Gong, Jie Fu, Yu Yan, Jiusheng Chen, Daxin Jiang, Jiancheng Lv, and Nan Duan. Rikinet: Reading wikipedia pages for natural question answering. *arXiv preprint arXiv:2004.14560*, 2020. 11

[26] Lajanugen Logeswaran and Honglak Lee. An efficient framework for learning sentence representations. *arXiv preprint arXiv:1803.02893*, 2018. 18

[27] Tomas Mikolov, Martin Karafiát, Lukas Burget, Jan Cernockỳ, and Sanjeev Khudanpur. Recurrent neural network based language model. In *Interspeech*, volume 2, pages 1045–1048. Makuhari, 2010. 9

[28] Alexander Miller, Adam Fisch, Jesse Dodge, Amir-Hossein Karimi, Antoine Bordes, and Jason Weston. Key-value memory networks for directly reading documents. *arXiv preprint arXiv:1606.03126*, 2016. 11

[29] Dan Moldovan, Marius Pasca, Sanda Harabagiu, and Mihai Surdeanu. Performance issues and error analysis in an open-domain question answering system. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 33–40, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. 2

[30] Yixin Nie, Songhe Wang, and Mohit Bansal. Revealing the importance of semantic retrieval for machine reading at scale. *arXiv preprint arXiv:1909.08041*, 2019. 11

[31] Matthew E. Peters, Mark Neumann, Mohit Iyyer, Matt Gardner, Christopher Clark, Kenton Lee, and Luke Zettlemoyer. Deep contextualized word representations. In *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, pages 2227–2237, New Orleans, Louisiana, June 2018. Association for Computational Linguistics. 17

[32] Alec Radford and Karthik Narasimhan. Improving language understanding by generative pre-training. 2018. 17

[33] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. Squad: 100,000+ questions for machine comprehension of text. *arXiv preprint arXiv:1606.05250*, 2016. 10

[34] Santosh Kumar Ray, Shailendra Singh, and Bhagwati P Joshi. A semantic approach for question classification using wordnet and wikipedia. *Pattern recognition letters*, 31(13):1935–1943, 2010. 11

[35] Stephen Robertson, Hugo Zaragoza, et al. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends® in Information Retrieval*, 3(4):333–389, 2009. 2, 27

[36] Hasim Sak, Andrew W Senior, and Françoise Beaufays. Long short-term memory recurrent neural network architectures for large scale acoustic modeling. 2014. 19

[37] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf. Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter. *arXiv preprint arXiv:1910.01108*, 2019. 19, 21, 28

[38] Abigail See, Peter J Liu, and Christopher D Manning. Get to the point: Summarization with pointer-generator networks. *arXiv preprint arXiv:1704.04368*, 2017. 10

[39] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 15, 17, 18

[40] Suzan Verberne, Hans van Halteren, Daphne Theijssen, Stephan Raaijmakers, and Lou Boves. Learning to rank for why-question answering. *Information Retrieval*, 14(2):107–132, 2011. 2

[41] Ellen M Voorhees et al. The trec-8 question answering track report. In *Trec*, volume 99, pages 77–82, 1999. 2

[42] Alex Wang, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R Bowman. Glue: A multi-task benchmark and analysis platform for natural language understanding. *arXiv preprint arXiv:1804.07461*, 2018. 19

[43] Jun Wang, Wei Liu, Sanjiv Kumar, and Shih-Fu Chang. Learning to hash for indexing big data—a survey. *Proceedings of the IEEE*, 104(1):34–57, 2016. 12

[44] Ikuya Yamada, Akari Asai, and Hannaneh Hajishirzi. Efficient passage retrieval with hashing for open-domain question answering. *arXiv preprint arXiv:2106.00882*, 2021. 3

[45] Sohee Yang and Minjoon Seo. Designing a minimal retrieve-and-read system for open-domain question answering. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 5856–5865, Online, June 2021. Association for Computational Linguistics. 21

[46] Wojciech Zaremba, Ilya Sutskever, and Oriol Vinyals. Recurrent neural network regularization. *arXiv preprint arXiv:1409.2329*, 2014. 7, 19