Open Domain Question Answering System

Hoang Nghia Tuyen

Supervisor: **Asst. Prof. Pun Chi Seng**

Co-supervisor: **Assoc. Prof. Joty Shafiq Rayhan**

School of Physical & Mathematical Sciences

A thesis submitted to the Nanyang Technological University
as part of the honours requirements for the degree of
Bachelor of Science in Mathematical Sciences

2022

Acknowledgements

I would like to express my sincere gratitude to my supervisor Asst. Prof. Pun Chi Seng and my co-supervisor Assoc. Prof. Joty Shafiq Rayhan for their continuous support, guidance, encouragement, and invaluable feedback throughout the final year project.

I would also like to extend my gratitude to my parents for their wise counsel and sympathetic ear. I could not have completed this project without their support and understanding.

Abstract

Deep learning methods have drawn tremendous attention from both the research community and the industrial practitioners thanks to their undeniable power in learning feature representation in higher dimensions without manual, handcrafting features. An application of deep learning that arises naturally is question answering, in which a question answering system must answer questions posed by humans. One of its sub-fields, open-domain question answering, attempts to answer questions about nearly anything, without being given relevant reference texts. Despite its impactful applications in search engines, chatbots and factual correction, research work in open-domain question answering is relatively under-explored due to its complex and large-scale nature.

In this work, we aim to advance the progress of recent open-domain question answering systems by developing various mathematical-driven methods. More specifically, in the first part of this thesis, we introduce the widely adopted two-stage paradigm in open-domain question answering and perform comprehensive error analysis on state-of-the-art models. Based on this, we are then able to formulate and develop methods aiming specifically at overcoming these weaknesses in the second part of the thesis. These approaches range from simple methods such as parameter sharing and data augmentation to more sophisticated methods such as designing new objective functions or pseudo data synthesis and semi-supervised learning. Finally, we unify these developed methods into a single framework that outperforms state-of-the-art models by a significant margin on common benchmarking datasets. The code to reproduce our experiments is released at `https://github.com/hnt4499/DPR`.

Contents

List of Figures

List of Tables

Symbols and Acronyms

Symbols

$\mathbb{R}^n$	the n-dimensional Euclidean space
$\odot$	the Hadamard (component-wise) product
$\langle \cdot, \cdot \rangle$	the inner product of two vectors

Acronyms

QA	Question Answering
MRC	Machine Reading Comprehension
OpenQA	Open-domain Question Answering
NLP	Natural Language Processing
IR	Information Retrieval
DPR	Dense Passage Retrieval [2]
MIPS	Maximum Inner Product Search [3]
TF-IDF	Term Frequency-Inverse Document Frequency
LSTM	Long Short-term Memory [4]
FiD	Fusion in Decoder [5]
NQ	Natural Questions [6]
EM	Exact Match
FFN	Feed-forward Neural Networks
NLL	Negative Log-likelihood Loss
ANCE	Approximate Nearest Neighbor Negative Contrastive Learning [7]
GANs	Generative Adversarial Networks [8]

Chapter 1

Introduction

Deep learning, a sub-field of machine learning capable of learning higher-dimension feature representations via neural networks and gradient descent, has attracted remarkable attention from researchers over the last few decades. Landmark works in deep learning, for example, the first attempt of using GPUs in training neural networks [9], the introduction of the self-supervised training technique capable of mapping words to representation space [10], or the introduction of the Transformer architecture [11], have actively and significantly driven the deep learning research community forward. On the one hand, the ability to recognize patterns presented in and extract information from a large amount of data, without relying on handcrafting rules and heuristics, has made deep learning methods incredibly powerful and attractive. On the other hand, countless applications of deep learning have been developed in many aspects of our lives, from vision tasks such as facial recognition [12], image restoration [13], natural language tasks such as machine translation [14], speech recognition [15], to more complex decision-making tasks such as autonomous driving [16] or healthcare treatment administration [17].

1.1 Question Answering

Question answering (QA), a naturally arisen task that aims at answering questions posed by humans, has generally received great attention thanks to the wide range of real-world applications that it offers. However, most of the previous works on QA have mainly placed their focus on closed-domain question answering and machine reading comprehension (MRC). While closed-domain QA limits the questions being asked to a specific domain

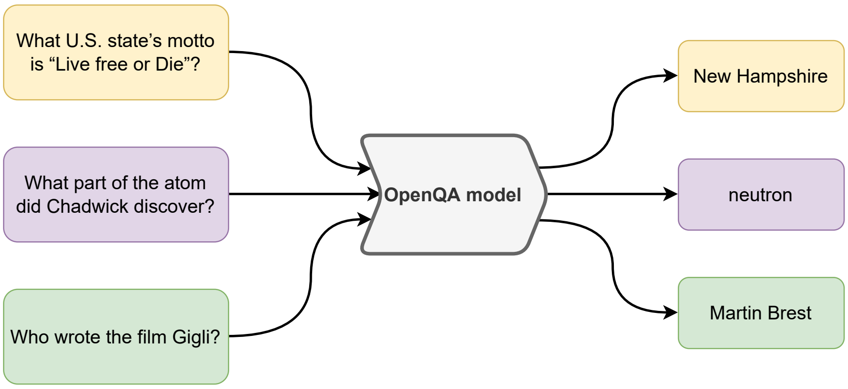

What U.S. state's motto is "Live free or Die"?
What part of the atom did Chadwick discover?
Who wrote the film Gigli?
OpenQA model
New Hampshire
neutron
Martin Brest

FIGURE 1.1: Examples of input-output pairs in open-domain question answering (OpenQA). The goal of OpenQA is to answer any factoid question about nearly anything. Adapted from [18].

knowledge (e.g., medicine), or to a specific type of questions (e.g., *when*), MRC systems are allowed to read and comprehend relevant context passages in order to answer the questions. In contrast, the long-standing problem of open-domain question answering [19] (OpenQA) has observed significantly less progress, potentially due to its complex and large-scale nature. Open-domain question answering is a much more challenging task than closed-domain QA or MRC as it requires cutting-edge techniques from both natural language processing (NLP) and information retrieval (IR). In OpenQA, the model must answer any factoid questions, possibly about anything, without being provided relevant contexts containing the answer. Figure 1.1 illustrates several examples of input-output pairs in OpenQA. Notably, OpenQA can be utilized in a wide range of applications, ranging from online customer service systems, chatbots (e.g., Siri) to search engines (e.g., Google), where user inputs are usually information-seeking questions. With OpenQA, we aim at developing intelligent systems that automatically retrieve and extract relevant information of a given question, and subsequently suggest a detailed answer.

1.2 Closed-book and Open-book Open-domain Question Answering

OpenQA can be further classified into two categories, namely closed-book OpenQA and open-book OpenQA.

1.2.1 Closed-book Open-domain Question Answering

In closed-book OpenQA, the system is not given access to any textual documents when answering the questions, and thus is expected to memorize factual knowledge ahead of time stored implicitly in its parameters (*parametric memory*). At test time, the system is expected to provide answers to questions that it has already encountered during training by retrieving from its memory, or to infer answers to unseen questions using general ontologies and common knowledge. This is analogous to a student memorizing possible question-answer pairs from a set of past year papers before coming to a closed-book final examination.

Despite its simplicity and straightforward formulation, there exists a number of inherent weaknesses in closed-book OpenQA systems. First, closed-book OpenQA models, which are generative models by design, have been long known to suffer from *factual hallucination* [20]. Factual hallucination is a common phenomenon in text generation in which the models fail to precisely retrieve factual information from their parametric memory, and thus unintendedly attempt to fabricate it. Such unfaithful behaviors should obviously be avoided in question answering where precise fact is of utmost importance. Second, closed-book OpenQA systems generally require a huge number of parameters to accommodate their parametric memory, which often far exceeds the amount of storage used to store factual knowledge in plain text. For example, a state-of-the-art closed-book OpenQA system [21] is only able to match the performance of a recent open-book OpenQA system [2] while requiring 11 times as many parameters. Third, it is not trivial to update closed-book OpenQA models with new information (for example recent events), as it would require re-training the models. This in turn gives rise to the *catastrophic forgetting* problem [22], another common issue in deep learning in which the model loses its generalizability and catastrophically forgets existing knowledge after being trained on new data.

1.2.2 Open-book Open-domain Question Answering

Open-book OpenQA has been proposed to tackle the aforementioned issues that closed-book OpenQA suffers from. At inference time, an open-book OpenQA system is provided access to a large knowledge base, for example Wikipedia articles, from which it can search for supporting passages to the given question. This formulation allows the system to retrieve precise factual knowledge from the non-parametric memory of passages, effectively overcoming the factual hallucination and catastrophic forgetting problems. At the same time, the number of parameters can be significantly reduced given that the models no longer have to memorize information by storing it in the parameters. Finally, such non-parametric knowledge source as Wikipedia is human interpretable and can be easily updated. Given these advantages, in this work, we mainly tackle the problem of open-book OpenQA and simply refer to it as *OpenQA* for brevity.

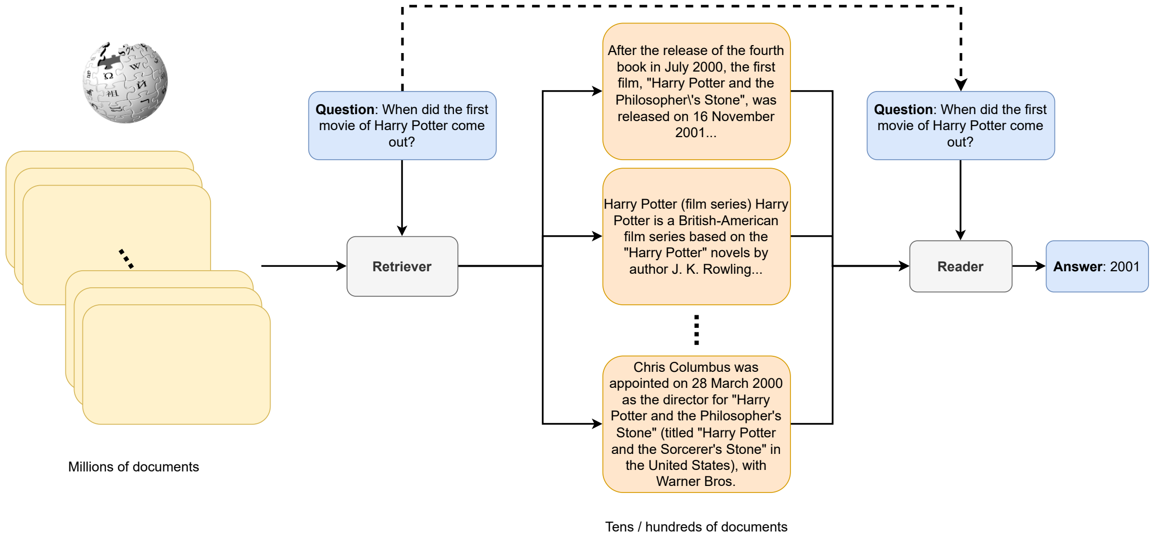

Question: When did the first movie of Harry Potter come out?
Millions of documents
Retriever
After the release of the fourth book in July 2000, the first film, "Harry Potter and the Philosopher\'s Stone", was released on 16 November 2001...
Harry Potter (film series) Harry Potter is a British-American film series based on the "Harry Potter" novels by author J. K. Rowling...
Chris Columbus was appointed on 28 March 2000 as the director for "Harry Potter and the Philosopher's Stone" (titled "Harry Potter and the Sorcerer's Stone" in the United States), with Warner Bros.
Tens / hundreds of documents
Question: When did the first movie of Harry Potter come out?
Reader
Answer: 2001

FIGURE 1.2: An overview of the two-stage retriever-reader model. The retriever first performs a highly efficient search algorithm to retrieve top-k most relevant passages from a large collection of documents (e.g., Wikipedia). The reader then reads this subset of documents to infer the answer to the given question.

It is important to highlight that we are processing millions of documents in open-book OpenQA given a single input query. Thus, it would be computationally infeasible to naively apply machine reading comprehension to every passage in the corpus and return the most probable answer. To overcome this challenge, Chen et al. [23] proposed a two-stage *retriever-reader* paradigm that has gained popularity recently due to its effectiveness and efficiency. Figure 1.2 presents an overview of this approach. In the first stage of this framework, a small number of relevant passages is retrieved by a *retriever* model using a highly efficient search algorithm. A *reader* model is then used to perform reading comprehension on this subset of passages to obtain the final answer. Analogously, in an open-book exam, the students should first identify relevant sections in the textbooks before actually delving into the details to find answers to a given question, instead of reading every paragraph one by one looking for the solutions.

In Chen et al. [23], a simple TF-IDF [24] weighted bag-of-words method is used as the retrieval method while an LSTM-based [4] model is employed as the MRC method. The retrieval method, which belongs to a family of non-trainable *sparse retrieval* approaches, performs a simple and efficient word matching step to find relevant passages. On the other hand, the reader model has a limited capability of understanding long sequences, be it the forgetfulness in RNN-based models or quadratic time complexity in Transformer-based models [1]. Since the introduction of the two-stage paradigm, many advances in Natural Language Processing (NLP) have been proposed to push the boundary of OpenQA systems further, especially the retriever model. Lee et al. [25] and Guu et al. [26] adopted trainable *dense retrieval* models, which are first pre-trained in an unsupervised manner before being fine-tuned on the downstream OpenQA data. Seo et al. [27] combined the complementary advantages of sparse retrieval and dense retrieval in a unified framework. Khattab and Zaharia [28] aimed to address a key challenge in dense retrieval, namely the *decomposability gap* which we will introduce shortly, with vector decomposition and late interaction to mimic the self-attention [11] mechanism. Dense Passage Retrieval (DPR) [2] later showed that pre-training is not needed for dense retrieval as the retriever model directly trained on OpenQA data with in-batch negatives and the negative log-likelihood (NLL) objective function significantly outperformed all other approaches.

In all of the previous algorithms, the query and passage texts will first be compressed into some vectorized form in order to enable a highly efficient search algorithm on the retriever side. For sparse retrieval, the documents are encoded with weighted word frequency, for which severe information loss will incur such as the loss of sequential information and semantic meaning of the texts. On the other hand, for dense retrieval, information compression is done by mapping each sequence of words to its high-dimensional vector representation. Not only does this approach benefit from the advantages of representation learning from supervised data, but it also enables the use of the well-studied, highly efficient Maximum Inner Product Search (MIPS) [3] algorithm. Notably, regardless of the retrieval type, the computational cost of extracting information from the knowledge base can be further amortized by pre-computing and pre-indexing it into a MIPS index. As a result, at test time, we only need to compress the input question (e.g., by feeding it through the dense retriever model to obtain its vector representation) before sending it to the MIPS index for efficient retrieval of relevant passages. This procedure effectively reduces the number of passages for the reader model from millions of documents to 50-100 documents in real-time. However, it also inevitably limits the ability of the model to capture mutual information between the passages and the input questions (so-called the *decomposability gap*). As discussed above, this is to some extent caused by the loss of information during the data compression step, even for dense retrievers. Moreover, it is important to note that the evidence documents and the input query are encoded independently of each other. Thus, the task of the retrievers is to find the best passage matches to a given question when both the passages and the question have already been encoded, without being allowed to comprehend the original texts. In other words, decomposability gap happens when the computationally expensive reading comprehension task is decomposed into two smaller and efficient tasks of feature representation and maximum inner product search. As a consequence, this performance trade-off significantly negatively affects the relevance of passages retrieved in the first stage, which in turn is detrimental to the reader performance, thus the overall OpenQA performance.

Decomposability gap is regarded as the most challenging yet attractive problem in OpenQA. Previous work on mitigating this challenge can be classified into several main directions. To explicitly tackle the decomposability gap, Khattab and Zaharia [28] aimed at mimicking the self-attention mechanism between the passage and the input query via late vector interaction, while Das et al. [29] allowed communication between the retriever and reader to iteratively fine-tune the retrieval results. Another research direction is query expansion, whereby the input query is first augmented with relevant keywords before compression, effectively pushing the representation of this query closer towards that of the gold passages in the feature representation space [30, 31]. Notably, a large body of works focused on improving the retriever performance with novel pre-training paradigms [25, 26, 32–35]. Lastly, Seo et al. [27], Lee et al. [36] and Luan et al. [37] combined the complementary strengths of sparse and dense retrieval to improve the overall retrieval performance. In this final year project, we extend upon the outstanding work Dense Passage Retrieval [2]. We explore the first three aforementioned directions as well as other directions and further propose multiple novel approaches with the main goal of improving the overall OpenQA performance, as well as with a particular focus on the decomposability gap of retrieval.

1.3 Major Contributions

Our main contributions can be stated as follows:

- We propose several simple yet under-explored multi-task learning approaches for OpenQA that yield sizeable performance improvements while reducing the memory footprint of the retriever, which is critical for real-time OpenQA systems.

- We explore a simple method aimed at capturing the mutual information between the question query and documents during retrieval. This method explicitly considers the decomposability gap and can be seen an alternative to the late interaction mechanism proposed in Khattab and Zaharia [28].

- We propose a simple data augmentation method to diversify the retrieval training data, thereby improving the previous state-of-the-art DPR retriever by a large margin.

- We investigate the effect of the objective function (so-called loss function) on retrieval training, from which we propose new objective functions for representation learning that achieve marginal gains over the baseline model.

- Finally, we propose a novel data synthesis - semi-supervised pre-training - query expansion paradigm with the goal of comprehensively improving both the retriever and reader, for which we achieve considerable performance improvements across several benchmarking datasets.

Chapter 2

Background

In this chapter, we present background that will be necessary to elaborate on our proposed approaches later in the report. First, we introduce a typical experimental setup of OpenQA along with notations that we will be using throughout the report. Next, we introduce Dense Passage Retrieval (DPR) [2], a recently published work with a novel retriever training method with the negative log-likelihood objective function and in-batch negative technique. We then describe the default experimental settings used in our experiments.

2.1 Notations

For OpenQA, we are given a large corpus of documents containing factual information from which we retrieve knowledge needed to perform question answering. It is a common practice in the OpenQA literature to split each of these documents into several text chunks of equal lengths [2, 5, 7, 32, 38–40], resulting in a corpus $\mathcal{C}$ of M passages $\mathcal{C} = \{p_1, p_2, \ldots, p_M\}$. This is because existing reading comprehension methods have a limited capability of reading long sequences as discussed in Section 1.2.2. Each passage p_i can further be viewed as a sequence of tokens (words) $w_1^{(i)}, w_2^{(i)}, \ldots, w_{|p_i|}^{(i)}$. Given an input question q, the task is to find its answer under the form of a text span $w_{s,e}^{(i)} = w_s^{(i)}, w_{s+1}^{(i)}, \ldots, w_e^{(i)}$ of a passage p_i, where s and e denote the *start* and *end* indices of the span, respectively. It is important to highlight the flexibility of OpenQA that during inference, any text corpus can be used to answer the question as long as it provides necessary information. In practice, the corpus size can range from millions (e.g., Wikipedia) to billions or trillions (e.g., the Web) of passages.

Under the two-stage retriever-reader paradigm, we have a retriever R and a reader S formulated as

Definition 2.1 (Retriever)**.** A retriever under the two-stage paradigm is a model that returns a small filtered subset of the given large set of passages.

$$R(q, \mathcal{C}) = \mathcal{C}_{\mathcal{F}} \tag{2.1}$$

Definition 2.2 (Reader)**.** A reader under the two-stage paradigm is a model that produces an answer to the given question given the filtered set from the retriever.

$$S(q, \mathcal{C}_{\mathcal{F}}) = w_{s,e}^{(i)} \tag{2.2}$$

where $\mathcal{C}_{\mathcal{F}}$ is a small set of k passages with $k \ll M$ and $w_{s,e}^{(i)}$ is the final answer of the OpenQA system to the question q. In other words, the task of the retriever is to efficiently retrieve a small set $\mathcal{C}_{\mathcal{F}}$ of passages that it considers relevant. The reader will then carefully comprehend the question q as well as each of the passages in $\mathcal{C}_{\mathcal{F}}$ to infer the answer. We refer readers to Figure 1.2 for an illustration of an OpenQA system.

Furthermore, throughout this report, we refer to an *encoder*, denoted as E, as a BERT model [1] that first appends a special token `[CLS]` to the input text sequence and then maps the sequence to the embedding of `[CLS]`.

Definition 2.3 (Encoder)**.** An encoder is a model that maps an input text sequence

$$p = \{\,\texttt{[CLS]}, w_1, w_2, \dots, w_{|p|}\}$$

to the vector embedding of the special token token `[CLS]`

$$E(p)_{\texttt{[CLS]}} = \mathbf{v}_{\texttt{[CLS]}} \in \mathbb{R}^d \tag{2.3}$$

where d is the embedding dimension, and $\mathbf{v}_{\texttt{[CLS]}}$ is known as the feature representation [1] of the special token `[CLS]`. Historically, `[CLS]` is appended to the input to serve as a contextualized embedding that encodes the semantic meaning of the *entire* sequence [1].

[1] In this report we will use *embedding*, *feature* and *representation* interchangeably.

With a proper objective function, we can train E to encode important semantic meaning of p into $\mathbf{v}_{\texttt{[CLS]}}$.

2.2 DPR Retriever

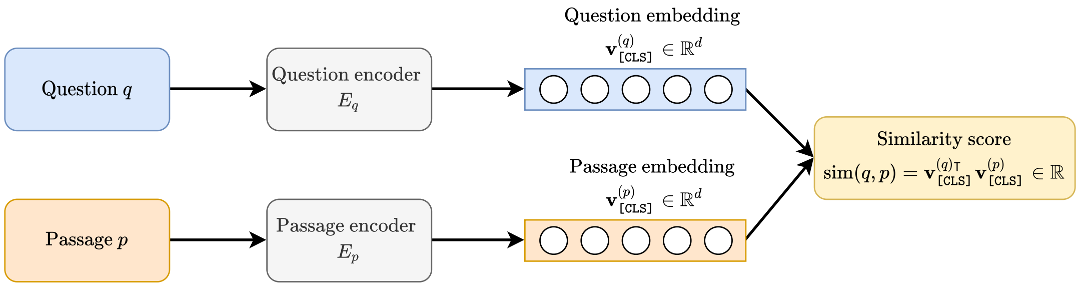

Question embedding
$\mathbf{v}^{(q)}_{\texttt{[CLS]}} \in \mathbb{R}^d$
Question q
Question encoder
E_q
Similarity score
$\text{sim}(q,p) = \mathbf{v}^{(q)\intercal}_{\texttt{[CLS]}} \mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R}$
Passage embedding
$\mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R}^d$
Passage p
Passage encoder
E_p

FIGURE 2.1: The two-tower architecture of the DPR retriever, consisting of a question encoder and a passage encoder. Each of the towers encodes input texts into their embedding vectors, and the final similarity score is computed as their dot product. Adapted from [18].

Despite its simplicity, DPR was able to outperform all prior arts by large margins when it was introduced. In this section, we describe the retriever component of DPR and refer interested readers to [2] for a detailed description of the reader component. The DPR retriever consists of a passage encoder E_P and a question encoder E_Q, whose task is to embed passages and questions to their representation space, respectively.

$$\begin{aligned} E_P(p)_{\texttt{[CLS]}} &= \mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R}^d \\ E_Q(q)_{\texttt{[CLS]}} &= \mathbf{v}^{(q)}_{\texttt{[CLS]}} \in \mathbb{R}^d \end{aligned} \tag{2.4}$$

We then define the similarity between a question and a passage as their dot product.

Definition 2.4 (Vector similarity)**.** The similarity between a question and a passage is defined as the dot product of their vector embeddings.

$$\text{sim}(q,p) = \mathbf{v}^{(q)\intercal}_{\texttt{[CLS]}} \mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R} \tag{2.5}$$

Figure 4.1 presents an illustration of the DPR retriever. This two-tower architecture is widely used in *metric learning* [41, 42] and *self-supervised learning* [43, 44] in which models are trained to maximize the similarity of similar inputs and minimize the similarity of dissimilar inputs. As discussed in Section 1.2.2, this design brings about the real-time performance of the well-studied Maximum Inner Product Search (MIPS) algorithm [3] but at the same time is detrimental to the retrieval performance due to the decomposability gap which we aim to address.

In this project, we follow the DPR implementation and use BERT [1] as the architecture for all encoders. As a result, each embedding vector is a $d = 768$-dimensional vector. We omit the details of BERT and refer interested readers to [1].

2.2.1 Training and Inference

Given an input question q_i posed by users, we denote p_i^+ as a positive passage that contains an answer to q_i, and p_i^- as a negative passage otherwise. Intuitively, we want to train the model to maximize the similarity between q_i and p_i^+, at the same time minimizing the similarity between q_i and p_i^-. Formally, under the metric learning framework, we want to construct an *latent space* such that relevant pairs of questions and passages are closer to each other than irrelevant pairs. Therefore, given a question q_i, a relevant passage p_i^+ and a set of n irrelevant passages $\{p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-\}$, we train the model to minimize the negative log-likelihood loss function:

Definition 2.5 (Negative log-likelihood)**.** The negative log-likelihood loss function of the positive passages.

$$L(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^{n} e^{\text{sim}(q_i, p_{i,j}^-)}} \tag{2.6}$$

where we apply the softmax function to the similarity vector before taking its negative log value.

Definition 2.6 (Softmax)**.** Softmax is the function that takes as input a vector of K real numbers and outputs a normalized probability distribution vector of the same length.

$$\sigma(\mathbf{z})_i = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}} \text{ for } i = 1, 2, \ldots, K \tag{2.7}$$

In practice, we consider passages to be positive if they contain a text span that matches the answer of the question, and negative otherwise.

In addition to the simple yet novel negative log-likelihood objective function, Karpukhin et al. [2] also proposed a simple in-batch negative training setting that works extremely well empirically. Suppose we have a mini-batch of B (question, positive passage) pairs. We first feed this batch through the two-tower retriever to obtain a question embedding matrix $\mathbf{Q} \in \mathbb{R}^{B \times d}$ and a passage embedding matrix $\mathbf{P} \in \mathbb{R}^{B \times d}$. The similarity matrix is then computed as:

$$\mathbf{S} = \mathbf{Q}\mathbf{P}^{\intercal} \in \mathbb{R}^{B \times B} \tag{2.8}$$

By doing so, we are effectively comparing B^2 question-passage pairs in the mini-batch, from which we want to minimize the negative log of the softmaxed values along the diagonal of the square similarity matrix $\mathbf{S}$. This is called *in-batch negatives* since we reuse computations from within the batch with positive passages of other questions as negative passages, thereby efficiently reducing the memory footprint and effectively scaling the batch size up. Furthermore, Karpukhin et al. [2] proposed to utilize an additional hard negative passage per each question in the minibatch, obtained from the sparse retrieval method BM25 [45]. In other words, each question will be associated with a single positive passage and $2B - 1$ negative passages, of which $2(B - 1)$ passages are in-batch negatives and one passage is hard negative.

During inference, we first build a FAISS index [3] by feeding all available passages in the corpus to E_P. This incurs a non-recurring cost since this step is only done once. At runtime, given an input question q, we obtain its embedding via $\mathbf{v}^{(q)}_{\texttt{[CLS]}} = E_Q(q)_{\texttt{[CLS]}}$, which gets sent to the FAISS index to perform top-k similarity search. A set $\mathcal{C}_{\mathcal{F}}$ of top-k most relevant passages according to the similarity score $\text{sim}(q, p_i)$ is produced by the algorithm in milliseconds, concluding the retrieval step.

2.3 Experimental Setup

In this section we briefly describe the default experimental setup used in our experiments. We note that we reuse a substantial portion of the code and data given by the DPR authors [2] [2], unless otherwise specified below. Therefore, we refer interested readers to [2] for further details on the setup.

[2] https://github.com/facebookresearch/DPR/

2.3.1 Data

Following [2], we use the English Wikipedia dump from Dec. 20, 2018 as the knowledge source from which passage retrieval is done. This knowledge base is provided by the DPR authors after post-processed to remove semi-structured data and split into chunks of 100-word passages. There are in total 21,015,324 passages in this corpus. On the other hand, we use Natural Questions (NQ) [6] as the question answering dataset on which we train and evaluate our retriever and reader models. This dataset provides questions mined from the real Google search queries as well as the corresponding positive Wikipedia passages along with the answer text span.

2.3.2 Evaluation Metrics

Following [2], we evaluate the retriever model by the *retrieval accuracy* and the reader model with *exact match (EM)* and *F1 score.* Specifically, retrieval accuracy, or retrieval recall, is measured as the percentage of top-k retrieved passages that contain the gold answer obtained from NQ. It can be understood as how relevant your retrieved passages are to a given question. On the reader side, exact match is a direct measure of the reader model performance. It is calculated as the percentage of answers returned by the reader model that precisely match with one of the gold answers. F1 score on the other hand allows some relexations on the returned answers by measuring the F1 score between the answer by the reader model and the gold answer.

Definition 2.7 (F1 score)**.** F1 score in the context of QA is calculated as the harmonic mean between the precision and recall of the returned answer and the gold answer.

$$F_1 = \frac{2}{\text{recall}^{-1} + \text{precision}^{-1}} \tag{2.9}$$

Chapter 3

Shared Encoders

3.1 Method

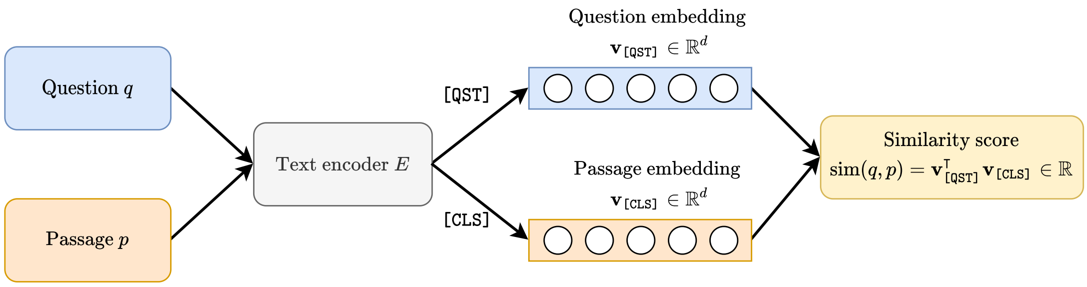

Question embedding
$\mathbf{v}_{\texttt{[QST]}} \in \mathbb{R}^d$
Question q
[QST]
Text encoder E
Similarity score
$\text{sim}(q,p) = \mathbf{v}_{\texttt{[QST]}}^{\top} \mathbf{v}_{\texttt{[CLS]}} \in \mathbb{R}$
Passage embedding
$\mathbf{v}_{\texttt{[CLS]}} \in \mathbb{R}^d$
[CLS]
Passage p

FIGURE 3.1: The two-tower architecture of the DPR retriever with parameter sharing. The same encoder will encode input texts into their embedding vectors with proper special tokens [QST] or [CLS] to distinguish the input types.

In this chapter, we present our first contribution to the DPR architecture [2] - *shared encoders*. Recall that the two-tower DPR retriever consists of a question encoder E_Q and a passage encoder E_P with the same architecture (i.e., BERT [1]) but different weights [1]. In this setup, the task of both encoders is to textual data to the same embedding space, one being question texts and one being passage texts. Therefore, it emerges naturally that sharing parameters of these two models could be beneficial.

More specifically, we use the same set of parameters for the two encoders while assigning the special token [CLS] to the passage encoder and [QST] to the question encoder. The

[1] In the context of deep learning, *weights* refers to the values of the parameters of the model.

Negative type	**Retriever**	Top-1	Top-5	Top-20	Top-100
BM25	DPR	42.01	64.54	76.48	84.29
	DPR (shared encoders)	**45.01**	**66.70**	**78.25**	**85.62**
DPR hard negatives	DPR	49.36	67.34	78.09	85.40
	DPR (shared encoders)	**53.02**	**71.30**	**80.89**	**86.93**

Table 3.1: Top-$\{1, 5, 20, 100\}$ retrieval accuracy on the Natural Questions test set, calculated as the percentage of top-k retrieved passages that contain the answer. We present the results on training with two different negative types, BM25 or hard negatives. The proposed shared encoders approach consistently and substantially outperforms the baseline DPR model on various settings with no additional cost.

similarity score between a question q and a passage p defined in (2.5) then becomes:

$$\text{sim}(q, p) = \mathbf{v}_{\texttt{[QST]}}^{\intercal} \mathbf{v}_{\texttt{[CLS]}} \in \mathbb{R} \tag{3.1}$$

Figure 3.1 illustrates the architectural design of this approach. Under this architecture, we allow the models to share the general world knowledge and natural language understanding capabilities and at the same time to distinguish the input types. Furthermore, this approach can be seen as a multi-task training algorithm, in which the general encoder E is trained to map both questions and passages to the same feature space.

3.2 Experimental Results

We provide the retrieval results on NQ in Table 4.1, where we train the DPR model with BM25 hard negative passages described in Section 2.2.1 and DPR hard negative passages, respectively. In the latter case, negative passages are obtained by performing retrieval with a DPR checkpoint then for each question taking the highest-scoring passage that does not contain the answer. We note that our results on the original DPR architecture do not match those reported in the original paper [2], as we trained all these models with a batch size of 24 instead of 128 given our computation budget. Nevertheless, we observe a consistent and considerable improvement of the shared encoders across different training settings and different top-k evaluation. This attests to our hypothesis earlier that this approach allows knowledge sharing and multi-task training that are beneficial to the model performance.

Intriguingly, we observe that the improvement of the shared encoders over the DPR baseline is consistently higher with DPR hard negatives than BM25 hard negatives. For example, for top-5 retrieval accuracy, the performance gain of DPR shared encoders with DPR hard negatives is 3.96 points which is almost double of that with BM25 hard negatives (2.16 points). This is opposite to the general intuition that it becomes increasingly difficult to improve a model when its performance is increased. We hypothesize that this attributes to the knowledge sharing power of shared encoders, which can capitalize more on such informative negatives as DPR hard negatives.

Additionally, we note that by sharing the parameters of the two encoders, we effectively reduce the memory footprint by half. This is especially critical in retrieval training where in-batch negatives are used, hence gradient accumulation is not sufficient to accommodate for a smaller batch size. We expect the shared encoders to outperform the baseline DPR model even further when trained on a larger batch size, which is an advantage of shared encoders brought about by the memory efficiency of the architectural design. We leave it to a future work to empirically verify this hypothesis.

Finally, we note that given its efficiency and effectiveness, we treat the DPR retriever with shared encoders as the baseline DPR model for all subsequent experiments, unless otherwise noted.

Chapter 4

Late Interaction

4.1 Method

In this chapter, we present our next contribution designed specifically to tackle the long-standing decomposability gap problem described in Section 1.2.2. Recall that in the original DPR architecture, the question q and passage p are encoded *independently* at run time (see (2.4)), which later gets multiplied to obtain the dot product as their similarity score. By doing so, we effectively decompose the task of reading both q and p together to the task of extracting important information from each q and p to embedding vectors and later comparing them. Our goal is to minimize the information loss caused by the extraction step.

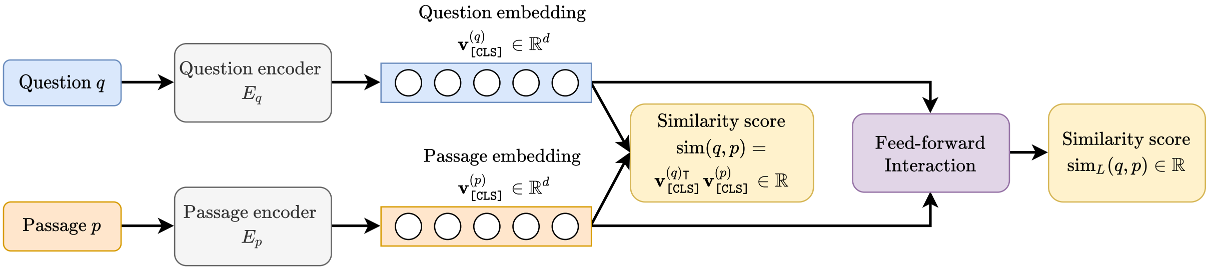

Question embedding
$\mathbf{v}^{(q)}_{\texttt{[CLS]}} \in \mathbb{R}^d$
Question q
Question encoder
E_q
Similarity score
$\text{sim}(q,p) =$
$\mathbf{v}^{(q)\top}_{\texttt{[CLS]}} \mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R}$
Feed-forward
Interaction
Similarity score
$\text{sim}_L(q,p) \in \mathbb{R}$
Passage embedding
$\mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R}^d$
Passage p
Passage encoder
E_p

FIGURE 4.1: The two-tower architecture of the DPR retriever with the late interaction mechanism. After the top-n retrieval with the usual similarity scores $\text{sim}(q,p)$, where $n > k$, embedding vectors are fed to the feed-forward interaction layer to obtain the late interaction similarity scores $\text{sim}_L(q,p)$, after which the final set of ranked passages is returned.

In this work, we propose to append a small parametric module after the information extraction step. This component is designed to *recover* necessary information from both the question and passage embeddings that will then be compared again for retrieval, thus called *late interaction*. Figure 4.1 presents an overview of our proposed method. Specifically, suppose we have a question embedding vector $\mathbf{v}^{(q)}_{\texttt{[CLS]}} \in \mathbb{R}^d$ and a passage embedding vector $\mathbf{v}^{(p)}_{\texttt{[CLS]}} \in \mathbb{R}^d$ obtained from the question encoder and passage encoder, respectively, using (2.4). We then allows simple interactions between the two vectors

$$\mathbf{v}^{(qp)}_{\texttt{[CLS]}} = [\mathbf{v}^{(q)}_{\texttt{[CLS]}}; \mathbf{v}^{(p)}_{\texttt{[CLS]}}; \mathbf{v}^{(q)}_{\texttt{[CLS]}} \odot \mathbf{v}^{(p)}_{\texttt{[CLS]}}; \mathbf{v}^{(q)}_{\texttt{[CLS]}} - \mathbf{v}^{(p)}_{\texttt{[CLS]}}] \in \mathbb{R}^{4d} \tag{4.1}$$

where $[\,\cdot\,;\,\cdot\,]$ denotes vector concatenation and $\odot$ denotes the Hadamard product, which is a element-wise operation. We obtain $\mathbf{v}^{(qp)}_{\texttt{[CLS]}}$, a vector containing information when the question representation is allowed to interact with the passage representation on the feature level via element-wise arithmetic operations. This information-rich vector is then fed through a simple feed-forward (FFN) parametric module to obtain the final similarity score

$$\mathrm{sim}_L(q,p) = \mathbf{m}_L^{\intercal}\mathbf{v}^{(qp)}_{\texttt{[CLS]}} \in \mathbb{R} \tag{4.2}$$

where $\mathbf{m}_L \in \mathbb{R}^{4d}$ is the weight vector of the FFN.

During training, we apply the same negative log-likehood objective function defined in Definition 2.5 to the novel late interaction similarity scores

$$L_L(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) = -\log \frac{e^{\mathrm{sim}_L(q_i, p_i^+)}}{e^{\mathrm{sim}_L(q_i, p_i^+)} + \sum_{j=1}^{n} e^{\mathrm{sim}_L(q_i, p_{i,j}^-)}} \tag{4.3}$$

The final objective function is then taken as sum of the two individual losses

$$L_{\text{late\_interaction}} = L(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) + \lambda L_L(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) \tag{4.4}$$

where λ is the weight of the late interaction objective component. In other words, we train the retriever model to minimize the losses with respect to both the usual similarity scores and late-interaction similarity scores, which can also be viewed as a multi-task training paradigm.

At test time, we use $\mathrm{sim}(q,p)$ for our retrieval method and $\mathrm{sim}_L(q,p)$ for our re-ranking method. In particular, given at input query q, we first retrieve a set $\mathcal{C}_{\mathcal{F}\prime}$ of top-n highest-scoring passages from a pre-built FAISS index as described in Section 2.2.1, where $n > k$. This means that we retrieve more passages than we need. Afterwards, we feed the question feature embeddings $\mathbf{v}^{(q)}_{\texttt{[CLS]}}$ as well as the embeddings of each passage in $\mathcal{C}_{\mathcal{F}\prime}$ to the late

Architecture	**Loss function**	Top-1	Top-5	Top-20	Top-100
DPR (shared encoders)	DPR loss	**53.02**	**71.30**	**80.89**	**86.93**
	DPR loss + late interaction loss	51.66	69.42	79.64	86.15

Table 4.1: Top-$\{1, 5, 20, 100\}$ retrieval accuracy on the Natural Questions test set of the DPR retriever (shared encoders) with and without the late interaction module, calculated as the percentage of top-k retrieved passages that contain the answer. The proposed interaction component degrades the baseline DPR model performance by a sizable margin.

interaction module via (4.1) and (4.2) to obtain the late-interaction similarity scores with minimal additional computational cost. These scores are then used to re-rank and filter the passages in $\mathcal{C}_{\mathcal{F}'}$ to obtain the final set $\mathcal{C}_{\mathcal{F}}$ of k most relevant passages.

4.2 Experimental Results

We conduct an experiment on NQ following the experimental settings described in Section 2.3 with a batch size of 24. The baseline model is taken as the DPR retriever with parameter sharing introduced in Chapter 3, and the hyperparameters are set to $\{n, \lambda\} = \{200, 1.0\}$. Table 4.1 presents the retrieval recall on the NQ test set.

We observe that the late interaction component brings about a marginal performance loss to the DPR retriever. We believe there are several reasons that make designing a late interaction layer difficult. First, the proposed interaction layer consists of a small parametric module with weights $\mathbf{m}_L \in \mathbb{R}^{4d}$, which might not be sufficient to fully capture such complex natural language interactions. Second, our interaction layer makes use of only element-wise operations ((4.1)), therefore is incapable of modeling the complex cross-feature interactions between the question embeddings and passage embeddings. Lastly, it is worth noting that we did not specifically tune the newly introduced hyperparameters $\{n, \lambda\}$ which can have a huge impact on the final model performance. Nevertheless, a concurrent work of Khattab and Zaharia [28] that shares a very similar idea to our proposed late interaction has shown that this mechanism is a valuable addon to the existing retriever to combat the problem of decomposability gap. Therefore, we refer interested readers to [28] for a complete work on this idea.

Chapter 5

Multi-similarity Loss

5.1 Method

In this chapter, we introduce our novel multi-similarity loss for the DPR retriever model. Recall that Karpukhin et al. [2] proposed a negative log-likelihood (NLL) objective function that aimed to maximize the similarity between relevant question-passage pairs while minimizing the similarity of the irrelevant ones, as formulated in Definition 2.5

$$L(q_i, p_i^+, p_{i,1}^-, p_{i,2}^-, \ldots, p_{i,n}^-) = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{e^{\text{sim}(q_i, p_i^+)} + \sum_{j=1}^{n} e^{\text{sim}(q_i, p_{i,j}^-)}} \tag{5.1}$$

On top of that, Karpukhin et al. [2] proposed the in-batch negative mechanism to take advantage of the computation used in extracting features from other passages within the same batch. Under the in-batch negative settings where we have a batch of B samples, in which each question q_i is associated with one positive passage p_i^+ and one hard negative passage p_i^-. We rewrite the question-passage loss function in (5.1) with respect to a single sample at index i as

$$L_i^{qp} = -\log \frac{e^{\text{sim}(q_i, p_i^+)}}{\underbrace{e^{\text{sim}(q_i, p_i^+)}}_{\text{positive}} + \underbrace{e^{\text{sim}(q_i, p_i^-)}}_{\text{hard negative}} + \underbrace{\sum_{\substack{j=1 \\ j \neq i}}^{B} e^{\text{sim}(q_i, p_j^+)} + \sum_{\substack{j=1 \\ j \neq i}}^{B} e^{\text{sim}(q_i, p_j^-)}}_{\text{in-batch negatives}}} \tag{5.2}$$

We take this direction one step further and propose to reuse the feature extraction computation of both the passages *and the questions* within the same batch.

More specifically, similar to question-passage similarity scores defined in Definition 2.4, we define the question-question similarity and passage-passage similarity scores as

$$\mathrm{sim}(q_i, q_j) = \mathbf{v}_{\texttt{[CLS]}}^{(q_i)\intercal} \mathbf{v}_{\texttt{[CLS]}}^{(q_j)} \in \mathbb{R} \tag{5.3}$$

and

$$\mathrm{sim}(p_i, p_j) = \mathbf{v}_{\texttt{[CLS]}}^{(p_i)\intercal} \mathbf{v}_{\texttt{[CLS]}}^{(p_j)} \in \mathbb{R} \tag{5.4}$$

Next, similar to the question-passage NLL function in (5.2), we define question-question and passage-passage objective functions as

$$L_i^{qq} = -\log \frac{e^{\mathrm{sim}(q_i,q_i)}}{\underbrace{e^{\mathrm{sim}(q_i,q_i)}}_{\text{in-batch positive}} + \underbrace{\sum_{\substack{j=1\\ j\neq i}}^{B} e^{\mathrm{sim}(q_i,q_j)}}_{\text{in-batch negatives}}} \tag{5.5}$$

$$L_i^{pp} = -\log \frac{e^{\mathrm{sim}(p_i,p_i)}}{\underbrace{e^{\mathrm{sim}(p_i,p_i)}}_{\text{in-batch positive}} + \underbrace{\sum_{\substack{j=1\\ j\neq i}}^{B} e^{\mathrm{sim}(p_i,p_j)}}_{\text{in-batch negatives}}} \tag{5.6}$$

where we introduce the notion of *in-batch positive* as the comparison between a question or passage against itself, which is analogous to the term *in-batch negative* [2]. Finally, we take the sum of all the question-passage, question-question and passage-passage objective components (hence the name *multi-similarity*) within the same batch as the final loss

$$L_{\text{multi\_similarity}} = \sum_{i=1}^{B} L_i^{qp} + \lambda_{qq} \sum_{i=1}^{B} L_i^{qq} + \lambda_{pp} \sum_{i=1}^{B} L_i^{pp} \tag{5.7}$$

where λ_{qq} and λ_{pp} are the coefficients of the question-question and passage-passage NLL objective functions, respectively. Not only does this objective function design efficiently reuse computation of both passages and questions, but it also effectively captures all similarity aspects within the same batch while incurring negligible computational overhead.

Architecture	**Loss function**	λ_{qq}	λ_{pp}	Top-1	Top-5	Top-20	Top-100
DPR (shared encoders)	DPR loss			**53.88**	72.49	82.38	87.62
	Multi-similarity loss	0.1	0.1	53.46	72.49	82.41	**87.87**
	Multi-similarity loss	0.5	0.5	53.21	72.47	**82.44**	87.76
	Multi-similarity loss	0.7	0.7	52.96	**72.74**	82.60	87.78

TABLE 5.1: Top-$\{1, 5, 20, 100\}$ retrieval accuracy on the Natural Questions test set of the DPR retriever (shared encoders) with and without multi-similarity loss, calculated as the percentage of top-k retrieved passages that contain the answer. The proposed multi-similarity loss mariginally improves over the DPR baseline across different loss coefficients λ_{qq} and λ_{pp}.

5.2 Experimental Results

We conduct an experiment on NQ following the experimental settings described in Section 2.3 with a batch size of 24 and different sets of multi-similarity coefficients λ_{qq} and λ_{pp}. The baseline model is taken as the DPR retriever with parameter sharing introduced in Chapter 3. Table 6.1 presents the retrieval recall on the NQ test set.

As seen from Table 6.1, the multi-similarity objective function produces a marginal improvement to the baseline DPR model with up to 0.27 points gain in performance. Interestingly, we observe that the proposed approach achieves a performance boost with top-$\{5, 20, 100\}$ while slightly degrading top-1 retrieval accuracy. We hypothesize that the multi-similarity loss has a regularization effect [46] that enforces the model to be consistent about the similarity between all pairs of texts in the corpus, with the top-1 degradation as a side product. This is evidently shown by the fact that the top-1 retrieval accuracy is decreased as we increase the values of the coefficients of the question-question and passage-passage similarities. Nevertheless, we note that the top-100 retrieval accuracy is the only metric of interest for retriever models as $k = 100$ is the actual number of passages to be retrieved during inference. Therefore we conclude that our multi-similarity objective function has a marignal, positive effect on retriever model training.

Chapter 6

Harder In-batch Negatives

6.1 Method

Recall that in DPR [2], each question in a mini-batch is assigned with a single hard negative and multiple in-batch negatives which are positive and negative passages of other question within the same batch. This can easily be understood from the reformulated equation (5.2). Karpukhin et al. [2] claimed that in-batch negative is a method that works consistently well both in their work and in previous work [47–49]. However, we argue that although this method is *efficient* in reusing computation from within the same batch, it does not necessarily provide informative in-batch negatives for an *effective* constrastive learning process. In other words, the in-batch negative passages drawn from other samples could potentially be completely irrelevant to the question and positive passage at hand, thereby producing minimal training signal to the model.

In a related work, Xiong et al. [7] showed that by using a DPR checkpoint to perform retrieval in parallel with training, a set of DPR hard negatives can be built and used for DPR training itself. This method, named ANCE [1], builds an index of negative passages considerably more informative than BM25 hard negatives, and thus was able to improve over the baseline DPR model by a substantial margin. However, this approach requires running a process in parallel with the training process that continuously builds the FAISS index and updates the retrieval results, which is extremely expensive in terms of both computation and memory. In this work we propose a better trade-off that inherits the efficiency of *in-batch negatives* and effectiveness of informative negatives without requiring

[1] **A**pproximate **N**earest neighbor negative **C**ontrastiv**E** Learning [7]

the expensive retrieval step. We term this approach *harder in-batch negatives*. Different from Xiong et al. [7], we want to improve the informativeness of in-batch negatives rather than hard negatives given its predominant presence in a mini-batch.

To achieve this goal, we train classification model on the DBpedia dataset [50] to classify document categories. More specifically, DBpedia [50] is a large-scale, structured, multilingual knowledge base extracted from Wikipedia. For our use case, it contains Wikipedia articles with their category classification with three different levels of granularity. For example, at the highest level we have a set of categories of $\{\texttt{agent}, \texttt{place}, \texttt{species}, \texttt{event}, \ldots\}$. We trained a BERT for classification [51] model to classify document categories with all three levels of granularity with multi-task training, with a performance of F1_score = $\{0.996, 0.975, 0.959\}$ for each of the three levels on the DBpedia test set. We note that the cost of this training step is amortized since it is only done once, as opposed to ANCE [7] which incurs a recurring expensive cost.

With the trained model, we then performed inference to infer the category types of each passage chunk in the Wikipedia dump introduced in Section 2.3.1, with all three classification levels. It is worth noting that we do not directly apply the category classification data from DBpedia to Wikipedia, because (1) they may use different sets of articles due to version mismatch; and (2) this classification labels are on the article level while we want it to be as granular as on the passage level. The inferred category information can then be used to construct a batch of informative in-batch negatives for DPR retrieval training. In particular, we implement a greedy algorithm that constructs training batches one by one such that the resulting positive passages within a batch share the same category classification of either the first, second or third level of granularity [2]. We also ensure a level of randomness in our algorithm so as to diversify the batch allocation across different epochs. By doing so, we are effectively creating a pool of similar passages within the same batch, generating strong training signal for the retriever model. We note that the theory of faster training convergence and higher model performance brought about by *harder* hard negatives has been well established in the literature of metric learning. We refer interested reader to Xiong et al. [7] for a comprehensive theoretical analysis on this matter.

[2] We refer interested readers to our code repository at `https://github.com/hnt4499/DPR` for the implementation details.

Architecture	**Batch construction**	Top-1	Top-5	Top-20	Top-100
DPR (shared encoders)	Random	53.41	71.24	80.66	86.90
	Harder in-batch negatives	**53.66**	**72.41**	**81.50**	**87.04**

Table 6.1: Top-$\{1, 5, 20, 100\}$ retrieval accuracy on the Natural Questions test set of the DPR retriever (shared encoders) with two different algorithms of batch construction, namely random sampling and our *harder in-batch negatives*. The retrieval accuracy is calculated as the percentage of top-k retrieved passages that contain the answer. The proposed batch construction significantly improves over the DPR baseline across different values of k.

6.2 Experimental Results

We present in Table 6.1 the experimental results on the NQ dataset with two different mini-batch construction algorithms, namely random sampling used by DPR [2] and *harder in-batch negatives* proposed in this work. The underlying architecture is set to the DPR retriever with parameter sharing as proposed in Chapter 3, and the batch size is set to 24 due to computational constraints. We observe that by using harder in-batch negatives, we are able to achieve substantial improvements over the baseline DPR model with up to 1.17 points in retrieval accuracy. This attests to the effectiveness of our proposed approach in providing informative training signal to speed up the convergence while being much computationally more efficient than ANCE [7].

We note that one could design a more sophisticated approach built on top of our proposed idea. For example, we can combine the complementary effectiveness of our proposed harder in-batch negatives with *harder hard negatives* used by ANCE [7], to obtain a general *harder negatives* training paradigm. Additionally, it is worth noting that our novel batch construction method can be applied to our novel multi-similarity objective function introducted in Chapter 5 to further boost training convergence by constructing similar question-passage, question-question and passage-passage pairs. We leave these ideas for a future work.

Chapter 7

Query Expansion

In this chapter, we describe our final yet most significant contribution, *query expansion*. First, we introduce four main lines of related research in NLP that inspire our idea, namely *attention mechanism*, *synthetic data generation*, *unsupervised pre-training* and *query augmentation*. We then move on to describe in detail our pipelined query expansion approach. Finally, we discuss experimental results in which our proposed paradigm outperforms state-of-the-art approaches in OpenQA by substantial margins.

7.1 Related Work

7.1.1 Attention Mechanism

Early work in neural machine translation [52, 53] revealed and refined a novel soft alignment technique called *attention*, which becomes a significant contribution in NLP and lays the foundation for many recent advances in natural language processing [1, 11], computer vision [54, 55] and reinforcement learning [56, 57]. The attention mechanism is proposed to mimic the excellent human capability of paying attention only to the relevant parts when presented with a piece of information (e.g., an image or a text passage). The idea is to guide the model to focus on those small but important pieces of evidence as it reads in the input sequence. This can be best illustrated using an example in Figure 7.1. Each cell M_{ij} in the attention matrix denotes attention scores of a neural machine translation model assigning to the j-th word in the source sentence when translating the i-th word in the target sentence. It is clear that most of the time, the model only looked at a

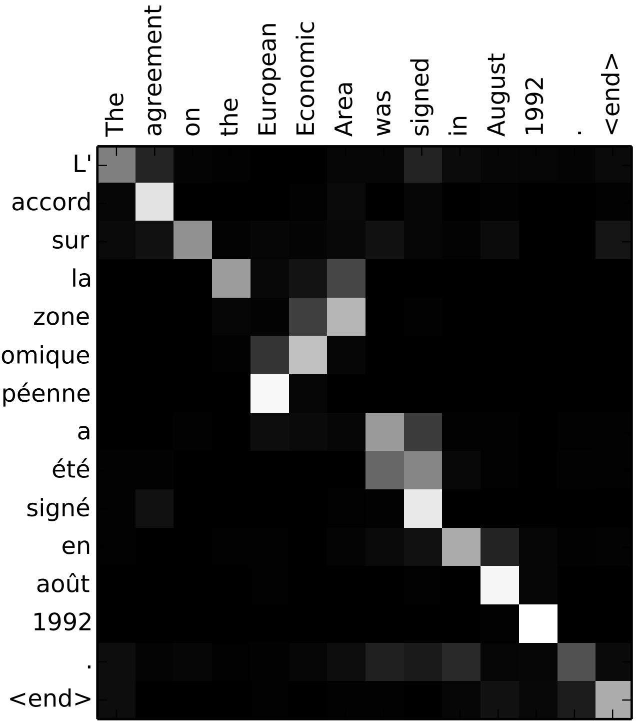


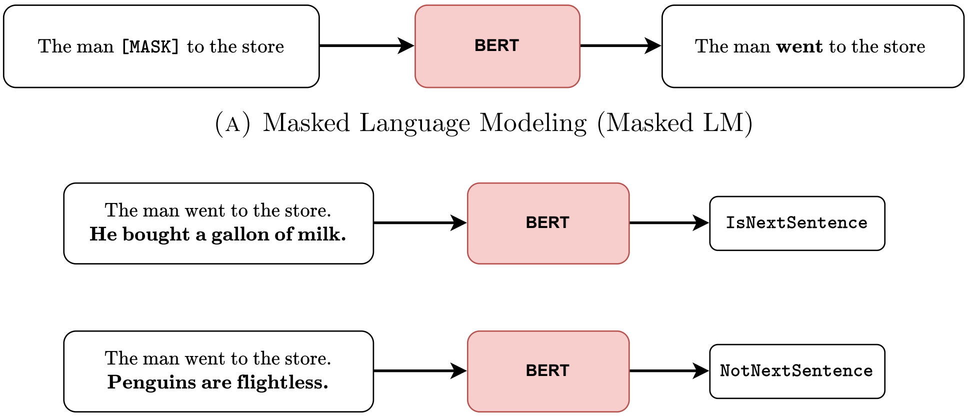
Figure 7.1: An attention matrix M of an attention-based neural machine translation model corresponding to a translation from English (x-axis) to French (y-axis). Each cell M_{ij} denotes the importance of the j-th word in the source sentence when the model is producing the i-th word in the target sentence, with white being the highest score and black being the lowest score. We can observe that most of the time the model only looked at a few source words while ignoring the rest, and that the there is a linear alignment between the source and the target texts. Interestingly, the entity *European Economic Area* was translated to *zone économique européenne* in French in a reverse direction, evidently shown by the attention matrix.

single word in the source sentence that is one-to-one translated to the words in the target sentence. By employing the attention mechanism, the model is allowed to ignore certain parts of the input sequence that are deemed irrelevant, and instead focus on relevant ones. It is important to highlight that the attention mechanism has become ubiquitous in deep learning recently due to the tremendous performance gain it brings about with little extra computational cost, and has transformed into many variants such as self-attention or cross-attention. We advance these observations one step further and propose a technique that takes advantage of cross-attention scores in an unsupervised manner to extract salient keywords from texts, which will be described later in Section 7.2.

7.1.2 Synthetic Data Generation

Synthetic data approaches aim at generating a set of artificially synthesized samples from the real, original data while trying to retain as much statistical distribution of the original data as possible. Synthetic data generation has become increasingly predominant in the deep learning research community [58–60], especially in the domains where the data is scarce such as medical treatment [61, 62]. Over the past decade, numerous synthetic data generation techniques have been proposed to tackle many aspects of machine learning, from early forms such as data augmentation [63, 64] or computer simulation [65, 66] to more recent advances such as Generative Adversarial Networks (GANs) [8, 67] or pseudo data generation [68, 69]. Existing work in OpenQA adopted synthetic answer generation and question generation to improve the performance of existing state-of-the-art models in two main ways: *unsupervised pre-training* and *query augmentation*.

7.1.3 Unsupervised Pre-training


The man [MASK] to the store
BERT
The man **went** to the store
The man went to the store.
He bought a gallon of milk.
BERT
IsNextSentence
The man went to the store.
Penguins are flightless.
BERT
NotNextSentence

(A) Masked Language Modeling (Masked LM)

(B) Next Sentence Prediction (NSP)

FIGURE 7.2: BERT [1] unsupervised training objectives. With Masked LM, the model is asked to predict words that have been masked out of the sentence. With Next Sentence Prediction, the model is asked to predict if a sentence is the grammatically and semantically possible next sentence of another sentence.

Together with the attention mechanism, *unsupervised pre-training* has become a must-have recipe in natural language processing that contributes to recent rapid development in the field. It all starts from the paper Devlin et al. [1] which proposed to pre-train deep bidirectional transformer [11] models on large unlabeled corpora of texts such as Wikipedia or Common Crawl. In unsupervised pre-training, we are supplied with huge corpora of texts that can be easily crawled from the Internet, from which the models are required to perform natural language reasoning and understanding without any supervision. This is made possible by carefully designed unsupervised objective functions, namely *Masked Language Model* (Masked LM) and *Next Sentence Prediction* (NSP). Figure 7.2 present illustrative examples of input-output pairs of the unsupervised training paradigm. In Masked LM, a portion of the input text sequence is masked out using a special token `[MASK]`, for which the model is trained to predict the correct corresponding unmasked words. On the other hand, Next Sentence Prediction takes a pair of consecutive sentences and randomly replaces the second sentence with another random sentence from the corpus. The model is then trained to classify whether the latter sentence reasonably comes after the former sentence. Importantly, unsupervised pre-training play a crucial role in many recent successes in deep learning [1, 70–72], especially in the era of big data where unlabeled data can be obtained with virtually no effort.

Synthetic data training/pre-training In OpenQA, pre-training models with synthetic corpora has attracted increasing attention from researchers recently because it allows to observe much larger, more diverse datasets to improve model generalizability and performance. Alberti et al. [73] used a sequence-to-sequence [74] model to generate synthetic questions from passages and a reading comprehension model to verify question answerability. Lewis et al. [75] sampled random nouns or entities from passages to generate a synthetic dataset of fill-in-the-blank cloze questions, which are then translated to natural questions using an unsupervised neural machine translation model. This work showed that models trained on only synthetic data are able to achieve decent performance, testifying to the usability of generated data. Lewis et al. [76] introduced a huge dataset of 65 million synthetic question-answer pairs using a sophisticated data generation pipeline. A simple QA-pair retriever model trained on this dataset is able to match the performance of a two-stage retriever-reader model while being significantly faster.

7.1.4 Query Augmentation

In OpenQA, query augmentation techiques aim at preempting the query questions with relevant information, thus are expected to improve the retrieval stage substantially. This technique is especially useful for multi-hop retrieval which requires information aggregation and complex reasoning over multiple pieces of evidence. Given an input question, instead of trying to retrieve seemingly-unrelated related documents, we can enrich the query with relevant keywords retrieved from a pre-cached parametric memory. It is important to note that to possibly incorporate all question-keywords pairs of a knowledge base to such memory, one must train a sequence-to-sequence model on a generated synthetic dataset that sufficiently covers all relationships within the knowledge base. Qi et al. [31] proposed to iteratively enrich the query with progressively obtained keywords from previous steps to tackle the task of multi-hop open-domain question answering. Mao et al. [30] demonstrated the advantages of query augmentation by performing only sparse retrieval using augmentated queries, achieving comparable performance as more computationally costly dense retrieval methods.

In this work, we take advantage of a huge synthetic question answering dataset for both synthetic data pre-training and query augmentation, which we will describe in detail in Section 7.2.

7.2 Method

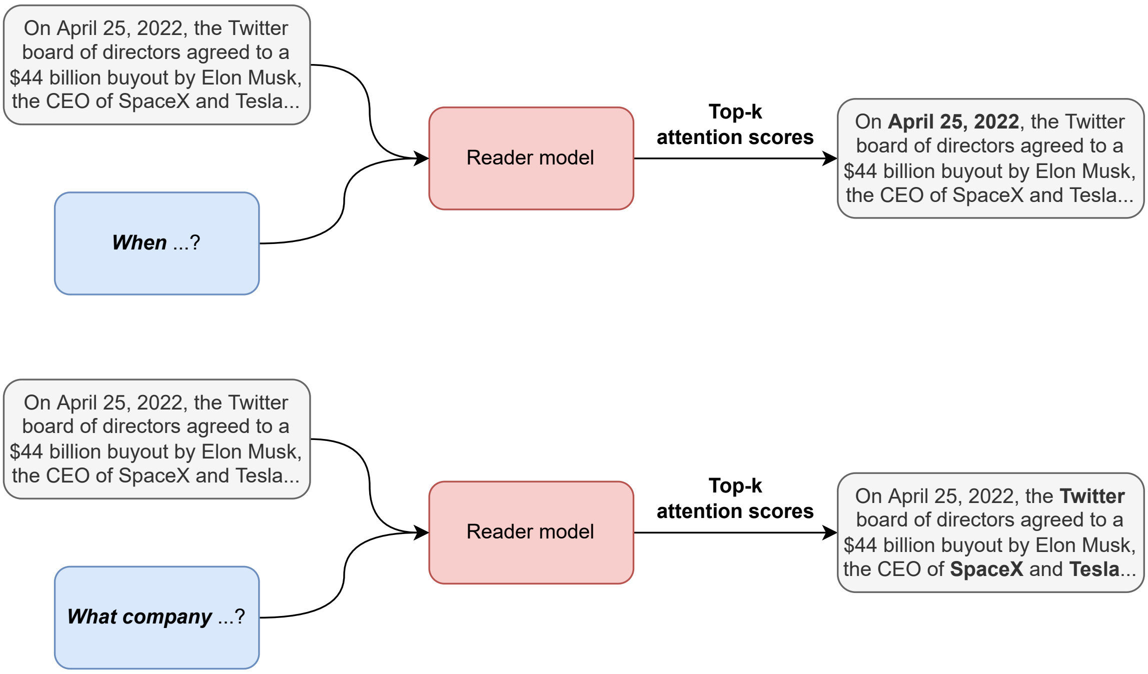

On April 25, 2022, the Twitter board of directors agreed to a $44 billion buyout by Elon Musk, the CEO of SpaceX and Tesla...
When ...?
Reader model
Top-k attention scores
On **April 25, 2022**, the Twitter board of directors agreed to a $44 billion buyout by Elon Musk, the CEO of SpaceX and Tesla...
On April 25, 2022, the Twitter board of directors agreed to a $44 billion buyout by Elon Musk, the CEO of SpaceX and Tesla...
What company ...?
Reader model
Top-k attention scores
On April 25, 2022, the **Twitter** board of directors agreed to a $44 billion buyout by Elon Musk, the CEO of **SpaceX** and **Tesla**...

FIGURE 7.3: Illustration of the idea of unsupervised keyword extraction using a trained reader model. When presented with a *when* question, the model pays most attention to timestamps in the input passage (e.g., *Apr 25, 2022*). On the other hand, the model answer to a *what company* question by attending to company names in the same input passage (e.g., *Twitter*, *SpaceX*, and *Tesla*).

7.2.1 Unsupervised Keyword Extraction

In this step, we generate salient keywords from passages in an unsupervised manner by exploiting attention scores as described in Section 7.1.1. This process is motivated by the observation that when generating an answer to a question, the generative reader model must pay attention to question type-specific tokens in the given relevant context passages. Figure 7.3 illustrates this idea, from which we observe that the model considers different sets of keywords highly depedent on the input question type.

In particular, suppose we have a large corpus $\mathcal{C}$ of K factual documents $\{p_1, p_2, \ldots, p_K\}$, e.g., Wikipedia. Furthermore, we are given a trained generative reader M which is a encoder-decoder transformer-based [11] model. When we supply the model with a wh-question word w (e.g., *who*) and a context passage $p \in \mathcal{C}$ consisting of n tokens $\{p^{(1)}, p^{(2)}, \ldots, p^{(n)}\}$, concatenated into a single sequence (e.g., *who* *[SEP]* $p^{(1)}$ $p^{(2)}$ $\ldots$ $p^{(n)}$), we can obtain the attention-based score matrices as intermediate products of the cross-attention mechanism *when the model is generating the first token in the decoder side* as

$$\mathbf{Q} \in \mathbb{R}^{1\times d}, \mathbf{K} \in \mathbb{R}^{n\times d}, \mathbf{V} \in \mathbb{R}^{n\times d} \tag{7.1}$$

where d is the feature dimension. Here we have $\mathbf{Q}$ as the query matrix in the decoder side, as well as $\mathbf{K}$ and $\mathbf{V}$ as the key and value matrices, respectively, in the encoder side. The cross-attention scores between the first token in the ouput sequence and each of the tokens in the input sequence are computed as

$$\text{Attention}(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = \text{softmax}(\mathbf{Q} \cdot \mathbf{K}^T) \cdot \mathbf{V} \in \mathbb{R}^{1\times d} \tag{7.2}$$

where softmax is the softmax activation function defined in (2.6) [1]. We refer interested readers to Vaswani et al. [11] for a complete description and discussion of attention mechanisms. Notice that we can omit the value matrix $\mathbf{V}$ and instead define

$$\text{Relevance}(\mathbf{Q}, \mathbf{K}) = \text{softmax}(\mathbf{Q} \cdot \mathbf{K}^T) \in \mathbb{R}^{1\times n} \tag{7.3}$$

as the *relevance score vector*, where the i-th value in the vector denotes how relevant the i-th input token $p^{(i)}$ is when generating the first token in the answer, which is utimately based on the given question type w. In practice, the transformer-based model [11] often comprises of multiple attention heads (so-called multi-head attention). This means that

[1] We omit the scaling factor for brevity

we can obtain M sets of query-keyword matrices $\{(\mathbf{Q}_1, \mathbf{K}_1), (\mathbf{Q}_2, \mathbf{K}_2), \ldots, (\mathbf{Q}_M, \mathbf{K}_M)\}$. We take advantage of the complementary power of these heads and compute the relevance scores by simply marginalizing over all heads as

$$\text{Relevance}(\mathbf{Q}, \mathbf{K}) = \frac{1}{M} \sum_{i=1}^{M} \text{softmax}(\mathbf{Q}_i \cdot \mathbf{K}_i^T) \in \mathbb{R}^{1\times n} \tag{7.4}$$

We then simply take tokens with the highest attention-based relevance scores based on the computed relevance vector as our *extracted keywords*

$$\text{keyword\_extraction}(w, p^{(1)}, p^{(2)}, \ldots, p^{(n)}) = \text{top\_k}(\text{Relevance}(\mathbf{Q}, \mathbf{K})) \tag{7.5}$$

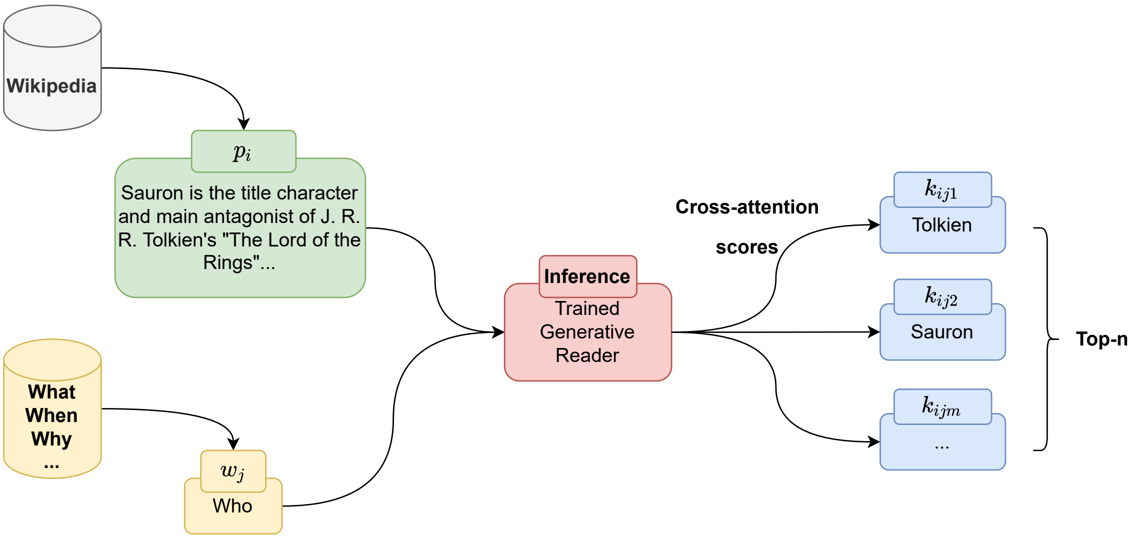

Wikipedia
p_i
Sauron is the title character and main antagonist of J. R. R. Tolkien's "The Lord of the Rings"...
What
When
Why
...
w_j
Who
Inference
Trained Generative Reader
Cross-attention scores
k_{ij1}
Tolkien
k_{ij2}
Sauron
k_{ijm}
...
Top-n

FIGURE 7.4: Illustration of the unsupervised keyword extraction pipeline. For each passage p_i in the Wikipedia corpus and each question type w_j, we generate a set of question-type-specific keywords $\{k_{ij1}, k_{ij2}, \ldots, k_{ijm}\}$ based on the cross-attention scores.

In practice, we first analyze the Natural Questions dataset to obtain a set of commonly asked question types $\{w_1, w_2, \ldots\}$. We then perform this unsupervised keyword extraction step on the entire Wikipedia corpus, obtaining a set of m keywords $\{k_{ij1}, k_{ij2}, \ldots, k_{ijm}\}$ for each combination of passage p_i and question type w_j, as illustrated in Figure 7.4.

7.2.2 Question Generation

Following existing work on synthetic question generation [73, 76–78], we train a question generator model using supervised information from the Natural Questions dataset as

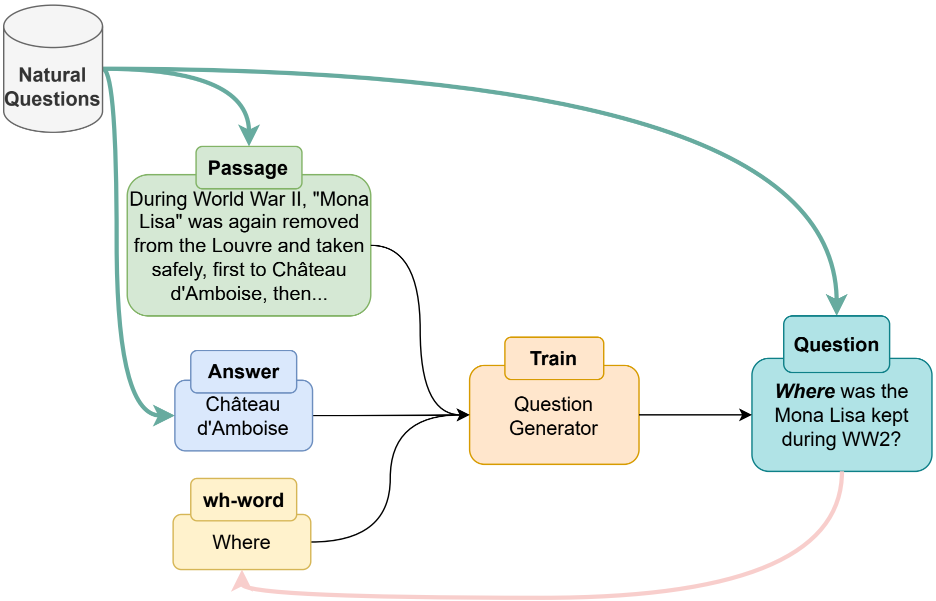

Natural Questions
Passage
During World War II, "Mona Lisa" was again removed from the Louvre and taken safely, first to Château d'Amboise, then...
Answer
Château d'Amboise
wh-word
Where
Train
Question Generator
Question
Where was the Mona Lisa kept during WW2?

FIGURE 7.5: Illustration of the question generator training pipeline. The model is trained to generate a plausible question given a context passage, an answer and a question type. Different from existing work, we infer the question type from the questions use it as one of the supervised signals.

shown in Figure 7.5. However, we also provide the model with the question types of the target questions, thereby enhancing the capability of the model at generating plausible questions with an additionaly input signal, as well as more diverse questions with various question prompts.

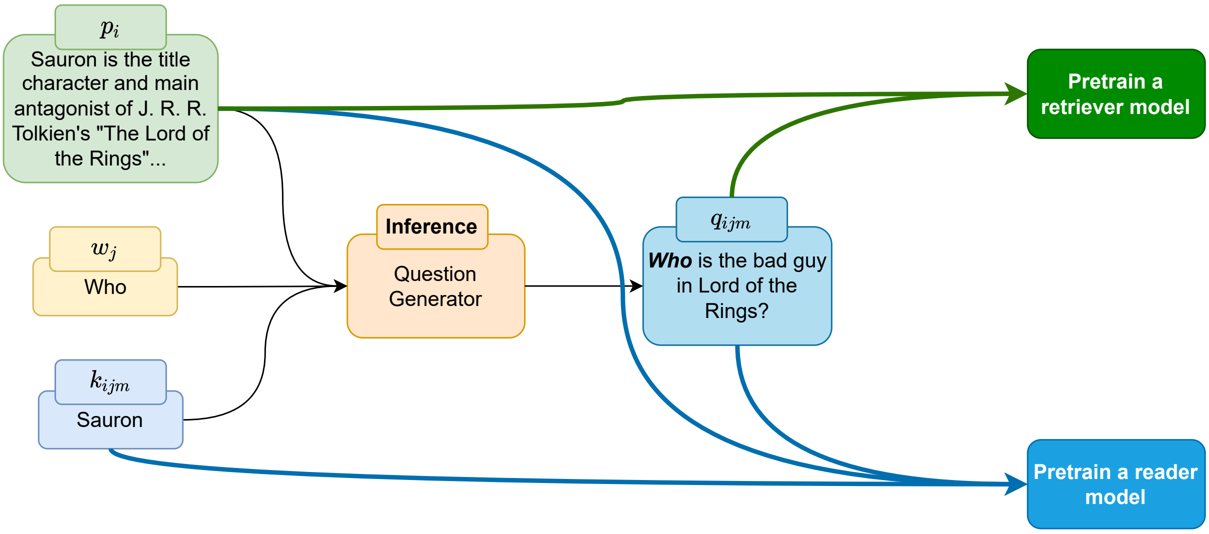
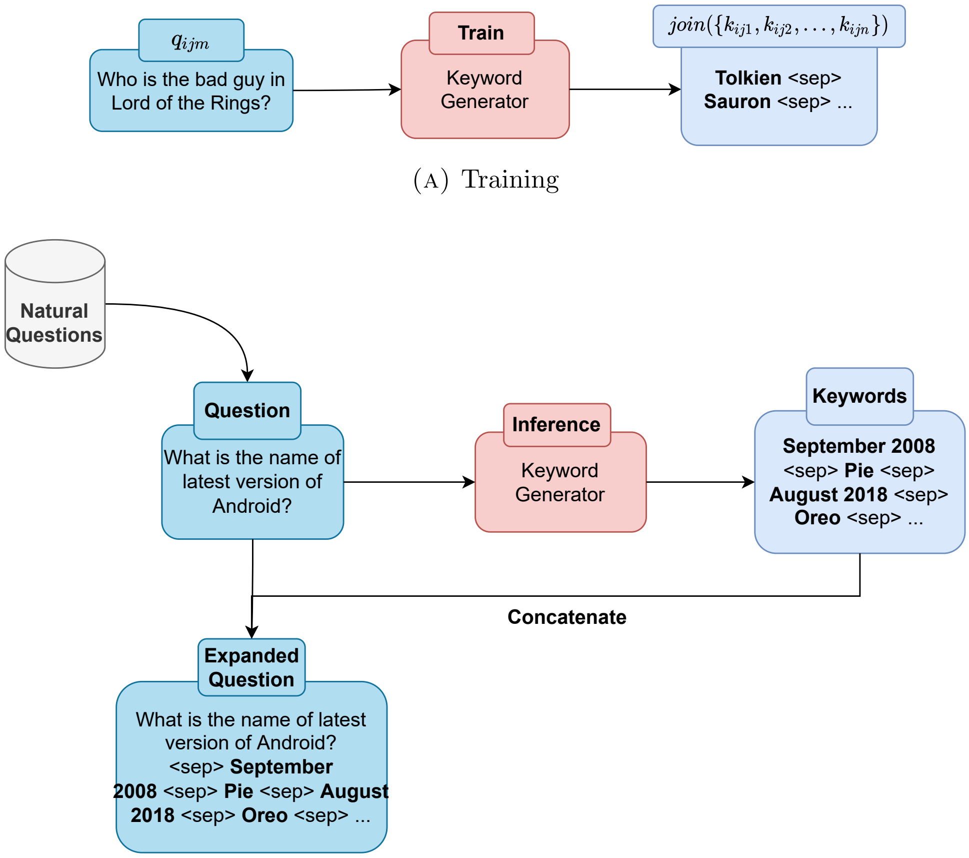

p_i
Sauron is the title character and main antagonist of J. R. R. Tolkien's "The Lord of the Rings"...
w_j
Who
k_{ijm}
Sauron
Inference
Question Generator
q_{ijm}
Who is the bad guy in Lord of the Rings?
Pretrain a retriever model
Pretrain a reader model

FIGURE 7.6: Illustration of the question generator inference pipeline. We take the trained question generator model to synthesize questions for passage-question type-keyword pairs obtained from the unsupervised keyword extraction step Section 7.2.1.

After finished, the trained question generator model will be used to infer a synthetic question q_{ijm} for each combination of passage p_i, question type w_j and keyword k_{ijm} obtained from the unsupervised keyword extraction step Section 7.2.1. This is illustrated in Figure 7.6. By doing so, we obtain an enormous, high-quality synthetic dataset of (passage p_i, question q_{ijm}, answer k_{ijm}) pairs. This synthetic dataset can be used to pre-train both the retriever and reader model in the two-stage retriever-reader framework described in Section 1.2.2, following the procedure and conventions described in Section 7.1.3 in an unsupervised manner.

7.2.3 Query Expansion


q_{ijm}
Who is the bad guy in Lord of the Rings?
Train
Keyword Generator
$join(\{k_{ij1}, k_{ij2}, \dots, k_{ijn}\})$
Tolkien <sep> **Sauron** <sep> ...
Natural Questions
Question
What is the name of latest version of Android?
Inference
Keyword Generator
Keywords
September 2008 <sep> **Pie** <sep> **August 2018** <sep> **Oreo** <sep> ...
Concatenate
Expanded Question
What is the name of latest version of Android? <sep> **September 2008** <sep> **Pie** <sep> **August 2018** <sep> **Oreo** <sep> ...

(A) Training

(B) Inference

Figure 7.7: Illustration of the query expansion training and inference pipelines.

The query expansion training and inference pipelines are illustrated in Figure 7.7.

Architecture	**Model**	Top-20	Top-100
DPR retriever [2]	Original (real)	78.4	85.4
	Pre-trained (synthetic)	70.2	81.5
	Pre-trained (synthetic) + fine-tune (real)	**80.5**	**86.7**

Table 7.1: Top-$\{20, 100\}$ retrieval accuracy on the Natural Questions test set of the DPR retriever with and without unsupervised pre-training

Architecture	**Model**	Exact Match
DPR reader [2]	Original (real)	41.5
	Pre-trained (synthetic)	28.4
	Pre-trained (synthetic) + fine-tune (real)	**44.4**

Table 7.2: Top-$\{20, 100\}$ retrieval accuracy on the Natural Questions test set of the DPR reader with and without unsupervised pre-training

Architecture	**Model**	Exact Match
FiD reader [5]	Original (real)	48.2
	Pre-trained (synthetic)	32.8
	Pre-trained (synthetic) + fine-tune (real)	**49.8**

Table 7.3: Top-$\{20, 100\}$ retrieval accuracy on the Natural Questions test set of the FiD reader with and without unsupervised pre-training

7.3 Experimental Results

We managed to implement and experiment with the processes of *unsupervised keyword extraction* (Section 7.2.1), *question generation* (Section 7.2.2) as well as unsupervised retriever and reader pre-training (Section 7.2.2).

The results are presented in Table 7.1, Table 7.2 as well as Table 7.3.

Chapter 8

Conclusion

Open-domain question answering (OpenQA), the task of answering information-seeking question about nearly anything given a large knowledge base, has attracted tremendous attention from both the research community and the industry due to its impactful applications yet under-explored challenging problems. In this final year project, our goal is to further advance the progress of current OpenQA systems by developing various mathematical-driven approaches. More specifically, we take the landmark work of Dense Passage Retrieval [2] (DPR) as the baseline, on which we perform a comprehensive error analysis to identify model weaknesses. We then propose several novel techniques that are designed to solve one model weakness at a time.

First, we introduce the simple parameter sharing idea on top of the DPR retriever in Chapter 3, which is evidently shown to be consistently and substantially more efficient and effective than the baseline. We then aim to tackle the long-standing *decomposability gap* problem of two-stage OpenQA systems by proposing a late interaction module in Chapter 4, which we show marginally degrade the model performance. Next, we take the idea of in-batch negatives [2] one step further and propose a novel objective function termed *multi-similarity loss* in Chapter 5. This loss function efficiently takes into account all question-passage, question-question and passage-passage similarities without incurring additional computational overhead, thereby improving over the baseline model by a small margin. Finally, in Chapter 6 we introduce a novel batch construction technique named *harder in-batch negatives* that groups similar samples into the same batch. By providing a much stronger, more informative signal for retrieval training, our proposed approach is able to speed up the convergence, achieving significant improvements over the DPR retriever.

Bibliography

[1] 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, 2019. v, 5, 9, 11, 14, 26, 28, 29

[2] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6769–6781, 2020. vi, vii, 3, 5, 6, 8, 10, 12, 13, 14, 15, 20, 21, 23, 25, 35, 36

[3] Jeff Johnson, Matthijs Douze, and Hervé Jégou. Billion-scale similarity search with GPUs. *IEEE Transactions on Big Data*, 7(3):535–547, 2019. vii, 5, 11, 12

[4] Sepp Hochreiter and Jürgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735–1780, 1997. vii, 5

[5] Gautier Izacard and Édouard Grave. Leveraging passage retrieval with generative models for open domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, 2021. vi, vii, 8, 35

[6] Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, et al. Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466, 2019. vii, 13

[7] Lee Xiong, Chenyan Xiong, Ye Li, Kwok-Fung Tang, Jialin Liu, Paul N Bennett, Junaid Ahmed, and Arnold Overwijk. Approximate nearest neighbor negative contrastive learning for dense text retrieval. In *International Conference on Learning Representations*, 2020. vii, 8, 23, 24, 25

[8] Ian Goodfellow, Jean Pouget-Abadie, Mehdi Mirza, Bing Xu, David Warde-Farley, Sherjil Ozair, Aaron Courville, and Yoshua Bengio. Generative adversarial nets. *Advances in neural information processing systems*, 27, 2014. vii, 28

[9] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. *Advances in neural information processing systems*, 25, 2012. 1

[10] Tomas Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean. Efficient estimation of word representations in vector space. *arXiv preprint arXiv:1301.3781*, 2013. 1

[11] 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. 1, 5, 26, 28, 31

[12] Iacopo Masi, Yue Wu, Tal Hassner, and Prem Natarajan. Deep face recognition: A survey. In *2018 31st SIBGRAPI conference on graphics, patterns and images (SIBGRAPI)*, pages 471–478. IEEE, 2018. 1

[13] Ruohan Gao and Kristen Grauman. On-demand learning for deep image restoration. In *Proceedings of the IEEE international conference on computer vision*, pages 1086–1095, 2017. 1

[14] Felix Stahlberg. Neural machine translation: A review. *Journal of Artificial Intelligence Research*, 69:343–418, 2020. 1

[15] Yogesh Kumar and Navdeep Singh. A comprehensive view of automatic speech recognition system-a systematic literature review. In *2019 international conference on automation, computational and technology management (ICACTM)*, pages 168–173. IEEE, 2019. 1

[16] B Ravi Kiran, Ibrahim Sobh, Victor Talpaert, Patrick Mannion, Ahmad A Al Sallab, Senthil Yogamani, and Patrick Pérez. Deep reinforcement learning for autonomous driving: A survey. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 1

[17] Chao Yu, Jiming Liu, Shamim Nemati, and Guosheng Yin. Reinforcement learning in healthcare: A survey. *ACM Computing Surveys (CSUR)*, 55(1):1–36, 2021. 1

[18] Danqi Chen and Wen-tau Yih. Open-domain question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 34–37, 2020. 2, 10

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

[20] Ziwei Ji, Nayeon Lee, Rita Frieske, Tiezheng Yu, Dan Su, Yan Xu, Etsuko Ishii, Yejin Bang, Andrea Madotto, and Pascale Fung. Survey of hallucination in natural language generation. *arXiv preprint arXiv:2202.03629*, 2022. 3

[21] Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, 2020. 3

[22] Ronald Kemker, Marc McClure, Angelina Abitino, Tyler Hayes, and Christopher Kanan. Measuring catastrophic forgetting in neural networks. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 32, 2018. 3

[23] Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In *ACL (1)*, 2017. 4, 5

[24] Karen Sparck Jones. A statistical interpretation of term specificity and its application in retrieval. *Journal of documentation*, 1972. 5

[25] 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, 2019. 5, 6

[26] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. Realm: Retrieval-augmented language model pre-training. *arXiv preprint arXiv:2002.08909*, 2020. 5, 6

[27] Minjoon Seo, Jinhyuk Lee, Tom Kwiatkowski, Ankur Parikh, Ali Farhadi, and Hannaneh Hajishirzi. Real-time open-domain question answering with dense-sparse phrase index. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4430–4441, 2019. 5, 6

[28] Omar Khattab and Matei Zaharia. Colbert: Efficient and effective passage search via contextualized late interaction over bert. In *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, pages 39–48, 2020. 5, 6, 7, 19

[29] Rajarshi Das, Shehzaad Dhuliawala, Manzil Zaheer, and Andrew McCallum. Multi-step retriever-reader interaction for scalable open-domain question answering. In *International Conference on Learning Representations*, 2018. 6

[30] Yuning Mao, Pengcheng He, Xiaodong Liu, Yelong Shen, Jianfeng Gao, Jiawei Han, and Weizhu Chen. Generation-augmented retrieval for open-domain question answering. In *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing (Volume 1: Long Papers)*, pages 4089–4100, 2021. 6, 30

[31] Peng Qi, Xiaowen Lin, Leo Mehr, Zijian Wang, and Christopher D Manning. Answering complex open-domain questions through iterative query generation. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 2590–2602, 2019. 6, 30

[32] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, et al. Retrieval-augmented generation for knowledge-intensive nlp tasks. *Advances in Neural Information Processing Systems*, 33:9459–9474, 2020. 6, 8

[33] Wei-Cheng Chang, Felix X Yu, Yin-Wen Chang, Yiming Yang, and Sanjiv Kumar. Pre-training tasks for embedding-based large-scale retrieval. *arXiv preprint arXiv:2002.03932*, 2020.

[34] Mike Lewis, Marjan Ghazvininejad, Gargi Ghosh, Armen Aghajanyan, Sida Wang, and Luke Zettlemoyer. Pre-training via paraphrasing. *Advances in Neural Information Processing Systems*, 33:18470–18481, 2020.

[35] Wenhan Xiong, Hong Wang, and William Yang Wang. Progressively pretrained dense corpus index for open-domain question answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 2803–2815, 2021. 6

[36] Jinhyuk Lee, Minjoon Seo, Hannaneh Hajishirzi, and Jaewoo Kang. Contextualized sparse representations for real-time open-domain question answering. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 912–919, 2020. 6

[37] Yi Luan, Jacob Eisenstein, Kristina Toutanova, and Michael Collins. Sparse, dense, and attentional representations for text retrieval. *Transactions of the Association for Computational Linguistics*, 9:329–345, 2021. 6

[38] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage bert: A globally normalized bert model for open-domain question answering. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5878–5882, 2019. 8

[39] Martin Fajcik, Martin Docekal, Karel Ondrej, and Pavel Smrz. Pruning the index contents for memory efficient open-domain qa. *arXiv preprint arXiv:2102.10697*, 2021.

[40] Gautier Izacard and Edouard Grave. Distilling knowledge from reader to retriever for question answering. In *International Conference on Learning Representations*, 2020. 8

[41] Xufeng Han, Thomas Leung, Yangqing Jia, Rahul Sukthankar, and Alexander C Berg. Matchnet: Unifying feature and metric learning for patch-based matching. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3279–3286, 2015. 10

[42] Lingxue Song, Dihong Gong, Zhifeng Li, Changsong Liu, and Wei Liu. Occlusion robust face recognition based on mask learning with pairwise differential siamese network. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 773–782, 2019. 10

[43] Jure Zbontar, Li Jing, Ishan Misra, Yann LeCun, and Stéphane Deny. Barlow twins: Self-supervised learning via redundancy reduction. In *International Conference on Machine Learning*, pages 12310–12320. PMLR, 2021. 10

[44] Jovana Mitrovic, Brian McWilliams, Jacob C Walker, Lars Holger Buesing, and Charles Blundell. Representation learning via invariant causal mechanisms. In *International Conference on Learning Representations*, 2020. 10

[45] Stephen Robertson and Hugo Zaragoza. *The probabilistic relevance framework: BM25 and beyond*. Now Publishers Inc, 2009. 12

[46] Sergey Ioffe and Christian Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning*, pages 448–456. PMLR, 2015. 22

[47] Wen-tau Yih, Kristina Toutanova, John C Platt, and Christopher Meek. Learning discriminative projections for text similarity measures. In *Proceedings of the fifteenth conference on computational natural language learning*, pages 247–256, 2011. 23

[48] Matthew Henderson, Rami Al-Rfou, Brian Strope, Yun-Hsuan Sung, László Lukács, Ruiqi Guo, Sanjiv Kumar, Balint Miklos, and Ray Kurzweil. Efficient natural language response suggestion for smart reply. *arXiv preprint arXiv:1705.00652*, 2017.

[49] Daniel Gillick, Sayali Kulkarni, Larry Lansing, Alessandro Presta, Jason Baldridge, Eugene Ie, and Diego Garcia-Olano. Learning dense representations for entity retrieval. *arXiv preprint arXiv:1909.10506*, 2019. 23

[50] Jens Lehmann, Robert Isele, Max Jakob, Anja Jentzsch, Dimitris Kontokostas, Pablo N Mendes, Sebastian Hellmann, Mohamed Morsey, Patrick Van Kleef, Sören Auer, et al. Dbpedia–a large-scale, multilingual knowledge base extracted from wikipedia. *Semantic web*, 6(2):167–195, 2015. 24

[51] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, et al. Huggingface's transformers: State-of-the-art natural language processing. *arXiv preprint arXiv:1910.03771*, 2019. 24

[52] Minh-Thang Luong, Hieu Pham, and Christopher D Manning. Effective approaches to attention-based neural machine translation. In *Proceedings of the 2015 Conference on Empirical Methods in Natural Language Processing*, pages 1412–1421, 2015. 26

[53] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio. Neural machine translation by jointly learning to align and translate. *arXiv preprint arXiv:1409.0473*, 2014. 26

[54] Alexey Dosovitskiy, Lucas Beyer, Alexander Kolesnikov, Dirk Weissenborn, Xiaohua Zhai, Thomas Unterthiner, Mostafa Dehghani, Matthias Minderer, Georg Heigold, Sylvain Gelly, et al. An image is worth 16x16 words: Transformers for image recognition at scale. In *International Conference on Learning Representations*, 2020. 26

[55] Ze Liu, Yutong Lin, Yue Cao, Han Hu, Yixuan Wei, Zheng Zhang, Stephen Lin, and Baining Guo. Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 10012–10022, 2021. 26

[56] Yujin Tang, Duong Nguyen, and David Ha. Neuroevolution of self-interpretable agents. In *Proceedings of the 2020 Genetic and Evolutionary Computation Conference*, pages 414–424, 2020. 26

[57] Xiangxiang Shen, Chuanhuan Yin, and Xinwen Hou. Self-attention for deep reinforcement learning. In *Proceedings of the 2019 4th International Conference on Mathematics and Artificial Intelligence*, pages 71–75, 2019. 26

[58] Qi Wang, Junyu Gao, Wei Lin, and Yuan Yuan. Learning from synthetic data for crowd counting in the wild. In *Proceedings of the IEEE/CVF Conference on computer vision and pattern recognition*, pages 8198–8207, 2019. 28

[59] Shashank Tripathi, Siddhartha Chandra, Amit Agrawal, Ambrish Tyagi, James M Rehg, and Visesh Chari. Learning to generate synthetic data via compositing. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 461–470, 2019.

[60] Alexandre Lacoste, Pau Rodríguez López, Frédéric Branchaud-Charron, Parmida Atighehchian, Massimo Caccia, Issam Hadj Laradji, Alexandre Drouin, Matthew Craddock, Laurent Charlin, and David Vázquez. Synbols: Probing learning algorithms with synthetic datasets. *Advances in Neural Information Processing Systems*, 33:134–146, 2020. 28

[61] Richard J Chen, Ming Y Lu, Tiffany Y Chen, Drew FK Williamson, and Faisal Mahmood. Synthetic data in machine learning for medicine and healthcare. *Nature Biomedical Engineering*, 5(6):493–497, 2021. 28

[62] Maayan Frid-Adar, Eyal Klang, Michal Amitai, Jacob Goldberger, and Hayit Greenspan. Synthetic data augmentation using gan for improved liver lesion classification. In *2018 IEEE 15th international symposium on biomedical imaging (ISBI 2018)*, pages 289–293. IEEE, 2018. 28

[63] Connor Shorten and Taghi M Khoshgoftaar. A survey on image data augmentation for deep learning. *Journal of big data*, 6(1):1–48, 2019. 28

[64] Toan Tran, Trung Pham, Gustavo Carneiro, Lyle Palmer, and Ian Reid. A bayesian data augmentation approach for learning deep models. *Advances in neural information processing systems*, 30, 2017. 28

[65] Stephan R Richter, Vibhav Vineet, Stefan Roth, and Vladlen Koltun. Playing for data: Ground truth from computer games. In *European conference on computer vision*, pages 102–118. Springer, 2016. 28

[66] Shuran Song, Fisher Yu, Andy Zeng, Angel X Chang, Manolis Savva, and Thomas Funkhouser. Semantic scene completion from a single depth image. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 1746–1754, 2017. 28

[67] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8110–8119, 2020. 28

[68] Eric Arazo, Diego Ortego, Paul Albert, Noel E O'Connor, and Kevin McGuinness. Pseudo-labeling and confirmation bias in deep semi-supervised learning. In *2020 International Joint Conference on Neural Networks (IJCNN)*, pages 1–8. IEEE, 2020. 28

[69] Qian Wang and Toby Breckon. Unsupervised domain adaptation via structured prediction based selective pseudo-labeling. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 6243–6250, 2020. 28

[70] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020. 29

[71] Mathilde Caron, Ishan Misra, Julien Mairal, Priya Goyal, Piotr Bojanowski, and Armand Joulin. Unsupervised learning of visual features by contrasting cluster assignments. *Advances in Neural Information Processing Systems*, 33:9912–9924, 2020.

[72] Priya Goyal, Mathilde Caron, Benjamin Lefaudeux, Min Xu, Pengchao Wang, Vivek Pai, Mannat Singh, Vitaliy Liptchinsky, Ishan Misra, Armand Joulin, et al. Self-supervised pretraining of visual features in the wild. *arXiv preprint arXiv:2103.01988*, 2021. 29

[73] Chris Alberti, Daniel Andor, Emily Pitler, Jacob Devlin, and Michael Collins. Synthetic qa corpora generation with roundtrip consistency. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 6168–6173, 2019. 29, 32

[74] Ilya Sutskever, Oriol Vinyals, and Quoc V Le. Sequence to sequence learning with neural networks. *Advances in neural information processing systems*, 27, 2014. 29

[75] Patrick Lewis, Ludovic Denoyer, and Sebastian Riedel. Unsupervised question answering by cloze translation. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4896–4910, 2019. 29

[76] Patrick Lewis, Yuxiang Wu, Linqing Liu, Pasquale Minervini, Heinrich Küttler, Aleksandra Piktus, Pontus Stenetorp, and Sebastian Riedel. Paq: 65 million probably-asked questions and what you can do with them. *Transactions of the Association for Computational Linguistics*, 9:1098–1115, 2021. 29, 32

[77] Ying-Hong Chan and Yao-Chung Fan. A recurrent bert-based model for question generation. In *Proceedings of the 2nd Workshop on Machine Reading for Question Answering*, pages 154–162, 2019.

[78] Md Arafat Sultan, Shubham Chandel, Ramón Fernandez Astudillo, and Vittorio Castelli. On the importance of diversity in question generation for qa. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 5651–5656, 2020. 32