Detecting Word Transfer in Open Domain Question Answering

by

Mostafa Yadegari

A thesis submitted in partial fulfillment of the requirements for the degree of

Master of Science

Department of Computing Science

University of Alberta

Abstract

There is essential information in the underlying structure of sentences and the relationships between words and phrases in natural language questions, and the use of this information has been extensively studied. This thesis studies the problem of word transfer from questions to answer passages in the context of open-domain question answering. Word transfer happens for both terms that are explicitly mentioned in questions and those that may be implied. On the same basis, this thesis is broken down to two parts. In the first part of the thesis, we study one particular structure, referred to as *frozen phrases*, that is highly expected to transfer as a whole from questions to answer passages. Frozen phrases, if detected, can be helpful in open-domain Question Answering (QA) where identifying the localized context of a given input question is crucial. To identify those phrases, we cast the problem as a sequence-labeling task and create synthetic data from existing QA datasets to train a model. We further plug this model into a sparse retriever that is made aware of the detected phrases. Our experiments reveal that detecting frozen phrases whose presence in answer documents are highly plausible yields significant improvements in retrievals as well as in the end-to-end accuracy of open-domain QA models. In the second part, a query expansion method is introduced to predict the terms that fall outside of a question but are expected to be found in the answer passages. For this task, we explore the capacity of modern language models under few-shot in-context learning. Our evaluation reveals that the proposed method is quite effective, achieving a new state-of-

the-art unsupervised query expansion on Natural Questions dataset.

Preface

The main skeleton of this thesis are based on papers that are either published or is about to be submitted. In particular, Chapter 3 is written based on the published paper provided below. Chapter 4 is based on a paper that is about to be submitted to the CIKM 2023 conference.

1. Yadegari, Mostafa, Ehsan Kamalloo, and Davood Rafiei. "Detecting Frozen Phrases in Open-Domain Question Answering." Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval. 2022.

To my amazing family for their unconditional support

Acknowledgements

First and foremost, I would like to express my deepest gratitude to my supervisor, Professor Dr. Davood Rafiei, for his continuous support, and his invaluable mentorship. He has patiently endured my mistakes and has never thought twice to dedicate his time to me whenever I needed it. I cannot imagine doing this thesis without Davood's help and support.

I would like to thank Ehsan Kamalloo for his time and help. He shared his knowledge and experience generously. I really enjoyed collaborating with him especially since he is an amazing friend as well.

Last but not least, I wish to thank my committee members: Dr. Lili Mou and Dr. Denilson Barbosa for taking the time to attend and providing valuable comments.

Contents

List of Tables

List of Figures

Chapter 1

Introduction

1.1 Motivation

In this section, we discuss two scenarios where retrieving the answer passage fails using simple bag-of-word models.

1.1.1 Scenario 1

Table 1.1 demonstrates an example where the question "When was the last time anyone was on the moon" is given to a retriever. Commonly used retrievers such as TF-IDF and BM25 [45], [78] will assign higher scores to the passage on the left column compared to the ground truth passage shown on the right column. The matching terms in both passages are highlighted. In practice, the retriever only looks up six uni-gram and bi-gram terms ("last", "time", "anyone", "moon", "last time", "time anyone") because other question terms are treated as stop words and are usually removed. The Inverse Document Frequency (IDF) [1] of those query terms that used for the retrieval, computed on our Wikipedia corpus, are as follows:

$$IDF(\text{“}last\text{”}) = 3.3,\ IDF(\text{“}time\text{”}) = 2.0,$$

$$IDF(\text{“}anyone\text{”}) = 5.9,\ IDF(\text{“}moon\text{”}) = 5.7,$$

$$IDF(\text{“}last\ time\text{”}) = 7.2,\ IDF(\text{“}time\ anyone\text{”}) = 11.7,$$

[1] logarithmic scaled of number of total documents divided by the number of documents containing the word

Table 1.1: The top retrieved passage (left) vs. the answer passage (right) for the question "When was the last time anyone was on the moon".

Top Passage	**Answer Passage**
an eye for the ladies, but is willing to stab anyone in the back. He has a relationship with Roxy Mitchell (Rita Simons), which is disliked by Roxy's sister Ronnie (Samantha Womack), and marries Janine Butcher (Charlie Brooks). In March 2013, it was announced that Shepherd decided to leave the role. Michael's last episode is on 1 November 2013, when he is killed by Janine. In reality, Shepherd quit to explore new roles. Michael arrives in Albert Square to lend money to his cousin Alfie Moon (Shane Richie) for the lease of The Queen Victoria public house. Alfie's wife, Kat	have landed on the Moon. This was accomplished with two US pilot-astronauts flying a Lunar Module on each of six NASA missions across a 41-month period starting on 20 July 1969 UTC, with Neil Armstrong and Buzz Aldrin on Apollo 11, and ending on 14 December 1972 UTC with Gene Cernan and Jack Schmitt on Apollo 17. Cernan was the last to step off the lunar surface. All Apollo lunar missions had a third crew member who remained on board the Command Module. The last three missions had a rover for increased mobility. In order to go to the Moon,

The query terms that used of the retrieval in the above example are mostly common words, and the term "anyone" has the highest IDF among uni-gram question terms. Our retrievers will assign a higher score to the passage that contains "anyone" (shown on the left column of the table), but the retrieved passage is irrelevant to the meaning of the question. However, one could easily predict some terms that are expected to be found in the answer passage of the question. The words "NASA," "astronauts," and "mission" are some of the terms that one can expect to see in the answer passage. Looking at the answer passage, we can see that adding those predicted terms can significantly increase the score assigned to the answer passage. To address this issue, which is known as vocabulary mismatch, our work expands questions by predicting passage terms from the given question.

1.1.2 Scenario 2

For our second scenario, suppose we are using an existing query expansion (QE) method. There are questions where applying a simple QE method on

them will lead to worse results. For instance, consider the question "who sang it must have been love but its over now," which has only three terms with an IDF of more than zero and the rest of uni-gram and bi-gram terms are ignored by the retriever. The IDF scores of the three terms are as follows:

$$IDF(\text{“}sang\text{”}) = 6.0,\ IDF(\text{“}must\text{”}) = 4.2,$$

$$IDF(\text{“}love\text{”}) = 4.1,$$

As our QE method, suppose we append 10 paraphrases of the question to itself. We can obtain those 10 paraphrases from T5 [63] paraphrase generator. The obtained paraphrases are:

1. Who sang it must have been love, but its over now?
2. Who sang 'I do' it must be 'I do' but its over now?
3. Who sang it must be love but its over now, a sad song?
4. Who sang it must have been love but its over now its over now its over now now its over.
5. Who sang it must have been love and how can it be ended now?
6. Who sang it must have been love but its over now, its over?
7. Who sang it must have been love but its over now.
8. Who sang it must have been love. its all over now, its over, its over now, it must have been love who sang it must have been love and its all over now.
9. Is it a cry of love?
10. Why was it true or was it simply love but the ending was over now?

The term "sang" is not part of the song name that we need which is the most important part of the question. However, it has appeared excessively in the expanding terms, and it has the highest IDF among the original query

Table 1.2: The top ranked passage for the question "who sang it must have been love but its over now" after a query expansion vs. the answer passage.

Top Passage	**Answer Passage**
album, had some legal wrangles regarding its copyright and track title which have now been resolved. The album includes three newly recorded cover versions of hit songs: "Sometimes When We Touch," originally sang by Dan Hill; "When I Need You," originally sang by Leo Sayer; and "For the First Time," originally sang by Kenny Loggins. Two other songs had not been previously released on a Rod Stewart album: "So Far Away", originally by Carole King, which had been released as a single in 1995 from that year's Carole King tribute album, "", and "All for Love," sang with Bryan Adams	t.A.T.u., Lena Katina and Julia Volkova. Both Lena and Julia knew each other before the auditions. Both girls stood out among the others, especially because of their appearance and vocal experience, but the producers decided to start with 14-year-old Katina, who sang "It Must Have Been Love" by Roxette. Katina began recording demos, including "Yugoslavia", a protest song about NATO bombing of Yugoslavia. After the demos were cut, Shapovalov insisted that another girl be added to the project. Thus, in late 1999, 14-year-old Julia Volkova was added to the group to complete the duo. She also started recording not long

terms. Table 1.2 shows a top passage retrieved by BM25 retriever based on the expanded question as well as the answer passage. The passage shown on the left column of the table contains multiple occurrences of the term "sang" and has received a higher score than the actual answer passage. We want to resolve this issue by detecting the invariant parts of the question to avoid query expansion methods from diminishing the retrieval accuracy.

1.2 Background

Open-domain Question Answering (OpenQA) aims at answering factoid questions over an enormous collection of text documents. Recent OpenQA models often follow a two-stage framework that consists of a retriever to find candidate documents, and a reader to extract answers from retrieved candidates [10], [35].

Retrieval has undoubtedly a profound role in OpenQA mainly because the

overall performance of the pipeline is arguably bounded by the performance of the retriever component [39], [54], [82]. Dense and sparse retrievers are the two types that modern methods leverage in their retrieval stage. Dense retrievers leverage neural networks and map the given query to a point in a high dimensional space where other corpus passages are also mapped to. Passages are ranked based on their similarity to the query in the embedding space. Sparse retrievers compare the query terms with the passage terms verbatim using a bag-of-word model. Sparse methods are unsupervised methods that can get reasonable results for day-to-day applications without demanding too much computational cost. Although dense retrievers tend to have a higher accuracy than sparse retrievers in general, sparse retrievers are more computationally efficient. This makes sparse retrievers a good choice when there are constraints on computational resources that are available.

The reader module takes candidate passages from a retriever and is responsible for producing the answer. It is worth mentioning that regardless of the type of the retriever, one can use either a generative or an extractive reader to provide the final answer. Generative readers are sequence-to-sequence models that generate the answer given the question and candidate passages. In contrast, extractive readers are designed to extract some part of the candidate passages as the final answer. Figure 1.2 illustrates the workflow of a modern end-to-end QA pipeline.

Query expansion (QE) is an indispensable part of the retrieval process [58], especially for sparse methods. In query expansion, some additional terms are added to the original query to tackle vocabulary mismatch[17], [85]. In other words, query expansion adds some terms to the question before passing the question for retrieval with the hope of increasing the chance of retrieving the answer passage.

Expanding a query does not always help the retrieval of the answer passages. Sometimes the relevance score between and expanded question and the answer passage is less than the relevance between the original question and the answer passage. In that case, we are having query drift which aggravates the retriever performance on the query [17]. Table 1.3 demonstrates an example

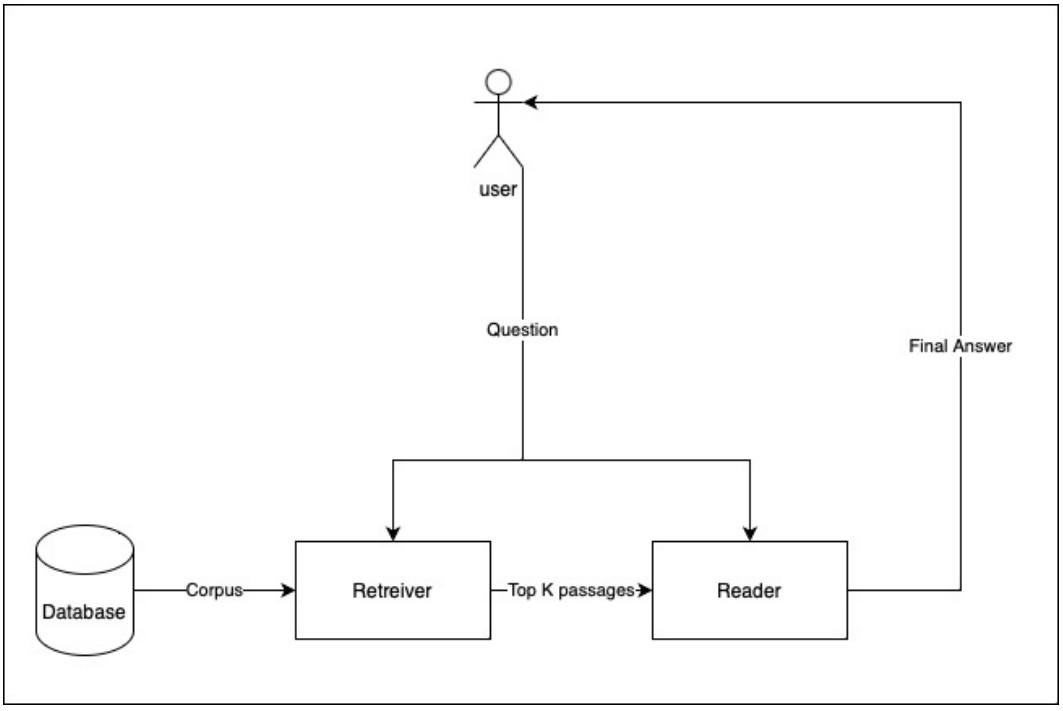

Figure 1.1: Workflow of the retriever-reader approaches in the task of question answering

Table 1.3: Examples of two situations where query expansion is helpful, and where it is harmful for retrieving the query "When was Earth released in India."

Helpful	**Harmful**
When was Earth released in India film movie Bollywood drama	When was Earth released in India Moon Sun China Pakistan

for two query expansion cases. The question "When was Earth released in India" is asking about a movie named "Earth" which was released in India. In the helpful case, the hypothetical query expansion (QE) method has successfully understood that the question is asking about the release date of an Indian movie. However, in the harmful case, the QE cannot understand that the question is about a movie, and the expanded query terms are related to the planet earth (Moon and Sun) and the country India (and the neighbouring countries China and Pakistan).

Query drift can be addressed by **query re-weighing** which changes the

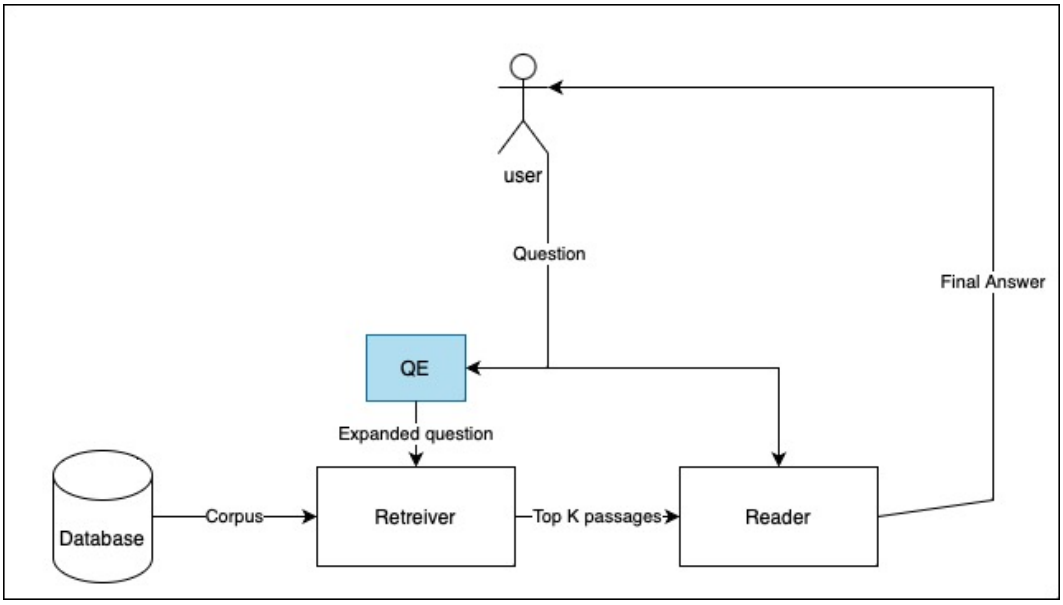

Figure 1.2: Workflow of the retriever-reader approaches in the task of question answering when they apply query expansion to the retriever

weights of query terms in the sparse representation [7], [17], [34]. Query re-weighing can be considered as a query expansion method in which only the terms in the original question can be generated and added to the question. Table 1.4 shows an example where re-weighing helps with the problem of query drift. The terms that expand the queries are highlighted based on their helpfulness. Given the red query, the re-weighing method successfully detects that the words "Earth," "India," and "released" are more important than other words. So it increases the weights of those words resulting in a query vector closer to the information need. Additionally, query re-weighing can make further adjustments to green query terms as shown in Table 1.4.

1.3 Problems Studied

Although a large body of work is dedicated to query expansion and query reformulation [8], [50], [83], there are still some challenges to this crucial element of retrieval, as shown in our earlier examples. There are situations where a given query does not have the terms required for the sparse retriever to find the answer passage. For example, the query "When was the last time anyone

Table 1.4: Effect of query re-weighting on the corresponding vector of red (where QE is harmful) and green (where QE is helpful) queries.

Terms	Red query	Red query re-weighted	Green query	Green query re-weighted
Bollywood	0	0	1	5
China	1	1	0	0
drama	0	0	1	2
Earth	1	5	1	5
film	0	0	1	5
in	1	1	1	1
India	1	5	1	5
moon	1	1	0	0
movie	0	0	1	5
Pakistan	1	1	0	0
released	1	5	1	3
sun	1	1	0	0
was	1	1	1	1
when	1	1	1	1

Table 1.5: Using query expansion as query re-weighting

	Original	**Expanded**
Query	earth in india released was when	earth earth earth in india india inida released released was when
Vector	[1, 1, 1, 1, 1, 1]	[3, 1, 3, 2, 1, 1]

was on the moon" is full of common words, and there are plenty of documents containing most of the words in the query. Many supervised query expansion methods have tackled the problem, but providing the training data and adapting to new domains is their disadvantage.

In another setting, queries have an adequate number of answer passage words, while the weights of the query words in the sparse representation are not reflecting the information need. Therefore the retriever is not capable of retrieving the answer passage. For instance, Table 1.4 demonstrates an example of such setting. Moreover, query expansion and augmentation methods are prone to aggravate the retriever performance for some queries (red queries), even when the methods are helpful for most of the dataset queries

(green queries) [15]. It has been acknowledged that there exists a trade-off between the number of red and green queries that exhibits the imperfection of query augmentation methods [15], [28]. Query re-weighing methods usually require modifications to the retriever scoring function or the index [4], [20], [86]. Furthermore, providing the training data for training such model is also challenging.

1.4 Thesis Statement

We hypothesize that pre-trained language models are effective in addressing vocabulary mismatch because of their awareness of the context in which terms occur. Furthermore, we argue that identifying the structure of terms, referred to as frozen phrases, provides invariant parts of a question. Our second research hypothesis is that detecting frozen phrases reduces query drift and that the performance of query expansion can be improved by avoiding or reducing query drift.

1.5 Contribution

In this research, we introduce a combination of query expansion and query re-weighting methods to address query mismatch and query drift at the same time. In the following two sub-sections we discuss our contributions in solving each one of the mentioned problems.

1.5.1 Vocabulary mismatch

We propose a zero-shot query expansion method leveraging a pre-trained large language model (LLM) [5]. We show that useful information can be collected about questions through some designated prompts asking LLM about different aspects of questions. Our query expansion method requires a minimal implementation, and it is effective in enhancing retrievals. Our experiments reveal that the enhancement is propagated to the reader, thus the end-to-end performance of question answering is improved.

Our LLM (GPT3 [5]) is an enormous language model trained on a large corpus of documents. The pre-trained model is so knowledgeable that it can be used directly to answer dataset questions. However, we aim at using the LLM knowledge only during the retrieval stage and achieve a better overall performance. The better performance is achieved mainly because LLM cannot answer questions on facts they have not seen during training. For example, an LLM trained last year cannot answer questions on the events that happened after. Our retrieval and end-to-end OpenQA results on the expanded questions suggest a new unsupervised state-of-the-art baseline for the QE task. We managed to close the gap between supervised and unsupervised query expansion by surpassing some prominent supervised QE methods [35], [50] in terms of both retrieval and end-to-end OpenQA performance.

Furthermore, we establish two effective techniques that boost the quality of expanding terms generated by a large language model. By multiple sampling from the language model, we show that the probability of each expanding term will be reflected in the number of times the term is generated, and the error produced in a sample will be normalized with this technique. Furthermore, we introduce a separate normalizing technique that does not require sampling and normalize the expanded questions more efficiently. At the end, we demonstrate the positive effect of combining these two techniques in retrieval and question answering.

We use multiple prompts for each question to get more information about the question and to reduce noise at the same time. Our zero-shot learning approach does not require any training or even tuning. In contrast to the simplicity of the proposed method, it effectively improves the performance of both retrieval and end-to-end of question answering tasks. Our query expansion method generates comparable results to GAR [50], one of the recent preeminent query expansion methods, when it is used without frozen phrase detection.

1.5.2 Query Drift

In this work, we study a particular structure, referred to as "frozen phrases," which is highly expected to transfer *as a whole* from questions to answer passages. Detecting such structures has major implications in retrieval, especially in OpenQA. Frozen phrases are introduced and studied to address the shortcomings of query expansion methods while using sparse retrieval models such as BM25, which have been a popular choice as a retriever [10], [11], [70], [79]. However, it is shown in this thesis that using frozen phrases improves the retrieval results even when no query expansion method is used.

Sparse representations are not designed to reflect the importance of phrases in the question vector. Consider the question "*Who wrote the country song I Can Only Imagine?*," taken from a well-known OpenQA dataset, Natural Questions-open [41]. "*I Can Only Imagine*" is the name of a song that has a high chance of matching verbatim in an answer document although its terms might even have lower IDFs than that of the rest of the question "*Who wrote the country song.*"

We characterize such phrases that are expected to appear in the target document as exactly as they are written in the question as Frozen Phrases. The terms of a frozen phrase are extremely likely to be seen together in the target document, no matter what their TF-IDF scores are in the question vector, and the ordering of the terms is expected to match closely.

For instance, in the question "*who said one man's vulgarity is another's lyric*" [38], the phrase "*one man's vulgarity is another's lyric*" is a famous quote, which is a frozen phrase in the question. However, not all frozen phrases are expected to be helpful in retrieval. In the above example, the phrase "*who sings*" may also be a frozen phrase, but that phrase is less likely to be helpful because it probably appears in many arbitrary document.

Detecting frozen phrases can be helpful in many applications including query expansion [13], [50], [87] and question clustering [23], [32]. In query expansion, adding more terms to a question generally shrinks the weights of the terms in the original question including those of a frozen phrase, and this

can negatively impact the retrievals. If the frozen phrases can be detected, query expansion can be better guided to leave the invariant parts of a question unchanged.

Detecting frozen phrases and their types (e.g. a song lyric) in a question can also provide more insight about the focal point of the question, which can help with further question classification.

In this thesis, we tackle the task of detecting frozen phrases in questions using a transformer based model [69]. A major challenge in training one such classifier is the absence of large enough annotated training data; hence, we propose an algorithm to automatically generate the training data based on existing QA corpora where questions, answer documents, and corpus statistics — e.g., TF, and IDF — are readily available.

To evaluate the performance of our proposed frozen phrase detection approach and the quality of our generated training set, we train our transformer based model on the generated dataset and predict frozen phrases in an unseen test set using the trained model. The queries in the test set are expanded by the detected frozen phrases and are retrieved by a sparse retriever. Our empirical results show a significant performance boost that effectively underscores the usefulness of our devised strategy in identifying frozen phrases. Our code and data are released at `https://github.com/Aashena/Frozen-Phrases`.

1.5.3 Contribution Summary

Our contributions can be summarized as follows:

1. We introduce a new state-of-the-art unsupervised query expansion that closes the gap between supervised and unsupervised techniques.

2. We propose two techniques that improve the overall query expansion performance through multiple sampling and normalizing the expanding term weights.

3. Through our experiments, we show how large language model can be effectively deployed for query expansion.

4. We introduce frozen phrases to capture the invariant parts of a question and show its importance in question answering.

5. We propose an algorithm to extract frozen phrases from a QA dataset that has answer documents, allowing us to create a training data from a QA corpus.

6. Through our evaluation, we show that frozen phrases can be successfully detected using our generated training data, and our model is able to improve upon the retriever in question answering.

1.6 Outline

The remainder of this thesis is organized as follows: Chapter 2 discusses the related work. Chapter 3 is dedicated to our method and experiments that tackles vocabulary mismatch. In chapter 4, we discuss our method and evaluation resolving query drift in QE methods. Chapter 5 concludes the thesis and proposes some future research directions.

Chapter 2

Related Work

Our work is related to the lines of work on (1) OpenQA retrieval, (2) query expansion and reformulation, (3) query re-weighting, and (4) query decomposition. This chapter reviews the relevant research in these areas that closely relate to our ours.

2.1 Retrieval in OpenQA

The task of OpenQA, as described in Section 1.2, differs from closed-domain question answering in both the size and the diversity of the corpus they are applicable to (e.g., medical, legal, and COVID domains are some of the common specialized domains). Unlike closed-domain, OpenQA corpus is not limited to a specific domain or corpus. Sparse retrieval models such as TF-IDF and BM25 have been widely adopted in both early multi-stage OpenQA pipelines [12], [14], [24] and modern retriever-reader models [10], [11], [70], [72], [79]. Early multi-stage OpenQA models include a question processing stage to extract information from the given question, question reformulation module for query optimization, search engine to find relevant documents based on the given query, and post-processing to find the final answer in the relevant documents.

Multiple retriever-reader systems [52], [73], [80] employ sparse retrieval models to locate documents. However, when a vocabulary mismatch occurs, these models are not very effective. To address this, three strategies are used: (1) expanding the document or question [50], [57], (2) re-ranking the results

[40], [55], and (3) leveraging dense retrieval [35], [36], [42]

Chen *et al.* propose DrQA [10] as one of the pioneer question answering systems among modern retriever-reader models. They use a sparse retriever and a recurrent neural network as reader to tackle the OpenQA task. Later work leverage this idea and make noticeable improvements to the model. For instance, Yang *et al.* develop BERTserini [79] in which a BERT based model is used as reader and Anserini toolkit as retriever.

Sparse retrievers often suffer from the so-called *vocabulary mismatch* bottleneck [46]. As a remedy, several models [39], [54] offer an intermediate stage to re-rank the initial retrieved results via a neural model.

Instead of re-ranking, some authors make use of neural nets in the retriever. Wang *et al.* introduce a BERT based retriever that gains improvements by utilizing generalizations made by a deep neural network [72]. By learning those generalizations one could tackle the problem of vocabulary-mismatch. More recently, dense retrieval models [35], [37], [41], [61], [75] and retrieval-augmented models [27], [43] have become popular. However, these models often struggle with entity-centric questions for which sparse retrievers usually work well [1]. They also generalize poorly to new domains without supervision [29], [68].

In our research, we tackle the task of retrieval in OpenQA. We improve sparse retrievers by adding terms to the question. Adding terms can be from question terms or the terms that do not exist in the question. Our query expansion is not based on any prior retrieval and does not require any changes to the existing tools (e.g., the retriever and the index).

2.2 Query Expansion and Reformulation

One of the strategies that tackles vocabulary mismatch is query expansion (QE) [65] where the question is augmented with supplementary text to boost the matching likelihood. One strategy is to generate question paraphrases [22], which we also exploit in our model, but we ensure that frozen phrases are preserved. Similarly, other useful content, if available, may be added too.

An early work on query expansion, reported in 1960 [51], was within a mechanized library system. Relevance feedback, proposed in 1971, suggests using user feedback to improve the retrieval [65]. In 1990s, we started seeing large volumes of data being released on the web, and search engines were emerged to address society's information need. The size of document collections grew fast while the number of distinct queries did not grow at the same pace. That increased the ambiguity of query terms in finding answer documents and initiated some work on solving vocabulary mismatch via query expansion[2]. During that era, as one of the prominent early work, Buckeley *et al.* introduce pseudo-relevant feedback [6]. They attempt to retrieve the documents in two stages, treating the documents retrieved in the first stage as the user feedback. They use the feedback to expand the question and retrieve the final results based on the expanded query.

Since then QE methods have improved with a fast pace. As some of the recent work, GAR [50] expands questions with automatically generated sentences to enrich each question with clues—e.g., expected answers or sentences containing an answer—that are fetched from a pre-trained language model. Alternatively, Nogueira and Cho propose a reinforcement learning approach that uses post-retrieval signals as a reward function to reformulate the query [56].

Yu *et al.* train a model to encode pseudo relevance feedback signals to improve the query vector using a dense retrieval [83]. This can be viewed as a query expansion method for dense retrieval. There are some other recent papers investigating query expansion for dense retrieval [3], [71].

There are many pieces of work using language models in query expansion [3], [18]. Collins-Thompson and Callan apply a probabilistic model on a language model to generate expanding terms [16]. In addition, plenty of other works try to expand queries by probabilistic approaches [19], [49], [77]. In our work, we also study a new language model based query expansion method.

In our research, we aim to address the shortcomings of query expansion methods by placing more emphasis on the frozen phrases during the retrieval. Similarly, many scholars acknowledge risks and drawbacks of query expansion

and aim at reducing those risks [15], [28], [58].

2.3 Query Re-weighing

One part of the thesis can be viewed as a query re-weighing technique. We devise a way to detect frozen phrases as an important part of a question, and we increase their weight only in the retrieval stage, which consequently improves the end-to-end OpenQA performance. We generate training data to detect frozen phrases in a supervised fashion. In short, our frozen phrase detection can be considered as a supervised query re-weighing method for sparse retrievers, and we evaluate our method on the OpenQA task.

Zheng and Callan construct DeepTR [86], a tool to re-weight question terms. They assume that the proper weight of each term is the recall that they define for each term. The recall of a term is defined as the ratio of the number of related documents that contain the term to the total number of related documents. They estimate term recalls using a deep neural model directly. Zheng and Callan design their term re-weighing method based on language models, and they use sparse retrievers to retrieve the answer passage for the re-weighted query[86]. However, they change the scoring function of those retrievers to add their term weights.

Dai and Callan introduce DeepCT, which is a term weighing framework [20]. They leverage the BERT language model [21] to capture some contextual features. The features are then converted to term weights with a mapping function. Their main focus is on first-stage passage retrieval, and the suggested term weights should be used as term frequency in the inverted index.

SparTerm learns an sparse representation for vocabulary terms, and converts bag-of-word representations to a space containing more information about terms[4]. The authors argue that their learned sparse representation provides both term-weighting and query expansion. SPLADE is built on top of SparTerm that adds logarithmic activation and sparse regularization to the model [25]. Using some additional hyper-parameter tuning, the authors of SPLADE manage to outperform SparTerm [25].

In contrast to all the mentioned work in this section, there is no need to make any changes to the sparse retriever in our introduced re-weighing method. Only the questions change, and the index and the retrieval scoring function stays untouched. In addition, our method does not predict the term weights directly. We predict the important parts of the question that are expected to be seen in the answer document.

If questions are expanded, term weights are typically predicted again on the newly expanded questions. However, the model needs to be trained on the new distribution of expanded questions which can be time inefficient. On the other hand, our frozen phrase detection method is only applied on the original question, and the detected phrases are remembered to be added again to the expanded question.

2.4 Question Decomposition

Detecting frozen phrases can be seen as decomposing the question into different phrases. Each phrase can potentially describe different entities or form a piece of a longer question. Therefore our method can be used to extract important entities or phrases in a complex question for decomposition.

Li *et al.* use two neural models to extract different entities from questions and map them to their existing referencesin Wikipedia [44]. In contrast, our frozen phrases are not limited to only entities. For instance, we identify a famous quote as a frozen phrase because we expect it to appear as a whole in the answer passage. Additionally, unlike entity linking, there may not be a reference entity outside the questions for each frozen phrase.

For complex questions such as multi-hop questions [81], decomposing them into simpler sub-questions is a known technique [53], [59], [74]. Our method is analogous to these methods in that we also detect sub-sequences in questions. However, our objective is inherently different as we find sequences that are expected to match answer documents verbatim.

Qi *et al.* propose an iterative strategy to find potentially relevant documents at each step of the retrieval using a semantic overlap method [60].

Specifically, the overlap is computed via a longest common sub-sequence/sub-string algorithm between a target document and the current context. We also employ a similar approach to align questions with their corresponding answer documents.

Chapter 3

Frozen Phrase Detection

We introduced frozen phrases in Section 1.5.2 using examples. In this chapter, we study the problem of detecting them in natural language questions. However, before discussing our algorithms, we first provide a more formal definition.

Definition 1 *Frozen phrases are spans of text in questions that are expected to transfer as a whole to answer passages.*

Examples of frozen phrases are "I can only imagine" and "one man's vulgarity is another's lyric" in the questions "Who wrote the country song I Can Only Imagine" and "Who said one man's vulgarity is another's lyric" respectively. Using the above definition, we discuss our frozen phrase detection method in Section 3.1 before presenting our experiments in Section 3.2.

3.1 Frozen Phrase Detection

The problem of detecting frozen phrases in questions can be cast as a sequence labelling problem. Given a word sequence $w_1, \ldots, w_n$, denoting a question, we seek to find sub-sequences $w_i, \ldots, w_j$ of consecutive terms such that $w_i, \ldots, w_j$ is expected to transfer as a whole to the answer document.

For example, given the question "*Who sang I ran all the way home*," from NQ-open [38], we want to identify the song title "*I ran all the way home*" as a frozen phrase. Clearly, detecting phrases with a low selectivity [1] is more

[1]The selectivity of a term in a corpus is the fraction of documents or passages in the

desirable since these phrases are less likely to appear in arbitrary non-answer documents and they can be more effective in retrievals.

A major challenge in training a model to detect such phrases is the lack of annotated data for this purpose. It may seem at first that frozen phrases can be annotated in a QA corpus by leveraging the Longest Common Sub-sequence (LCS) between a question and its answer document. However, given that answer documents are long, every question term is likely to appear somewhere in the answer passage and will be included in an LCS. Such sequences may not really form a phrase and are not the subject of our study.

The Longest Common Sub-string (LCStr) may be considered as an alternative for annotating frozen phrases. However, our experiments show that LCStr misses many phrases that do not *exactly* transfer to the answer due to minor differences such as misspelling (see Section 3.2.5 for our evaluation of LCStr).

Inspired by the Smith-Waterman (SW) local alignment algorithm [67], we align the words sequence of a question with its answer document and extract frozen phrases from the question.

Let Q and X respectively denote the word sequences of a question and its answer document. Given the word sequences of a question Q and its answer X, let $H_{i,j}$ be the maximum alignment score between their prefixes $Q_1, ...Q_j$ and $X_1, ..., X_i$. We define the scoring function as:

$$H_{0,0} = 0 \ , \ H_{i,0} = -i.W_X \ , \ H_{0,j} = -j.W_Q,$$
$$H_{i,j} = \max \begin{cases} H_{i-1,j-1} + S_{i.j}(X, Q) \\ H_{i-1,j} - W_X \\ H_{i,j-1} - W_Q, \end{cases}$$

where W_X and W_Q are the gap penalties in the document text and the question respectively, and $S_{i.j}(X, Q)$ is the matching score of X_i and Q_j, which is defined as follows:

$$S_{i.j}(X, Q) = \begin{cases} IDF_{uni}(X_i) & \text{if } X_i{=}Q_i \wedge X_{i-1}{\neq}Q_{i-1}, \\ IDF_{uni}(X_i) + IDF_{bi}(X_{i-1}.X_i) & \text{if } X_i{=}Q_i \wedge X_{i-1}{=}Q_{i-1}, \\ -\infty & \text{if } X_i{\neq}Q_i, \end{cases} \tag{3.1}$$

corpus that contain the term. Phrases that are uncommon and only appear in very few documents are considered to have a low selectivity.

where $IDF_{uni}(.)$ and $IDF_{bi}(.)$ respectively denote the IDFs of a unigram and a bigram. Using the IDF of a term helps us to assign higher scores to the terms with low selectivity, and using bigram IDF boosts the score of longer phrases. $S_{i.j}(X, Q)$ is set to $-\infty$ when X_i and Q_i are not equal to force the alignment to consider gaps.

The choice of a gap penalty is important on how the phrases are formed. A phrase that is perfectly transferred to an answer will not have gaps but often the wording of a question has extra terms, for example, due to misspellings, or misses out terms that appear in the answer. We are not expecting many extra terms in question phrases, hence we set W_Q to a constant.

However, the gap structure in the answer documents is a bit more complex. For example, a question that refers to a song title can miss out a few words. Similar observations are made in detecting molecular sub-sequences where the gap penalty is divided into an opening gap penalty P_{open} and a continuing gap penalty P_{cont}, with $P_{cont} < P_{open}$. For example, Smith, Waterman, *et al.* set P_{open} to 1.33 and P_{cont} to 0.33 [67].

Generally, as the length of a frozen phrase increases, the chance that a user misses out words or writes them inaccurately in the question also increases. That inaccuracy breaks the chain of the matching words in the frozen phrase. For example, consider the phrase "*You Can't Always Get What You Want*," the name of a song by The Rolling Stones rock band, that is mentioned in a document, but a question refers to it as "*you can't get what you want*." There is a mismatch in the middle of the phrase, with *always* dropped in the question.

Figure 3.1 illustrates two possible alignments of the phrases mentioned above. A1: (``You Can't'', ``you can't''), (``Always'',-), (``Get What You Want'',``get what you want''), and A2: (``You Can't Always'',-), (``Get What You Want'',``get what you want''). The terms "*you*" and "*can't*" and the phrase "*you can't*" are expected to have low IDF since they are common terms, and it is likely that the matching score of "*you can't*", Eq. (3.1), minus the opening gap penalty (applied after) to be less than the gap penalty $-3.P_{cont}$. That means A2 will be selected over A1, and the first

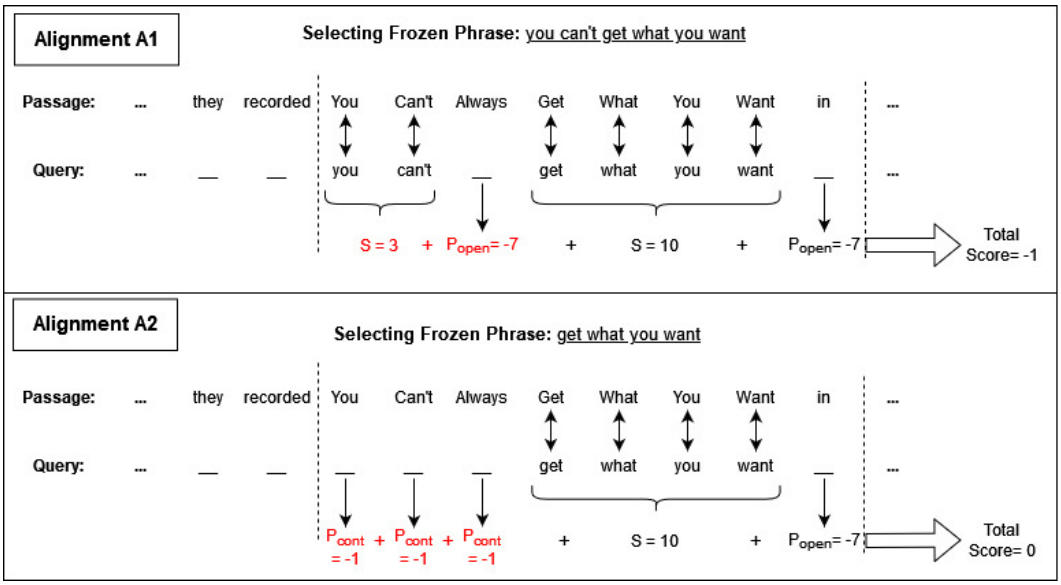

Figure 3.1: An illustration of a problem with the alignment algorithm. Some part of the frozen phrase are dropped when the opening gap penalty is not distributed over multiple gaps (Alignment A2 will be chosen over A1). S denotes the matching score of a phrase, defined in Eq. (3.1). P_{cont} and P_{open} are the continuing gap penalty and the opening gap penalty, respectively. Here, we set P_{cont} to -1 and P_{open} to -7. The matching scores of "*you can't*" and "*get what you want*" are 3 and 10, respectively.

part of the phrase will be ignored.

To avoid such cases, we need to distribute the opening penalty in the first few gaps instead of applying it all at once. We need the opening gap to increase proportional to the likelihood of having longer gaps inside a phrase.

Consider a Question $Q = \{Q_1, \ldots, Q_l\}$ of length l and let X denote the term sequence of the answer document. Let $Q_{i+1}, \ldots, Q_{i+k}$ be a sub-sequence of Q of length k such that the preceding term Q_i and the following term Q_{i+k+1} are matched with terms in X but the terms in $Q_{i+1}, \ldots, Q_{i+k}$ are not matched. Suppose π_k denotes the probability of not having Q_i and Q_{i+k+1} in the same frozen phrase; we want our opening gap penalty to increase proportional to π_k. If we denote with t the smallest positive integer where $\pi_t \approx 1$, then the general formula for W_X can be expressed as:

$$W_X(k) = \begin{cases} P_{cont} & (k > t) \\ \frac{\pi_k . P_{open}}{\sum_{i=1}^{t} \pi_i} & (0 < k \leq t). \end{cases}$$

As for setting the values of P_{cont} and P_{open}, one can leverage the following

formula:

$$\sum_{k=0}^{n} S(X_{i+k}, Q_{j+k}) - P_{open} > -(n+t) \times P_{cont}, \tag{3.2}$$

where $Q_j, Q_{j+1}, ..., Q_{j+n}$ and $X_i, X_{i+1}, ..., X_{i+n}$ are respectively question and document phrases that are matched. For good phrases, we want the above inequality to be satisfied and those phrases to be selected, and for bad phrases, we want the inequality not to be satisfied and the phrases to be ignored.

The annotation task, implemented using a dynamic programming algorithm, maps each question to a sequence of labels "`SEQ`" and "`O`", with "`SEQ`" indicating the terms that are part of a frozen phrase and "`O`" indicating the terms that are not. Using this procedure, we can create automatically annotated silver data on top of existing QA datasets.

Finally, we train a sequence-labelling model on the silver data. The model is employed in predicting frozen phrases for arbitrary questions and retrieving answer documents from a corpus. Table 3.1 provides examples of the terms extracted by the trained model and the alignment algorithm.

To summarize this section, we develop an alignment algorithm that aligns a question to its answer document to extract an invariant part of the question. The alignment algorithm labels each question term and identifies whether a question term is in a frozen phrase or not. Figure 3.2 demonstrates an example of such labeling.

After labeling all the question terms in the training data, A transformer based model [69] is trained to predict frozen phrases of a question. The trained model is then applied to the development set, and its predicted frozen phrases are added to the original question. This is done to give those phrases a higher weight in the sparse retriever query vector and emphasise more on them. Figure 3.3 shows the general framework of frozen phrase detection.

3.2 Frozen Phrase Experiments

In our evaluation for this section, we seek to answer the following questions:

Table 3.1: Examples of the frozen phrases derived by our alignment algorithm (Silver Standard) vis-a-vis the extracted phrases by the model (Prediction), trained on the silver data.

Silver Standard (our proposed alignment)	**Prediction**
hazels boyfriend in the fault in our stars	hazels boyfriend in the fault in our stars
when does the day of the dead end	when does the day of the dead end
where is the citrus bowl held this year	where is the citrus bowl held this year
what year does the quiet man take place	what year does the quiet man take place
how many seasons of rules of engagement is there	how many seasons of rules of engagement is there
who plays dusty in the movie pure country	who plays dusty in the movie pure country
how tall is the actor who plays hagrid in harry potter	how tall is the actor who plays hagrid in harry potter

1. how informative the frozen phrases extracted by our alignment algorithm are in terms of their recall in open-domain QA and how much information is lost by only keeping such phrases,

2. how effective the predicted frozen phrases are in improving the performance of sparse/dense retrievers and what role they play in query expansion, and

3. if the end-to-end performance of open-domain QA is improved by leveraging the predicted frozen phrases during retrieval.

3.2.1 Dataset

To generate our training data, we ran our alignment algorithm, discussed in Section 3, on the training set of the Natural Questions (NQ) dataset [38]. Our testing was done on the development set of NQ-open dataset [41] that consists

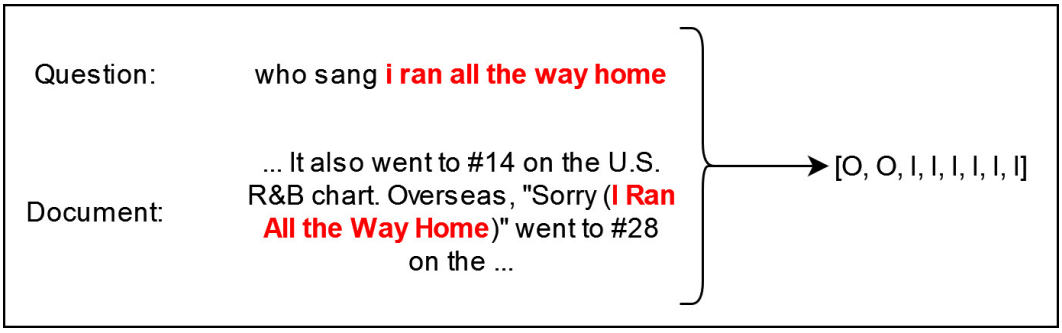

Figure 3.2: An example of the inputs and outputs of the alignment algorithm

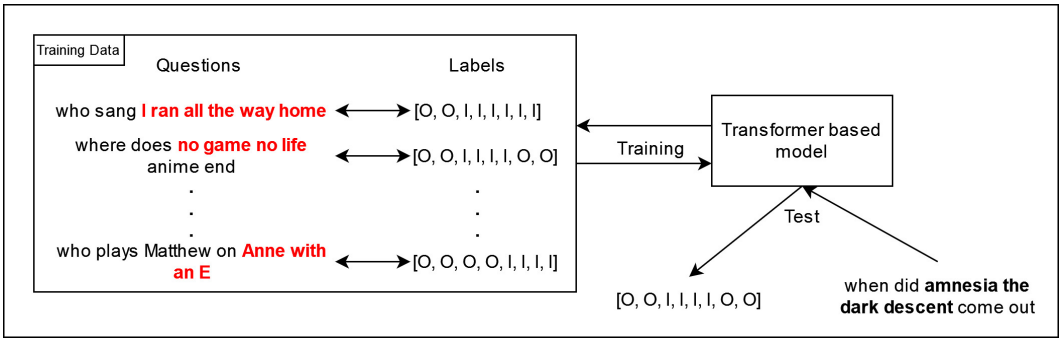

Figure 3.3: General framework of frozen phrase detection

of 3,610 questions. We excluded the questions that NQ-open did not have their answer document annotated in NQ dataset, reducing the test set to 3072 questions.

3.2.2 Alignment Hyperparameters

We randomly selected a small subset of our training data, on which we manually observed the alignment algorithm output. Since π_t is the probability at which two adjacent phrases do not form a frozen phrase, we can estimate it based on our statistical observation of the subset. We start with a sequence length 1 and increase the length to t where $\pi_t \approx 1$ ($\pi_k < \pi_{k+1}$ for every k). For setting P_{open} and P_{cont}, we leverage some negative and positive samples from the selected subset and find a setting where Eq. (3.2) is likely to hold for positive examples and it is less likely to hold for negative examples.

We want to assign a small value for W_Q (0 or close to 0) because we want the gap penalty in the question to be small to encourage the algorithm to ignore

the terms that are not in a frozen phrase. A high value for W_Q forces the algorithm to select as many terms as it can from the question, thus reducing our algorithm to LCS. Throughout our experiments, we used the following hyperparameters for the alignment algorithm:

$$P_{cont} = 1,\ P_{open} = 7,\ W_Q = 0.1,\ t = 3,\ \pi_1 = 0.25,\ \pi_2 = 0.5,\ \pi_3 = 1$$

3.2.3 Sequence-Labelling Model

To predict frozen phrases, we fine-tuned a pre-trained $\text{RoBERTa}_{\text{base}}$ model [47] with a token classification head[2] on our silver training data. The output of the model is a sequence of frozen phrases with possible gaps. Those gaps are replaced with an out-of-vocabulary term to avoid forming bigrams that are not in the original question. We refer to these generated questions as Frozen Phrase Questions (FPQ). We trained our model with a learning rate of $1.0e^{-4}$, a batch size of 64, and early stopping for 70 epochs.

3.2.4 OpenQA Pipeline

For sparse retrieval, we adopted the BM25 implementation from Pyserini [45]. Retrieval is done over passages of 100 words that are derived from Wikipedia, following [35]. As reader, the base model of Fusion-in-Decoder [31] was used. The top 100 retrieved passages are fed into the reader, as done in [31]. For our query expansion, a T5 transformer [63], fine-tuned on the Quora question paraphrase dataset, was used to generate up to 10 paraphrases for each question[3]. The paraphrases are concatenated with the original question with an out-of-vocabulary term added between the concatenated questions to avoid undesirable bigrams. We refer to the new expanded queries as *Orig10Par*.

3.2.5 Evaluation

As a measure of the informativeness of the frozen phrases, we introduce a metric based on the overlap between the terms in extracted frozen phrases

[2] `https://github.com/ThilinaRajapakse/simpletransformers`

[3] `https://github.com/ramsrigouthamg/Paraphrase-any-question-with-T5-Text-To-Text-Transfer-Tran`

and the title of answer document. Retrievers often assign higher weights to a match in the title than a match in the document body [26]. Moreover, since the document title is already prepended to the passages in the corpus, retaining the title terms becomes important in questions. Hence, we measure title recall, defined as the fraction of title terms that are preserved in a question (either FPQ or original question), i.e.,

$$Recall_{title} = \frac{\sum_{m=1}^{M} |Q^m \cap T^m|}{\sum_{m=1}^{M} |T^m|},$$

where M, Q^m, and T^m are the number of samples in the dataset, the question of the m-th sample, and the title of the m-th sample respectively. We consider an FPQ informative if its $Recall_{title}$ is close to that of the original question, meaning any loss is minimal.

In addition to $Recall_{title}$, we use top-k retrieval accuracy and Exact match (EM) to evaluate our retriever and reader, respectively. The two metrics are widely used in prior work [35], [50].

Information Loss

To evaluate the informativeness of frozen phrases, we converted the set of questions in our test set to FPQs using our proposed alignment algorithm. The length of an FPQ is only 54% of the length of original questions on average. As a baseline for comparison, we also generated another dataset by replacing each question with its corresponding LCStr. The information loss is measured for both FPQ and LCStr using $Recall_{title}$.

As shown in Table 3.2, $Recall_{title}$ of our alignment algorithm (FPQ) is very close to that of the original questions (Orig). Even though the algorithm drops 46.3% of the question terms, it only drops 1% of the title terms. Moreover, the average IDF of the 1% dropped terms is quite low — i.e., 2.76 for unigrams and 6.2 for bigrams in our dataset.

As another measure of a possible information loss in retrieval, we also evaluated the performance of passage retrieval over the three query sets Orig, FPQ, and LCStr. As presented in Table 3.2, the retrieval accuracy for FPQ is higher than LCStr by a large margin, which shows how the algorithm is

successfully selecting important terms. Furthermore, while we used almost half of the question terms, the performance declines only by around 2%. This confirms that the extracted frozen phrases play a significant role in retrieval.

Table 3.2: $Recall_{title}$ and Passage level Accuracy to evaluate the Information loss

Dataset	$Recall_{title}$	Acc@1	Acc@10	Acc@100
Orig	0.48	**24.38**	**57.97**	**80.73**
LCStr	0.38	11.91	33.76	59.05
FPQ	0.47	24.12	56.35	78.91

Retriever Performance

To evaluate the performance of our frozen phrase prediction, we trained our transformer model on the data annotated by our alignment algorithm. The trained model was applied to the questions in our test set to generate FPQs.

To evaluate if the use of frozen phrases can improve the retrievals, we concatenated each FPQ to its original question in our test set to increase the weight of the predicted frozen phrases in the question vector. As shown in Table 3.3, adding frozen phrases to the questions (this is referred to as Orig+FPQ) significantly improves the accuracy. As a baseline for comparison, we also did a similar experiment but, instead of adding frozen phrases, we added named entities (Orig+NER) and singular nouns (Orig+NN) to increase their weight. Unlike frozen phrases, adding named entities (Orig+NER) and singular nouns does not improve the retrieval.

To evaluate the performance of our model prediction in query expansion, we generated up to 10 paraphrases for each question in our test set, using the method described in Section 3.2, and added those paraphrases to the original question (Orig10Par). Then, we added 10 copies of FPQs to the expanded query set and created a new set of questions (Orig10Par+10FPQ). We further added 10 copies of the original test questions to the expanded query (Orig10Par+10Orig) to see if our predictions are more helpful than the original questions for the expanded query. The results of the BM25 retriever

on the aforementioned question sets as well as the original questions (Orig) are reported in Table 3.3.

The best result is achieved when 10 paraphrases and 10 copies of the FPQ are added to the original questions (Orig10Par+10FPQ), and the questions that only have the FPQ in addition to the original question (Orig+FPQ) stand second. These results show that we are increasing the weight of the right terms, and our frozen phrase extraction method improves the retrieval.

We also combine our enhanced BM25 retrievers with DPR [35], a prominent dense retriever, by taking a weighted mean of their retrieval scores, following [35]. The results are consistent with the previous results where we used only sparse retrieval.

End-to-End Performance

Finally, Table 3.4 shows the end-to-end exact-match accuracy of our models where the reader is applied to the two question sets that have the best retrieval performance as well as to the original set of test questions. We can observe that the performance boost in the retrieval translates to a better performance of the reader, and that the end-to-end improvement is statistically significant.

Table 3.3: Retrieval accuracy at top-k on different question sets. † denotes statistical significance (p-value < 0.01) over Orig for each retriever, BM25 and DPR+BM25.

Question Set	Acc@1	Acc@10	Acc@100
BM25 on Orig	24.38	57.97	80.73
BM25 on Orig+NER	24.61	58.69	80.37
BM25 on Orig+NN	23.568	58.07	80.18
BM25 on Orig+FPQ	25.65†	60.06†	82.13†
BM25 on Orig10Par	25.29	57.45	80.99
BM25 on Orig10Par+10Orig	25.16	58.89	81.61
BM25 on Orig10Par+10FPQ	**26.43**†	**60.25**†	**82.52**†
DPR on Orig	46.87	77.25	88.54
DPR+BM25 on Orig	48.80	78.81	89.20
DPR+BM25 on Orig+FPQ	**50.01**†	**79.07**	89.40
DPR+BM25 on Orig10Par+10FPQ	49.64	78.66	**90.10**†

Table 3.4: End-to-end exact-match accuracy based on the best question sets for two retrievers. † denotes statistical significance (p-value < 0.01) over Orig.

Question Set	EM for BM25	EM for DPR+BM25
Orig	42.84	48.57
Orig+FPQ	<u>43.72</u>†	<u>49.15</u>
Orig10Par+10FPQ	**44.40**†	**49.97**†

Chapter 4

Query Expansion

In this chapter, we present our proposed query expansion method in Section 4.1 and our evaluation of its performance in Section 4.2.

4.1 Methodology

The task of query expansion is to add new terms to questions with the goal of including more terms from the answer passages of the expanded queries. Annotated data is not always at our disposal, especially in specialized domains. It is also well established that supervised query expansion methods are not time-wise efficient in contrast to unsupervised methods [85]. Therefore, we opt for methods that do not require labeled data. Although few-shots learning methods are considered supervised, they do not demand excessive training data, and deploying them becomes appealing for our task. Also, few-shot learning methods are not as expensive as regular supervised methods especially if one deploys a pre-trained model without fine tuning.

Multitask learning is a class of learning algorithms that improves generalization by transferring information from one related task to another [9]. Radford *et al.* demonstrate that language models can be exploited for multitask learning [62]. Since language models are trained to predict a probability distribution $P(x)$ where x is a sample from the sequence $s_1, s_2, ..., s_n$, for every $0 < k < n$, one can sample $P(s_{n-k+1}, ..., s_n | s_1, s_2, ..., s_{n-k})$ using the language model. Radford *et al.* exhibit the conditional probability as $P(output|input, task)$ [62]. Note that for few-shot learning the input can con-

tain some training samples as well. Since language models tend to have a better performance as the model grows in size, we select GPT-3, as one of the largest available language models [5].

To generate the words that can expand a query, we sample from the conditional probability $P(words|question, prompt)$ where *words* refer to the predicted words for query expansion, *question* indicates the question string, and *prompt* suggests the task that we are asking the language model to perform. Our goal is to design the prompts which lead to the generation of answer passage terms so that the retrieval of the question is improved. In the rest of this section we introduce various tasks and different prompts to predict terms from the answer passage. Table 4.1 demonstrates an example with the generated text for each introduced prompt.

Two relevant tasks are studied in the context of generative question answering, namely *answer passage title* generation and *answer sentence* generation. Mao *et al.* train a model to generate the answer document title from a corresponding question in the NQ dataset [50]. They use the model to predict for each question the title of the answer passge as a context of the question. They concatenate the question with the predicted title to perform a retrieval [50]. They also predict the final answer and the answer sentence given the question. However, they show that by only appending the predicted title to the question the performance at top 100 retrieved passages increases more than appending the predicted answer or the answer sentence. The above observation motivated us to investigate if one can use large language models such as GPT-3 to generate some potential titles per question for enhancing the retrieval. We use zero-shot learning, where the model is simply asked to generate some titles for the answer document of the given question. The generated titles are then deployed to expand the question.

As motioned earlier, Mao *et al.* train a model to generate answer sentences for a given question in a supervised fashion [50]. In doing so, many model learning parameters are wasted for learning how to generate a proper English sentence as the answer sentence. Additionally, since the generated answer sentence follows the normal distribution of English words in a sentence, there

would be many common words in the generated sentences which are not useful for retrieval.

Based on these observations, in one of our prompts, we attempt to acquire answer passage terms rather than sentences. We directly ask the model to generate some words that are expected to appear in the answer passage of a question. We also emphasize that the output words should not include stop words or question words. In another prompt, we leverage the same prompt setting as before, but we emphasize that the model should try to output rare words as much as possible. Without emphasizing on rare words, the language model usually does not return words with high IDFs.

Finally, in one more prompt, we make the language model to predict an answer for each question, then the predicted answers are utilized to expand the question. The reported exact match accuracy of GPT-3 on NQ dataset under zero-shot setting is 14.6, which is quite low [5]. We hypothesize that there are some useful words in the generated answer while the the answer itself is not always accurate. For example, the model has seen documents about released movies and their characters during pre-training. Thus, when the name of the movie is mentioned in the question, the model can generate related information to the movie as the finally answer by mistake. One can use that related information for query expansion with the hope that it includes some answer passage terms.

4.2 Experiments

We seek to answer the following questions in our experiments:

1. how informative the terms generated by large language models are in terms of their recall in open-domain QA and how much information is lost by only retrieving such terms,

2. how effective the generated terms are in improving the performance of sparse/dense retrievers and what role they play in query expansion, and

3. if the end-to-end performance of open-domain QA is improved by leveraging the generated terms during retrieval.

Experimental Setup. We deployed the Pyserini [45] implementation of BM25 with the default parameters as our sparse retriever. The text completion model of GPT3 is used as the large language model (LLM) for generating the terms for query expansion. The model "text-davinci-003" with a temperature of 0.95 is selected, and other parameters are set to default in our experiment.

Dataset. Our dataset is the Natural Questions-open dataset [42]. The test set has 3610 questions written in natural language. Our corpus is the Wikipedia corpus with about 20 millions passages such that each passage contains 100 words [35]. Additionally, we use a reduced dataset in Table 4.3 that only contains NQ questions that their designated answer passage contain the final answer [76].

4.2.1 Baseline models

We compare our method to various prominent retrieval models and query expansion methods including supervised, unsupervised, dense, and sparse retrieval models. For the unsupervised setting, we use Contriever the state-of-the-art unsupervised retriever as our baseline [30]. We also provide the results for the unsupervised baselines used in the Contriver paper (Inverse Cloze Task and Masked Salient Spans). Contriever uses contrastive learning with different techniques to provide training data automatically to train their dense retriever. They even go further and pre-train their model on the MS-MARCO dataset to perform zero-shot on NQ dataset (Contriever MSMARCO). Although our method does not include any pre-training, we use Contriever MSMARCO as another baseline. It is worth mentioning that our method leverages a sparse retriever while Contriever is an unsupervised dense retriever.

AR2-G [84], RetroMAE [48], and ART [66] are three recent state-of-the-art supervised retrievers. We use these baselines to show the accuracy difference with the best supervised models. DPR [35] is a classic dense retriever, GAR

[50] is a prominent supervised QE method that aims at closing the gap between sparse retrievers (ex. BM25) and dense retrievers (ex. DPR).

4.2.2 Study of QE Model

Information Loss Per Prompt

In this section, we evaluate the informativeness of the terms generated by the large language model within our introduced prompts. First, we investigate how much information is lost when only retrieving the output of each prompt. Table 4.2 shows the retrieval accuracy when language model generated text is used for the retrieval, instead of the original question. Using the generated potential title and answer, instead of the original question, can improve the retrieval result. However, only slight performance gain (less than one percent) is achieved with the generated answer at top 100 accuracy. On the other hand, the accuracy at top 5 is boosted by about 14 percent and that at top 20 is boosted by more than 5 percent. This is due to the fact that if any of the generated words are in the answer passage, the answer passage ranking will improve, whereas if any of the generated words are not in the answer passage, the ranking will worsen. We observe that the changes in the answer passage rankings are radical for each generated term set, and this does not let the accuracy at top 100 to improve much. However, this accuracy is important for the task of question answering.

This problem with query expansion can be avoided by reducing the weight of each generated word in the expanded query. In this case, generating one "*bad*" word will not impact our retrieval performance noticeably. We want to predict the probability $P(w|q)$ from the output of a large language model where w is a generated word for expansion and q is the question. However, if we get only one sample from the language model, all the generated words will be assumed to have the same probability. In that case, when an out of passage term is generated, it will have a huge negative impact on the retrieval because its probability is assumed to be too high. We address this issue with our query normalization as discussed next.

Normalizing the Query by Multiple Sampling

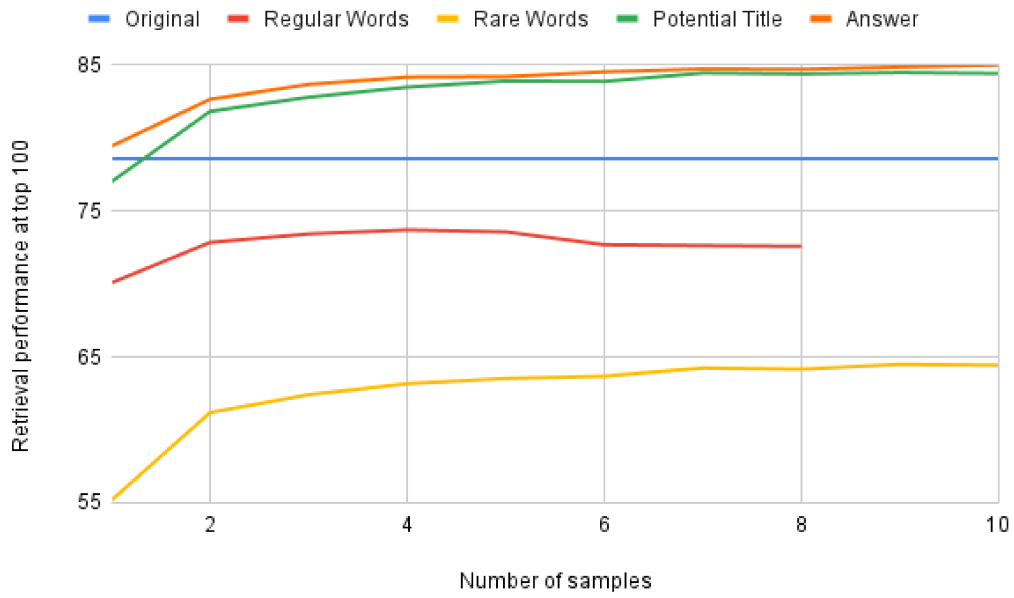

Figure 4.1: Retrieval performance, in terms of to 100 accuracy, when multiple samples are acquired from the language model

Figure 4.1 demonstrates the effect of multiple sampling (from the language model) on the retrieval performance. The performance in terms of retrieval accuracy improves for all the prompts when multiple sampling is applied. Theoretically, all performance numbers should converge to the performance of the predicted $P(w|q)$. As we increase the number of samples, the number of occurrences of w in all the samples reflects the predicted $P(w|q)$ more accurately. While the expanded query gets lengthier, the impact of adding a new sample will decrease as the ratio of the new sample length to the query length in the previous step shrinks. Therefore, sampling too many times will not be effective either, and it only adds costs to our retrieval.

In Figure 4.1, expanding the query with a generated answer and a potential title show similar performances, whereas they both surpass retrieving original questions by a large margin at top 100 accuracy. In contrast, expanding the query with the generated regular words and rare words cannot make any improvements to the original questions even with multiple sampling. Multiple

sampling is more effective for rare words since they tend to have words with higher IDFs. If a word with a high IDF is generated in the first sample by mistake, it will worsen the performance more than generating a low IDF word incorrectly. Thus, we recommend more sampling if the generated terms have high IDF values.

Normalizing the Query by Question Repetition

Due to the high cost of large few-shot in-context learners such as GPT3 [5], we introduce the task of prompt reduction which aims to reduce the total cost of prompts produced. We tackle the task by introducing distribution mixture effect, which can be applied to prompt outputs such that we benefit from the diversity of the generated words. However, this cannot be done without minimizing the noisy effect of diverse words, which can deflect our query vector radically if it is not kept under control.

As we mentioned earlier and saw in Figure 4.1, it is important to reduce the weight of the first sample to reduce its noise. In Figure 4.1, it is done by multiple sampling. We can do the same thing by leveraging question terms as weight regulators. Adding more question terms reduces the weight of expanding terms. Adding question terms incurs no cost, hence using them instead of multi-sampling can reduce the cost of the query expansion.

Figure 4.2 illustrates the effect of weight normalization over question terms. In contrast to the multi-sampling method, the accuracy at top 100 surpasses that of the original questions for all our prompt types. When we sample many times from the model, the query will approximately have the same distribution that the model is estimated. Therefore, it only converges to the accuracy of that distribution. On the other hand, adding a different distribution can but will not necessarily increase the retrieval accuracy of a prompt type above its convergence limit. It is worth mentioning that all the accuracies reported in Figure 4.2 converge to the accuracy of the original questions, i.e. the blue line with an accuracy of 78.5, as the weights of the original question terms increase up to infinity.

Now the question may rises about the number of original questions that

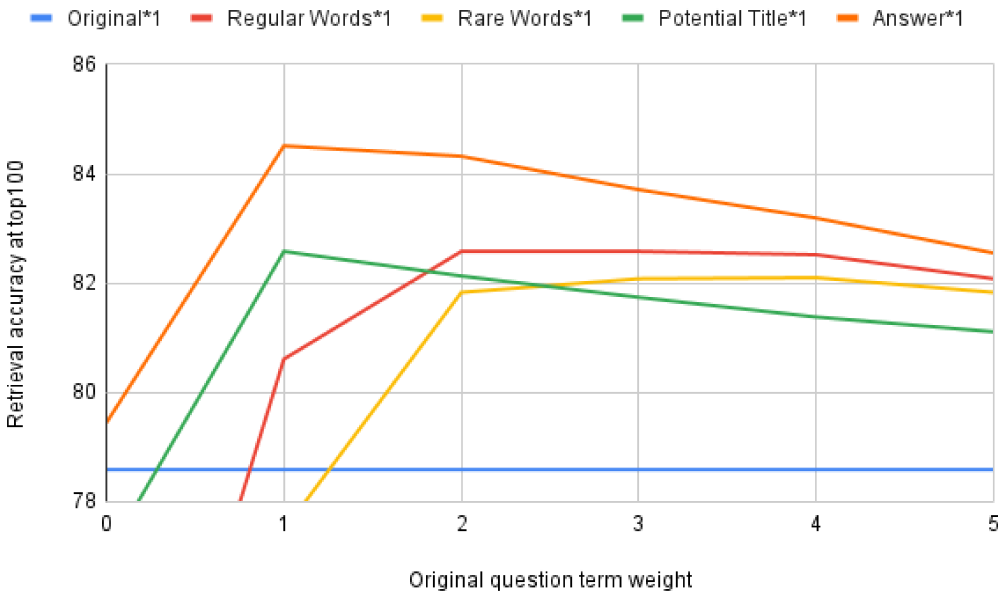

Figure 4.2: The effect of original question terms weight on retrieval accuracy of introduced prompts. From each prompt only one sample is taken and used

should be ideally added to a set of prompt output. We recommend conducting tests similar to the one in Figure 4.2 on a sample of questions from the training data. We leave the problem of estimating an ideal weighting of original question terms as a future direction.

Normalizing the Query by Frozen Phrases

As another approach to tackle the normalization of the terms that expand a query, we investigate the effect of adding frozen phrases [76] to the question. In Chapter 3, we introduced Frozen phrases as important question phrases that are expected to be in the answer passages of questions. We leverage our fine-tuned model for that task to predict frozen phrases of questions. Then we normalize the expanded question by adding the detected frozen phrases. In another experiment, we leverage our large language model and implement the frozen phrase detection by providing the model a few shot samples. In particular, we use five samples derived from the training data and annotated manually.

Table 4.3 demonstrates the results for the two normalization methods.

"FPQ" refers to the frozen phrases of questions, detected by our fine-tuned model from Chapter 3, which are added to the expanded questions. The column "GPT3" refers to the model where important phrases are detected by GPT3 and the expanded questions are normalized based on those terms. "No normalization" indicates that the question set only contains the original question and the expanding terms. It is worth mentioning that the reduced NQ dataset used in Chapter 3 is utilized for this experiment. In each row of Table 4.3, the highest accuracy is marked by an underscore.

Normalization results in a better performance boost when it is applied to the original questions (No QE) and the expanded questions with regular and rare words. However, normalization does not improve the performance much when query expansion is done by generating the answers and potential titles. This phenomenon is due to the fact that the generated answer and the potential title do not need extra normalization. They reach their maximum performance when only one copy of the original question is added to them (Figure 4.2). Moreover, although the frozen phrases predicted by GPT3 are more aligned with human annotation, FPQ surpasses GPT3 in the normalization task. It suggests that the alignment algorithm used to generate the training data for frozen phrase detection is beyond just detecting titles and quotes as it was our main purpose. The alignment algorithm trains the model to somehow predict the question terms that exist in the answer passages.

4.2.3 Retrieval Performance with QE

We compare the retrieval performance of our query expansion to other state-of-the-art methods in Table 4.4. As our QE method, we expand questions in NQ dataset by taking 7 samples from each of the four prompt types introduced earlier. We further normalize the expanded queries by adding 6 copies of the original questions to the expanded queries. Therefore, a total of 7 copies of the original questions are included in the resulting expanded queries. Our method surpasses the fully unsupervised Contriever [30] in retrieval accuracy, and we get comparable result to the pre-trained Contriever, which is known as the state-of-the-art unsupervised QE method (Contriever-

MSMARCO) [30]. Our QE method has a better retrieval accuracy (1.3%) than Contriever-MSMARCO at top 5 retrieved passages. However, as we increase the number of top retrieved passages that we consider for our evaluation, Contriever-MSMARCO performs better than ours. The mentioned phenomenon makes it unclear to decide which one of the approaches are the state-of-the-art unsupervised approach for the end-to-end question answering task. Thus, we report the end-to-end results in Section 4.2.4. Additionally, our QE method makes an sparse retriever (the BM25) surpasses DPR dense retriever [35]. Not only do we close the gap between sparse and dense retrieval, but also we close the gap between supervised and unsupervised methods. The retrieval accuracy of our method is higher than that of GAR [50] which is one of the prominent supervised QE methods.

4.2.4 End-to-End Effectiveness

Table 4.5 compares the end-to-end performance of our proposed query expansion method to that of other state-of-the-art query expansion methods. We leverage Fusion-in-Decoder (FiD) reader [31] with default parameters, and we apply it on top 100 passages. The input question to the reader is the original dataset questions and our expanded questions are only utilized in the retrieval stage.

The exact match (EM) accuracy of FiD on our retrieved passages is higher than the state-of-the-art unsupervised query expansion method Contreiver-MSMARCO [29] by almost one percent which clearly addresses the dilemma in section 4.2.3. However, still most recent prominent supervised methods surpass our method in terms of EM.

Table 4.1: Examples of different types of prompts used for query expansion. The generated text is highlighted.

Task	**Completed Prompt Example**
Regular Word Generation	Suggest 10 words that are expected to appear around the answer of the given question. Please do not suggest stop words and words that are in the question. Question: when was the last time anyone was on the moon Words: Astronauts, Apollo, Lunar, Module, Module-Lander, Orbit
Rare Word Generation	What words are expected to appear around the answer of the given question? Try to suggest rare words as much as possible. Question: when was the last time anyone was on the moon Words: Apollo, astronauts, mission, lunar, roving, 1971, terrain
Title Generation	Write a potential title for the passage including the answer of the given question. The title should include words that are not in the question.. Question: when was the last time anyone was on the moon Title: A Historical Milestone: Last Moon Landing in 1972
Answer Generation	Answer the following question. Question: when was the last time anyone was on the moon Answer: The last human to be on the moon was Eugene Cernan, Commander of

Table 4.2: BM25 performance on retrieving the output of different GPT3 prompts without concatenation of original questions.The full NQ dataset is used.

Question set	Acc@5	Acc@20	Acc@100
Orig	45.12	64.10	78.59
Regular Words	43.80	57.53	70.08
Rare Words	31.63	42.55	55.18
Potential Title	49.61	64.04	77.01
Answer	**59.06**	**69.70**	**79.45**

Table 4.3: Accuracy at top 100 of BM25 performance on retrieving the output of different GPT3 prompts with different normalization methods. The reduced NQ dataset is used.

QE method	No normalization	GPT3	FPQ
No QE	80.73	80.24	82.13
Regular Words	82.10	83.27	85.32
Rare Words	78.97	80.31	83.66
Potential Title	84.93	84.80	84.70
Answer	86.85	86.88	86.85

Table 4.4: State-of-the-art query expansion methods on NQ dataset. The full NQ dataset is used.

	QE Method	Acc@5	Acc@20	Acc@100
Unsupervised	No QE	45.12	64.10	78.59
	Inverse Cloze Task [30]	32.3	50.9	66.8
	Masked Salient Spans [30]	41.7	59.8	74.9
	Contriever† [30]	47.6	67.6	81.9
	Contriever_MSMARCO† [30]	65.2	79.1	87.4
	Ours	67.0	78.2	86.8
	Ours+Contriever	66.1	79.9	87.8
	Ours+Contriever_MSMARCO	**69.9**	82.0	**88.9**
Supervised	DPR_{single} [35]	-	78.4	85.4
	DPR_{multi} [35]	-	79.4	86.0
	GAR [50]	60.9	74.4	85.3
	ART [66]	-	80.2	88.4
	RetroMAE [48]	-	81.72	88.12
	$AR2-G^0$ [84]	69.7	80.8	87.1
	AR2-G [84]	**77.9**	**86.0**	**90.1**

Table 4.5: Accuracy of several open-domain QA models on NQ-open.

Model	**K**	**EM**	**F1**
InstructGPT (zero-shot) [5]	-	14.6	-
InstructGPT (few-shot) [5]	-	29.9	-
DPR [35]	50	40.9	47.8
FiD [31]	100	46.5	53.7
RocketQAv2 & FiD [64]	100	47.7	55.6
Contriever & FiD [33]	100	47.9	55.4
FiD-KD [33]	100	49.6	57.4
GAR+ & FiD [50]	100	49.8	57.4
EviGen [33]	20	49.8	57.0
EMDR^2 [33]	50	51.5	**59.5**
R2-D2 [33]	25	**52.4**	59.0
Ours & FiD	100	48.9	57.0
Ours+Contriever	100	49.1	57.1
Ours+Contriever_MSMARCO	100	49.7	57.7

Chapter 5

Conclusion

In the first part of the thesis, we examine the importance of contiguous term spans, namely frozen phrases, that are expected to appear verbatim in answer passages. Frozen phrases embody a type of locality that is crucial in finding potential answer passages in OpenQA. However, detecting them is challenging due to an absence of existing annotated data. We address this problem by introducing a strategy to construct synthetic silver data from existing QA datasets. We show that by incorporating frozen phrases, the retrieval accuracy as well as the end-to-end performance substantially improves. Frozen phrases can also be integrated into the backbone of dense retrieval models, which is an interesting direction for future work.

In the second part of the thesis, we explore the potential of multi-task learners in query expansion. It is shown that combining two different query term distributions (e.g., terms in questions and terms generated by LLM to expand the questions) makes more improvements than simply sampling more terms from one distribution. Using the mentioned technique, we reduce the number of prompts we use in our query expansion.

As a future research direction, the ideal ration that two expanding word distributions should be divided by can be studied and estimated for further improving the query expansion using large language models. The syntax and semantic of prompts can be explored to optimize them for getting better output (expanding terms) from the LLM. Moreover, Some ablation studies can be performed on the frozen phrase alignment algorithm to see the effects of each

parameter on the silver standard.

References

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

[2] H. K. Azad and A. Deepak, "Query expansion techniques for information retrieval: A survey," *Information Processing & Management*, vol. 56, no. 5, pp. 1698–1735, 2019.

[3] J. Bai, D. Song, P. Bruza, J.-Y. Nie, and G. Cao, "Query expansion using term relationships in language models for information retrieval," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005, pp. 688–695.

[4] Y. Bai, X. Li, G. Wang, *et al.*, "Sparterm: Learning term-based sparse representation for fast text retrieval," *arXiv preprint arXiv:2010.00768*, 2020.

[5] T. Brown, B. Mann, N. Ryder, *et al.*, "Language models are few-shot learners," *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[6] C. Buckeley, G. Salton, J. Allan, and A. Stinghal, "Automatic query expansion using smart," in *Proceedings of the 3rd Text Retrieval Conference*, 1994, pp. 69–80.

[7] D. Carmel, A. Mejer, Y. Pinter, and I. Szpektor, "Improving term weighting for community question answering search using syntactic analysis," in *Proceedings of the 23rd acm international conference on conference on information and knowledge management*, 2014, pp. 351–360.

[8] C. Carpineto and G. Romano, "A survey of automatic query expansion in information retrieval," *Acm Computing Surveys (CSUR)*, vol. 44, no. 1, pp. 1–50, 2012.

[9] R. Caruana, *Multitask learning*. Springer, 1998.

[10] D. Chen, A. Fisch, J. Weston, and A. Bordes, "Reading wikipedia to answer open-domain questions," *arXiv preprint arXiv:1704.00051*, 2017.

[11] C. Clark and M. Gardner, "Simple and effective multi-paragraph reading comprehension," *arXiv preprint arXiv:1710.10723*, 2017.

[12] C. L. Clarke, G. V. Cormack, and T. R. Lynam, "Exploiting redundancy in question answering," in *Proceedings of the 24th annual international ACM SIGIR conference on Research and development in information retrieval*, 2001, pp. 358–365.

[13] V. Claveau, "Query expansion with artificially generated texts," *arXiv preprint arXiv:2012.08787*, 2020.

[14] O. Cody Kwok, "Scaling question answering to the web," *ACM Transactions on Information Systems (TOIS)*, vol. 19, no. 3, pp. 242–262, 2001.

[15] K. Collins-Thompson, "Reducing the risk of query expansion via robust constrained optimization," in *Proceedings of the 18th ACM conference on Information and knowledge management*, 2009, pp. 837–846.

[16] K. Collins-Thompson and J. Callan, "Query expansion using random walk models," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005, pp. 704–711.

[17] R. Crimp and A. Trotman, "Automatic term reweighting for query expansion," in *Proceedings of the 22nd Australasian Document Computing Symposium*, 2017, pp. 1–4.

[18] S. Cronen-Townsend, Y. Zhou, and W. B. Croft, "A language modeling framework for selective query expansion," MASSACHUSETTS UNIV AMHERST CENTER FOR INTELLIGENT INFORMATION RETRIEVAL, Tech. Rep., 2004.

[19] H. Cui, J.-R. Wen, J.-Y. Nie, and W.-Y. Ma, "Probabilistic query expansion using query logs," in *Proceedings of the 11th international conference on World Wide Web*, 2002, pp. 325–332.

[20] Z. Dai and J. Callan, "Context-aware term weighting for first stage passage retrieval," in *Proceedings of the 43rd International ACM SIGIR conference on research and development in Information Retrieval*, 2020, pp. 1533–1536.

[21] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[22] L. Dong, J. Mallinson, S. Reddy, and M. Lapata, "Learning to paraphrase for question answering," *arXiv preprint arXiv:1708.06022*, 2017.

[23] P. Duboue and J. Chu-Carroll, "Answering the question you wish they had asked: The impact of paraphrasing for question answering," in *Proceedings of the Human Language Technology Conference of the NAACL, Companion Volume: Short Papers*, 2006, pp. 33–36.

[24] S. Dumais, M. Banko, E. Brill, J. Lin, and A. Ng, "Web question answering: Is more always better?" In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval*, 2002, pp. 291–298.

[25] T. Formal, B. Piwowarski, and S. Clinchant, "Splade: Sparse lexical and expansion model for first stage ranking," in *Proceedings of the 44th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2021, pp. 2288–2292.

[26] V. N. Gudivada, V. V. Raghavan, W. I. Grosky, and R. Kasanagottu, "Information retrieval on the world wide web," *IEEE Internet Computing*, vol. 1, no. 5, pp. 58–68, 1997.

[27] K. Guu, K. Lee, Z. Tung, P. Pasupat, and M. Chang, "Retrieval augmented language model pre-training," in *International Conference on Machine Learning*, PMLR, 2020, pp. 3929–3938.

[28] E. Hoque, O. Hoeber, and M. Gong, "Evaluating the trade-offs between diversity and precision for web image search using concept-based query expansion," in *2011 IEEE/WIC/ACM International Conferences on Web Intelligence and Intelligent Agent Technology*, IEEE, vol. 3, 2011, pp. 130–133.

[29] G. Izacard, M. Caron, L. Hosseini, *et al.*, "Towards unsupervised dense information retrieval with contrastive learning," *arXiv preprint arXiv:2112.09118*, 2021.

[30] G. Izacard, M. Caron, L. Hosseini, *et al.*, "Unsupervised dense information retrieval with contrastive learning," 2022.

[31] G. Izacard and E. 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*, Online: Association for Computational Linguistics, Apr. 2021, pp. 874–880. DOI: 10.18653/v1/2021.eacl-main.74. [Online]. Available: https://aclanthology.org/2021.eacl-main.74.

[32] J. Jeon, W. B. Croft, and J. H. Lee, "Finding similar questions in large question and answer archives," in *Proceedings of the 14th ACM international conference on Information and knowledge management*, 2005, pp. 84–90.

[33] E. Kamalloo, N. Dziri, C. L. Clarke, and D. Rafiei, "Evaluating open-domain question answering in the era of large language models," *arXiv preprint arXiv:2305.06984*, 2023.

[34] P. Karisani, M. Rahgozar, and F. Oroumchian, "A query term re-weighting approach using document similarity," *Information Processing & Management*, vol. 52, no. 3, pp. 478–489, 2016.

[35] V. Karpukhin, B. Oğuz, S. Min, *et al.*, "Dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2004.04906*, 2020.

[36] O. Khattab, C. Potts, and M. Zaharia, "Relevance-guided supervision for openqa with colbert," *Transactions of the association for computational linguistics*, vol. 9, pp. 929–944, 2021.

[37] O. Khattab, C. Potts, and M. Zaharia, "Relevance-guided supervision for openqa with colbert," *Transactions of the Association for Computational Linguistics*, vol. 9, pp. 929–944, 2021.

[38] T. Kwiatkowski, J. Palomaki, O. Redfield, *et al.*, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, 2019.

[39] J. Lee, S. Yun, H. Kim, M. Ko, and J. Kang, "Ranking paragraphs for improving answer recall in open-domain question answering," *arXiv preprint arXiv:1810.00494*, 2018.

[40] J. Lee, S. Yun, H. Kim, M. Ko, and J. Kang, "Ranking paragraphs for improving answer recall in open-domain question answering," *arXiv preprint arXiv:1810.00494*, 2018.

[41] K. Lee, M.-W. Chang, and K. Toutanova, "Latent retrieval for weakly supervised open domain question answering," in *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, Florence, Italy: Association for Computational Linguistics, Jul. 2019, pp. 6086–6096. DOI: 10.18653/v1/P19-1612. [Online]. Available: https://www.aclweb.org/anthology/P19-1612.

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

[43] P. Lewis, E. Perez, A. Piktus, *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 9459–9474.

[44] B. Z. Li, S. Min, S. Iyer, Y. Mehdad, and W.-t. Yih, "Efficient one-pass end-to-end entity linking for questions," *arXiv preprint arXiv:2010.02413*, 2020.

[45] J. Lin, X. Ma, S.-C. Lin, J.-H. Yang, R. Pradeep, and R. Nogueira, "Pyserini: A python toolkit for reproducible information retrieval research with sparse and dense representations," in *SIGIR*, 2021.

[46] J. Lin, R. Nogueira, and A. Yates, "Pretrained transformers for text ranking: BERT and beyond," *Synthesis Lectures on Human Language Technologies*, vol. 14, no. 4, pp. 1–325, 2021. DOI: 10.2200/S01123ED1V01Y202108HLT053.

[47] Y. Liu, M. Ott, N. Goyal, *et al.*, "RoBERTa: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[48] Z. Liu and Y. Shao, "Retromae: Pre-training retrieval-oriented transformers via masked auto-encoder," *arXiv preprint arXiv:2205.12035*, 2022.

[49] D. Mahler, "Holistic query expansion using graphical models.," *New Directions in Question Answering*, vol. 2004, pp. 203–227, 2004.

[50] Y. Mao, P. He, X. Liu, *et al.*, "Generation-augmented retrieval for open-domain question answering," *arXiv preprint arXiv:2009.08553*, 2020.

[51] M. E. Maron and J. L. Kuhns, "On relevance, probabilistic indexing and information retrieval," *Journal of the ACM (JACM)*, vol. 7, no. 3, pp. 216–244, 1960.

[52] S. Min, D. Chen, H. Hajishirzi, and L. Zettlemoyer, "A discrete hard em approach for weakly supervised question answering," *arXiv preprint arXiv:1909.04849*, 2019.

[53] S. Min, V. Zhong, L. Zettlemoyer, and H. Hajishirzi, "Multi-hop reading comprehension through question decomposition and rescoring," *arXiv preprint arXiv:1906.02916*, 2019.

[54] Y. Nie, S. Wang, and M. Bansal, "Revealing the importance of semantic retrieval for machine reading at scale," *arXiv preprint arXiv:1909.08041*, 2019.

[55] Y. Nie, S. Wang, and M. Bansal, "Revealing the importance of semantic retrieval for machine reading at scale," *arXiv preprint arXiv:1909.08041*, 2019.

[56] R. Nogueira and K. Cho, "Task-oriented query reformulation with reinforcement learning," *arXiv preprint arXiv:1704.04572*, 2017.

[57] R. Nogueira, W. Yang, J. Lin, and K. Cho, "Document expansion by query prediction," *arXiv preprint arXiv:1904.08375*, 2019.

[58] P. Ogilvie, E. Voorhees, and J. Callan, "On the number of terms used in automatic query expansion," *Information Retrieval*, vol. 12, no. 6, pp. 666–679, 2009.

[59] E. Perez, P. Lewis, W.-t. Yih, K. Cho, and D. Kiela, "Unsupervised question decomposition for question answering," *arXiv preprint arXiv:2002.09758*, 2020.

[60] P. Qi, X. Lin, L. Mehr, Z. Wang, and C. D. Manning, "Answering complex open-domain questions through iterative query generation," *arXiv preprint arXiv:1910.07000*, 2019.

[61] Y. Qu, Y. Ding, J. Liu, *et al.*, "Rocketqa: An optimized training approach to dense passage retrieval for open-domain question answering," *arXiv preprint arXiv:2010.08191*, 2020.

[62] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[63] C. Raffel, N. Shazeer, A. Roberts, *et al.*, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020. [Online]. Available: `http://jmlr.org/papers/v21/20-074.html`.

[64] R. Ren, Y. Qu, J. Liu, *et al.*, "Rocketqav2: A joint training method for dense passage retrieval and passage re-ranking," *arXiv preprint arXiv:2110.07367*, 2021.

[65] J. J. Rocchio Jr, "Relevance feedback in information retrieval," *The SMART retrieval system: experiments in automatic document processing*, 1971.

[66] D. S. Sachan, M. Lewis, D. Yogatama, L. Zettlemoyer, J. Pineau, and M. Zaheer, "Questions are all you need to train a dense passage retriever," *arXiv preprint arXiv:2206.10658*, 2022.

[67] T. F. Smith, M. S. Waterman, *et al.*, "Identification of common molecular subsequences," *Journal of molecular biology*, vol. 147, no. 1, pp. 195–197, 1981.

[68] N. Thakur, N. Reimers, A. Rücklé, A. Srivastava, and I. Gurevych, "BEIR: A heterogeneous benchmark for zero-shot evaluation of information retrieval models," in *Thirty-fifth Conference on Neural Information Processing Systems Datasets and Benchmarks Track (Round 2)*, 2021.

[69] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, "Attention is all you need," *Advances in neural information processing systems*, vol. 30, 2017.

[70] S. Wang, M. Yu, X. Guo, *et al.*, "R^3: Reinforced ranker-reader for open-domain question answering," in *AAAI*, 2018.

[71] X. Wang, C. Macdonald, and I. Ounis, "Improving zero-shot retrieval using dense external expansion," *Information Processing & Management*, vol. 59, no. 5, p. 103 026, 2022.

[72] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage bert: A globally normalized bert model for open-domain question answering," *arXiv preprint arXiv:1908.08167*, 2019.

[73] Z. Wang, P. Ng, X. Ma, R. Nallapati, and B. Xiang, "Multi-passage bert: A globally normalized bert model for open-domain question answering," *arXiv preprint arXiv:1908.08167*, 2019.

[74] T. Wolfson, M. Geva, A. Gupta, *et al.*, "Break it down: A question understanding benchmark," *Transactions of the Association for Computational Linguistics*, vol. 8, pp. 183–198, 2020.

[75] L. Xiong, C. Xiong, Y. Li, *et al.*, "Approximate nearest neighbor negative contrastive learning for dense text retrieval," in *ICLR*, 2021.

[76] M. Yadegari, E. Kamalloo, and D. Rafiei, "Detecting frozen phrases in open-domain question answering," in *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, 2022, pp. 1990–1996.

[77] R. Yan and A. Hauprmann, "Query expansion using probabilistic local feedback with application to multimedia retrieval," in *Proceedings of the sixteenth ACM conference on Conference on information and knowledge management*, 2007, pp. 361–370.

[78] P. Yang, H. Fang, and J. Lin, "Anserini: Enabling the use of lucene for information retrieval research," in *Proceedings of the 40th international ACM SIGIR conference on research and development in information retrieval*, 2017, pp. 1253–1256.

[79] W. Yang, Y. Xie, A. Lin, *et al.*, "End-to-end open-domain question answering with bertserini," *arXiv preprint arXiv:1902.01718*, 2019.

[80] W. Yang, Y. Xie, A. Lin, *et al.*, "End-to-end open-domain question answering with bertserini," *arXiv preprint arXiv:1902.01718*, 2019.

[81] Z. Yang, P. Qi, S. Zhang, *et al.*, "Hotpotqa: A dataset for diverse, explainable multi-hop question answering," *arXiv preprint arXiv:1809.09600*, 2018.

[82] X. Yao, B. Van Durme, and P. Clark, "Automatic coupling of answer extraction and information retrieval," in *Proceedings of the 51st Annual Meeting of the Association for Computational Linguistics (Volume 2: Short Papers)*, 2013, pp. 159–165.

[83] H. Yu, C. Xiong, and J. Callan, "Improving query representations for dense retrieval with pseudo relevance feedback," *arXiv preprint arXiv:2108.13454*, 2021.

[84] H. Zhang, Y. Gong, Y. Shen, J. Lv, N. Duan, and W. Chen, "Adversarial retriever-ranker for dense text retrieval," *arXiv preprint arXiv:2110.03611*, 2021.

[85] Z. Zhang, Q. Wang, L. Si, and J. Gao, "Learning for efficient supervised query expansion via two-stage feature selection," in *Proceedings of the 39th International ACM SIGIR conference on Research and Development in Information Retrieval*, 2016, pp. 265–274.

[86] G. Zheng and J. Callan, "Learning to reweight terms with distributed representations," in *Proceedings of the 38th international ACM SIGIR conference on research and development in information retrieval*, 2015, pp. 575–584.

[87] Z. Zheng, K. Hui, B. He, X. Han, L. Sun, and A. Yates, "Bert-qe: Contextualized query expansion for document re-ranking," *arXiv preprint arXiv:2009.07258*, 2020.