eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

HAP/LAP Master
Master's thesis

Unsupervised information retrieval using large language models

Author

Carlos Domínguez Becerril

2022

eman ta zabal zazu
Universidad del País Vasco
Euskal Herriko Unibertsitatea
INFORMATIKA FAKULTATEA
FACULTAD DE INFORMÁTICA

HAP/LAP Master
Master's thesis

Unsupervised information retrieval using large language models

Author

Carlos Domínguez Becerril

Supervisors
Eneko Agirre Bengoa and Gorka Azkune Galparsoro

With the help of
Jon Ander Campos and Aitor Soroa

Abstract

Nowadays, to tackle the open domain Question Answering (QA) problem, a neural architecture with two main components is usually used: the information retriever, whose task is to search for the most relevant documents with respect to the question, and the reader, which given the question and the extracted documents that serve as context, generates the appropriate answer.

In this project, we propose to investigate open domain QA, with a special focus on the first component of the architecture, that is, the information retrieval component. We want to train several dense neural retrievers in an unsupervised manner by generating questions from the documents using Large Language Models (LLM). Currently, most LLMs provide several checkpoints with different amounts of parameters, and we want to use those checkpoints to generate questions, train a dense neural retriever system for each LLM checkpoint, and finally, compare if the generated questions have any influence in the performance of the systems. The developed system must be able to search for the necessary information in external sources, usually organized in text documents. For that purpose, the BEIR benchmark [Thakur et al., 2021] will be used in a zero-shot manner to test the performance.

As the result of this exploration, we found that using LLMs to generate questions can be helpful in order to train information retrieval systems as it achieves similar performance to supervised systems. More concretely, we found that: (i) the more parameters the LLM has the more performance we obtain, (ii) using sampling to generate questions can further increase the performance, and (iii) generating more questions using a smaller language model is not worth it as a checkpoint with more parameters can do a better job. Moreover, we found that using different prompts and/or domain adaptation on a specific dataset can improve the performance slightly.

Keywords: Artificial Intelligence, Deep Learning, Natural Language Processing, Dense Passage Retrieval, Question Generation, Unsupervised training.

Contents

Abstract i

Contents iii

List of figures vii

List of tables ix

1 Introduction 1

2 Related work 5

2.1 Deep learning architectures 5

2.1.1 Multilayer perceptron 5

2.1.2 Recurrent neural Networks 6

2.1.3 Transformers 8

2.1.4 Two tower architecture 9

2.2 Open domain information retrieval architectures 9

2.2.1 Supervised systems 10

2.2.2 Unsupervised systems 11

2.2.3 Evaluation datasets 13

3 Generating synthetic datasets for information retrieval **19**

3.1 Dataset generation using Contriever 19

3.2 Large language models for question generation 19

3.3 Open Pre-trained Transformer (OPT) 24

3.4 Dataset generation using OPT 24

4 Experiments and Results **29**

4.1 Training 29

4.2 Experiments 30

4.3 Evaluation 32

4.4 Metrics 33

4.5 Using hard negatives for training 34

4.6 Domain adaptation 35

4.7 Results 35

4.7.1 Is sampling useful to generate questions? 36

4.7.2 Does increasing the number of parameters of the model for generating questions improve the performance of the systems? 36

4.7.3 Does increasing the number of examples improve the performance? 39

4.7.4 Is the crop generation method better than Large Language Models? 40

4.7.5 Does domain adaption improve the performance of the systems? . 41

4.7.6 Does prompting improve the generation of questions? 42

4.7.7 Does using hard negatives allow to improve the performance of the system? 43

4.7.8 How does our system compare to other state-of-the-art systems? . 44

4.7.9 Summary 46

5 Conclusions and future work **47**

5.1 Conclusions 47

5.2 Future work 48

Appendix

A Appendix **53**

A.1 Experiment results for Recall@100 53

A.1.1 Is sampling useful to generate questions? 54

A.1.2 Does increasing the number of parameters of the model for generating questions improve the performance of the systems? 55

A.1.3 Does increasing the number of examples improve the performance? 56

A.1.4 Is the crop generation method better than Large Language Models? 57

A.1.5 Does domain adaption improve the performance of the systems? . 58

A.1.6 Does prompting improve the generation of questions? 58

A.1.7 Does using hard negatives allow to improve the performance of the system? 59

A.1.8 How does our system compare to other state-of-the-art systems? . 60

A.2 Project objectives report 61

A.2.1 Project description and goals 61

A.2.2 Project planning 62

A.2.3 Methodology 66

A.2.4 Risks 66

Bibliography **69**

List of figures

2.1 MLP architecture diagram with an input layer, k hidden layers, and an output layer. Source: Stanford university. 6

2.2 RNN architecture diagram, where a is the hidden-state. Source: Stanford university. 7

2.3 The encoder-decoder structure of the Transformer architecture. Taken from "Attention Is All You Need"[Vaswani et al., 2017]. 8

2.4 Two tower architecture for question-answering systems (retrieval part). . . 9

3.1 Prompt used to generate the questions. For the fourth example, we replace {document text} with a document from the MS MARCO dataset. The same prompt was used by [Bonifacio et al., 2022a]. 22

3.2 Graph showing the time required for generating the dataset using different OPT checkpoints. 26

A.1 Work Breakdown Structure of the project. 63

A.2 Gantt chart of the project. 65

List of tables

2.1 The MS MARCO dataset format. 14

2.2 The MS MARCO dataset splits. 14

2.3 An example of the MS MARCO dataset. 15

2.4 The Scifact dataset splits. 15

2.5 An example of the Scifact dataset. 16

2.6 The NFCorpus dataset splits. 16

2.7 An example of the NFCorpus dataset. 16

3.1 Examples of how crop method generates (Contriever generation method) query-document pairs. 20

3.2 Comparison of automatically generated questions using different models in a zero-shot setting. 23

3.3 Available OPT models and their number of parameters. #L refers to the number of layers, #H refers to the number of attention heads, and d_{model} refers to the embedding size. 24

3.4 Top 10 starting words for each OPT checkpoint. † indicates that the dataset has been generated using 8-bit optimization techniques. x5 means that 5 questions per document have been generated. 25

3.5 Time required to generate the dataset using the different available OPT checkpoints. To generate the dataset, we are making use of Nvidia A100 GPUs with 80GB of VRAM memory. † indicates that the dataset has been generated using 8-bit optimization techniques. x5 means that 5 questions per document have been generated. 26

3.6 The number of questions lost during inference from 8,069,749 documents. † indicates that the dataset has been generated using 8-bit optimization techniques. x5 means that 5 questions per document have been generated. 27

4.1 Time required to train DPR and Contriever systems. 30

4.2 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether sampling is helpful to improve the performance of the systems. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. 37

4.3 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether increasing the number of parameters is helpful to improve the performance of the systems. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. 38

4.4 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether generating more questions faster using a smaller checkpoint is better than generating slower high-quality questions. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. x5 means that five questions have been generated per document. 39

4.5 In-domain and zero-shot performances on BEIR benchmark. In this table, we show the performance of using the crop method to generate queries-document pairs. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. 40

4.6 In-domain and zero-shot performances on BEIR benchmark. In this table, we show whether domain adaption is beneficial to improve the performance of datasets from another domain. All scores denote NDCG@10. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised. 41

4.7 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether changing the prompt to generate questions allows to improve the performance. All scores denote NDCG@10. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised. 43

4.8 In-domain and zero-shot performances on BEIR benchmark. In this table, we check whether hard negatives allow to improve the performance of the systems trained. All scores denote NDCG@10. + indicates that the system has been finetuned on top of marco-supervised. 43

4.9 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare our system to other state-of-the-art systems. All scores denote NDCG@10. The best scores for supervised and unsupervised settings on a given dataset are marked in bold, in the case of the unsupervised setting the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. . . . 45

A.1 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether sampling is helpful to improve the performance of the systems. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. 54

A.2 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether increasing the number of parameters is helpful to improve the performance of the systems. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. 55

A.3 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether generating more questions faster using a smaller checkpoint is better than generating slower high-quality questions. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. x5 means that five questions have been generated per document. 56

A.4 In-domain and zero-shot performances on BEIR benchmark. In this table, we show the performance of using the crop method to generate queries-document pairs. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. 57

A.5 In-domain and zero-shot performances on BEIR benchmark.In this table, we show whether domain adaption is beneficial to improve the performance of datasets from another domain. All scores denote Recall@100. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised. 58

A.6 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether changing the prompt to generate questions allows to improve the performance. All scores denote Recall@100. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised. 58

A.7 In-domain and zero-shot performances on BEIR benchmark. In this table, we check whether hard negatives allow to improve the performance of the systems trained. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. + indicates that the system has been finetuned on top of marco-supervised. . 59

A.8 In-domain and zero-shot performances on BEIR benchmark. In this table, we compare our system to other state-of-the-art systems. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. 60

A.9 Time estimates for each work unit. 65

A.10 Deliverables and their deadlines. 66

CHAPTER 1

Introduction

One of the most important uses we give to our devices, both computers and smartphones, is to search for information on a specific topic. To do this, we usually resort to search engines integrated into our web browsers, or increasingly, we use virtual assistants such as Alexa, Siri, or Google Assistant. One of the key technologies behind these assistants are Question Answering (QA) engines, which, given a question in text format, generate the appropriate answer, also in text.

Since the breakthrough of deep learning a decade ago, the world of natural language processing has advanced significantly, and the field of QA has been no exception. Today, we have deep neural networks that show satisfactory performance in various QA tasks. When talking about QA, there are actually several variants of the task that result in different systems or approaches.

Open domain question answering is a type of artificial intelligence (AI) that involves building a system that can answer a wide range of questions on any topic. In contrast to closed domain QA systems, which are designed to answer questions within a specific domain or on a specific topic, open domain QA systems are designed to be able to answer any question on any topic.

Nowadays, to tackle the open domain QA problem, a neural architecture with two main components is usually used:

- **The information retriever**: its task is to search for the most relevant documents with respect to the question.

- **The reader**: given the question and the extracted documents that serve as context, its goal is to generate the appropriate answer.

The most successful system following this paradigm is the so-called Fusion in Decoder (FiD) [Izacard and Grave, 2020]. This system is based on the well-known neural information retriever Dense Passage Retrieval (DPR) [Karpukhin et al., 2020]. In it, two neural text encoders are used to encode both the question and each of the accessible documents, which are trained contrastively. In this way, a document index is constructed according to the representation generated by the corresponding text encoder. Given a new query, and making use of the similarity between representations, DPR generates a ranking of the most relevant documents. The idea of FiD is to process these documents in a seq2seq (sequence-to-sequence) system based on a language model. This architecture has the advantage that it can use several documents at the same time to generate a single answer since the evidences can be distributed.

Since the arrival of DPR and FiD, the community has been working to develop new information retrieval systems given the problems with the original DPR. The premise being followed is that if the information retriever is improved, the reader component will be able to find better the right answers. Following this philosophy, the following information retrievers can be distinguished:

- **Retrieval with identifiers**: One approach to tackle the information retrieval with autoregressive models makes use of identifiers, that is, string pointers to documents that are easier to generate than the full document itself. In tasks where such data are available, such as Wikipedia-based entity linking (a form of page-level retrieval), titles have been shown to work well as identifiers [Cao et al., 2020].

- **Term weighting**: All modern approaches for sparse information retrieval based on string matching make use of bag-of-words, document indexing with an inverted index, a data structure that maps terms to documents, or, more generally locations in a corpus [Robertson and Zaragoza, 2009]. Retrieval performance in this environment depends heavily on term weighting schemes, and many recent papers propose sophisticated and contextualized weightings for both queries and documents [Formal et al., 2021].

- **Query/document expansion**: A line of research that often involves autoregressive language models is that of document and query expansion. For example, stored

documents can be augmented by generating possible queries that could be answered by them [Nogueira et al., 2019].

- **Query probability models**: Another connected line of research is query probability models, which, in their latest incarnations, use autoregressive models to rerank passages according to the probability $P(q|p)$ of a given query q [Lesota et al., 2021].
- **"Learning to Google"**: Recently, it has been shown that language models can directly generate search queries for modern web search engines [Lazaridou et al., 2022].

All these systems are supervised, that is, for each question, the documents in which the answers appear have been annotated. Annotating databases of this type is very expensive, so the generated systems cannot scale easily. Therefore, some unsupervised neural systems have also been proposed [Izacard et al., 2021], although little research has yet been done in this area. In fact, the reference unsupervised information retriever is still BM25 [Robertson and Zaragoza, 2009], which is not neural.

In this project, we propose to investigate open domain QA, with a special focus on the first component of the architecture, that is, the information retrieval component. We want to train several DPR systems in an unsupervised manner by generating questions from the documents using Large Language Models (LLM). Currently, most LLMs provide several checkpoints with different amounts of parameters, and we want to use these checkpoints to generate questions, train a DPR system for each LLM checkpoint, and finally, compare if the generated questions have any influence in the performance of the systems. The developed system must be able to search for the necessary information in external sources, usually organized in text documents. For that purpose, the BEIR benchmark [Thakur et al., 2021] will be used in a zero-shot manner to test the performance.

Overall, we want to answer the following questions:

1. Whether using LLMs with more parameters improves the quality of generated questions, and hence the information retrieval component.
2. Whether it is more relevant to generate a lot of low-quality questions faster using smaller language models or less high-quality questions slower using bigger models. Quantity vs. Quality vs. Time.
3. Whether the prompts used to generate the questions affect the performance of the IR systems.

4. Whether it is better to generate questions by selecting the most probable word or by sampling it.

5. To empirically check which method is the best to generate query-document pairs given a new domain for IR.

Some other contributions include:

1. The generation of three datasets in an unsupervised manner using different methods as explained in Chapter 3.

2. A thorough evaluation of the datasets by training several systems and measuring their performance.

For reproducibility, the code is available at https://github.com/CarlosDominguezBecerril/InformationRetrieval

CHAPTER 2

Related work

To understand the following chapters well, some background knowledge is introduced that will become essential in the coming explanations. Because it is not possible to go in-depth into all the theoretical aspects that would be required to fully understand the content in the next chapters, the reader is assumed to have an introductory level understanding of basic Deep Learning (DL) techniques. The reader is referred to [Goodfellow et al., 2016] for a much more complete coverage on DL techniques.

Finally, we will introduce the three datasets used to evaluate our system.

2.1 Deep learning architectures

2.1.1 Multilayer perceptron

The goal of a multilayer perceptron is to approximate some function f^*. For example, a classifier $y = f^*(x)$ maps an input x to a category y. A multilayer perceptron defines a mapping $y = f(x;\theta)$ and learns the value of the parameters θ that result in the best function approximation.

Multilayer perceptrons are called multilayer because they are typically represented by composing together many different functions. The model is associated with a directed acyclic graph describing how the functions are composed together. For example, given three functions $f^{(1)}$, $f^{(2)}$, and $f^{(3)}$ connected in a chain, to form $f(x) = f^{(3)}(f^{(2)}(f^{(1)}(x)))$.

These chained structures are the most commonly used structures of neural networks, where $f^{(1)}$ refers to the first layer, $f^{(2)}$ to the second layer, and $f^{(3)}$ to the third layer. The first layer of a multilayer perceptron is called the input layer, the intermediate layers are called hidden layers, and the final layer is called the output layer. The number of layers determines the depth of the model. An example of this architecture can be seen in Figure 2.1.

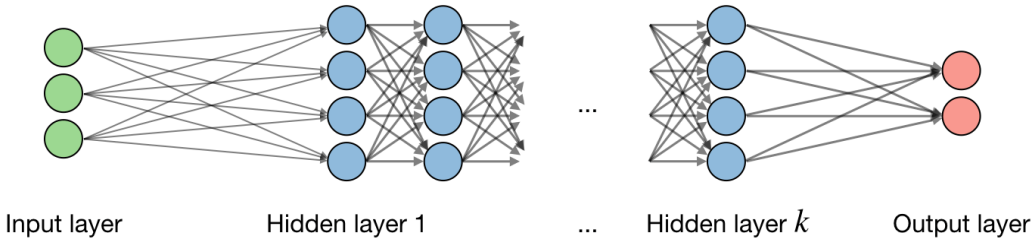

Figure 2.1: MLP architecture diagram with an input layer, k hidden layers, and an output layer. Source: Stanford university.

If a multilayer perceptron has a linear activation function in all neurons, that is, a linear function that maps the weighted inputs to the output of each neuron, then linear algebra shows that any number of layers can be reduced to a two-layer input-output model. In MLPs some neurons use a nonlinear activation function that was developed to model the frequency of action potentials, or firing, of biological neurons.

Two common non-linear activation functions are the sigmoid (σ) and the rectified linear unit ($ReLU$) functions described by Equations 2.1 and 2.2:

$$\sigma(x) = \frac{1}{1+e^{-x}} \tag{2.1}$$

$$ReLU(x) = max(0,x) \tag{2.2}$$

There are various ways to learn the suitable weights of a neural network but this project will only focus on backpropagation [Rumelhart et al., 1986] combined with gradient-based optimization techniques such as stochastic gradient descent [Robbins, 2007].

2.1.2 Recurrent neural Networks

Recurrent neural networks (RNNs) are a family of deep learning architectures that are specialized in processing data of sequential nature. RNNs are well-suited for several NLP

tasks due to the sequential essence of language and they are used extensively in the field. Some examples of the use of RNNs in NLP and language-related topics are machine translation [Wu et al., 2016], language modeling [Józefowicz et al., 2016], reading comprehension [Shen et al., 2017], question-answering [Andreas et al., 2016], and text summarization [Mahmood and Len, 2017] to name a few.

There are many types of RNNs, however, they are all based on the same fundamental ideas. A sequence of elements are processed one by one and to process any element, two inputs are needed: a vector representation of the element and a state vector (also called hidden-state) which encodes all the elements seen so far. Using these two inputs a RNN cell will produce the next hidden-state as an output which also can be used to feed the next RNN cell state.

A RNN at time-step t has hidden-state h_{t-1} and processes an element x_t from a sequence x. The RNN computes the next hidden-state h_t as function f of the previous hidden-state h_{t-1} and the element that is being processed x_t as seen in Figure 2.2.

$$h_t = f(x_t, h_{t-1}) \tag{2.3}$$

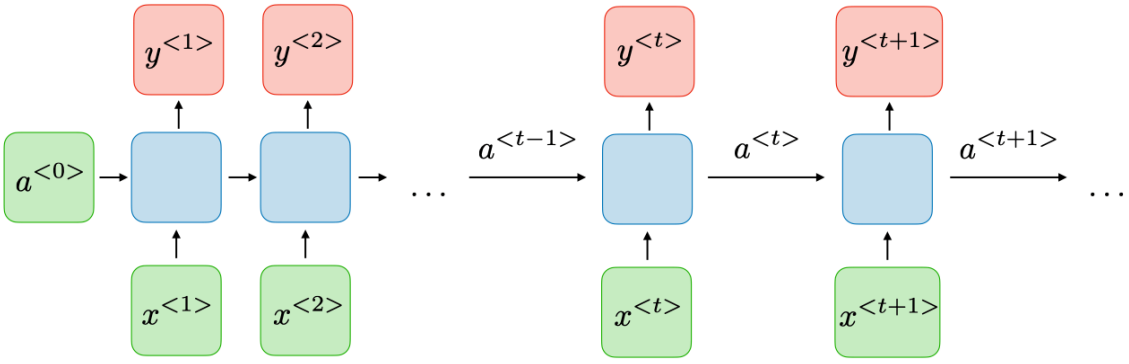

Figure 2.2: RNN architecture diagram, where a is the hidden-state. Source: Stanford university.

f needs to be suitable for DL techniques, therefore, it has to be a parametric function $f = f_\theta$ with parameters θ and differentiable with respect to θ, i.e, $\exists \frac{df}{d\theta}$ to be able to use gradient-based methods and optimize the parameters efficiently.

Finally, f should be a non-linear function so that the capacity of the model can scale with the depth of the deep learning model. The choice of f is important, for that reason, different types of RNN architectures have been proposed such as Long-Short Term Memory cells (LSTM) [Hochreiter and Schmidhuber, 1997] or Gated Recurrent Units (GRU) [Cho et al., 2014] to name a few.

2.1.3 Transformers

Transformers are deep learning models that use the mechanism of self-attention, differentially weighting the significance of each part of the input data. Like recurrent neural networks (RNNs), transformers are designed to process sequential input data, such as natural language. Some examples of the use of transformers in NLP and language-related topic are machine translation [Wu et al., 2016], language modeling [Józefowicz et al., 2016], reading comprehension [Shen et al., 2017], question-answering [Andreas et al., 2016], and text summarization [Mahmood and Len, 2017] to name a few.

The main difference between transformers and RNNs is that transformers process the entire input all at once, whereas RNNs process the input in steps. This can be done because the attention mechanism provides context for any position in the input sequence, which at the same time, allows for more parallelization than RNNs. An example of this architecture can be seen in Figure 2.3

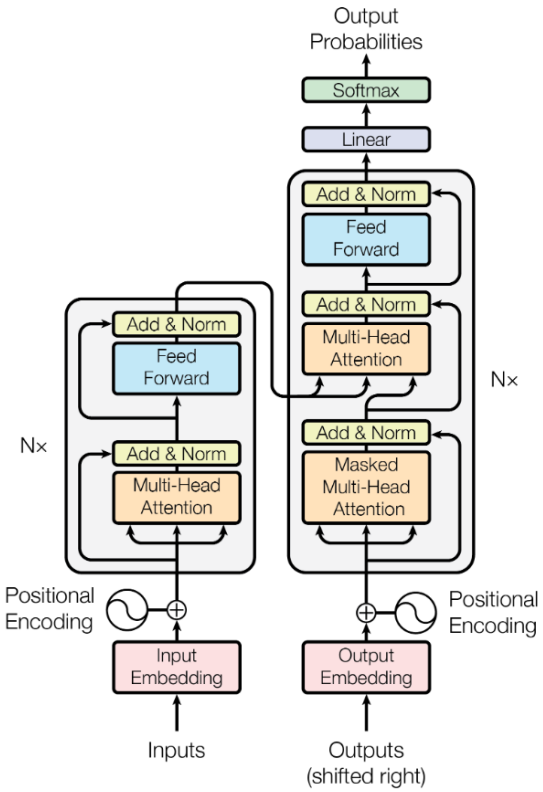

Figure 2.3: The encoder-decoder structure of the Transformer architecture. Taken from "Attention Is All You Need"[Vaswani et al., 2017].

2.1.4 Two tower architecture

The two tower architecture is a system that makes use of two neural networks to obtain embedding representations. The first neural network (query encoder) matches query features x_{query} to query embeddings $\psi(x_{query})$ and the second one (context encoder) matches context features $x_{context}$ to context embeddings $\phi(x_{context})$. The output of the model can be defined as the dot product of $< \psi(x_{query}), \phi(x_{context}) >$, which can be seen as the similarity between the query and the context.

Some examples of the use of the two tower architectures are recommendation systems [Wang et al., 2021] and question-answering [Andreas et al., 2016] systems. An example of this architecture can be seen in Figure 2.4

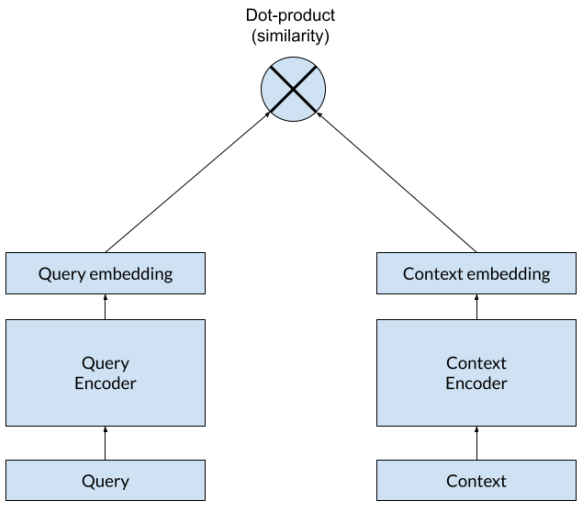

Figure 2.4: Two tower architecture for question-answering systems (retrieval part).

2.2 Open domain information retrieval architectures

Open domain Question-Answering (QA) is an important task in Natural Language Processing (NLP), which aims to answer a question in the form of natural language based on large-scale unstructured documents. Nowadays, to tackle the open domain QA problem, a neural architecture with two main components is usually used: the information retriever, whose task is to search for the most relevant documents with respect to the question, and the reader, which given the question and the extracted documents that serve as con-

text, generates the appropriate answer. In this section, we will provide some background information about State-Of-The-Art (SOTA) systems, both supervised and unsupervised.

2.2.1 Supervised systems

Dense Passage Retrieval

Dense Passage Retrieval (DPR) [Karpukhin et al., 2020] is focused on the retrieval component of open domain QA. Given a collection of M text passages, the goal is to index all the passages in a low-dimensional and continuous space so that it can efficiently retrieve the top k passages relevant to the input question.

DPR uses a two tower model architecture, that is, it uses two encoders: the first encoder $E_P(\cdot)$ maps any text passage to a d-dimensional real valued vectors and builds an index for all the M passages that we will use for retrieval; the second encoder $E_Q(\cdot)$ maps the input question to a d-dimensional vector, and retrieves k passages of which vectors are the closest to the question vector.

The similarity between the question and the passage is defined by using the dot product of their vectors as seen in Equation 2.4.

$$sim(q,p) = E_Q(q)^T E_P(p) \tag{2.4}$$

Training the encoders so that the dot-product similarity becomes a good ranking function for retrieval is essentially a metric learning problem. The goal is to create a vector space such that relevant pairs of questions and passages will have smaller distance, that is, higher similarity, than the irrelevant ones.

Let $D = \{< q_i, p_i^+ >\}_{i=1}^m$, be the training data of m instances, where q_i is one question and p_i^+ is the passage where the answer to the question appears.

We want to optimize the negative log-likelihood of the positive passages:

$$L(q_i, p_i^+, p_{i,1}^-, ..., p_{i,n}^-) = -log \frac{e^{sim(q_i, p_i^+)}}{e^{sim(q_i, p_i^+)} + \sum_{j=1}^n e^{sim(q_i, p_{i,j}^-)}} \tag{2.5}$$

where $p_{i,j}^-$ are negative passages. The negative passages that DPR uses are the In-batch negatives, that is, for each question in the batch, we will consider the positive passages of

the other questions as the negative passages. For a batch size of size B, we will have for each question 1 positive passage and $B-1$ negative passages.

2.2.2 Unsupervised systems

BM25

Historically, in information retrieval, queries and documents are represented as sparse vectors where each element of the vectors corresponds to a term of the vocabulary. BM25 [Robertson and Zaragoza, 2009] is a ranking function that ranks a set of documents based on the query terms within a document, regardless of the inter-relationship between the query terms within a document. The ranking function is not a single function, that is, is a whole family of scoring functions, with different components and parameters.

Given a query Q, containing keywords q_1, ..., q_n, the BM25 score of a document D is:

$$score(D,Q) = \sum_{i=1}^{n} IDF(q_i) \cdot \frac{f(q_i,D) \cdot (k_1+1)}{f(q_i,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{avgdl}\right)} \tag{2.6}$$

where $f(q_i,D)$ is the number of times that q_i occurs in the document D, $|D|$ is the length of the document D in words, and $avgdl$ is the average document length in the text collection from which documents are drawn. k_1 and b are free parameters, usually chosen, in absence of an advanced optimization, as $k_1 \in [1.2, 2.0]$ and $b = 0.75$. $IDF(q_i)$ is the inverse document frequency (IDF) weight of the term q_i. It is usually computed as:

$$IDF(q_i) = ln\left(\frac{N - n(q_i) \cdot (k_1+1)}{n(q_i) + 0.5} + 1\right) \tag{2.7}$$

where N is the total number of documents in the collection, and $n(q_i)$ is the number of documents containing q_i.

BM25 is widely used by search engines to estimate the relevance of documents to a given search query. This relevance of documents can be extended to other domains such as question-answering, where BM25 works as the information retrieval system. Nevertheless, a well-known limitation of this approach is that it relies on a near-exact match to retrieve documents.

Contriever

Contriever [Izacard et al., 2021] explores the limits of contrastive pre-training to learn dense text retrievers. First of all, compared to DPR, Contriever uses negative pairs across batches, that is, it stores representations from previous batches in a queue and use them as negative examples in the loss. This approach allows for a smaller batch size but slightly changes the loss function (Equation 2.5) by making it asymmetric between the current queries of the batch and the "keys" (the elements stored in the queue). Gradient is only backpropragated through the current batch, as the representation of the "keys" is considered fixed. This method leads to a drop of performance when the network rapidly changes, but to overcome the problem [He et al., 2019] propose a system called Momentum Contrastive (MoCo). MoCo proposes to generate the representation of the "keys" from a second network updated more slowly, that is, we are going to have two networks one for the "keys" parametrized by θ_k and one for the queries parametrized as θ_q. The parameters for the query encoder are updated with backpropragation and stochastic gradient descent, similarly to when using in-batch negatives, while the parameters of the key encoder is updated from the parameters of the query network by using a exponential moving average:

$$\theta_k \leftarrow m\theta_k + (1-m)\theta_q \tag{2.8}$$

where m is the momentum parameter that takes its value in $[0,1]$.

Moreover, the Contriever paper proposes to use independent cropping when unsupervised data is supplied as training data. Independent cropping is a common independent data augmentation used for images where views are generated independently by cropping the input. In natural language processing, it would be equivalent to sampling a span of tokens. For the information retrieval task, we sample independently two spans from a document, one is going to be used as the query and the other one as the document. This will lead to an overlap between the two, hence encouraging the network to learn exact matches between the query and document, in a way that is similar to lexical matching method like BM25. Furthermore, in addition to random cropping, they also consider different data augmentation methods such as word deletion, replacement, or masking.

Promptagator

Promptagator [Dai et al., 2022] proposes to make use of Large Language Models (LLM) as a few-shot query generator, and the creation of task-specific retrievers based on the

generated data.

The key idea of promptagator is to transform a few examples into many more examples. It consists of three components: prompt-based query generation, consistency filtering, and retriever training. During prompt-based query generation, a task-specific prompt will be combined with a large language model to produce queries for the target task. The LLM used is FLAN [Wei et al., 2021] with 137 billion parameters. The next step is a filtering step that cleans the generated data based on round-trip consistency, that is, a query should be answered by the document from which the query was generated. Finally, a retriever and a cross-attention reranker will be trained based on the generated data for the task of information retrieval.

Promptagator is the closest system to what we are doing in this master thesis. The main difference is that whereas promptagator uses a unique LLM for query generation, we propose to make use of an LLM with different checkpoints (different amounts of parameters in each checkpoint) and see whether this is an important factor when generating questions. As mentioned in Section 1, a lower amount of parameters allows a faster generation of low-quality queries, whereas a large amount of parameters generates higher-quality queries but slower. Is the number of parameters important to generate good query-document pairs? Is it better to generate a lot of questions per document using a smaller LLM in the same time that it takes to generate one question per document using a large LLM?

2.2.3 Evaluation datasets

Three datasets (MS MARCO [Nguyen et al., 2016], NFCorpus [Boteva et al., 2016], Scifact [Wadden et al., 2020]) are selected to train and evaluate over systems and one benchmark (BEIR benchmark [Thakur et al., 2021]) for evaluating the results in a zero-shot manner.

From the three training datasets MS MARCO is the one that is going to be used for training the system and NFcorpus and Scifact will be used to evaluate whether domain adaption (as explained in Chapter 4.6) is helpful or not to improve the performance.

MS MARCO training dataset

MS MARCO [Nguyen et al., 2016] is a large scale MAchine Reading COmprehension dataset. The dataset comprises 1,010,916 anonymized questions sampled from Bing's

search query logs with 1,026,758 unique answers. In addition, the dataset contains 8,841,823 passages extracted from 3,563,535 web documents retrieved by Bing that provide the information necessary for curating the natural language answers.

A question in the MS MARCO dataset may have multiple answers or no answers at all, on average each question contains a set of 10 passages which may contain or not the answer to the question. All the contents and the format for each example can be found in Table 2.1.

Field	Description
Query	A question query issued to Bing.
Passages	Top 10 passages from Web documents as retrieved by Bing. The passages are presented in ranked order to human editors. The passage that the editor uses to compose the answer is annotated as *is_selected* : 1.
Document URLs	URLs of the top ranked documents for the question from Bing. The passages are extracted from these documents.
Answer(s)	Answers composed by human editors for the question, automatically extracted passages, and their corresponding documents.
Well Formed Answer(s)	Well-formed answer rewritten by human editors, and the original answer.
Segment	QA classification. E.g., tallest mountain in south america belongs to the ENTITY segment because the answer is an entity (Aconcagua).

Table 2.1: The MS MARCO dataset format.

The MS MARCO dataset is divided into three splits: train, development and test. The number of questions in each split can be found in Table 2.2. An example of how the dataset looks is found in Table 2.3

Split	Questions	Total %
Train	808,731	80%
Developement	101,093	10%
Test	101,092	10%
Total	1,010,916	100%

Table 2.2: The MS MARCO dataset splits.

Document	Query
The average Walgreens salary ranges from approximately $15,000 per year for Customer Service Associate / Cashier to $179,900 per year for District Manager. Average Walgreens hourly pay ranges from approximately $7.35 per hour [. . .]	walgreens store salary average

Table 2.3: An example of the MS MARCO dataset.

Scifact training dataset

The SciFact dataset [Wadden et al., 2020] is a unique and valuable resource for research that aims to uncover novel insights into sentiment, fact-checking, and trustworthiness of scientific claims. With 1.3K expert-written scientific claims paired with evidence-containing abstracts and contents, as well as human-generated structured annotations containing labels and rationales, this dataset provides ample opportunity for researchers to explore the nuances of science communication.

The Scifact dataset is divided into two splits: train and test. In order to be able to select the best model without checking the test dataset we will take 10% of the training dataset and convert it to a development dataset. The number of questions in each split can be found in Table 2.4. An example of how the dataset looks is found in Table 2.5.

Split	Questions	Total %
Train	828	66%
Developement	92	7%
Test	340	27%
Total	1,260	100%

Table 2.4: The Scifact dataset splits.

Document	Query
OBJECTIVES To carry out a further survey of archived appendix samples to understand better the differences between existing estimates of the prevalence of subclinical infection with prions after the bovine spongiform encephalopathy epizootic and to see whether a broader birth cohort was affected, and to understand better the implications for the management of blood and blood products and for the handling of surgical instruments. [...]	1/2000 in UK have abnormal PrP positivity.

Table 2.5: An example of the Scifact dataset.

NFCorpus training dataset

NFCorpus dataset [Boteva et al., 2016] is a full-text English retrieval dataset for Medical Information Retrieval. It contains a total of 134,294 query-pair documents extracted from the NutrionFacts.org site and PubMed.

The NFCorpus dataset is divided into three splits: train, development, and test. The number of questions in each split can be found in Table 2.6. An example of how the dataset looks is found in Table 2.7.

Split	Questions	Total %
Train	110,575	10%
Developement	11,385	10%
Test	12,334	10%
Total	134,294	100%

Table 2.6: The NFCorpus dataset splits.

Document	Query
Lipid rafts/caveolae are membrane platforms for signaling molecules that regulate various cellular functions, including cell survival. To better understand the role of rafts in tumor progression and therapeutics, [...]	Do Cholesterol Statin Drugs Cause Breast Cancer?.

Table 2.7: An example of the NFCorpus dataset.

BEIR benchmark

Benchmarking-**IR** (BEIR) is a robust and heterogeneous evaluation benchmark for information retrieval, comprising 18 retrieval datasets for comparison and evaluation of model generalization. BEIR is focused on diversity, that is, the benchmark includes nine different retrieval tasks: fact checking, citation prediction, duplicate question retrieval, argument retrieval, news retrieval, question answering, tweet retrieval, bio-medical IR, and entity retrieval. Furthermore, datasets from diverse domains are included to cover broad topics like Wikipedia, specialized topics like COVID-19, different text types like news topics and tweets, and datasets with different sizes, query lengths, and document lengths. The statistics of each of the datasets included can be found in Table 1 of [Thakur et al., 2021].

This benchmark will be used to measure the performance of our system and have comparable results.

CHAPTER 3

Generating synthetic datasets for information retrieval

In this chapter, we introduce the different available methods for generating queries. We will focus mainly on two: Contriever [Izacard et al., 2021] which is a non-neural based method and Open Pre-trained Transformer (OPT) [Zhang et al., 2022] which is a Large Language Model (LLM)

3.1 Dataset generation using Contriever

The easiest way of generating 'queries' in an unsupervised setting, is the process that the Contriever system [Izacard et al., 2021] follows. Contriever generates queries by applying random cropping to the document itself and then deleting words with a probability of 10%. The remaining words after the process are used as the query for the example. Moreover, Contriever applies the same process to the document itself. A better explanation of how the dataset is generated can be found in Section 2.2.2. Examples of this method can be found in Table 3.1. Moving forward we will be calling to the Contriever generation method 'crop' to avoid confusion with the system itself.

3.2 Large language models for question generation

Large language models (LLM), which are often trained for hundreds of thousands of computing days, have shown remarkable capabilities for zero- and few-shot learning. Given

Document	**Query (random cropping)**	**Final Query (random cropping + words deletion)**	**Document (random cropping)**	**Final Document (random cropping + words deletion)**
Color hex is a easy to use tool to get the color codes information including color models (RGB,HSL,HSV and CMYK), css and html color codes.	Color hex is a easy to use tool	Color hex tool	get the color codes information including color models (RGB,HSL,HSV and CMYK)	color codes information color models (RGB,HSL,HSV and CMYK)
Why Visa Stock Is Dropping Today Visa stock is dropping today due to a court ruling regarding a $5.7 billion settlement deal that is being rejected by New York appeals court. Both MasterCard (MA) and Visa (V) stocks are dropping today on the news that new litigation is problem the next step in their ongoing battle with retailers over allegations that their credit-card fees were improperly fixed.	$5.7 billion settlement deal that is being rejected by New York appeals court. Both MasterCard (MA) and Visa (V) stocks are dropping today on the news that new litigation	$5.7 billion settlement deal rejected by New York. Both MasterCard and Visa dropping news that new litigation	stocks are dropping today on the news that new litigation is problem the next step in their ongoing battle with retailers	stocks dropping on the news new litigation is next battle with retailers
Although the European powers did make military interventions in Latin America from time to time after the Monroe Doctrine was announced, the Americans did not look for war. They did, however, use the doctrine as justification for taking Texas in 1842 under President John Tyler.	military interventions in Latin America from time to time after the Monroe Doctrine was announced	military interventions in Latin after Monroe	the Americans did not look for war. They did, however, use the doctrine as justification for taking Texas	Americans look for war. the doctrine justification taking Texas

Table 3.1: Examples of how crop method generates (Contriever generation method) query-document pairs.

their computational cost, these models are difficult to replicate without significant capital. For the few that are available through APIs, no access is granted to the full model weights, making them difficult to study.

In this project, we check the suitability of 4 LLMs: OPT [Zhang et al., 2022], Bloom [Scao et al., 2022], GPT-neo [Black et al., 2021], and GPT-neoX [Black et al., 2022].

First of all, in this project, we want to answer whether a model with more parameters (which usually contribute to a higher quality text-generation) helps the Information Retrieval (IR) systems to improve more. In order to evaluate which model is the most appropriate one, we need to see how many checkpoints of different parameters each system offers.

- **OPT**: Checkpoints of 125 million, 350 million, 1.3 billion, 2.7 billion, 6.7 billion, 13 billion, 30 billion, 60 billion, and 175 billion parameters.
- **Bloom**: Checkpoints of 560 million, 1.1 billion , 1.77 billion, 3 billion, 7.1 billion, and 176 billion parameters.
- **GPT-neo(X)**: Checkpoints of 125 million, 1.3 billion, 2.7 billion, and 20 billion parameters. We are going to consider GPT-neo and GPT-neoX to be of the same family of models.

Looking at the checkpoints available for each transformer, we believe that OPT is the most appropriate one for this task since it offers a wider amount of checkpoints of different sizes.

Next, we measure how well each transformer is able to generate questions. In order to generate questions in an unsupervised way (zero-shot question generation), we need to make use of a prompt. Figure 3.1 shows the prompt used to generate the questions using the different models. This prompt is the same one that [Bonifacio et al., 2022a] has used to generate a dataset using GPT-3 [Brown et al., 2020]. To include diversity in the questions generated, we apply sampling and a top p of 0.9 (only the smallest set of most probable tokens with probabilities that add up to top p or higher are kept for generation.)

In Table 3.2, we compare the capability of different models of similar size in generating questions in an unsupervised setting.

Overall, in a limited qualitative analysis, we can see that the quality of generation of OPT is better than Bloom and GPT-neo(X). The reasons are the following: first, Bloom 7B

Example 1:
Document: We don't know a lot about the effects of caffeine during pregnancy on you and your baby. So it's best to limit the amount you get each day. If you are pregnant, limit caffeine to 200 milligrams each day. This is about the amount in 1½ 8-ounce cups of coffee or one 12-ounce cup of coffee.
Relevant Query: Is a little caffeine ok during pregnancy?

Example 2:
Document: Passiflora herbertiana. A rare passion fruit native to Australia. Fruits are green-skinned, white fleshed, with an unknown edible rating. Some sources list the fruit as edible, sweet and tasty, while others list the fruits as being bitter and inedible.
Relevant Query: What fruit is native to Australia?

Example 3:
Document: The Canadian Armed Forces. 1 The first large-scale Canadian peacekeeping mission started in Egypt on November 24, 1956. 2 There are approximately 65,000 Regular Force and 25,000 reservist members in the Canadian military. 3 In Canada, August 9 is designated as National Peacekeepers' Day.
Relevant Query: How large is the Canadian military?

Example 4:
Document: {document text}
Relevant Query:

Figure 3.1: Prompt used to generate the questions. For the fourth example, we replace {document text} with a document from the MS MARCO dataset. The same prompt was used by [Bonifacio et al., 2022a].

Document	OPT-1.3B	OPT-30B	Bloom 1.7B	Bloom 7B	GPT-neo 1.3B	GPT-neoX 20B
Color hex is a easy to use tool to get the color codes information including color models (RGB,HSL,HSV and CMYK), css and html color codes.	How can color hex code help me?	What color hex is?	What color is in Hex color model number 0c00a00?	Html Color Code	I need to convert the color hex codes to RGB and then I need to do this for all my website color codes for a custom website I am developing. Ar	What is CSS and HSL?
Why Visa Stock Is Dropping Today Visa stock is dropping today due to a court ruling regarding a $5.7 billion settlement deal that is being rejected by New York appeals court. Both MasterCard (MA) and Visa (V) stocks are dropping today on the news that new litigation is problem the next step in their ongoing battle with retailers over allegations that their credit-card fees were improperly fixed.	Why is Visa stock dropping today?	Why are MasterCard and Visa stocks dropping today?	Why Visa Stock Is Dropping Today	What is causing the Visa stock to drop today?	How has Visa stock dropped today?	What court ruling is being rejected?
Although the European powers did make military interventions in Latin America from time to time after the Monroe Doctrine was announced, the Americans did not look for war. They did, however, use the doctrine as justification for taking Texas in 1842 under President John Tyler.	What is the Monroe Doctrine?	How was the Monroe Doctrine used in the 1840s?	When did the US and European powers make military intervention in Latin America?	What country supported a campaign that gave Texas into Mexico's hands?	What is the Monroe Doctrine?	When did the United States make military interventions in Latin America?

Table 3.2: Comparison of automatically generated questions using different models in a zero-shot setting.

and GPT-neo 1.3B are not able to generate a question for the first document, second, the question generated by Bloom 7B in the third document is not correct, and finally, OPT-30B seems to be generating more diverse questions as it is using different starting words compared to Bloom and GPT-neo(X) [1]. Given this initial analysis, we decided to focus on OPT as our question generation LLM.

3.3 Open Pre-trained Transformer (OPT)

Open Pre-trained Transformer (OPT) [Zhang et al., 2022] is a series of open-sourced large causal language models which perform similarly in performance to GPT3. The different sizes of OPT and the architecture details can be found in Table 3.3.

Parameters	**#L**	**#H**	d_{model}
125M	12	12	768
350M	24	16	1024
1.3B	24	32	2048
2.7B	32	32	2560
6.7B	32	32	4096
13B	40	40	5120
30B	48	56	7128
60B	64	72	9216
175B	96	95	12288

Table 3.3: Available OPT models and their number of parameters. #L refers to the number of layers, #H refers to the number of attention heads, and d_{model} refers to the embedding size.

3.4 Dataset generation using OPT

In this project, we want to check whether the quality of the generated queries is important or not. For this purpose, let's assume that the more parameters OPT has, the higher the quality of queries is going to be, and hence the better performance of the information retrieval system. Table 3.4, shows the top 10 starting words for each OPT checkpoint.

The table shows that the top 3 words are the same regardless of the number of parameters and that they amount to about 60 to 70% of the whole dataset. The remaining words show

[1] This reasoning comes from a bigger sample of questions available at https://tinyurl.com/y2nvzkd3.

Top 10 words	125M	1.3B	30B†	125M - x5
Top 1	What (31.66%)	What (37.74%)	What (48.64%)	What (31.68%)
Top 2	How (21.38%)	How (25.65%)	How (17.23%)	How (21.40%)
Top 3	Is (10.37%)	Is (6.87%)	Is (5.92%)	Is (10.40%)
Top 4	Why (5.02%)	Can (4.79%)	Which (2.42%)	Why (5.02%)
Top 5	Can (5.00%)	Why (4.35%)	Where (2.18%)	Can (5.00%)
Top 6	Do (4.74%)	Which (2.94%)	Can (1.86%)	Do (4.74%)
Top 7	Does (4.68%)	Does (2.64%)	Why (1.51%)	Does (4.68%)
Top 8	Are (3.92%)	What's (2.17%)	Who (1.19%)	Are (3.92%)
Top 9	Who (2.41%)	Where (2.00%)	Does (1.12%)	Who (2.41%)
Top 10	Which (2.38%)	Are (1.99%)	When (1.08%)	Which (2.37%)
Total	91.56%	91.14%	83.15%	91.62%

Table 3.4: Top 10 starting words for each OPT checkpoint. † indicates that the dataset has been generated using 8-bit optimization techniques. x5 means that 5 questions per document have been generated.

that a model with fewer parameters tends to use the same starting word more frequently than the one with more parameters, this can be seen clearly when comparing the total of the top 10 words. The 125 million parameters model shows that the top 10 words appear a 91.56% of the total dataset. In contrast, the 30 billion one only 83.15%, meaning that more parameters provide a wider variety of generated questions.

Moreover, we want also to check whether it is more relevant to generate a lot of low-quality questions faster using smaller language models or less high-quality questions slower using bigger models, that is, Quantity vs. Quality vs. Time. Table 3.5 shows the time required to do inference on OPT using different amounts of parameters.

As expected, the table shows that a higher amount of parameters require more time. All the systems scale quite well in time when models with more parameters are used. The only exception is the 30 billion one, in this case, due to the high amount of parameters the memory of the GPU is filled almost completely, requiring to reduce considerably the batch size, and making the system extremely slow. The use of 8-bit optimization techniques for the 30 billion parameters reduces almost 4 times the time required to generate the dataset. This technique becomes a feasible solution to generate datasets using models with large amount of parameters as we do not lose any noticeable performance and the time is reduced considerably (See [Dettmers et al., 2022] for more information about 8-bit optimization techniques).

Parameters	Time (1 GPU)	Time (8 GPUs)
125M	32 hours (1.3 days)	4 hours
350M	48 hours (2 days)	6 hours
1.3B	80 hours (3.3 days)	10 hours
2.7B	5.3 days	0.66 days
6.7B	8 days	1 day
13B	13 days	1.63 days
30B	213 days	26.63 days
125M - x5	6.5 days	20 hours
30B†	55 days	7 days

Table 3.5: Time required to generate the dataset using the different available OPT checkpoints. To generate the dataset, we are making use of Nvidia A100 GPUs with 80GB of VRAM memory. † indicates that the dataset has been generated using 8-bit optimization techniques. x5 means that 5 questions per document have been generated.

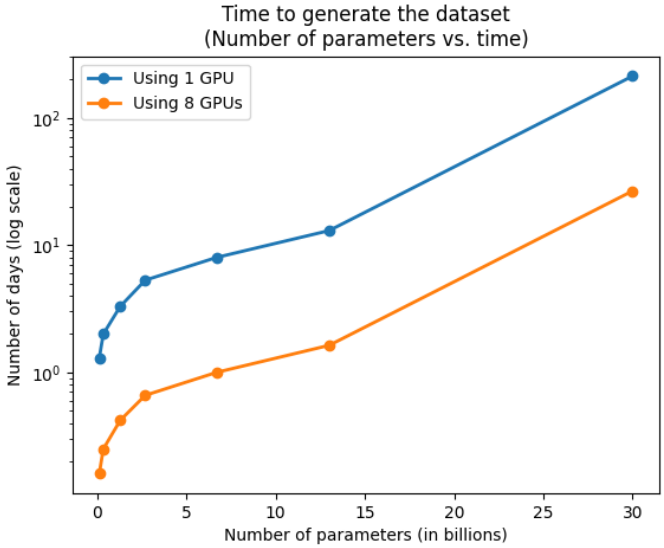

Figure 3.2: Graph showing the time required for generating the dataset using different OPT checkpoints.

After generating the questions using the train split of MS MARCO, Table 3.6 shows the number of questions obtained for each OPT checkpoint. Note that inference is not perfect and sometimes OPT is not able to generate a question, therefore, we will also be providing the number of questions lost during generation.

Parameters	**Nº Questions**	**Lost**
125M	8,062,506	7,243 (0.09%)
1.3B	8,067,497	2,252 (0.028%)
30B†	8,067,107	2,642 (0.03%)
125M - x5	40,312,055	36,690 (0.09%)

Table 3.6: The number of questions lost during inference from 8,069,749 documents. † indicates that the dataset has been generated using 8-bit optimization techniques. x5 means that 5 questions per document have been generated.

Looking at the table, the amount of lost questions during inference is insignificant compared to the whole dataset, hence we have decided to ignore these to avoid adding any kind of supervision to the project. Moreover, we can see that the more parameters the model has, the less prone it is to lose questions.

CHAPTER 4

Experiments and Results

In this chapter, we will discuss the experiments we perform, the metrics to measure the performance of the IR systems and compare them, and finally, a deep analysis of the results.

4.1 Training

In this project, we focus on two IR systems: DPR and Contriever. In the original paper of DPR [Karpukhin et al., 2020], they train the system for 40 epochs, but mention that after 25 epochs there is not an increase in performance. In our case, we have decided to train our system for only 4 epochs because MS MARCO provides for each question 10 documents on average, and hence, we will generate 10 questions from these documents that are related to the same topic. We consider that these 4 epochs are equivalent to the 40 epochs of the original DPR.

In the case of Contriever, we use their original script with the same parameters and train the system for 500,000 steps with a batch size of 64. In total, the system will see a total of 32,000,000 examples, and considering that our dataset contains around 8,000,000 examples, it is more or less equivalent to 4 epochs. The time required to train each system can be seen in Table 4.1.

System	Epochs/Steps	Time (1 GPU)	Time (2 GPUs)
DPR	4 epochs	4 days	2 days
Contriever	500,000 steps	3.5 days	1.75 days

Table 4.1: Time required to train DPR and Contriever systems.

4.2 Experiments

For this project, we want to perform as many experiments as possible using different configurations. We divide the configuration of the experiments as follows:

- Supervised IR systems
  - **marco-supervised**: We train the IR system using the MS MARCO original dataset, i.e., we use query-document pairs provided by the dataset itself, so the IR system is completely supervised.
- Unsupervised IR systems using Large Language models:
  - **opt-125m**: We train the system by generating questions for each document in the MS MARCO dataset using the OPT checkpoint with 125 million parameters. Thus this IR system is trained without human supervision.
  - **opt-350m**: We train the system by generating questions for each document in the MS MARCO dataset using the OPT checkpoint with 350 million parameters.
  - **opt-1.3b**: We train the system by generating questions for each document in the MS MARCO dataset using the OPT checkpoint with 1.3 billion parameters.
  - **opt-125m-sampling**[1]: We train the system by generating questions for each document in the MS MARCO dataset using the OPT checkpoint with 125 million parameters, applying sampling, and a top p [2] of 0.9.
  - **opt-125m-sampling-x5**: The same as opt-125m-sampling but for each document we generate 5 questions.

[1] sampling short for questions that are generated using sampling with a top p of 0.9

[2] top p: only the smallest set of most probable tokens with probabilities that add up to top_p or higher are kept for generation

  - **opt-1.3b-sampling**: We train the system by generating questions for each document in the MS MARCO dataset using the OPT checkpoint with 1.3 billion parameters, applying sampling, and a top p of 0.9.
  - **opt-30b-sampling-8bit**: We train the system by generating questions for each document in the MS MARCO dataset using the OPT checkpoint with 30 billion parameters, applying sampling, a top p of 0.9, and 8-bit quantization to reduce the inference time.

- Crop[3]:
  - **crop-queries**: We train a system by generating queries for each document in the MS MARCO dataset using the crop generation method.
  - **crop-queries-documents**: We train a system by generating queries-document pairs for each document in the MS MARCO dataset using the crop generation method.

- Domain adaptation:
  - **scifact-supervised-ft**[4]: We train a system by finetuning on the scifact supervised dataset. The IR system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).
  - **scifact-opt-30b-sampling-8bit-ft**: We train the system by generating questions for each document in the scifact dataset by using the OPT 30 billion parameter checkpoint, sampling, a top p of 0.9, and 8-bit quantization method. The system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).
  - **scifact-crop-ft**: We train the system by generating queries and documents for each document in the scifact dataset by using the crop method. The system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).
  - **nfcorpus-supervised-ft**: We train a system by finetuning the nfcorpus supervised dataset. The system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).

[3]Crop: This method refers to the Contriever generation method.
[4]ft short for systems that are finetuned on top of the ms marco supervised system

  - **nfcorpus-opt-30b-sampling-8bit-ft**: We train the system by generating questions for each document in the nfcorpus dataset using the OPT 30 billion parameter checkpoint, sampling, a top p of 0.9, and 8-bit quantization method. The system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).
  - **nfcorpus-crop-ft**: We train the system by generating queries and documents for each document in the nfcorpus dataset by using the crop method. The system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).

- Prompting:

  - **scifact-opt-30b-sampling-8bit-prompt=keywords-ft**: The same as scifact-opt-30b-sampling-8bit-ft, but in this case, before generating the question, we first generate some keywords (using OPT) and then condition the generation of the question to the document and keywords.
  - **scifact-opt-30b-sampling-8bit-prompt=title-ft**: The same as scifact-opt-30b-sampling-8bit-ft, but in this case, before generating the question, we first generate a title (using OPT) and then condition the generation of the question to the document and the title.
  - **scifact-instruct-gpt-ft**: We train a system by generating questions using InstructGPT [Ouyang et al., 2022] for each document in the scifact dataset. The system is trained starting from the supervised MS MARCO checkpoint (marco-supervised).

- Hard negatives:

  - **scifact-supervised-with-negatives-ft**: The same as scifact-supervised-ft but in this case for each document we add hard negative documents extracted using BM25. Check Chapter 4.5 for more information about hard negatives.

4.3 Evaluation

Existing neural information retrieval models have often been studied in homogeneous and narrow settings, which has considerably limited insights in out-of-domain datasets. As

mentioned in Chapter 2.2.3, [Thakur et al., 2021] developed a benchmark called **Be**nchmarking-**IR** (BEIR) to measure the capabilities of IR models outside of their domain. In this project, we will make use of BEIR in order to measure and compare the performance of the systems.

4.4 Metrics

Typical classification and regression metrics measure whether the predicted value is close to the actual value. Unfortunately, these metrics do not take into consideration the order of prediction, which is important in information retrieval, as it is not the same to find the answer in the first document or in the last one. To evaluate an information retrieval system we need to measure how relevant the results are and how good the ordering is.

The most used metric in information retrieval systems, and which BEIR uses, is the nDCG@K metric, more specifically nDCG@10 with k=10. Contriever [Izacard et al., 2021] states that nDCG@K is good at evaluating rankings returned to humans, for example in a search engine, but that Recall@100 is relevant to evaluate retrievers that are used in machine learning systems, such as question answering. In this project, we are going to provide both metrics.

In order to understand the nDCG@K metric we are going to split it into the different parts that is composed of:

- **Gain**: Gain is the relevance score for each document proposed.
- **Cumulative Gain (CG)**: Cumulative gain at K is the sum of gains of the first K document proposed. Equation 4.2 shows the formula used to calculate the cumulative gain.

$$CG@K = \sum_{i=1}^{K} G_i \tag{4.1}$$

- **Discounted Cumulative Gain (DCG)**: Discounted cumulative gain weighs each relevance score based on its position. The proposed documents at the top get a higher weight while the relevance of those at the bottom get a lower weight. Equation 4.2 shows the formula used to calculate the discounted cumulative gain.

$$DCG@K = \sum_{i=1}^{K} \frac{G_i}{log_2(i+1)} \tag{4.2}$$

- **Normalized Discounted Cumulative Gain (NDCG)**: The normalized discounted cumulative gain is the DCG with a normalization factor in the denominator. The denominator is the ideal DCG score when we proposed the most relevant documents first. Equation 4.3 shows the normalized discount cumulation gain, whereas 4.4 shows the ideal discounted cumulative gain.

$$nDCG@K = \frac{DCG@K}{iDCG@k} \tag{4.3}$$

$$iDCG@K = \sum_{i=1}^{K^{ideal}} \frac{G_i^{ideal}}{log_2(i+1)} \tag{4.4}$$

Recall@K is one of the most interpretable and popular offline evaluation metrics. It measures how many relevant documents were returned (true positives) against how many relevant documents exist in the entire dataset (true positives + false negatives).

$$Recall@K = \frac{TruePositives}{TruePositives + FalseNegatives} \tag{4.5}$$

The K in this and all other offline metrics refers to the number of items returned by the IR system. In our case, this is going to be always $K = 100$ in order to compare our system to Contriever.

4.5 Using hard negatives for training

In dense passage retrieval, hard negatives refer to passages or documents that are similar to the query but are not actually relevant or useful. These passages can be difficult to distinguish from truly relevant passages and can make it more challenging to train a model that can effectively retrieve relevant information. Hard negatives can be generated by using techniques such as active learning, where the model is used to identify passages that it is unsure about, or by using heuristics such as those based on the cosine similarity of the passage and the query. In our case, we use BM25 to extract the hard negatives, that is, for each question we calculate a score for every document and the document with the lowest score is selected as hard negative.

4.6 Domain adaptation

Domain adaptation is the process of adapting a model that has been trained on one domain (or dataset) to work effectively on a different but related domain. This is often necessary when there is a mismatch between the distribution of data on which a model was trained and the distribution of data on which it will be used in practice.

In our case, we apply domain adaption to two datasets, Scifact and NFCorpus, as they have documents of a completely different domain.

4.7 Results

In this section, we will compare the results of the systems mentioned previously in order to be able to answer the research questions proposed in Chapter 1. Concretely we will answer the following questions:

- Is sampling useful to generate questions?
- Does increasing the number of parameters of the model for generating questions improve the performance of the IR systems?
- Does increasing the number of examples improve the performance?
- Is the crop generation method better than using Large Language Models?
- Does domain adaption improve the performance of the systems?
- Does prompting improve the generation of questions?
- Does using hard negatives allow to improve the performance of the systems?
- How does our work compare to other state-of-the-art systems?

In order to measure and be able to compare all the systems equally, we are going to make use of the BEIR benchmark. We will provide the NDCG@10 and Recall@100 metrics, as they are the most useful metrics for information retrieval. We will also explain the results obtained on the NDCG@10 metric because as mentioned, this metric is good at

evaluating rankings returned to humans. As the trends for both metrics are similar, the results for Recall@100 can be found in the Appendix A.1

After carefully reviewing the results obtained using the Contriever training system using MoCo, we realized that they were not making sense and therefore, we have decided not to include these results to avoid confusion.

All the following results are obtained using the dense passage retrieval (DPR) system from Haystack end-to-end framework [5] and trained using MS MARCO dataset (unless otherwise stated). The 'crop' method refers to the Contriever generation method.

4.7.1 Is sampling useful to generate questions?

The first step we need to check is whether sampling helps generate more diverse questions and hence improve the system's performance in contrast to greedy generation. Table 4.2 show the results obtained by generating the dataset using the OPT checkpoint with 125 million and 1.3 billion parameters.

Table 4.2 shows that using sampling allows to improve the performance of the systems, and even more when the LLM has more parameters. This can be seen clearly when we compare opt-125m and opt-125m* NDCG@10 metric, if we look at every dataset we see that opt-125m* is better in 11 out of 15 datasets. On the other hand, if we increase the number of parameters we see that opt-1.3b* is better than opt-1.3b in all the datasets. Moreover, we can see that opt-125m* gets closer to opt-1.3b on average, which allows to reduce the time for generating questions while obtaining similar performance.

In this section, we conclude that sampling makes a huge difference in terms of performance. This is especially important for LLMs with a higher amount of parameters as it can boost the performance of the system considerably.

4.7.2 Does increasing the number of parameters of the model for generating questions improve the performance of the systems?

In the next set of experiments, we aim to investigate the impact on the system's performance based on the number of parameters that the question generation LLM has. Based

[5] The Haystack end-to-end framework is available at https://github.com/deepset-ai/haystack

Dataset \ model name	Greedy		Sampling	
	opt-125m	opt-1.3b	opt-125m*	opt-1.3b*
Average	27.64	<u>29.54</u>	29.10	**32.86**
MS-MARCO	18.23	<u>19.46</u>	18.74	**21.61**
TREC-covid	**41.34**	37.08	39.27	<u>39.77</u>
NFCorpus	24.29	23.39	<u>24.62</u>	**26.20**
NaturalQuestions	17.75	<u>21.92</u>	19.25	**25.49**
HotpotQA	23.45	<u>32.41</u>	32.35	**38.18**
FiQA	14.84	<u>17.80</u>	16.90	**20.76**
ArguAna	<u>45.21</u>	42.28	45.01	**47.74**
Tóuche-2020	11.18	<u>11.97</u>	10.47	**13.44**
CQAdupstack	19.85	<u>21.52</u>	20.66	**23.70**
Quora	<u>74.64</u>	74.51	72.23	**78.15**
DBpedia	19.41	22.18	<u>22.39</u>	**24.36**
Scidocs	9.73	9.35	<u>10.72</u>	**10.83**
Fever	25.58	<u>39.90</u>	31.46	**45.74**
Climate-fever	10.04	11.97	<u>12.34</u>	**13.14**
Scifact	49.58	47.25	<u>49.80</u>	**52.52**

Table 4.2: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether sampling is helpful to improve the performance of the systems. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9.

on what we have learned in the previous section, we will use sampling in each system. Table 4.3 shows the results obtained for OPT using checkpoints of 125 million, 1.3 billion, and 30 billion parameters.

Dataset \ model name	opt-125m*	opt-1.3b*	opt-30b*†
Average	29.10	32.86	**33.31**
MS-MARCO	18.74	21.61	**22.88**
TREC-covid	39.27	**39.77**	37.50
NFCorpus	24.62	26.20	**29.25**
NaturalQuestions	19.25	**25.49**	24.63
HotpotQA	32.35	38.18	**38.52**
FiQA	16.90	20.76	**22.21**
ArguAna	45.01	**47.74**	47.32
Tóuche-2020	10.47	13.44	**14.39**
CQAdupstack	20.66	23.70	**24.64**
Quora	72.23	**78.15**	75.27
DBpedia	22.39	24.36	**25.57**
Scidocs	10.72	10.83	**11.21**
Fever	31.46	45.74	**50.46**
Climate-fever	12.34	**13.14**	13.01
Scifact	49.80	**52.52**	52.35

Table 4.3: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether increasing the number of parameters is helpful to improve the performance of the systems. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset.

Table 4.3 shows that the performance of the models generally increases as the number of parameters increases. However, this is not always the case as we can see in the NDCG@10 results of the TREC-covid, Natural Questions, ArguAna, Quora, Climate-fever, and Scifact datasets. In these datasets, opt-1.3b* is slightly better than opt-30b*†. Moreover, TREC-covid dataset seems to perform slightly better in opt-125m* than opt-30b*†.

This outcome could be explained because opt-30b*† is using the 8-bit optimization technique, which improves inference speed, but also decreases the performance slightly, especially in outliers. Notice that "covid"could be considered an outlier due to the lack of datasets about the topic.

Overall, we can see that increasing the number of parameters allows for an increase in

the performance of the system. We believe that this is not a big issue since questions are generated once and, once generated, they can be reused multiple times for free.

4.7.3 Does increasing the number of examples improve the performance?

In this section, we want to answer if quantity is more important than quality. Table 4.4 shows the results obtained after generating five questions for the OPT checkpoint with 125 million parameters.

Dataset \ model name	opt-125m*	opt-125m-x5*	opt-1.3b*
Average	29.10	30.02	**32.86**
MS-MARCO	18.74	20.19	**21.61**
TREC-covid	39.27	**41.69**	39.77
NFCorpus	24.62	**27.23**	26.20
NaturalQuestions	19.25	19.98	**25.49**
HotpotQA	32.35	33.88	**38.18**
FiQA	16.90	18.11	**20.76**
ArguAna	45.01	35.42	**47.74**
Tóuche-2020	10.47	11.27	**13.44**
CQAdupstack	20.66	22.84	**23.70**
Quora	72.23	74.08	**78.15**
DBpedia	22.39	23.59	**24.36**
Scidocs	10.72	**11.85**	10.83
Fever	31.46	32.83	**45.74**
Climate-fever	12.34	12.56	**13.14**
Scifact	49.80	**54.99**	52.52

Table 4.4: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether generating more questions faster using a smaller checkpoint is better than generating slower high-quality questions. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. x5 means that five questions have been generated per document.

Table 4.4 shows that generating more questions improves the overall performance of the system when we compare it to the same system but only using one question. If we compare opt-1.3b* with opt-125m-x5*, we can see that opt-1.3b* is in general better for most of the datasets.

From these experiments, we conclude that it is not worth generating more questions using a smaller checkpoint even if there is a significant improvement over generating one ques-

tion using the same checkpoint. The main reason is that training the DPR system takes 10 days instead of the usual 2. We think that the extra 8 days it takes could be used to generate higher quality questions instead.

4.7.4 Is the crop generation method better than Large Language Models?

In Chapter 3, we mentioned the crop generation method makes use of random cropping and random deletion of words of documents to create question-document pairs. Table 4.5 shows the results obtained when generating queries using the crop method.

Dataset \ model name	marco-supervised	opt-30b*†	Crop queries	Crop queries-documents
Average	**37.07**	33.31	13.08	19.32
MS-MARCO	**27.75**	22.88	5.59	10.97
TREC-covid	**58.90**	37.50	16.32	21.01
NFCorpus	26.98	**29.25**	3.50	10.30
NaturalQuestions	**31.93**	24.63	8.43	8.80
HotpotQA	**40.45**	38.52	14.01	15.53
FiQA	**22.61**	22.21	4.49	5.50
ArguAna	45.25	**47.32**	8.42	47.13
Tóuche-2020	**19.17**	14.39	4.37	3.48
CQAdupstack	**27.46**	24.64	14.51	12.95
Quora	**77.59**	75.27	71.20	78.98
DBpedia	**29.77**	25.57	8.03	14.29
Scidocs	**12.87**	11.21	2.67	3.04
Fever	**56.25**	50.46	5.12	11.52
Climate-fever	**16.89**	13.01	2.04	3.94
Scifact	**52.92**	52.35	20.06	34.00

Table 4.5: In-domain and zero-shot performances on BEIR benchmark. In this table, we show the performance of using the crop method to generate queries-document pairs. All scores denote NDCG@10. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset.

Table 4.5 shows that generating queries using the crop method provides bad results in all the datasets, not even getting close to the opt-125m* checkpoint with 29 points for the NDCG@10 metric. We notice that for the crop method, it is completely mandatory to apply their approach to both the query and the document (we are technically interested in

generating only queries) because if we apply the approach to only get the query we lose 5 points in the performance of the IR system.

Nevertheless, we need to say that the crop generation method is extremely cheap, taking less than 5 minutes to generate 8 million query-document pairs. In comparison, opt-30b*† takes 7 days using 8xA100 80GB GPUs.

4.7.5 Does domain adaption improve the performance of the systems?

With the most important questions already answered, the next thing that we want to know is whether domain adaption works in order to improve the results of a specific dataset. Table 4.6 shows the results of finetuning the Scifact and NFCorpus using different methods on top of marco-supervised.

Model name \ dataset	NFCorpus	Scifact
marco-supervised	26.98	52.92
scifact-supervised+	-	68.55
scifact-opt-30b+*†	-	54.55
scifact-crop+	-	57.35
nfcorpus-supervised+	28.30	-
nfcorpus-opt-30b+*†	33.20	-
nfcorpus-crop+	16.47	-

Table 4.6: In-domain and zero-shot performances on BEIR benchmark. In this table, we show whether domain adaption is beneficial to improve the performance of datasets from another domain. All scores denote NDCG@10. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised.

Before looking at the results, it is important to make it clear that we are not interested in the supervised setting of these new datasets, what we want to know is if a completely new unseen dataset without annotated questions can be finetuned on top of another system in order to improve its performance.

Table 4.6 shows that for the NFCorpus dataset either using the NFCorpus supervised dataset or the NFCorpus dataset generated using the OPT 30 billion parameter checkpoint increases the performance 1.32 and 6.22 points respectively. In the case of using the crop method we lose almost 10 points in performance. We think that the length of documents of NFCorpus can contribute to this bad performance using this last method.

On the other hand, we can see that Scifact dataset improves regardless of the method. In this case, the supervised method is the best providing an extra 15.63 points, followed by the crop method with an extra 4.43, and finally the OPT checkpoint with 30 billion parameters that improves only 1.63 points. We think that the difference between the results of Scifact and NFCorpus could be due to the size of the dataset, as Scifact barely contains 900 examples and NFCorpus 110,000.

In this section, we conclude that domain adaption helps to improve the performance of a given dataset. Unfortunately, there is not a method that we can say it is better than other, therefore, we think that the best option is to try the three of them and select the best one.

4.7.6 Does prompting improve the generation of questions?

The next question that we had is which method of prompting is the best for question generation. In Chapter 3.2, we mentioned that we were using the prompt used by [Bonifacio et al., 2022a]. Nevertheless, in this section we want to check whether changing the prompt has any influence in the performance. More specifically we will check the following prompts:

- Keywords: In this prompt, we generate first keywords related to the document and then we generate the question taking into consideration both, the document and the generated keywords.
- Title: The idea is the same as the keywords one, but instead of generating keywords, we generate first a title for the document and then we generate the question taking into consideration both.
- Instruct-GPT: A different way of approaching this is by asking a model to do something according to an instruction. In our case, we check whether the instruction "Create a question for this document: {}"can generate high-quality questions. We wanted to try OPT-IML but this model is still not publicly available, therefore, we will make use of Instruct-GPT [Ouyang et al., 2022].

Table 4.7 shows the results obtained when changing the prompt to generate the questions using the Scifact dataset.

Table 4.7 shows that using the keywords or title prompt allows to have a slight improvement of 0.72 and 1.24 points on the NDCG@10 metric, meaning that prompting has some importance. Instruct-GPT is the one that benefits the most by improving the performance

Model Name \ Dataset	scifact
scifact-opt-30b+*†	54.55
scifact-instruct-gpt	57.13
scifact-keywords-opt-30b+*†	55.27
scifact-title-opt-30b+*†	55.79

Table 4.7: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether changing the prompt to generate questions allows to improve the performance. All scores denote NDCG@10. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised.

by 2.58 points on the NDCG@10 metric. Nevertheless, we need to say that this might be due to Instruct-GPT being a completely different generative architecture and trained on a different dataset.

In this section, we conclude that changing the prompt is in general beneficial to improve the performance of the systems. Nevertheless, this two step prompting of generating first the title or the keywords and then the question comes with a drawback, which is that is going to take twice the time to generate a single question. This approach using small OPT checkpoints or small datasets should not be a big issue, but for OPT 30 billion with the MS MARCO dataset it is, as it would take 14 days to generate all the questions instead of 7 days.

4.7.7 Does using hard negatives allow to improve the performance of the system?

Before comparing our results with other state-of-the-art systems, the last thing that we want to check is whether hard negatives have an influence when training the system.

Table 4.8 shows the results after using hard negatives obtained using BM25.

Model Name \ Dataset	scifact
scifact-supervised+	68.55
scifact-hard-negatives-supervised+	67.67

Table 4.8: In-domain and zero-shot performances on BEIR benchmark. In this table, we check whether hard negatives allow to improve the performance of the systems trained. All scores denote NDCG@10. + indicates that the system has been finetuned on top of marco-supervised.

Table 4.8 shows that the performance difference between using hard negatives or not is minimal. We can see that the NDCG@10 metric decreases by a point meaning that in theory it is not recommended to use them.

In this section, we conclude that using hard negatives can be harmful to the performance of the system. Nevertheless, these hard negatives are obtained using BM25 which probably is not the best method to do it but is the cheapest and simplest one. We think that more thorough research should be made about hard negatives.

4.7.8 How does our system compare to other state-of-the-art systems?

In this last section, we will compare our system to other state-of-the-art systems. Unfortunately, there are not many unsupervised systems, and the ones available are trained on a different dataset. Moreover, most of the experiments can not be replicated due to not having the necessary resources available.

Table 4.9 shows the comparison table of our systems against other supervised and unsupervised models.

In table 4.9 we can differentiate two methodologies: supervised and unsupervised. The supervised ones are GenQ and marco supervised. In the case of marco supervised, it is a DPR system trained on MS MARCO dataset, while GenQ trains a T5 (base) model using MS MARCO to generate questions, then for each of the datasets creates five questions using the recently trained model and finally it trains a TAS-B bi-encoder model for each dataset (domain adaptation). Comparing these two, we can see that GenQ is better than marco supervised in all the datasets except Tóuche-2020. Unfortunately, these results are not 100% comparable due to GenQ creating a specific model for each dataset (domain adaptation). Moreover, BM25, an unsupervised method, is still better in some of the datasets.

For the unsupervised setting, we have BM25, LaPraDoR, Contriever, and our opt-30b*† system. BM25 is a lexical system, hence is not trained in any datasets. LaPraDor [Bonifacio et al., 2022b] (without Lexicon Enhanced Dense Retrieval) is trained on MS MARCO dataset. Contriever is trained on a combination of CCNET and Wikipedia. Finally, opt-30b*† is trained only on MS MARCO using questions generated by the LLM. Comparing these four systems we can see that still BM25 is the best method in most of the datasets with an average of 43.01 points, followed by Contriever with an average of 37.06, the opt-30b*† with 33.31, and finally LaPraDor with 30.21 points. Unfortunately, due to Contriever being

Dataset \ Model Name	Supervised		Unsupervised			
	marco supervised	GenQ	BM25	LaPraDoR	Contriever	opt-30b*†
Average	37.07	**42.49**	**43.01**	30.21	37.06	33.31
MS-MARCO	27.75	**40.80**	22.80	-	20.60	**22.88**
TREC-covid	58.90	**61.90**	**65.60**	22.70	27.40	37.50
NFCorpus	26.98	**31.90**	**32.50**	31.10	31.70	29.25
NaturalQuestions	31.93	**35.80**	**32.90**	18.10	25.40	24.63
HotpotQA	40.45	**53.40**	**60.30**	30.30	48.10	38.52
FiQA	22.61	**30.80**	**23.60**	20.30	24.50	22.21
ArguAna	45.25	**49.30**	31.50	45.90	37.90	**47.32**
Tóuche-2020	**19.17**	18.20	**36.70**	9.40	19.30	14.39
CQAdupstack	27.46	**34.70**	**29.90**	22.00	28.40	24.64
Quora	77.59	**83.00**	78.90	78.70	**83.50**	75.27
DBpedia	29.77	**32.80**	**31.3**0	25.00	29.20	25.57
Scidocs	12.87	**14.30**	**15.80**	13.30	14.90	11.21
Fever	56.25	**66.90**	**75.30**	36.80	68.20	50.46
Climate-fever	16.89	**17.50**	**21.30**	13.80	15.50	13.01
Scifact	52.92	**64.40**	**66.50**	55.50	64.90	52.35

Table 4.9: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare our system to other state-of-the-art systems. All scores denote NDCG@10. The best scores for supervised and unsupervised settings on a given dataset are marked in bold, in the case of the unsupervised setting the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset.

trained on a different dataset the results are not 100% comparable. Despite the fact that our system is worse, the difference in performance is not that big as Contriever is trained in a total of 1,024,000,000 examples, whereas our system is trained on only 32,000,000 examples.

When we compare the marco supervised and opt-30b*† systems, that is, the same model but trained in a supervised and unsupervised way respectively, we find that the supervised setting is better in all the datasets except NFCorpus and ArguAna. In these two datasets, we find that the unsupervised setting is slightly better by 2.27 and 2.07 points respectively. We think that the lengths of the documents of these two datasets might have an impact during human annotations, as it might be difficult to create a good representing query for a document with too many words. In general, in all the datasets, we find that the difference is not extremely big and we believe that in the future neural unsupervised methodologies can surpass the supervised setting.

4.7.9 Summary

Overall, we can conclude that unsupervised question generation for information retrieval can lead to results that are close to the state-of-the-art BM25 system. Current LLM are able to generate high quality question that are relevant to the documents, and hence we believe that neural networks could understand better the hidden underlying patterns found between documents and questions.

In this project, we need to highlight the importance of the number of parameters of the model and the use of sampling, as it allows to generate higher quality questions, hence improve the performance of the system. We can also see, that while increasing the number of generated questions improves the performance, usually spending this time in a model with more parameters but generating only one question tends to work better. Generating query-document pairs using crop generation method is extremely cheap, but also does not perform as well as using Large Language Models. Moreover, using hard negatives can be harmful to the performance of the system, though more thorough research should be done on this. Finally, if we want to obtain an extra performance we can either modify the prompt used to generate the questions or apply domain adaption on a specific dataset.

CHAPTER 5

Conclusions and future work

In this chapter, the conclusions of the project are provided and directions for future work are proposed.

5.1 Conclusions

In this project, we propose to investigate open domain QA, with a special focus on the first component of the architecture, that is, the information retrieval component. We trained several DPRs in an unsupervised manner by generating questions from the documents using Large Language Models (LLM). Currently, most LLMs provide several checkpoints with different amounts of parameters, and we used these checkpoints to generate questions, train a DPR system for each LLM checkpoint, and finally, compare if the generated questions have any influence in the performance of the systems. The developed system must be able to search for the necessary information in external sources, usually organized in text documents. For that purpose, the BEIR benchmark was used in a zero-shot manner to test the performance.

This work makes use of OPT as LLM with sampling in order to generate questions in a zero-shot manner. The checkpoints utilized in this work are the ones with 125 million parameters, 1.3 billion parameters, and 30 billion parameters. A different method to generate query-document pairs that can be used to generate is the crop method which applies random cropping and word deletion in order to create the pairs. In this project, all the

MS MARCO dataset documents are used to generate the questions necessary to train the systems.

In this work, we also consider domain adaption, that is, after training on MS MARCO dataset we finetune on another dataset on top of it in order to see if the performance can be improved for this specific dataset, prompting, which means that we use different prompts in order to generate questions, and the use of hard negatives to train the dense passage retrieval system.

Overall, we can conclude that unsupervised question generation for information retrieval can lead to results that are close to the state-of-the-art BM25 system. We need to highlight the importance of the number of parameters of the model and the use of sampling, as it allows to generate higher quality questions, hence improve the performance of the system. We can also see, that while increasing the number of generated questions improves the performance, usually spending this time in a model with more parameters but generating only one question tends to work better. Generating query-document pairs using crop generation method is extremely cheap, but also does not perform as well as using Large Language Models. Moreover, using hard negatives can be harmful to the performance of the system, though more thorough research should be done on this. Finally, if we want to obtain an extra performance we can either modify the prompt used to generate the questions or apply domain adaption on a specific dataset.

5.2 Future work

This project covers the basic scope of how useful can be generating questions for information retrieval systems. The findings of this project can be continued in the following research lines:

- OPT-IML: OPT-IML is an instruction based transformer similar to Instruct-GPT. In this project, we focus on the latter due to OPT-IML not being publicly available. We think that OPT-IML could provide more comparable results in order to decide whether instructions as prompts are helpful or not.

- Generate questions without using LLM: Currently in order to use LLM we are required to use expensive GPUs. Crop generation method allows to generate query-document pairs extremely cheap using a CPU and it is almost instantly for large da-

tasets. We think that there might be some better solutions compared to their greedy method that would allow to generate higher quality pairs.

- Hard negatives: Another important aspect that we should research is whether techniques to mine hard negatives can improve the performance of the system. In this thesis, we concluded that BM25 can be harmful when used with the Scifact dataset as it was reducing the performance by 1 point. We believe that building an alternative to BM25 could be helpful in order to improve the performance.

- Integration of the information retriever into the overall open domain QA system: Today, models such as FiD [Izacard and Grave, 2020] already use neural models to retrieve relevant documents, but these models do not learn during training for QA. That is, they are frozen and thus depend entirely on the task on which they have been trained. We propose to take the information retriever of this project and keep adjusting it during the QA phase. One of the major problems is that during this training phase, we will only know if the overall system has been able to answer the question well, but we will not know if the hit or miss is due to the documents retrieved by the retriever. Therefore, we propose to introduce reinforcement learning techniques, with the hit or miss being the reinforcement signal that allows us to better adjust the information retriever.

Appendix

CHAPTER A

Appendix

A.1 Experiment results for Recall@100

In this section, we provide the results of the experiments for the Recall@100 metric.

A.1.1 Is sampling useful to generate questions?

Dataset \ model name	opt-125m	opt-1.3b	opt-125m*	opt-1.3b*
Average	48.15	51.72	52.59	**55.24**
MS-MARCO	61.28	66.08	64.60	**70.18**
TREC-covid	7.38	5.40	**7.44**	5.81
NFCorpus	23.36	22.06	**24.15**	23.80
NaturalQuestions	60.64	70.51	66.70	**76.43**
HotpotQA	40.80	51.09	50.22	**55.75**
FiQA	46.16	47.24	51.17	**53.42**
ArguAna	94.88	95.24	96.23	**97.37**
Tóuche-2020	34.08	35.48	36.12	**37.38**
CQAdupstack	47.93	50.07	50.70	**54.40**
Quora	95.94	96.41	95.81	**97.40**
DBpedia	26.98	32.35	35.31	**37.13**
Scidocs	24.81	23.80	**28.08**	27.51
Fever	58.90	78.28	73.43	**84.67**
Climate-fever	30.38	34.30	35.69	**37.61**
Scifact	81.92	81.87	**85.26**	84.70

Table A.1: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether sampling is helpful to improve the performance of the systems. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9.

A.1.2 Does increasing the number of parameters of the model for generating questions improve the performance of the systems?

Dataset \ model name	opt-125m*	opt-1.3b*	opt-30b*†
Average	52.59	<u>55.24</u>	**56.52**
MS-MARCO	64.60	<u>70.18</u>	**72.66**
TREC-covid	**7.44**	5.81	<u>6.17</u>
NFCorpus	<u>24.15</u>	23.80	**25.95**
NaturalQuestions	66.70	<u>76.43</u>	**77.54**
HotpotQA	50.22	<u>55.75</u>	**56.33**
FiQA	51.17	<u>53.42</u>	**54.44**
ArguAna	96.23	<u>97.37</u>	**97.58**
Tóuche-2020	36.12	<u>37.38</u>	**39.57**
CQAdupstack	50.70	<u>54.40</u>	**56.00**
Quora	95.81	**97.40**	<u>96.90</u>
DBpedia	35.31	<u>37.13</u>	**38.81**
Scidocs	<u>28.08</u>	27.51	**28.54**
Fever	73.43	<u>84.67</u>	**88.15**
Climate-fever	35.69	<u>37.61</u>	**38.79**
Scifact	<u>85.26</u>	84.70	**86.53**

Table A.2: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether increasing the number of parameters is helpful to improve the performance of the systems. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset.

A.1.3 Does increasing the number of examples improve the performance?

Dataset \ model name	opt-125m*	opt-125m-x5*	opt-1.3b*
Average	52.59	53.67	**55.24**
MS-MARCO	64.60	66.71	**70.18**
TREC-covid	7.44	**8.10**	5.81
NFCorpus	24.15	**26.39**	23.80
NaturalQuestions	66.70	70.00	**76.43**
HotpotQA	50.22	51.33	**55.75**
FiQA	51.17	52.07	**53.42**
ArguAna	96.23	92.32	**97.37**
Tóuche-2020	36.12	37.05	**37.38**
CQAdupstack	50.70	53.31	**54.40**
Quora	95.81	96.10	**97.40**
DBpedia	35.31	36.30	**37.13**
Scidocs	28.08	**28.98**	27.51
Fever	73.43	75.47	**84.67**
Climate-fever	35.69	36.08	**37.61**
Scifact	85.26	**87.87**	84.70

Table A.3: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether generating more questions faster using a smaller checkpoint is better than generating slower high-quality questions. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. x5 means that five questions have been generated per document.

A.1.4 Is the crop generation method better than Large Language Models?

Dataset \ model name	marco-supervised	opt-30b*†	Crop queries	Crop queries-documents
Average	**58.08**	56.52	25.53	35.68
MS-MARCO	**78.27**	72.66	22.59	42.61
TREC-covid	**8.89**	6.17	2.05	2.15
NFCorpus	25.20	**25.95**	9.45	16.29
NaturalQuestions	**80.76**	77.54	31.16	35.15
HotpotQA	**58.75**	56.33	30.10	32.83
FiQA	53.54	**54.44**	17.06	17.67
ArguAna	97.23	**97.58**	28.24	95.88
Tóuche-2020	**39.82**	39.57	8.06	7.40
CQAdupstack	**58.27**	56.00	34.79	36.12
Quora	**97.48**	96.90	93.71	97.29
DBpedia	**41.26**	38.81	14.15	23.45
Scidocs	**32.35**	28.54	11.83	15.09
Fever	**88.37**	88.15	13.28	29.19
Climate-fever	**44.56**	38.79	8.82	17.24
Scifact	**86.63**	86.53	54.70	73.84

Table A.4: In-domain and zero-shot performances on BEIR benchmark. In this table, we show the performance of using the crop method to generate queries-document pairs. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset.

A.1.5 Does domain adaption improve the performance of the systems?

Model name \ dataset	NFCorpus	Scifact
marco-supervised	25.20	86.63
scifact-supervised+	-	94.37
scifact-opt-30b+*†	-	88.47
scifact-crop+	-	89.53
nfcorpus-supervised+	33.57	-
nfcorpus-opt-30b+*†	29.05	-
nfcorpus-crop+	18.51	-

Table A.5: In-domain and zero-shot performances on BEIR benchmark.In this table, we show whether domain adaption is beneficial to improve the performance of datasets from another domain. All scores denote Recall@100. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised.

A.1.6 Does prompting improve the generation of questions?

Model Name \ Dataset	scifact
scifact-opt-30b+*†	88.47
scifact-instruct-gpt	88.2
scifact-keywords-opt-30b+*†	88.467
scifact-title-opt-30b+*†	88.467

Table A.6: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare whether changing the prompt to generate questions allows to improve the performance. All scores denote Recall@100. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset. + indicates that the system has been finetuned on top of marco-supervised.

A.1.7 Does using hard negatives allow to improve the performance of the system?

Model Name \ Dataset	scifact
scifact-supervised+	94.37
scifact-hard-negatives-supervised+	93.77

Table A.7: In-domain and zero-shot performances on BEIR benchmark. In this table, we check whether hard negatives allow to improve the performance of the systems trained. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. + indicates that the system has been finetuned on top of marco-supervised.

A.1.8 How does our system compare to other state-of-the-art systems?

Dataset \ Model Name	Contriever	marco supervised	opt-30b*†
Average	**59.61**	58.08	56.52
MS-MARCO	67.20	**78.27**	72.66
TREC-covid	**17.20**	8.89	6.17
NFCorpus	**29.40**	25.20	25.95
NaturalQuestions	77.10	**80.76**	77.54
HotpotQA	**70.40**	58.75	56.33
FiQA	**56.20**	53.54	54.44
ArguAna	90.10	97.23	**97.58**
Tóuche-2020	22.50	**39.82**	39.57
CQAdupstack	**61.40**	58.27	56.00
Quora	**98.70**	97.48	96.90
DBpedia	**45.30**	41.26	38.81
Scidocs	**36.00**	32.35	28.54
Fever	**93.60**	88.37	88.15
Climate-fever	44.10	**44.56**	38.79
Scifact	**92.60**	86.63	86.53

Table A.8: In-domain and zero-shot performances on BEIR benchmark. In this table, we compare our system to other state-of-the-art systems. All scores denote Recall@100. The best score on a given dataset is marked in bold, and the second best is underlined. The average corresponds to the average of all the datasets except MS MARCO. * indicates that the dataset has been generated using sampling and a top p of 0.9. † indicates that 8-bit optimization techniques have been used to generate the dataset.

A.2 Project objectives report

In this section, the objectives of the project are defined. The project definition covers an overall description of the project, the concrete goals of the project, and the planning and the methodology to achieve those goals. Finally, a list of identified risks that can compromise the project is given.

A.2.1 Project description and goals

The main goal of the project is to improve current language models using external memory. The solutions for such an ambitious goal will be tested first in a more specific environment: improving current open domain QA systems, where the documents will act as external memory. Therefore the main objective is decomposed into the following more specific objectives:

- **Self-supervised learning models for information retrieval**: Given the scalability problems of supervised models derived from the need for labeled datasets, we propose the development of new self-supervised learning methods. The main idea is to automatically generate answerable and unanswerable questions from paragraphs extracted from text documents using Large Language Models (LLMs). Thus, positive and negative examples can be generated for each document very cheaply. We will investigate how to use contrastive learning models in these scenarios and their complementarity with current methods.

- **Development of language models with external memory**: Some researchers are working on increasing the scalability of systems such as FiD so that they can process more than 100 documents when answering a question. In this project, we propose to go in another direction, arguing that so many documents are rarely needed to answer user queries on a day-to-day basis. Therefore, we will analyze how to make current architectures more flexible so that the relationship between the information retriever and the reader is much more flexible, with the ultimate goal that the retriever becomes the access point of a vast memory (all indexed documents) that the reader can use in flexibly and efficiently. In our vision, the reader will be able to make several requests to the retriever, depending on the query and the documents that the retriever brings.

To accomplish the previous objectives the work can be divided into smaller tasks:

- Study the literature and understand the problem.
- Evaluate state-of-the-art implementations.
- Define and evaluate the dataset.
- Evaluate different ways to generate questions.
- Define the architecture.
- Define the metrics.
- Systems experiments and comparison.

The previous tasks will give a deep understating of how the research world works for real-life problems.

A.2.2 Project planning

WBS diagram

A Work Breakdown Structure (WBS) is used to outline the work that needs to be done for this project and is shown in Figure A.1. The project time estimates for each work unit can be seen in Table A.9.

Work units

In this section, a summary of each work unit and the estimation of the time that will be spent on each one are going to be given. Because of the difficulty of this project, some of these tasks may require more time than expected.

Planning

The work related to organizing the project that includes: what the project is about, what are the goals, what are the tasks and when they need to be delivered, what milestones have to be achieved and the deadline for those milestones, and what are the risks of this project. The deliverable of this task is this report.

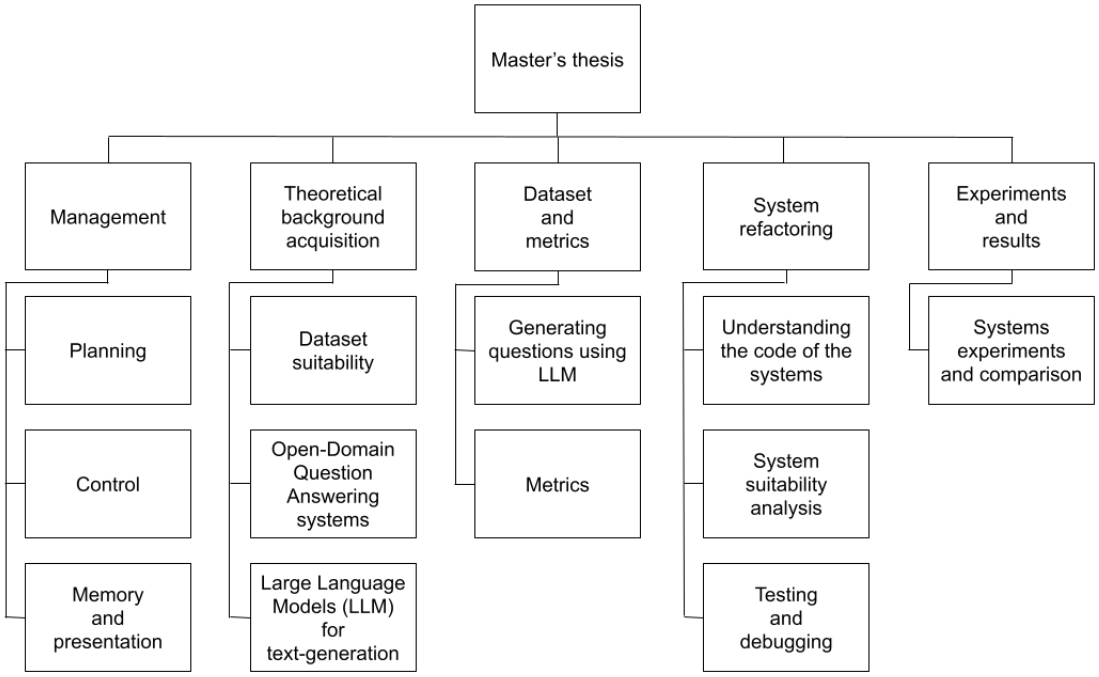

Figure A.1: Work Breakdown Structure of the project.

Control

Keeping the project focused, achieving the goals proposed according to the established schedule, and solving any problem that could affect the project. This work is performed in control meetings every week.

Memory and presentation

Writing the memory of the project with the work that has been made and the public defense of it which will require to prepare materials for the presentation.

Datasets suitability

Search and study of datasets suitable for the project.

Open-domain question answering systems

Investigate open-domain question answering systems. This task requires to read papers about State-Of-The-Art (SOTA) systems in order to understand what has been done.

Large Language Models (LLM) for text-generation

Investigate Large Language Models (LLM) for text-generation. This task requires to read

papers about text-generation, in order to choose the most appropriate system for this project.

Generating questions using LLM

Using the LLM that we think is the most appropriate for generating questions, use it on the dataset suitable for the project.

Metrics

Understand and select the most appropriate metrics for the information retrieval task. This work unit includes the selection of the benchmark to use for testing the systems.

Understanding the code of the systems

Understanding the code of state-of-the-art systems. This will include understanding how Dense Parse Retrievals (DPRs) and Contriever systems are implemented and the potential improvements that can be applied to the systems in order to improve their performance.

System suitability analysis

Analyze if the systems are suitable for our project requirements and what changes need to be done in order to fit our ideas.

Testing and debugging

Testing and debugging of DPR and Contriever systems in order to ensure that they work properly.

Systems experiments and comparison

Perform experiments using the current SOTA systems and the dataset generated. Finally, compare the results using the BEIR benchmark [Thakur et al., 2021].

Gantt chart

A Gantt chart is used to illustrate the project schedule. This chart lists the tasks to be performed on the vertical axis and the time intervals on the horizontal axis. The Gantt chart is shown in Figure A.2.

Milestones

Table A.10 shows the deadline dates for the deliverables.

Work-unit	**Time estimate (hours)**
Management	**170**
Planning	25
Control	45
Memory and presentation	100
Theoretical background acquisition	**230**
Dataset suitability	30
Open-Domain Question Answering systems	100
Large Language Models (LLM) for text-generation	100
Dataset and metrics	**130**
Generating questions using LLM	100
Metrics	30
System refactoring	**1700**
Understanding the code of the systems	80
System suitability analysis	40
Testing and debugging	50
Experiments and results	**50**
Systems experiments and comparison	50
Total	**750**

Table A.9: Time estimates for each work unit.

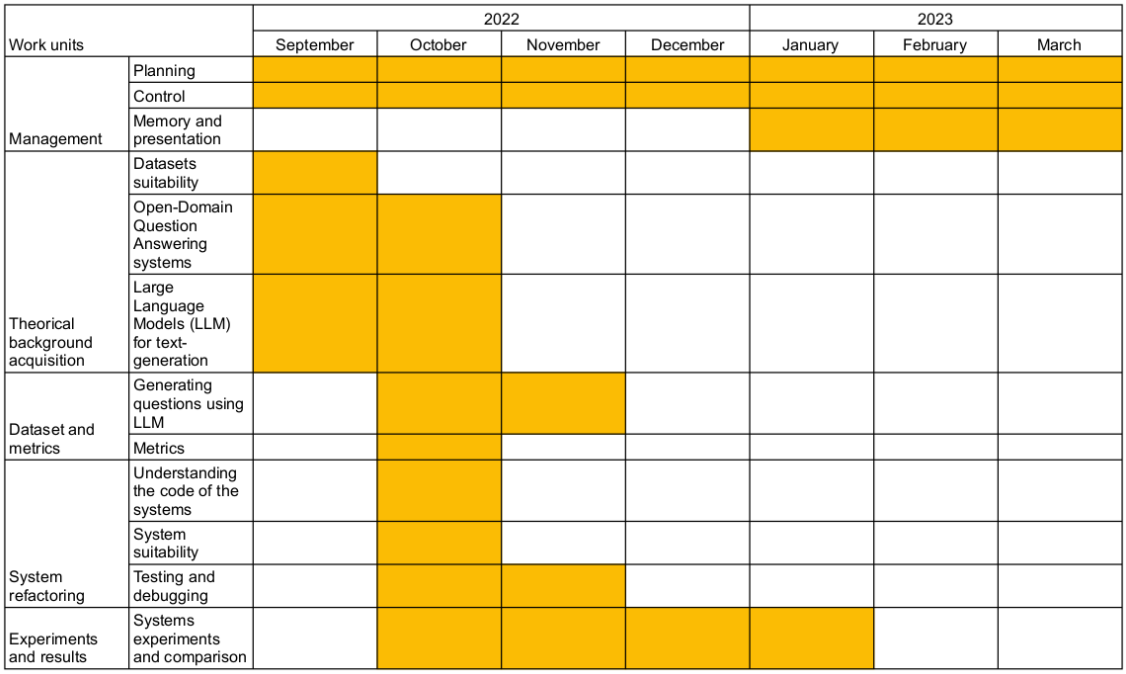

Figure A.2: Gantt chart of the project.

Deliverable	Date
Implementation and experiments	31/12/2022
Memory	31/01/2023
Presentation	01/03/2023 - 03/03/2023

Table A.10: Deliverables and their deadlines.

A.2.3 Methodology

This master's thesis is carried out as a project of the IXA natural language processing research group. The student receives support from two IXA supervisors (Eneko Agirre Bengoa and Gorka Azkune Galparsoro) along with two assistants (Jon Ander Campos and Aitor Soroa). The student is allowed to use hardware resources from the IXA research group in the form of server nodes with CPU and GPU capabilities.

Meetings

Regular meetings are arranged in a fixed slot every week with the two IXA instructors. These meetings are held, either in the faculty or remotely using Google Hangouts. These meetings' objective is to control the progress of the project and talk about problems and their possible solutions. Meetings outside the fixed time are scheduled to attend to specific doubts.

Work place

The student will work from home in a relaxed environment and good internet connection.

A.2.4 Risks

Given the size and scope of the project, there may be some uncertainties that will put the project at risk. The following list describes some of the risks identified:

- **COVID-19 situation:** The uncertainty of the current situation may incur some unexpected situations, such as the impossibility of holding the meetings in the faculty and the student or instructors getting infected and therefore, the impossibility to work and hold meetings for a while.

- **Compute capabilities:** Deep learning requires high amounts of computing and memory resources. The student is given the possibility to use the IXA research group's shared resources in case the online free resources are not enough. These shared resources may be in high demand for a given period, therefore, it is difficult to know when they will be free.

Acknowledgments

We thank the IXA research members Jon Ander Campos and Aitor soroa for their helpful discussions and insightful feedback.

Bibliography

[Andreas et al., 2016] Andreas, J., Rohrbach, M., Darrell, T., and Klein, D. (2016). Learning to compose neural networks for question answering. *CoRR*.

[Black et al., 2022] Black, S., Biderman, S., Hallahan, E., Anthony, Q., Gao, L., Golding, L., He, H., Leahy, C., McDonell, K., Phang, J., Pieler, M., Prashanth, U. S., Purohit, S., Reynolds, L., Tow, J., Wang, B., and Weinbach, S. (2022). Gpt-neox-20b: An open-source autoregressive language model.

[Black et al., 2021] Black, S., Leo, G., Wang, P., Leahy, C., and Biderman, S. (2021). GPT-Neo: Large Scale Autoregressive Language Modeling with Mesh-Tensorflow. If you use this software, please cite it using these metadata.

[Bonifacio et al., 2022a] Bonifacio, L., Abonizio, H., Fadaee, M., and Nogueira, R. (2022a). Inpars: Data augmentation for information retrieval using large language models.

[Bonifacio et al., 2022b] Bonifacio, L., Abonizio, H., Fadaee, M., and Nogueira, R. (2022b). Inpars: Data augmentation for information retrieval using large language models.

[Boteva et al., 2016] Boteva, V., Gholipour, D., Sokolov, A., and Riezler, S. (2016). A full-text learning to rank dataset for medical information retrieval. In Ferro, N., Crestani, F., Moens, M.-F., Mothe, J., Silvestri, F., DiÑunzio, G. M., Hauff, C., and Silvello, G., editors, *Advances in Information Retrieval*, pages 716–722, Cham. Springer International Publishing.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter,

C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language models are few-shot learners. *CoRR*, abs/2005.14165.

[Cao et al., 2020] Cao, N. D., Izacard, G., Riedel, S., and Petroni, F. (2020). Autoregressive entity retrieval. *CoRR*, abs/2010.00904.

[Cho et al., 2014] Cho, K., van Merrienboer, B., Bahdanau, D., and Bengio, Y. (2014). On the properties of neural machine translation: Encoder-decoder approaches. *CoRR*.

[Dai et al., 2022] Dai, Z., Zhao, V. Y., Ma, J., Luan, Y., Ni, J., Lu, J., Bakalov, A., Guu, K., Hall, K. B., and Chang, M.-W. (2022). Promptagator: Few-shot dense retrieval from 8 examples.

[Dettmers et al., 2022] Dettmers, T., Lewis, M., Belkada, Y., and Zettlemoyer, L. (2022). Llm.int8(): 8-bit matrix multiplication for transformers at scale.

[Formal et al., 2021] Formal, T., Lassance, C., Piwowarski, B., and Clinchant, S. (2021). SPLADE v2: Sparse lexical and expansion model for information retrieval. *CoRR*, abs/2109.10086.

[Goodfellow et al., 2016] Goodfellow, I., Bengio, Y., and Courville, A. (2016). *Deep Learning*. MIT Press.

[He et al., 2019] He, K., Fan, H., Wu, Y., Xie, S., and Girshick, R. B. (2019). Momentum contrast for unsupervised visual representation learning. *CoRR*, abs/1911.05722.

[Hochreiter and Schmidhuber, 1997] Hochreiter, S. and Schmidhuber, J. (1997). Long short-term memory. *Neural Computation*.

[Izacard et al., 2021] Izacard, G., Caron, M., Hosseini, L., Riedel, S., Bojanowski, P., Joulin, A., and Grave, E. (2021). Towards unsupervised dense information retrieval with contrastive learning. *CoRR*, abs/2112.09118.

[Izacard and Grave, 2020] Izacard, G. and Grave, E. (2020). Leveraging passage retrieval with generative models for open domain question answering. *CoRR*, abs/2007.01282.

[Józefowicz et al., 2016] Józefowicz, R., Vinyals, O., Schuster, M., Shazeer, N., and Wu, Y. (2016). Exploring the limits of language modeling. *CoRR*.

[Karpukhin et al., 2020] Karpukhin, V., Oguz, B., Min, S., Wu, L., Edunov, S., Chen, D., and Yih, W. (2020). Dense passage retrieval for open-domain question answering. *CoRR*, abs/2004.04906.

[Lazaridou et al., 2022] Lazaridou, A., Gribovskaya, E., Stokowiec, W., and Grigorev, N. (2022). Internet-augmented language models through few-shot prompting for open-domain question answering.

[Lesota et al., 2021] Lesota, O., Rekabsaz, N., Cohen, D., Grasserbauer, K. A., Eickhoff, C., and Schedl, M. (2021). A modern perspective on query likelihood with deep generative retrieval models. *CoRR*, abs/2106.13618.

[Mahmood and Len, 2017] Mahmood, Y.-A. and Len, H. (2017). Text summarization using unsupervised deep learning. *Expert Systems with Applications*.

[Nguyen et al., 2016] Nguyen, T., Rosenberg, M., Song, X., Gao, J., Tiwary, S., Majumder, R., and Deng, L. (2016). MS MARCO: A human generated machine reading comprehension dataset. *CoRR*, abs/1611.09268.

[Nogueira et al., 2019] Nogueira, R. F., Yang, W., Lin, J., and Cho, K. (2019). Document expansion by query prediction. *CoRR*, abs/1904.08375.

[Ouyang et al., 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback.

[Robbins, 2007] Robbins, H. (2007). A stochastic approximation method. *Annals of Mathematical Statistics*.

[Robertson and Zaragoza, 2009] Robertson, S. and Zaragoza, H. (2009). The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

[Rumelhart et al., 1986] Rumelhart, D. E., Hinton, G. E., and Williams, R. J. (1986). Learning representations by back-propagating errors. *Nature*.

[Scao et al., 2022] Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., Gallé, M., Tow, J., Rush, A. M., Biderman, S.,

Webson, A., Ammanamanchi, P. S., Wang, T., Sagot, B., Muennighoff, N., del Moral, A. V., Ruwase, O., Bawden, R., Bekman, S., McMillan-Major, A., Beltagy, I., Nguyen, H., Saulnier, L., Tan, S., Suarez, P. O., Sanh, V., Laurençon, H., Jernite, Y., Launay, J., Mitchell, M., Raffel, C., Gokaslan, A., Simhi, A., Soroa, A., Aji, A. F., Alfassy, A., Rogers, A., Nitzav, A. K., Xu, C., Mou, C., Emezue, C., Klamm, C., Leong, C., van Strien, D., Adelani, D. I., Radev, D., Ponferrada, E. G., Levkovizh, E., Kim, E., Natan, E. B., De Toni, F., Dupont, G., Kruszewski, G., Pistilli, G., Elsahar, H., Benyamina, H., Tran, H., Yu, I., Abdulmumin, I., Johnson, I., Gonzalez-Dios, I., de la Rosa, J., Chim, J., Dodge, J., Zhu, J., Chang, J., Frohberg, J., Tobing, J., Bhattacharjee, J., Almubarak, K., Chen, K., Lo, K., Von Werra, L., Weber, L., Phan, L., allal, L. B., Tanguy, L., Dey, M., Muñoz, M. R., Masoud, M., Grandury, M., Šaško, M., Huang, M., Coavoux, M., Singh, M., Jiang, M. T.-J., Vu, M. C., Jauhar, M. A., Ghaleb, M., Subramani, N., Kassner, N., Khamis, N., Nguyen, O., Espejel, O., de Gibert, O., Villegas, P., Henderson, P., Colombo, P., Amuok, P., Lhoest, Q., Harliman, R., Bommasani, R., López, R. L., Ribeiro, R., Osei, S., Pyysalo, S., Nagel, S., Bose, S., Muhammad, S. H., Sharma, S., Longpre, S., Nikpoor, S., Silberberg, S., Pai, S., Zink, S., Torrent, T. T., Schick, T., Thrush, T., Danchev, V., Nikoulina, V., Laippala, V., Lepercq, V., Prabhu, V., Alyafeai, Z., Talat, Z., Raja, A., Heinzerling, B., Si, C., Salesky, E., Mielke, S. J., Lee, W. Y., Sharma, A., Santilli, A., Chaffin, A., Stiegler, A., Datta, D., Szczechla, E., Chhablani, G., Wang, H., Pandey, H., Strobelt, H., Fries, J. A., Rozen, J., Gao, L., Sutawika, L., Bari, M. S., Al-shaibani, M. S., Manica, M., Nayak, N., Teehan, R., Albanie, S., Shen, S., Ben-David, S., Bach, S. H., Kim, T., Bers, T., Fevry, T., Neeraj, T., Thakker, U., Raunak, V., Tang, X., Yong, Z.-X., Sun, Z., Brody, S., Uri, Y., Tojarieh, H., Roberts, A., Chung, H. W., Tae, J., Phang, J., Press, O., Li, C., Narayanan, D., Bourfoune, H., Casper, J., Rasley, J., Ryabinin, M., Mishra, M., Zhang, M., Shoeybi, M., Peyrounette, M., Patry, N., Tazi, N., Sanseviero, O., von Platen, P., Cornette, P., Lavallée, P. F., Lacroix, R., Rajbhandari, S., Gandhi, S., Smith, S., Requena, S., Patil, S., Dettmers, T., Baruwa, A., Singh, A., Cheveleva, A., Ligozat, A.-L., Subramonian, A., Névéol, A., Lovering, C., Garrette, D., Tunuguntla, D., Reiter, E., Taktasheva, E., Voloshina, E., Bogdanov, E., Winata, G. I., Schoelkopf, H., Kalo, J.-C., Novikova, J., Forde, J. Z., Clive, J., Kasai, J., Kawamura, K., Hazan, L., Carpuat, M., Clinciu, M., Kim, N., Cheng, N., Serikov, O., Antverg, O., van der Wal, O., Zhang, R., Zhang, R., Gehrmann, S., Pais, S., Shavrina, T., Scialom, T., Yun, T., Limisiewicz, T., Rieser, V., Protasov, V., Mikhailov, V., Pruksachatkun, Y., Belinkov, Y., Bamberger, Z., Kasner, Z., Rueda, A., Pestana, A., Feizpour, A., Khan, A., Faranak, A., Santos, A., Hevia, A., Unldreaj, A., Aghagol, A., Abdollahi, A., Tammour, A., HajiHosseini, A., Behroozi, B., Ajibade, B.,

Saxena, B., Ferrandis, C. M., Contractor, D., Lansky, D., David, D., Kiela, D., Nguyen, D. A., Tan, E., Baylor, E., Ozoani, E., Mirza, F., Ononiwu, F., Rezanejad, H., Jones, H., Bhattacharya, I., Solaiman, I., Sedenko, I., Nejadgholi, I., Passmore, J., Seltzer, J., Sanz, J. B., Fort, K., Dutra, L., Samagaio, M., Elbadri, M., Mieskes, M., Gerchick, M., Akinlolu, M., McKenna, M., Qiu, M., Ghauri, M., Burynok, M., Abrar, N., Rajani, N., Elkott, N., Fahmy, N., Samuel, O., An, R., Kromann, R., Hao, R., Alizadeh, S., Shubber, S., Wang, S., Roy, S., Viguier, S., Le, T., Oyebade, T., Le, T., Yang, Y., Nguyen, Z., Kashyap, A. R., Palasciano, A., Callahan, A., Shukla, A., Miranda-Escalada, A., Singh, A., Beilharz, B., Wang, B., Brito, C., Zhou, C., Jain, C., Xu, C., Fourrier, C., Periñán, D. L., Molano, D., Yu, D., Manjavacas, E., Barth, F., Fuhrimann, F., Altay, G., Bayrak, G., Burns, G., Vrabec, H. U., Bello, I., Dash, I., Kang, J., Giorgi, J., Golde, J., Posada, J. D., Sivaraman, K. R., Bulchandani, L., Liu, L., Shinzato, L., de Bykhovetz, M. H., Takeuchi, M., Pàmies, M., Castillo, M. A., Nezhurina, M., Sänger, M., Samwald, M., Cullan, M., Weinberg, M., De Wolf, M., Mihaljcic, M., Liu, M., Freidank, M., Kang, M., Seelam, N., Dahlberg, N., Broad, N. M., Muellner, N., Fung, P., Haller, P., Chandrasekhar, R., Eisenberg, R., Martin, R., Canalli, R., Su, R., Su, R., Cahyawijaya, S., Garda, S., Deshmukh, S. S., Mishra, S., Kiblawi, S., Ott, S., Sang-aroonsiri, S., Kumar, S., Schweter, S., Bharati, S., Laud, T., Gigant, T., Kainuma, T., Kusa, W., Labrak, Y., Bajaj, Y. S., Venkatraman, Y., Xu, Y., Xu, Y., Xu, Y., Tan, Z., Xie, Z., Ye, Z., Bras, M., Belkada, Y., and Wolf, T. (2022). Bloom: A 176b-parameter open-access multilingual language model.

[Shen et al., 2017] Shen, Y., Huang, P.-S., Gao, J., and Chen, W. (2017). *ReasoNet: Learning to Stop Reading in Machine Comprehension*. Association for Computing Machinery.

[Thakur et al., 2021] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. (2021). BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models. *CoRR*, abs/2104.08663.

[Vaswani et al., 2017] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.Ñ., Kaiser, L., and Polosukhin, I. (2017). Attention is all you need. *CoRR*, abs/1706.03762.

[Wadden et al., 2020] Wadden, D., Lo, K., Wang, L. L., Lin, S., van Zuylen, M., Cohan, A., and Hajishirzi, H. (2020). Fact or fiction: Verifying scientific claims. *CoRR*, abs/2004.14974.

[Wang et al., 2021] Wang, T., Brovman, Y. M., and Madhvanath, S. (2021). Personalized embedding-based e-commerce recommendations at ebay. *CoRR*, abs/2102.06156.

[Wei et al., 2021] Wei, J., Bosma, M., Zhao, V. Y., Guu, K., Yu, A. W., Lester, B., Du, N., Dai, A. M., and Le, Q. V. (2021). Finetuned language models are zero-shot learners. *CoRR*, abs/2109.01652.

[Wu et al., 2016] Wu, Y., Schuster, M., Chen, Z., Le, Q. V., Norouzi, M., Macherey, W., Krikun, M., Cao, Y., Gao, Q., Macherey, K., Klingner, J., Shah, A., Johnson, M., Liu, X., Kaiser, L., Gouws, S., Kato, Y., Kudo, T., Kazawa, H., Stevens, K., Kurian, G., Patil, N., Wang, W., Young, C., Smith, J., Riesa, J., Rudnick, A., Vinyals, O., Corrado, G., Hughes, M., and Dean, J. (2016). Google's neural machine translation system: Bridging the gap between human and machine translation. *CoRR*.

[Zhang et al., 2022] Zhang, S., Roller, S., Goyal, N., Artetxe, M., Chen, M., Chen, S., Dewan, C., Diab, M., Li, X., Lin, X. V., Mihaylov, T., Ott, M., Shleifer, S., Shuster, K., Simig, D., Koura, P. S., Sridhar, A., Wang, T., and Zettlemoyer, L. (2022). Opt: Open pre-trained transformer language models.