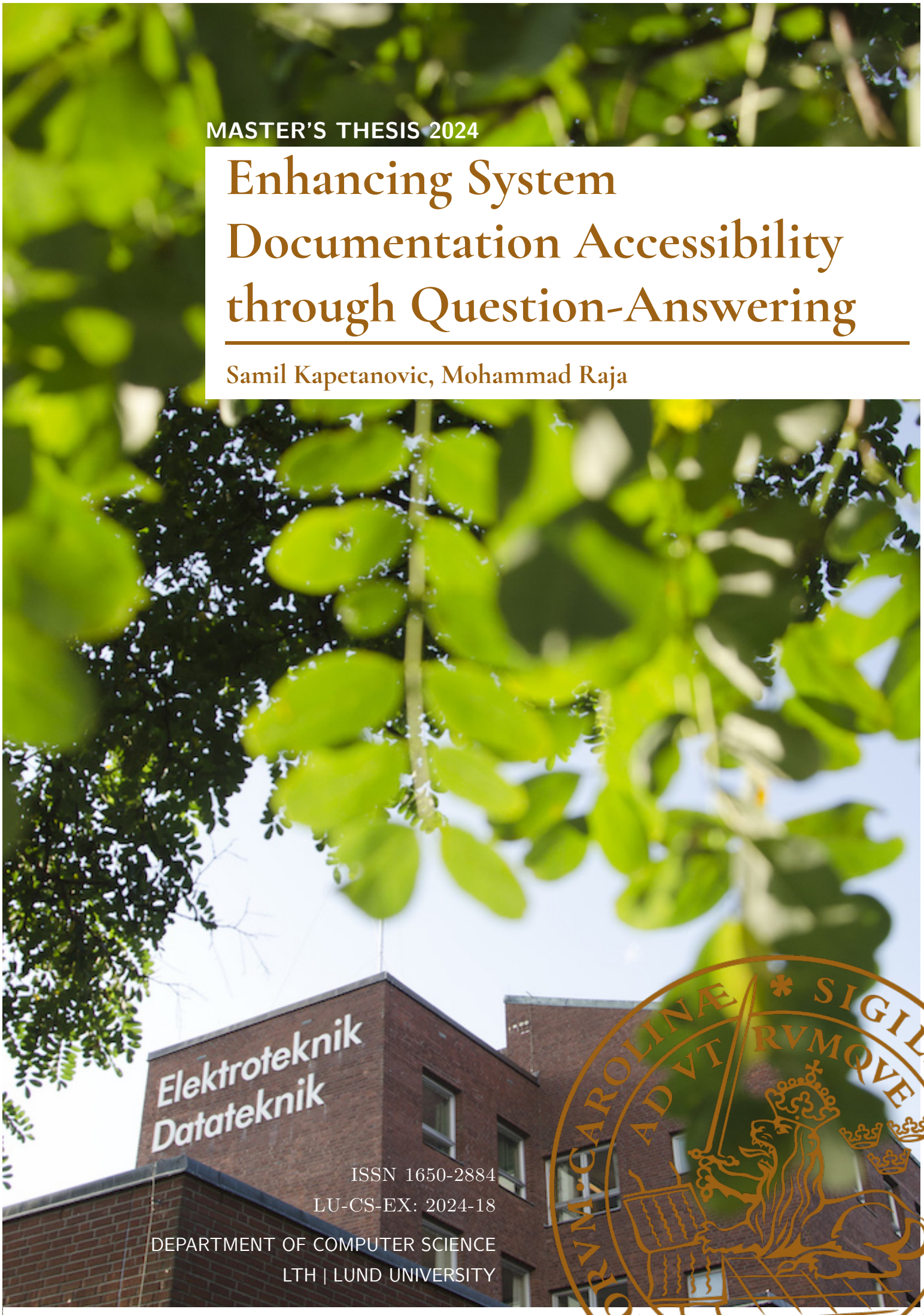


MASTER'S THESIS 2024

Enhancing System Documentation Accessibility through Question-Answering

Samil Kapetanovic, Mohammad Raja

ISSN 1650-2884
LU-CS-EX: 2024-18

DEPARTMENT OF COMPUTER SCIENCE
LTH | LUND UNIVERSITY

EXAMENSARBETE
Datavetenskap

LU-CS-EX: 2024-18

Enhancing System Documentation Accessibility through Question-Answering

Att förbättra tillgängligheten för systemdokumentation genom frågebesvarande system

Samil Kapetanovic, Mohammad Raja

Enhancing System Documentation Accessibility through Question-Answering

Samil Kapetanovic
sa8266ka-s@student.lu.se

Mohammad Raja
mo4524ra-s@student.lu.se

May 21, 2024

Master's thesis work carried out at Consafe Logistics AB.

Supervisors: Marcus Klang, marcus.klang@cs.lth.se
Eltayeb Bayomi, Eltayeb.Bayomi@consafelogistics.com

Examiner: Jacek Malec, Jacek.Malec@cs.lth.se

Abstract

In recent years, the challenge of quickly finding answers to questions has increased, with individuals often spending a lot of time searching for relevant information. This has led to an increasing interest in developing efficient systems to enhance accessibility. This thesis investigates the usage of open-retrieval question-answering systems and their potential to enhance the accessibility of system documentation. Through a comprehensive exploration, we investigate various open-retrieval question-answering configurations, including sparse and dense retrievers, extractive and generative readers, and additional experiments across three different datasets.

We present the system performances on both benchmark datasets and a private dataset created in collaboration with Consafe Logistics AB. These systems exhibit strong performance across the benchmark datasets. With ConsafeQA, we developed a system configuration that achieves an F1 score of up to 79.68 and another configuration that reaches an F1 score of up to 81.96 on benchmark datasets. Our results also indicate that different combinations of components and hyperparameters excel in different contexts. Therefore, no fixed pipeline consistently achieves optimal performance on all datasets, emphasizing the significance of tuning the system components and parameters according to the domain.

Keywords: Question-Answering, Information Retrieval, Transformers, Large Language Models, Natural Language Processing, Technical system documentation

Acknowledgements

We extend our gratitude to Consafe Logistics AB in Lund for providing us the opportunity to conduct our thesis work at their company and for offering valuable guidance throughout the process. Our sincere thanks go to all Consafe Logistics employees who assisted us in any way during our work and to everyone we had the pleasure of meeting. We want to give special appreciation to Eltayeb Bayomi, Bernt Hylén, and Andreas Anyuru for generously sharing their time and insights during this project. Additionally, we express our deep thanks to our supervisor, Marcus Klang, from the Department of Computer Science at the Faculty of Engineering at Lund University, for his invaluable support throughout our master's thesis.

Contents

Chapter 1
Introduction

Natural Language Processing (NLP) is a field focused on studying the capabilities of computers to understand and interpret human language. This field tackles several areas such as generated text summarization, sentence similarity, sentiment classifiers, and what we study in this paper question-answering (QA). Question-Answering is the task of providing an answer to a given question. For example, when a user submits a question like "What is the largest ocean on Earth?" the system is expected to provide an answer, such as "The Pacific Ocean". The industry as of late is more used to question-answering systems such as OpenAI's *ChatGPT* which differs from open-retrieval question-answering systems. However, there is a growing trend towards open-retrieval question-answering, which provides answers by searching for relevant information in a collection of documents.

In this Master Thesis, we collaborate with Consafe Logistics AB to showcase the performance of different open-retrieval question-answering system configurations on open-source benchmark datasets as well as a curated technical system documentation dataset.

We evaluate these systems based on the metrics Exact Match, F1, BERTScore, and Recall@k for performance and computational runtime, to provide a wider view of reference for which aspects to consider when implementing such a QA system. This will assist anyone interested in understanding the advantages and drawbacks of certain design choices when implementing an open-retrieval question-answering system.

1.1 Background

Consafe Logistics AB is a Swedish-founded company with over 40+ years of experience developing solutions within logistics and supply chain, with around 20+ of those years being software-related. Their main product offering revolves around warehouse management and control through Astro. With any large software solution, there exists extensive documentation of its inner workings in the form of technical system documentation. Searching through the current documentation is done on a keyword basis, and Consafe Logistics aims to try to improve the efficiency and precision of searching for answers in their documentation of Astro, by exploring the capabilities of question-answering with Large Language Models.

The company hopes that this will improve the ease of use of answer searching within documentation for new developers and customer support, and hopefully be a further endeavor into the possibilities of utilizing question-answering systems.

1.2 Research Questions

The research questions that we answer in this paper are the following:

- What combination of Open-Retrieval Question-Answering components achieves the highest performance on benchmark datasets and technical system documentation domain?
- What properties affect the performance of Question-Answering systems in the technical system documentation domain?

1.3 Delimitations

With more and more datasets that include samples of predicting a *No Answer*, and with us utilizing both extractive and generative models, we decided that for a question-answering system providing a wrong answer is equivalent to providing a no answer. Thus we have placed the following delimitation:

- All documents, questions and answers are in English.
- All question-answering systems are never expected to provide an answer that is a *No Answer/Empty*.

1.4 Contribution

The result from this thesis will contribute to Consafe Logistics AB's research into solutions for easier and more efficient search in documentation. This work will give a clearer and wider view of how effective open-retrieval question-answering systems can be in real-world applications. It will also make it easier for companies to see the performance differences

between a variety of QA pipelines, to make better decisions about what initial configurations could perform well on their specific task.

1.5 Related Work

Interest in developing open-retrieval question-answering systems has increased lately, due to their ability to retrieve relevant information from large document collections to answer the user questions. In recent years, significant advancements in neural network architectures have greatly enhanced their effectiveness. This section provides an overview of the development and key milestones in the implementation of QA systems.

Transformer-based Architectures

The introduction of the Transformer architecture by Vaswani et al. in 2017 represented a big advancement in the evolution of question-answering (QA) systems. The traditional conventional recurrent or convolutional neural networks had limited ability to effectively capture long-range dependencies in sequences, the Transformer model solved that problem by introducing an innovative self-attention mechanism, enabling the capture of long-range dependencies within sequences with greater efficiency. This architecture revolutionized natural language processing tasks by allowing models to process input sequences in parallel, making it highly scalable and efficient (Vaswani et al., 2017).

Advancements in Open-Domain QA

Machine reading comprehension (MRC) has evolved rapidly with advancements in machine learning, particularly through transformer architectures like BERT. These models trained on vast amounts of text data, excel in understanding and responding to questions, transforming natural language processing (Chen and Yih, 2020).

Chen et al. (2017) introduced a method that integrates a Retriever, serving as an information retrieval (IR) component, with a Reader, that works as a neural machine reading comprehension component. The Retriever's role is to retrieve relevant documents from a database based on the question, while the Reader formulates an answer using these documents and the question itself.

Sentence-Transformers and Dense Retrieval

The introduction of Sentence-Transformers and Dense Retrieval represents a significant milestone in the development of question-answering (QA) systems, particularly in the realm of semantic search and context-aware retrieval. Unlike the Transformer architecture, which primarily focuses on sequence-to-sequence tasks and language modeling, Sentence-Transformers and Dense Retrieval use dense representations of text for semantic search and information retrieval (Karpukhin et al., 2020).

Dense Retrieval techniques enhance QA systems by using dense representations to enable semantic search and relevant information extraction from knowledge bases. Unlike traditional

sparse retrieval methods, which rely on keyword matching or sparse embeddings, Dense Retrieval techniques encode information into dense vectors, enabling deeper understanding and retrieval of textual information (Karpukhin et al., 2020).

1.6 Work Division

Both authors have contributed equally to this work. The important aspects of the work were performed and reviewed by both authors to ensure that a high quality was maintained in the experiments. Tasks were split between authors to benefit parallel work. Samil Kapetanovic worked mainly on code integrations, retriever/ranker and runtime evaluations. Mohammad Raja worked mainly on data analysis, reader, and end-to-end evaluation.

However, both have contributed to all these parts in some fashion. The aspects that were performed together were data collection, data preprocessing, annotation of ConsafeQA, and all experiments performed with the ConsafeQA dataset. The work on the report has been distributed equally between the authors for each section.

Chapter 2
Literature Review

2.1 Question-Answering Concepts

Section 2.1.1, defines the task of open-retrieval question-answering systems and the existing paradigms of question-answering. In Section 2.1.2, an overview of the entire system is given, and the individual components are presented.

2.1.1 Task Description

The task of open-retrieval question-answering systems is, given the problem of answering a factoid question, use a knowledge base/collection of unstructured documents to find the answer (Chen et al., 2017). Given a question, the process of finding an answer can be narrowed down to two main steps. The first step is to search within a collection of documents to find the most relevant ones that could contain the answer. The second step is to 'Reason' around the retrieved documents to try and answer the question. This process is very similar to how humans try to look for an answer inside an encyclopedia (Chen et al., 2017).

Paradigms of Question-Answering

There are numerous frameworks of question-answering systems, however, a majority of them can be categorized into two different paradigms (a) open-domain question-answering, and (b) closed-domain question-answering (Su et al., 2023). For this paper, open-domain question-answering will be the focus and we will instead use open-retrieval question-answering, to refer to systems that provide contexts from a large document collection, this is because of the double meaning of open-domain as "covering topics from many domains" (Asai et al., 2020).

An open-retrieval (or *open-book*Su et al. (2023)) question-answering system involves a retriever that selects relevant contexts from a large collection of documents and a reader model

for 'reasoning' around the retrieved contexts (Das et al., 2019). This 'reasoning' from the reader model for open-retrieval question-answering can be of two types: (a) extracting the answer, or (b) generating an answer, which is often an implementation choice.

A closed-retrieval (or *closed-book*) question-answering system instead only relies on the reader model without any external knowledge to try, directly answering a question, by leveraging the parametric knowledge stored in pre-trained language models (Su et al., 2023).

2.1.2 Components of Open-Retrieval Question-Answering Systems

An open-retrieval question-answering system can be built in several different configurations, utilizing different components, however, it always includes two fundamental components (Chen et al., 2017).

The *Retriever* receives a question and then attempts to retrieve top_k relevant documents from the knowledge base. A knowledge base, also referred to as a *document store*, is an efficient way of storing a large corpus of documents for document search (Kononenko et al., 2014).

The other main component is the *Reader* which takes both a question and a document as input and then provides top_k answers to the question. These answers can either be an extracted answer (a span) from the retrieved document or a generated answer.

Of course, there exist even more components that can be introduced into the system to try and improve it, most notably a *Ranker* (Dong et al., 2023). There are also numerous other configurations utilizing aspects such as Query Transformations (e.g. *HyDE* (Gao et al., 2023)) and advanced retrieval utilizing metadata filtering. The fundamental components and a Ranker will be utilized in different ways for the experiments in this paper.

When it comes to the term top_k, in the context of open-retrieval question-answering systems, "top" refers to the highest-ranking candidates, while "k" means a specific quantity or maximum number of such candidates to be selected. These candidates can be documents for a Retriever/Ranker or possible answers for a Reader. Figure 2.1 shows an overview of the relation between the components.

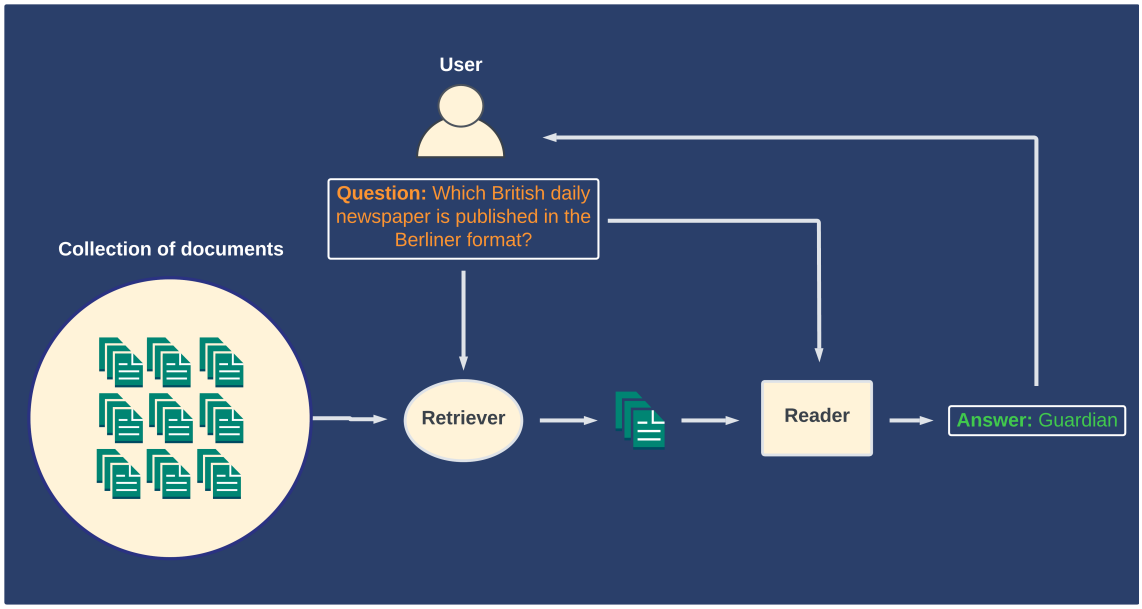


Figure 2.1: An overview of the relation between components of a Retriever-Reader open-retrieval question-answering system

Retriever

Information retrieval plays a significant role in the efficiency of open-retrieval question-answering systems. Retrievers are one of the fundamental components encompassing such a system, which are designed to filter through the vast collection of documents to retrieve relevant information in response to user queries (Farea et al., 2022). For a large collection of documents, it is a reasonable strategy to try and find a smaller collection of documents relevant to a query, instead of going through each document separately. Retrievers can be divided into two categories, sparse- and dense retrievers (Arabzadeh et al., 2021). Both categories of retrievers, function on the same idea of doing a similarity search within a vector space.

Sparse. Traditional methods such as TF-IDF and BM25, utilize sparse retrieval techniques. TF-IDF leverages term frequency and inverse document frequency to assign numerical weights to terms, creating sparse vectors that try to represent the document content.

As seen in the equation, (2.1), TF means Term frequency which is the number of times a term has been mentioned in a corpus of text ($n_{t,d}$) divided by the total number of terms in the corpus. On the other hand, IDF as the equation, (2.2) shows, is the Inverse Document Frequency which measures how important is a word within a corpus of text. It is calculated by dividing the total number of documents (D) within the corpus by the number of documents in which the term (d) appears. Finally, the TF-IDF score is calculated as the multiplication of TF and IDF (2.3).

$$\mathrm{TF}(t,d) = \frac{n_{t,d}}{\sum_{t' \in d} n_{t',d}} \tag{2.1}$$

$$\text{IDF}(t, D) = \log\left(\frac{|D|}{|\{d \in D : t \in d\}|}\right) \tag{2.2}$$

$$\text{TF-IDF}(t, d, D) = \text{TF}(t, d) \times \text{IDF}(t, D) \tag{2.3}$$

Sparse retrievers are usually viewed as a bag-of-words technique, that only looks at exact word matches (Li et al., 2022), meaning documents not containing words within the query will not be retrieved. These approaches do not take the semantics around the document into account, including the ordering of the words within a document (Thakur et al., 2021). Sparse retrieval methods have been shown to often fail to rank the most relevant documents at the top, but offer an overall recall that is high (Li et al., 2022). A key aspect of sparse retrievers is the fact that large amounts of data are not required for training.

BM25. One of the most widely used sparse retrievers is BM25, or Best Matching 25. It is an extension of the TF-IDF method and has been around for a long time (Robertson and Walker, 1994). BM25 uses three properties: term frequencies, document frequencies, and document lengths (Rosa et al., 2021). Even though it is a very traditional method, it has been shown that for new application areas with no training data, it still outperforms state-of-the-art results from dense retrievers (Izacard et al., 2021).

Dense. Dense Retrievers have shown a significant advancement in information retrieval, by utilizing dense vector representations. Unlike traditional sparse representations, dense retrievers utilize embeddings, to try and capture the semantics of words within a document (Karpukhin et al., 2020). This means that even if a query does not contain the actual keyword of a document, they can still be mapped to vectors relatively close to each other. Dense representations have become more and more popular throughout the years, and this is heavily fuelled by the success of pre-trained language models like BERT, because of the advantage of capturing contextual information (Devlin et al., 2018). Most contextual dense retrievers follow the dual-encoder (or *BI-encoder* architecture) (Dong et al., 2023).

GTE. **G**eneral **T**ext **E**mbedding, or GTE, is a fairly recent model for creating dense embeddings for text, that can be utilized for retrieval tasks. It is based on the vanilla dual-encoder (or *BI-encoder*) architecture and has been shown to rank highly in retrieval benchmarks (Li et al., 2023). GTE is trained through contrastive learning on open-source datasets, it is first pre-trained on unsupervised data ranging from web pages, scientific papers, community QA forums, social media, knowledge bases, and code repositories. It was then fine-tuned on supervised data with human annotation, including web search, open-domain QA, fact verification, and paraphrases (Li et al., 2023). GTE comes in three versions *small*, *base*, and *large*, all initialized from the base language models *microsoft/MiniLM-L12-H384-uncased*, *bert-base-uncased*, and *bert-large-uncased* respectively (Li et al., 2023).

Ranker

Rankers are an optional addition to an open-retrieval question-answering system that is utilized between the retriever and reader components. After the initial retrieval of documents,

the ranker steps in to further refine the relevance and ranking of documents. A ranker follows some type of cross-encoder architecture (Dong et al., 2023). The essence of a cross-encoder takes a query-document pair as input and outputs a relevance score. The reranking process is performed independently for each pair (Ma et al., 2023), ensuring a nuanced evaluation that captures the unique relationships between the query and the retrieved documents. From a human perspective, a ranker can be seen as an expert in re-organizing a pile of documents.

Reader

In any question-answering system, there exists a Reader component. The reader has the role of comprehending and extracting meaningful answers from the retrieved set of documents. It is designed to interpret the content within given documents and either extract or generate an answer to the user question. A brief explanation of certain terms used within the following sections will be highlighted:

1. *Span*, in the context of question-answering systems, a "span" is a continuous sequence of words or tokens extracted from a document. It is represented by a start and end index from a document.

2. *Head*, in the context of model architecture, the "head" refers to the final layer of a model architecture. This layer is added to produce the desired output of a model for a specific task. In the context of question-answering it is a classification layer that outputs probability for each token to be the start or end of the answer.

3. *Tokenization*, in natural language processing is used to transform the provided text into the desired input of a model. This involves splitting a text into tokens, depending on the decided tokenization strategy, these tokens can be words, subwords, or characters. Each token is then converted to a numerical representation through some type of look-up table.

Extractive. Also known as encoder-based transformers. An extractive reader answers a question by extracting the span of the answer from the provided context. This is done by fine-tuning an encoder-based transformer by adding a classification head after its last layer to output the probabilities of each token in the context being the start or end token of the answer span (Devlin et al., 2018). The input to the model is the tokenized question and context concatenated, the special classification token [CLS] is then added at the beginning of the input and [SEP] is put in to separate the question and context (Devlin et al., 2018). Encoder-based models are popular for extracting answers because of their ability to capture the semantic relationship between all the words in the question and context.

Generative. Also known as encoder-decoder or decoder-based transformers. While extractive readers output the span of the answer, generative readers instead generate the answer in an autoregressive way, meaning its decoder is only allowed to view past outputs (Raffel et al., 2019). In an encoder-decoder transformer, the encoder, instead of predicting the start and end tokens directly, is used as a contextual representation for the decoder (Luo et al., 2022). The decoder is the part of the transformer that generates the answer sequence token by token. This continues until a stopping criterion is met, such as reaching a maximum sequence

length. The input for an encoder-decoder and decoder do differ. For a decoder-only transformer, the prompt with the question and context is sent directly to the decoder to start predicting the next tokens (Radford et al., 2018). For an encoder-decoder transformer, the prompt is sent to the encoder where the output from the encoder is used by the decoder when predicting the next tokens (Raffel et al., 2019). The generative reader is quite advantageous in situations where the answer requires a more complex response or where the answer is not explicitly mentioned in the context.

2.2 Datasets

We employ three monolingual datasets, restricted to the English language, to evaluate all the different configurations of our open-retrieval question-answering systems. The reason for only evaluating the English language is because the documents provided by Consafe Logistics are exclusively in English and the ease of finding better-quality benchmark data. In Section 2.2.1 we present the MRQA 2019 Shared Task dataset, a benchmark dataset, consisting of numerous other common datasets used for reading comprehension tasks. Section 2.2.2 continues with TriviaQA, another benchmark dataset, commonly used in open-retrieval question-answering tasks. Lastly, in Section 2.2.3, we introduce ConsafeQA, a private question-answering dataset created by Consafe Logistics and ourselves.

2.2.1 MRQA 2019 Shared Task

The MRQA 2019 Shared Task dataset encapsulates several diverse sub-domain datasets. The goal of the dataset as outlined by the *MRQA team* (Fisch et al., 2019), is to evaluate the generalization capabilities of reading comprehension systems. The training data of MRQA, consists of six sub-domain datasets, *SQuADv1* (Rajpurkar et al., 2016), *NewsQA* (Trischler et al., 2016), *TriviaQA* (Joshi et al., 2017), *SearchQA* (Dunn et al., 2017), *HotpotQA* (Yang et al., 2018) and *Natural Questions* (Kwiatkowski et al., 2019). The part of development data referred to as out-of-domain data, consists of six sub-domain datasets *BioASQ* (Tsatsaronis et al., 2015)), *DROP* (Dua et al., 2019), *DuoRC* (Saha et al., 2018), *RACE* (Lai et al., 2017), *RelationExtraction* (Levy et al., 2017) and *TextbookQA* (Kembhavi et al., 2017). The sample distribution of the dataset can be seen in tables 2.1 and 2.2. For the remainder of this paper, we will utilize only the out-of-domain development data for evaluation purposes and will refer to it as *MRQA*. For more details about each sub-domain dataset, we refer to the original papers.

Dataset	Context	Samples
SQuAD	Wikipedia	86,588
NewsQA	News articles	74,160
TriviaQA	Web snippets	61,688
SearchQA	Web snippets	117,384
HotpotQA	Wikipedia	72,928
Natural Question	Wikipedia	104,071
Total		516,819

Table 2.1: MRQA training data

Dataset	Context	Samples
BioASQ	Science articles	1,504
DROP	Wikipedia	1,503
DuoRC	Movie plots	1,501
RACE	English Examinations	674
RelationExtraction	Wikipedia	2,948
TextbookQA	Textbook	1,503
Total		9,633

Table 2.2: MRQA out-of-domain development data

Format

Each sub-domain dataset was adapted to follow a unified, extractive format (Fisch et al., 2019). First, the answer to each question had to appear as a span within the passage, making it extractive. Second, the passages that were longer than 800 tokens were truncated, and included in the dataset if the answer appeared in those 800 tokens (Fisch et al., 2019).

2.2.2 TriviaQA

TriviaQA is a question-answering dataset containing around 650,000 question-answer-evidence triples (Joshi et al., 2017). The dataset is divided into a Wikipedia and a Web version, where the distinct difference is where the evidence documents originate from. For each version, there exist different splits, and the one we will be focusing on is the verified development data. Where the answer for each question within the dataset has been verified to be correct and exists within the evidence document (Joshi et al., 2017).

A distinctive feature of TriviaQA is that the questions originate from trivia enthusiasts independently of the evidence documents, a deliberate strategy aimed at mitigating biases in the formulation of questions (Joshi et al., 2017). It also emphasizes an increased prevalence of lexical and syntactic variations between questions and answers, which often require multi-sentence reasoning (Joshi et al., 2017).

Format

There is a provided conversion script for converting TriviaQA to a SQuAD format (Joshi et al., 2017). We adapted this script to include all the possible relevant variations of the correct answer for each question-answer-evidence triple. For simplicity reasons and to make the open retrieval question-answering task generalize better, we decided to merge the two verified versions of Wikipedia and Web into one. We performed cleaning on the merged dataset by excluding based on our previous delimitations, any samples that have an empty/no answer and any duplicate question samples. In the rest of this paper, we will refer to this merged version of the verified datasets of TriviaQA as *TriviaQA*. Table 2.3 shows examples of the samples from the SQuAD formatted TriviaQA dataset.

Truncated Context	**Question**	**Answer**
The Jurassic (; from Jura Mountains) is a geologic period and system that extends from 201.3 Mya (million years ago) to 145.5 Mya; from the end of the Triassic to the beginning of the **Cretaceous**. ...	What is the next in the series: Carboniferous, Permian, Triassic, Jurassic?	Cretaceous
Confessions of an English Opium-Eater (1821) is an autobiographical account written by **Thomas De Quincey**, about his laudanum (opium and alcohol) addiction and its effect on his life. The Confessions was " the first major work **De Quincey** published and the one which won him fame almost overnight ...	Who wrote 'Confessions of an English Opium Eater'?	Thomas De Quincey **or** De Quincey
... **Grace Slick** (born **Grace Barnett Wing**, October 30, 1939 in Evanston, Illinois) is an American singer and songwriter, who was one of the lead singers of the rock groups Jefferson Airplane, Jefferson Starship, Starship and also as a solo artist, for nearly three decades, from the mid-1960s to the mid-1990s. ...	Who was the main female singer with the groups Jefferson Airplane, Jefferson Starship and Starship?	Grace Slick **or** Grace Barnett Wing

Table 2.3: Sample format of TriviaQA, where **or** indicates several possible candidate answers

Statistics

The entire TriviaQA dataset contains 95,000 question-answer pairs, which on average each have six supporting evidence documents, resulting in around 650,000 question-answer-evidence triples (Joshi et al., 2017). From our inspection, there are in total 486,993 unique documents. From the merged TriviaQA version that we formatted, there are in total of 545 questions which on average have around 1.6 possible answers. Table 2.4 shows an overview of the sample statistics around the TriviaQA verified merged dataset.

Statistic	Value
# Documents	486 993
# Questions	545
# Answers	878
Average number of words per document	2238
Average question length (measured by words)	13.1
Average answer length (measured by words)	1.6

Table 2.4: TriviaQA verified merged data statistics

2.2.3 ConsafeQA

It was challenging to find a dataset that aligns with our domain, therefore we created a dataset ConsafeQA, which is designed to meet our specific requirements. This dataset was collaboratively created by ourselves and employees at Consafe Logistics, using an annotation program developed for this purpose.

ConsafeQA uses a set of 179 documents. The documents consist of internal technical system documentation, where the main subjects revolve around logistics and supply chains. From these documents, tables and figures were excluded to align with the specific requirements of our research objectives. To ensure a diverse representation of the content, the annotation program randomly selects passages from these documents. Each passage is meant to be read by the user, who then needs to formulate a question about the passage and extract an appropriate answer for this question. This follows very similarly to the SQuAD format (Rajpurkar et al., 2016).

In total, ConsafeQA has 376 question-answer pairs, providing a domain-specific resource for evaluating and enhancing question-answering systems within the technical documentation domain. ConsafeQA contains proprietary data from Consafe Logistics AB and can thus not be shared with contexts. However, we do provide a synthetic made-up example of the format in table 2.5 and a data analysis of ConsafeQA in the following section.

Truncated Context	Question	Answer
... as defined by the system parameter sys23lav. This parameter defines the saving days also for all other assignment records. The default value is **20 days**. ...	What is the default value for the system parameter sys23lav?	20 days
... The background program L24CART searches ... When a cart is full, the program will automatically release the cart for **replenishment**. ...	When a cart is full, the L24CART program will automatically release it for what purpose?	replenishment

Table 2.5: Sample format of ConsafeQA with synthetic made-up examples

Data Analysis

Data analysis has been performed separately on both ConsafeQA and all the 179 documents it uses 2.2.3. The first part of table 2.6 presents statistics performed on the documents, while the second part shows the average question and answer lengths for ConsafeQA.

All the data analysis performed in this section, excluding Table 2.6, involved data cleaning by removing punctuation, converting the text to lowercase, and eliminating stop-words.

Statistic	Value
Total number of documents	179
Average number of words per document	3354.63
Total number of words across all documents	600478
Total number of unique words	12735
Average word length across all documents (measured by character)	4.87
Average sentence length across all documents (measured by words)	14.98
Total number of questions	376
Total number of answers	376
Average question length (measured by words)	13.21
Average answer length (measured by words)	6.80

Table 2.6: Basic Statistics

The word cloud presented in Figure 2.2 shows the most frequent words in the corpus, providing quick insights into the most important words. On the other hand, the TF-IDF analysis presented in Figure 2.3 provides a more comprehensive representation of word importance, where the most important words in Consafe documents are displayed along with their TF-IDF scores.

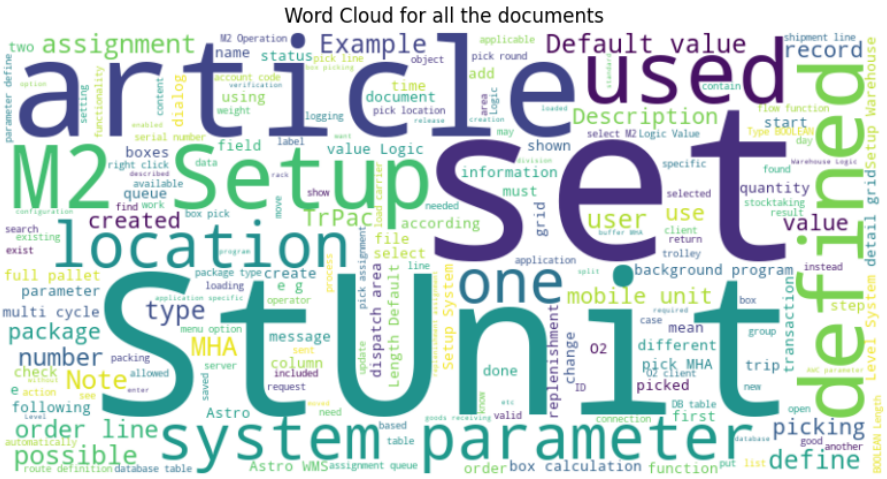


Figure 2.2: Word Cloud for all the documents that represent most frequent words

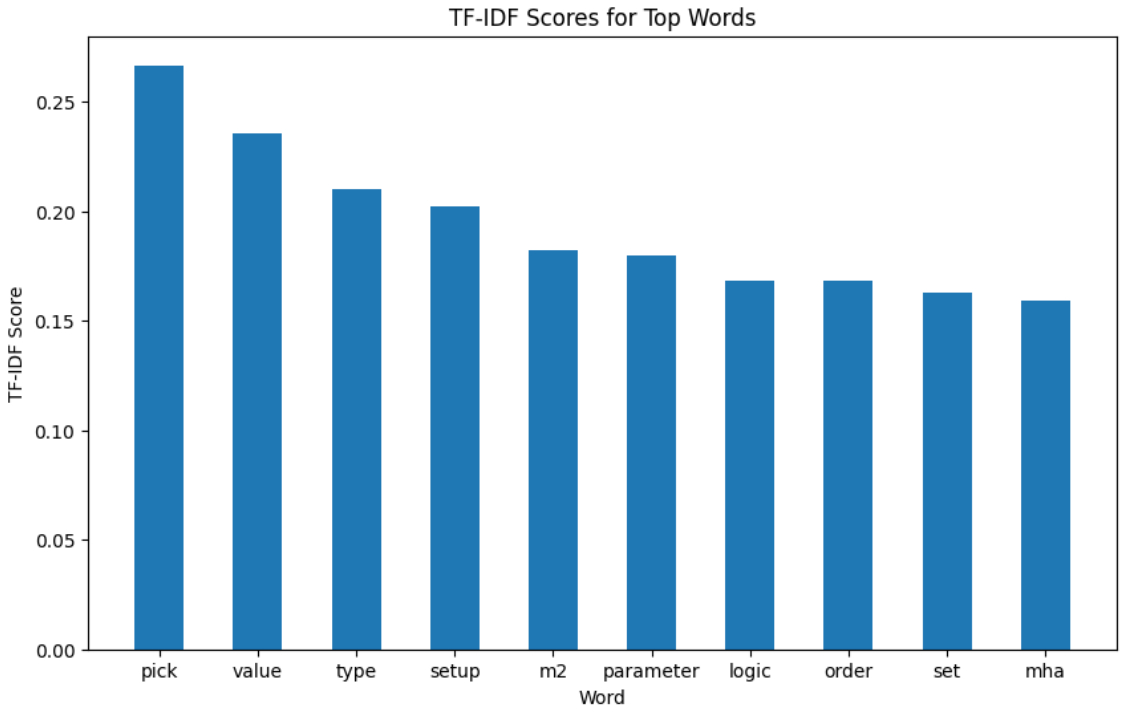


Figure 2.3: Most important words in all documents based on TF-IDF

Figure 2.4 shows the frequency distribution of question types within the ConsafeQA dataset, determined by the occurrence of the most common question words in the questions. The most common keywords measured by frequency for both the questions and the answers in ConsafeQA are displayed in Figures 2.5 and 2.6 respectively.

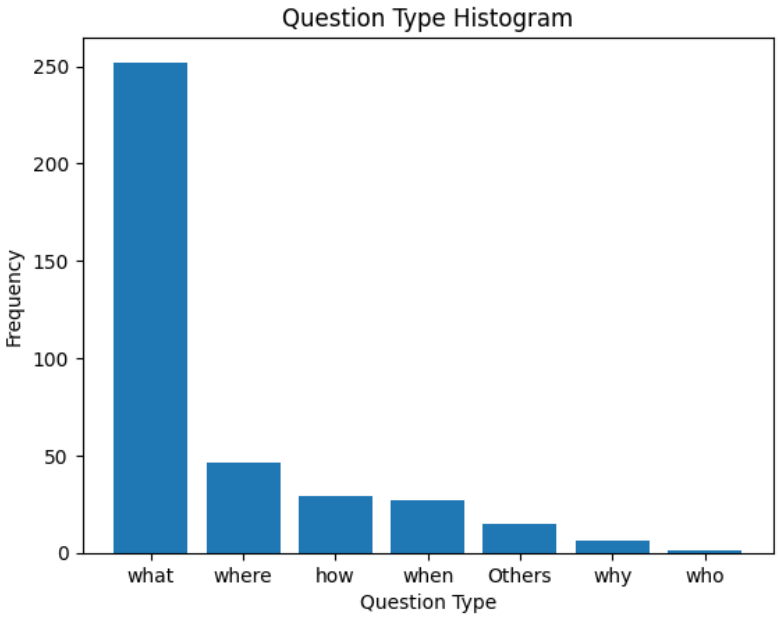


Figure 2.4: Question types

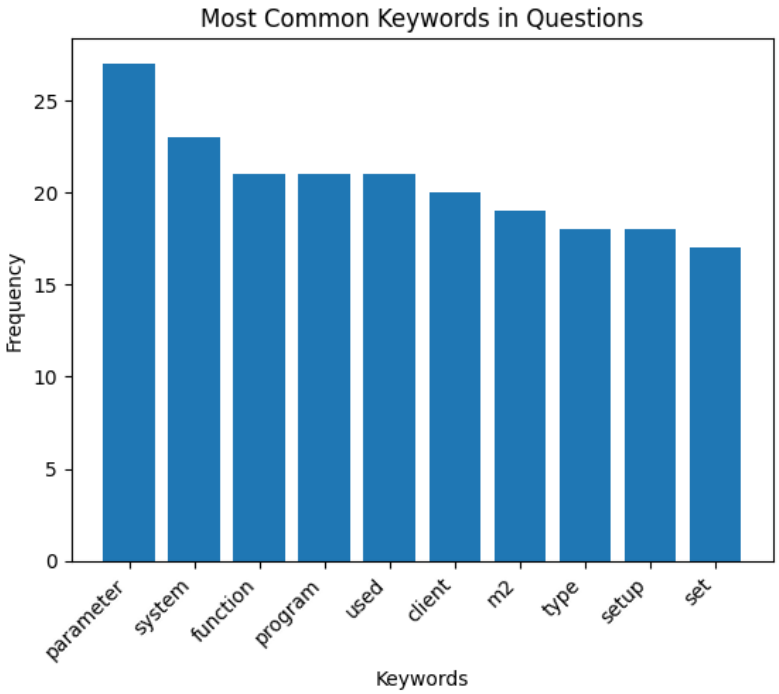


Figure 2.5: Most Common Keywords in questions

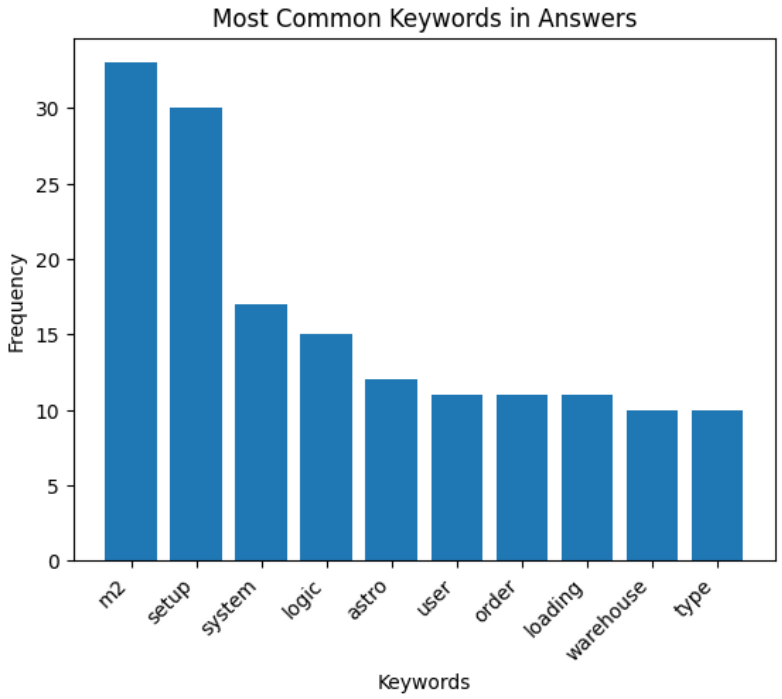


Figure 2.6: Most Common Keywords in answers

2.3 Evaluation Metrics

Evaluation metrics play a decisive role when deciding what components to use in a pipeline and also provide insights into the overall performance of a specific system configuration. When it comes to assessing a question-answering system, various types of evaluation metrics come into play.

Evaluating a question-answering system is difficult and the choice of the evaluation metrics is crucial. Some traditional metrics are commonly used in various NLP tasks such as Exact Match (EM), and F1. They focus on surface-level linguistic matching and have limitations in capturing the semantic meaning of the content. Therefore new metrics like BERTScore have emerged, offering a different perspective on system evaluation.

There are many evaluation metrics and they focus on evaluating different components. Metrics like Recall@k, for instance, primarily focus on assessing the output from the retriever, measuring the system's ability to retrieve relevant documents. In contrast, the F1 score, Exact Match (EM), and BERTScore concentrate on evaluating the output of the reader. Em and F1 will reviewed in sections 2.3.2 and 2.3.3 respectively, BERTScore in section 2.3.4 and Recall@k in section 2.3.1

2.3.1 Recall@k

Recall@k measures the retriever's ability to retrieve correct documents by determining whether the retrieved documents include any of the golden documents. The top$_k$-Retriever refers to

the number of documents retrieved by the retriever from the document store (Zuva and Zuva, 2012). Equation (2.4) shows the formula Recall@k.

$$\text{Recall@k} = \frac{|Relevant \cap Retrieved|}{|Relevant|} \tag{2.4}$$

2.3.2 Exact Match

The Exact Match (EM) score is a metric that evaluates if the predicted output exactly matches the expected answer. It is a binary meaning, meaning that it is equal to 1 if the two answers are exactly identical and 0 otherwise (2.5) (Rajpurkar et al., 2016).

$$EM = \begin{cases} 1 & \text{if Prediction = Gold answer} \\ 0 & \text{otherwise} \end{cases} \tag{2.5}$$

2.3.3 Macro average F1 score

It is the average lexical overlap between the predicted answer and the golden answer. Both the predicted answer and the golden answer can be viewed as lists of words, which are then compared to each other, and the number of identical words is counted. This count is then used to calculate both precision and recall (Rajpurkar et al., 2016). Precision measures the accuracy of positive predictions, focusing on the correctly predicted words in the predicted answer (2.6). In contrast, recall focuses on the proportion of words in the golden answer that has been correctly predicted (2.7). The F1 score is the harmonic mean between precision and recall, which ensures that precision and recall contribute equally to the overall score 2.8. Macro average means that the F1 score is calculated individually for each sample, and then the mean of all these sample scores is computed without assigning any weights to specific samples.

$$\text{Precision} = \frac{\#\text{identical words}}{\#\text{words in the predicted answer}} \tag{2.6}$$

$$\text{Recall} = \frac{\#\text{identical words}}{\#\text{words in the golden answer}} \tag{2.7}$$

$$\text{F1} = \frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}} \tag{2.8}$$

2.3.4 BERTScore

The advancements in deep learning have given rise to new metrics that mimic human judgment and measure both syntactic and semantic meaning. The previous metrics do not evaluate similarity in a contextual way, in other words, they are purely lexical metrics. Just comparing words matching might be negative, because there could exist correct answers with the same meaning but using different words. However, BERTScore evaluates semantic meaning and correlates more closely with human judgment. In the paper written by Zhang* et al. (2020), multiple experiments show how BERTScore outperforms all other metrics such as

BLEU (Papineni et al., 2002) and ITER (Panja and Naskar, 2018)

BERTScore leverages large pre-trained language models (LLMs), specifically BERT, to generate scores for similarities between sentences. It computes precision, recall, and F1 measure, by measuring the pairwise cosine similarity between the BERT contextual embeddings of individual tokens in both the prediction and the golden answer (Zhang* et al., 2020).

Although BERTScore performs well overall and provides a more accurate evaluation than other existing metrics, it still has some drawbacks. According to Hanna and Bojar (2021), BERTScore struggles to give low scores to sentences that differ in content but share similar words with the reference answer. Also, it seems less effective for complex sentences compared to simpler ones (Hanna and Bojar, 2021). Multiple variants of BERT models can be used, such as RoBERTa and DeBERTa, with multiple model sizes and variants, each offering unique features and performance characteristics.

2.4 Word embeddings

When trying to represent text, it might not be enough to only take into account the bag-of-words encompassing the text, but instead what relationship each word within the text has to each other. This is achieved through dense vector representation, which is today called *word embeddings*. They are constructed in a way where operations between vectors provide a result that follows the semantic rules between the vectors. For example, if you have the four words King, Queen, Male, Female, and with each of their vector representations perform the operation of “King - Man + Woman”, you should get a vector that is relatively close to "Queen" (Mikolov et al., 2013b).

These word embeddings can be both static and contextual, static encompassing more traditional methods such as GloVe (Pennington et al., 2014) and word2vec (Mikolov et al., 2013a), while contextual being what is usually created through some form of transformer-based model that has been trained on a large amount of text data (Li et al., 2023). An integral part of improving modern machine learning techniques centered around text is constructing better text embeddings to be used in various downstream tasks (Devlin et al., 2018).

2.5 Transfer Learning and Fine-tuning

Large Language Models (LLM) are as their name suggests, quite large in parameter size. This creates limitations in the availability of pre-training a model from scratch for a new task or domain because of the heavy computational costs (Yao et al., 2021).

Transfer learning addresses this issue by utilizing the knowledge acquired from an already pre-trained model, and adapting it to perform well in a specific downstream task (Devlin et al., 2018). This is typically achieved by freezing some or all the existing parameters within a pre-trained model and appending a final layer head. This additional layer is then trained on a designated downstream task dataset, such as question-answering, classification, etc (Zhuang

et al., 2019).

Fine-tuning can involve both appending a final layer head and also selectively unfreezing parts of, or the entire pre-trained model and further adapting its parameters to the specific task or domain at hand. Fine-tuning can also be done depending on the model and task with very minimal added task-specific parameters or none, and simply fine-tuning all pre-trained parameters (Radford et al., 2018).

It is also necessary to consider the nature of the downstream task and the domain shift between the pre-training and target tasks. Making sure that the similarity between these domains enhances the effectiveness of fine-tuning. However, if the domain the model was pre-trained on diverges significantly from the domain it is being fine-tuned on it can result in *negative transfer*, which means the additional training can have negative results for the specific domain (Liu et al., 2019).

2.6 Architectures

Section 2.6.1 will explain the Transformers architecture, including various variants and an overview of the attention mechanism. In Section 2.6.2, a detailed review of transformer-based pre-trained models will be provided. Section 2.6.3 will review and outline the approach for quantizing LLMs.

2.6.1 Transformers

Transformers is a type of neural network that transformed the field of deep learning when it came out in 2017. The architecture that was introduced by the paper was an Encoder-Decoder architecture. The encoder takes the input and processes it through encoding layers, while the decoder generates text using decoding layers. Then adding attention mechanisms to these components enables the model to develop a comprehensive understanding of sentence structure and the semantic meaning of individual words.

The following subsections will delve into these components in more detail. However, this section serves as an overview of the transformer architecture. For a more comprehensive understanding, refer to the original paper (Vaswani et al., 2017).

Encoder-Decoder

The original transformer was introduced as an Encoder-Decoder architecture, but it comes in different alternatives such as encoder-only, decoder-only, and full complete transformers. Each of these is appropriate to use depending on different tasks.

In the Encoder-Decoder architecture, the encoder component's role is to process the input text and extract relevant information from it. So it receives an input sentence, processes it, and produces a fixed-size vector representation. This vector representation is then passed to

the decoder component, which is responsible for generating the output sentence. This architecture is useful for tasks such as language translation where the encoder receives a sentence in one language and then the decoder produces the translated sentence (Vaswani et al., 2017).

Encoder-Only uses only the encoder part of the full transformer architecture. The Encoder-Only architecture is commonly used for tasks where only the encoding of the input sequences is required, some examples of such tasks: are text classification or information retrieval. Whereas, the Decoder-only is useful for tasks that involve generating output sequences without the necessity of encoding input sequences, such as language generation (Vaswani et al., 2017).

Cross-Encoder & Dual-Encoder

Both cross-encoder and dual-encoder are two transformer-based architectures designed to measure the similarity between two sentences. However, they are constructed differently.

A cross-encoder employs a single encoder component that simultaneously receives both sentences as input and outputs a similarity score between 0 and 1, indicating the similarity of the two sentences (Reimers and Gurevych, 2019).

In contrast, a dual-encoder, also known as a bi-encoder, consists of two encoder components. Each encoder independently processes one of the two sentences. After that, the sentences are mapped independently into a shared feature space, where cosine similarity is used to measure their similarity (Karpukhin et al., 2020). Refer to Figure 2.7 to see the structural differences.

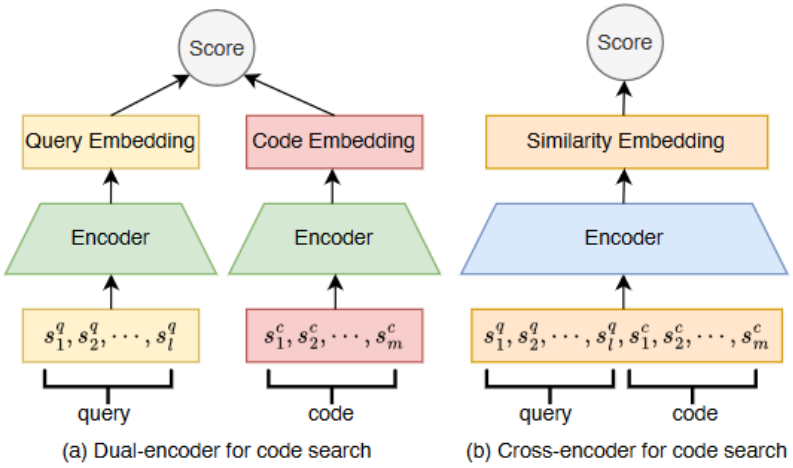


Figure 2.7: Comparing Bi-Encoder and Cross-Encoder Architectures, from (Dong et al., 2023).

Attention

Attention is a mechanism for correctly associating words in a sentence. It works by calculating the similarity of each word to all of the words in the sentence. Once the similarities

are calculated, they are used to determine how the transformer encodes each word. Afterward, positional encoding is performed to keep track of word order. For each word, a vector x_i is obtained. After that, three vectors referred to as the query q_i, key k_i, and value v_i are created for each of these x_i vectors. The attention is calculated according to 2.9, where the matrices V, K, and Q contain the value, key, and query vectors respectively for all words and d_k denotes the dimension of K. This attention function is called scaled dot-product attention because dot-product is calculated between the query and the corresponding key. There are other types of compatibility functions than the dot-product, but this was the attention function that was introduced by the original paper (Vaswani et al., 2017).

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V \tag{2.9}$$

Furthermore, we can extend the idea of attention to include multi-head attention. Here, we project the queries, keys, and values into different dimensions, known as heads h. Each head independently computes its output using 2.9. These outputs are then concatenated and projected, resulting in the final values. By doing this, we can process information more efficiently. Each head captures different aspects of the input sequence, enhancing our understanding (Vaswani et al., 2017). Figure 2.8 illustrates the two mechanisms.

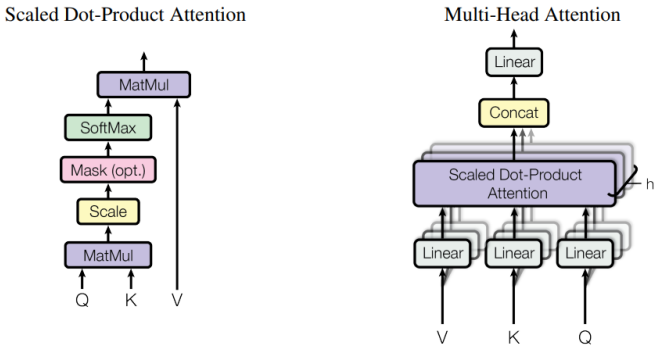


Figure 2.8: Scaled Dot-Product Attention vs Multi-Head Attention, from (Vaswani et al., 2017).

The Transformer Encoder-Decoder architecture heavily relies on the multi-head attention module. Additionally, it employs multi-head self-attention, with a key difference lying in their input sources. While multi-head attention integrates inputs from various sequences, multi-head self-attention gets inputs from the same sequence. In this architecture, the Encoder includes a multi-head self-attention module, and the Decoder includes both multi-head attention and multi-head self-attention modules. A comprehensive overview of the Architecture is illustrated in Figure 2.9.

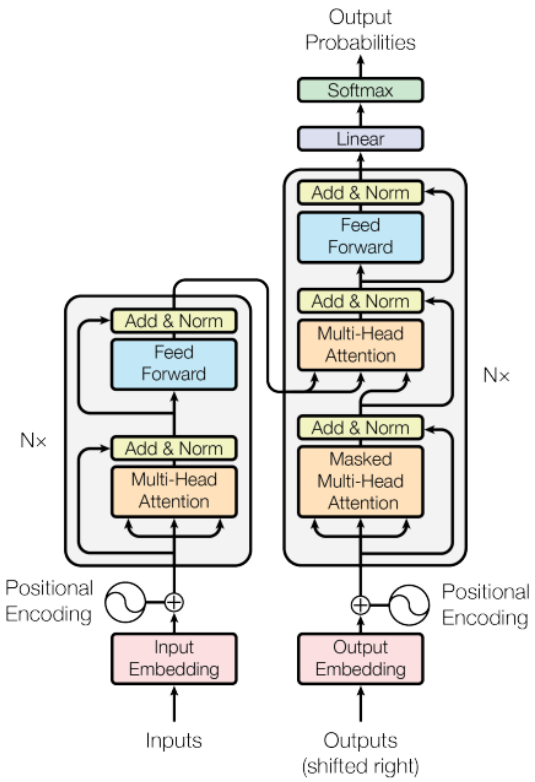


Figure 2.9: The Transformer - model architecture, from (Vaswani et al., 2017).

2.6.2 Pre-trained models

Pre-trained Language Models (PLMs) are transformer-based neural networks that have been trained on very rich training datasets and then fine-tuned on a specific task. During training the model learns to predict the next word in a sentence. But with each iteration, the model adjusts its internal parameters to reduce the difference between its predictions and the actual outcomes. That process allows the model to learn a language and grasp the diverse contextual roles of words. In this section, we will review in more details only the reader models that were used across all our experiments, in other words the ones the most promising ones according to our first experiment.

ELECTRA

ELECTRA stands for Efficiently Learning an Encoder that Classifies Token Replacements Accurately. Uses a new pre-training technique called Replaced Token Detection (RTD) to improve representation learning for language (Clark et al., 2020). This method outperformed the BERT method both by performance and efficiency 2.10. The main idea of this technique is training the model to distinguish between real and synthetic input tokens, in other words detecting and correcting errors in the input text, rather than predicting the next word in a sequence that was proposed by Devlin et al. (2018) and called masked language modeling (MLM). The key concept of MLM is to hide or mask a subset of the input sequence from the model. Afterwards, the models predict these hidden tokens based on the surrounding text,

and the model is then trained by minimizing the difference between the predictions and the masked tokens.

Model	Train FLOPs	Params	SQuAD 1.1 dev EM	SQuAD 1.1 dev F1	SQuAD 2.0 dev EM	SQuAD 2.0 dev F1	SQuAD 2.0 test EM	SQuAD 2.0 test F1
BERT-Base	6.4e19 (0.09x)	110M	80.8	88.5	–	–	–	–
BERT	1.9e20 (0.27x)	335M	84.1	90.9	79.0	81.8	80.0	83.0
SpanBERT	7.1e20 (1x)	335M	88.8	94.6	85.7	88.7	85.7	88.7
XLNet-Base	6.6e19 (0.09x)	117M	81.3	–	78.5	–	–	–
XLNet	3.9e21 (5.4x)	360M	**89.7**	**95.1**	87.9	**90.6**	87.9	90.7
RoBERTa-100K	6.4e20 (0.90x)	356M	–	94.0	–	87.7	–	–
RoBERTa-500K	3.2e21 (4.5x)	356M	88.9	94.6	86.5	89.4	86.8	89.8
ALBERT	3.1e22 (44x)	235M	89.3	94.8	87.4	90.2	88.1	90.9
BERT (ours)	7.1e20 (1x)	335M	88.0	93.7	84.7	87.5	–	–
ELECTRA-Base	6.4e19 (0.09x)	110M	84.5	90.8	80.5	83.3	–	–
ELECTRA-400K	7.1e20 (1x)	335M	88.7	94.2	86.9	89.6	–	–
ELECTRA-1.75M	3.1e21 (4.4x)	335M	**89.7**	94.9	**88.0**	**90.6**	**88.7**	**91.4**

Figure 2.10: ELECTRA performance against other models on the SQuAD dataset, from (Clark et al., 2020).

DeBERTa

The latest versions of the DeBERTa model, originally inspired by BERT, have undergone substantial enhancements. Previous versions of DeBERTa relied on the conventional Masked Language Modeling (MLM) approach during pre-training. The latest version of DeBERTa, introduced in 2023 and named DeBERTaV3, adopted the Replaced Token Detection (RTD) method, inspired by ELECTRA (He et al., 2021). This transition aims to enhance both training efficiency and overall model performance. DeBERTaV3 addressed some issues encountered by ELECTRA regarding embedding sharing and solved them by introducing the Gradient-Disentangled Embedding Sharing (GDES) method. GDES disentangles attention mechanisms, which in turn enhances model convergence. The result of this led to enhanced training efficiency and the creation of a higher-quality pre-trained model. Figure 2.11 shows how DeBERTaV3 outperforms other models on different datasets.

Model	MNLI-m/mm Acc	SQuAD v2.0 F1/EM	RACE Acc	ReCoRD F1/EM	SWAG Acc	NER F1
$\text{BERT}_{\text{large}}$	86.6/-	81.8/79.0	72.0	-	86.6	92.8
$\text{ALBERT}_{\text{large}}$	86.5/-	84.9/81.8	75.2	-	-	-
$\text{RoBERTa}_{\text{large}}$	90.2/90.2	89.4/86.5	83.2	90.6/90.0	89.9	93.4
$\text{XLNet}_{\text{large}}$	90.8/90.8	90.6/87.9	85.4	-	-	-
$\text{ELECTRA}_{\text{large}}$	90.9/-	-/88.1	-	-	-	-
$\text{Megatron}_{\text{336M}}$	89.7/90.0	88.1/84.8	83.0	-	-	-
$\text{DeBERTa}_{\text{large}}$	91.1/91.1	90.7/88.0	86.8	91.4/91.0	90.8	93.8
$\text{DeBERTaV3}_{\text{large}}$	**91.8/91.9**	**91.5/89.0**	**89.2**	**92.3/91.8**	**93.4**	**93.9**
$\text{ALBERT}_{\text{xxlarge}}$	90.8/-	90.2/87.4	86.5	-	-	-
$\text{Megatron}_{\text{1.3B}}$	90.9/91.0	90.2/87.1	87.3	-	-	-
$\text{Megatron}_{\text{3.9B}}$	91.4/91.4	91.2/88.5	89.5	-	-	-
$\text{DeBERTa}_{\text{1.5B}}$	91.7/91.9	92.2/89.7	90.8	94.5/94.0	92.3	-

Figure 2.11: DeBERTa performance against other models on multiple datasets, from (He et al., 2021).

Flan-T5

Flan-T5, short for Fine-tuned Language Net and Text-to-Text Transfer Transformer, is a pre-trained encoder-decoder model that has been trained on a wide range of datasets and language tasks. It comes in various sizes, ranging from Small to XXL, with parameter counts ranging from 80 million to 11 billion (Chung et al., 2022). Because of the different ways it was fine-tuned, significant improvements have been made in its reasoning capabilities. Figure 2.12 displays the fine-tuning approach which involves instruction fine-tuning and also using Chain-of-Thoughts Fine-tuning (CoT fine-tuning). CoT fine-tuning is a method that enables LLMs to think step-by-step, like following a set of instructions, making them more adept at handling complex reasoning tasks (Wei et al., 2022). Because Flan-T5 has been trained on many different tasks, it exhibits remarkable adaptability across different domains and tasks, making it highly customizable.

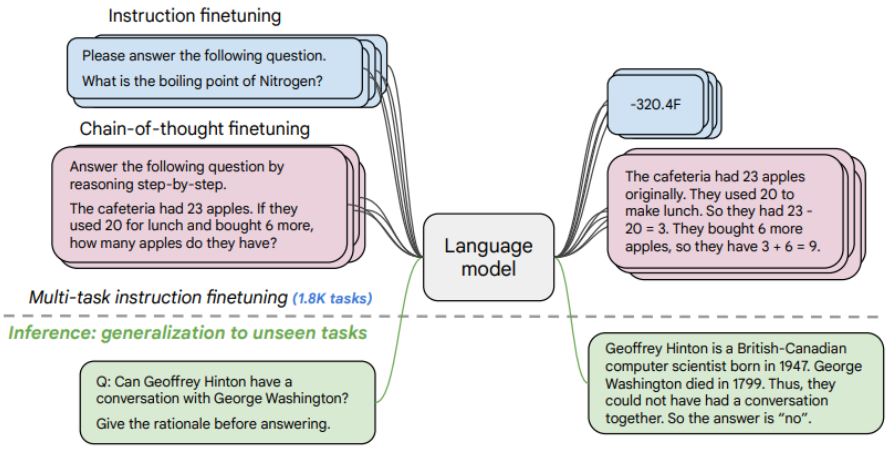


Figure 2.12: Finetuning tasks phrased as instructions, from (Chung et al., 2022).

BLOOMZ

BLOOM (BigScience Large Open-science Open-access Multilingual) is a decoder-only pre-trained large language model designed to be publicly accessible, addressing the inaccessibility of many LLMs to the public (Workshop et al., 2022). It performs well not only in English but also in other languages as it generates text in 46 natural languages. BLOOMZ is a finetuned variant of BLOOM that uses the multitask-prompted finetuning (MTF) approach to finetune the BLOOM model. MTF aims to fine-tune models on a wide variety of tasks, which has been shown to enhance their performance even on new, unseen tasks (Sanh et al., 2021). However, this approach has primarily been tested on English models and English data, whereas BLOOMZ focuses on multilingual capabilities (Muennighoff et al., 2022).

2.6.3 Quantized models

Increasing the parameter count of Language Models (LLMs) has been proven to enhance their performance (Frantar et al., 2022). However, too large models can pose challenges, especially when deployed on less powerful hardware with limited memory and computational

resources. Hence, quantized models have been introduced to solve these challenges by reducing the model's size while preserving its performance. This is achieved by reducing the precision of the model's weights. For instance, instead of using parameters represented in 32-bit floats, using 16, 8, or even fewer bits significantly reduces the model's size. Another important benefit of using a quantized model is to reduce the inference time.

Quantization can be performed using various methods and techniques. For instance, some techniques involve Post-Training Quantization (PTQ), which occurs after model training, while others use Quantization-Aware Training (QAT) during the model training stage. However, the GPTQ approach will be discussed in more detail in the following subsection.

GPTQ

GPTQ (Gradient-based Post-Training Quantization) is an approach that focuses on models that have Transformer-based architectures. GPTQ quantizes the weights of the LLM individually, processing each weight matrix in isolation. It transforms the floating-point parameters of each weight matrix into quantized integers, aiming to minimize the squared error between the quantized and original layers, this method is called Layer-Wise Quantization (Frantar et al., 2022). It also optimizes the rounding during the quantization, by using small batches of data, GPTQ adjusts rounding operations to maintain precision while reducing information loss.

A major benefit of GPTQ is its speed and efficiency. Even with models containing billions of parameters, GPTQ can quantize them in just a few hours, far quicker than other methods. Moreover, GPTQ consistently outperforms other quantization techniques, especially at lower-bit precisions. While some methods struggle to maintain accuracy below 8 bits, GPTQ maintains high-performance levels (Frantar et al., 2022). For more details and a comprehensive understanding of the GPTQ algorithm, please refer to the original paper (Frantar et al., 2022).

Chapter 3
Approach

When designing a performant open-retrieval question-answering system, several factors must be considered. This chapter delves into our approach to investigating underlying components within the system pipeline, overall system performance, and the optimal hyperparameters. An overview of the high-level experiments to identify the highest-performing system is outlined below:

1. Finding Readers that are the most suitable candidates for further experimentation. Section 3.1 goes into further detail.

2. Finding optimal components and hyperparameters within Retriever-Reader pipelines to identify their underlying impact on system performance. Section 3.2 goes into further detail.

3. Introducing a Ranker component, to see if the addition of a simple component improves system performance. Section 3.3 goes into further detail.

4. Runtime evaluation of the retriever, ranker, and reader components, to see what components consume the most runtime of the entire system. Section 3.4 goes into further details.

5. Evaluate the system using Consafe Logistics-specific data. With the purpose to see how well the system performs on real-life data. Section 3.5 goes into further detail.

3.1 Evaluation of Reader

To begin with, all the Reader models introduced in table 3.1 will be evaluated across all the six out-of-domain MRQA datasets. The objective of this experiment is to investigate and determine the four most promising reader models for further exploration in subsequent experiments. The choice of the MRQA dataset over others is because of the diverse domains

represented by the six datasets, which ensures the chosen model is generalized to various domains.

Each reader model will be evaluated independently on every MRQA dataset. As described before, each dataset sample has a text passage, a question, a corresponding golden answer, and some additional information. For every sample, the reader model is provided with the text passage and the question, and its task is to predict an answer for the question, meaning that the top_k-Reader value will be fixed to 1. The predicted answer will then be compared to the golden answer according to the dataset, and performance metrics such as Exact Match (EM), F1 score, and BERTScore will be calculated. The average performance of each model on these metrics across all six datasets will be calculated to have a comprehensive comparison to the other model when choosing the most promising reader models. The outline for this experiment is shown in figure 3.1.

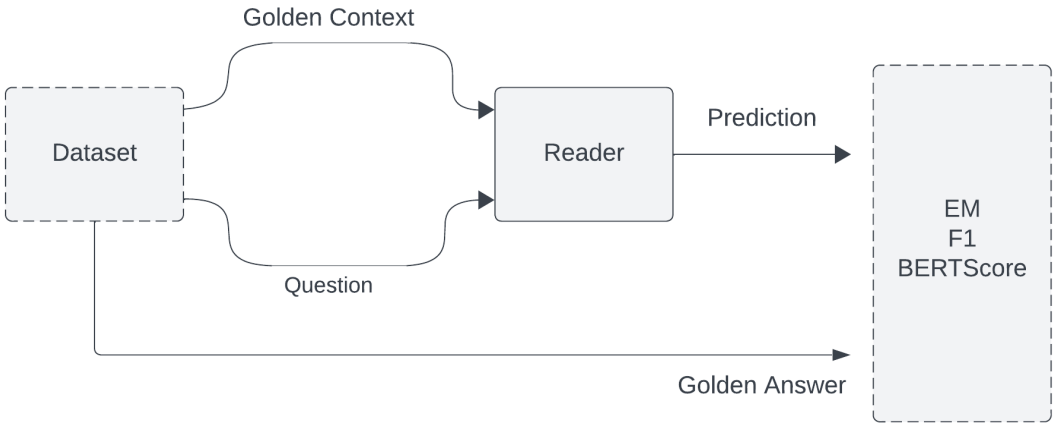


Figure 3.1: Experimentation process for evaluation of readers

The results of this experiment will be presented in two tables. The results for the extractive readers will be presented in a table separate from the performance results for generative readers. An important note is that we did not fine-tune the encoder-based models ourselves. These reader models were chosen because of their popularity and to cover diverse model categories, including encoder-only, decoder-only, and encoder-decoder models.

3.2 Retriever-Reader Pipeline Evaluation

After determining the promising reader models to proceed with, we now transition to a more real-world question-answering pipeline that incorporates a retriever component. This experiment is conducted using the TriviaQA dataset. Similar to the MRQA dataset, each sample in the dataset includes a text passage, a question, and golden answers. However, a key distinction in this experiment compared to the previous one is that the reader model does not receive the correct text passage associated with the question directly. Instead, it receives the text passage that is fetched by the retriever. An overall outline of the experimentation process is shown in figure 3.2.

Model name	Type	Fine-tuned on	Huggingface Identifier
DeBERTa	Encoder	MRQA	VMware/deberta-v3-base-mrqa
ELECTRA	Encoder	MRQA	VMware/electra-base-mrqa
RoBERTa	Encoder	MRQA	VMware/roberta-base-mrqa
BLOOMZ	Decoder	-	bigscience/bloomz-1b7
Mistral	Decoder (Quantized)	-	TheBloke/Mistral-7B-Instruct-v0.1-GPTQ
Flan-T5	Encoder-Decoder	-	google/flan-t5-large
mT0	Encoder-Decoder	-	bigscience/mt0-large

Table 3.1: Reader

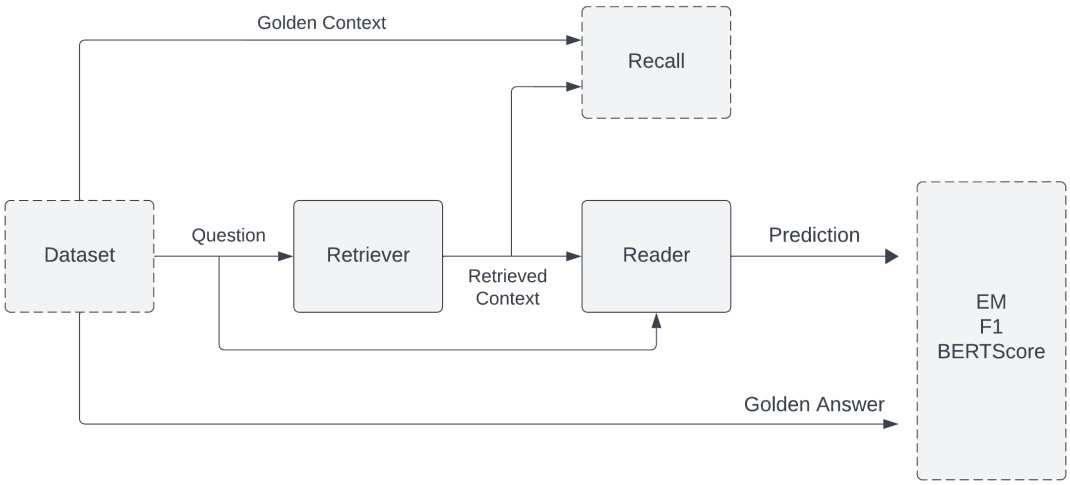


Figure 3.2: Experimentation process for evaluation of retriever-reader pipelines

As mentioned in the literature review section, the retriever is linked to the document store containing all documents. When given a question, the retriever's task is to retrieve documents it deems relevant. These selected documents are provided to the reader model, which in turn uses them to answer the question.

Two types of evaluation are performed in this experiment. The first one will be presented in more detail in the following subsection and will assess the retriever's ability to retrieve accurate documents. The second evaluation focuses on end-to-end performance, specifically examining whether the pipeline accurately predicts the correct answer for the given question. The retrievers to be experimented with can be seen in table 3.2. We chose BM25 and GTE to evaluate different retrieval methods, including both sparse and dense approaches.

Also, BM25 is well known for its strong performance in retrieval tasks and GTE is top-performing on the MTEB leaderboard [1] on retrieval datasets with a reasonable model size, making them good choices for our study.

[1] https://huggingface.co/spaces/mteb/leaderboard

Model name	Huggingface Identifier
BM25	-
GTE	thenlper/gte-base

Table 3.2: Retriever

3.2.1 Retriever Robustness Evaluation

The primary goal of this experiment is to assess the robustness of both the GTE and BM25 retrievers. This is achieved by varying the number of documents in the documents store, starting from 50,000, then 120,000, followed by 240,000, and finally including all TriviaQA documents totaling 482,993. Each split includes the smaller split and all the golden documents from the dataset have been added separately, for example, the 120,000 set includes all documents in the 50,000 set with an additional 70,000 documents. The goal of this experiment is to investigate whether increasing the number of documents in the document store affects the retriever's performance. We aim to determine if having access to more documents makes it more challenging for the retriever to accurately retrieve the correct document.

In this experiment, the top_k-Retriever hyperparameter is varied between 1-50. Two types of recall evaluations were performed, one to assess whether the retrieved documents contained the exact golden document and another to determine if a document among the retrieved documents explicitly contained the golden answer.

3.2.2 End-to-End Performance Evaluation

The end-to-end performance evaluation focuses on assessing the accuracy of the Retriever-Reader pipeline in predicting answers to the question from the TriviaQA dataset. Utilizing both the GTE and BM25 retrievers, together with the four promising reader models determined in the first experiment, a total of eight retriever-reader pipelines will be constructed.

The top_k-Retriever value is 1 across all these evaluations, meaning that the retriever only sends 1 document to the reader. For each pipeline, metrics such as Exact Match (EM), F1, and BERTScore will be calculated to determine the best retriever-reader combination.

Due to time constraints and consistent patterns observed across all eight pipelines, we selected four pipelines for further exploration. Specifically, we focused on those employing extractive readers. This involved varying the top_k-Retriever value from 1 to 7 with a step size of 1, and from 7 to 19 with a step size of 3. The top_k-Reader value was equal to 1 for all these experiments. Only the F1 score will be presented, as we consider it to be the most representative metric for demonstration.

The eight pipelines are:

- GTE with DeBERTa
- GTE with ELECTRA
- GTE with BLOOMZ
- GTE with Flan-T5
- BM25 with DeBERTa
- BM25 with ELECTRA
- BM25 with BLOOMZ
- BM25 with Flan-T5

3.3 Retriever-Ranker-Reader Pipeline Evaluation

When involving the ranker component in the pipeline, it will be placed between the retriever and the reader. The documents retrieved by the retriever are later passed to the ranker, which assesses and ranks these documents based on their relevance and quality. The ranker utilized for all subsequent experiments is ms-marco-MiniLM-L-6-v2 because this ranker has a reasonable model size and achieved high scores when compared to other cross-encoders models [2]. Following the ranking process, the top_k-Ranker value will specify how many of these documents will be forwarded to the reader model.

Retriever-Ranker-Reader Pipeline evaluation only concentrates on the end-to-end performance, however, three different experiments will be performed. The first one involves setting the top_k-Ranker and top_k-Reader to 1 while varying the top_k-Retriever, the values tested are between 1 and 50 with a step size of 3. The objective of this experiment is to identify the optimal top_k-Retriever value and proceed with it in the subsequent two experiments. Both the GTE and BM25 retriever models together with the single ranker model, will be employed in the pipeline. However, due to time constraints, only the two extractive reader models will be utilized. As a result, a total of four different pipelines will be evaluated.

The second experiment is similar to the end-to-end performance that was conducted on the retriever-reader pipeline. The distinction is adding a new component, a ranker model, with the top_k-Ranker value set to 1. Additionally, the top_k-Retriever value will be configured to the optimal value identified in the previous experiment (34).

The final experiment aims to investigate the impact of the top_k-Ranker value on the end-to-end performance. The top_k-Reader will be fixed to 1 and the top_k-Retriever will once again be set to the optimal value determined in the first experiment (34). The four pipelines

[2] https://www.sbert.net/docs/pretrained-models/ce-msmarco.html

utilized in this experiment are the same as those used in the initial experiment. The top_k-Ranker values tested will be from 1 to 7 with a step size of 1, and from 7 to 19 with a step size of 3.

3.4 Runtime of Individual Components

So far, only metrics related to how well each configured pipeline performs have been utilized. One big component when implementing a question-answering system is how well it will perform, however, another important aspect is also what components contribute the most when it comes to the runtime of the entire system. Here we will evaluate the runtime of the retriever, ranker, and reader components, this will be done using the TriviaQA dataset. The retrievers will be provided with each sample question within the TriviaQA dataset and the runtime will be measured accordingly to calculate an average runtime. Average runtime will be measured similarly for the ranker and readers, but will instead be provided with each golden question-document pair in TriviaQA, where each sample is one golden question with its corresponding golden document.

3.5 Evaluation of Open-Retrieval Question-Answering system on ConsafeQA

3.5.1 Recall evaluation

We first start by experimenting on ConsafeQA by performing recall evaluations on both our retrievers, similar to what we have done on TriviaQA, by varying the top_k-Retriever between 1-50. But the distinction is that we only use full document split, avoiding making and evaluating on smaller splits, because of the limited amount of documents available in ConsafeQA (179) compared to TriviaQA (486,933). For the same reason, we only perform exact document match, unlike TriviaQA where we also did an evaluation that takes into consideration documents containing the exact answer within the retrieved document. After that, we assess the retriever-ranker combinations to determine if the recall score improves when adding the ranker component. The top_k-Ranker varies from 1 to 50, while the top_k-Retriever is set to 50.

3.5.2 End-to-end performance

To evaluate the end-to-end performance on ConsafeQA, we adopt a methodology similar to what was used on TriviaQA for evaluating the retriever-ranker-reader pipeline. The top_k-Retriever is set to 50, while the top_k-Ranker and top_k-Reader are set to 1. The only difference is, that we also experiment with titles relevant to each document. An approach where the titles and passages are concatenated, before storing them in the document store. This is to see if information from titles in structured texts can provide an improvement in the end-to-end performance.

3.5.3 GTE model size

As mentioned earlier, there are three different sizes for the GTE model, small, base, and large. So far we have only used the base version. An interesting experiment would be to compare these retriever sizes and observe how the end-to-end performance differs among them. The pipeline for this experiment will follow the retriever-ranker-reader configuration, with the GTE model size varying among the three different configurations. The ranker component exists in the pipeline, although it's not necessary. This is to compare the results with the earlier end-to-end performance experiment. Additionally, we'll use the next best-performing reader from the previous end-to-end performance experiment to see if it will outperform the best pipeline. The top_k-Retriever is set to 50, while the top_k-Ranker and top_k-Reader are set to 1.

3.5.4 Fine-tuning a Ranker

Lastly, we experimented with fine-tuning the ranker used in all experiments (ms-marco-MiniLM-L-6-v2) on the ConsafeQA dataset, to see if adapting the ranker to the ConsafeQA domain would improve the overall end-to-end performance. The top_k-Retriever is set to 50, while the top_k-Ranker and top_k-Reader are set to 1. With ConsafeQA having a relatively small sample size, it was decided that for evaluation a 5-fold cross-validation would be performed to make sure the result generalizes well. We did not thoroughly experiment with the training setup, because of time limitations. For each run of the cross-validation, the ranker was fine-tuned with the following parameters, the epochs were 10 using a learning rate of 1e-5, weight decay of 1e-2, and a batch size of 4.

3.6 Additional Experiment Information

3.6.1 Documentstore and Retriever

For GTE, we use FAISS as a document store (Johnson et al. (2017)). Even though our dataset for TriviaQA can be considered large we still perform an exact search using cosine similarity between all embedding vectors for all experiments. This means that for FAISS, the only relevant hyperparameter when performing an exact match is that the index_factory utilized is "Flat", meaning it performs brute force. There exist numerous other vector databases that offer similar functionalities, but FAISS stood out as a user-friendly library with open documentation, which became a compelling reason for its usage. The reasons behind using brute-force search and cosine similarity is because of the relatively small quantity of documents within ConsafeQA and the desire to evaluate the system's optimal conditions with the resources available. For BM25 we use Elasticsearch as a document store, which means the BM25 algorithm is based on the Elasticsearch implementation.

3.6.2 Preprocessing documents

With the token limits of GTE and the storage limitations with larger datasets, some preprocessing was performed for both TriviaQA and ConsafeQA documents. The RAM limitation

was the biggest priority, with the token limit coming after. For both datasets, the documents are considered large in the number of words contained within them, thus all documents were split into smaller chunks with overlap between adjacent chunks. For extractive readers, they all use a sliding-window approach which results in long texts not being an issue. For generative readers, the token limits often exist, because their positional embeddings were trained on a specific token limit, however with enough memory the input sequence length can be increased, which could result in worse performance for the model.

TriviaQA

With TriviaQA consisting of 486,993 documents being long texts, having too small chunk sizes resulted in large memory limitations and runtime limitations, because of the exact search with GTE. Quantifying the limitations, memory requirements would exceed the utilized 32Gb that were available, and the runtime of finding a relevant document to one question would exceed 8+ seconds. Thus we split all TriviaQA documents into 800 word chunks with a 40 word overlap between chunks. A limitation with this size is that GTE and the ranker used in the experiments have a token limit of 512, this results in the chunks being truncated into the first 512 tokens when creating embedding vectors, thus this only affects the retriever and ranker part of the pipelines. This means that not the entirety of the context for each chunk is taken into account when retrieving relevant documents for GTE and ranker.

ConsafeQA

ConsafeQA documents are considered semi-structured texts, where each passage of text is always part of a section consisting of a title. With this in mind, the documents were split first based on their sections, extracting the text and title of each section. Each section is then split into smaller chunks of 200 words with a 30-word overlap. For each passage, cleaning was performed by removing any leading or trailing white spaces of each extracted passage, and enforcing that any continuous sequences of newlines were never more than two ("\n\n").

3.6.3 Answers

In the entirety of our experiments, we exclusively evaluate predictions by comparing them against the golden answers. If some questions have several valid golden answers, then the golden answer and predicted answer that results in the highest score is the one that contributes to the overall score. Which document the answer came from is not considered during this comparison, and as a result is a less stricter evaluation.

3.6.4 Metrics

For all experiments, when calculating the end-to-end performance metrics EM, F1, and BERTScore, we follow the official evaluation script for SQuAD (Rajpurkar et al. (2016)) when it comes to normalization rules, by lowering the text, removing punctuation's, articles and extra whitespace. The BERTScore is computed with the library *bert-score* using the *microsoft/deberta-large-mnli* model and only focusing on the BERTScore F1 metric. According to the **bert-score**

library which is the official support from the research paper [3], DeBERTa stands out as one of the best options and is recommended over the default *roberta-large* model.

3.6.5 Prompt

The prompt below is used in all experiments for the generative models.

> **Extract the answer to the question from the given context.**
> **Question: {query};**
> **Context: {documents};**
> **Answer:,**

3.6.6 Implementation

All experiments are implemented using the packages **sentence-transformers==2.2.2**, **farm-haystack==1.22.1**, **transformers==4.36.0** and **bert-score==0.3.13**, **scikit-learn==1.3.2**. The experiments are all run using an Nvidia T4 GPU with 32 GB of RAM available.

[3] `https://github.com/Tiiiger/bert_score`

Chapter 4
Result

In this section, we present the results obtained from the previously mentioned experiments. First, we start by evaluating several popular models on the MRQA dataset. Second, we evaluate different Retriever-Reader pipelines on TriviaQA verified dataset, this includes results on Retriever only and end-to-end performance. Third, we introduce a Ranker into the pipeline and show the performance. Lastly, we show the performance of several pipelines on the ConsafeQA dataset

4.1 Evaluation of Reader on MRQA dataset

4.1.1 Extractive Readers

Table 4.1 shows the results of DeBERTA, ELECTRA and RoBERTa on all individual Out-of-Domain development datasets of MRQA. For all the datasets, DeBERTA showed the highest scores for EM, F1 and BERTScore. The combination of dataset and extractive Reader that showed the highest and lowest scores, was RelationExtraction with DeBERTA for the former and RACE with RoBERTA as the latter.

Dataset	Metric	Readers		
		DeBERTa	**ELECTRA**	**RoBERTa**
BioASQ	EM	**55.98**	48.54	52.93
	F1	**71.26**	64.9	69.12
	BERTScore	**82.13**	78.20	80.63
DROP	EM	**47.37**	38.06	34.26
	F1	**57.31**	48.94	45.19
	BERTScore	**75.65**	70.28	67.71
DuoRC	EM	**53.30**	48.17	50.90
	F1	**62.98**	57.81	61.58
	BERTScore	**77.35**	74.48	76.36
RACE	EM	**36.80**	30.86	29.67
	F1	**49.91**	43.87	42.78
	BERTScore	**69.42**	65.35	65.19
RelationExtraction	EM	**77.88**	75.31	75.64
	F1	**88.20**	85.93	86.33
	BERTScore	**92.16**	90.81	91.08
TextbookQA	EM	**50.57**	48.50	42.98
	F1	**60.00**	57.70	50.42
	BERTScore	**77.00**	76.12	72.62
Average	EM	**53.65**	48.24	47.73
	F1	**64.95**	59.87	59.24
	BERTScore	**78.95**	75.87	75.60

Table 4.1: EM, F1 and BERTScore for different extractive readers on MRQA Out-of-Domain dev dataset (in %). The reader with the highest score for each metric in every dataset is marked in bold.

4.1.2 Generative Readers

Table 4.2 shows the results of BLOOMZ, Mistral, Flan-T5 and mT0 on all individual Out-of-Domain dev datasets of MRQA. For a majority of the datasets, Flan-T5 showed the highest scores for EM, F1 and BERTScore. The combination of dataset and generative Reader that showed the highest and lowest scores, was RelationExtraction with Flan-T5 for the former and RACE with Mistral as the latter.

		Readers			
Dataset	**Metric**	**BLOOMZ**	**Mistral**	**Flan-T5**	**mT0**
BioASQ	EM	55.98	27.93	**67.69**	55.39
	F1	65.81	55.96	**78.46**	66.81
	BERTScore	77.69	71.15	**87.29**	80.10
DROP	EM	40.05	19.23	**77.05**	42.32
	F1	46.90	40.54	**84.06**	49.89
	BERTScore	69.91	61.30	**90.59**	72.79
DuoRC	EM	58.23	35.44	**63.56**	58.03
	F1	66.69	53.11	**72.94**	66.14
	BERTScore	80.15	69.87	**83.48**	79.89
RACE	EM	37.10	18.55	**40.50**	37.24
	F1	**50.96**	37.51	47.75	49.99
	BERTScore	69.19	60.20	66.46	**70.19**
RelationExtraction	EM	72.93	41.96	**81.68**	75.71
	F1	82.38	62.27	**89.87**	84.33
	BERTScore	88.42	75.12	**93.61**	90.68
TextbookQA	EM	47.44	47.24	**53.03**	50.03
	F1	55.89	**60.36**	59.21	58.94
	BERTScore	73.13	74.62	76.00	**77.66**
Average	EM	51.95	31.72	**63.93**	53.12
	F1	61.44	51.63	**72.05**	62.68
	BERTScore	76.41	68.71	**82.91**	78.55

Table 4.2: EM, F1 and BERTScore for different generative readers on MRQA Out-of-Domain dev dataset (in %). The reader with the highest score for each metric in every dataset is marked in bold.

4.2 Evaluation of Retriever-Reader pipeline on TriviaQA

4.2.1 Evaluation of Retriever with robustness

Figures 4.1 and 4.2 show the result of two retrievers BM25 and GTE on different splits of the TriviaQA dataset as outlined previously. It can be seen that for subfigure (a) there exists the pattern of 50 000 Documents resulting in the highest Recall and 486 993 Documents resulting in the lowest Recall. It can also be observed that GTE confidently for subfigure (a) had higher Recall for all splits compared to BM25. Comparing subfigures (a) and (b) for both GTE and BM25, it can be seen that the difference between Recall scores is quite substantial.

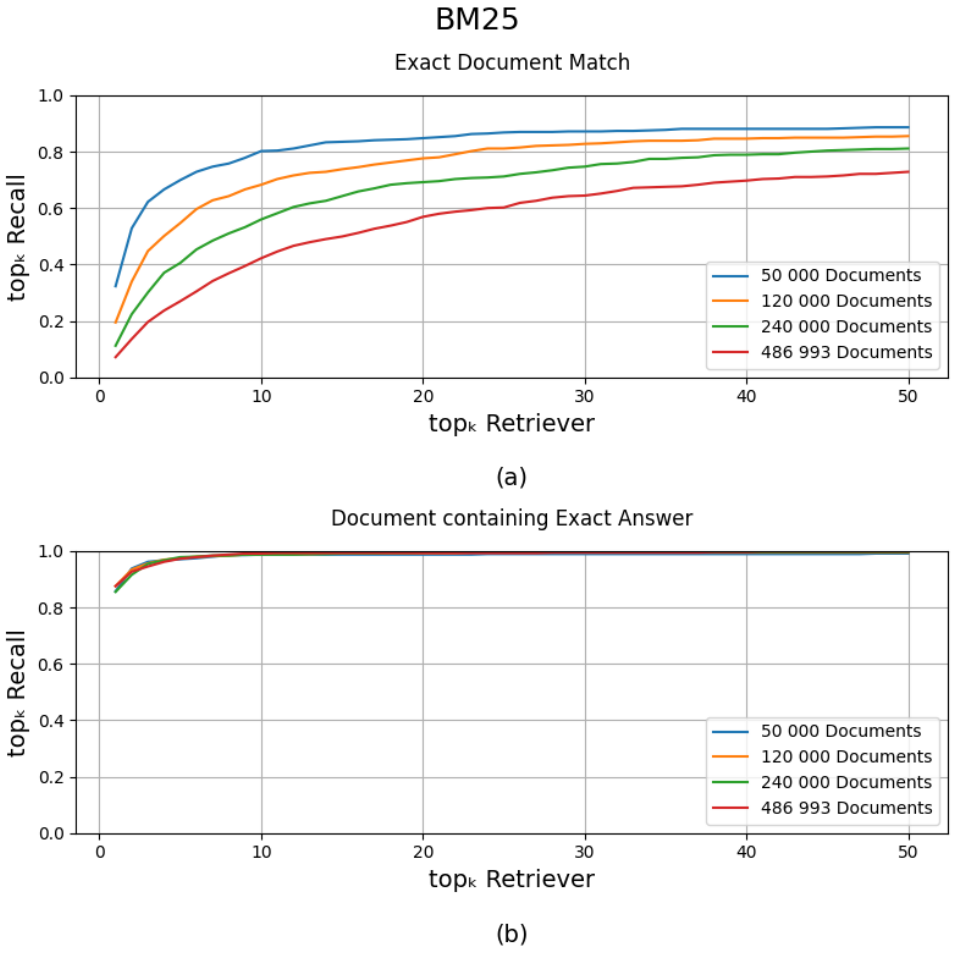


Figure 4.1: BM25 retriever recall against the number of documents retrieved on different TriviaQA splits where a document is considered relevant if (a) Exact Match of the gold document, and (b) Document containing the Exact Answer of gold answer.

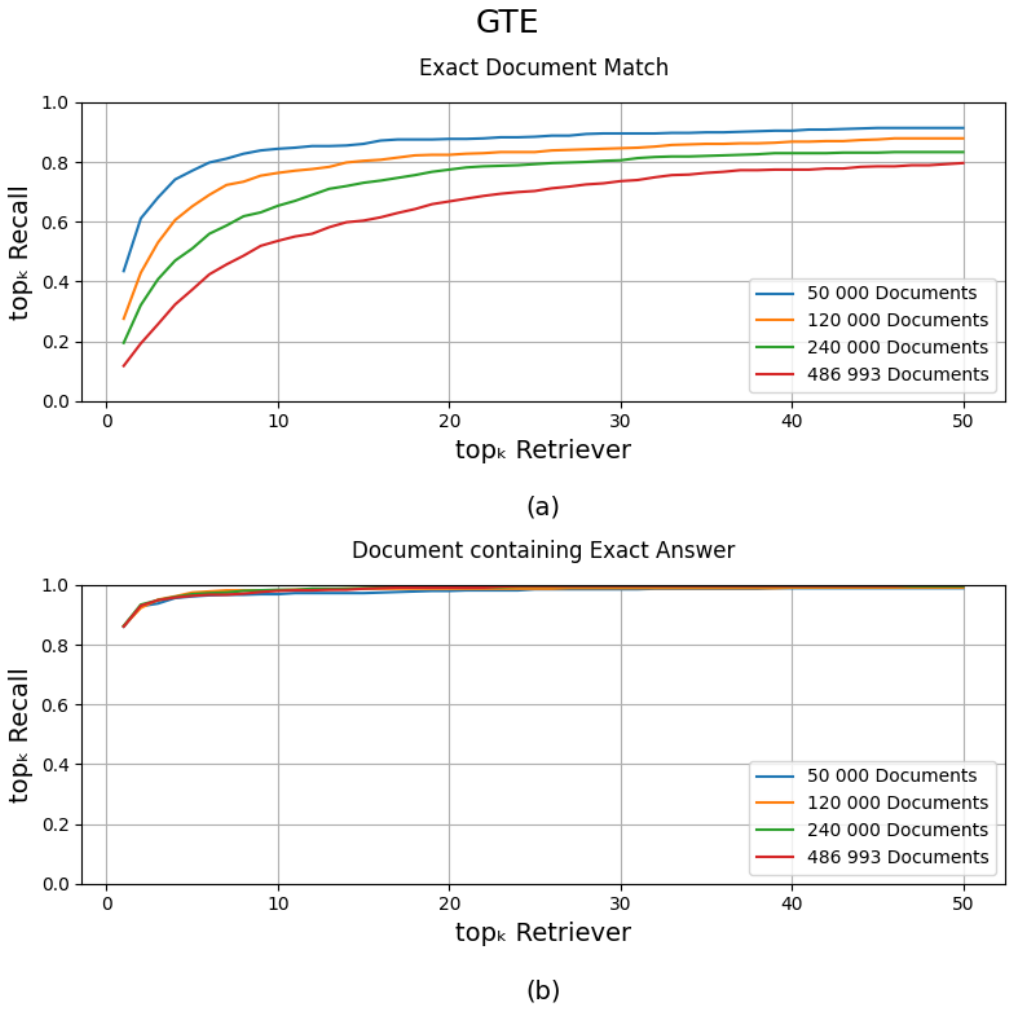


Figure 4.2: GTE retriever recall against the number of documents retrieved on different TriviaQA splits where a document is considered relevant if (a) Exact Match of the gold document, and (b) a Document containing the Exact Answer of the gold answer.

4.2.2 End-to-end performance

Table 4.3 shows the results of the entire Retriever-Reader pipeline on TriviaQA where the retrievers are combined with DeBERTA, ELECTRA, Flan-T5 and BLOOMZ. For all pipelines top_k-Retriever = top_k-Reader = 1. It can be seen that Flan-T5 with BM25 results in the highest EM, F1 and BERTScore of 68.8, 76.5 and 86.7 respectively. The lowest performing pipeline is BLOOMZ with BM25, resulting in EM, F1 and BERTScore of 51.0, 59.7 and 75.5 respectively. It can also be seen that for Flan-T5 and DeBERTA, the BM25 retriever was outperforming GTE, while for ELECTRA and BLOOMZ it was the opposite.

		Metrics		
Reader	**Retriever**	**EM**	**F1**	**BERTScore**
DeBERTA	GTE	60.7	69.2	81.8
	BM25	60.7	69.4	82.4
ELECTRA	GTE	56.7	65.3	80.0
	BM25	56.3	65.7	80.3
Flan-T5	GTE	68.1	75.2	86.0
	BM25	**68.8**	**76.5**	**86.7**
BLOOMZ	GTE	53.8	62.1	76.1
	BM25	51.0	59.7	75.5

Table 4.3: End-to-end EM, F1 and BERTscore for different Retriever-Reader pipeline configurations. All experiments are run with top_k-Retriever = top_k-Reader = 1. Using the entire TriviaQA dataset. The pipeline with the highest score for each metric is marked in bold

Figure 4.3 show the F1 score of four Retriever-Reader pipelines on TriviaQA where top_k-Reader = 1, and top_k-Retriever is varied in x-axis. Values tested in the x-axis are [1,7] with a step size of 1 and [7,19] with a step size of 3. It can be observed that for the four pipelines, there is an increase in F1 score consistently until a maximum that then starts to decline. There is also a pattern of the pipelines with BM25 having the maximum point on a lower top_k-Retriever value than their GTE counterpart.

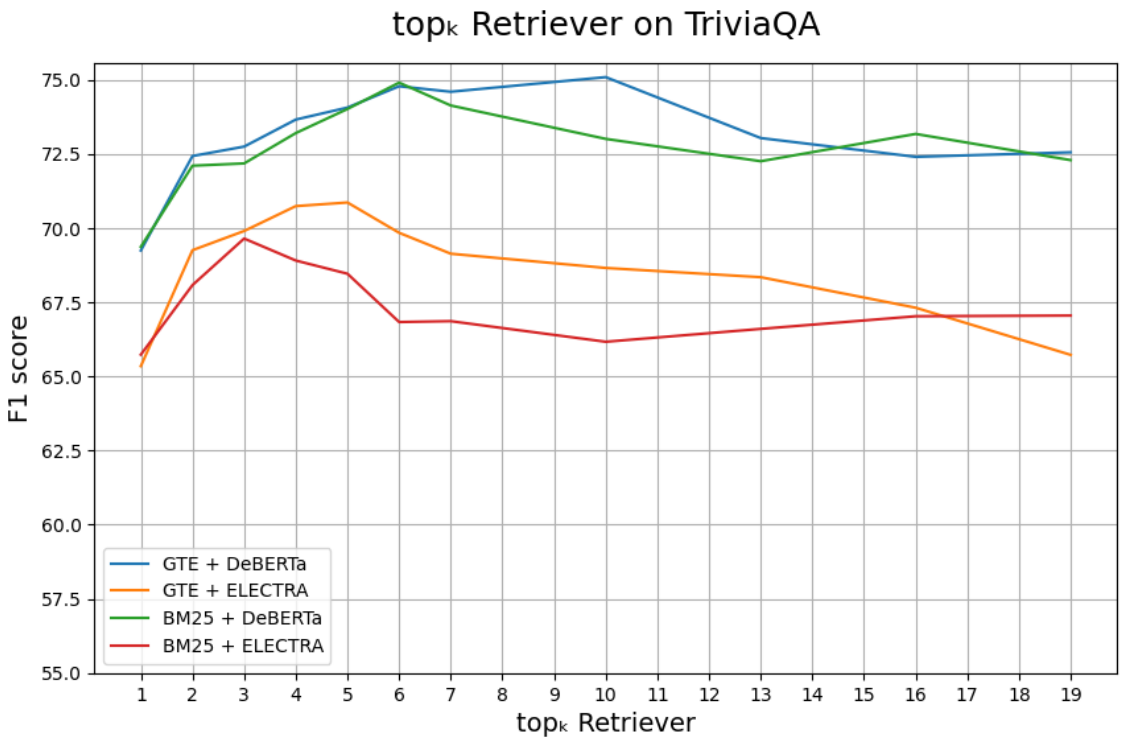


Figure 4.3: End-to-end F1 score when top$_k$-Retriever is varying, with four distinct Retriever-Reader pipelines on TriviaQA. Values tested are [1,7] with a step size of 1 and [7,19] with a step size of 3. For all experiments top$_k$-Reader = 1.

4.3 Evaluation of Retriever-Ranker-Reader pipeline on TriviaQA

4.3.1 End-to-end performance

Figure 4.4 show the F1 score of four Retriever-Ranker-Reader pipelines on TriviaQA where top_k-Ranker = top_k-Reader = 1, and top_k-Retriever is varied in x-axis. Values tested in the x-axis are [1,50] with a step size of 3. The results show that the pipelines with GTE as a retriever benefit more from introducing a Ranker and increasing the top_k-Retriever. The same pattern can be seen from all pipelines, where the highest increase in F1 occurs when top_k-Retriever has a value of between 1-7 and where the F1 score shows no significant change for larger top_k-Retriever values.

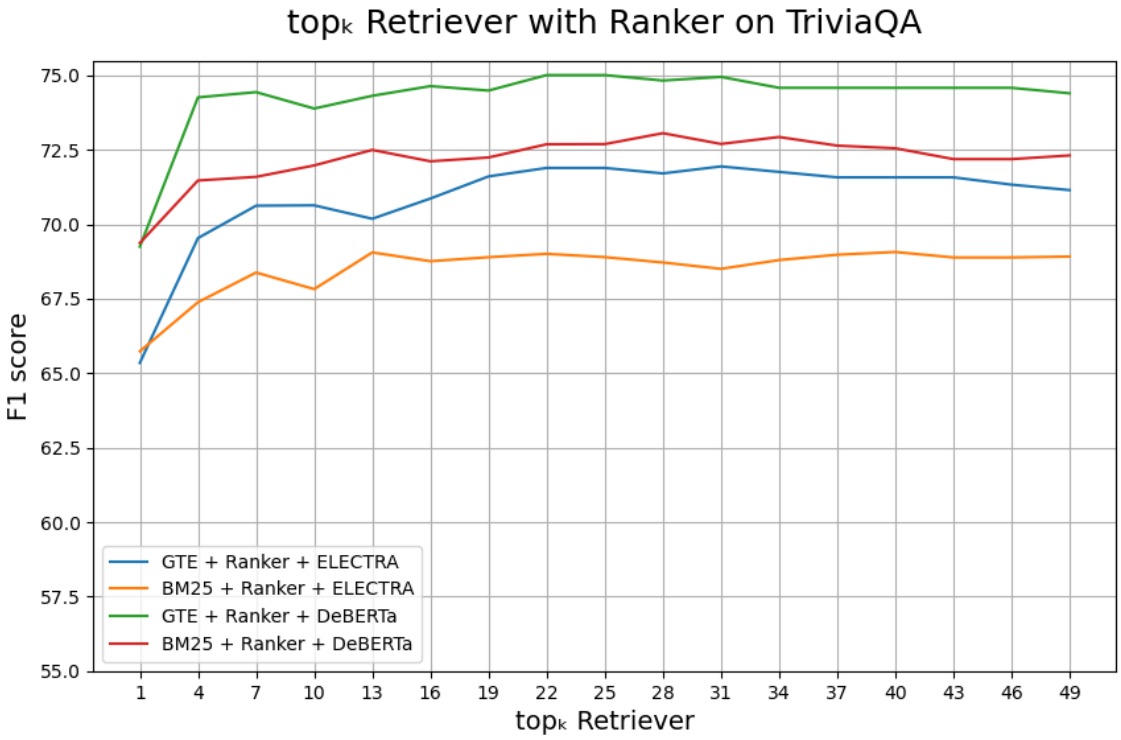


Figure 4.4: End-to-end F1 score when top_k-Retriever is varying, with four distinct Retriever-Ranker-Reader pipelines on TriviaQA. Values tested are [1,50] with a step size of 3. For all experiment top_k-Ranker = top_k-Reader = 1.

Table 4.4 shows the results of the entire Retriever-Ranker-Reader pipeline on TriviaQA. For all pipelines top_k-Retriever = 34 and top_k-Ranker = top_k-Reader = 1. It can be seen that for all pipelines with the ranker, Flan-T5 with GTE results in the highest EM, F1 and BERTScore of 73.94, 81.96 and 89.63 respectively. The lowest performing pipeline with the ranker is BLOOMZ with BM25, resulting in EM, F1 and BERTScore of 53.94, 62.09 and 75.54 respectively. It can also be seen that all pipelines with GTE as a retriever are outperforming their counterparts with BM25 as a retriever.

		Metrics		
Reader	**Retriever**	**EM**	**F1**	**BERTScore**
DeBERTa	GTE	65.32	74.58	85.08
	BM25	63.85	72.93	84.16
ELECTRA	GTE	61.65	71.76	83.25
	BM25	59.27	68.80	81.84
Flan-T5	GTE	**73.94**	**81.96**	**89.63**
	BM25	71.74	79.62	88.46
BLOOMZ	GTE	55.78	64.01	76.77
	BM25	53.94	62.09	75.54

Table 4.4: End-to-end EM, F1 and BERTscore for different Retriever-Ranker-Reader pipeline configurations. All experiments are run with top_k-Retriever = 34, top_k-Ranker = top_k-Reader = 1. Using the entire TriviaQA dataset. The pipeline with the highest score for each metric is marked in bold

Figure 4.5 show the F1 score of four Retriever-Ranker-Reader pipelines on TriviaQA where top_k-Retriever = 34, top_k-Reader = 1 and top_k-Ranker is varied in x-axis. Values tested in the x-axis are [1,7] with a step size of 1 and [7,19] with a step size of 3. The result for all pipelines shows a pattern, where there is a consistent increase in F1 until a maximum that then starts to decline. The maximum for all pipelines occurs when top_k-Ranker value is between 1-5.

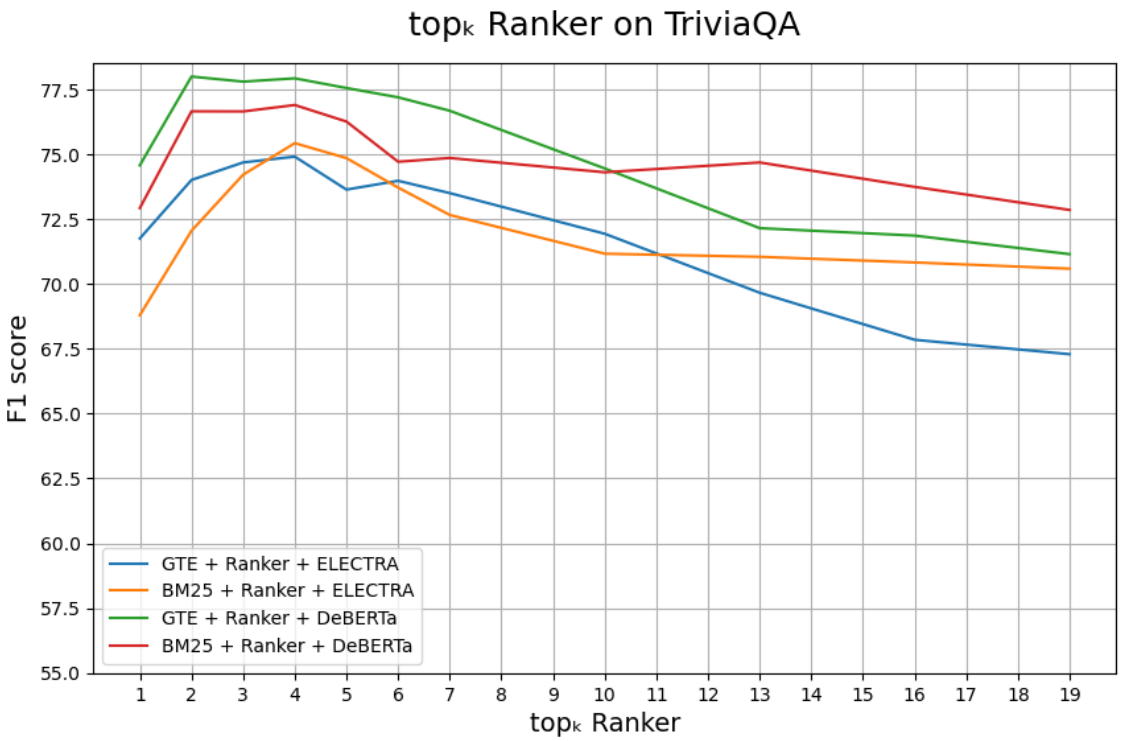


Figure 4.5: End-to-end F1 score when top_k-Ranker is varying, with four distinct Retriever-Ranker-Reader pipelines on TriviaQA. For all experiments top_k-Retriever = 34, top_k-Reader = 1. Values tested are [1,7] with a step size of 1 and [7,19] with a step size of 3

4.4 Runtime of Individual Components on TriviaQA

Figure 4.6 contains three subfigures where each shows the average runtime of an individual component. The subfigure for Retriever shows that GTE has a higher runtime compared to BM25. Reader runtime for answering a question given one document from lowest to highest was ELECTRA, DeBERTa, Flan-T5 and BLOOMZ.

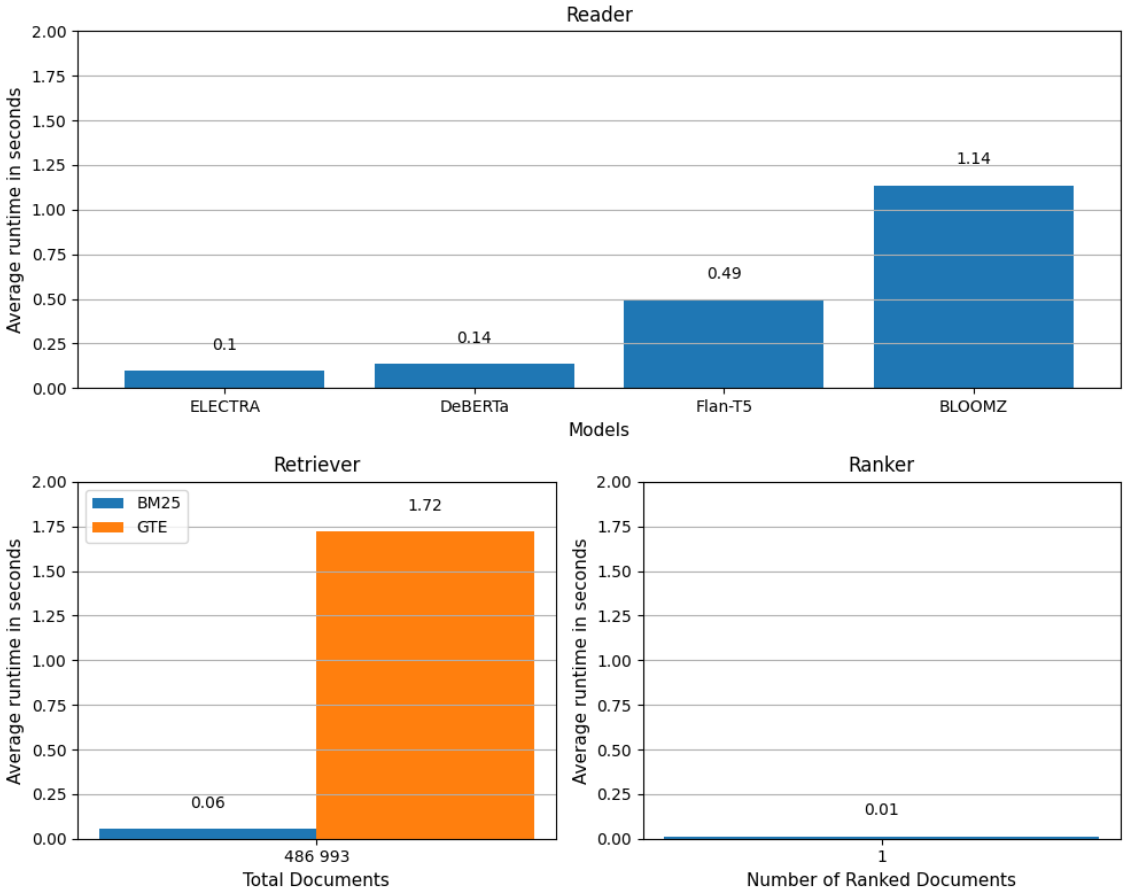


Figure 4.6: Runtime for individual components of an Open-Retrieval Question-Answering system

4.5 Evaluation of Open-Retrieval Question-Answering system on ConsafeQA

4.5.1 Recall Evaluation

Figure 4.7 contains two subfigures (a) and (b), that show the result of top$_k$ recall. It can be seen that for subfigure (a), BM25 achieves higher recall for all values of top$_k$-Retriever. For Subfigure (b), top$_k$-Retriever = 50 top$_k$-Ranker was varied in x-axis for all pipelines. It can be seen that BM25 (with Titles) + Ranker achieved the highest recall for all top$_k$-Ranker values. What is also shown is that concatenating relevant titles for each passage resulted in an overall increase in recall for both BM25 and GTE.

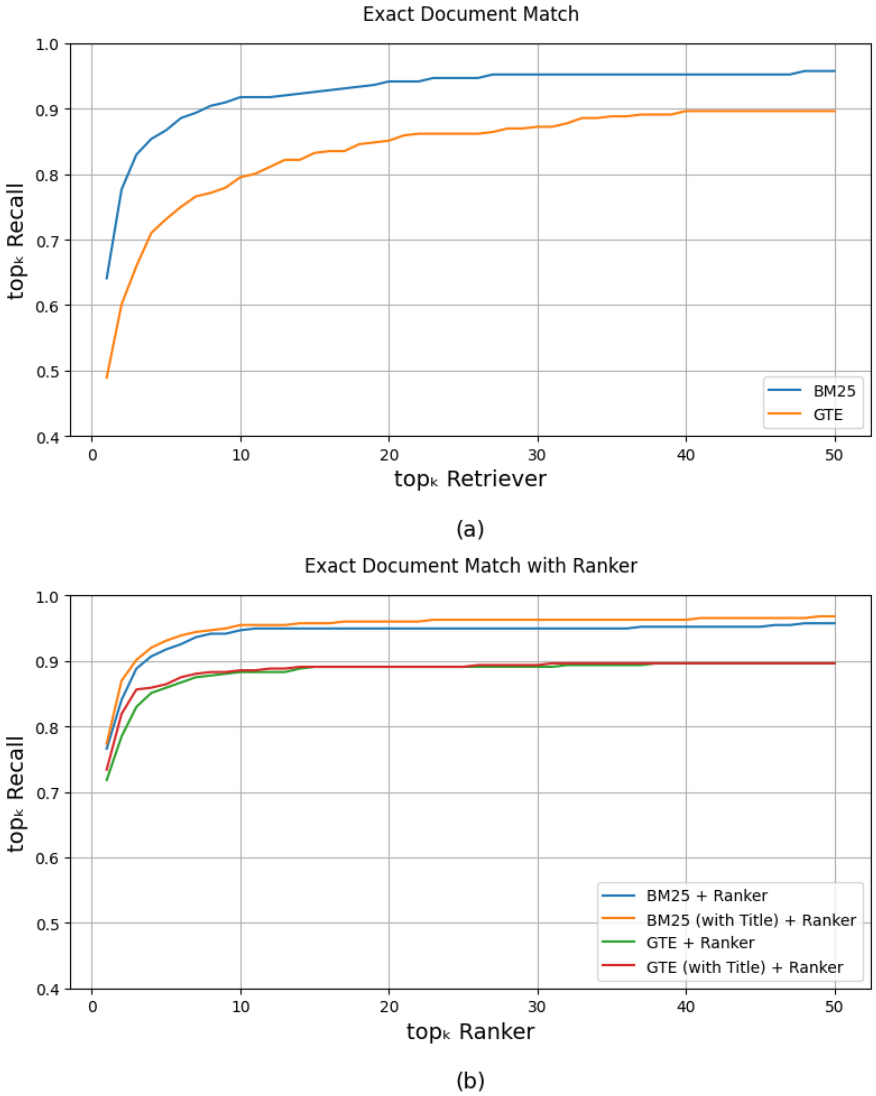


Figure 4.7: Retriever top-k recall against the number of documents retrieved on ConsafeQA where a document is considered relevant if it is Exact Match of gold document, (a) Retriever only, and (b) Retriever + Ranker.

4.5.2 End-to-end performance

Table 4.5 shows the results of the entire Retriever-Ranker-Reader pipelines on ConsafeQA. For all pipelines top_k-Retriever = 50 and top_k-Ranker = top_k-Reader = 1. It can be seen that the best retriever for all readers was BM25 + Titles. The pipeline that resulted in the highest EM, F1 and BERTScore was DeBERTa with BM25 + Titles, with 55.85, 77.31 and 84.02 respectively. The pipeline with the lowest scores was BLOOMZ with GTE, resulting in EM, F1 and BERTScore of 20.74, 42.13 and 53.69 respectively. It can also be seen that introducing Titles, results in higher EM, F1 and BERTscore for all pipelines.

Reader	Metric	Retriever			
		BM25	**BM25 + Titles**	**GTE**	**GTE + Titles**
DeBERTa	EM	55.59	**55.85**	51.86	53.19
	F1	76.62	**77.31**	72.38	73.92
	BERTScore	83.70	**84.02**	81.30	82.14
ELECTRA	EM	49.47	50.27	45.48	47.61
	F1	71.36	73.30	67.22	69.77
	BERTScore	80.26	81.51	77.81	79.33
Flan-T5	EM	50.53	50.80	46.28	47.34
	F1	74.11	74.66	69.83	70.89
	BERTScore	81.55	82.04	78.94	79.84
BLOOMZ	EM	23.40	23.67	20.74	22.07
	F1	44.77	45.19	42.13	43.33
	BERTScore	55.56	56.24	53.69	54.88

Table 4.5: End-to-end EM, F1 and BERTscore for different Retriever-Ranker-Reader pipeline configurations. All experiments are run with top_k-Retriever = 50, top_k-Ranker = top_k-Reader = 1. Using ConsafeQA dataset. The pipeline with the highest score for each metric is marked in bold.

4.5.3 GTE model size influence on performance

Figure 4.8 show the F1 score of three Retriever-Ranker-Reader pipelines on ConsafeQA where top$_k$-Retriever = 50, top$_k$-Ranker = top$_k$-Reader = 1. The reader used for all pipelines is Flan-T5, and the retriever is the varied sizes of GTE. The sizes for GTE$_{small}$, GTE$_{base}$ and GTE$_{large}$ are 30M, 110M, 330M parameters respectively. The exact F1 score from smallest to largest sized GTE model is 69.76, 69.83 and 69.97. The results show that an increased size of GTE does give very small improvements in performance.

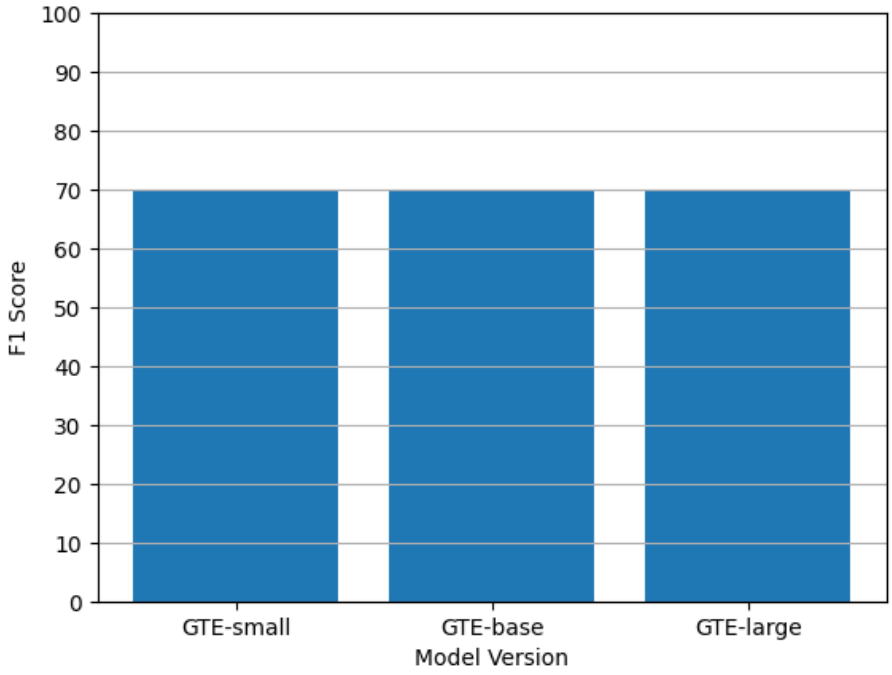


Figure 4.8: End-to-end F1 score on three Retriever-Ranker-Reader pipelines on ConsafeQA where top$_k$-Retriever = 50, top$_k$-Ranker = top$_k$-Reader = 1. The varying component is the model size of the GTE retriever

4.5.4 Fine-tuned Ranker

Table 4.6 shows the result of 5-fold cross-validation on the Retriever-Ranker-Reader pipelines on ConsafeQA, where the ranker is fine-tuned as previously stated. For all pipelines top$_k$-Retriever = 50 and top$_k$-Ranker = top$_k$-Reader = 1. The EM, F1 and BERTScore for the pipelines with the original ranker are the same as from table 4.5. The pipeline that resulted in the highest EM, F1 and BERTScore was DeBERTa with BM25 + FR, with 58.49, 79.68 and 85.38 respectively. It can be seen that for all pipelines the fine-tuned ranker compared to the original ranker, achieves higher EM, F1 and BERTScore.

Reader	Metric	Retriever + Ranker			
		BM25 + OR	**BM25 + FR**	**GTE + OR**	**GTE + FR**
DeBERTa	EM	55.59	**58.49**	51.86	55.04
	F1	76.62	**79.68**	72.38	75.71
	BERTScore	83.70	**85.38**	81.30	83.06
ELECTRA	EM	49.47	51.84	45.48	48.12
	F1	71.36	74.36	67.22	70.50
	BERTScore	80.26	81.77	77.81	79.50
Flan-T5	EM	50.53	52.66	46.28	48.94
	F1	74.11	75.90	69.83	72.02
	BERTScore	81.55	82.57	78.94	80.17
BLOOMZ	EM	23.40	24.46	20.74	22.86
	F1	44.77	45.47	42.13	43.21
	BERTScore	55.56	56.43	53.69	54.95

Table 4.6: End-to-end EM, F1 and BERTscore on Retriever-Ranker-Reader pipeline configurations. All experiments are run with top_k-Retriever = 50, top_k-Ranker = top_k-Reader = 1. Using ConsafeQA dataset. Metrics calculated through 5-fold cross-validation. The pipeline with the highest score for each metric is marked in bold. OR = Original Ranker, FR = Fine-tuned Ranker

Chapter 5
Discussion

In this chapter, we will discuss the results of the experiments performed. The goal is to provide a deeper understanding of our findings and comment on the work that was discarded for various reasons as well as the methodology. In Section 5.1 we will discuss everything surrounding the results and if any conclusions can be drawn from them. Section 5.2, will discuss the challenges and flaws with the different evaluation metrics. Section 5.3, focuses on the work that was discarded from this paper and the reasons behind it. In Section 5.4, we will highlight and discuss the ethical aspects as well as the environmental effects of these types of systems. Lastly, Section 6.1, will discuss the possible future work that can performed to expand on this work.

5.1 Result Findings

5.1.1 Reader Only

The evaluation of the MRQA dataset aimed to assess the performance of various reader models, both extractive and generative. Among the extractive readers, including DeBERTa, ELECTRA, and RoBERTa, DeBERTa shows the best performance based on the evaluation metrics employed as shown in Table 4.1. An explanation for this result might lie in the advancements introduced by Electra. As discussed in the literature review, Electra implemented a pre-training technique known as Replaced Token Detection (RTD), addressing issues found in older large language models (LLMs) such as RoBERTa. Additionally, DeBERTa, being the most recent model, draws inspiration from Electra, as highlighted in the literature review. DeBERTa aims to refine the approaches introduced by Electra while addressing some issues. Furthermore, DeBERTa has the largest number of parameters among these models, as indicated in Table 4.1, which could also contribute to its superior performance.

The generative readers showed varying levels of performance, and Flan-T5 demonstrated su-

perior results among the generative readers. The top-performing reader overall was Flan-T5. On average, generative models, except for Mistrial, outperformed extractive models. One possible explanation is the greater parameter count in generative models compared to the extractive ones.

An issue that was observed when using Mistral is that it tends to generate more text compared to other generative reader models, table 5.1 shows some examples from the MRQA dataset. This could be attributed to the fact that this model was initially too large but has been quantized to reduce its size. However, this size reduction may increase the risk of hallucinations and the output of incorrect information, which contributes to the fact that Mistral achieved very lower scores compared to the other models 4.2. Another reason could be that it is sensitive to the prompt being used, and it may need a prompt that is more suited to it.

Question	Golden Answer	Model	Generated Answer
What type of enzyme is peroxiredoxin 2 (PRDX2)?	antioxidant	Mistral	Peroxiredoxin 2 (PRDX2) is an antioxidant enzyme involved in protein and lipid protection against oxidative injury and in cellular signaling pathways regulating apoptosis.
		Flan-T5	antioxidant enzymes
		BLOOMZ	antioxidant enzymes
		mT0	antioxidant
Who had a higher average of yards per caught pass, Clayton or Nathan?	Clayton	Mistral	Nathan had a higher average of yards per caught pass, 83/10 = 8.3 yards per caught pass, while Clayton had 92/6 = 15.3 yards per caught pass.
		Flan-T5	Nathan
		BLOOMZ	Nathan
		mT0	Clayton

Table 5.1: Answers from different generative reader models on some questions from the MRQA dataset

5.1.2 Retriever-Reader

We see clearly in the top graph of figure 4.1 and figure 4.2 that the retriever performs better as the number of documents in the knowledge base decreases. This improvement happened because, in the smaller splits, fewer documents are disturbing the retriever. However, the overall top_k recall scores for both these graphs are notably low, particularly when considering low top_k retriever values. If we observe when the top_k-retriever is equal to 1 for the entire TriviaQA split in Figures 4.1 and 4.2, we notice that the top_k recall is around 0.1 for both GTE and BM25. This contradicts the results found in Table 4.3, which shows the performance for the overall system, resulting in high scores of EM, F1, and BERTScore for all configurations compared to the top_k recall being low. The conclusion for this disparity arises from both the nature of the TriviaQA dataset and the decision to combine the web and wiki versions,

which contain many similar documents where the correct answers for many questions appear in multiple documents. Therefore, during the evaluation process, when questions from TriviaQA are posed, the retriever returns documents that it deems relevant, even though only one of them is correct according to the TriviaQA dataset. As a result, the top_k recall is very low. On the other hand, in terms of end-to-end performance, since correct answers are found in multiple places, there is a higher probability for the reader to find the correct answer among the retrieved documents, therefore the scores tend to be higher. The bottom graphs in Figure 4.1 and Figure 4.2 consider a document correct if it contains the golden answer, which supports our argument as both graphs achieve a top_k recall of approximately 0.85 when the top_k retriever is 1. It is important to note that the results from lower graphs may be overly optimistic because they only consider a document correct if it contains the golden answer. In such cases, the document context may be irrelevant but still considered correct. Therefore, the actual representative top_k recall value exists between these two graphs.

Based on that, we cannot definitively determine how well a retriever performs by itself on TriviaQA. Therefore, it is not possible to draw a conclusive assessment from the evaluation of the retriever with robustness experiment on TriviaQA. On the other hand, the end-to-end performance provides a more comprehensive evaluation of the system's efficacy, as it considers the retriever's output combined with the reader's ability to accurately process and interpret the retrieved information.

Although GTE is considered to be more advanced than BM25 and utilizes embeddings, we can clearly see in Table 4.3 that both of them achieve very similar results. One explanation for this behavior could be that GTE has a token limit of 512, while TriviaQA was split into 800-word chunks, and therefore only the first 512 words of the 800 words were used by the retriever. This limitation might prevent the GTE retriever from performing at its best.

Furthermore, 4.3 shows that the optimal top_k value for achieving the maximum F1 score differs between DeBERTa and ELECTRA. This suggests that there isn't a universal hyperparameter that guarantees the best performance across all models, rather the optimal hyperparameter choice depends entirely on each model.

It's important to note that the TriviaQA dataset used in these experiments is considered small, containing only 545 questions. Therefore, it is challenging to draw definitive conclusions when using such a dataset.

5.1.3 Ranker

The introduction of a ranker into the pipeline configurations has shown a consistent improvement in overall performance. This is because of the effective reranking of documents, ensuring that relevant documents end up higher in the ordering of the retrieved documents from the retriever.

Figure 4.4 shows the capability of a ranker, handling an increased volume of input documents, and avoiding confusion while still effectively ranking relevant documents higher. Thus maintaining a stable F1 score. It can also be observed that the pipelines with GTE benefited the

most from incorporating a ranker. An explanation for this could be that because both the retriever and ranker understand the semantics around the question and documents they pair better together. It could also be that GTE is better overall at retrieving a larger amount of relevant documents compared to BM25, meaning documents that are initially ranked at 30-50 might still be somewhat relevant which the ranker can take advantage of during the reranking process.

When comparing figure 4.5 to 4.3, where the reader goes through multiple documents to provide an answer, the impact of the Ranker becomes interesting. The maximum increase in F1 score for all pipelines becomes closer when a ranker is employed, in the case of DeBERTa pipelines the maximum occurs earlier, which is quite logical when the relevant document is ranked higher more frequently. An interesting observation is the odd behavior of BM25 with ELECTRA, where the maximum F1 score surpasses that of its GTE counterpart. It is hard to pinpoint what might be the reason behind this, we believe that if a substantially larger dataset was utilized then this might not occur, thus further readers would need to be tested to see if this behavior is an outlier for ELECTRA.

5.1.4 Practical Aspects

In real-world applications, achieving optimal performance is not the only goal. Practical considerations determine that the system must also be efficient and manageable within the constraints of available hardware. Thus, decisions regarding system speed and hardware limitations are important to the design process, ensuring that the system meets both performance and practicality requirements.

The bottom left subfigure of Figure 4.6 illustrates the average runtime for the retrievers. The GTE retriever consumes more time compared to the BM25. This increase in time consumption is because the GTE retriever requires more time when dealing with more documents, as it needs to compute more embedding comparisons. It is worth noting that utilizing ANN (Approximative Nearest Neighbor) search instead of brute force could give improved runtime performance, at the cost of worsening the precision of finding relevant documents.

Additionally, the average runtime of the ranker initially shows that is relatively fast compared to the other components. This stems from the inherent function of a ranker, if it was tasked with ranking all 486,993 documents within TriviaQA, it would significantly underperform compared to the other components. However, when scaled down to rank fewer than 5 documents, the ranker demonstrates a runtime that is faster compared to all other components.

Furthermore, we observe that reader models with fewer parameters generally require less runtime. There is also a significant gap between extractive and generative readers. Except for the model parameter size, one reason for this disparity could be that generative readers need to generate one token at a time, which in turn generally consumes more time than predicting the start and end tokens of an answer.

5.1.5 ConsafeQA

A clear difference between the performance on the benchmark datasets and ConsafeQA is the components that produced the highest scores. It can be seen from figure 4.7, that BM25 outperforms GTE by a considerable amount, this stays the same even while introducing a Ranker. The reason behind this based on the result and the data analysis performed on ConsafeQA as shown in the literature review, is most likely because of the system-specific jargon used within the documents that GTE is not trained on to understand the relationship between these unique words.

It can also be seen when comparing the end-to-end performance from benchmark datasets and ConsafeQA, that extractive readers seem to perform the best for ConsafeQA with DeBERTa having the highest scores with BM25 + Titles. The dataset being relatively small, and annotated in a way that benefits having concise and extractive answers, we can not confidently claim that extractive readers have an advantage over generative readers. It can also be seen that the score difference between EM and F1 is approximately twice as large for ConsafeQA systems compared to TriviaQA. Furthermore from TriviaQA results to ConsafeQA, it can be observed that EM has a decrease of around 26-63%, F1 a decrease of 30% at worst and an increase of 2% at best, and lastly BERTScore a decrease of around 1-30%. We believe the reason behind the large difference between the metrics, lies in the nature of the ConsafeQA dataset. In ConsafeQA there exist instances where two completely different questions might have answers where parts of them are similar. Meaning when a question is asked and the wrong document is retrieved it might still provide an answer that is incorrect but partial words within the answer are correct. It can also be seen from the literature review, that the average word length of answers for ConsafeQA is around four times larger compared to TriviaQA. Meaning there are more instances where a word could be missing from the predicted answer. This would explain why EM decreases substantially more than the other metrics.

It is important to note that the mentioned worst-case decreases in all metrics stem from the BLOOMZ model. BLOOMZ performs significantly worse than the other readers on ConsafeQA compared to benchmark datasets. From our investigation, we concluded that BLOOMZ was hallucinating significantly and in several cases when provided a question was generating 0 tokens. This is most likely because of how BLOOMZ is a model trained on multilingual data, as mentioned in the literature review, and is being evaluated on ConsafeQA which is a more difficult task because of the domain being outside of BLOOMZ training capabilities.

From observing both figure 4.7 and table 4.5, we can see that introducing titles into the contexts through concatenation did produce an increase in performance for all configurations. However, the increase is very negligible with less than 1.0 F1 increase, and with the given dataset, this can be as much as answering one more sample correctly.

Qualitative Reader Comparison

The score results on ConsafeQA can be difficult to interpret, thus it is beneficial to examine the predicted and generated answers to understand the challenges the readers encounter

when handling this type of dataset.

Table 5.2 shows five samples from four pipelines with BM25 + Ranker being identical and Reader varying. It can be seen that from the first three rows, BLOOMZ and ELECTRA seem to struggle with understanding that the answer encompasses more than what they produce. In these samples, it is shown that they are missing the start or end of the complete answer, the reason behind this can be difficult to find, however, it could be related to certain unique characteristics of the dataset. For example, very often locations of certain aspects are described by menu choices with the symbol ">", indicating from left to right where to proceed, these models might find it difficult to correlate this symbol as a guidance tool. In the case of the first and last row, it can also be seen that BLOOMZ and ELECTRA can have difficulties understanding certain jargon, for example, that "Location Blocks" is a name for a grid. An interesting observation with Flan-T5 as can be seen in row four, was that in certain instances it tried to answer the question too well, however in the context of the dataset performing the action of "Release trip" and "book transport" is considered one step, in a list of enumerated steps.

These samples were chosen to reflect the most common occurrences that were not always confidently incorrect based on assessing around 150 samples that had at least one reader produce an exact match of zero. Thus we can not draw a confident conclusion that these types of challenges only affect certain readers, because of the small sample size. However, it can be an indication of what readers in general struggle with when evaluated on ConsafeQA.

Question	Golden Answer	Reader	Generated/Predicted Answer
What is the available balance of an article in Astro?	the sum of the contents in all not blocked StUnits	DeBERTa	the sum of the contents in all not blocked StUnits
		ELECTRA	the sum of the contents in all not blocked StUnits
		Flan-T5	the sum of the contents in all not blocked StUnits
		BLOOMZ	the sum of the contents
Where can an overview of the pallet pool log records be viewed?	O2 > Maintenance > 12. Pallet Pool > F2 Pallet Pool	DeBERTa	O2 > Maintenance > 12. Pallet Pool > F2 Pallet Pool
		ELECTRA	F2 Pallet Pool
		Flan-T5	In O2 > Maintenance > 12. Pallet Pool > F2 Pallet Pool
		BLOOMZ	F2 Pallet Pool
Where are the Crane Multi-Cycles defined?	M2 Setup > Warehouse - Logic > Multi Cycle	DeBERTa	M2 Setup > Warehouse - Logic > Multi Cycle
		ELECTRA	M2 Setup
		Flan-T5	M2 Setup > Warehouse - Logic > Multi Cycle
		BLOOMZ	M2 Setup > Warehouse - Logic
What is the first step when using manual booking?	Release trip and book transport	DeBERTa	Release trip and book transport
		ELECTRA	Release trip and book transport
		Flan-T5	Release trip
		BLOOMZ	Release trip and book transport
In what grid can the existing blockings of a location be seen?	Location Blocks	DeBERTa	Location Blocks
		ELECTRA	detail grid
		Flan-T5	Location Blocks
		BLOOMZ	you can see existing blockings

Table 5.2: Sampled question-answer pairs from ConsafeQA where at least one reader produced a 0 Exact Match answer. The samples are from each readers respective BM25 + Ranker + Reader pipeline.

5.2 Evaluation Metrics

Within our work, we have utilized a range of different evaluation metrics, each with its unique characteristics. In this section, we want to discuss more of the flaws in each of the metrics that we have encountered during our work.

Exact Match and F1, are lexical metrics that have the flaw of not being able to provide an appropriate score for predicted answers that are synonyms of the golden answer. For F1 there also exists the aspect of every word within the predicted answer having the same amount of importance when the metric is calculated. For example, if the predicted answer is *Pacific Ocean* and the golden answer is *Atlantic Ocean*, depending on the question, a human evaluation would be higher or lower because of the importance of the words in relation to the question. In the case of Exact Match (EM), it has been shown to be a strict metric that is unforgiving to partial correctness in predicted answers. Thus, these metrics may not always be suitable for more complex or longer forms of answers.

BERTScore, from our observations has shown the flaw of often having a bias towards providing not a low enough score for predicted answers that are incorrect. This can be because of several reasons but we believe that it mainly pertains to two factors, the dependency on pre-trained models and the interpretability of the score. BERTScore is provided by the underlying pre-trained model which means its comparisons are affected by the biases in the models. Interpreting the score can be difficult for a human when it originates from the training of large amounts of text. For example, if the predicted answer was incorrect but written in the same style and manner as the golden answer, the score from a human perspective might not be exactly zero. This is why we believe that BERTScore often deviates from a score of zero.

We are unable to make confident claims about which metrics are better than the other because of the limited experimentation in this area. However, we do think that several evaluation metrics should be taken into account when trying to understand the performance of a question-answering system.

5.3 Work Considered and Discarded

Various approaches were explored that augmented the pipelines to increase the performance. However, after thorough experimentation and consideration, we opted not to pursue certain avenues further. Two specific aspects that underwent initial experimentation but were afterward discarded from our work are discussed below:

Experimentation with Query Expansion. We experimented with query expansion techniques, specifically focusing on a similar approach to HyDE (Hypothetical Document Embeddings). Where given a query, Flan-T5 is used to generate a hypothetical document, that is subsequently used to retrieve relevant documents instead of the original question. However, after experimenting initially on TriviaQA, we saw that the end-to-end performance was significantly impacted negatively, for all of our system configurations. There can be several factors that could be the reason for this result, however, we believe that the largest reason is

because of the size of the LLM used to generate the hypothetical document. In the case of other literature, models used usually had extremely large amounts of parameters, and they were evaluated in a setting where the k most relevant document retrieved was in the 1000+ ranges.

Experimentation with a Merged Retriever of Titles and Contexts. Another aspect of experimentation was with a merged retriever on ConsafeQA, which simultaneously considered titles and contexts during the retrieval process. We mainly utilized a retriever that searched for relevant titles and contexts independently of each other based on the given question and returned the scores. The objective was to evaluate the effects of merging scores, where contexts and titles belonging to the same original passage would have their scores merged. Despite the initial experimentation, we decided not to proceed with this approach due to skewed weights. Particularly where the weight hyperparameter had an extreme bias toward favoring the scores of the context. In our experimentation, we were never able to get a higher end-to-end performance compared to only retrieving contexts. This is most likely because the titles by themselves do not contain enough distinct information, meaning several titles are very similar. However, as discussed in section 5.1, combining titles and contexts through concatenation produced better results. We do believe that if there was an effort in the naming of titles to be more distinct for each document, then this approach has potential.

5.4 Ethical Aspects

5.4.1 Environmental effects

Large language models, due to their size, consume substantial amounts of energy during training, fine-tuning, and evaluation processes. With their growing popularity, using these massive models has become increasingly common. However, this accessibility raises concerns about environmental sustainability and conflicts with global goals to promote responsible resource consumption.

As the demand for these models rises, it's essential to consider their environmental impact and explore strategies for minimizing energy consumption in the pursuit of sustainable AI development.

5.4.2 Labor market

There are a significant portion of the workforce that could experience impacts on their jobs due to the introduction of LLMs, affecting different industries and income levels. While LLMs hold promise for enhancing efficiency and productivity, they also raise concerns about potential job displacement and changes in the nature of work. Moreover, the increased use of LLMs could make inequality worse and create a bigger difference between high-paying and low-paying jobs.

5.4.3 Safety

It's important to keep in mind that QA systems are machines, not humans. Therefore, there's a possibility of these systems providing incorrect information. This becomes particularly problematic when using a generative reader, which can produce inaccurate or irrelevant responses. In contrast, extractive readers retrieve answers only from the available knowledge base, minimizing the risk of misinformation. Another primary ethical concern around QA systems is the potential for bias and unfairness.

Chapter 6
Conclusion

In this thesis, we explored the use of open-retrieval question-answering systems to improve system documentation accessibility. In this section, we present our conclusions, addressing the research questions posed in this study. Various types of experiments have been performed on three different datasets. These experiments included a wide range of system configurations, including sparse and dense retrievers, extractive and generative readers, rankers, and additional experiments.

Our best-performing system configuration for ConsafeQA, BM25 + Fine-tuned Ranker + DeBERTA, achieves scores of 58.49/79.68/85.38 on EM/F1/BERTScore, respectively. For TriviaQA, the top-performing setup, GTE + Ranker + Flan-T5, achieves scores of 73.94/81.96/89.63.

As a standalone reader performance on the MRQA dataset, DeBERTa achieved the best overall performance. However, our findings indicate that various reader models excel in different contexts. For instance, the extractive model DeBERTa, performed better on ConsafeQA, whereas the generative model Flan-T5, demonstrated superior performance on TriviaQA. This indicates the importance of selecting the most appropriate model depending on the specific task.

Fine-tuning the ranker and concatenating titles and passages on ConsafeQA have both shown improvement compared to the original pipeline. In particular, Fine-tuning the ranker achieved the overall best result.

Lastly the answers to our research questions are provided below:

- What combination of Open-Retrieval Question-Answering components achieves the highest performance on benchmark datasets and technical system documentation domain?
    - There is no component combination that performs best on all datasets, instead

it is dataset-specific. The best stand-alone Reader that performed best on the MRQA dataset was Flan-T5 with an average F1 score of 82.9. For TriviaQA, if a Ranker is not utilized then a combination of BM25 and Flan-T5 performs the best achieving a 76.5 F1 score, however, utilizing a ranker does achieve the highest performance out of all configurations on TriviaQA, where the combination is GTE and Flan-T5 and the F1 score is 81.96. There is no combination of components that is far superior to the rest on ConsafeQA, however, the best-performing combination achieves a 79.68 F1 score and consists of BM25 and DeBERTa, while utilizing a fine-tuned ranker.

- What properties effect the performance of Question-Answering systems in the technical system documentation domain?
  - The largest factor that impacts the performance in the technical system documentation domain seem to be with questions regarding providing navigation to a requested object. Moreover, the examples in 5.1.5 also show that question-answering systems can have difficulties with handling technical jargon.

6.1 Future Work

Numerous aspects can be further worked upon within this field. However, we will outline mainly three aspects for future work that we decided based on our performed work and results.

Extensive Evaluation of Retriever. The first aspect of future work involves conducting a more comprehensive evaluation of the retriever component of the system. This evaluation should involve several domains and datasets, thus exploring its generalizability. However, these datasets should be more task-specific for information retrieval to provide a better understanding of the strengths and weaknesses associated with different types of retrievers. Thus more variations of sparse and dense retrievers should be investigated to see which perform better in different settings.

Multilingual Evaluation. The second aspect is evaluating the system in a multilingual setting. Evaluating how well the system handles questions and retrieves relevant documents in languages other than English can be a good representation of its usability on a larger scale. This means finding or creating multilingual datasets that are appropriate for open-retrieval question-answering, and experimenting with retrievers and readers that are trained in languages other than English.

Dataset Expansion. The third aspect is the creation of a larger dataset sourced from several companies operating in domains similar to Consafe Logistics. A diverse dataset that spans different companies will contribute to evaluating the robustness and generalizability of the overall system, thus showing how performant this type of open-retrieval question-answering system is in a broader context of real-life internal technical system documentation.

References

Arabzadeh, N., Yan, X., and Clarke, C. L. A. (2021). Predicting efficiency/effectiveness trade-offs for dense vs. sparse retrieval strategy selection. *CoRR*, abs/2109.10739.

Asai, A., Kasai, J., Clark, J. H., Lee, K., Choi, E., and Hajishirzi, H. (2020). XOR QA: cross-lingual open-retrieval question answering. *CoRR*, abs/2010.11856.

Chen, D., Fisch, A., Weston, J., and Bordes, A. (2017). Reading wikipedia to answer open-domain questions. *CoRR*, abs/1704.00051.

Chen, D. and Yih, W.-t. (2020). Open-domain question answering. In Savary, A. and Zhang, Y., editors, *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics: Tutorial Abstracts*, pages 34–37, Online. Association for Computational Linguistics.

Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., et al. (2022). Scaling instruction-finetuned language models. *arXiv*, abs/2210.11416.

Clark, K., Luong, M., Le, Q. V., and Manning, C. D. (2020). ELECTRA: pre-training text encoders as discriminators rather than generators. *CoRR*, abs/2003.10555.

Das, R., Dhuliawala, S., Zaheer, M., and McCallum, A. (2019). Multi-step retriever-reader interaction for scalable open-domain question answering. *CoRR*, abs/1905.05733.

Devlin, J., Chang, M., Lee, K., and Toutanova, K. (2018). BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805.

Dong, H., Lin, J., Leng, Y., Chen, J., and Xie, Y. (2023). Retriever and ranker framework with probabilistic hard negative sampling for code search. *arXiv*, abs/2305.04508.

Dua, D., Wang, Y., Dasigi, P., Stanovsky, G., Singh, S., and Gardner, M. (2019). DROP: A reading comprehension benchmark requiring discrete reasoning over paragraphs. *CoRR*, abs/1903.00161.

Dunn, M., Sagun, L., Higgins, M., Güney, V. U., Cirik, V., and Cho, K. (2017). Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179.

Farea, A., Yang, Z., Duong, K., Perera, N., and Emmert-Streib, F. (2022). Evaluation of question answering systems: Complexity of judging a natural language. *arXiv*, abs/2209.12617.

Fisch, A., Talmor, A., Jia, R., Seo, M., Choi, E., and Chen, D. (2019). MRQA 2019 shared task: Evaluating generalization in reading comprehension. In *Proceedings of 2nd Machine Reading for Reading Comprehension (MRQA) Workshop at EMNLP*.

Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2022). Gptq: Accurate post-training quantization for generative pre-trained transformers. *arXiv*, abs/2210.17323.

Gao, L., Ma, X., Lin, J., and Callan, J. (2023). Precise zero-shot dense retrieval without relevance labels. In Rogers, A., Boyd-Graber, J., and Okazaki, N., editors, *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 1762–1777, Toronto, Canada. Association for Computational Linguistics.

Hanna, M. and Bojar, O. (2021). A fine-grained analysis of BERTScore. In Barrault, L., Bojar, O., Bougares, F., Chatterjee, R., Costa-jussa, M. R., Federmann, C., Fishel, M., Fraser, A., Freitag, M., Graham, Y., Grundkiewicz, R., Guzman, P., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Kocmi, T., Martins, A., Morishita, M., and Monz, C., editors, *Proceedings of the Sixth Conference on Machine Translation*, pages 507–517, Online. Association for Computational Linguistics.

He, P., Gao, J., and Chen, W. (2021). Debertav3: Improving deberta using electra-style pretraining with gradient-disentangled embedding sharing. *CoRR*, abs/2111.09543.

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.

Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with gpus. *CoRR*, abs/1702.08734.

Joshi, M., Choi, E., Weld, D. S., and Zettlemoyer, L. (2017). Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension. *CoRR*, abs/1705.03551.

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.

Kembhavi, A., Seo, M., Schwenk, D., Choi, J., Farhadi, A., and Hajishirzi, H. (2017). Are you smarter than a sixth grader? textbook question answering for multimodal machine comprehension. In *2017 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 5376–5384.

Kononenko, O., Baysal, O., Holmes, R., and Godfrey, M. W. (2014). Mining modern repositories with elasticsearch. In *IEEE Working Conference on Mining Software Repositories*.

Kwiatkowski, T., Palomaki, J., Redfield, O., Collins, M., Parikh, A., Alberti, C., Epstein, D., Polosukhin, I., Devlin, J., Lee, K., Toutanova, K., Jones, L., Kelcey, M., Chang, M.-W., Dai, A. M., Uszkoreit, J., Le, Q., and Petrov, S. (2019). Natural questions: A benchmark for question answering research. *Transactions of the Association for Computational Linguistics*, 7:452–466.

Lai, G., Xie, Q., Liu, H., Yang, Y., and Hovy, E. H. (2017). RACE: large-scale reading comprehension dataset from examinations. *CoRR*, abs/1704.04683.

Levy, O., Seo, M., Choi, E., and Zettlemoyer, L. (2017). Zero-shot relation extraction via reading comprehension. *CoRR*, abs/1706.04115.

Li, H., Wang, S., Zhuang, S., Mourad, A., Ma, X., Lin, J., and Zuccon, G. (2022). To interpolate or not to interpolate: Prf, dense and sparse retrievers. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval*, SIGIR '22. ACM.

Li, Z., Zhang, X., Zhang, Y., Long, D., Xie, P., and Zhang, M. (2023). Towards general text embeddings with multi-stage contrastive learning. *arXiv*, abs/2308.03281.

Liu, R., Shi, Y., Ji, C., and Jia, M. (2019). A survey of sentiment analysis based on transfer learning. *IEEE Access*, 7:85401–85412.

Luo, M., Hashimoto, K., Yavuz, S., Liu, Z., Baral, C., and Zhou, Y. (2022). Choose your qa model wisely: A systematic study of generative and extractive readers for question answering. *arXiv*, abs/2203.07522.

Ma, X., Zhang, X. C., Pradeep, R., and Lin, J. J. (2023). Zero-shot listwise document reranking with a large language model. *arXiv*, abs/2305.02156.

Mikolov, T., Chen, K., Corrado, G. S., and Dean, J. (2013a). Efficient estimation of word representations in vector space. In *International Conference on Learning Representations*.

Mikolov, T., Yih, W.-t., and Zweig, G. (2013b). Linguistic regularities in continuous space word representations. In Vanderwende, L., Daumé III, H., and Kirchhoff, K., editors, *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 746–751, Atlanta, Georgia. Association for Computational Linguistics.

Muennighoff, N., Wang, T., Sutawika, L., Roberts, A., Biderman, S., Scao, T. L., Bari, M. S., Shen, S., Yong, Z.-X., Schoelkopf, H., et al. (2022). Crosslingual generalization through multitask finetuning. *arXiv*, abs/2211.01786.

Panja, J. and Naskar, S. K. (2018). ITER: Improving translation edit rate through optimizable edit costs. In Bojar, O., Chatterjee, R., Federmann, C., Fishel, M., Graham, Y., Haddow, B., Huck, M., Yepes, A. J., Koehn, P., Monz, C., Negri, M., Névéol, A., Neves, M., Post, M., Specia, L., Turchi, M., and Verspoor, K., editors, *Proceedings of the Third Conference on Machine Translation: Shared Task Papers*, pages 746–750, Belgium, Brussels. Association for Computational Linguistics.

Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a method for automatic evaluation of machine translation. In Isabelle, P., Charniak, E., and Lin, D., editors, *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

Pennington, J., Socher, R., and Manning, C. (2014). GloVe: Global vectors for word representation. In Moschitti, A., Pang, B., and Daelemans, W., editors, *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1532–1543, Doha, Qatar. Association for Computational Linguistics.

Radford, A., Narasimhan, K., Salimans, T., Sutskever, I., et al. (2018). Improving language understanding by generative pre-training. *OpenAI*.

Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2019). Exploring the limits of transfer learning with a unified text-to-text transformer. *CoRR*, abs/1910.10683.

Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). Squad: 100, 000+ questions for machine comprehension of text. *CoRR*, abs/1606.05250.

Reimers, N. and Gurevych, I. (2019). Sentence-bert: Sentence embeddings using siamese bert-networks. *CoRR*, abs/1908.10084.

Robertson, S. E. and Walker, S. (1994). Some simple effective approximations to the 2-poisson model for probabilistic weighted retrieval. In *Annual International ACM SIGIR Conference on Research and Development in Information Retrieval*.

Rosa, G. M., Rodrigues, R. C., de Alencar Lotufo, R., and Nogueira, R. F. (2021). Yes, BM25 is a strong baseline for legal case retrieval. *CoRR*, abs/2105.05686.

Saha, A., Aralikatte, R., Khapra, M. M., and Sankaranarayanan, K. (2018). Duorc: Towards complex language understanding with paraphrased reading comprehension. *CoRR*, abs/1804.07927.

Sanh, V., Webson, A., Raffel, C., Bach, S. H., Sutawika, L., Alyafeai, Z., Chaffin, A., Stiegler, A., Scao, T. L., Raja, A., Dey, M., Bari, M. S., Xu, C., Thakker, U., Sharma, S., Szczechla, E., Kim, T., Chhablani, G., Nayak, N. V., Datta, D., Chang, J., Jiang, M. T., Wang, H., Manica, M., Shen, S., Yong, Z. X., Pandey, H., Bawden, R., Wang, T., Neeraj, T., Rozen, J., Sharma, A., Santilli, A., Févry, T., Fries, J. A., Teehan, R., Biderman, S., Gao, L., Bers, T., Wolf, T., and Rush, A. M. (2021). Multitask prompted training enables zero-shot task generalization. *CoRR*, abs/2110.08207.

Su, D., Patwary, M., Prabhumoye, S., Xu, P., Prenger, R. J., Shoeybi, M., Fung, P., Anandkumar, A., and Catanzaro, B. (2023). Context generation improves open domain question answering. In *In Findings of the Association for Computational Linguistics: EACL 2023*, page 781–796.

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.

Trischler, A., Wang, T., Yuan, X., Harris, J., Sordoni, A., Bachman, P., and Suleman, K. (2016). NewsQA: A Machine Comprehension Dataset. *CoRR*, abs/1611.09830.

Tsatsaronis, G., Balikas, G., Malakasiotis, P., Partalas, I., Zschunke, M., Alvers, M., Weißenborn, D., Krithara, A., Petridis, S., Polychronopoulos, D., Almirantis, Y., Pavlopoulos, J., Baskiotis, N., Gallinari, P., Artieres, T., Ngonga Ngomo, A.-C., Heino, N., Gaussier, E., Barrio-Alvers, L., and Paliouras, G. (2015). An overview of the bioasq large-scale biomedical semantic indexing and question answering competition. *BMC Bioinformatics*, 16:138.

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

Wei, J., Wang, X., Schuurmans, D., Bosma, M., Chi, E. H., Le, Q., and Zhou, D. (2022). Chain of thought prompting elicits reasoning in large language models. *CoRR*, abs/2201.11903.

Workshop, B., Scao, T. L., Fan, A., Akiki, C., Pavlick, E., Ilić, S., Hesslow, D., Castagné, R., Luccioni, A. S., Yvon, F., et al. (2022). Bloom: A 176b-parameter open-access multilingual language model. *arXiv*, abs/2211.05100.

Yang, Z., Qi, P., Zhang, S., Bengio, Y., Cohen, W. W., Salakhutdinov, R., and Manning, C. D. (2018). Hotpotqa: A dataset for diverse, explainable multi-hop question answering. *CoRR*, abs/1809.09600.

Yao, X., Zheng, Y., Yang, X., and Yang, Z. (2021). NLP from scratch without large-scale pretraining: A simple and efficient framework. *CoRR*, abs/2111.04130.

Zhang*, T., Kishore*, V., Wu*, F., Weinberger, K. Q., and Artzi, Y. (2020). Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Zhuang, F., Qi, Z., Duan, K., Xi, D., Zhu, Y., Zhu, H., Xiong, H., and He, Q. (2019). A comprehensive survey on transfer learning. *CoRR*, abs/1911.02685.

Zuva, K. and Zuva, T. (2012). Evaluation of information retrieval systems. *International journal of computer science & information technology*, 4(3):35.

INSTITUTIONEN FÖR DATAVETENSKAP | LUNDS TEKNISKA HÖGSKOLA | PRESENTERAD 2024-04-04

EXAMENSARBETE Enhancing System Documentation Accessibility through Question-Answering
STUDENTER Samil Kapetanovic, Mohammad Raja
HANDLEDARE Marcus Klang (LTH), Eltayeb Bayomi (Consafe Logistics AB)
EXAMINATOR Jacek Malec (LTH)

Frågebesvarande system: nyckeln till effektiv dokumentationstillgänglighet

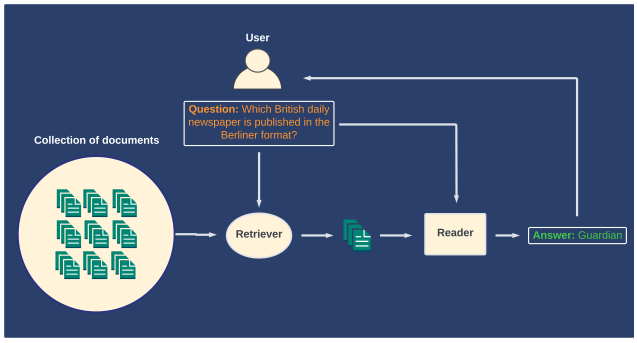
POPULÄRVETENSKAPLIG SAMMANFATTNING **Samil Kapetanovic, Mohammad Raja**

Frågebesvarande system kan ge snabb tillgång till efterfrågad information med hjälp av maskininlärning. I detta arbete har vi undersökt och utvärderat prestandan av olika typer av frågebesvarande system inom bland annat teknisk dokumentationsdomänen.

För att effektivisera och förenkla processen att hitta svar på frågor utan att behöva lägga ner omfattande tid på att söka efter relevant information, har de senaste framstegen inom språkteknologi öppnat upp nya möjligheter. Genom användning av maskininlärning har frågebesvarande system förmågan att snabbt och precist ge tillgång till svar på användares frågor. Det är dock vanligt att dessa system utvärderas med generella dataset istället för specifika näringslivsdata, vilket kan begränsa deras användbarhet i praktiken. Med Consafe Logistics AB demonstrerar vi potentialen hos frågebesvarande system inom den tekniska dokumentationsdomänen. Utveckligen av ett frågebesvarande system inom denna domän, möjliggör en mer effektiv och produktiv användning av tiden genom att minska behovet av manuell sökning och tolkning av information inom verksamheten.

I detta examensarbete har vi undersökt och utvärderat olika konfigurationer av frågebesvarande system. Dessa system bygger på en grundläggande tvådelad arkitektur som först letar efter relevanta passager i en stor mängd textdokument, och sedan tolkar den givna frågan med de hämtade textpassager för att producera ett svar.

Det finns olika varianter av varje komponent i denna tvådelade arkitektur, och vi har kombinerat olika varianter och typer för att skapa olika konfigurationer av frågebesvarande system. Vi har utvärderat prestandan hos dessa konfigurationer med hjälp av två allmänna dataset samt ett dataset som är specifikt anpassat för teknisk dokumentationsdomänen. Detta dataset utvecklades i samarbete med Consafe Logistics AB.



Våra resultat indikerar att olika kombinationer av komponenter visar varierande prestanda vid utvärdering av de olika datasets, vilket understryker vikten av att anpassa systemets komponenter efter den specifika domänen. Dessutom visar resultaten några egenskaper som påverkar prestandan hos frågebesvarande system för teknisk dokumentationsdomänen.