Universität Heidelberg
Institut für Informatik
Arbeitsgruppe Datenbanksysteme

Master-Arbeit

Building an adaptable and resource constrained Conversational Information Search System

Name:	Stephan Lenert
Matrikelnummer:	Matrikelnummer der Autorin/des Autors
Betreuer:	Name der Betreuerin / des Betreuers
Datum der Abgabe:	December 26, 2023

Ich versichere, dass ich diese Master-Arbeit selbstständig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe und die Grundsätze und Empfehlungen "Verantwortung in der Wissenschaft" der Universität Heidelberg beachtet wurden.

Abgabedatum: December 26, 2023

Zusammenfassung

Die Zusammenfassung muss auf Deutsch **und** auf Englisch geschrieben werden. Die Zusammenfassung sollte zwischen einer halben und einer ganzen Seite lang sein. Sie soll den Kontext der Arbeit, die Problemstellung, die Zielsetzung und die entwickelten Methoden sowie Erkenntnisse beschreiben.

Abstract

The abstract has to be given in German **and** English. It should be between half a page and one page in length. It should cover in a readable and comprehensive style the context of the thesis, the problem setting, the objectives, and the methods developed in this thesis as well as key insights and results.

Contents

List of Acronyms

List of Figures

List of Tables

1. Introduction

This chapter is an introduction to the topic of this thesis. It starts with a brief overview of the current state of the art in the field of question answering and chatbots. Then, it describes the motivation behind this thesis and the goals that are to be achieved. Finally, it gives an overview of the structure of this thesis.

2. Background and Related Work

This chapter provides essential background information and reviews relevant prior research. It commences with an introduction to the sub-task of Question Answering (QA), as presented in Section 2.1. As previously mentioned in the Introduction (Chapter 1), this chapter maintains a clear distinction between QA and Conv QA. Consequently, Section 2.2 extends upon the foundational knowledge of QA and introduces the requisite concepts for the transformation of a QA-System into a Conv QA-System. Section 2.4 will delve into the related work, providing a comprehensive overview of the current state-of-the-art in the field of QA and Conv QA over textual knowledge sources.

2.1. Question Answering

The evolution of QA as a research field provides a solid foundation for understanding current research initiatives and methodologies. Among the early contributions is BASEBALL, an automated QA system developed by researchers at Massachusetts Institute of Technology (MIT) in 1961. This QA system demonstrated its capability to answer questions related to baseball using natural English language [Green et al., 1961].

In 1999, Text REtrieval Conference (TREC) (Text Retrieval Conference) initiated the TREC-8 Question Answering track, which marked "the first large-scale evaluation of domain-independent question-answering systems" [Voorhees, 1999]. A more well-known QA system is *Watson* by IBM, an open-domain QA system that won a the TV show Jeopardy! in 2011 [Ferrucci, 2012]. It is evident that an evolutionary process has occurred between the early research in 1961 and today's systems like *ChatGPT* by OpenAI. To understand the dimensions in which these systems differ, their components, and how to distinguish them will be introduced in Section 2.1.1, while subsequent sections will delve deeper into specific components.

In 1999, the TREC initiated the TREC-8 Question Answering track, marking "the first large-scale evaluation of domain-independent question-answering systems" [Voorhees, 1999]. A more renowned QA system is IBM's *Watson*, an open-domain QA system that famously triumphed on the television game show Jeopardy! in 2011 [Ferrucci, 2012]. It is evident that an evolutionary process has transpired between the early research in 1961 and contemporary systems such as OpenAI's *ChatGPT*. In section 2.1.1 we will lay the groundwork by introducing the fundamental aspects of QA-Systems and the techniques used to differentiate and categorize them. Following that, subsequent sections will delve deeper into the examination of specific system components.

2.1.1. Basics

Jurafsky and Martin define a QA-System as a system "designed to satisfy human information needs" [Jurafsky and Martin, 2023]. Hence, it primarily functions as an Information Retrieval System, with its primary objective being to provide users with the desired and accurate information in response to natural language requests.

The research community has yet to establish a universally accepted classification framework for Question Answering (QA) systems. For instance, Hao et al. and Farea et al. [Hao et al., 2022, Farea et al., 2022] take a comprehensive approach to classify QA systems but differ in certain aspects, such as their treatment of question types and knowledge sources. On the other hand, other researchers [Zhu et al., 2021, Jurafsky and Martin, 2023, Etezadi and Shamsfard, 2023, Zhang et al., 2023b] employ a similar classification methodology but often focus solely on retrieval-based approaches, thereby lacking a holistic perspective.

The classification proposed by Farea et al. [Farea et al., 2022] goes a step further by distinguishing between the **QA-Framework** and **QA-Paradigms**, enhancing its versatility for comparing classical and modern QA systems. An adaptation of this classification will be utilized in this thesis. The originally proposed QA algorithms have been extended to include the Retrieval-based approach, and the Question Types have been revised based on the typology introduced by Mishra et al. in their 2016 survey [Mishra and Jain, 2016], which was further elaborated upon by Etezadi et al. [Etezadi and Shamsfard, 2023]. Also the Answer Types where adjusted to align with the classifications used in [McDonald et al., 2022, Dasigi et al., 2021]. In this context, a crucial distinction is made between a **QA** and **ConvQA** system, guided by the criteria outlined in [Zamani et al., 2023]: a QA system exclusively handles standalone questions, while any inquiry exceeding a single question and involving conversational context falls within the domain of a **ConvQA** system.

The **QA-Framework** encompasses external factors such as Question and Answer Types, while also considering system-related factors like the QA Algorithm and Knowledge Source [Farea et al., 2022, Hao et al., 2022]. Conversely, the **QA-Paradigm** defines the fundamental underlying concept of a system and can be seen as a subset of possible combinations within the **QA Framework**. Currently, three dominant paradigms

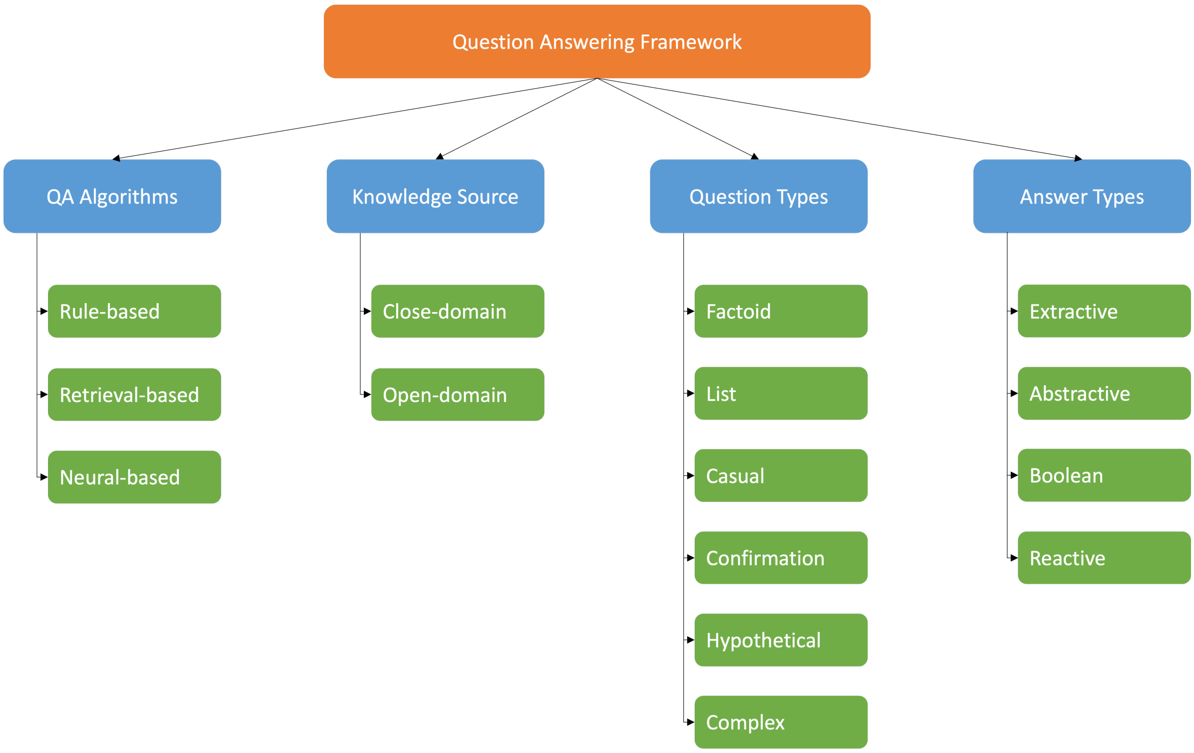

Figure 2.1.: Adjusted QA Framework Classification by Farea et al. [Farea et al., 2022]

prevail:

1. **Information Retrieval (IR)-Based QA**: This paradigm involves searching through extensive multi-modal data based on a user's question and using the retrieved passages to generate an answer.

2. **Knowledge Base (KB) QA**: In this approach, a semantic representation of the question is constructed, and a knowledge base is queried using this representation. The returned results are then used to generate an answer.

3. **Generative Question Answering**: Here, knowledge is fully implicit, and a neural network (NN) generates answers based on its trained parameters.

For visual clarity, a diagram illustrating the adjusted QA Framework Classification by Farea et al. is provided in Figure 2.1.

Figure 2.1 illustrates the aforementioned classification. The primary distinguishing factor is the employed **QA Algorithm**. Rule-based approaches involve the manual crafting of feature extractions from user questions, which are then compared to the knowledge base. Rule-based approaches are typically employed in closed-domain QA systems exclusively [Etezadi and Shamsfard, 2023].

Retrieval-based approaches are the classic Information Retrieval (IR)-based QA systems, comprising two key components: an intent classifier and a retriever. The intent classifier's objective is to discern the question's intent and identify important entities. Subsequently, the retriever searches the knowledge source and identifies the most relevant passages [Farea et al., 2022, Zhu et al., 2021].

The Neural-based approach, often referred to as the generative approach, utilizes a Sequence-to-Sequence (S2S) model to generate accurate answers to given questions. In this paradigm, the information is stored directly in the neural network's parameters, otherwise the neural network is part of a Retrieval-based approach. Most datasets in these contexts consist of triples of question, context, and answer pairs [Jurafsky and Martin, 2023]. Notably, widely used datasets such as SQuAD and QASPER originally emerged from the field of machine reading comprehension, representing a foundational step in the evolution of QA systems [Rajpurkar et al., 2016, Dasigi et al., 2021, Zhu et al., 2021].

In addition to the **QA Algorithms**, the **Knowledge Source** plays a pivotal role in distinguishing various aspects of Question Answering (QA) systems. The nature of the knowledge source can range from structured to unstructured or semi-structured, and it may encompass diverse data modalities, including text, audio, and video. A common point of comparison in the QA landscape is between closed and open-domain systems.

In the broad sense, a **closed-domain** QA system operates within the confines of a specific knowledge domain, which means it has limited access to information. In contrast, **open-domain** QA systems grapple with an extensive array of knowledge sources, necessitating a more versatile approach [Farea et al., 2022].

Furthermore, a closed-domain setup often entails limitations on the types of questions it can handle, primarily focusing on factoid questions or predefined templates. Additionally, it frequently relies on structured knowledge bases like graphs or logically organized repositories [Hao et al., 2022].

Conversely, open-domain QA systems are designed to tackle a wide spectrum of user queries, ranging from factoids to more complex inquiries. They typically deal with unstructured knowledge sources, which can be substantial and diverse in content [Zhu et al., 2021, Farea et al., 2022, Jurafsky and Martin, 2023].

An alternative perspective for distinguishing QA-Systems lies in the **Question Types** that users can input into the system. Questions can fall into various categories, such as *factoid, list, casual, confirmation, hypothetical* [Mishra and Jain, 2016], or *complex* [Etezadi and Shamsfard, 2023].

- *Factoid questions*, the most common type, are typically signaled by question words (what, when, which, who, how) and yield a concise factual answer.

- *List questions* represent a specialized subset of factoid questions, where the answer comprises a list of facts.
- *Casual questions* encompass inquiries that deviate from the factoid format, often involving words like *how* or *why* and requiring more advanced reasoning.
- *Confirmation questions* seek simple yes or no responses, frequently employed in personal assistant applications.
- *Hypothetical questions* delve into hypothetical scenarios (e.g., "what would happen if"), aiming for plausible rather than definitive answers.
- *Complex questions* can be further categorized into *answer-retrieval-complex* and *question-understanding-complex*. In the case of question-understanding-complex questions, the complexity arises from nuances like multiple constraints, making the question itself intricate to comprehend. In contrast, answer-retrieval-complex questions involve complexities in finding the correct answer, often requiring the combination of information from multiple documents or similar sources. This is commonly referred to as long-form QA.

Lastly, a QA-System can be characterized by the **Answer Types** it offers, a concept closely intertwined with Question Types. Farea et al. [Farea et al., 2022] delineate three categories of answers: *extractive, abstractive, boolean* and *reactive.*

- *Extractive answers* represent the most common type, where the answer is a specific factual excerpt presented as a span of tokens.
- *Abstractive answers* typically correspond to complex questions that necessitate the system to consider multiple documents and information sources to formulate a response. In such cases, no predefined or annotated answer exists.
- *Boolean answers* are typically the result of confirmation questions, where the answer is either *yes* or *no.*
- *Reactive answers* often arise in response to confirmation questions and can be a system-generated reaction based on the user's provided answer.

2.1.2. Information Retrieval Architectures

As stated in the previous section (Section 2.1.1), there are three major paradigms in QA: Information Retrieval (IR)-based QA, Knowledge Base (KB)-based QA, and Generative

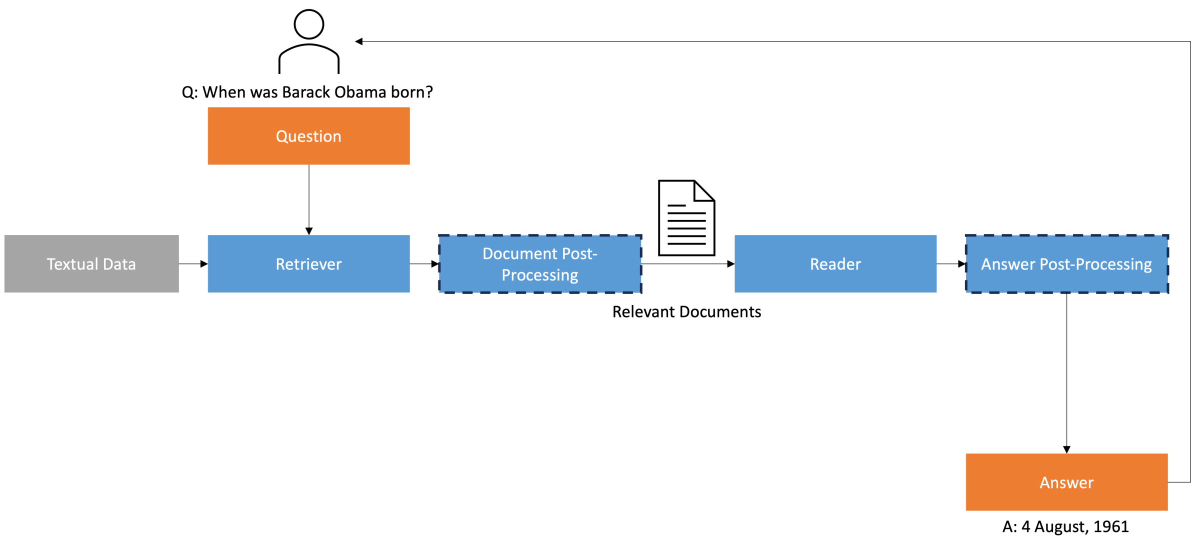

Figure 2.2.: Reader-Retriever-System Architecture for QA by Zhu et al. [Zhu et al., 2021]. The dashed lines indicate optional modules.

QA. This section will primarily concentrate on the first paradigm, IR-based QA, as it holds the most promise for addressing the objectives of this thesis topic.

This thesis will not focus on KB QA, as this approach requires the mapping of the query to a structured data representation. As the task of this thesis is to develop a general system, which is adaptable to different data inputs, KB QA will be excluded [Dimitrakis et al., 2020].

Generative QA is often denoted as *Retriever-free* or *Neural-based* approaches. The central characteristic of this paradigm is that knowledge resides within the parameters of a neural network. Consequently, the knowledge is implicit, and the QA system will not furnish a specific document, passage, or other source from which it extracted the information. Instead, it offers a textual excerpt. While these systems can achieve competitive performance compared to IR-based QA systems, they are not under consideration for this thesis due to their lack of reference, which is a crucial requirement for the system to be developed [Roberts et al., 2020].

Figure 2.2 depicts the general architecture of a **Retriever-Reader-System**, as defined by Zhu et al. [Zhu et al., 2021]. This architecture serves as the foundational framework for IR-Based QA systems and was initially introduced by Harabagiu et al. [Harabagiu et al., 2003]. In this framework, all modules operate independently, can be trained separately, and are subject to independent evaluation.

The **Retriever** module's primary role is to retrieve relevant documents, passages, or other pertinent information from a knowledge source and rank them based on their relevance to answering the user's query. Subsequently, the **Reader** module extracts the answer from the retrieved documents and presents it to the user. This task bears a close resemblance to Machine Reading Comprehension (MRC), with the key distinction that in IR-Based QA, the system must handle multiple documents and comprehend them to formulate a response, unlike classical MRC tasks, which typically involve only one context document.

The **Document Post-Processor** module's role is to curate and refine the set of documents that will be forwarded as "Relevant Documents" to the subsequent stage, the Reader. Concurrently, the **Answer Post-Processor** assists the Reader in addressing complex questions for which the answer may not be found in a single document alone [Zhu et al., 2021, Jurafsky and Martin, 2023].

It's worth noting that some researchers include a **Question Analysis** module preceding the Retriever, which aims to preprocess the received question for more efficient query execution in the Retriever [Nassiri and Akhloufi, 2023]. However, for the purposes of this thesis, we adhere to Zhu et al.'s definition [Zhu et al., 2021], where this functionality is considered part of the Retriever.

Conceptually, there are three distinct approaches to the Retriever itself: *Sparse Retrieval, Dense Retrieval,* and *Iterative Retrieval.* The specifics of these approaches will be thoroughly explored in Section 2.1.4.

Document Post-Processors can be categorized into *Supervised Learning, Reinforcement Learning,* and *Transfer Learning*-based approaches. A detailed discussion of these approaches is also provided in Section 2.1.4.

In Section 2.1.5, we will delve into the finer details of Reader approaches and Answer Post-processing. Broadly speaking, there are two primary types of Readers: *Extractive* and *Generative* Readers. As for Answer Post-processing, it involves two key categories: *Rule-based* and *Learning-based* approaches.

There are also **End-to-End** approaches that employ a single module to execute the entire QA task. Excluding generative approaches, two common categories of such approaches are **Retriever-Reader** and **Retriever-only** models.

An End-to-End Retriever-Reader aims to train both the Retriever and Reader in a single backpropagation step, and in some cases, it introduces additional knowledge sources beyond the traditional IR framework. An illustrative example is RAG [Lewis et al., 2021]. RAG consists of a pre-trained Generator with implicit knowledge encoded in its parameters and a pre-trained Retriever. For each question, the Retriever identifies the most relevant documents and generates a latent vector based on them. This latent vector, along with the original question, is fed into the Generator. Section 2.1.5 will delve into details regarding the RAG architecture.

Another end-to-end approach, similar to RAG, is Retrieval-Augmented Language

Model pre-training (REALM) [Guu et al., 2020]. While these previous two approaches extended the capabilities of pre-trained Sequence-to-Sequence (seq-2-seq) models, Nishida et al. pursued a different path by training a single Neural Network (NN) to perform both tasks simultaneously: IR and MRC [Nishida et al., 2018].

It is noteworthy that all these end-to-end approaches have demonstrated competitive performance compared to state-of-the-art methods on specific QA datasets.

An essential yet often underestimated question is: What defines textual data, and how should one preprocess formats such as PDFs to extract this textual content? While many datasets already comprise small contextual snippets [Wang, 2022], it's crucial not to overlook the entire process of extracting snippets from unstructured PDFs, for example. Approaches to tackle this challenge will be explored in detail in the upcomming Section 2.1.3.

2.1.3. Extraction Approaches

As discussed in the previous Section 2.1.1, the knowledge source for a QA-System can take the form of textual or multimodal data. The specific type of data may necessitate certain requirements or specific adjustments to the Retriever used for IR.

In the context of this thesis, the primary knowledge source to be employed is PDF documents. In the research field, three major approaches exist for extracting textual information from unstructured data types like PDFs: *visual* [Tito et al., 2021], *direct* [Wang et al., 2019], and *alternative* [Dasigi et al., 2021] extraction methods.

It's important to note upfront that the chosen extraction method is intricately connected to the subsequent retrieval approach. The specifics, including metadata alongside pure textual data and quality requirements, may vary among different extraction and retrieval methods.

The visual approach is closely aligned with the research field of *Document Question Answering*. A well-known example dataset in this field is Document Visual Question Answering (DocVQA) [Tito et al., 2021]. The primary concept behind the visual approach to document question answering is to capture not only the text of a PDF but also additional information such as the document's structure, various hierarchies on a page (e.g., sections, subsections), and the ability to analyze tables and figures. These hierarchical structures can be leveraged to create two-stage retrieval approaches. In these approaches, initially, a collection of relevant files is identified based on higher-level attributes like the document's title and abstract. Subsequently, a more granular retrieval process is executed over lower-level attributes such as passages within the relevant files. These *Iterative Retrievers* will be further discussed in Section 2.1.4 [Liu et al., 2021b].

The challenge of *Visual Document Question Answering* typically involves taking images of PDF pages as inputs and mapping question-answer pairs to them. The answers are extracted from either a single paragraph or a combination of multiple paragraphs [Mathew et al., 2021]. Nonetheless, the extraction pipeline in this case usually resembles the *Retriever-Reader* architecture, where the extracted information from the visual processing is fed into such a system afterward. Researchers in this field often employ a pipeline that includes a *Document Layout Analysis* model, followed by the application of an Optical Character Recognition (OCR) tool to the detected regions [McDonald et al., 2022]. Examples of a *Document Layout Analysis* model include the Document Image Transformer by Li et al. [Li et al., 2022b].

The direct approach is the most prevalent method in the field of Question Answering (QA) and Information Retrieval (IR). The primary concept behind this approach is to extract textual information from PDFs and store it in a database. The extraction process can be accomplished using various tools such as *PDFMiner* or *Adobe Extract* [Meuschke et al., 2023]. However, a lingering question is how to effectively split the extracted textual data, especially considering that they are often not cleaned after extraction.

A common practice when employing a Language Model (LLM) is to optionally cleanse the text corpus and then divide it based on a predefined token size. This approach is evident in two notable open-source LLM projects: *Langchain* and the *Retrieval Plugin for ChatGPT* by OpenAI [Langchain, 2023, OpenAI, 2023]. In the original Dense Retrieval paper by Karpukhin et al., a sliding window of token size 5 was utilized [Karpukhin et al., 2020]. Therefore, it can be assumed that for contemporary LLM applications, the precise quality of the data, ensuring that a document contains syntactically correct sentences, may not be as critical.

Apart from modern approaches involving text clipping, previous methods aimed to identify paragraphs and similar structures within the extracted texts [Zhu et al., 2021].

An alternative approach involves the methodology employed in constructing the QASPER dataset. In this case, the authors conducted a pre-filtering of scientific papers' PDFs, selecting only those with freely accessible LaTeX files. They then utilized the S2ORC tool to extract cleaned textual data from these LaTeX files [Dasigi et al., 2021]. It's important to note that this approach is highly specific to the QASPER dataset and cannot be universally applied. Nonetheless, it serves as an illustration of alternative methods for extracting textual data from PDFs.

2.1.4. Retrieval Approaches

The traditional state-of-the-art in IR relies on **Sparse Retrievers**, with one notable example being BM25. BM25 is renowned as "one of the most empirically successful retrieval models and is widely used in current search engines" [Zhu et al., 2021]. Nandan et al. even demonstrated that on modern Open-Domain Question Answering (ODQA) datasets, BM25 remains a viable baseline for zero-shot IR [Thakur et al., 2021].

BM25 was originally introduced by Robertson et al. [Robertson and Zaragoza, 2009]. It operates by utilizing the TF-IDF token weights between a question q containing tokens $q_1, \ldots, q_T$ and a set of passages P, where $p \in P$.

$$\mathbf{s}_{q,p}^{\text{BM25}} = \sum_{i=1}^{T} \log\left(\frac{|\mathcal{P}|}{N(q_i, \mathcal{P})}\right) \frac{n(q_i, p)(k_1 + 1)}{k_1\left(1 - b + \frac{b|p|}{avpl}\right) + n(q_i, p)} \tag{2.1}$$

Equation 2.1 illustrates the BM25 score for a question q and a passage p. In this equation, $N(q_i, \mathcal{P})$ represents the count of passages in $\mathcal{P}$ that contain the token q_i, while $n(q_i, p)$ indicates the frequency of token q_i within the passage p. The variable $|p|$ signifies the length of passage p, and $avpl$ stands for the average passage length in $\mathcal{P}$. The parameters k_1 and b are free parameters, typically set to $k_1 = 0.9$ and $b = 0.4$ [McDonald et al., 2022, Robertson and Zaragoza, 2009].

Traditionally, this lexical Information Retrieval (IR) approach has been capable of providing satisfactory retrieval results. However, in 2020, Karpukhin et al. demonstrated for the first time that a **Dense Retrieval** approach could outperform the Sparse Retrieval approach across multiple ODQA datasets [Karpukhin et al., 2020]. Consequently, the search for a general Dense Retrieval model has been ongoing, as these Dense Retrieval approaches offer advantages such as semantic matching and the ability to handle lengthy documents [Zhu et al., 2021].

In general, there are three types of Dense Retrieval approaches [Zhu et al., 2021]: the **Representation-based Retriever**, often referred to as the *dual-encoder* [Karpukhin et al., 2020]; the **Interaction-based Retriever**, often referred to as the *cross-encoder*; and the **Representation-interaction Retriever**, often referred to as the *multi-stop retriever*. Figure 2.3 illustrates the general architecture of these three types of Dense Retrievers.

The **Dense Passage Retriever (DPR)** by Karpukhin et al. serves as a notable example to explain the **Representation-Based Retriever**. Given a collection M of text passages p and a question q, the objective of DPR is to identify the k most similar passages to the question. To achieve this, DPR employs two distinct **BERT** [Devlin et al., 2019] Encoders. One Encoder, denoted as $E_Q(\cdot)$, encodes the question q into a d-dimensional vector, where $d = 768$. The other Encoder, labeled as $E_P(\cdot)$, encodes the passage p into a d-dimensional vector at the `[CLS]` token. The similarity between these two vectors is computed using the inner product:

$$\mathbf{s}_{q,p}^{DPR} = \mathbf{E}_Q(q)^\top \mathbf{E}_P(p) \tag{2.2}$$

The choice of the inner product as the similarity function is motivated by its computational efficiency and the demonstrated, comparable performance [Karpukhin et al., 2020]. It is crucial for the dot-product to yield a small value for pairs of questions and passages that are genuinely related. The training dataset D comprises m instances, where q_i represents the question, p_i^+ denotes the positive passage, and p_{i,n^-} represents the negative passage:

$$\mathbf{D} = \left\{ \left(q_i, p_i^+, p_{i,1}^-, \ldots, p_{i,n}^- \right) \right\}_{i=1}^{m} \tag{2.3}$$

The loss function is optimized using the negative log likelihood of p_i^+:

$$\mathcal{L}_{DPR} = -\log \frac{\exp\left(\mathbf{s}_{q_i,p_i^+}^{DPR}\right)}{\exp\left(\mathbf{s}_{q_i,p_i^+}^{DPR}\right) + \sum_{j=1}^{n} \exp\left(\mathbf{s}_{q_i,p_{i,j}^-}^{DPR}\right)} \tag{2.4}$$

It's important to note that in [Karpukhin et al., 2020], the selection of negative passages was not arbitrary. Instead, two additional approaches were employed: BM25 top passages that do not contain the answer and positive passages paired with other questions.

One significant advantage of the Representation-Based Retriever is that passages can be pre-indexed locally rather than at runtime. This reduction in latency between the question and the response may, however, come with trade-offs in the quality of the retrieved passages.

The **Interaction-Based Retriever** incorporates both the question q and the passage p within a single model, separated by a `[SEP]` indicator. These models offer various approaches for modeling the relationship between q and p. For instance, one common method is to utilize the `[CLS]` classifier as an indicator of whether the passage is relevant to the question. This approach was first introduced with Bidirectional Encoder Representations from Transformers (BERT) [Devlin et al., 2019]. While these models perform competitively with previous Representation-Based Retrievers, it's important to note that they are 100-1000 times more computationally expensive [Khattab and Zaharia, 2020].

To address this latency issue, models like ColBERT introduced the concept of **co**ntextualized **l**ate ineraction [Khattab and Zaharia, 2020]. In this thesis and subsequently in research, it is referred to as the **Representation-Interaction Retriever** [Zhu et al., 2021].

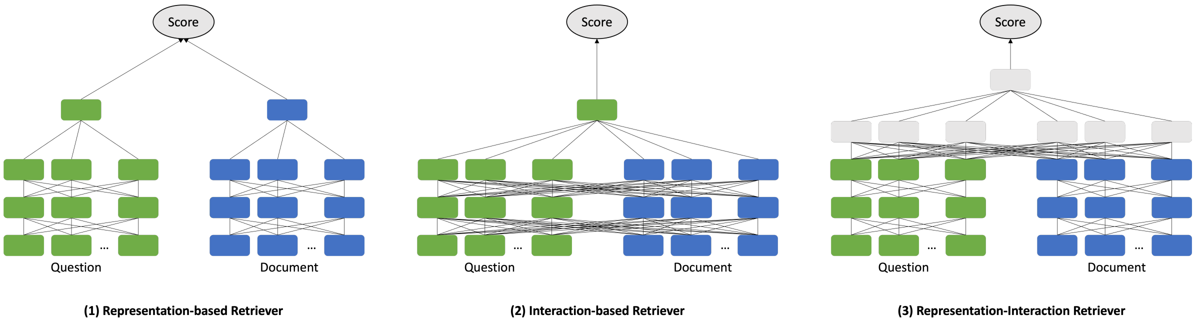

Figure 2.3.: Types of Dense Retriever by Zhu et al. [Zhu et al., 2021].

ColBERT, like Dense Passage Retrieval (DPR), employs two BERT Encoders, denoted as $E_Q(\cdot)$ and $E_P(\cdot)$. However, it introduces a late interaction mechanism. When provided with a query q, it is initially tokenized into BERT-based Wordpiece tokens, resulting in $q_1, \ldots, q_T$. Following the `[CLS]` token, a `[Q]` token is appended to signify the question. If the length of the tokenized question is less than N_q, a predetermined token length, the remaining portion of the question is padded with BERT's `[mask]` token. Otherwise, it is truncated. This process, known as *query augmentation*, allows BERT to re-weight existing terms or expand the query, and it is pivotal to ColBERT's performance. The generated embeddings are then passed through a linear layer to reduce the output dimensions to a fixed size m, which is smaller than the original dimensions of BERT. The output is subsequently normalized to ensure that the L2 norm of each result equals one.

For each passage p, $E_P(\cdot)$ is employed for encoding. Similar to the question encoding process, p is segmented into its $p_1, \ldots, p_{T_d}$ Wordpiece tokens. The special token `[D]` indicates a passage. Short passages are not padded with a `[mask]` token. After the classical BERT output, a similar post-processing step is applied to the encoded passages, and all embeddings corresponding to punctuation are filtered out.

$$\mathbf{E}_q := Normalize(CNN(BERT("[Q]q_0q_1 \ldots q_T[mask] \ldots [mask]"))) \tag{2.5}$$

$$\mathbf{E}_p := Filter(Normalize(CNN(BERT("[D]p_0p_1 \ldots p_T")))) \tag{2.6}$$

The late interaction mechanism applied to the encodings involves computing the maximum similarity, which utilizes cosine similarity through dot-products. This is made possible by the earlier normalization applied to the embeddings:

$$\mathbf{s}_{q,p}^{ColBERT} = \sum_{I \in [|\mathbf{E}_q|]} \max_{j \in [|\mathbf{E}_d|]} \mathbf{E}_{q,i} \cdot \mathbf{E}_{p,j}^{\top} \tag{2.7}$$

The interaction mechanism has no trainable parameters. ColBERT is differentiable end-to-end. During training, for example, with a triple (q, p^+, p^-), ColBERT independently produces a score for each passage and is subsequently optimized pairwise using softmax cross-entropy loss over the scores of p^+ and p^- [Khattab and Zaharia, 2020].

Another type of Retriever is the **Iterative Retriever**. Iterative Retrievers are necessary when dealing with questions that are more complex than simple factoid questions, which can be answered by identifying the right passage in the knowledge source. An example is the HotpotQA dataset [Yang et al., 2018], designed specifically for multi-hop questions. The fundamental concept here is that such questions cannot be answered with just one precise piece of evidence. They require multiple passages from different documents at the very least. Iterative Retrievers encompass three stages in the pipeline: (1) document retrieval, (2) query reformulation, and (3) retrieval stopping.

An example is BEAM, currently holding the title of the highest-performing[1], QA-System across multi-hop QA datasets such as HotpotQA [Zhang et al., 2023a]. The document retrieval component can take the form of any retrieval model, including options like ColBERT, BM25, or DPR. In the case of BEAM, it leverages an Interaction-Based Retriever using DeBERTa. For each candidate passage p_c, BEAM calculates a relevance score concerning this passage within the context of all previously identified relevant passages p_r and the question q, using the embeddings of the `[CLS]` tokens [He et al., 2020]. The second step, query reformulation, can be executed explicitly or implicitly, meaning it can either be expressed in natural language or as a dense embedding. The advantage of using natural language lies in its interpretability, while employing dense embeddings operates within a semantic space and does not lack vocabulary interpretability [Zhu et al., 2021]. BEAM adopts a natural language-based approach. Specifically, after each hop, it appends the newly identified passage to the previously identified ones and feeds this information into DeBERTa.

$$\mathrm{s}_{q,p}^{BEAM} = \text{Classifier}(\text{DeBERTa}(\text{"}[CLS] q\, p_{r_1} \,\ldots\, p_{r_i}\text{"})) \quad | \quad p_c \in P \tag{2.8}$$

The nature of query reformulation depends on the type of retriever in use. Lastly retrieval stopping poses its own set of challenges. A common approach involves setting either a fixed number of hops or a maximum limit on the retrieved documents. Alternatively, some methods introduce a new token, such as `[EOE]` (End-of-Evidence), to signal the end of retrieval [Zhu et al., 2021]. BEAM, for example, employs a fixed number of hops, specifically 2, as determined through empirical evaluation.

[1]Status as of September 23, 2023, according to https://paperswithcode.com and the authors of [Zhang et al., 2023a]

The task of **Document Post-Processing** is to reduce the number of passages forwarded to the Reader, aiming to eliminate irrelevant ones. Traditional Retrievers, like Sparse Retrievers, often required a Document Post-Processor. However, Dense Retrievers often incorporate ranking and retrieval simultaneously, rendering this module unnecessary [Zhu et al., 2021]. Nevertheless, it remains possible to construct multi-stage Retrievers to, for instance, increase latency. This can be achieved by using a simpler Dense Retriever for pre-filtering passages and subsequently applying a more accurate one [Liu et al., 2021b].

2.1.5. Reader Approaches

Readers originally emerged from the field of MRC, where the objective is to extract an answer from a given context. A well-known example is the SQuAD [Rajpurkar et al., 2016] dataset, which was mentioned in Section 2.1.1. However, unlike the original MRC task, a Reader in a Retrieval-Reader-System must process multiple passages to determine the relevant information needed to answer a given question [Zhu et al., 2021].

Modern readers rely on Transformer-based Pre-trained Language Model (PrLM)s since they establish new baselines on well-known datasets [Luo et al., 2022]. In general, there are two types of Readers that use PrLMs: **Extractive Readers** and **Generative Readers** [Jurafsky and Martin, 2023, Zhu et al., 2021, Luo et al., 2022].

In general, an **Extractive Reader** employs an encoder to identify the token sequence span that is relevant for answering a question. These encoders can be any autoencoder models, such as BERT [Devlin et al., 2019], DeBERTa [He et al., 2020], or RoBERTa [Liu et al., 2019]. Luo et al. [Luo et al., 2022] even utilized the encoder components of established encoder-decoder models like T5 [Raffel et al., 2023] and BART [Lewis et al., 2019]. They demonstrated that, after fine-tuning, these models can outperform encoder-only models on certain tasks.

Figure 2.4 illustrates the span labeling process performed by the extractive reader. The question tokens $q_1, \ldots, q_n$ and the passage tokens $p_1, \ldots, p_m$ are input into the encoder, separated by a `[SEP]` token. The encoder learns two new embeddings, S and E, which represent span-start and -end tokens, respectively. To obtain the span start probability for an output token p'_i, the dot product between the output token and S is computed and then normalized by a softmax function over all output tokens. The process is similar for the span-end token. The score of a span from position i to j is calculated as $S * p'_i + E * p'_j$. The span with the highest score, where $j \geq i$, is selected as the answer span. If the total length of tokens in q and p exceeds the maximum input length of the encoder, the passage is split into multiple segments, and the process is

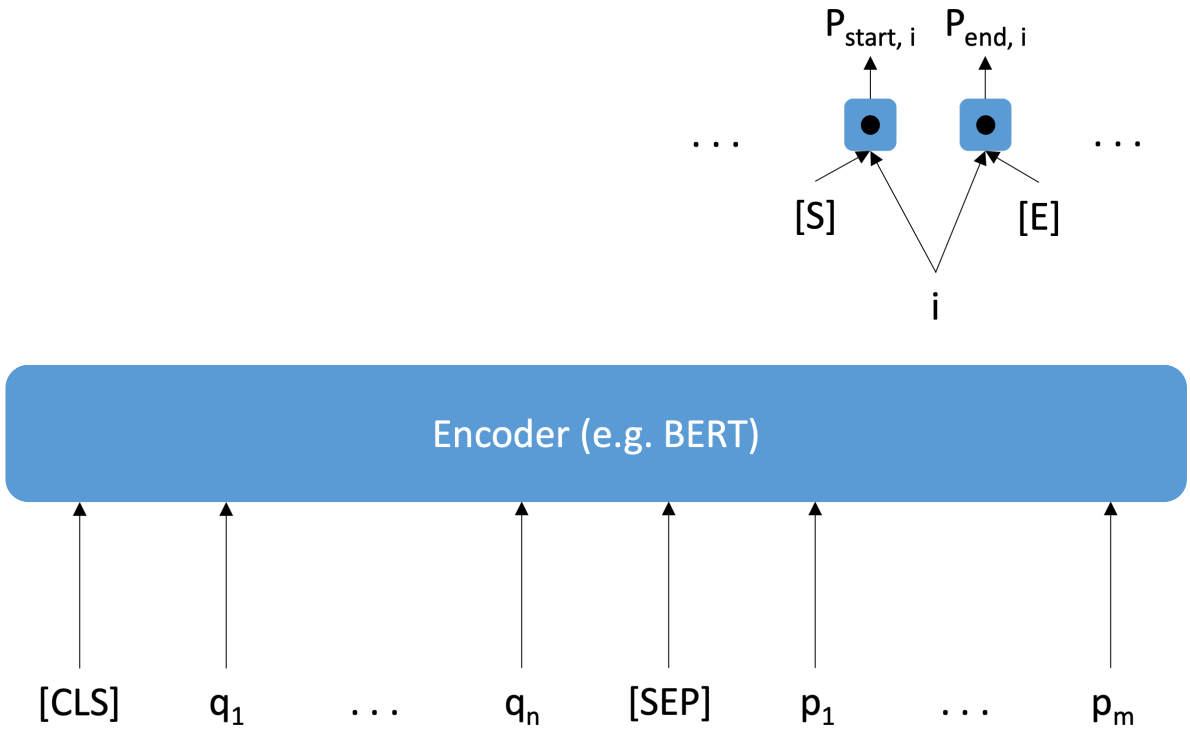

Figure 2.4.: Adjusted Graphic of the Extractive Reader by Jurafsky et al. [Jurafsky and Martin, 2023]

repeated for each segment [Jurafsky and Martin, 2023, Luo et al., 2022].

The **Generative Reader** operates straightforwardly when familiar with a seq-2-seq encoder-decoder model. Given a dataset containing (q, p, a) tuples, the encoder takes q and p as input and outputs the contextual representation h. Then, it is the decoder's task to generate a token sequence based on h and attention. The training objective can be described as minimizing the following loss function:

$$\mathcal{L}_{\text{Gen}} = \sum_{\mathbf{i=1}}^{\mathbf{K}} \log \mathbf{P}\left(\mathbf{a_i} \mid \mathbf{h}, \mathbf{a}_{:\mathbf{i}}\right) \tag{2.9}$$

Here, K represents the length of tokens in a, a_i is the i^{th} token in a, and a_0 is a special beginning of sequence token. In cases where the answer is not contained within the passages, the `[CLS]` token indicates this situation [Luo et al., 2022, Zhu et al., 2021].

Latest research projects like Visconde [Pereira et al., 2022] even employ LLM as Generative Readers. The performance and usability of these models remain active topics of research.

Luo et al. conducted the first survey comparing state-of-the-art Extractive and Generative Readers [Luo et al., 2022]. They discovered that "on average, extractive readers

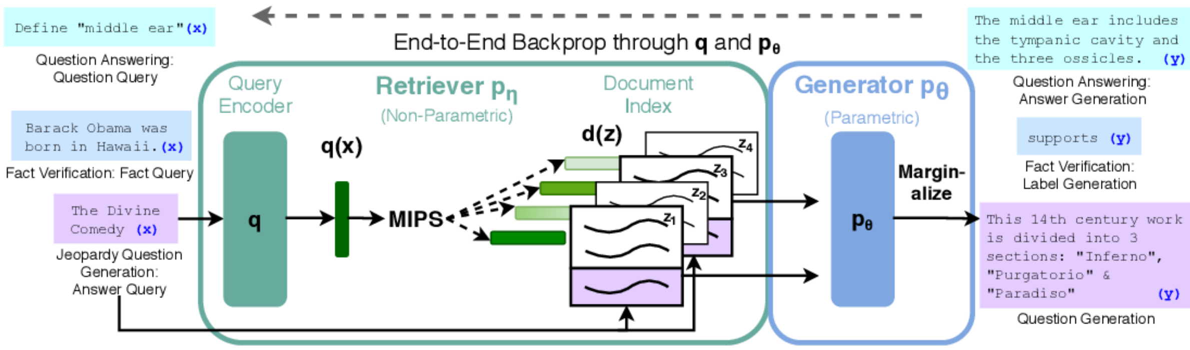

Figure 2.5.: Overview of RAG by Lewis et al. [Lewis et al., 2021]

perform better than generative ones" [Luo et al., 2022], except in cases involving long context passages, where generative approaches outperform the extractive ones.

RAG can be seen as a generative reader, but with a much more capable NN as the reader, specifically the idea is that the reader itself is a LLM with implicit knowledge encoded in its parameters, which it uses to generate an answer, the retrieved passages intentionally function as support in order to guide the reader an reduce risk of hallucination.

Figure 2.5 is taken from the original paper by Lewis et al. [Lewis et al., 2021] and displays the general approach of RAG. The original idea of RAG is to have an end-to-end backpropagation in order to train the retriever and reader (generator) at once and on the same data, not seperate as in most Retriever-Reader-Systems. The used retriever in the original RAG is a DPR. Other retrievers can be used, as this is just a decission to make as the generator does not directly depend on the type of retriever. More interestingly is the kind of implementation of the generator. RAG implements a *sequence-based* generator, while future work, such as FiD [Izacard and Grave, 2021] use an *attention-based* generator. The sequence-based generator works the following way: Given an arbitrary encoder-decoder $p_\theta(y_i|q, p, y_{1:i-1})$, the query q, the k-relevant passages p, and the previously generated tokens $y_{1:i-1}$, the generator computes the probability distribution over the next token y_i. q and p where simply concatenated.

Further RAG generators are attention-based like FiD [Izacard and Grave, 2021]. Here the encoder and decoder of the generator are slightly decoupled as to the classic RAG. Given a question q, the retriever retrieves the top-k passages p. The encoder ecodes every sinlge passage in a question, title, passage triple (q, t, p). The encodings of multiple passages are afterwards concatenated and passed into the encoder-decoder attention of the decoder. An illustration can be found in Figure 2.6 This allows for the combination of multiple passages, so there is no input token limitation as for the classical RAG. Also experiments by Izacard et al. showed, that the performance imrpoves over multiple

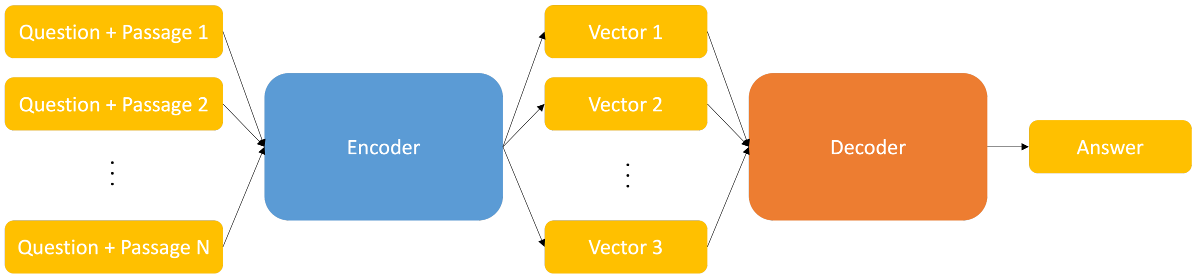

Figure 2.6.: Overview of FiD by Izacard et al. [Izacard and Grave, 2021]

tasks, as multi-passage relations can easily be resolved by the decoder. The latest work of Izacard et al. is ATLAS, which set new state-of-the-art benchmarks on multiple evaluation tasks [Izacard et al., 2022]. ATLAS extends on the idea of FiD.

Still over all RAG approaches, the main idea is to have a fully end-to-end backpropagation during training or fine-tuning of the systems.

The **Answer Post-Processor** is similar to the Document Post-Processor, serving as an optional component. Its primary task is to provide support for multi-hop complex questions, helping determine the final answer from a set of answers extracted by the reader component [Zhu et al., 2021]. Depending on the implementation of the Reader, this component may become obsolete.

2.1.6. Limitations

The evaluation metrics for IR systems will be discussed in detail in Section 4. In general, selecting the components and models for an IR system always involves a trade-off between accuracy, memory consumption, and inference speed [Zhang et al., 2023b].

Accuracy is primarily determined by the chosen Retriever-Reader-System. Sparse retrievers often lack a certain degree of semantic understanding, resulting in less accurate retrieved passages. In contrast, Dense Retrievers can achieve higher levels of accuracy but require thorough evaluation and training for the desired use case. Thakur et al. demonstrated that high-accuracy Dense Retrievers like DPR can underperform in zero-shot scenarios compared to BM25 by -47.7% [Thakur et al., 2021]. This highlights another crucial limitation of all NN-based retrievers and readers: training. BM25 is, by nature, an unsupervised model for IR, while common approaches for Dense Retrieval usually belong to the group of supervised models. These models heavily depend on their training data, whereas a Sparse Retriever like BM25 can be used without any training. According to experiments conducted by Thakur et al. [Thakur et al., 2021], the best-performing out-of-distribution Retrievers are Representation-Interaction Retrievers like ColBERT.

Constructing a training dataset for a QA task can be a tedious process, as these datasets must consist of tuples in the form of (question, context, answer), which is not always feasible. One established research direction to address this issue is Automatic Question Generation (QG) [Serban et al., 2016]. In QG, a seq-2-seq model is employed to generate questions and answers based on a given passage.

Zhang et al. provide an example of DPR applied to the Natural Questions dataset in their survey on efficient ODQA [Zhang et al., 2023b]. The total processing time for a query is 0.91 seconds[2]. This time is divided into 74.79% for evidence search and 23.95% for reading. The total memory cost is 79.32GB, with the index occupying 81.95%, the raw corpus 16.39%, and the model 1.66%. Approaches to optimize this may include:

1. Reducing Processing Time: (1) Accelerating Evidence Search, (2) Accelerating Reading

2. Reducing Memory Cost: (1) Reducing Index Size, (2) Reducing Corpus Size, (3) Reducing Model Size

3. One-stage Frameworks: (1) Directly Generating Answers, (2) Directly Retrieving Answers

Techniques used in this context may include:

1. Data-based: (1) Passage Filtering, (2) Dimension Reduction, (3) Product Quantization

2. Model-based: (1) Model Pruning, (2) Knowledge Distillation, (3) Knowledge Source

A common technique, which is used in nearly every experimental setup for QA-Systems, is FAISS[Johnson et al., 2017], a GPU optimized implementation of the exact k-means clustering algorithm.

For a detailed overview of approaches towards more efficient ODQA systems, please refer to the comprehensive survey by Zhang et al. [Zhang et al., 2023b].

2.2. Conversational Question Answering

The differentiation of Conv QA towards QA will be discussed in Section 2.2.1. This Section also introduces the fundamental concepts of Conv QA which are necessary to

[2]It's important to mention that DPR is a Representation-based Retriever, which allows offline storage of passage embeddings. The result was obtained using an Nvidia GeForce Rtx 2080 Ti GPU, averaged over 1000 examples

understand challenges and necessary components compared to a regular QA-System. Section 2.2.2 will cover approaches towards the concept of query expansion. Section 2.2.3 will clampse on the concept of initative and further approaches towards a Conversation Manager. Lastly Section 2.2.4 will cover the usability of LLMs and especially the concept of Chain of Thoughts for Conv QA.

2.2.1. Basics

Core concepts in the field of Open-Domain Conversational Question Answering (ODCQA) towards a conversation in terms of Conv QA are: *Turns*, *Hisotry*, *Memory*, *Session* and *Dialog Features* and *Dialog State* [Zamani et al., 2023]. It's important to mention, that in other subdomains/-tasks of CIS more concepts are introduced, such as *State*, those are not necessary or applicable for ODCQA [Zaib et al., 2021].

Figure 2.7 shows the core concepts based on a chat. Firstly, a **Turn** is a question-response pair. Whereas a conversation usually consists of multiple turns (multi-turn). Conversational Question Answering (CoQA) is a dataset published in 2019 by researchers at Stanford in order to extend the known QA dataset SQuAD towards a conversational dataset, whereas on average one conversation session consists of 15 turns [Reddy et al., 2018]. Multi-turns are the main distinguisher betwenn the in Section 2.1 introduced task, to a Conv QA task. In a multi-turn scenario natural language phenomena like *coreference* (multiple expressions refereing to the same thing) or *ellipsis* (omitting words or topics implied by the context) can occur. While in regular QA the System will only be challenged with single-turn scenarios, so only one question, which needs an answer, in Conv QA the systems have to face multi-turn scenarios, where a user might also ask followup question or in general mutliple questions aftereachother. A **Hisotry** is consequently a set of turns which belong to one conversation session. A **Session** is a in it completed conversation. Lastly, the **Memory** is the abstract entity in which the Conv QA-System stores knowledge related to a history, session or even user in general [Zamani et al., 2023, Gao et al., 2022]. This depends of the implementation of memory in the Conv QA pipeline, which will be discussed in Section 2.2.2.

Dialog Features need to be assesed extra to the other mentioned concepts. While the other concepts tackle the conversations frame, the dialog feature evaluates the user questions themself. Possible dialog features may include: *drilling-down* questions, *topic-shift*, *clarification* or *definition*. Different dialog features call for different responses by the system [Gupta et al., 2020]. The **Dialog State** has to be assesed similar. The dialog state represents the relation between turns. In cases of pre-defined domains methods like state slots are used, e.g. `Date _`, `Location _`, `Artist _` have to be

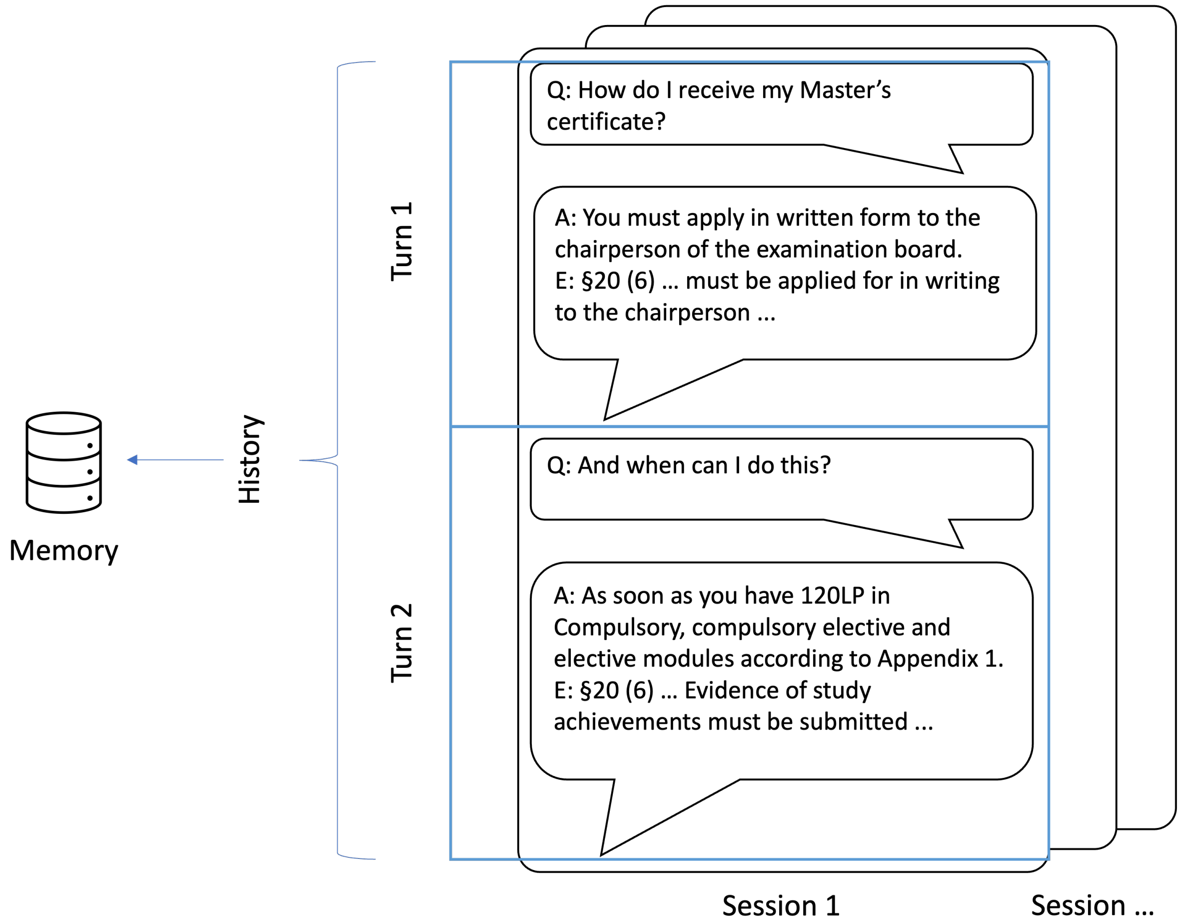

Figure 2.7.: Concepts of a Conversation in regards to a CIS

filled during the conversation in order to retrieve the correct information from the KB [Rastogi et al., 2020]. Open-Domain Conv QA usually don't track the state *explicitly*, but rather track it *implicitly* via the type of implementation of the *Contextual Query Understanding* unit.

Regarding the System architecture of a Conv QA there is no one fits them all solution at the moment, but Gao et al. [Gao et al., 2022] presented a modern system architecture, which represents commonly used appraoches and their corresponding components in a genral fashion. This general architecture can be observed in Figure 2.8. Modern Conv QA systems are closely realted to QA systems, but lag certain generalizing components in order to be full CIS systems [Zamani et al., 2023].

Similar to the retriever-reader architecture introduced in Section 2.1.2, a Conv QA is made up of those two components as well, whereas in the case of a Conv QA the retrieval as also the reader component have to handle more. The retriever has to understand the context, so the history of a conversation and retrieve based on that the

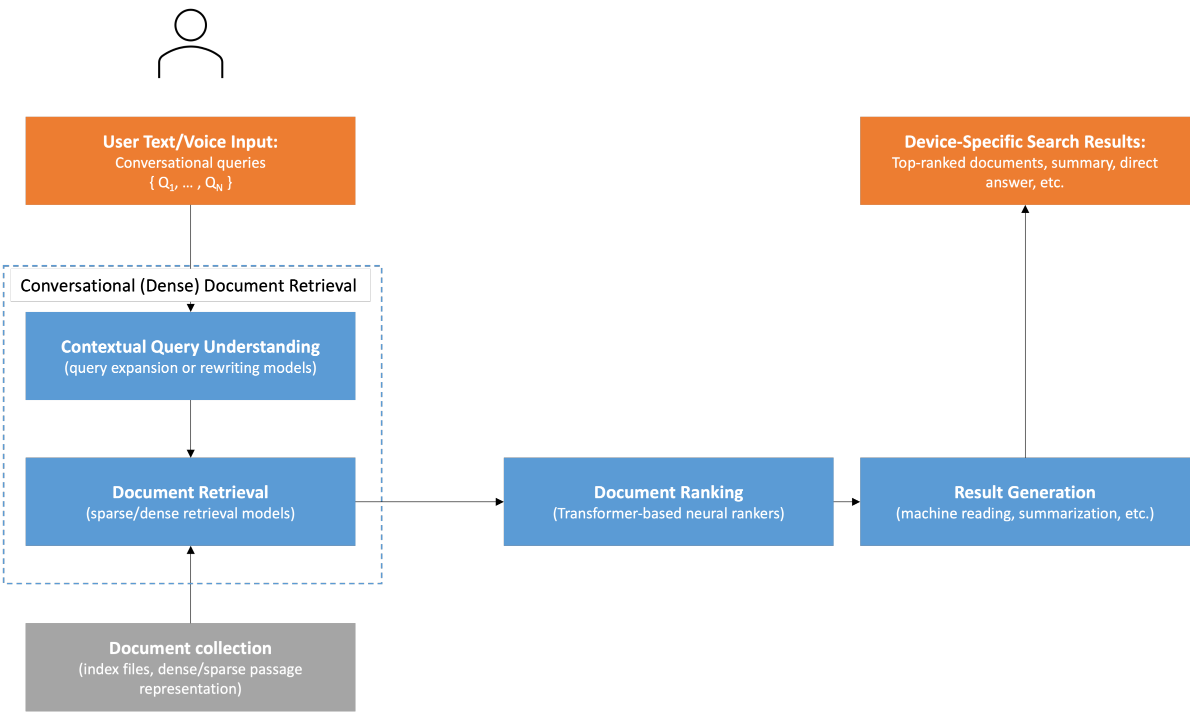

Figure 2.8.: General System Architecture of a Conv QA System by Gao et al. [Gao et al., 2022]

most relevant documents. The reader on the other hand is close related to the reader of a classic retriever-reader architecture [Zamani et al., 2023, Gao et al., 2022]. Some implementations even feed into the reader component the context in order to rank the retrieved passages better and generate a more accurate answer [Owoicho et al., 2022].

2.2.2. Contextual Query Understanding

How do we implement *Memory*? Is the core question of this Section 2.2.2.

There are two main distinguishing appraoches towards history implementation. The first is a simple heuristic of using the last-k turns for **Query Expansion**, **Query Rewriting** or **Conversational Retrievers**. The second is to extract the important parts of the history in regards to a question and use them for Query Expansion or Rewriting [Gao et al., 2022].

A good example to explain the appraoch of the second approach towards extracting improtant parts of the history $H = (q_1, a_1), \ldots, (q_i, a_i)$ given a new question q_{i+1} is **Qu**ery **Re**solution by **Te**rm **C**lassification (QuReTeC)[Voskarides et al., 2020]. QuReTeC consits of two components essentially: one BERT-based model and a trainable classification layer. The H is being passed through the BERT model, whereas the following structur of concatination is being used:

$$\text{BERT}([CLS], H, [SEP], q_{i+1}) \tag{2.10}$$

On every first sub-token of a term of the H the term classification layer is applied, which is a network consisting of a dropout layer, a linear layer and a sigmoid function. The term classification layer predicts a label between $0 - 1$ indicating it's importance for answering the new question q_i. This leads to a set of terms I which need to be incorporated into the retrieval [Voskarides et al., 2020]. This is generaly also known as **Query Expansion**, whereas we add terms to a given query for retrieval. Next to this supervised, trained approach, there are also implementations which work unsupervised like Historical Query Expansion (HQExp) [Yang et al., 2019], which was one of the best performing models in the TREC CAsT 2019 [Dalton et al., 2020].

Modern neural appraoches more often implement a **Query Rewriting** module which is build on top of seq-2-seq-models to rewrite a query q_{i+1} given a history H in order to use the generated new query for retrieval using an established QA retriever [Owoicho et al., 2022]. The main advantage of this appraoch is the abscence of the need of large supervised datasets as for Conversational Retrievers [Dai et al., 2022a]. One of the top performing models in the TREC CAsT 2022 was HEATWAVE by a Team of the University of Cambridge England [Liusie et al., 2022]. HEATWAVE utilized a query rewriter and a classical lexical BM25 retriever in combination with a BERT-based re-ranker. The rewriter uses a T5-based Transformer model and gives as input $ctx - n - m$, where n referes to the last n-many user utterances and m to the m-many system responses. In general the task can be simply broken down to the follwoing:

$$q_{rewritten} = \text{Rewriter}(ctx - n - m) \tag{2.11}$$

For training of this model they used the among others the canard dataset [Elgohary et al., 2019] a manually annotated version of the QuAC dataset, specifically for the task of query rewriting given a conversation history H.

State-of-the-art reasearch utilizes more and more LLM for the task of Query Rewriting, as they can handle long context histories H and are in general strong zero- or few-shot models [Mao et al., 2023]. This is also the main approach frameworks like *Langchain* [Langchain, 2023] or *ChatGPT by OpenAI* [OpenAI, 2023] use.

Lastly the appraoch of **Conversational Retrievers** exists. Those use compared to classical QA retrievers not a pair of (q, p) in order to calculate a similarity $sim(q, p)$ between the question q and the passage p, but use Conv QA datapoints from conversational interactions like $(q_1, a_1, \ldots, q_i, a_i, q_{i+1}, p)$, in short (H, q_{i+1}, p), so combining a conversation history H with a new question q_{i+1} and the relevant passage p to answer this question given the history H [Gao et al., 2022, Dai et al., 2022a]. High performing zero-shot or subdomain-adapted Conversational Retrievers do not exist, as it is extremly time consuming to create a dataset for this type of Retriever. To close this gap in researchers proposed sufficient data augmentation techniques to generate those datasets, given a document. One example is the work of Dai et al. [Dai et al., 2022a] which introduced the technique of "Dialog Inpainting" [Dai et al., 2022a].

2.2.3. Initiative

All modern human-computer interactions follow a one-sided initative model, where either the user- or the system-initative is given. In mixed-initative scenarios of Conv QA the system can take initative without explicit comands of the user. Examples for initative are: *Topic Shifts*, *Clarification Questions* or *Question Recommendations* [Zamani et al., 2023]. In this thesis we will focus on *Clarification Questions* only.

2.2.4. Large Language Model based Agents / Chain of Thought ???

2.3. Efficient Large Language Models

With the increasing size of large language models (LLM), **L**arge **La**nguage **M**odels with **A**dapters (Llama) 2 offers models ranging from 7 billion to 70 billion parameters [Touvron et al., 2023]. Even these models are considered relatively small compared to the largest models like PaLM 2 [Anil et al., 2023] with 340 billion parameters. The challenge arises when running such models in scenarios with limited computational resources, especially on smaller domains or tasks. This challenge is particularly relevant to the task presented in this thesis, which involves building a Conv QA system for a custom set of PDFs.

While several surveys [Ling et al., 2023, Zhao et al., 2023] have explored the topic of efficient LLM usage in resource-constrained systems, Treviso et al. [Treviso et al., 2023] present the most comprehensive taxonomy of methods and approaches in this context. Figure 2.9 provides a high-level overview of the stages at which efficiency-improving methods can be implemented in LLMs. Given the specific focus of this thesis, not all stages will be discussed in detail. For more comprehensive insights, please refer to the original survey by Treviso et al. [Treviso et al., 2023].

Section 2.3.1 will explore possibilities to enhance efficiency during the fine-tuning process, while Section 2.3.2 will delve into the topic of model compression, which is

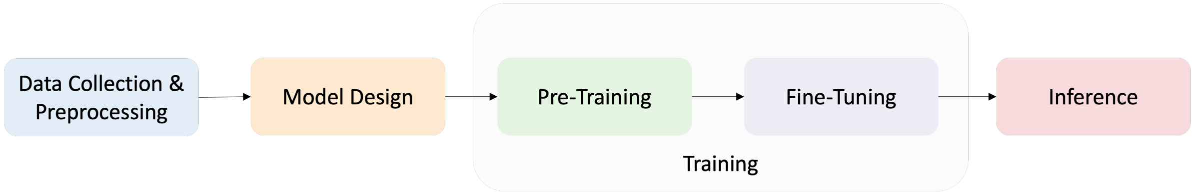

Figure 2.9.: Adapted Stages of Efficiency Improvement for LLM by Treviso et al. [Treviso et al., 2023]

applicable to the *Inference* step in Figure 2.9.

2.3.1. Fine-Tuning

Hu et al. [Hu et al., 2021] demonstrated the significant benefits of fine-tuning GPT-3 for few-shot applications, highlighting the remarkable improvements fine-tuning can achieve. This is further supported by the experiments conducted by Chung et al. [Chung et al., 2022].

Efficient fine-tuning of LLMs can be categorized into three distinct approaches: *Parameter Efficiency*, *Multi-task Learning*, and *Prompting*. Figure 2.10 provides an overview of these approaches along with their corresponding methods.

Parameter Efficiency is commonly referred to as **Parameter-efficient Fine-tuning (PEFT)** [Huggingface, 2023]. A notable PEFT approach is LoRa, developed by Hu et al. [Hu et al., 2021]. LoRa falls under the category of Adapters, a term coined because it revolves around the concept of freezing the parameters of the LLM and fine-tuning only a small set of task-specific parameters, which can be swapped depending on the desired downstream task. Unlike some other Adapter-based methods [Houlsby et al., 2019], LoRa does not introduce additional inference latency due to the merging of trainable matrices with frozen weights. Moreover, LoRa can be seamlessly combined with many other PEFT methods.

In practice, given a pre-trained weight matrix $W_0 \in \mathbb{R}^{d\times k}$, which is typically full-rank between layers, the update can be constrained to be a low-rank composition: $W_0 + \triangle W = W_0 + BA$, where $B \in \mathbb{R}^{d\times r}$, $A \in \mathbb{R}^{r\times k}$, and the rank $r \ll \min(d, k)$. While W_0 remains frozen during training, A and B become trainable parameters. The forward pass $h = W_0 x$ can be represented as the following sum:

$$h = W_0 x + \triangle W x = W_0 x + BAx \tag{2.12}$$

Figure 2.11 illustrates the architecture and initialization during training. The parameters of A are randomly sampled using Gaussian initialization, while B is initialized to

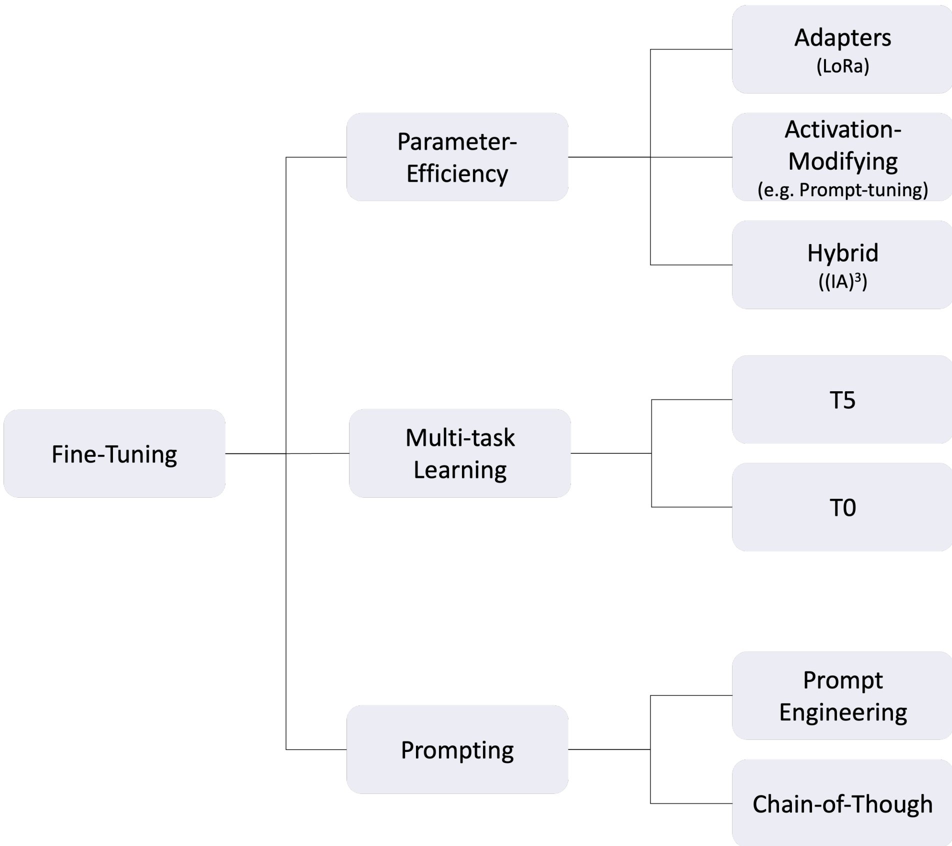

Figure 2.10.: Adapted Fine-Tuning Approaches for LLM by Treviso et al. [Treviso et al., 2023]

0.

Other PEFT techniques include **prompt-tuning** [Lester et al., 2021] and **prefix-tuning** [Li and Liang, 2021]. Both approaches are similar in the way they leverage task-specific modifications to the input to guide the model's behavior. They involve concatenating learned vectors to activations or embedding sequences, making them activation-modifying PEFT methods.

A PEFT approach that can be considered a hybrid between LoRa and activation-modifying techniques is $\mathbf{(IA)}^3$ [Liu et al., 2022]. What sets $(\mathrm{IA})^3$ apart is its focus on LLMs designed explicitly for multi-task learning, as all existing PEFT techniques significantly underperformed in experiments conducted by Liu et al. [Liu et al., 2022]. In $(\mathrm{IA})^3$, the model's activations are rescaled using element-wise multiplication with learned vectors, known as adaptors. Specifically, $(\mathrm{IA})^3$ employs three learned vectors: $l_k \in \mathbb{R}^{d_k}$ for keys and $l_v \in \mathbb{R}^{d_v}$ for values in self-attention and encoder-decoder attention mechanisms, as well as $l_{ff} \in \mathbb{R}^{d_{ff}}$ for the feed-forward network. The rescaling is incorporated into the attention mechanism as follows:

$$\operatorname{softmax}\left(\frac{Q\left(l_k \odot K^T\right)}{\sqrt{d_k}}\right)(l_v \odot V) \tag{2.13}$$

For the feed-forward network, the rescaling is implemented as follows, where γ represents the feed-forward activation:

$$(l_{ff} \odot \gamma(W_1 x))W_2 \tag{2.14}$$

In summary, $\mathbf{(IA)}^3$ is a PEFT approach specifically designed for multi-task learning, and it appears to outperform LoRa in terms of the number of parameters added and training computation costs.

Multi-task Learning refers to the concept of fine-tuning a PrLM on various tasks to achieve better zero-shot and fine-tuning performance. One of the most prominent PrLMs trained on multiple NLP tasks is T5 [Raffel et al., 2023]. Liu et al. [Liu et al., 2022] demonstrated that a generically trained T5 model (T0) can be few-shot fine-tuned with approximately 10% of the parameters of LoRa, at a lower computational cost, while achieving higher accuracy in classification tasks[3]. This is made possible, due to using $(\mathrm{IA})^3$ as PEFT and the multi-task pre-training of the used model. Still this also includes the drawback of having in general a larger model.

Prompting refers to the general concept of presenting a task as a textual instruction to a LLM [Brown et al., 2020]. Recent advances have even led to the development of

[3]Currently, there is no experiment comparing LoRa and $(\mathrm{IA})^3$ on QA tasks

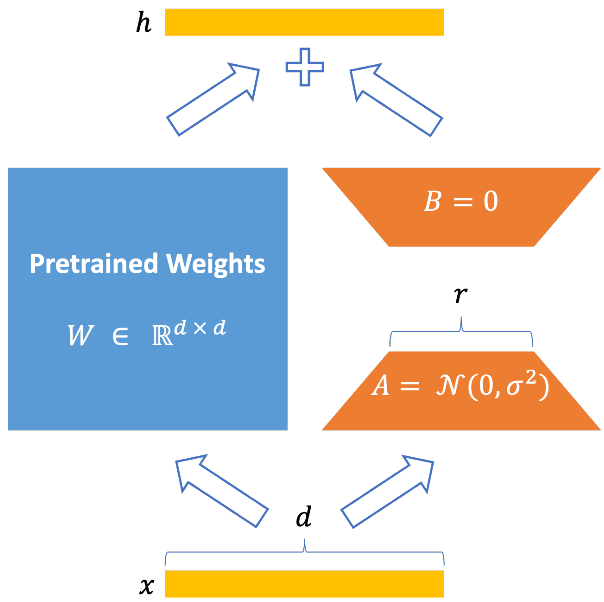

Figure 2.11.: LoRa by Hu et al. [Hu et al., 2021]

a new sub-task and job role called *Prompt Engineering* [White et al., 2023]. It's worth noting that different prompts with the same intent can yield different results, making the selection of the right prompt a challenge in itself [Liu et al., 2021a]. Furthermore, concepts like chain-of-thought (CoT) prompts have been developed. In CoT prompts, the given example in a Few-shot prompt is redesigned to mimic step-by-step reasoning and conclusions known from the way humans think, aiming to achieve higher performance in Zero- and Few-shot scenarios simply by adjusting the explicit natural language prompt [Wei et al., 2023].

2.3.2. Compression

Compression aims to reduce the size of a model, whether it's the number of parameters, while maintaining the same level of accuracy on the downstream task, or the actual storage required for the model. There are three primary approaches to this task: *Pruning*, *Knowledge Distillation*, and *Quantization* [Treviso et al., 2023, Zhu et al., 2023]. Figure 2.12 provides an overview of these approaches and their corresponding methods.

Pruning can be further categorized into *structured* and *unstructured* pruning. Structured pruning involves removing specific patterns of weights or activations from a model, with the goal of maintaining a dense matrix representation to ensure compatibility with existing implementations and hardware. One notable example of a structured pruner for LLMs is LLM-Pruner [Ma et al., 2023]. On the other hand, unstructured pruning entails removing individual weights or activations from a model, resulting in a sparse

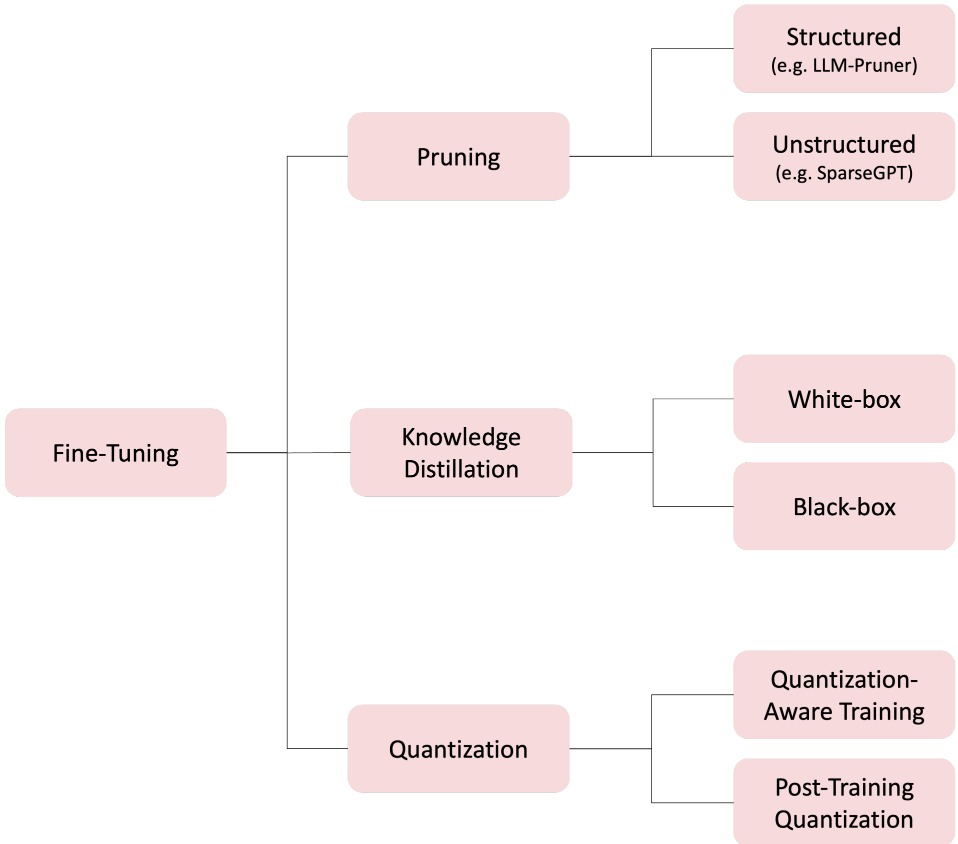

Figure 2.12.: Adapted Compression Approaches for LLM by Treviso et al. [Treviso et al., 2023]

matrix representation. This approach may require specialized hardware or software implementations to efficiently compute and achieve speed improvements of 1.5× to 2.16×, while reducing up to 60% of the parameters [Frantar and Alistarh, 2023]. Examples of engines designed specifically for unstructured pruning include NVIDIA's CUTLASS library for GPUs [Frantar and Alistarh, 2023] and DeepSparse [noa, 2023a] for CPUs.

Knowledge Distillation is an approach that involves using a generally well-performing LLM as a teacher to instruct a significantly smaller student model [Hinton et al., 2015]. Zhu et al. distinguish between *White-box* and *Black-box* knowledge distillation. In the former, the student has full access to the teacher's parameters, while in the latter, only the teacher's predictions are accessible to the student [Zhu et al., 2023]. An example of white-box knowledge distillation is MiniLLM [Gu et al., 2023], where the distribution of the final layer's outputs for both the teacher and the student, given a prompt, is compared using the Kullback-Leibler divergence. This comparison is used in a loss function for backpropagation in the student model.

Black-box approaches are more commonly used in knowledge distillation. In these cases, a LLM is employed to either directly provide its predictions based on a prompt [Huang et al., 2022], offer assisting explanations [Li et al., 2022c], or sort the training data by difficulty and artificially generate more data points [Jiang et al., 2023], among other techniques. For a comprehensive overview, please refer to Section 2.2 *Knowledge Distillation* in Zhu et al.'s survey [Zhu et al., 2023]. Experiments with different distillation approaches have shown that distillation has its limitations, and for specific downstream tasks, fine-tuning can outperform knowledge distillation [Zhu et al., 2022].

Quantization is an approach that involves reducing the datatype representation of weights or activations, which are typically floating-point numbers, to smaller representations in terms of bits, such as 8-bit integers or even smaller discrete formats [Gholami et al., 2021]. Generally, there is a distinction between *Quantization-aware Training* and *Post-Training Quantization*. The names are self-explanatory; the former involves applying and adjusting quantization during the training process (either pre-training or fine-tuning) [Liu et al., 2023], while the latter pertains to quantization after the training is completed [Frantar et al., 2023a]. In both cases, numerous approaches and methods exist, applying different paradigms and quantization techniques, including decisions regarding which parameters to quantize, structured vs. unstructured quantization, quantization strength, and many others. It's not possible to discuss all of these here, but for a comprehensive overview, please refer to Section 2.3 *Quantization* in Zhu et al.'s survey [Zhu et al., 2023].

The most prominent example of Post-Training Quantization is GPTQ, which is the only ready-to-use implementation available in the Huggingface Transformer Library [noa, 2023b], and for a good reason. GPTQ was the first method to achieve high compression for LLMs with over 175 billion parameters, while maintaining high accuracy compared to prior state-of-the-art algorithms. Specifically, with a 4-bit quantization of the weights, GPTQ achieved approximately 5× compression for BLOOM-176B and OPT-175B, two openly available LLMs, while experiencing only a ≤ 0.25 decrease in perplexity compared to the original model. Therefore, the following section will explain the Post-Training Quantization approach of GPTQ in detail.

GPTQ builds upon Optimal Brain Quantization (OBQ), the previous work by Frantar et al. on Post-Training Quantization [Frantar et al., 2023b]. With GPTQ, their objective was to reduce the runtime complexity of OBQ, which is $O(d_{row} * d_{col}^3)$, making it compatible with LLMs containing billions of parameters. The central idea behind GPTQ is a layer-wise optimization approach. The aim is to discover quantized weights $\widehat{\mathbf{W}}$ that minimize the squared error compared to the full-precision layer $\mathbf{W}$ output, using a given set of input data points $\mathbf{X}$:

$$\operatorname{argmin}_{\widehat{\mathbf{W}}} \|\mathbf{W}\mathbf{X} - \widehat{\mathbf{W}}\mathbf{X}\|_2^2 \tag{2.15}$$

In OBQ, we denote the next weight to be quantized as w_q. We define the function $\operatorname{quant}(w)$, which rounds a weight w to the nearest value on the quantization grid.

$$w_q = \operatorname{argmin}_{w_q} \frac{(\text{ quant } (w_q) - w_q)^2}{\left[\mathbf{H}_F^{-1}\right]_{qq}} \tag{2.16}$$

In the context of **GPTQ**, a column of weights is always updated simultaneously. Therefore, $\operatorname{quant}(W_{:,j})$ refers to the following:

$$\operatorname{quant}(W_{:,j}) := \forall\, w_q \in W_{:,j} \tag{2.17}$$

The Hessian matrix $\mathbf{H_F} = 2X_F X_F^T$ is utilized for both weight updates and quantization error calculations. Once all columns within a block B are quantized, the weight update is computed as follows, where Q represents the set of indices corresponding to quantized weights:

$$\boldsymbol{\delta}_F = -\left(\mathbf{w}_Q - \text{ quant } (\mathbf{w}_Q)\right)\left(\left[\mathbf{H}_F^{-1}\right]_{QQ}\right)^{-1}\left(\mathbf{H}_F^{-1}\right)_{:,Q} \tag{2.18}$$

Furthermore, the Hessian is updated in the following manner, avoiding the need for recomputation; instead, columns corresponding to quantized weights are simply dropped from the Hessian.

$$\mathbf{H}_{-Q}^{-1} = \left(\mathbf{H}^{-1} - \mathbf{H}_{:,Q}^{-1} \left[\left(\mathbf{H}^{-1} \right)_{QQ} \right]^{-1} \mathbf{H}_{Q,:}^{-1} \right)_{-Q}. \tag{2.19}$$

This leads to the following algorithm:

Algorithm 1 Quantize W given inverse Hessian $H^{-1} = (2XX^T + \lambda I)^{-1}$ and blocksize B by Frantar et al. [Frantar et al., 2023a]

$Q \leftarrow \mathbf{0}_{d_{\text{row}} \times d_{\text{col}}}$ ▷ Quantized output
$E \leftarrow \mathbf{0}_{d_{\text{row}} \times B}$ ▷ Block quantization errors
$H^{-1} \leftarrow \text{Cholesky}(H^{-1})^T$ ▷ Hessian inverse information
for $i \leftarrow 0, B, 2B, \ldots$ **do**
  for $j \leftarrow i, \ldots, i+B-1$ **do**
    $Q_{:,j} \leftarrow \text{quant}(W_{:,j})$ ▷ Quantize column
    $E_{:,j-i} \leftarrow \frac{W_{:,j} - Q_{:,j}}{[H^{-1}]_{j,j}}$ ▷ Quantization error
    $W_{:,j:(i+B)} \leftarrow W_{:,j:(i+B)} - E_{:,j-i} \cdot H^{-1}_{j,j:(i+B)}$ ▷ Update weights in block
  end for
  $W_{:,(i+B):} \leftarrow W_{:,(i+B):} - E \cdot H^{-1}_{i:(i+B),(i+B):}$ ▷ Update all remaining weights
end for

To enable GPTQ to be applicable to LLMs with billions of parameters, the authors have introduced three key optimizations:

1. *Arbitrary Order:* In the case of large models, the order in which weights are quantized becomes irrelevant. Therefore, GPTQ updates all weights in the same order for all rows. This means that the set of unquantized weights, denoted as F, and H_F^{-1}, the Cholesky Form - Inverse Layer Hessian, remain constant across all rows. This is because H_F depends solely on X_F and is independent of the weights. This reduction in the number of times H needs to be updated simplifies the process from $d_{col} \times d_{row}$ updates to just d_{col} updates.

2. *Lazy Batch-Updates:* Quantization of a column depends solely on updates to that particular column. Therefore, GPTQ employs batches of columns (with a batch size of $B = 128$). Equations 2.18 and 2.19 can be executed after the computation of a full batch B. The set of indices Q corresponds to the indices of quantized weights in the batch.

3. *Cholesky Reformulation:* To address numerical errors that arise from repeated application of equation 2.19, a Cholesky reformulation is applied to calculate all the necessary information about H^{-1} in advance. As the complete Cholesky decomposition cannot be applied, a mild damping factor is applied to the diagonal.

Additionally, an accessible quantization package called **AutoGPTQ** has been developed, which implements the GPTQ algorithm in PyTorch [William, 2023]. This package has been adopted by Hugging Face and is currently the only ready-to-use quantization technique available in the Transformers library [noa, 2023b].

2.4. Related Work

2.4.1. Question Answering based on PDFs

PDF Question Answering is the task of providing answers to questions related to the content of one or multiple documents [Mathew et al., 2021]. The field of research which actively explores this the closest is Visual Document Question Answering. It works on the development of an IR-QA system that operates on images of documents. An exemplary architecture and a general pipeline for transforming PDFs into an IR-QA system is presented by McDonald et al. [McDonald et al., 2022]. They developed their zero-shot framework around the QASPER dataset but used the original PDFs instead of extracted text via LaTeX. Moreover, readily available open-source tools like V-Doc [Ding et al., 2022] simplify the deployment and testing of datasets, models, and IR-QA systems of the Visual Document Question Answering domain.

More recently, the open-source framework *Langchain* has gained tremendous attention[4]. Langchain focuses on harnessing LLMs using chains, which are essentially prompts for an LLM that can be chained together [Langchain, 2023]. They also provide documentation on building a QA system based on PDFs [Langchain, 2023]. Similarly, *OpenAI* offers a Retrieval Plugin for *ChatGPT* [OpenAI, 2023], also an open-source repository. These QA systems adhere to the paradigms established in previous works such as [Karpukhin et al., 2020, Ni et al., 2021, Neelakantan et al., 2022, Lewis et al., 2021]. Specifically, this entails:

- Given a text corpus, documents can be retrieved by extracting relevant passages. Data cleaning of the corpus is optional but not necessary. Therefore, these systems employ a *direct extraction* approach, especially when dealing with PDFs.
- Utilizing large-scale, diversely trained encoders. Representation-based Retrievers, when equipped with sufficient trainable parameters and diverse training datasets, often yield comparable results to fine-tuned, more complex retrieval models [Ni et al., 2021, Neelakantan et al., 2022].

[4] As of September 24, 2023, Langchain has received 63k stars on GitHub

- Using the LLM as a generative reader for QA, as demonstrated in the work of Izacard et al. [Izacard and Grave, 2021].

Non-LLM research for QA based on PDFs is notably scarce. In the field of ODQA, discussions regarding applicable frameworks that encompass the entire pipeline from PDFs to QA are infrequent. Instead, the focus often revolves around constructing QA systems using predefined and well-supervised datasets. However, there is some research that explores the feasibility of deploying high-performing QA systems in out-of-domain scenarios, bypassing the initial stage of data preprocessing (from PDFs to passages). This research strives to outline possibilities for using a QA system in real-world passage collections.

Applying Dense Retrievers Out-of-Domain: As emphasized by Thakur et al. in their "Heterogeneous Benchmark for Zero-shot Evaluation of Information Retrieval Models" (BEIR) [Thakur et al., 2021], dense retrievers exhibit weak out-of-domain performance. Lyu et al. [Farea et al., 2022] also demonstrate the limited generality of dense retrievers when trained in one subdomain and subsequently applied in a different one. This underscores the conclusion that there are two approaches to employing retrievers in out-of-domain scenarios: (1) fine-tuning or (2) zero-shot, but with large encoders that have been trained on diverse datasets [Ni et al., 2021].

The challenge with fine-tuning lies in the unavailability of labeled data, which is typically required for supervised models in the form of tuples such as $(question, answer, context)$. Several diverse approaches have been developed to address this issue. One approach employs QG techniques, as exemplified by Promptagator [Dai et al., 2022b], which utilizes LLMs. Another strategy involves the use of Mixture-of-Experts and meta-learning algorithms [Chen, 2021]. Some researchers have explored semi-supervised training datasets, as demonstrated by Sachan et al. [Sachan et al., 2023], who developed ART, a training framework for dense retrievers that only requires questions and surpasses the standard DPR training implementation. At the current point in time there is no state-of-the-art appraoch to fine-tune a dense retriever on a small subdomain dataset.

In their study, Reddy et al. [Reddy et al., 2022] addressed the challenge of creating a QA-System for Covid-19-related documents, where no supervised QA dataset was available. Consequently, they conducted a comparison between the performance of zero-shot BM25 and DPR. Their findings revealed that BM25 outperformed DPR on the BiosQA QA dataset, closely related to the Covid-19 domain. Throughout their experiments, they evaluated various approaches, including simple zero-shot techniques, fine-tuning of DPR using QG via BART, which yielded superior results. Notably, the most effective retriever for unsupervised domain adaptation was a combination of BM25 and unsupervised fine-tuned DPR.

Furthermore, Gururangan et al. [Gururangan et al., 2020] demonstrated in their experiments that fine-tuning PrLMs on domain-specific language or, even better, task-specific data led to a significant performance boost.

Gholami et al. [Gholami and Noori, 2021] experimented with non-fine-tuned dense retrievers on a non-QA dataset, specifically a collection of AWS documentations. Their results, particularly for the retrieval component, were sobering, aligning with the findings of benchmark studies by Thakur et al. [Thakur et al., 2021] and Lyu et al. [Farea et al., 2022].

On the other hand, there exist reader components with a high degree of generalizability, as demonstrated by UnifiedQA-v2 [Khashabi et al., 2022], an extractive reader, and T5 [Raffel et al., 2023], a generative reader. So the main challenge, when building a IR-QA-System, lays within the implementation and adaptation of the retriever component.

2.4.2. Open-domain Conversational Question Answering ???

From single-turn Question Answering to Conversation

Synthetic Datageneration

3. Open-domain QA Chatbot

This chapter outlines the methods and techniques employed in the development of a conversational question-answering system designed for the use with a collection of documents as knowledge source. The chapter is structured as follows: Section 3.1 provides an overview of the desired use case, its objectives, and constraints concerning a Conversational Question Answering System. Section 3.3 presents a general framework that can be utilized as a decision tree for the practical implementation of a Conversational Question Answering System. Its subsections will highlight and discuss the components introduced within the framework.

3.1. Overview and Objective

The primary use case addressed in this thesis can be summarized as follows: Imagine having a collection of PDF files, and our goal is to create a chatbot capable of engaging in conversations about the knowledge within these PDFs. This chatbot should provide accurate answers to questions based on the content of the PDFs and furnish supporting evidence from these documents. Furthermore, it should enable users to have a conversational query experience, allowing them to ask follow-up questions and engage in dialogue with the chatbot based on its previous responses. Figure 3.1 illustrates an example of this use case.

Currently, to the best of my knowledge, there is no scientific paper or similar resource offering a comprehensive framework or pipeline to address this use case. This thesis aims to bridge this gap by presenting a framework and pipeline designed to tackle this specific scenario. Figure 3.2 provides an overview of the system architecture. The system follows the RAG architecture, as detailed in Section 2.1.4, which extends the classical Retriever-Reader with a LLM as a Reader, capable of incorporating parametric knowledge. To extend RAG to a Conv QA, a Contextual Query Understanding (CQU) unit, as introduced in Section 2.2.2, is essential. This novel architecture will be termed **Conversational Retrieval-Augmented Generation (ConRAG)**. The extraction pipeline will be discussed in Section 3.3.1, with its primary tasks being the extraction of passages from the provided set of Documents, the creation of an index, and the

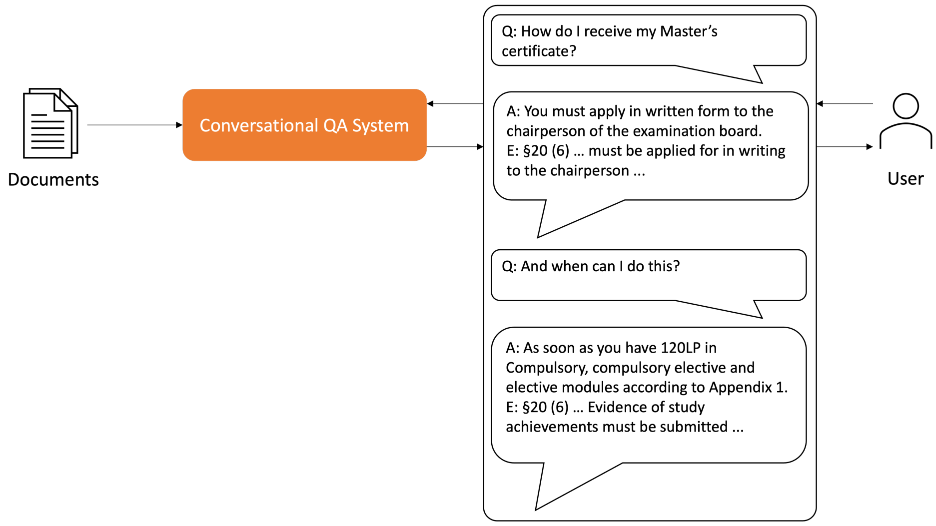

Figure 3.1.: Overview of the Example Use-Case

optional generation of synthetic training data. The three major modules comprising the architecture, namely the *Retriever*, *Reader*, and *CQU*, will be elaborated in their respective sections: 3.3.3, 3.3.4, and 3.3.2.

To summarize, the objectives of the QA capabilities of the system are as follows:

1. Utilize **Documents** as the primary **knowledge source**.
2. Enable the QA-System to handle a **variety of answer types**, including: **extractive**, **abstractive**, and **boolean**.
3. **Provide references** to document snippets **as evidence to answers**.
4. Ensure the pipeline's generalizability by using **open-domain methods**, allowing it to adapt to new domains or knowledge sources with **minimal or no supervision** and **small datasets**.
5. Design the pipeline to be **feasible without the need for datacenter-grade hardware resources**, making it accessible for development on standard research hardware.
6. **Prioritize accuracy as the primary objective**, as constraining memory consumption is indirectly covered in point (5). **Latency is not a primary concern**, as the system is not intended for real-time use and will not be optimized for that.

Regarding the ConvQA-System, the objectives are as follows:

1. Enable the ConvQA-System to **handle** the following follow-up **question types: drilling-down, clarification, topic shift** and **comparison**.

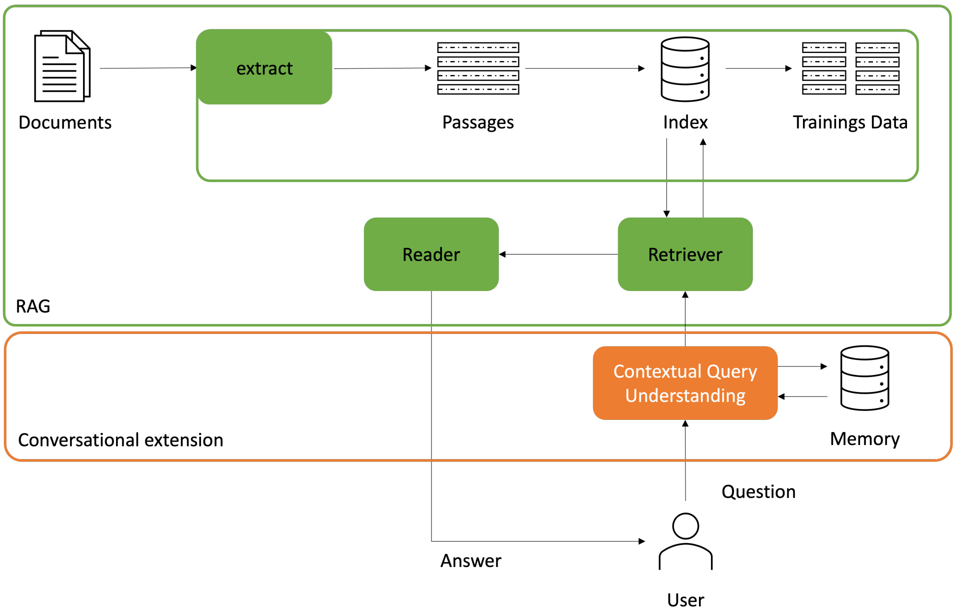

Figure 3.2.: Overview of the System Architecture

2. Be able to take Initiative in the form of **clarifying questions**.

3. The **memory** will be **limited to a session**.

3.2. Document Enquiry Model

The following Section will layout the problem of document-based Conv QA.

Substantially the fundamental source of knowledge are *Documents*. A *Document* can be any type of structured or unstructured file, which is being used for storing and displaying information. Examples could be HTML (structured) or PDF (unstructured) files. A *Document* consits of content C_d, a collection of information units c_d, whereas the content C_d has to have at least one c_d, but can contain also multiple. On top a *Document* contains metadata M_d, which are a collection of key-value-pairs, an example could be $M_d = \{(title, \text{Examination Regulation Master Data and Computer Science}),$ $(publish\_date, 14.06.2022)\}$. Next to the mentioned components of a *Dcoument* it has also a unique identifier UID_d, therefore a *Document* $d = (C_d, M_d, UID_d)$. For this thesis we will only cosider textual information units in C_d and not figures or images. Out of the necessity, for knowledge granularity and precise information context, we will define *Passages* next. A *Passage* p is a subset of a string of textual content $c_d \in C_d$. The granularity of p can be defined use-case specific. If p is a sentence, 100 tokens or the full textual content c_d. Nevertheless, every p contains a reference to the orgianl document d

it was taken from and has it's own unique identifier UID_p. This leads to the following *Passage Model*:

Definition 1 ***(Passage Model)*** *A passage p is a subset of a textual content string $c_d \in C_d$ of a document $d = (C_d, M_d, UID_d)$, whereas $p = (content, UID_p, UID_d)$.*

For the ease of notation, we will refer to the content of a passage p as p itself in the following. The collection of all *Passages P* will be refered to as the *Knowledge Source*.

For the following definitions it's important to clarify the concept of *Intent* first. A simple example to illustrate intent could be the following:

Question: When was Barack Obama born?

Answer 1: 4. August 1961
Answer 2: Barack Obama was the 44th president of the USA.
Answer 3: Either 04.08.1961 or 05.09.1962 I'm not sure.

The intent of the question is fullfilled given answer 1, so the answer contains the information the user was looking for, when starting the search, but answer 2 misses the search intent. Answer 3 is somewhere inbetween, as it understands the search intent, but doesn't fullfill it correctly. Therefore intent can be summarized as the users true information need he had when generating a search query. Mathematically we define intent the following way:

Definition 2 ***(Intent)*** *Intent is an operation $\mathcal{I}(x, y)$ between two elements, which can either be two questions or a question and an answer. This operation returns a value between 0 and 1, which indicates the overlap of the two intents of x and y, $\mathcal{I}(x, y) = [0, 1]$*

Next we need to define what a *Question* is. For this problem, a question consits of a string. Generaly a question also has an intent, which can be for example measured against the golden answer (the correct answer to a question). The *Question Model* can therefore be defined as follows:

Definition 3 ***(Question Model)*** *A question q contains of a string. Given the golden answer a_q, $\mathcal{I}(q, a_q) = 1$.*

Naturally where there is a *Question*, there has to be an *Answer*. An *Answer a* is a string, which is the answer to a *Question q*. It can be considered a gold answer, when $\mathcal{I}(q, a) = 1$. Formally we define an *Answer* as:

Definition 4 ***(Answer Model)*** *An answer a is a string, which answers a given question q. If $\mathcal{I}(q, a) = 1$, a is considered a gold answer to q.*

In terms of conversations, we split an exchange between two agents into *Turns* as described in Section 2.2.1. Generally speaking, a *Turn* h consists of a tuple $\langle q, a \rangle$, whereas the a is the response to q. *Turns* happen in order and therefore have a logical relation. We refer to the sequence of multiple *Turns* within one conversation as *History* H.

Definition 5 ***(History Model)*** *A history H is a seuqnce of turns h, whereas $h = \langle q, a \rangle$, $H = \langle h_1, h_2, \ldots h_i \rangle$.*

As we now have elaborated, what *Questions*, *Knowledge Source*, *Answers* and *History* are, we're ready to define the problem of Conv QA:

Definition 6 ***(Conversational Question Answering Task)*** *Given a new question q_{i+1} and a history H with i-many turns h, a model ($\mathbf{M}$) should generate an answer a_{i+1}, based on the provided knowledge in the knowledge source P, which satisfies the search intent i of q_{i+1}. Next to the answer a_{i+1}, $\mathbf{M}$ should return p as evidence from P. Formally:*

$$\mathbf{M} : (q_{i+1}, H, P) \rightarrow (a_{i+1}, p)$$

3.3. Conversational Retrieval-Augmented Generation

In order to provide a solution to the Task of Conv QA as defined in Definition 6, the system must be able to perform evidence selection based on a *Knowledge Source* which is an important criterion also layed out in Section 3.1.

In order to now create a system architecture, which fullfills the task of model $\mathbf{M}$ (see Definition 6), we will split the main task of Conv QA into multiple subtasks:

1. **Information Extraction:** Given a set of documents D, extract the textual content C_d of each document $d \in D$ and create a knowledge source P based on C_d of every document $d \in D$.

2. **Contextual Query Understanding:** Given a history H and a new question q_{i+1}, generate a contextualized question q_c based on H, sothat $\mathcal{I}(q_c, q_{i+1}) = 1$.

3. **Passage Retrieval:** Given a contextualized question q_c and a knowledge source P, retrieve the k-most relevant passages p from P and combine them in an evidence set E.

4. **Response Generation:** Given a contextualized question q_c and a set of passages E, generate an answer a to q_c based on E, so that $\mathcal{I}(q_{i+1}, a) = 1$.

There may exist other approaches to break down the task of $\mathbf{M}$ into sub-tasks, but for this thesis, we will focus on a solution based on the four sub-tasks outlined above. The reason is simply the existing state-of-the-art research in every field of this sub-tasks, especially when it comes to the adoption of new domains. It is to be highlighed, that breaking the task of $\mathbf{M}$ implies also an order in which the sub-tasks have to be performed. Sub-task (1) will be performed once, while (2-4) will be repeated on every new question q_{i+1}.

In order to develop a system which can be applied to this abstract task, we match every task to a component. The *Information Extraction* sub-task will be solved by the *extract* component, further detailed in Section 3.3.1. *Passage Retrieval* will be covered by the *Retriever* component, further discribed in Section 3.3.3. The *Response Generation* will be handled by the *Reader* component, more precisely in this thesis we will focus on LLMs with intrinsic parametric knowledge as *Reader*. This will lead to a RAG system consisting of the *Retriever* and *Reader*. This choice has been made due to the fact, that the latest reasearch breakthroughs sparked the interest in RAG systems in compraison towards classical Retriever-Reader systems (check herefore the related work Section 2.4). Details on the *Reader* component will be layed out in Section 3.3.4. In order to now handle conversations, a CQU unit as described in Section 2.2.2 is necessary to handle the sub-task of *Contextual Query Understanding*. Section 3.3.2 will dive into the details.

Figure 3.3 illustrates the combination of the four components, which make up the model $\mathbf{M}$ in their corresponding sub-tasks. This is the abstract Model $\mathbf{M}$ which leads, when applied to a real use-case, to the ConRAG system architecture.

3.3.1. extract

The sub-task of *Inforamtion Extraction* was defined in the previous Section as sub-task (1) of the model $\mathbf{M}$. Given a set of documents D, the knowledge source P needs to be extracted using *Information Extraction* techniques in combination with *Passage Extraction* operations. As synthetic data generation is also part of the extraction component according to Section 3.1, it will also be discussed in this section. This is originally not

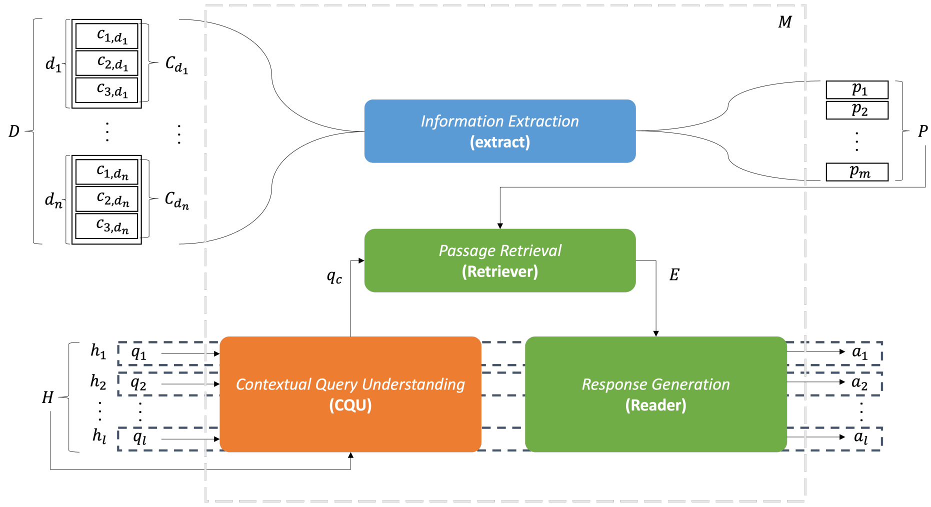

Figure 3.3.: Overview of the System Architecture in context of the sub-tasks of **M**

part of the model **M**, but makes from a system architecture sense, to place these operations in this system component. Synthetic Data can be seen as an seperate task, which has nothing to do with the original task of Conv QA, but is a necessary step in order to train components of the model **M** or evaluate those.

Information Extraction: When it comes to extracting text from any document d, there are many approaches to choose from. Some extract structures, metadata, or similar, which can be further utilized, while others extract unstructured text only. In any case, this extraction process highly depends on the source document type. An HTML website requires different approaches and tools compared to a PDF, for instance. An example tool for direct extraction of PDFs is Py2PDF [PyPDF2,]. Regardless of the source document and tool used to extract textual information C_d, there are two major possible outcomes given a set of documents D:

1. *Structured Extraction:* Denoted as $f_{StrucExt}(\cdot)$, in this extraction operation, C_d can be extracted into logical segments directly: $f_{StrucExt}(d) = \{c_{d_i} \subseteq C_d : i \in \{1, 2, \ldots, n\}, C_d = \bigcup_{i=1}^{n} c_{d_i}\}$. If $f_{StrucExt}$ is applied to all documents d in D, the resulting set C contains all the outputs of $f_{StrucExt}$ for each document in D. Formally, $f_{StrucExt}(D) := C = \{C_{d_1}, C_{d_2}, \ldots, C_{d_m}\}$, where each document d is transformed into a set of logical segments C_{d_i} representing its content.

2. *Unstructured Extraction:* Denoted as $f_{UnstExt}(\cdot)$, when applied to D, it results in a concatenated text corpus $C_d = \{c_d\}$ containing the textual content of a document d: $f_{UnstExt}(D) := C = \{C_{d_1}, C_{d_2}, \ldots, C_{d_n}\}$, where each document d is transformed into a single text snippet c_d representing its content.

In order to generate now snippets p from C, *Passage Extraction* has to be applied on C.

Passage Extraction: The implementation of passage splitting depends on the nature of C and the desired granularity of the output. In general, there are three operations for constructing passages p based on a text snippet c_d:

1. *Paragraphs:* $f_p(\cdot)$ is an operation that transforms a collection C_d of texts c_d into a set of passages $P = \{p_1, p_2, \ldots, p_n\}$. It operates similarly to $f_{StrucExt}(\cdot)$, but instead of D, it operates on C_d. The output of $f_p(\cdot)$ consists of passages p that represent logical segments of the text corpus c_d. Usually, it operates in a *rule-based* manner, meaning that a paragraph is defined by a token indicating a paragraph (e.g., <p/> in HTML). The length l of each p_i is variable, and the number of paragraphs $|P|$ can also vary.

2. *Snippets:* $f_s(\cdot)$, when you have a fixed passage length l, divides the concatenated text c_d into $|c_d| \bmod l + 1$ passages p per c_p. Alternative approaches may involve specifying minimum and maximum lengths, denoted as $l_{\min}$ and $l_{\max}$. The exact point of division depends on whether a sentence ends within the specified window or not. If a sentence ending is found within the window, the snippet concludes at that point. Otherwise, it concludes at the end of the window. The individual length of the extracted passages $P = \{p_1, p_2, \ldots, p_n\}$ is not fixed in the case of syntactic snippets, as well as the number of paragraphs $|P|$.

3. *Sliding Windows:* $f_w(\cdot)$ utilizes a window size l, a concatenated text c_d, and a step size s. The window slides over the text c_d, and the text within the window is used as a passage p. This results in $\frac{|c_d|-l}{s}$ passages, denoted as $P = \{p_1, p_2, \ldots, p_n\}$.

These operations can be combined in a pipeline fashion. For example, first $f_p(C_d) := P_{paragraphs} = \{p_{paragraphs,1}, p_{paragraphs,2}, \ldots, p_{paragraphs,n}\}$ is constructed, and afterwards, on the logical paragraphs, $f_w(f_p(C_d)) := f_w(P_{paragraph}) = P_{window} = \{p_{window,1}, p_{window,2}, \ldots, p_{window,i}\}$ is applied. The way these operations are combined is highly use-case specific and needs to be evaluated for each use-case individually. Another important factor influencing the decision regarding the parameters l, in general, the operation used, is the desired model for the *Retriever* and *Reader* as they may be trained on a specific token length as input or have a maximum length of tokens they accept as input or require a certain clarity of data points.

Synthetic Training Data Generation: This can be interpreted as a knowledge distillation task. The main idea is to use a model $ED(\cdot)$ to generate a synthetic dataset for the sub-task and problem field of *Passage Retrieval* and *Response Generation*:

$$\mathbf{ED}(\mathbf{P}) := (\mathbf{P}, \mathbf{Q}, \mathbf{I}) \tag{3.1}$$

Where $P = \{p_1, p_2, \ldots, p_n\}$ corresponds to a corpus of passages to retrieve from, Q is a set of questions, and I is the underlying search intent for a question $q \in Q$. If $\mathcal{I}(q, p) = 1$, this means that the passage reflects the question's search intent. Given this task, there are several synthetic dataset types:

1. *Questions given Context* $ED_{qp}(p, I) := q_s$: Given a passage p, generate a synthetic question q_s that satisfies the desired search intent I. Applying $ED_{qp}(P)$ to a set of passages will generate a set of questions $Q_s = \{q_1, q_2, \ldots, q_n\}$, with one question for every passage, i.e., $|Q_s| = |P|$. ED_{gq} can also be applied multiple times with different $i \in I$ to generate multiple different questions $q_{j,s,1}, q_{j,s,2}, \ldots, q_{j,s,i}$ for a single passage p_j.

2. *Question-Context-Answer Triples* $ED_{qpa}(p, I) := (q_s, p, a_s, I)$: Given a passage p, generate a synthetic question q_s that satisfies the search intent I and provides an answer a_s, where $\mathcal{I}(q_s, a_s, p) = 1$. The result of applying $ED_{qpa}(P)$ to a set of passages is a set of question-passage-answer triples $QPA_s = \{(q_{1,s}, p_1, a_{1,s}, I), (q_{2,s}, p_2, a_{2,s}, I), \ldots, (q_{n,s}, p_n, a_{n,s}, I)\}$, where $|QPA_s| = |P|$. ED_{qpa} can also be applied multiple times with different $i \in I$ to generate multiple different questions $q_{j,s,1}, q_{j,s,2}, \ldots, q_{j,s,i}$ and corresponding answers $a_{j,s,1}, a_{j,s,2}, \ldots, a_{j,s,i}$ for a single passage p_j.

While the previous two approaches focus on single tuples of either (q, p) or triples of (q, p, a), there is also a problem field of generating conversations based on passages $P = \{p_1, p_2, \ldots, p_n\}$ from the same underlying document d. Therefore, the task given in Equation 3.1 is changed to:

$$\mathbf{ED}(\mathbf{P}) := (\mathbf{H}, \mathbf{P}, \mathbf{I}) \tag{3.2}$$

Where H corresponds to a set of conversation histories h containing multiple turns. The first turn is always a question q_1, followed by an answer a_1 based on a passage $p \in P$, also given a search intent I over the whole history h. Synthetic datasets for this task can be generated in the following ways:

3. *Conversational Question-Context-Answer Histories* $ED_{cqpa}(E, I) := H_s$: Given a subset of passages $E \subset P$, the task of the model ED_{cqpa} is it to construct a realistic conversation with an intent I over the corpus E. The result of applying $ED_{cqpa}(E, I)$ on a set of passages is a set of conversation histories $H_s = \{h_1, h_2, \dots, h_n\}$. ED_{cqpa} can also be applied multiple times with different $i \in I$ in order to generate multiple different conversations $h_{j,s,1}, h_{j,s,2} \dots, h_{j,s,i}$ for a single subset of passages $E \subset P$.

There are several ways to implemnt the models ED_{qp}, ED_{qpa} or ED_{cqpa}. Implementations of those appraoches will be discussed in Chapter 4.

3.3.2. Contextual Query Understanding

Section 3.3 introduced the model $\mathbf{M}$ for Conv QA. The CQU component is responsible for the *Contextual Query Understanding* task. Essentially, the goal of this task is to identify the necessary information in the history H of a conversation to adapt the expression of the question q_{i+1} into a contextualized question q_c, such that $\mathcal{I}(q_c, q_{i+1}) = 1$. This adaptation is crucial, as the *Passage Retrieval* task is executed using a single question q rather than the entire history H. Therefore, natural language challenges may arise between turns, such as pronoun resolution (e.g., it, he, she) or turn references.

$$CQU(h_{i-k:i}, q_{i+1}) := q_c \mid \mathcal{I}(q_c, q_{i+1}) = 1 \tag{3.3}$$

The depth of the history $h_{i-k:i}$ that CQU operates on is a hyperparameter to be chosen. Generally, there are three levels of depth:

1. **K-many:** Consideration of the last k turns.
2. **All:** Utilization of the entire history, $k = i - 1$.
3. **Memory:** Inclusion of not only the current history but also previous chat histories by this user. The concept of memory will not be further discussed in this thesis, as it opens a whole new field of research and was excluded in the problem statement in Section 3.1.

The decision on depth already implies limitations, as the *K-many* approach may lead to reference errors in a conversation, while too large k increases computational complexity. This leads to $h_{i-k:i} \subset H$, which is used by the CQU.

The CQU unit itself can be either an implicit or explicit method. Implicit methods use a one-step approach and are based on the *Transformer* architecture, while explicit methods apply a two-step approach and can vary in implementation:

$$CQU_\theta(h_{i-k:i}, q_{i+1}) := q_c \mid \mathcal{I}(q_c, q_{i+1}) = 1 \tag{3.4}$$

Equation 3.4 shows the transformer-based CQU unit with trainable parameters. During training of the parameters, it is necessary to have an annotated dataset of turns and contextualized questions $q_{c,i}$ for every turns question q_i. The loss of the CQU will be defined between the human-annotated contextualized question q_c^* and the generated contextualized question q_c.

Explicit approaches consist of two operations:

$$rel_{CQU}(x, q_{i+1}) := \begin{cases} x \in C, & \text{if } x \text{ is relevant for } q_{i+1} \\ x \notin C, & \text{if } x \text{ is not relevant for } q_{i+1} \end{cases} \tag{3.5}$$

$$rep_{CQU}(C, q_{i+1}) := q_c \tag{3.6}$$

Where rel_{CQU} is an operation to determine, for a string $x \subset h_{i-k,i}$ (which can be a token, word, or snippet of a turn $h_{i-k,i}$), if it is relevant context to enrich the question q_{i+1}. If so, it is added to the context set C. The operation rep_{CQU} works over C and q_{i+1} to rephrase and generate a contextualized question q_c.

The relevance operation can be implemented in multiple ways. Some include seq-2-seq models or entity matching. The rephrasing operation is a typical MRC summary task. It's inputs are the set of context relevant information C and the question q_{i+1}.

3.3.3. Retriever

In Section 3.3, we have divided the main task of a model $\mathbf{M}$ for Conv QA into multiple sub-tasks. The Retriever component handles the *Passage Retrieval* task. The goal of this component is to identify relevant passages p from the knowledge source P given a contextualized question q_c. The contextualized question q_c is generated by the CQU unit (see Section 3.3.2). Approaches using instead of q_c the whole history H for retrieval will not be discussed in this thesis. The identification of passages includes a scoring of relevance for each passage p given a question q_c. The scoring function $p_{\text{Ret}}(p|q_c)$ can be defined as follows:

$$p_{\text{Ret}}(p|q_c) = \text{Score}(q_c, p) \tag{3.7}$$

Here, $\text{Score}(q_c, p)$ is a value determining the relevance of a passage p, enabling the ranking of passages p given a question q_c. Based on the score, the passages will be ordered in descending order, and the top-k passages will be combined into an evidence set E_{q_c}.

Within the evidence set E_{q_c}, a passage p will be represented as a tuple $(p, \text{Score}(q_c, p))$. Concerning the evidence set E_{q_c} in relation to the question q and the underlying search intent i, the following cases exist:

$$\forall q \in Q, \exists! E_{q_c} \subset P := \begin{cases} 1.\ \mathcal{I}(q, E_{q_c} = \emptyset) = 1 \\ 2.\ \mathcal{I}(q, E_{q_c} = \{p\}) = 1 \\ 3.\ \mathcal{I}(q, E_{q_c} = \{p_1, p_2, \ldots, p_n\}) = 1 \end{cases} \tag{3.8}$$

For every question, there exists an evidence set. In order for intent $\mathcal{I}(q, E_{q_c})$ to be 1, there are three cases: either the evidence set E_{q_c} has to be empty, E_{q_c} has to have exactly one element p, or E_{q_c} has to have multiple passages p. To put it more simply, for a given question, the answer could be determined either by the evidence that there is no supporting passage p, or that there is exactly one passage p containing the requested information, or that multiple passages p contain the necessary information to answer the question q.

Retriever: Depending on the chosen method for a Retriever, the Retriever component can either be trainable or have adjustable parameters:

$$r_{\text{Ret},\theta} = \text{Retriever}_\theta(q_c, P) \tag{3.9}$$

$$r_{\text{Ret},\Theta} = \text{Retriever}_\Theta(q_c, P) \tag{3.10}$$

Here, θ is the set of trainable parameters, and Θ is a set of adjustable parameters. Examples of the first type of retriever include dense retrievers like *DPR*, as introduced in Section 2.1.4. Examples of the second type of retriever include sparse retrievers like *BM25*, also introduced in the same section. Regardless of the type of retriever, the operation from Equation 3.7 is applied to each passage $p \in P$, and the top-k passages are selected to form the evidence set E_{q_c}. As outlined in Equation 3.8, some questions may require no evidence to fulfill the search intent. However, with the proposed Retrievers, this is not possible. A retriever r_{Ret} will always assign a $Score(q, p)$ to every passage given a question and its search intent. Filters could be applied to filter out passages with a low score, but this is not part of the retriever itself. The evaluation of the evidence in respect to the search intent and correctly identifying the case at hand will be handled by the Reader component (see Section 3.3.4).

Mixture-of-Experts: To enhance the quality of the evidence set E_{q_c}, state-of-the-art research and related work have emphasized the effectiveness of combining multiple retrievers (see Section 2.4). This combination can be achieved in various ways, which can also be combined further:

- **Re-Ranking**: This involves the combination of multiple retrievers in a pipeline fashion. In most cases, a fast but imprecise Retriever is used to retrieve a large evidence set, e.g., $r_{BM25}(q_c, P) = E_{q_c,1}$, where $|E_{q_c,1}| = 100$. Subsequently, a second, more accurate Retriever is employed to re-rank the evidence set E_{q_c}, e.g., $r_{CE}(q_c, E_{q_c,1}) = E_{q_c,2}$, where $|E_{q_c,2}| = 25$.

- **Ensemble**: This idea involves combining multiple retrievers into a single retriever. This can be accomplished by either concatenating the evidence sets $E_{q_c,j}$ or by aggregating the scores of the individual retrievers r_j into a single score. Formally:

$$\forall r \in R : E_{q_c} = \bigcup_{j=1}^{|R|} r_j(q_c, P) \tag{3.11}$$

$$\forall p \in P : \text{Score}(q_c, p) = f(r_1(q_c, p), r_2(q_c, p), \ldots, r_{|R|}(q_c, p)) \tag{3.12}$$

  Here, $f(\cdot)$ represents a function that combines the scores of the individual retrievers r_j, for example, using $max(\cdot)$, and R is a set of retrievers r.

- **Weighting**: This approach involves running two retrievers in parallel and multiplying their scores for each passage p. It can be seen as a sub-case of Ensemble. Formally:

$$\forall p \in P : \text{Score}(q_c, p) = \prod_{j=1}^{|R|} \alpha_j \cdot r_j(q_c, p), \sum_{j=1}^{|R|} \alpha_j = 1 \tag{3.13}$$

  Here, R represents a set of retrievers r_j, and α_j is the weight assigned to each retriever r_j. These weights α_j can be either fixed or trainable.

Fine-Tuning: Given a dataset of tuples (q, p), as described in Section 3.3.1, a retriever with trainable parameters $r_{\text{Ret},\theta}$ can undergo fine-tuning. The first step involves creating the training data $\mathcal{TD} = \{\langle q_i, p_i^+, p_{i,1}^-, \ldots, p_{i,n}^- \rangle\}_{i=1}^m$. In this dataset, we already have tuples that contain a question q_i and a positive passage p_i^+. The primary objective is to sample n negative passages $p_{i,j}^-$ for each positive passage p_i^+. Various methods can be employed [Karpukhin et al., 2020]:

1. Randomly sampling passages from the knowledge source P, where $p^-_{i,j} \in P$ and $p^-_{i,j} \neq p^+_i$.

2. Retrieving evidence passages using a high-performing Out-of-Domain (OOD) retriever $r_{\text{Ret,OOD}}$ and sampling from the evidence set $p^-_{i,j} \in E_{q_i}$, ensuring $p^-_{i,j} \neq p^+_i$.

3. Sampling passages from other tuples (q_k, p_k) in the dataset, where $p_k \neq p^+_i$.

Method (1) is the simplest but does not yield high-quality negative passages $p^-_{i,j}$. Method (2), while more complex, typically provides a higher-quality evidence set E_{q_i} than method (1). Method (3) can also be applied in-batch, known as *in-batch negatives*, during the training process. This means that given a batch B of $|B|$-many (q_i, p^+_i) tuples, the negative passages $p^-_{i,j}$ are derived from the positive passages of the other tuples in the same batch B. Consequently, this results in $|B| \times |B| - 1$ negative passages $p^-_{i,j}$ per batch B per question q_i.

After construction of the training dataset $\mathcal{TD}$, the retriever $r_{\text{Ret},\theta}$ can be trained straightforward. The forward pass is simply the score prediction of the retriever for every passage, positive and negatives, within a tuple of the $\mathcal{TD}$. Therefore the loss function can be defined as following:

$$\mathcal{L}(q_i, p^+_i, p^-_{i,1}, \ldots, p^-_{i,n}) = -\log \frac{e^{\text{Score}(q_i, p^+_i)}}{e^{\text{Score}(q_i, p^+_i)} + \sum_{j=1}^{n} e^{\text{Score}(q_i, p^-_{i,j})}} \tag{3.14}$$

The calculated loss will be used in backpropagation to update θ.

The process of fine-tuning needs adaptation when using *synthetically generated data* due to the potential quality issues of the synthetic dataset. To address this, the dataset must be iteratively filtered during the training process. Given a synthetic dataset $\mathcal{TD}_\int = \{\langle q_i, p^+_i \rangle\}_{i=1}^{m}$, a retriever $r_{\text{Ret},\theta}$ with trainable parameters θ, and a high-performing out-of-domain (OOD) retriever $r_{\text{Ret,OOD}}$, the following procedure is based on the proposed approach of PROMPTAGATOR [Dai et al., 2022b]:

1. Extend $\mathcal{TD}_\int$ by adding n negative passages p^-_i for every tuple using Method (2) from above with $r_{\text{Ret,OOD}}$.

2. Train $r_{\text{Ret},\theta}$ for s iterations on $\mathcal{TD}_\int$ while applying in-batch negatives, as described in Method (3) above.

3. Filter $\mathcal{TD}_\int$ using $r_{\text{Ret},\theta}$ and $r_{\text{Ret,OOD}}$ by removing all tuples (q_i, p^+_i) where p^+_i is not in the Evidence set E_i, given a parameter k, for either $r_{\text{Ret},\theta}$ or $r_{\text{Ret,OOD}}$.

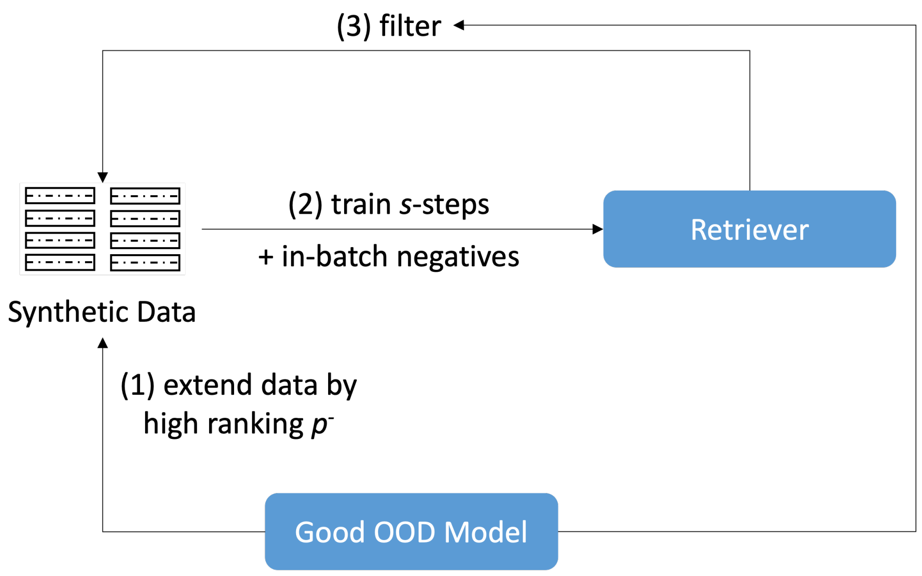

Figure 3.4.: Fine-Tuning Process for Retriever

Figure 3.4 illustrates the fine-tuning process with synthetic data. Steps (2) and (3) are repeated cyclically, with step (3) concluding each cycle. After s epochs, $\mathcal{TD}_f$ is filtered once, retrained for s epochs, refiltered, and finally trained again for s epochs. This process may appear counterintuitive, as the retriever is trained on the dataset it's supposed to filter. However, experiments from related work [Dai et al., 2022b] have demonstrated that this approach still yields favorable results.

Metadata-Filtering: In certain search scenarios, it may be necessary to filter the knowledge source P by metadata. State-of-the-art open-domain datasets and benchmarks usually don't develop solutions for this problem. However, in real-world scenarios, this is a common user desire.

A filtering request can be understood as the already established intent-fulfilling retrieval task performed over P, extended by a consideration of the metadata M_d of the documents $d \in D$ from which p originates. As established in Definition 1, $p = (content, UID_p, UID_d)$ and $d = (C_d, M_d, UID_d)$. Therefore, $\forall p \in P : \exists! M_d$ such that $M_p = M_d$. The task of filtering can therefore be understood as follows:

$$m_{\text{Ret}}(p|q_c) = \begin{cases} p \in E_m, & \bigcap_{i=p_1}^{E_{q_c}} M_i \subset M_p \\ p \notin E_m, & \bigcap_{i=p_1}^{E_{q_c}} M_i \not\subset M_p \end{cases} \tag{3.15}$$

Here, a passage p is only added to the evidence set E_m if the intersection of all metadata of the passages, in the perfect intention-fulfilling evidence set E_{q_c} of a question q_c, is a subset of the metadata M_p of the passage p.

In the retrieval process itself, the two tasks of metadata-filtering (m_{Ret}) and scoring (p_{Ret}) can be executed sequentially or combined. This depends on the implementation of the actual index. Some approaches towards metadata-filtering in Conv QA are the following:

1. **Metadata Passage Integration:** In this approach, the metadata M_p is integrated into the *content* of a passage p. This enables a one-step approach to retrieval. The retriever will perform scoring and metadata filtering simultaneously. However, in this approach, the original form of the metadata M_p, a collection of key-value pairs, will not be utilized. Instead, the metadata will be added implicitly (e.g., embedding addition) or explicitly (e.g., attaching keywords) to the *content* of p, leading to a loss in the functionality of the metadata, which could cause issues.

2. **Separate Metadata Index:** In this approach, two indices are created and utilized, one representing the passages' *content* and the other representing the passages' metadata. During retrieval, the retriever follows a two-step approach. Ideally, the metadata-filtering task is executed first, and then the scoring task is performed on E_m: $E_{q_c} = p_{\text{Ret}}(m_{\text{Ret}}(q_c, P))$.

3. **Hierarchical Index:** The hierarchical index employs a scoring function p_{Ret} as the metadata filter. In this approach, alongside the passage index, a secondary metadata index is utilized, typically representing the set of documents D. Instead of using M_d as key-value pairs, a document d is encoded in this index as *content*, resembling a passage p. This *content* may include a document summary, a concatenation of metadata keywords, or similar information. During retrieval, the retriever initially generates an evidence set E_m by executing $p_{\text{Ret}}(q_c, D)$, followed by the execution of $p_{\text{Ret}}(q_c, E_m)$ to form the ultimate evidence set E_{q_c}: $E_{q_c} = p_{\text{Ret}}(p_{\text{Ret}}(q_c, D))$.

3.3.4. Reader

4. Experimental Evaluation

The previous chapter, Chapter 3, layed out a holistic framework for Conv QA. This chapter aims to evaluate the applicability of the established system in a real-world scenario. Section 4.1 describes the available data for the real-world scenario and delves into applied data augmentation techniques. Section 4.2 introduces the metrics used to evaluate the performance of the individual system components, as well as the complete Conv QA system. These metrics are selected based on those used in related work. Section 4.3 details the experimental setup, implementation specifics, and provides an implementation framework for similar use cases. Finally, Section 4.4 presents both quantitative and qualitative results from the experiments.

4.1. Data

To evaluate the proposed system architecture and framework discussed in Chapter 3, we will implement and assess its performance using the PDFs of the University of Heidelberg's examination regulations (ER). The university provides these regulations on two separate websites: one in German[1] and the other in English[2]. It's important to note that only the German ER holds legal authority, and in cases of ambiguity between the English translation and the German original, the German ER version takes precedence.

Both websites offer structured access to the ER of nearly all faculties at the University of Heidelberg. However, it's worth mentioning that the German website is regularly updated, while the English version primarily contains outdated ER. Since there is no centralized source for accessing all English ER, the decission was made to utilize both the outdated English ER and the current German ER for this thesis. This decision aligns with the primary objective of this thesis, which is to demonstrate the proof-of-concept (PoC) of the introduced framework. Obtaining the latest English ER of the University of Heidelberg is beyond the scope of this thesis.

[1] https://www.uni-heidelberg.de/de/studium/studienorganisation/downloadcenter/studien-und-pruefungsordnungen

[2] https://www.uni-heidelberg.de/courses/download/examination_rules_regulations.html

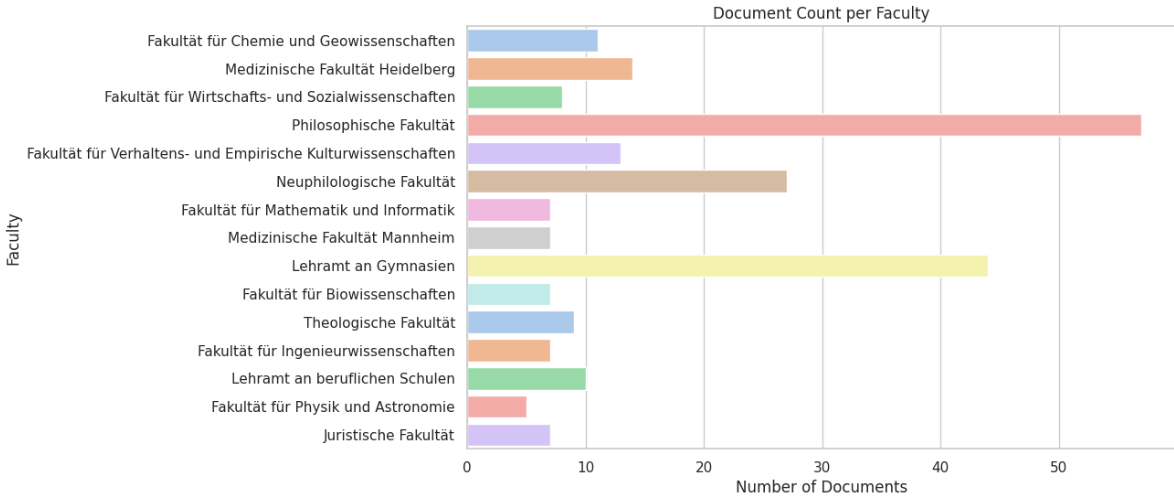

Figure 4.1.: Documents per Faculty of the German ER Dataset

As part of the experiments outlined in Section 4.3, the knowledge source of the German ER will be translated. Additional details on this process can be found in Section 4.3.1.

The two datasets (German/English) differ in their statistics. Therefore, Figure 4.1 displays the number of documents per faculty regarding the German ER. Figure 4.2 provides the same statistics for the English ER dataset. In general, the German dataset consists of 233 individual ER, while the English dataset contains only 151 individual ER. As observed, some faculties are overrepresented in both datasets, such as the philosophical faculty. The impact of the document distribution on the system will be discussed in Section 4.4.

While the previous statistics described the underlying documents of the PoC, what's even more interesting are the statistics related to the final knowledge source of passages that will be used throughout the PoC. Details on how these passages have been extracted from the PDFs will be provided in Section 4.3.1. In this section, we will focus on the statistical analysis.

Figure 4.3 illustrates the distribution of passages across documents in the German dataset, offering insights into the diverse influence that documents can have on the knowledge source in terms of the number of passages. These differences in passages are mainly due to variations in the length of the examination regulations (ER). The total number of passages in the German dataset is 39,039, resulting in an average of 167.55 passages per document. In comparison, the English dataset has an average of 212.94 passages per document. The statistics for the translated dataset are similar to those of the German dataset. An apparent difference between these two datasets is that the distribution of German passages per document exhibits a slight camel-like shape

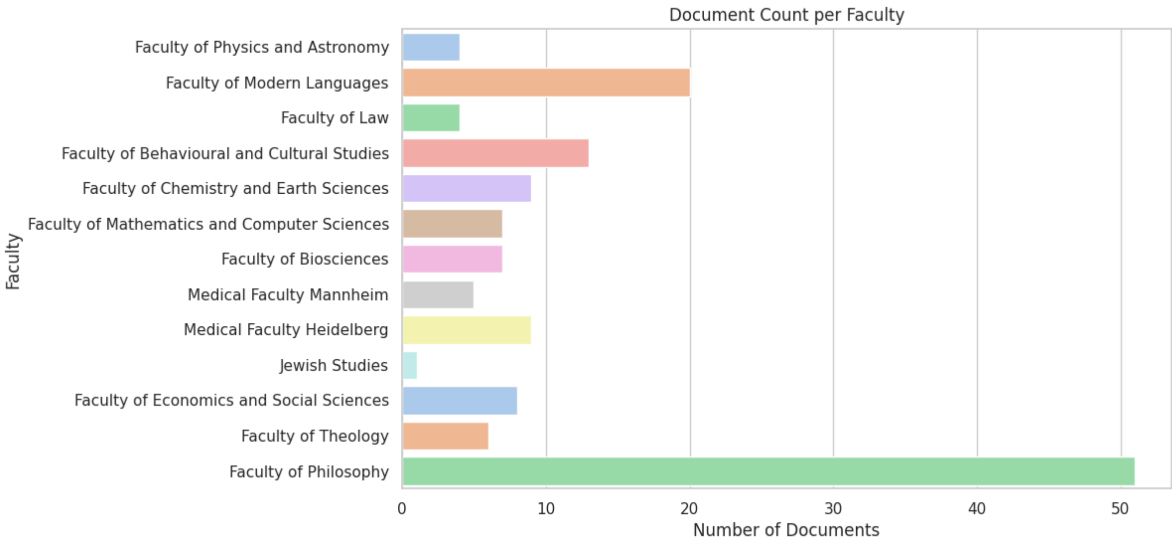

Figure 4.2.: Documents per Faculty of the English ER Dataset

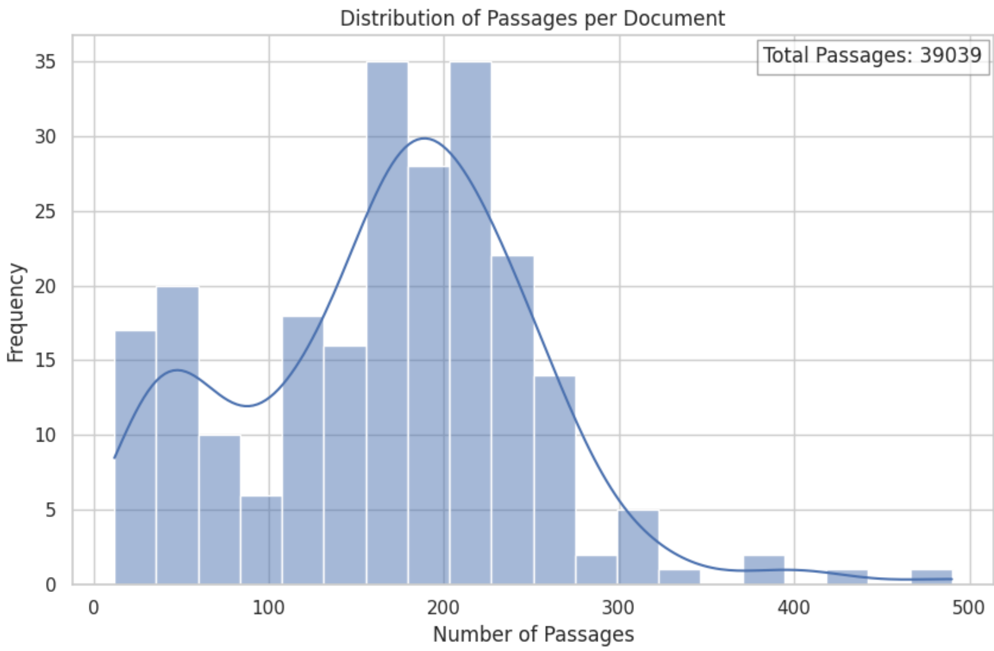

Figure 4.3.: Passages per Document of the German ER Dataset

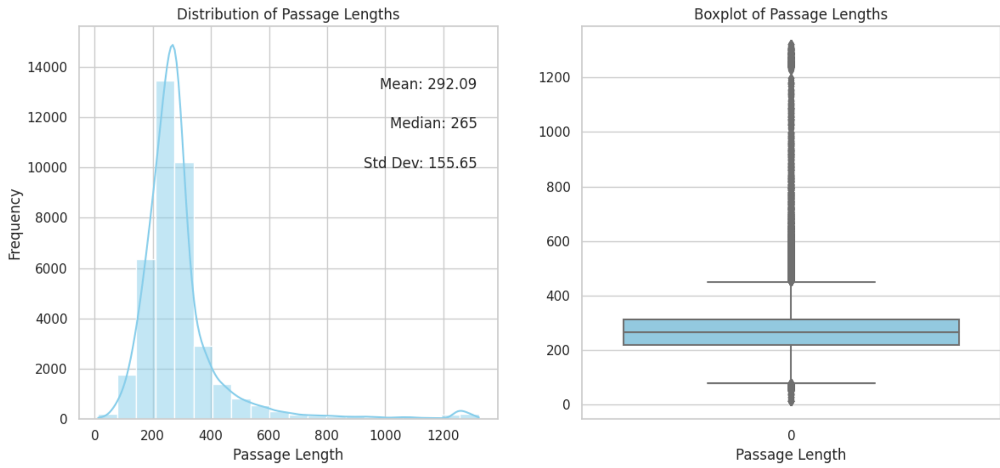

Figure 4.4.: Passage Length Distribution of the German ER Dataset

with a local maximum, while the English distribution follows a clear Gaussian pattern. However, these distribution differences are not further evaluated, as we assume that they do not significantly impact the system's quality.

Figure 4.5 presents the same statistics for the English dataset. As observed, the English knowledge source is smaller (31,659 vs. 38,642) than the German one, which is expected given that the English dataset contains only 151 ER compared to the 233 German ER. Still, the average number of passages per document is higher in the English dataset, as mentioned earlier. The total number of passages in the knowledge source indicates a highly closed-domain scenario. For comparison, MS MARCO [Bajaj et al., 2016], an often-used dataset for open-domain QA, has a knowledge source comprising 3.2 million passages, with an average length of 442 characters and passages ranging from 19 to 1,167 characters.

Figure 4.4 depicts the distribution of passage lengths in the German dataset. It's evident that the majority of passages fall within the range of 200 to 350 characters. The same statistics for the English dataset are presented in Figure 4.6. As observed, they are quite similar. The shortest passage is 5 characters, as we applied a lower filter to the extracted passages. The longest passages are 1,300 characters long, also subjected to filtering. Considering statistics such as standard deviation, mean, and median, the English and German datasets do not significantly differ in terms of passage length.

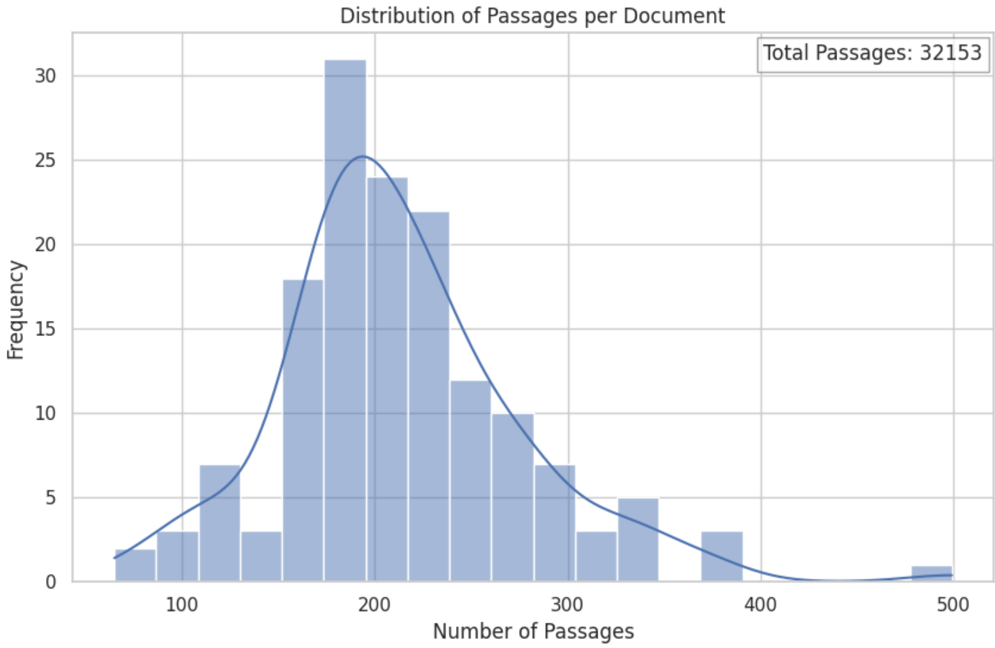

Figure 4.5.: Passages per Document of the English ER Dataset

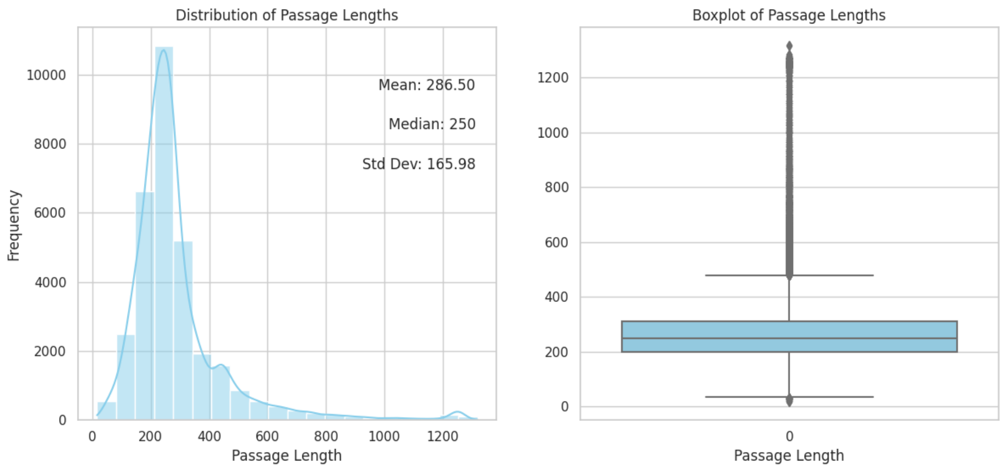

Figure 4.6.: Passage Length Distribution of the English ER Dataset

4.2. Evaluation Metrics

When it comes to Evaluation Metrics, it's important to differentiate between the components or models being evaluated. For the evaluation, we will categorize the evaluation scopes as follows:

1. Retrieval
2. Reader
3. Conversational Question Answering

Evaluating the CQU component is only possible with high quality human supervised datasets, therefore for the performance of the CQU section 4.3.4 refers to previously by other works performed benchmarks of the used zero-shot models. The exact metrics and paradigms for each individual evaluation will be discussed in the following sections.

4.2.1. Retrieval Evaluation

Evaluating a Retriever largely depends on the use-case and the evaluation data available. Since the data introduced in Section 4.1 lacks a supervised dataset for (*question*, *passages*) pairs, we will evaluate it using the synthetic dataset created, as also established in Section 4.1. This dataset consists of $(question, passages)$ pairs, where for every question, there is an exact matching passage. Therefore, this dataset is essentially a binary task, where a passage is either the correct one or not. An alternative approach would be a graded relevance task, where each passage has a certain relevance score in relation to the question. However, for our use-case, we opted for the simpler metrics Hit-Ratio (HR)@k and Mean Reciprocal Rank (MRR), instead of the Normalized Cumulative Gain (NDCG) used in benchmarks like BEIR [Thakur et al., 2021]. We chose these metrics because it's crucial for our system to retrieve the correct passage, and we don't have a relevance score for every passage in relation to every question.

Given a pair of $(q, \hat{p})$, where $\hat{p}$ corresponds to the correct passage and $\forall q, \exists! \hat{p} \in P$ and a retriever model $p_\eta(p|q) = \text{Score}(q, p)$ (as defined in Definition **??**) that assigns a score to every passage $p \in P$ in relation to the question q is used. We can rank all passages based on their relevance to q. Each passage receives a rank r_q, p based on its score in relation to q. These passages are then ranked in descending order of r_q, p, and the top k passages are added to the retrieved set R_q.

- **HR@k** This metric calculates the proportion of questions for which the correct passage is retrieved within the top k retrieved passages.

$$\text{HR@k} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} 1 & \text{if } \hat{p}_q \in R_{q,k} \\ 0 & \text{otherwise} \end{cases} \in [0, 1] \tag{4.1}$$

  The HR@k is a straightforward metric that provides a value between 0 and 1, with a higher value indicating the percentage of cases where the correct passage was retrieved within the top k passages.

- **MRR** This metric computes the mean reciprocal rank of the correct passage. It is similar to HR@k but considers the position of the correct passage $\hat{p}q$ within the ranking R_q.

$$\text{MRR} = \frac{1}{|Q|} \sum_{q \in Q} \frac{1}{r_{q,\hat{p}_q}} \in [0, 1] \tag{4.2}$$

  In an ideal system, the MRR would be 1, indicating that the correct passage is always retrieved in the first position ($r_{q,\hat{p}} = 1$) for all $q \in Q$.

4.2.2. Reader Evaluation

Evaluating the task of answer generation, particularly the MRC aspect of the reader component, presents challenges similar to those discussed for the retrieval task evaluation in Section 4.2.1. For automatic and manual evaluation, we will utilize the synthetic dataset generated in Section 4.1. This dataset comprises triples of (question, passages, answer), where *answer* refers to a gold answer that has been syntactically generated.

In the context of a triple $(q, \hat{p}, \hat{a})$, where $\hat{p}$ corresponds to the correct passage and $\hat{a}$ corresponds to the correct answer in relation to a question q, we employ a reader model $p_\theta(a' \mid q, \hat{p}) := \hat{a}$ (as defined in Definition **??**) to predict the answer $\hat{a}$ given the question q and the passage $\hat{p}$. The predicted answer a' is then evaluated using the following metrics:

- **BLUE-1:** This precision-oriented metric compares the occurrence of unigrams (words $w \in \hat{a}$) in the predicted answer a' and the gold answer $\hat{a}$.

$$\text{BLUE-1} = \frac{\sum_{w \in a'} \min(\text{count}_{a'}(w), \text{count}_{\hat{a}}(w))}{\sum_{w \in a'} \text{count}_{a'}(w)} \in [0, 1] \tag{4.3}$$

  Here, $\text{count}_{a'}(w)$ represents the number of occurrences of the word w in the predicted answer a'. Bilingual Evaluation Understudy (BLEU) is particularly useful for evaluating extractive questions [Papineni et al., 2002].

- **ROUGE-L:** This recall-oriented metric, especially Recall-Oriented Understudy for Gisting Evaluation (ROGUE)-L, compares the longest common subsequence (LCS) between the predicted answer a' and the gold answer $\hat{a}$.

$$R_{LCS} = \frac{LCS(\hat{a}, a')}{|\hat{a}|} \tag{4.4}$$

$$P_{LCS} = \frac{LCS(\hat{a}, a')}{|a'|} \tag{4.5}$$

$$\text{ROUGE-L} = \frac{(1+\beta^2)R_{LCS}P_{LCS}}{R_{LCS} + \beta^2 P_{LCS}} \in [0,1] \tag{4.6}$$

  Here, β is a parameter to balance between precision and recall. Rouge operates similarly to BLEU but focuses on lexical matching [Lin, 2004].

- **F1-BERTscore**: BERTscore is a seq-2-seq-model-based evaluation metric for comparing two text fragments: x, which is the reference, and $\hat{x}$, which is the prediction. In this context, the predicted answer a' is compared to the gold answer $\hat{a}$. Essentially, the score between two tokens, a'_i and $\hat{a}_i$, is calculated as the inner product of their respective BERT embeddings: $\text{BERT}(a'_i)^T\text{BERT}(\hat{a}_i)$. For simplicity, we'll use the $\text{BERT}(a'_i) \rightarrow a'_i$ in the following equations. The final scores of F1-BERTscore are weighted by the inverse document frequency (idf) of each word-piece token:

$$P_{BERT} = \frac{\sum_{a'_j \in a'} \text{idf}(a') \max_{\hat{a}_i \in \hat{a}}(\hat{a}_i^T a'_j)}{\sum_{a'_j \in a'} \text{idf}(a')} \tag{4.7}$$

$$R_{BERT} = \frac{\sum_{\hat{a}_i \in \hat{a}} \text{idf}(\hat{a}_i) \max_{a'_j \in a'}(\hat{a}_i^T a'_j)}{\sum_{\hat{a}_i \in \hat{a}} \text{idf}(\hat{a}_i)} \tag{4.8}$$

$$F1_{BERT} = \frac{2P_{BERT}R_{BERT}}{P_{BERT} + R_{BERT}} \tag{4.9}$$

  The advantage of F1-BERTscore lies in its reliance on semantic matching between the gold answer $\hat{a}$ and the predicted answer a' rather than mere lexical matching [Zhang et al., 2020].

- **Accuracy** This metric calculates the proportion of questions for which the predicted answer a' matches the gold answer $\hat{a}$.

$$\text{Accuracy} = \frac{1}{|Q|} \sum_{q \in Q} \begin{cases} 1 & \text{if } a' = \hat{a} \\ 0 & \text{otherwise} \end{cases} \in [0,1] \tag{4.10}$$

This metric is useful for evaluating question, answer realtions, where there is only one correct answer. In order to define if an answer is correct or not, the following approaches will be used:

- **LLM-based:** A LLM can be prompted to determine a binary value (0 or 1) indicating whether the underlying message of a' and $\hat{a}$ matches. The prompt used is based on the work of [Kamalloo et al., 2023]:

  `Question:` q

  `Gold Answer:` $\hat{a}$

  `Predicted Answer:` a'

  `Is the predicted answer correct?` Yes/No

  This appraoch is especially useful for evaluating generative questions, as it allows for semantic matching.

- **Human-based:** A human evaluator is asked to assign a binary value of 0 or 1 to indicate whether there is a match in the underlying message of a' and $\hat{a}$. 1 if the answer a' covers the information from $\hat{a}$, and 0 if it does not. The evaluator is provided with the question q, the important passage $\hat{p}$, the gold answer $\hat{a}$, and the generated answer a'. This approach is particularly useful for evaluating generative questions and closely resembles real-world applications. For this thesis two evaluaters will receive the same 100 randomly sampled answers and will be asked to assign a binary value to each answer. The inter-rater agreement will be calculated using Cohen's Kappa [Cohen, 1960].

4.2.3. Conversational Question Answering Evaluation

The most challenging aspect of evaluation within the context of the system developed here is assessing Conv QA as a holistic system. Instead of presenting evaluation metrics in this section, we will explore the approaches for evaluating Conv QA, and as final metrics, we will use those introduced in Sections 4.2.2 and 4.2.1. Generally, two approaches can be considered:

The first approach, known as **Manual Human Evaluation (manual human evaluation (MaHuEval))**, operates as follows:

1. A human evaluator initiates a conversation with a question, either based on their own intuition or using provided questions from the supervised dataset. Evaluators are encouraged to pose context-dependent questions.

2. The evaluator continues the conversation, asking follow-up questions or engaging in a general discussion with the Conv QA-System for 8-10 turns.

3. After the conversation, the evaluator has access to the retrieved passages p for each turn and all other passages P in the index. The evaluator assesses the answer provided by the system and the relevance of the retrieved passages p. They are required to provide the following information:
   - *Question Intent:* The evaluator specifies the intent of the question, which can be categorized as either *extractive*, *abstractive*, or *boolean*.
   - *Answer:* The evaluator assigns a binary value (0 or 1) to the system's answer. A score of 1 indicates a correct answer, while 0 represents an incorrect answer.
   - *Passage:* The evaluator assigns a binary value (0 or 1) to the relevance of the retrieved passages p. A score of 1 indicates that the passages are relevant for generating the correct answer, while 0 indicates irrelevance.
4. Based on the provided scores, metrics for the retriever and reader will be calculated for each turn, the entire conversation history, and the entire dataset. *Accuracy* will be computed for answers, and *HR@k* will be computed for passages. All results will be grouped by turn, enabling deeper insights into the performance and errors of the system.

To mitigate bias, human manual evaluations are conducted by two different evaluators. Each evaluator conducts 10 conversations with the system in the first round. In the second round, the evaluators initiate conversations with the first question from the 10 conversations of the other evaluator. This results in a total of 40 conversations and, ideally, 20 overlapping conversations that can be used to calculate inter-rater agreement [Cohen, 1960].

The second approach, **automatic evaluation (AutoEval)**, operates as follows. In this approach, a synthetic dataset is provided, consisting of lists of triples $(q, \hat{p}, \hat{a})$, where each triple corresponds to one turn, and a list represents a conversation history $h \in H$:

1. Initially, the system is presented with the first question q_1 from the conversation history. The response a'_1 is evaluated using the *F1-BERTscore* in comparison to the gold answer $\hat{a}$ of the history h. This score is computed for each turn, assessing the system's response against the gold answer for that turn.
2. Subsequently, to mitigate the impact of an incorrect system response on the ongoing conversation, the following question (e.g., q_2) is augmented with the gold answer $\hat{a}_1$. This augmentation helps address missing information or coreference problems caused by an erroneous response. This method, originally introduced by

Li et al. [Li et al., 2022a], improves the automatic evaluation of Conv QA systems. The model used to resolve the relationship between the gold answer and the question is the standard CQU model. Additionally, within the Conv QA history, the answer a'_1 is replaced by the gold answer $\hat{a}_1$ for the CQU unit.

3. After enhancing the question, the system is tasked with generating a response a'_2 to the augmented question q'_2. Once again, the response a'_2 is evaluated using the *F1-BERTscore* against the gold answer $\hat{a}_2$ from the history h.

4. Steps (2) and (3) are repeated until the end of the conversation history h.

5. All scores will be groupable by turn depth, enabling deeper insights into the performance and errors of the system.

Both MaHuEval and AutoEval will be employed to evaluate the Conv QA system, and the results will be reported separately.

4.3. Experimental Setup and Implementation

This section outlines the experimental setup and provides details on the implementation of the System Architecture and Framework introduced in Chapter 3. First and foremost, we apply the theoretically outlined model (M) to create an implementation decision map. Figure 4.7 illustrates a decision tree showcasing the possible implementations of the *Extract* pipeline. Figure 4.8 displays the main components, namely the *Retriever*, *Reader*, and *CQU*, in a similar decision matrix, where each column represents a decision to be made during implementation. Opting not to make a decision is also a valid choice. Developing an implementation can then be easily achieved using a decision tree, as shown in Figure A.1. For the implementation and benchmarking, ressources of the BwUniCluster[3] have been used.

4.3.1. Extraction

Figure 4.9 illustrates the extraction pipeline implemented for this thesis' PoC, resulting in the creation of the *document model* of the passages forming the knowledge source. This pipeline comprises the following steps:

1. *Extract:* Visual extraction using the Google Cloud Vision API for PDF OCR[4], followed by post-processing.

[3] `https://wiki.bwhpc.de/e/BwUniCluster2.0`
[4] `https://cloud.google.com/vision/docs/pdf`

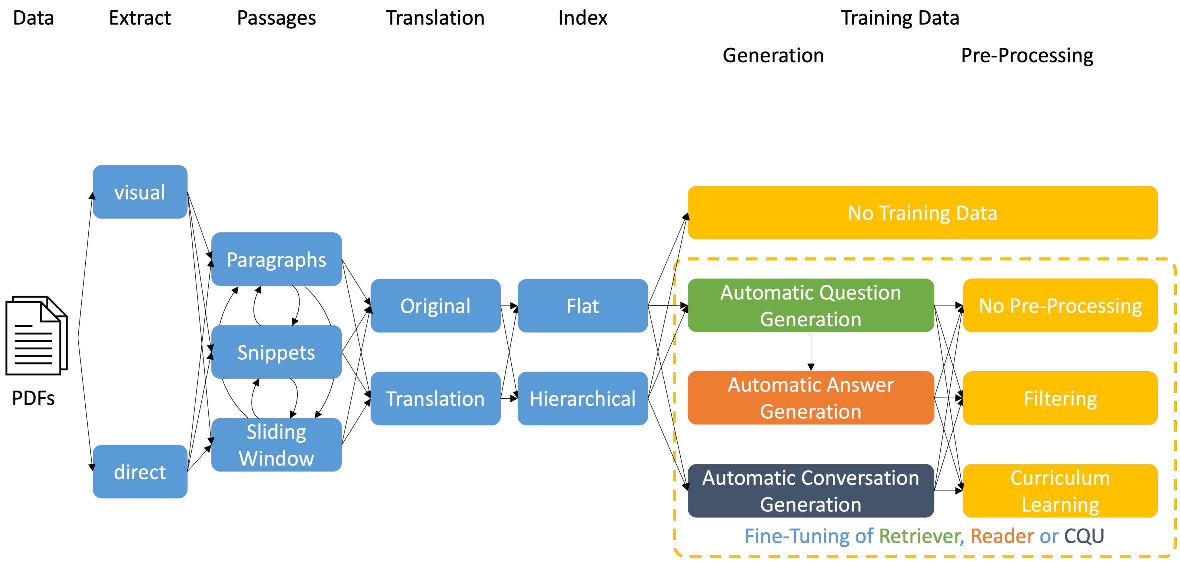

Figure 4.7.: Possible Implementations of the Extract Pipeline

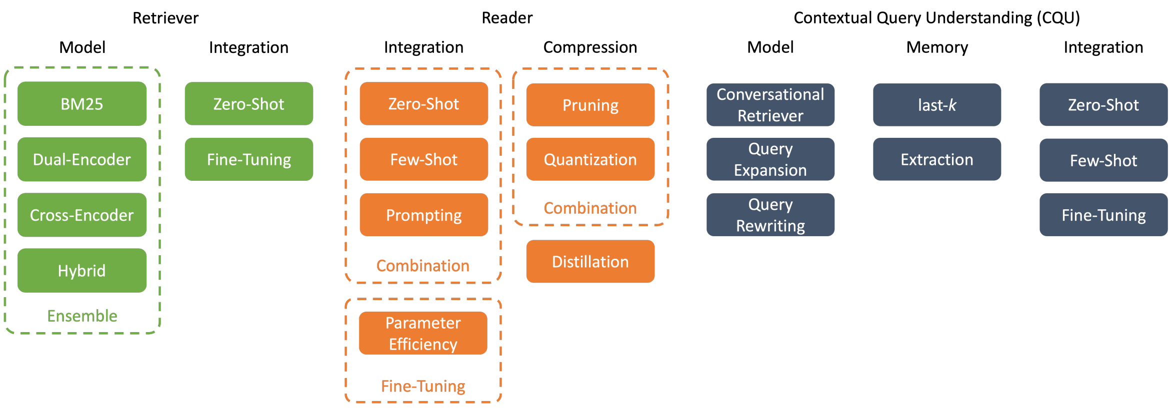

Figure 4.8.: All Components of the System Architecture

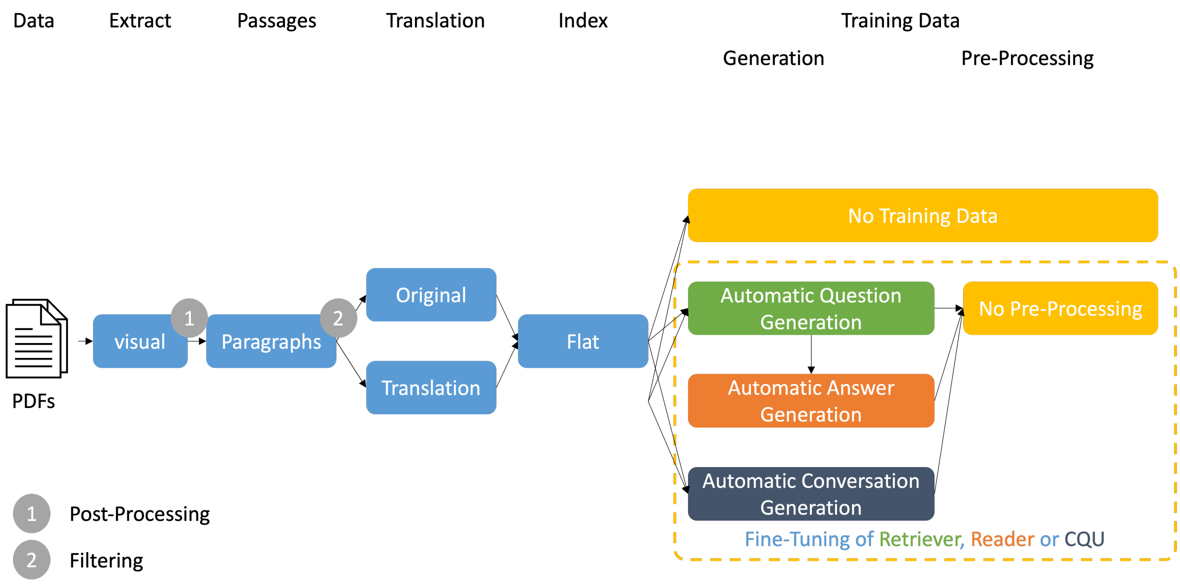

Figure 4.9.: Implemented Extraction Pipeline

2. *Passages:* Extraction of paragraphs using the NLTK tokenizer-based Text Splitter by Langchain[5], followed by filtering.

3. *Translation:* Retaining the original data and providing translations from German to English and English to German using the Google Cloud Translation API[6].

4. *Index:* A flat index, with each passage extended to include the title of the document.

5. *Training Data:* Application of Automatic Question Generation, Automatic Answer Generation, and Automatic Conversation Generation using the Llama2-7b-chat and LeoLama-7b models quantized using GPTQ.

Extraction Pipeline: In step (1), when given a PDF, the textual content of every page is extracted as plain text using OCR without paragraph awareness. This choice was made after simple qualitative experiments, which revealed that using direct methods leads to an unclean text corpus for the given PDFs. In the OCR, line breaks are detected and inserted as characters like `\n`. The textual content of separate pages is concatenated using a linebreak character `\n`. For post-processing, all `\n` characters will be replaced by spaces `" "`. This process results in a fully concatenated text corpus for every PDF.

In step (2), the NLTK-based Text Splitter receives a text corpus and identifies sentences in a first step based on punctuation. This list of sentences is then combined

[5]https://python.langchain.com/docs/modules/data_connection/document_transformers/text_splitters/split_by_token#nltk

[6]https://cloud.google.com/translate/docs/overview

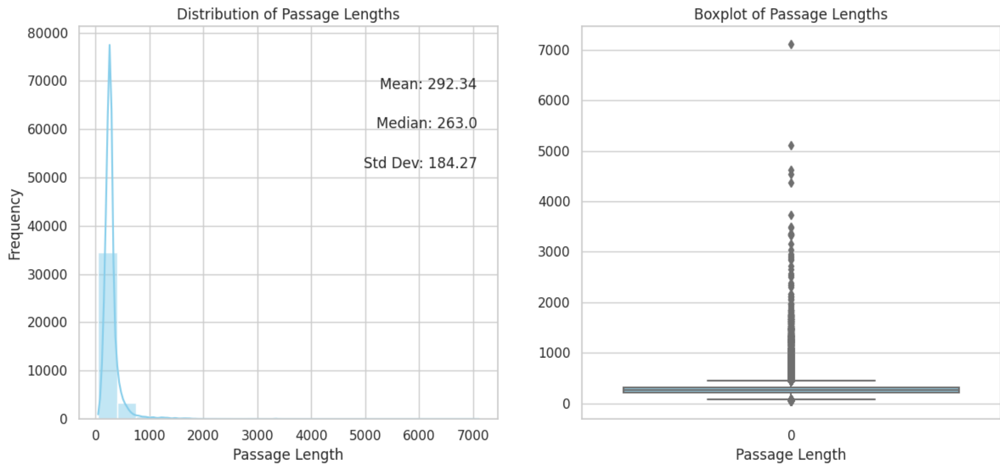

Figure 4.10.: Passage Length Distribution of the German ER Dataset before Filtering

recursively to ensure it does not exceed the desired maximum length of 240 tokens. This choice of token length was made based on reference works and their chosen token lengths (see Section 2.4). It's also important to consider the input token sizes of the later-implemented Reader components. If an identified sentence itself has more than 240 tokens, e.g., 400, it will still be kept as a 400-token-long passage. Misidentifying sentences can occur quickly, e.g. text originally corresponding to a table, which cannot be easily split into sentences:

```
Coding reference Appendix 1:  Semester 1 30 CPS A03-16-3 Semester
2 30 CP Semester 3 30 CPS Key qualifications:  Patient Orientation,
Consultation, Moderation / Presentation, English, Interdisciplinary
Collaboration 4 CP 2 CP 4 CP Scientific Writing 1 Thematic Area
I: Scientific Principles and Methods Scientific Writing ...
```

To ensure that passages do not exceed a maximum character length, identified passages will be truncated at the next empty space after reaching 1300 characters. For comparison and to understand the impact of this filtering on the knowledge source, Figure 4.10 displays the distribution of passage lengths before filtering, compared to the distribution after filtering in Figure 4.4, which illustrates the influence of filtering. The filtering process removes outliers at 1300 characters, a decision that aligns with the later components of the system.

In Step (3), every passage and all in Step (5) generated questions of the German and English ER dataset are translated using the Google Translate API. This creates a two more dataset in addition to the existing two, which are directly based on the document's language.

The final Step (4) toward the *knowledge source* is the index creation. To simplify the later RAG system, the decision was made not to use a hierarchical index structure. However, this could lead to problems when users need to find specific information within one document and want to perform metadata-filtering. To address this, all passages have the title of the document from which they were extracted concatenated to them in the following way:

{passage} + ` - ` + {documentName}

This approach enables the retriever and reader to identify passages and their corresponding documents in a single step, as opposed to hierarchical index implementations that necessitate multiple steps and potentially multiple databases for embeddings and metadata of the passages. This method has been chosen because, in this use-case, the only utilized and easy extractable metadata of a document is a documents title (e.g., *Examination Regulation for the Master Data and Computer Science*, with some even including the date). Other use-cases may require more complicated appraoches. For the translated datasets, the document name will be translated, as described in Step (3).

Data Augmentation: Step (5) of the extraction pipeline involves data augmentation. For *Question Generation* based on a given passage, the few-shot approach from *PROMPTAGATOR* [Dai et al., 2022b] has been employed. For the English ER dataset, the following prompt is used to generate a question, given a passage and i examples:

```
System Prompt:
You are an assistant generating one question given a context.  Please
start your generated question with the indicator 'Generated Question:'
Here are some examples:
{i}-Example:
Context: {passage}
Generated Question: {question}

User:
Generate the question based on the following context:
{passage}
```

The snippet in the System Prompt is looped four times to accommodate the number of examples. The examples represent different question types, which themeself indicate different answer types, in order to add some variety to the generated data. The used examples can be found in the appendix A.2.1. The Prompt is generally designed to be used with a chat fine-tuned model, as it showed better results in a qualitative evaluation. For English question generation, the *Llama2-7B-Chat-GPTQ*[7] is employed. It's a quantized version of the original Llama2-7B-Chat Model by Meta [Touvron et al., 2023]. For the German ER, the same prompt translated to English is used. For the German tasks, the *Leo-Hessianai-7B-Chat-GPTQ*[8] model is used, which is a quantized version of a German language fine-tuned Llama2-7B-Chat Model [Plüster, 2023a].

It's worth noting that the English Llama2-7B-Chat was capable of directly understanding the system prompt and generating questions with the indicator `Generated Question`, which made using the generated text easier. However, the German LLM struggled with this Few-Shot task and often generated multiple responses that didn't start with the indicator `Generierte Frage`. Additionally, the model sometimes included answers after the questions. To extract the single question from the generated text in the German LLM output, the text is cropped after a `:` and `?`, so the string enclosed by these characters is used as the question for a given passage.

The choice of Llama2-based models was based on their new state-of-the-art results in multiple benchmarks, as demonstrated by these models in the open-source LLM niche [Touvron et al., 2023]. Additionally, the work of Plüster et al. [Plüster, 2023a] on LeoLama opens the opportunity to compare the performance of German and English LLMs from the same family.

For *Answer Generation*, the previously generated questions per passage are utilized. The LLM is prompted in the following manner:

```
System Prompt:
You are an assistant who generates gold answers for a given question
and context.
User:
Generate the answer based on the following question and context:
Question: {question}
Context: {passage}
```

This prompt was applied in both German and English to generate gold answers for all datasets. A random selection of 2000 samples was drawn from the original German and English ER datasets and their translated versions, resulting in four selections of 2000 question-context pairs each[9]. On this smaller selection, *gpt-3.5-turbo* was employed with

[7] `https://huggingface.co/TheBloke/Llama-2-7b-Chat-GPTQ`
[8] `https://huggingface.co/TheBloke/leo-hessianai-7B-chat-GPTQ`
[9] This sampling approach will also be employed later to mitigate the costs associated with the OpenAI API

the mentioned prompt to generate high-quality question-context-answer triples. Additionally, *leo-hessianai-7B-chat-GPTQ* and *Llama2-7B-Chat-GPTQ* (See Section 4.3.3) were used to generate gold answers over the entire dataset. The sampled high-quality gold-answer sets will be employed in Section 4.4.3 to compare the gold-answer generation quality of the smaller models against the larger *gpt-3.5-turbo* model. In order to extend the triples to contain not only the correct context, but also other high probable contexts per question, the large DPR retriever (See Section 4.3.2) was used per triple to retrieve the top 10 passages. Those were appended to the context of the triple. This will be used in the reader evaluation later.

4.3.2. Retriever

The implemented retrievers can be found in Figure 4.11. Given the limited availability of (synthetic) data suitable for fine-tuning, the decision was made to exclusively utilize top-performing OOD zero-shot retrievers. Fine-tuning a model on such a small dataset could lead to a high risk of overfitting. To assess retriever performance, the synthetic dataset will be used for benchmarking. Among the top-performing retrievers from the BEIR benchmarks, the following three retrievers have been choosen:

1. **BM25:** This employs the standard lexical-based best match algorithm with hyperparameters $k_1 = 1.5$, $b = 0.75$, and $\epsilon = 0.25$. It is implemented using the open-source project *rank-bm25* [10].

2. **BM25 + CE:** This uses a BM25 retriever in combination with a Cross-Encoder re-ranker, which serves as the baseline established in BEIR [Thakur et al., 2021]. This combination continues to perform as the state-of-the-art. The Cross-Encoder utilized is the same as the one in the BEIR baseline implementation: *ms-marco-MiniLM-L-6-v2* [Wang et al., 2020], specifically the implementation provided on Hugging Face [11]. It has close to 17 million parameters. For German indices, a German version of *ms-marco-MiniLM-L-6* is used: *ms-marco-MiniLM-L-6-en-de-v1* [12].

3. **Large DPR:** For the large DPR, the embedding model from OpenAI, *text-embedding-ada-002* [13], is employed. The parameter size of this model is estimated to be around 350 million parameters[Muennighoff, 2022][14].

[10] `https://github.com/dorianbrown/rank_bm25`
[11] `https://huggingface.co/cross-encoder/ms-marco-MiniLM-L-6-v2`
[12] `https://huggingface.co/cross-encoder/msmarco-MiniLM-L6-en-de-v1`
[13] `https://platform.openai.com/docs/models/embeddings`
[14] As OpenAI does not publicly provide information on their model specs, there exists only estimates

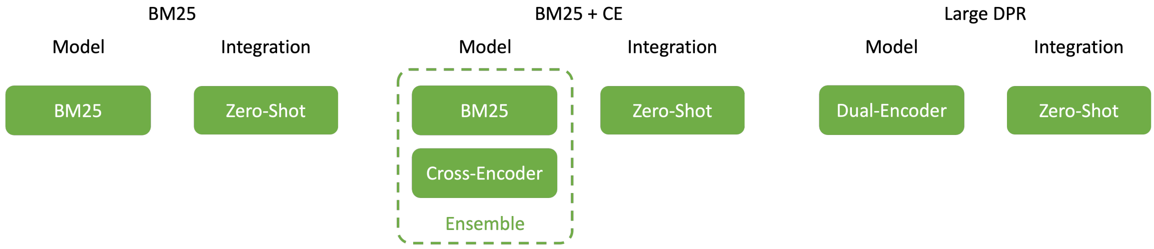

Figure 4.11.: Implemented Retrievers

These retrievers have been implemented and evaluated based on the synthetic data, and the results are presented in Section 4.4.

4.3.3. Reader

The implemented readers can be found in Figure 4.12. Due to the limited hardware resources and high-quality data, the decision was made to only use zero-shot pre-trained LLMss for the reader component and not fine-tune any. The following readers have been chosen:

1. **gpt-3.5-turbo:** GPT 3.5 Turbo is part of the OpenAI chat completion family[15]. Details on the parameters are not available. However, the latest published research by OpenAI indicates 175B parameters for the model on which GPT 3.5 Turbo was built. In general, the model is an LLM trained using human feedback on chat-like completion tasks [Ouyang et al., 2022]. It accepts up to 4,096 tokens as input.

2. **Llama2-7B-Chat-GPTQ:** This is a quantized version of the original Llama2-7B-Chat Model by Meta [Touvron et al., 2023]. The model is an LLM trained on chat-instruction datasets [Ouyang et al., 2022]. The model is available on Hugging Face[16]. The model is quantized using GPTQ [Muennighoff, 2022]. However, the provider did not report any benchmarks indicating the accuracy drop of the GPTQ version compared to the original. It accepts up to 4,096 tokens as input.

3. **leo-hessianai-7B-chat-GPTQ:** *Linguistisch Erweitertes Offenes Language Model* (LeoLM) is a German tasks fine-tuned version of the original Llama2-7B-Chat model. The original Llama2 is fine-tuned on a large corpus of German language texts, and multiple adjustments are applied to overcome the problem of forgetting. This results in a maximum input token length of 8,000 tokens. In addition to original German chat instruction-based datasets, automatic translated English

[15] https://platform.openai.com/docs/models/gpt-3-5
[16] https://huggingface.co/TheBloke/Llama-2-7b-Chat-GPTQ

benchmark datasets are used as well. Benchmarking results indicate an increase in German language capabilities while maintaining English language task performance [Plüster, 2023b].

Generally, **gpt-3.5-turbo** acts as a higher-end benchmark. It is a state-of-the-art LLM with high popularity in the media and developer community. **Llama2-7B-Chat-GPTQ** should be a hardware resource-efficient alternative pre-trained model, especially the quantized version that can run on consumer hardware. **leo-hessianai-7B-chat-GPTQ** is representative of a German native model, providing further insights into language dependencies and the performance of LLMs, as the use case involves multiple indices in German and English.

Additional approaches, such as evaluating different fine-tuning strategies, have been excluded, as they would open up a vast problem space that cannot be adequately covered within the scope of this thesis. Therefore, the thesis predominantly concentrates on a comprehensive PoC to demonstrate and evaluate one possible implementation approach of the framework introduced in Section 3, utilizing the mentioned LLMs.

However, to address specific questions in the evaluation (See Section 4.4.3), such as *How many contexts can the model receive and still determine the correct answer?*, the reader prompt was structured to accept multiple contexts as inputs to the question:

```
System Prompt:
You are an assistant who generates an answer given multiple contexts,
which not necessarily contain the answer.  You only respond to the
"Assistant".  Please generate an answer or state, "I can't provide
an answer given the context." if you can not provide an answer based
on the provided context.
User:
Please generate an answer for the following question:
{question}
Use the following contexts:
for context in contexts:
{context}
Assistant:
```

This prompt is applied consistently across all models. In cases where the model is applied on a German dataset, a German version of this prompt is utilized.

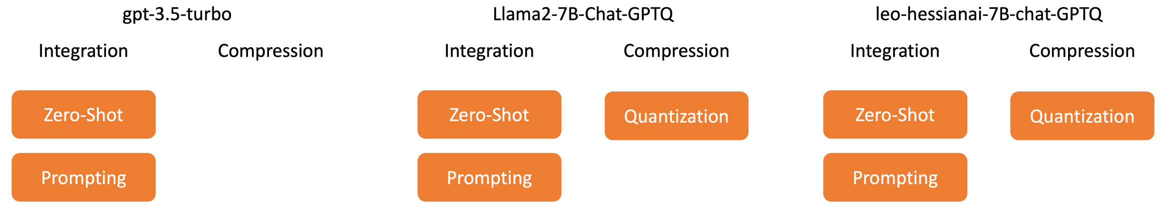

Figure 4.12.: Implemented Readers

4.3.4. CQU

4.4. Experimental Results

This section evaluates the implemented system and the introduced framework, using metrics introduced in Section 4.2. The evaluation covers the following components: *Extraction*, especially synthetic data generation (Section 4.4.1), *Retriever* (Section 4.4.2), *Reader* (Section 4.4.3), and the complete Conv QA system (Section 4.4.4). The evaluation is based on the synthetic dataset described in Section 4.3.1, generated from the dataset introduced in Section 4.1.

4.4.1. Data Augmentation Quality

As detailed in Section 4.3.1, the two main indices, namely *indexEnglish* and *indexGerman*, are based on the corresponding ERs. During the generation of the question-context datasets, qualitative issues arose when using *leo-hessianai-7B-chat-GPTQ*. As indicated by System Prompt 4.3.1, the model was prompted to start the generated answer after a defined indication text. In multiple cases, the German model *leo-hessianai-7B-chat-GPTQ* was unable to fulfill this request, necessitating post-processing as described in Section 4.3.1. This issue was not present when using the English model *Llama2-7B-Chat-GPTQ*.

In addition to this qualitative issue, a quantitative evaluation is possible when comparing the retrieval performance on both datasets and their translated versions: *translated-indexEnglish* and *translated-indexGerman*. The performance results of the large DPR retriever are shown in Figure 4.13. They indicate, despite a small language performance gap, a significant difference in performance between the two indices *indexEnglish* and *indexGerman* and their corresponding translated versions. Of course, a gap in question difficulty could also be the case, but as both models used the same examples, it can be assumed that they did not necessarily generate significantly different questions in terms of difficulty to answer (e.g., the English model could mainly generate extractive questions, which are easier to find the corresponding passage to). This can be excluded

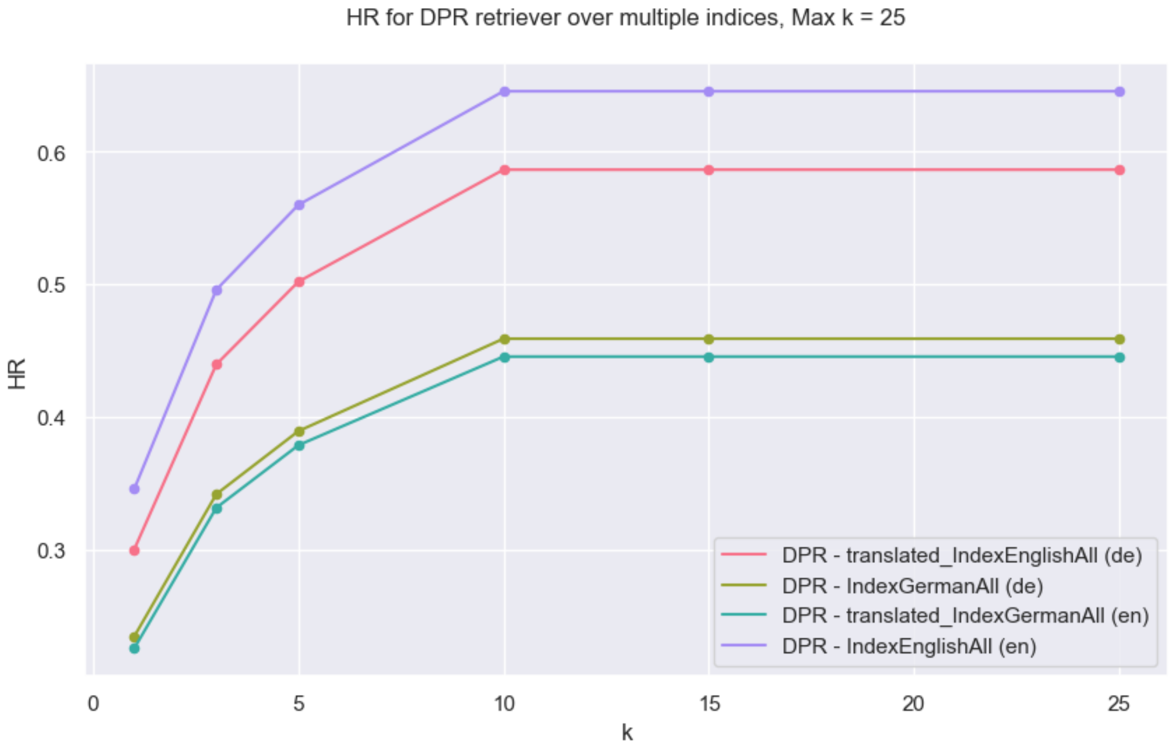

Figure 4.13.: Retriever performance comparison over all datasets

when taking a look at Figure 4.18, this is indicating a high performance of the *BM25* retriever which could relate to an at least keyword based close realtion between questions and passages, therefore easier extractive questions. This hypothesis can further be supported when taking a qualitative look at questions generated in the *indexGerman*, an example:

```
Generierte Antwort:
Was ist der Wert von $(4+5) \\times {(4+5) \\div {2 \\div {1 \\
times {2}}}}$, wobei die mündliche Prüfung von einer Person
abgenommen wird, die nicht prüft?

Gegebener Kontext:
Das Ergebnis ist der zu prüfenden Person im Anschluss an die münd-
liche Prüfung bekannt zu geben.\n\n(4) (5) Mündliche Prüfungen
werden vor einer prüfenden Person in Gegenwart einer beisitzen-
den Person gem. 6 Abs.\n\n4 abgelegt. - Prüfungsordnung Mathematik
MA (2022-10-05)
```

This observation highlights a discernible zero-shot performance gap between the English model *Llama2-7B-Chat-GPTQ* and the German model *leo-hessianai-7B-chat-GPTQ*. Consequently, the subsequent evaluations will concentrate on the *indexEnglish* and *translated-indexEnglish* datasets.

How does the gold-answer-generation performance differ between the models?

4.4.2. Retrieval Results

The evaluation based on the described metrics (See Section 4.2) delivers multiple insights:

1. COMPARISON BM25 AND DPR
2. The ensemble of *BM25-CE* outperforms the regular *BM25* retriever for three out of four indices (all except for the *indexGerman*) when considering small values of k. Figure 4.14 compares the MRR between the standard *BM25* retriever and *BM25-CE* retriever with the BM25 k-value set to 500. As the *Reader* component ideally takes as small a set of passages as possible as input, the *BM25-CE* retriever is the better choice. The internal k-value of the first stage *BM25* retriever can be as large as possible, as it only improves the performance of the ensemble. However, depending on the dataset and use-case, an increasing k-value may not necessarily improve performance, as it seems to exist an upper performance limit in the cross-encoder component (See Figure 4.15).
3. LANGUAGE COMPARISON BM25 AND DPR
4. In the comparison of *BM25* and DPR retrievers for different languages, the final choice of the *Retriever* component should consider the best MRR value for English and German datasets for $k \leq 10$. Figure 4.17 compares *BM25-CE-500* with the large *DPR* retriever over all datasets. The DPR retriever significantly outperforms the *BM25-CE-500* retriever for all datasets, indicating a high multi-language zero-shot performance of the *text-embedding-ada-002* model by OpenAI. Nevertheless, considering the diffference in parameters (350 million vs. 17 million), the *BM25-CE-500* retriever is a good alternative.

The influence of the Retriever component will be further evaluated in Section 4.4.4. Here the bottlenecks of the system will be identified and discussed.

4.4.3. Reader Results

QUESTIONS I WANT TO ANSWER:

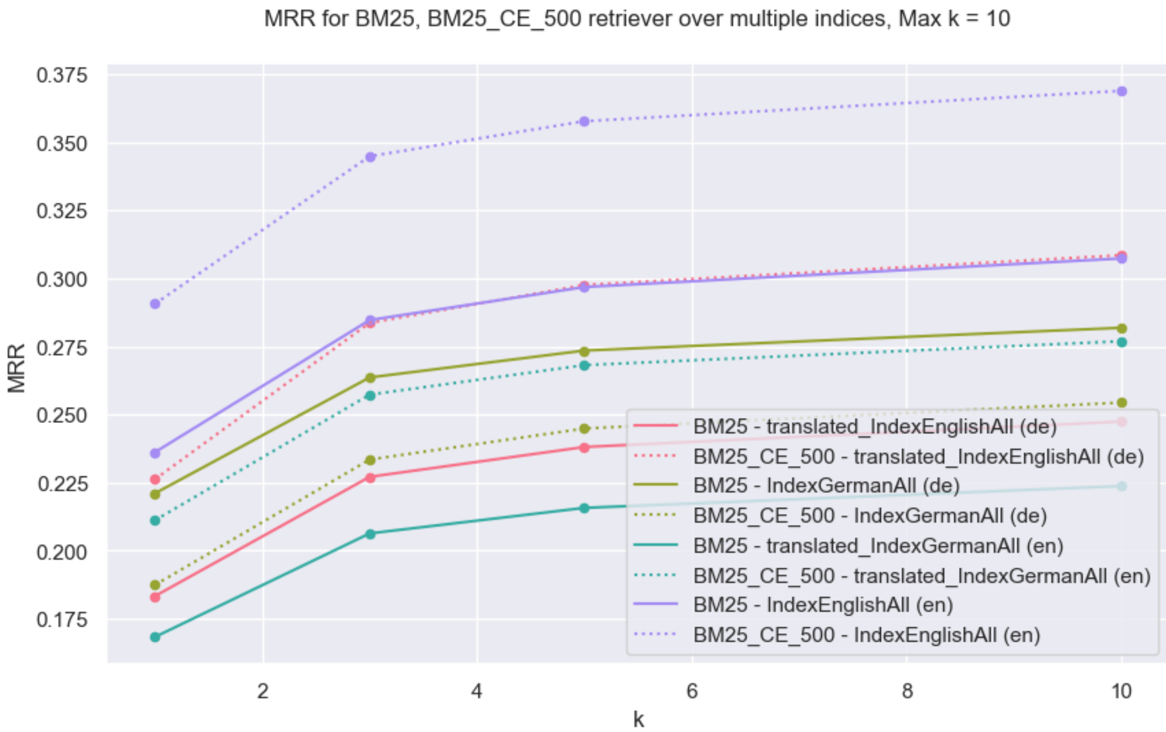

Figure 4.14.: BM25 and BM25-CE MRR perfromance for $k = 10$

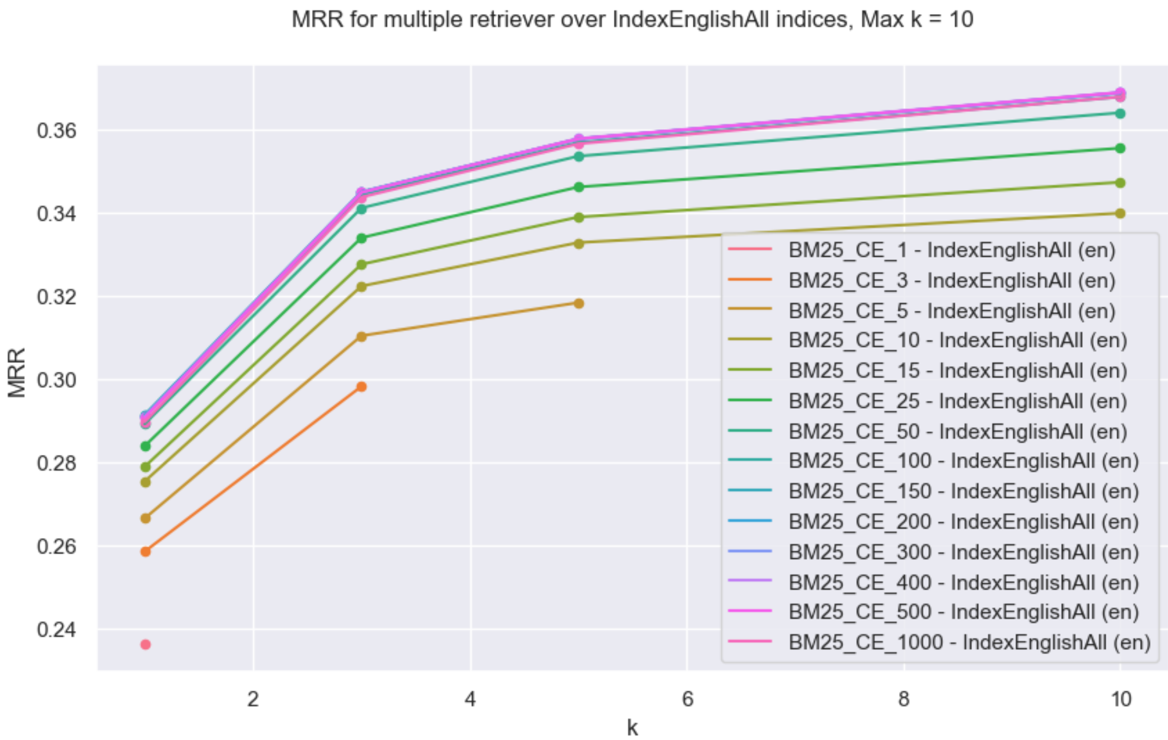

Figure 4.15.: BM25-CE HR perfromance for different BM25 k-values

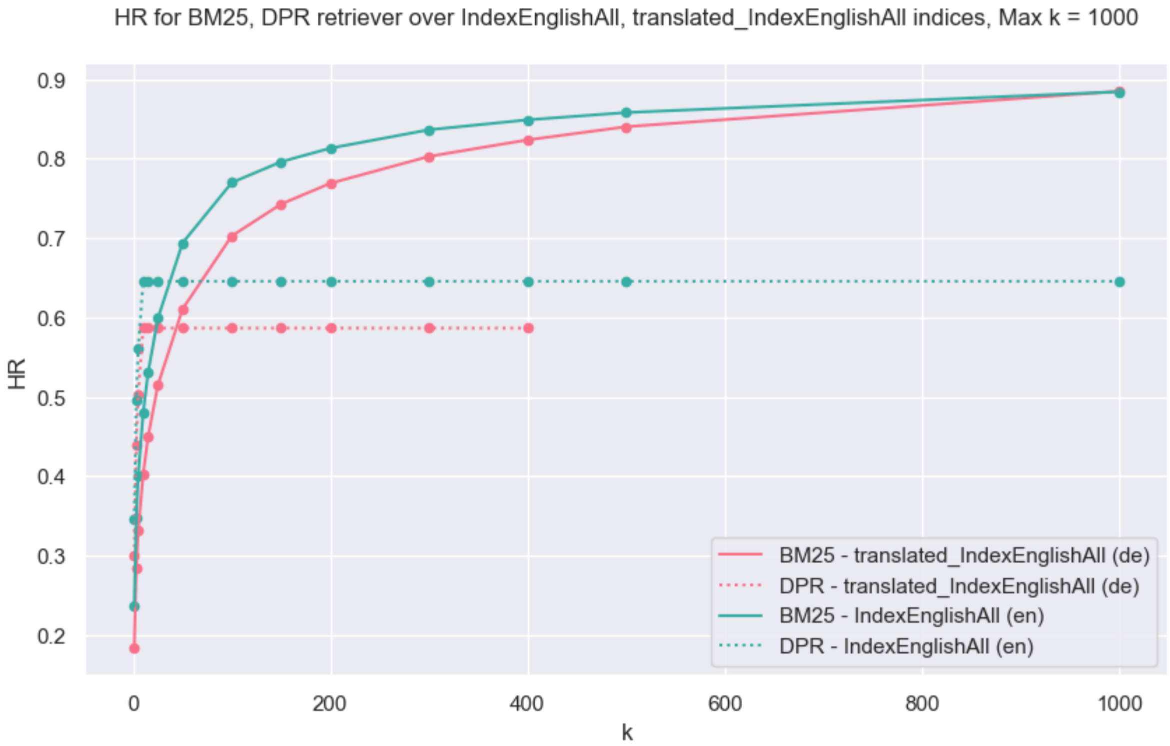

Figure 4.16.: DPR and BM25 performance comparison for indexEnglish and translated-indexEnglish

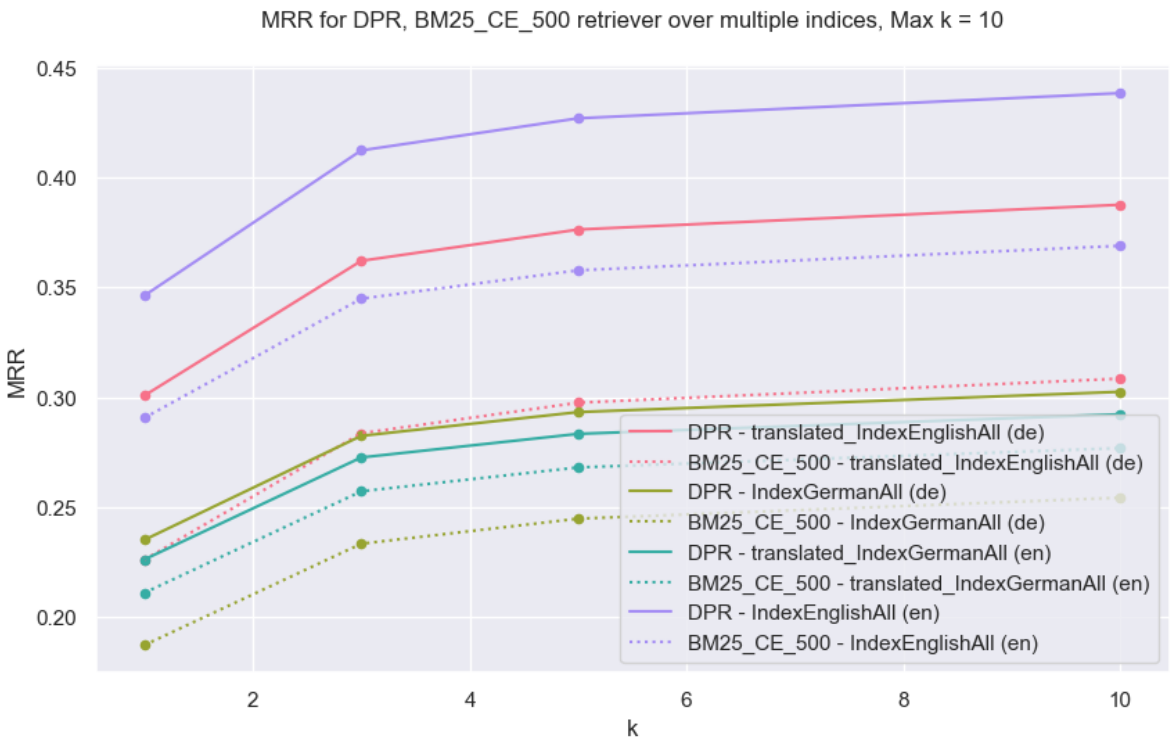

Figure 4.17.: BM25-CE and DPR MRR perfromance for $k = 10$

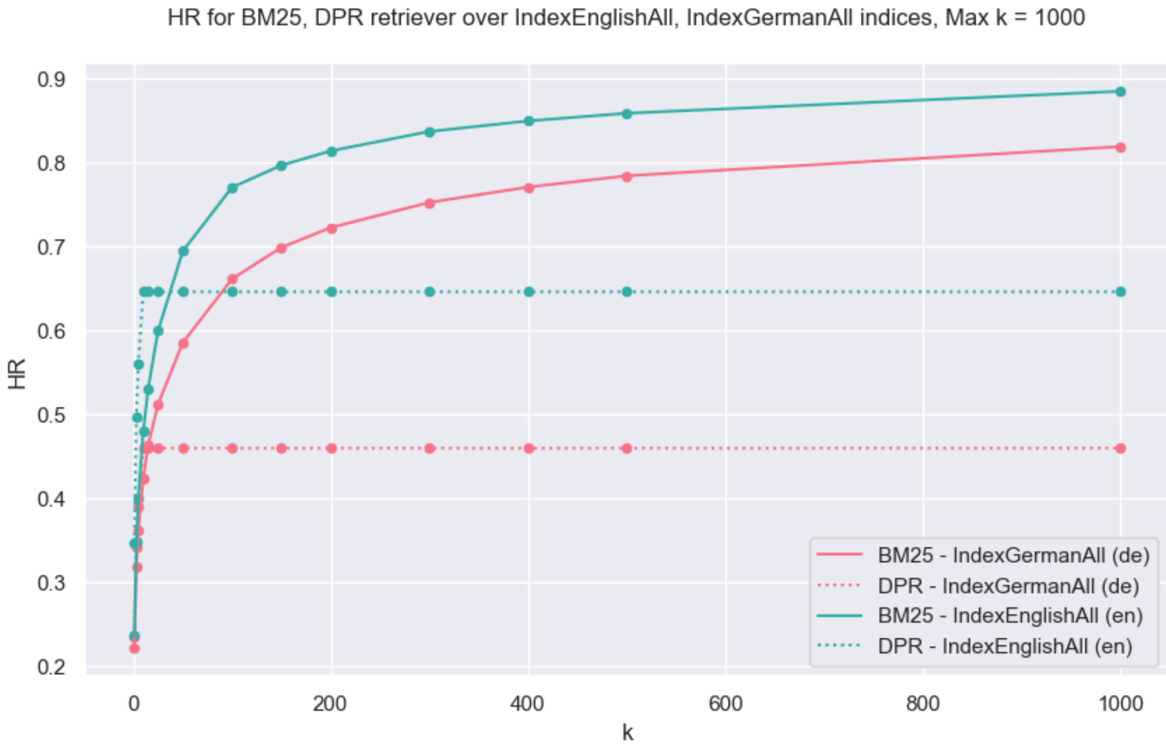

Figure 4.18.: BM25 and Large DPR performance comparison for $k = 1000$

1. How many context passages will still lead to a correct answer?
2. Is the order of the contexts important?
3. Can the reader identify if the correct context was not provided?
4. How much do the models differ in language?
5. How much better is OpenAI?

4.4.4. Conversational Question Answering Results

5. Conclusions and Future Work

This chapter is the conclusion of the thesis.

A. Appendix A

A.1. Implementation Trees

A.1.1. Example

A.1.2. Extract

A.2. Data Augmentation Implementation

A.2.1. Examples for Few-Shot Prompt

1. Example:
Context: The master's program in Data and Computer Science includes an application area. Annex 3 lists the possible application areas. Upon request, the examination board can also approve a different application area. - Master Data and Computer Science
Question: I study Data and Computerscience. Can I choose application areas that are not listed?

2. Example:
Context: (5) The final failure in a mandatory module leads to the loss of the examination claim. In elective mandatory modules, if provided for in the module handbook, the failure can be compensated for by the successful completion of another elective mandatory module or another performance within the respective module. § 4 Paragraph 2 remains unaffected. - Bachelor Physics
Question: When does the failure of an exam lead to the termination of the study?

3. Example:
Context: (3) The application must be made in writing to the examination

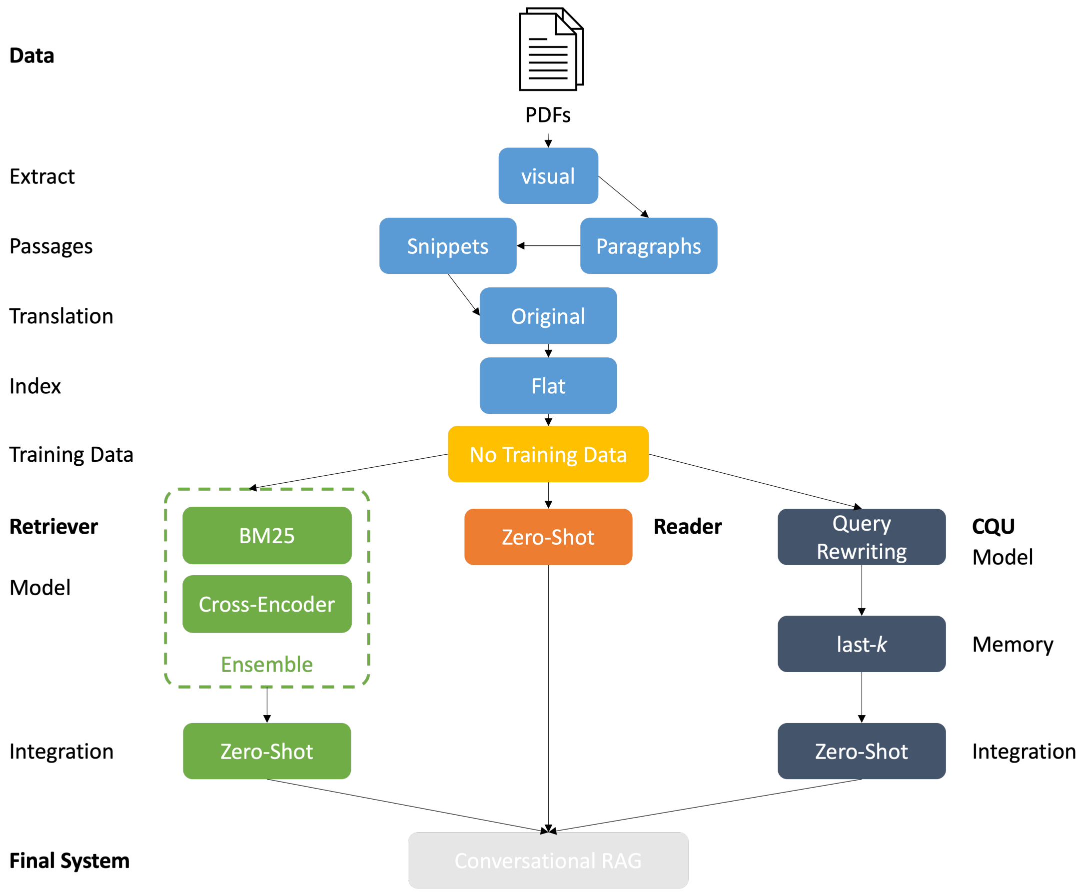

Figure A.1.: Example Implementation Zero-Shot Baseline

board. It is the responsibility of the applicant to provide the necessary information about the performance to be recognized. The burden of proof for the existence of a significant difference in academic achievements lies with Heidelberg University; the obligations of the applicant, especially according to Sentence 1 and Sentence 2, remain unaffected. The burden of proof for the existence of equivalence in non-academic achievements lies with the applicant. - Bachelor Philosophy
Question: What would happen, if I can't provide information on my academic achievements?

4. Example:
Context: The intermediate examination consists of successful participation in the exercises for beginners in the subjects Civil Law, Public Law, and Criminal Law. The partial performances of the exercise (homework and supervisory work under examination conditions) must in principle be performed in the exercise of a semester; § 4 paragraph 5 remains unaffected. - Civil Law
Question: What are the components of the intermediate examination in Civil Law, Public Law, and Criminal Law?

Bibliography

[noa, 2023a] (2023a). DeepSparse. original-date: 2020-12-14T17:40:38Z.

[noa, 2023b] (2023b). Quantize Transformers models.

[Anil et al., 2023] Anil, R., Dai, A. M., Firat, O., Johnson, M., Lepikhin, D., Passos, A., Shakeri, S., Taropa, E., Bailey, P., Chen, Z., Chu, E., Clark, J. H., Shafey, L. E., Huang, Y., Meier-Hellstern, K., Mishra, G., Moreira, E., Omernick, M., Robinson, K., Ruder, S., Tay, Y., Xiao, K., Xu, Y., Zhang, Y., Abrego, G. H., Ahn, J., Austin, J., Barham, P., Botha, J., Bradbury, J., Brahma, S., Brooks, K., Catasta, M., Cheng, Y., Cherry, C., Choquette-Choo, C. A., Chowdhery, A., Crepy, C., Dave, S., Dehghani, M., Dev, S., Devlin, J., Díaz, M., Du, N., Dyer, E., Feinberg, V., Feng, F., Fienber, V., Freitag, M., Garcia, X., Gehrmann, S., Gonzalez, L., Gur-Ari, G., Hand, S., Hashemi, H., Hou, L., Howland, J., Hu, A., Hui, J., Hurwitz, J., Isard, M., Ittycheriah, A., Jagielski, M., Jia, W., Kenealy, K., Krikun, M., Kudugunta, S., Lan, C., Lee, K., Lee, B., Li, E., Li, M., Li, W., Li, Y., Li, J., Lim, H., Lin, H., Liu, Z., Liu, F., Maggioni, M., Mahendru, A., Maynez, J., Misra, V., Moussalem, M., Nado, Z., Nham, J., Ni, E., Nystrom, A., Parrish, A., Pellat, M., Polacek, M., Polozov, A., Pope, R., Qiao, S., Reif, E., Richter, B., Riley, P., Ros, A. C., Roy, A., Saeta, B., Samuel, R., Shelby, R., Slone, A., Smilkov, D., So, D. R., Sohn, D., Tokumine, S., Valter, D., Vasudevan, V., Vodrahalli, K., Wang, X., Wang, P., Wang, Z., Wang, T., Wieting, J., Wu, Y., Xu, K., Xu, Y., Xue, L., Yin, P., Yu, J., Zhang, Q., Zheng, S., Zheng, C., Zhou, W., Zhou, D., Petrov, S., and Wu, Y. (2023). PaLM 2 Technical Report. arXiv:2305.10403 [cs].

[Bajaj et al., 2016] Bajaj, P., Campos, D., Craswell, N., Deng, L., Gao, J., Liu, X., Majumder, R., McNamara, A., Mitra, B., Nguyen, T., et al. (2016). Ms marco: A human generated machine reading comprehension dataset. *arXiv preprint arXiv:1611.09268*.

[Brown et al., 2020] Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D. M., Wu, J., Winter, C., Hesse, C., Chen, M., Sigler, E., Litwin, M., Gray, S., Chess, B., Clark, J., Berner, C., McCandlish, S., Radford, A., Sutskever, I., and Amodei, D. (2020). Language Models are Few-Shot Learners. arXiv:2005.14165 [cs].

[Chen, 2021] Chen, H. (2021). Improving Out-of-Domain Question Answering with Mixture of Experts.

[Chung et al., 2022] Chung, H. W., Hou, L., Longpre, S., Zoph, B., Tay, Y., Fedus, W., Li, Y., Wang, X., Dehghani, M., Brahma, S., Webson, A., Gu, S. S., Dai, Z., Suzgun, M., Chen, X., Chowdhery, A., Castro-Ros, A., Pellat, M., Robinson, K., Valter, D., Narang, S., Mishra, G., Yu, A., Zhao, V., Huang, Y., Dai, A., Yu, H., Petrov, S., Chi, E. H., Dean, J., Devlin, J., Roberts, A., Zhou, D., Le, Q. V., and Wei, J. (2022). Scaling Instruction-Finetuned Language Models. arXiv:2210.11416 [cs].

[Cohen, 1960] Cohen, J. (1960). A Coefficient of Agreement for Nominal Scales. *Educational and Psychological Measurement*, 20(1):37–46.

[Dai et al., 2022a] Dai, Z., Chaganty, A. T., Zhao, V., Amini, A., Rashid, Q. M., Green, M., and Guu, K. (2022a). Dialog Inpainting: Turning Documents into Dialogs. Publisher: arXiv Version Number: 2.

[Dai et al., 2022b] Dai, Z., Zhao, V. Y., Ma, J., Luan, Y., Ni, J., Lu, J., Bakalov, A., Guu, K., Hall, K. B., and Chang, M.-W. (2022b). Promptagator: Few-shot Dense Retrieval From 8 Examples. arXiv:2209.11755 [cs].

[Dalton et al., 2020] Dalton, J., Xiong, C., and Callan, J. (2020). TREC CAsT 2019: The Conversational Assistance Track Overview. arXiv:2003.13624 [cs].

[Dasigi et al., 2021] Dasigi, P., Lo, K., Beltagy, I., Cohan, A., Smith, N. A., and Gardner, M. (2021). A Dataset of Information-Seeking Questions and Answers Anchored in Research Papers. arXiv:2105.03011 [cs].

[Devlin et al., 2019] Devlin, J., Chang, M.-W., Lee, K., and Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. arXiv:1810.04805 [cs].

[Dimitrakis et al., 2020] Dimitrakis, E., Sgontzos, K., and Tzitzikas, Y. (2020). A survey on question answering systems over linked data and documents. *Journal of Intelligent Information Systems*, 55(2):233–259.

[Ding et al., 2022] Ding, Y., Huang, Z., Wang, R., Zhang, Y., Chen, X., Ma, Y., Chung, H., and Han, S. C. (2022). V-Doc : Visual questions answers with Documents. arXiv:2205.13724 [cs].

[Elgohary et al., 2019] Elgohary, A., Peskov, D., and Boyd-Graber, J. (2019). Can You Unpack That? Learning to Rewrite Questions-in-Context. In *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP)*, pages 5918–5924, Hong Kong, China. Association for Computational Linguistics.

[Etezadi and Shamsfard, 2023] Etezadi, R. and Shamsfard, M. (2023). The state of the art in open domain complex question answering: a survey. *Applied Intelligence*, 53(4):4124–4144.

[Farea et al., 2022] 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:2209.12617 [cs].

[Ferrucci, 2012] Ferrucci, D. A. (2012). Introduction to “This is Watson”. *IBM Journal of Research and Development*, 56(3.4):1:1–1:15. Conference Name: IBM Journal of Research and Development.

[Frantar and Alistarh, 2023] Frantar, E. and Alistarh, D. (2023). SparseGPT: Massive Language Models Can Be Accurately Pruned in One-Shot.

[Frantar et al., 2023a] Frantar, E., Ashkboos, S., Hoefler, T., and Alistarh, D. (2023a). GPTQ: Accurate Post-Training Quantization for Generative Pre-trained Transformers. arXiv:2210.17323 [cs].

[Frantar et al., 2023b] Frantar, E., Singh, S. P., and Alistarh, D. (2023b). Optimal Brain Compression: A Framework for Accurate Post-Training Quantization and Pruning. arXiv:2208.11580 [cs].

[Gao et al., 2022] Gao, J., Xiong, C., Bennett, P., and Craswell, N. (2022). Neural Approaches to Conversational Information Retrieval. arXiv:2201.05176 [cs].

[Gholami et al., 2021] Gholami, A., Kim, S., Dong, Z., Yao, Z., Mahoney, M. W., and Keutzer, K. (2021). A Survey of Quantization Methods for Efficient Neural Network Inference. arXiv:2103.13630 [cs].

[Gholami and Noori, 2021] Gholami, S. and Noori, M. (2021). Zero-Shot Open-Book Question Answering. arXiv:2111.11520 [cs].

[Green et al., 1961] Green, B. F., Wolf, A. K., Chomsky, C., and Laughery, K. (1961). Baseball: an automatic question-answerer. In *Papers presented at the May 9-11, 1961,* *western joint IRE-AIEE-ACM computer conference*, IRE-AIEE-ACM '61 (Western), pages 219–224, New York, NY, USA. Association for Computing Machinery.

[Gu et al., 2023] Gu, Y., Dong, L., Wei, F., and Huang, M. (2023). Knowledge Distillation of Large Language Models. arXiv:2306.08543 [cs].

[Gupta et al., 2020] Gupta, S., Rawat, B. P. S., and Yu, H. (2020). Conversational Machine Comprehension: a Literature Review. In *Proceedings of the 28th International Conference on Computational Linguistics*, pages 2739–2753, Barcelona, Spain (Online). International Committee on Computational Linguistics.

[Gururangan et al., 2020] Gururangan, S., Marasović, A., Swayamdipta, S., Lo, K., Beltagy, I., Downey, D., and Smith, N. A. (2020). Don't Stop Pretraining: Adapt Language Models to Domains and Tasks. *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 8342–8360. Conference Name: Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics Place: Online Publisher: Association for Computational Linguistics.

[Guu et al., 2020] Guu, K., Lee, K., Tung, Z., Pasupat, P., and Chang, M.-W. (2020). REALM: Retrieval-Augmented Language Model Pre-Training. arXiv:2002.08909 [cs].

[Hao et al., 2022] Hao, T., Li, X., He, Y., Wang, F. L., and Qu, Y. (2022). Recent progress in leveraging deep learning methods for question answering. *Neural Computing and Applications*, 34(4):2765–2783.

[Harabagiu et al., 2003] Harabagiu, S. M., Maiorano, S. J., and Paşca, M. A. (2003). Open-domain textual question answering techniques. *Natural Language Engineering*, 9(3):231–267. Publisher: Cambridge University Press.

[He et al., 2020] He, P., Liu, X., Gao, J., and Chen, W. (2020). DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION.

[Hinton et al., 2015] Hinton, G., Vinyals, O., and Dean, J. (2015). Distilling the Knowledge in a Neural Network. arXiv:1503.02531 [cs, stat].

[Houlsby et al., 2019] Houlsby, N., Giurgiu, A., Jastrzebski, S., Morrone, B., de Laroussilhe, Q., Gesmundo, A., Attariyan, M., and Gelly, S. (2019). Parameter-Efficient Transfer Learning for NLP. arXiv:1902.00751 [cs, stat].

[Hu et al., 2021] Hu, E., Shen, Y., Wallis, P., Allen-Zhu, Z., Li, Y., Wang, S., Wang, L., and Chen, W. (2021). LORA: LOW-RANK ADAPTATION OF LARGE LANGUAGE MODELS.

[Huang et al., 2022] Huang, Y., Chen, Y., Yu, Z., and McKeown, K. (2022). In-context Learning Distillation: Transferring Few-shot Learning Ability of Pre-trained Language Models. arXiv:2212.10670 [cs].

[Huggingface, 2023] Huggingface (2023). PEFT.

[Izacard and Grave, 2021] Izacard, G. and Grave, E. (2021). Leveraging Passage Retrieval with Generative Models for Open Domain Question Answering. In *Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume*, pages 874–880, Online. Association for Computational Linguistics.

[Izacard et al., 2022] Izacard, G., Lewis, P., Lomeli, M., Hosseini, L., Petroni, F., Schick, T., Dwivedi-Yu, J., Joulin, A., Riedel, S., and Grave, E. (2022). Atlas: Few-shot Learning with Retrieval Augmented Language Models. Publisher: arXiv Version Number: 3.

[Jiang et al., 2023] Jiang, Y., Chan, C., Chen, M., and Wang, W. (2023). Lion: Adversarial Distillation of Closed-Source Large Language Model. arXiv:2305.12870 [cs].

[Johnson et al., 2017] Johnson, J., Douze, M., and Jégou, H. (2017). Billion-scale similarity search with GPUs. arXiv:1702.08734 [cs].

[Jurafsky and Martin, 2023] Jurafsky, D. and Martin, J. H. (2023). *Speech and Language Processing*. Palo Alto, 3 edition.

[Kamalloo et al., 2023] Kamalloo, E., Dziri, N., Clarke, C. L. A., and Rafiei, D. (2023). Evaluating Open-Domain Question Answering in the Era of Large Language Models. arXiv:2305.06984 [cs].

[Karpukhin et al., 2020] Karpukhin, V., Oğuz, B., Min, S., Lewis, P., Wu, L., Edunov, S., Chen, D., and Yih, W.-t. (2020). Dense Passage Retrieval for Open-Domain Question Answering. arXiv:2004.04906 [cs].

[Khashabi et al., 2022] Khashabi, D., Kordi, Y., and Hajishirzi, H. (2022). UnifiedQA-v2: Stronger Generalization via Broader Cross-Format Training. arXiv:2202.12359 [cs].

[Khattab and Zaharia, 2020] Khattab, O. and Zaharia, M. (2020). ColBERT: Efficient and Effective Passage Search via Contextualized Late Interaction over BERT. arXiv:2004.12832 [cs].

[Langchain, 2023] Langchain (2023). langchain-ai/langchain: Building applications with LLMs through composability.

[Langchain, 2023] Langchain (2023). Question Answering | Langchain.

[Lester et al., 2021] Lester, B., Al-Rfou, R., and Constant, N. (2021). The Power of Scale for Parameter-Efficient Prompt Tuning. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 3045–3059, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

[Lewis et al., 2019] Lewis, M., Liu, Y., Goyal, N., Ghazvininejad, M., Mohamed, A., Levy, O., Stoyanov, V., and Zettlemoyer, L. (2019). BART: Denoising Sequence-to-Sequence Pre-training for Natural Language Generation, Translation, and Comprehension.

[Lewis et al., 2021] Lewis, P., Perez, E., Piktus, A., Petroni, F., Karpukhin, V., Goyal, N., Küttler, H., Lewis, M., Yih, W.-t., Rocktäschel, T., Riedel, S., and Kiela, D. (2021). Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks. arXiv:2005.11401 [cs].

[Li et al., 2022a] Li, H., Gao, T., Goenka, M., and Chen, D. (2022a). Ditch the Gold Standard: Re-evaluating Conversational Question Answering. arXiv:2112.08812 [cs].

[Li et al., 2022b] Li, J., Xu, Y., Lv, T., Cui, L., Zhang, C., and Wei, F. (2022b). DiT: Self-supervised Pre-training for Document Image Transformer. arXiv:2203.02378 [cs].

[Li et al., 2022c] Li, S., Chen, J., Shen, Y., Chen, Z., Zhang, X., Li, Z., Wang, H., Qian, J., Peng, B., Mao, Y., Chen, W., and Yan, X. (2022c). Explanations from Large Language Models Make Small Reasoners Better. arXiv:2210.06726 [cs].

[Li and Liang, 2021] Li, X. L. and Liang, P. (2021). Prefix-Tuning: Optimizing Continuous Prompts for Generation. arXiv:2101.00190 [cs].

[Lin, 2004] Lin, C.-Y. (2004). ROUGE: A Package for Automatic Evaluation of Summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics.

[Ling et al., 2023] Ling, C., Zhao, X., Lu, J., Deng, C., Zheng, C., Wang, J., Chowdhury, T., Li, Y., Cui, H., Zhang, X., Zhao, T., Panalkar, A., Cheng, W., Wang, H., Liu, Y., Chen, Z., Chen, H., White, C., Gu, Q., Pei, J., and Zhao, L. (2023). Domain Specialization as the Key to Make Large Language Models Disruptive: A Comprehensive Survey. arXiv:2305.18703 [cs].

[Liu et al., 2022] Liu, H., Tam, D., Muqeeth, M., Mohta, J., Huang, T., Bansal, M., and Raffel, C. (2022). Few-Shot Parameter-Efficient Fine-Tuning is Better and Cheaper than In-Context Learning. Publisher: arXiv Version Number: 2.

[Liu et al., 2021a] Liu, X., Zheng, Y., Du, Z., Ding, M., Qian, Y., Yang, Z., and Tang, J. (2021a). GPT Understands, Too. arXiv:2103.10385 [cs].

[Liu et al., 2021b] Liu, Y., Hashimoto, K., Zhou, Y., Yavuz, S., Xiong, C., and Yu, P. S. (2021b). Dense Hierarchical Retrieval for Open-Domain Question Answering. arXiv:2110.15439 [cs].

[Liu et al., 2019] Liu, Y., Ott, M., Goyal, N., Du, J., Joshi, M., Chen, D., Levy, O., Lewis, M., Zettlemoyer, L., and Stoyanov, V. (2019). RoBERTa: A Robustly Optimized BERT Pretraining Approach.

[Liu et al., 2023] Liu, Z., Oguz, B., Zhao, C., Chang, E., Stock, P., Mehdad, Y., Shi, Y., Krishnamoorthi, R., and Chandra, V. (2023). LLM-QAT: Data-Free Quantization Aware Training for Large Language Models. arXiv:2305.17888 [cs].

[Liusie et al., 2022] Liusie, A., Qian, M., Li, X., and Gales, M. (2022). UNIVERSITY OF CAMBRIDGE AT TREC CAST 2022.

[Luo et al., 2022] 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:2203.07522 [cs].

[Ma et al., 2023] Ma, X., Fang, G., and Wang, X. (2023). LLM-Pruner: On the Structural Pruning of Large Language Models. arXiv:2305.11627 [cs].

[Mao et al., 2023] Mao, K., Dou, Z., Chen, H., Mo, F., and Qian, H. (2023). Large Language Models Know Your Contextual Search Intent: A Prompting Framework for Conversational Search. Publisher: arXiv Version Number: 1.

[Mathew et al., 2021] Mathew, M., Tito, R., Karatzas, D., Manmatha, R., and Jawahar, C. V. (2021). Document Visual Question Answering Challenge 2020. arXiv:2008.08899 [cs].

[McDonald et al., 2022] McDonald, T., Tsan, B., Saini, A., Ordonez, J., Gutierrez, L., Nguyen, P., Mason, B., and Ng, B. (2022). Detect, Retrieve, Comprehend: A Flexible Framework for Zero-Shot Document-Level Question Answering. arXiv:2210.01959 [cs].

[Meuschke et al., 2023] Meuschke, N., Jagdale, A., Spinde, T., Mitrović, J., and Gipp, B. (2023). A Benchmark of PDF Information Extraction Tools using a Multi-Task and Multi-Domain Evaluation Framework for Academic Documents. volume 13972, pages 383–405. arXiv:2303.09957 [cs].

[Mishra and Jain, 2016] Mishra, A. and Jain, S. K. (2016). A survey on question answering systems with classification. *Journal of King Saud University - Computer and Information Sciences*, 28(3):345–361.

[Muennighoff, 2022] Muennighoff, N. (2022). SGPT: GPT Sentence Embeddings for Semantic Search. arXiv:2202.08904 [cs].

[Nassiri and Akhloufi, 2023] Nassiri, K. and Akhloufi, M. (2023). Transformer models used for text-based question answering systems. *Applied Intelligence*, 53(9):10602–10635.

[Neelakantan et al., 2022] Neelakantan, A., Xu, T., Puri, R., Radford, A., Han, J. M., Tworek, J., Yuan, Q., Tezak, N., Kim, J. W., Hallacy, C., Heidecke, J., Shyam, P., Power, B., Nekoul, T. E., Sastry, G., Krueger, G., Schnurr, D., Such, F. P., Hsu, K., Thompson, M., Khan, T., Sherbakov, T., Jang, J., Welinder, P., and Weng, L. (2022). Text and Code Embeddings by Contrastive Pre-Training. arXiv:2201.10005 [cs].

[Ni et al., 2021] Ni, J., Qu, C., Lu, J., Dai, Z., Ábrego, G. H., Ma, J., Zhao, V. Y., Luan, Y., Hall, K. B., Chang, M.-W., and Yang, Y. (2021). Large Dual Encoders Are Generalizable Retrievers. arXiv:2112.07899 [cs].

[Nishida et al., 2018] Nishida, K., Saito, I., Otsuka, A., Asano, H., and Tomita, J. (2018). Retrieve-and-Read: Multi-task Learning of Information Retrieval and Reading Comprehension. In *Proceedings of the 27th ACM International Conference on Information and Knowledge Management*, pages 647–656. arXiv:1808.10628 [cs].

[OpenAI, 2023] OpenAI (2023). ChatGPT Retrieval Plugin. original-date: 2023-03-23T06:06:22Z.

[Ouyang et al., 2022] Ouyang, L., Wu, J., Jiang, X., Almeida, D., Wainwright, C. L., Mishkin, P., Zhang, C., Agarwal, S., Slama, K., Ray, A., Schulman, J., Hilton, J., Kelton, F., Miller, L., Simens, M., Askell, A., Welinder, P., Christiano, P., Leike, J., and Lowe, R. (2022). Training language models to follow instructions with human feedback. arXiv:2203.02155 [cs].

[Owoicho et al., 2022] Owoicho, P., Dalton, J., Aliannejadi, M., Azzopardi, L., Trippas, J. R., and Vakulenko, S. (2022). TREC CAsT 2022: Going Beyond User Ask and System Retrieve with Initiative and Response Generation.

[Papineni et al., 2002] Papineni, K., Roukos, S., Ward, T., and Zhu, W.-J. (2002). Bleu: a Method for Automatic Evaluation of Machine Translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pages 311–318, Philadelphia, Pennsylvania, USA. Association for Computational Linguistics.

[Pereira et al., 2022] Pereira, J., Fidalgo, R., Lotufo, R., and Nogueira, R. (2022). Visconde: Multi-document QA with GPT-3 and Neural Reranking.

[Plüster, 2023a] Plüster, B. (2023a). LeoLM: Ein Impuls für Deutschsprachige LLM-Forschung | LAION.

[Plüster, 2023b] Plüster, B. (2023b). LeoLM: Ein Impuls für Deutschsprachige LLM-Forschung | LAION.

[PyPDF2,] PyPDF2. Welcome to PyPDF2 — PyPDF2 documentation.

[Raffel et al., 2023] Raffel, C., Shazeer, N., Roberts, A., Lee, K., Narang, S., Matena, M., Zhou, Y., Li, W., and Liu, P. J. (2023). Exploring the Limits of Transfer Learning with a Unified Text-to-Text Transformer. arXiv:1910.10683 [cs, stat].

[Rajpurkar et al., 2016] Rajpurkar, P., Zhang, J., Lopyrev, K., and Liang, P. (2016). SQuAD: 100,000+ Questions for Machine Comprehension of Text. arXiv:1606.05250 [cs].

[Rastogi et al., 2020] Rastogi, A., Zang, X., Sunkara, S., Gupta, R., and Khaitan, P. (2020). Schema-Guided Dialogue State Tracking Task at DSTC8.

[Reddy et al., 2022] Reddy, R. G., Iyer, B., Sultan, M. A., Zhang, R., Sil, A., Castelli, V., Florian, R., and Roukos, S. (2022). Synthetic Target Domain Supervision for Open Retrieval QA. arXiv:2204.09248 [cs].

[Reddy et al., 2018] Reddy, S., Chen, D., and Manning, C. D. (2018). CoQA: A Conversational Question Answering Challenge.

[Roberts et al., 2020] Roberts, A., Raffel, C., and Shazeer, N. (2020). How Much Knowledge Can You Pack Into the Parameters of a Language Model? In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 5418–5426, Online. Association for Computational Linguistics.

[Robertson and Zaragoza, 2009] Robertson, S. and Zaragoza, H. (2009). The Probabilistic Relevance Framework: BM25 and Beyond. *Foundations and Trends in Information Retrieval*, 3:333–389.

[Sachan et al., 2023] Sachan, D. S., Lewis, M., Yogatama, D., Zettlemoyer, L., Pineau, J., and Zaheer, M. (2023). Questions Are All You Need to Train a Dense Passage Retriever. arXiv:2206.10658 [cs].

[Serban et al., 2016] Serban, I. V., García-Durán, A., Gulcehre, C., Ahn, S., Chandar, S., Courville, A., and Bengio, Y. (2016). Generating Factoid Questions With Recurrent Neural Networks: The 30M Factoid Question-Answer Corpus. In *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 588–598, Berlin, Germany. Association for Computational Linguistics.

[Thakur et al., 2021] Thakur, N., Reimers, N., Rücklé, A., Srivastava, A., and Gurevych, I. (2021). BEIR: A Heterogenous Benchmark for Zero-shot Evaluation of Information Retrieval Models. arXiv:2104.08663 [cs].

[Tito et al., 2021] Tito, R., Karatzas, D., and Valveny, E. (2021). Document Collection Visual Question Answering. volume 12822, pages 778–792. arXiv:2104.14336 [cs].

[Touvron et al., 2023] Touvron, H., Martin, L., Stone, K., Albert, P., Almahairi, A., Babaei, Y., Bashlykov, N., Batra, S., Bhargava, P., Bhosale, S., Bikel, D., Blecher, L., Ferrer, C. C., Chen, M., Cucurull, G., Esiobu, D., Fernandes, J., Fu, J., Fu, W., Fuller, B., Gao, C., Goswami, V., Goyal, N., Hartshorn, A., Hosseini, S., Hou, R., Inan, H., Kardas, M., Kerkez, V., Khabsa, M., Kloumann, I., Korenev, A., Koura, P. S., Lachaux, M.-A., Lavril, T., Lee, J., Liskovich, D., Lu, Y., Mao, Y., Martinet, X., Mihaylov, T., Mishra, P., Molybog, I., Nie, Y., Poulton, A., Reizenstein, J., Rungta, R., Saladi, K., Schelten, A., Silva, R., Smith, E. M., Subramanian, R., Tan, X. E., Tang, B., Taylor, R., Williams, A., Kuan, J. X., Xu, P., Yan, Z., Zarov, I., Zhang, Y., Fan, A., Kambadur, M., Narang, S., Rodriguez, A., Stojnic, R., Edunov, S., and Scialom, T. (2023). Llama 2: Open Foundation and Fine-Tuned Chat Models. arXiv:2307.09288 [cs].

[Treviso et al., 2023] Treviso, M., Lee, J.-U., Ji, T., Aken, B. V., Cao, Q., Ciosici, M. R., Hassid, M., Heafield, K., Hooker, S., Raffel, C., Martins, P. H., Martins, A. F. T., Forde, J. Z., Milder, P., Simpson, E., Slonim, N., Dodge, J., Strubell, E., Balasubramanian, N., Derczynski, L., Gurevych, I., and Schwartz, R. (2023). Efficient

Methods for Natural Language Processing: A Survey. *Transactions of the Association for Computational Linguistics*, 11:826–860.

[Voorhees, 1999] Voorhees, E. (1999). The TREC-8 Question Answering Track Report.

[Voskarides et al., 2020] Voskarides, N., Li, D., Ren, P., Kanoulas, E., and de Rijke, M. (2020). Query Resolution for Conversational Search with Limited Supervision. In *Proceedings of the 43rd International ACM SIGIR Conference on Research and Development in Information Retrieval*, pages 921–930. arXiv:2005.11723 [cs].

[Wang et al., 2020] Wang, W., Wei, F., Dong, L., Bao, H., Yang, N., and Zhou, M. (2020). MiniLM: Deep Self-Attention Distillation for Task-Agnostic Compression of Pre-Trained Transformers. arXiv:2002.10957 [cs].

[Wang, 2022] Wang, Z. (2022). Modern Question Answering Datasets and Benchmarks: A Survey. Publisher: arXiv Version Number: 1.

[Wang et al., 2019] Wang, Z., Ng, P., Ma, X., Nallapati, R., and Xiang, B. (2019). Multi-passage BERT: A Globally Normalized BERT Model for Open-domain Question Answering. arXiv:1908.08167 [cs].

[Wei et al., 2023] Wei, J., Wang, X., Schuurmans, D., Bosma, M., Ichter, B., Xia, F., Chi, E., Le, Q., and Zhou, D. (2023). Chain-of-Thought Prompting Elicits Reasoning in Large Language Models. arXiv:2201.11903 [cs].

[White et al., 2023] White, J., Fu, Q., Hays, S., Sandborn, M., Olea, C., Gilbert, H., Elnashar, A., Spencer-Smith, J., and Schmidt, D. C. (2023). A Prompt Pattern Catalog to Enhance Prompt Engineering with ChatGPT. arXiv:2302.11382 [cs].

[William, 2023] William (2023). AutoGPTQ. original-date: 2023-04-13T02:18:11Z.

[Yang et al., 2019] Yang, J.-H., Lin, S.-C., Lin, J., Tsai, M.-F., and Wang, C.-J. (2019). Query and Answer Expansion from Conversation History.

[Yang et al., 2018] 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. arXiv:1809.09600 [cs].

[Zaib et al., 2021] Zaib, M., Zhang, W. E., Sheng, Q. Z., Mahmood, A., and Zhang, Y. (2021). Conversational Question Answering: A Survey.

[Zamani et al., 2023] Zamani, H., Trippas, J. R., Dalton, J., and Radlinski, F. (2023). Conversational Information Seeking. arXiv:2201.08808 [cs].

[Zhang et al., 2023a] Zhang, J., Zhang, H., Zhang, D., Liu, Y., and Huang, S. (2023a). Beam Retrieval: General End-to-End Retrieval for Multi-Hop Question Answering. arXiv:2308.08973 [cs].

[Zhang et al., 2023b] Zhang, Q., Chen, S., Xu, D., Cao, Q., Chen, X., Cohn, T., and Fang, M. (2023b). A Survey for Efficient Open Domain Question Answering. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 14447–14465, Toronto, Canada. Association for Computational Linguistics.

[Zhang et al., 2020] Zhang, T., Kishore, V., Wu, F., Weinberger, K. Q., and Artzi, Y. (2020). BERTScore: Evaluating Text Generation with BERT. arXiv:1904.09675 [cs].

[Zhao et al., 2023] Zhao, W. X., Zhou, K., Li, J., Tang, T., Wang, X., Hou, Y., Min, Y., Zhang, B., Zhang, J., Dong, Z., Du, Y., Yang, C., Chen, Y., Chen, Z., Jiang, J., Ren, R., Li, Y., Tang, X., Liu, Z., Liu, P., Nie, J.-Y., and Wen, J.-R. (2023). A Survey of Large Language Models. arXiv:2303.18223 [cs].

[Zhu et al., 2021] Zhu, F., Lei, W., Wang, C., Zheng, J., Poria, S., and Chua, T.-S. (2021). Retrieving and Reading: A Comprehensive Survey on Open-domain Question Answering. arXiv:2101.00774 [cs].

[Zhu et al., 2023] Zhu, X., Li, J., Liu, Y., Ma, C., and Wang, W. (2023). A Survey on Model Compression for Large Language Models.

[Zhu et al., 2022] Zhu, Y., Liu, N., Xu, Z., Liu, X., Meng, W., and Wang, Y. (2022). Teach Less, Learn More: On the Undistillable Classes in Knowledge Distillation.