UniversitÃ¤t Heidelberg
Institut fÃ¼r 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:	September 24, 2023

Ich versichere, dass ich diese Master-Arbeit selbststÃ¤ndig verfasst und nur die angegebenen Quellen und Hilfsmittel verwendet habe und die GrundsÃ¤tze und Empfehlungen “Verantwortung in der Wissenschaft” der UniversitÃ¤t Heidelberg beachtet wurden.

Abgabedatum: September 24, 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 bzw. Ergebnisse übersichtlich und verständlich 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 Conversational Question Answering (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 2.3 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 [4].

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" [5]. A more well-known QA system is *Watson* by IBM, an open-domain QA system that won a the TV show Jeopardy! in 2011 [6]. 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" [5]. 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 [6]. 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" [3]. 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. [7, 1] 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 [2, 3, 8, 9] 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. [1] 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 [10], which was further elaborated upon by Etezadi et al. [8]. Also the Answer Types where adjusted to align with the classifications used in [11, 12]. In this context, a crucial distinction is made between a **QA** and **ConvQA** system, guided by the criteria outlined in [13]: 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 [1, 7]. 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 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

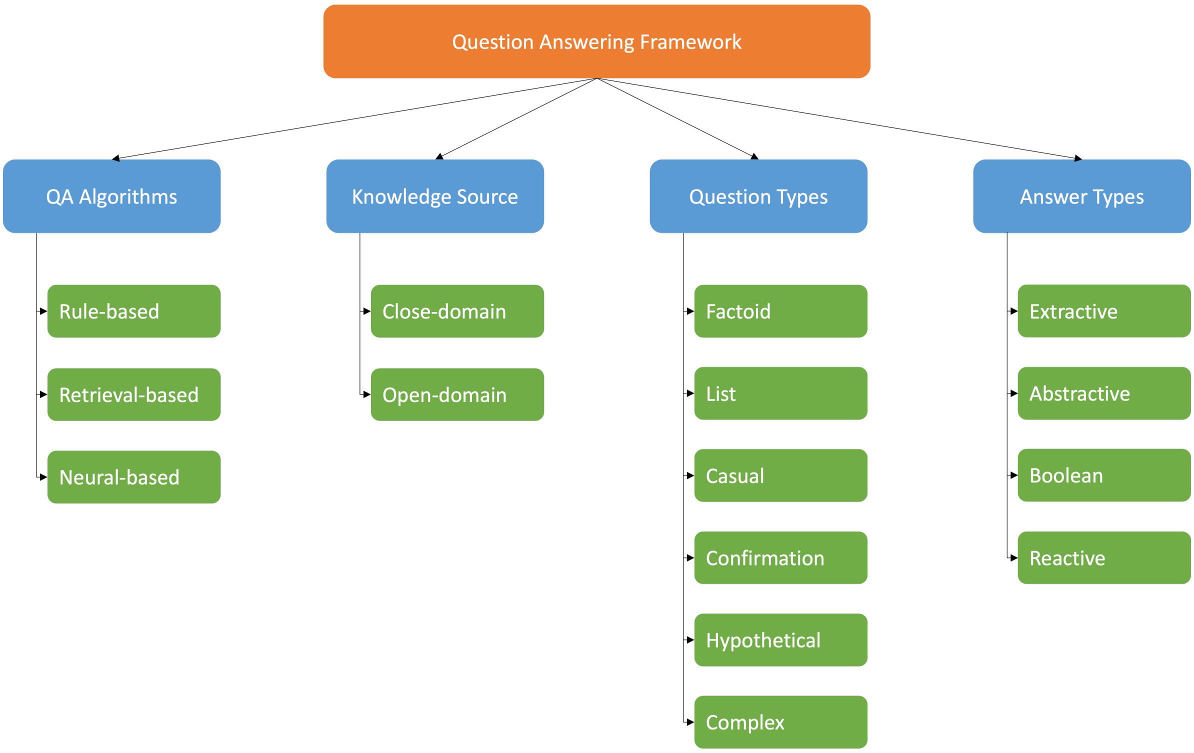

Figure 2.1: Adjusted QA Framework Classification by Farea et al. [1]

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 [8].

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 [1, 2].

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 [3]. 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 [14, 12, 2].

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 [1].

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 [7].

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 [2, 1, 3].

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* [10], or *complex* [8].

- *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. [1] 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 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 [15].

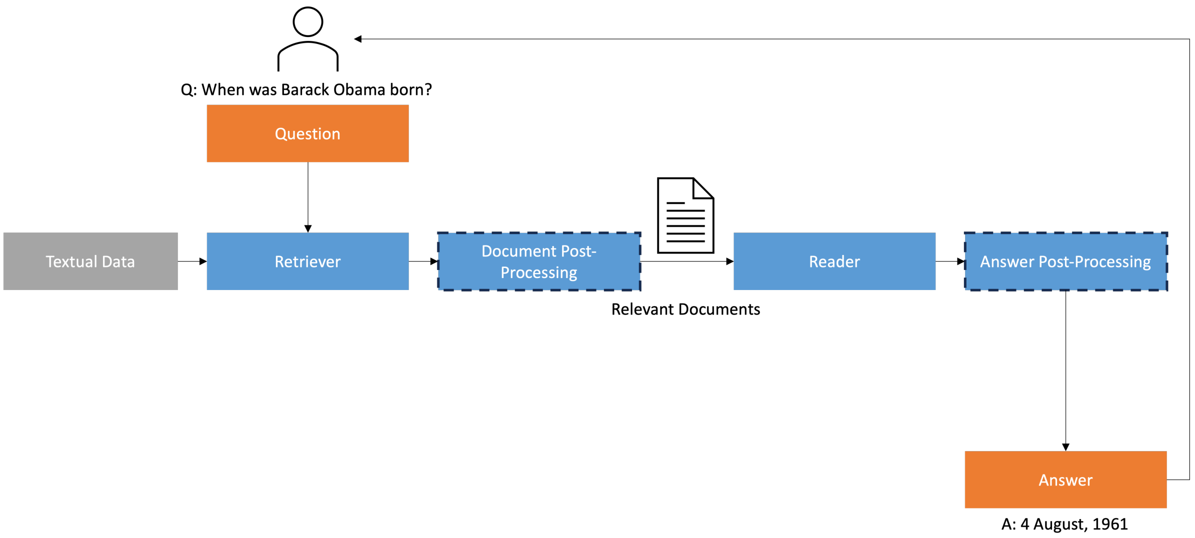

Figure 2.2: Reader-Retriever-System Architecture for QA by Zhu et. al. [2]. The dashed lines indicate optional modules.

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 [16].

Figure 2.2 depicts the general architecture of a **Retriever-Reader-System**, as defined by Zhu et al. [2]. This architecture serves as the foundational framework for IR-Based QA systems and was initially introduced by Harabagiu et al. [17]. 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 [2, 3].

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 [18]. However, for the purposes of this thesis, we adhere to Zhu et al.'s definition [2], 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 Retrieval-Augmented Generation (RAG) [19]. 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.

Another end-to-end approach, similar to RAG, is Retrieval-Augmented Language Model pre-training (REALM) [20]. 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 [21].

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 [22], 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* [23], *direct* [24], and *alternative* [12] 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) [23]. 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 [25].

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 [26]. 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 [11]. Examples of a *Document*

Layout Analysis model include the Document Image Transformer by Li et al. [27].

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* [28]. 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 (Large Language Model (LLM)) is to optionally cleanse the text corpus and then divide it based on a pre-defined token size. This approach is evident in two notable open-source LLM projects: *Langchain* and the *Retrieval Plugin for ChatGPT* by OpenAI [29, 30]. In the original Dense Retrieval paper by Karpukhin et al., a sliding window of token size 5 was utilized [31]. 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 [2].

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 [12]. 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" [2]. Nandan et al. even demonstrated that on modern Open-Domain Question Answering (ODQA) datasets, BM25 remains a viable baseline for zero-shot IR [32].

BM25 was originally introduced by Robertson et al. [33]. It operates by utilizing the TF-IDF token weights between a question q containing tokens $q_1, \dots, q_T$ and a set of passages P, where $p \in P$.

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

Equation 2.1 illustrates the BM25 score for a question q and a passage p. In this equation, $N\left(q_i, \mathcal{P}\right)$ represents the count of passages in $\mathcal{P}$ that contain the token q_i, while $n\left(q_i, p\right)$ 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 $b = 0.4$ [11, 33].

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 [31]. 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 [2].

In general, there are three types of Dense Retrieval approaches [2]: the **Representation-based Retriever**, often referred to as the *dual-encoder* [31]; 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** [34] 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 [31]. 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:

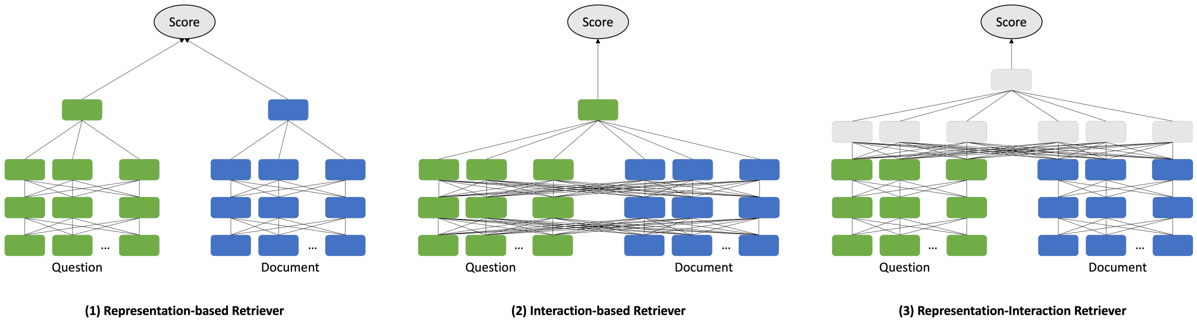
Score
Score
Score
Question
Document
Question
Document
Question
Document
(1) Representation-based Retriever
(2) Interaction-based Retriever
(3) Representation-Interaction Retriever

Figure 2.3: Types of Dense Retriever by Zhu et. al. [2].

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

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 [31], 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) [34]. While these models perform competitively with previous Representation-Based Retrievers, it's important to note that they are 100-1000 times more computationally expensive [35].

To address this latency issue, models like ColBERT introduced the concept of **co**ntextualized **l**ate ineraction [35]. In this thesis and subsequently in research, it is referred to as the **Representation-Interaction Retriever** [2].

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.

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

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

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^- [35].

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 [36], 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 [37]. 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 [38]. 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 [2]. 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 [2]. BEAM, for example, employs a fixed number of hops, specifically 2, as determined through empirical evaluation.

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 [2]. 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 [25].

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 [14] dataset, which was mentioned in Section 2.1.1. However, unlike the original MRC task, a Reader

[1] Status as of September 23, 2023, according to https://paperswithcode.com and the authors of [37]

in a Retrieval-Reader-System must process multiple passages to determine the relevant information needed to answer a given question [2].

Modern readers rely on Transformer-based Pre-trained Language Model (PrLM)s since they establish new baselines on well-known datasets [39]. In general, there are two types of Readers that use PrLMs: **Extractive Readers** and **Generative Readers** [3, 2, 39].

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 [34], DeBERTa [38], or RoBERTa [40]. Luo et al. [39] even utilized the encoder components of established encoder-decoder models like T5 [41] and BART [42]. 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 repeated for each segment [3, 39].

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:

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

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 [39, 2].

Latest research projects like Visconde [43] 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 Gener-

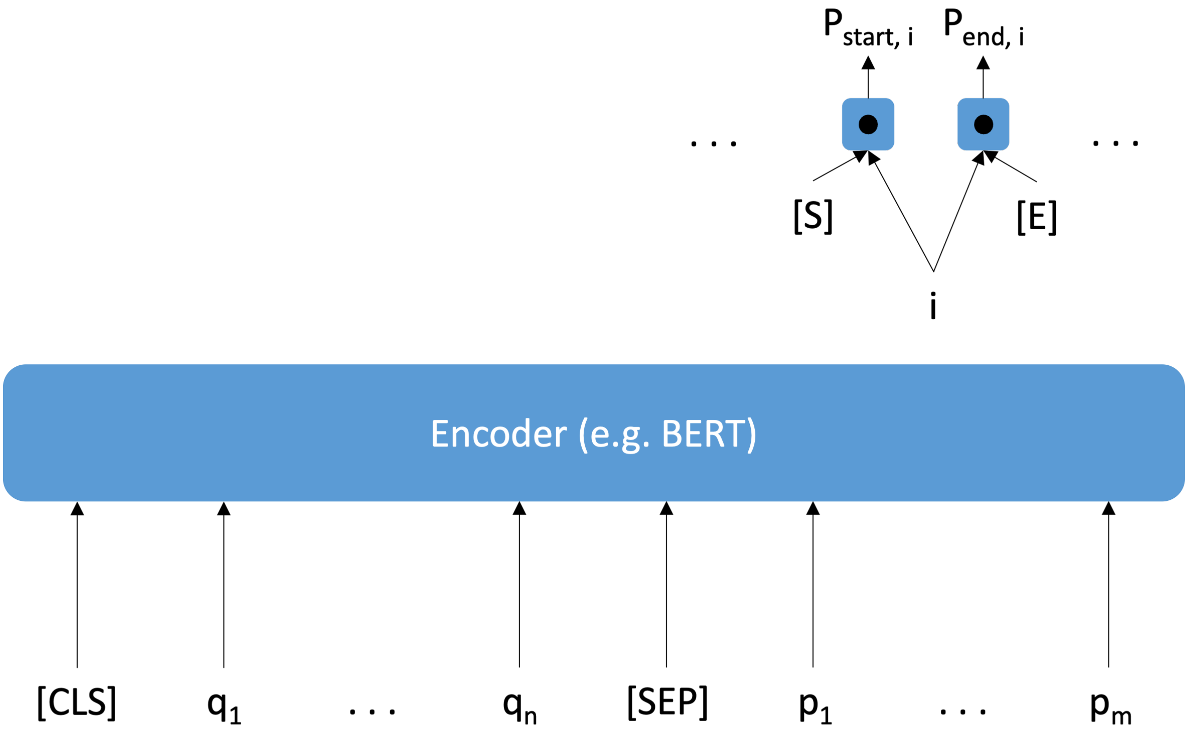

Figure 2.4: Adjusted Graphic of the Extractive Reader by Jurafsky et al. [3]

ative Readers [39]. They discovered that “on average, extractive readers perform better than generative ones” [39], except in cases involving long context passages, where generative approaches outperform the extractive ones.

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 [2]. 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 [9].

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% [32]. This highlights another crucial limitation of all NN-based retrievers and readers: training. These models heavily depend on their training data, while a Sparse Retriever like BM25 can be used without any training. According to experiments conducted by Thakur et al. [32], 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 task, as these datasets need to 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) [44]. 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 [9]. 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

For a detailed overview of approaches towards more efficient ODQA systems, please refer to the comprehensive survey by Zhang et al. [9].

[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

2.2 Conversational Question Answering

2.2.1 Basics

2.2.2 Query Expansion

2.2.3 Initiative

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

2.3 Related Work

2.3.1 Document-based Question Answering

2.3.2 Open-domain Conversational Question Answering ???

3 Open-domain QA Chatbot over PDFs

This chapter is the main achievment of the Thesis. It consits of laying out different possible solutions to the given problem.

4 Experimental Evaluation

This chapter is the evaluation of the proposed solution. It consists of laying out different possible solutions to the given problem.

5 Conclusions and Future Work

This chapter is the conclusion of the 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.

Bibliography

[1] Amer Farea, Zhen Yang, Kien Duong, Nadeesha Perera, and Frank Emmert-Streib. Evaluation of question answering systems: Complexity of judging a natural language.

[2] Fengbin Zhu, Wenqiang Lei, Chao Wang, Jianming Zheng, Soujanya Poria, and Tat-Seng Chua. Retrieving and reading: A comprehensive survey on open-domain question answering.

[3] Dan Jurafsky and James H. Martin. *Speech and Language Processing.* 3 edition.

[4] Bert F. Green, Alice K. Wolf, Carol Chomsky, and Kenneth Laughery. 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. Association for Computing Machinery.

[5] E. Voorhees. The TREC-8 question answering track report.

[6] D. A. Ferrucci. Introduction to this is watson. 56(3):1:1–1:15. Conference Name: IBM Journal of Research and Development.

[7] Tianyong Hao, Xinxin Li, Yulan He, Fu Lee Wang, and Yingying Qu. Recent progress in leveraging deep learning methods for question answering. 34(4):2765–2783.

[8] Romina Etezadi and Mehrnoush Shamsfard. The state of the art in open domain complex question answering: a survey. 53(4):4124–4144.

[9] Qin Zhang, Shangsi Chen, Dongkuan Xu, Qingqing Cao, Xiaojun Chen, Trevor Cohn, and Meng Fang. 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. Association for Computational Linguistics.

[10] Amit Mishra and Sanjay Kumar Jain. A survey on question answering systems with classification. 28(3):345–361.

[11] Tavish McDonald, Brian Tsan, Amar Saini, Juanita Ordonez, Luis Gutierrez, Phan Nguyen, Blake Mason, and Brenda Ng. Detect, retrieve, comprehend: A flexible framework for zero-shot document-level question answering.

[12] Pradeep Dasigi, Kyle Lo, Iz Beltagy, Arman Cohan, Noah A. Smith, and Matt Gardner. A dataset of information-seeking questions and answers anchored in research papers.

[13] Hamed Zamani, Johanne R. Trippas, Jeff Dalton, and Filip Radlinski. Conversational information seeking.

[14] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang. SQuAD: 100,000+ questions for machine comprehension of text.

[15] Eleftherios Dimitrakis, Konstantinos Sgontzos, and Yannis Tzitzikas. A survey on question answering systems over linked data and documents. 55(2):233–259.

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

[17] Sanda M. Harabagiu, Steven J. Maiorano, and Marius A. Pasca. Open-domain textual question answering techniques. 9(3):231–267. Publisher: Cambridge University Press.

[18] Khalid Nassiri and Moulay Akhloufi. Transformer models used for text-based question answering systems. 53(9):10602–10635.

[19] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich KÃ¼ttler, Mike Lewis, Wen-tau Yih, Tim RocktÃ¤schel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive NLP tasks.

[20] Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: Retrieval-augmented language model pre-training.

[21] Kyosuke Nishida, Itsumi Saito, Atsushi Otsuka, Hisako Asano, and Junji Tomita. 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.

[22] Zhen Wang. Modern question answering datasets and benchmarks: A survey. Publisher: arXiv Version Number: 1.

[23] RubÃ¨n Tito, Dimosthenis Karatzas, and Ernest Valveny. Document collection visual question answering. volume 12822, pages 778–792.

[24] Zhiguo Wang, Patrick Ng, Xiaofei Ma, Ramesh Nallapati, and Bing Xiang. Multi-passage BERT: A globally normalized BERT model for open-domain question answering.

[25] Ye Liu, Kazuma Hashimoto, Yingbo Zhou, Semih Yavuz, Caiming Xiong, and Philip S. Yu. Dense hierarchical retrieval for open-domain question answering.

[26] Minesh Mathew, Ruben Tito, Dimosthenis Karatzas, R. Manmatha, and C. V. Jawahar. Document visual question answering challenge 2020.

[27] Junlong Li, Yiheng Xu, Tengchao Lv, Lei Cui, Cha Zhang, and Furu Wei. DiT: Self-supervised pre-training for document image transformer.

[28] Norman Meuschke, Apurva Jagdale, Timo Spinde, Jelena Mitrovic, and Bela Gipp. A benchmark of PDF information extraction tools using a multi-task and multi-domain evaluation framework for academic documents. volume 13972, pages 383–405.

[29] langchain-ai/langchain: Building applications with LLMs through composability.

[30] ChatGPT retrieval plugin. original-date: 2023-03-23T06:06:22Z.

[31] Vladimir Karpukhin, Barlas Oguz, Sewon Min, Patrick Lewis, Ledell Wu, Sergey Edunov, Danqi Chen, and Wen-tau Yih. Dense passage retrieval for open-domain question answering.

[32] Nandan Thakur, Nils Reimers, Andreas RÃ¼cklÃ©, Abhishek Srivastava, and Iryna Gurevych. BEIR: A heterogenous benchmark for zero-shot evaluation of information retrieval models.

[33] Stephen Robertson and Hugo Zaragoza. The probabilistic relevance framework: BM25 and beyond. 3:333–389.

[34] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: Pre-training of deep bidirectional transformers for language understanding.

[35] Omar Khattab and Matei Zaharia. ColBERT: Efficient and effective passage search via contextualized late interaction over BERT.

[36] Zhilin Yang, Peng Qi, Saizheng Zhang, Yoshua Bengio, William W. Cohen, Ruslan Salakhutdinov, and Christopher D. Manning. HotpotQA: A dataset for diverse, explainable multi-hop question answering.

[37] Jiahao Zhang, Haiyang Zhang, Dongmei Zhang, Yong Liu, and Shen Huang. Beam retrieval: General end-to-end retrieval for multi-hop question answering.

[38] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen. DEBERTA: DECODING-ENHANCED BERT WITH DISENTANGLED ATTENTION.

[39] Man Luo, Kazuma Hashimoto, Semih Yavuz, Zhiwei Liu, Chitta Baral, and Yingbo Zhou. Choose your QA model wisely: A systematic study of generative and extractive readers for question answering.

[40] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. RoBERTa: A robustly optimized BERT pretraining approach.

[41] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer.

[42] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Ves Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension.

[43] Jayr Pereira, Robson Fidalgo, Roberto Lotufo, and Rodrigo Nogueira. Visconde: Multi-document QA with GPT-3 and neural reranking.

[44] Iulian Vlad Serban, Alberto GarcÃa-DurÃ¡n, Caglar Gulcehre, Sungjin Ahn, Sarath Chandar, Aaron Courville, and Yoshua Bengio. 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. Association for Computational Linguistics.