Universität Hamburg
DER FORSCHUNG | DER LEHRE | DER BILDUNG

FAKULTÄT
FÜR MATHEMATIK, INFORMATIK
UND NATURWISSENSCHAFTEN

Language Technology

MASTERTHESIS

Retrieval-Augmented Generation for Benchmark Datasets: Techniques, Challenges, and Applications

Klejda Alushi

Field of Study: Intelligent Adaptive Systems
Matriculation No.: 7594775
1st Examiner: Prof. Dr. Chris Biemann, Universität Hamburg
2nd Examiner: Dr. Martin Semmann, Universität Hamburg

Language Technology
Department of Informatics
Faculty of Mathematics, Informatics and Natural Sciences

Universität Hamburg
Hamburg, Germany

A thesis submitted for the degree of

Master of Science (M. Sc.)

Retrieval-Augmented Generation for Benchmark Datasets: Techniques, Challenges, and Applications

Masters's Thesis submitted by: Klejda Alushi

Date of Submission: 16.10.2025

Supervisor(s):
Jan Strich, Universität Hamburg

Committee:
1st Examiner: Prof. Dr. Chris Biemann, Universität Hamburg
2nd Examiner: Dr. Martin Semmann, Universität Hamburg

Universität Hamburg, Hamburg, Germany
Faculty of Mathematics, Informatics and Natural Sciences
Department of Informatics

Language Technology

Affidavit

Hiermit versichere ich an Eides statt, dass ich die vorliegende Arbeit im Masterstudiengang Intelligent Adaptive Systems selbstständig verfasst und keine anderen als die angegebenen Hilfsmittel – insbesondere keine im Quellenverzeichnis nicht benannten Internet-Quellen – benutzt habe. Alle Stellen, die wörtlich oder sinngemäß aus Veröffentlichungen entnommen wurden, sind als solche kenntlich gemacht. Ich versichere weiterhin, dass ich die Arbeit vorher nicht in einem anderen Prüfungsverfahren eingereicht habe und die eingereichte schriftliche Fassung der auf dem elektronischen Speichermedium entspricht. Sofern im Zuge der Erstellung der vorliegenden Abschlussarbeit generative Künstliche Intelligenz (gKI) basierte elektronische Hilfsmittel verwendet wurden, versichere ich, dass meine eigene Leistung im Vordergrund stand und dass eine vollständige Dokumentation aller verwendeten Hilfsmittel gemäß der Guten Wissenschaftlichen Praxis vorliegt. Ich trage die Verantwortung für eventuell durch die gKI generierte fehlerhafte oder verzerrte Inhalte, fehlerhafte Referenzen, Verstöße gegen das Datenschutz- und Urheberrecht oder Plagiate."

I hereby declare and affirm that this thesis for the master's degree program Intelligent Adaptive Systems is my own work and that I have not used any other sources other than those indicated—especially Internet sources that are not specifically listed in the bibliography. All passages from publications which have been cited literally or summarized are marked accordingly. I further guarantee that I have not previously submitted this thesis in another examination procedure and the written version submitted corresponds to that on the electronic storage medium. If electronic resources based on generative artificial intelligence (gAI) were used in the course of writing this dissertation, I confirm that my own work was the main and value-adding contribution and that complete documentation of all resources used is available in accordance with good scientific practice. I am responsible for any erroneous or distorted content, incorrect references, violations of data protection and copyright law or plagiarism that may have been generated by the gAI.

Hamburg, 16.10.2025

Place, Date

Signature
(Klejda Alushi)

There are more ways to branch
than any cedar pencil will ever find.

A thing can travel everywhere,
just by holding still.

— Richard Powers *The Overstory*, 2018

Abstract

Retrieval-augmented generation (RAG) is an approach that aims to tackle the problem of LLMs having obsolete data or limited domain-specific knowledge. This is done by storing external data into vector databases, and introducing a retriever module which uses a nearest-neighbour approach to fetch the most relevant data for each user query, and sending them both along to a large language model (LLM) known as a generator. This approach, titled Naive RAG, still faced some challenges especially when faced with ambiguous queries or noisy data, which is why advanced RAG methods were proposed that enhance different areas of the RAG pipeline. This paper aims to discover whether advanced RAG methods perform better than Naive RAG when faced with multiple conversational, question-answer datasets. The research also looks to find what effect the differences between the datasets have on the retriever and generator performance.

The Reranker and Hybrid BM25 approaches were found to have an increased F1 performance regarding the answer quality, and HyDE was the leading approach regarding MRR. However, Base RAG was a close third in terms of F1, showing that it is still a worthwhile competitor, especially when its lower computing complexity is taken into account. Certain features of the datasets were also found to have an effect on the retriever quality, notably the total amount of contexts and the conversation lengths. The research concludes that Naive RAG is still a beneficial method and advanced RAG methods can improve the performance greatly although only when they are chosen for and tailored to the dataset in question.

Contents

List of Figures

List of Tables

1 Introduction

In recent years, the use of pre-trained Large Language Models (LLMs) has increased exceptionally with the introduction of state-of-the-art models (SOTA) such as the GPT [1], Llama [2], and PaLM [3] families. Although many LLMs first started as an exploration into natural language processing tasks such as text recognition and understanding, text generation, and information retrieval [4], they are now used in applications in various fields, not limited to software engineering [5], education [6], and medicine [7].

Despite the advancements mentioned above, there are still some issues that have arisen. One of those is that after the LLMs are trained, it becomes much more computationally and financially difficult to retrain them with up-to-date data. This means that for long periods of time, LLMs may be using inaccurate or obsolete data, thus reducing their reliability for factual information and increasing the chance of generating false or contradicting data [8]. Another problem that ties into this is the lack of domain-specific knowledge, which can pose problems when LLMs are used in highly context-sensitive fields such as law, medicine, or science. The problems can range from the generation of ambiguous or incorrect information to the overstatement of the significance of outliers or minor biases [9].

There are currently two main techniques that are used to combat the issues mentioned above, those being fine-tuning and Retrieval-Augmented Generation (RAG) algorithms [10]. With fine-tuning, the LLM is trained on new data, which is typically task-specific and narrower in scope and size. While this method has been shown to optimize LLMs for specialized tasks and has improved data efficiency by leveraging a pre-trained model, it is still quite resource inefficient, particularly when there is a constant stream of new data that needs to be learned, and in some cases can lead to overfitting, which occurs when the model becomes too specialized in the training data, and is not able to perform as well on the test data [11].

This is where the second method, RAG, comes in, which aims to use external data sources more quickly and efficiently. RAG uses various search and retrieval methods to scan an external database, collect the documents that best match the given

user query, and use the information within those documents to generate a response [12]. This core process of RAG, also known as *Naive RAG*, faces some significant challenges. Retrieval problems such as inadequate text segmentation [13] or unclear input queries may lead to imprecise or noisy retrieved contexts, which can distract LLMs and impede their performance [14]. Furthermore, generation issues can arise, in which the generated outcome might not be in the desired format or might include hallucinated information [15].

These problems have led to the development of many advanced RAG algorithms, which employ many pre- and post-retrieval enhancements, and modular RAG algorithms, which combine these enhancements, all the while transcending the traditional linear structure using loops, conditions, and branching [16]. This paper looks into these different techniques to determine whether they lead to an improvement in performance, and if so, to what extent and in which manner.

1.1 Research Questions

- **R1: How do advanced RAG methods impact answer quality in comparison to a baseline Naive RAG?**

  The advanced RAG methods that will be looked at, all add extra layers of complexity to the baseline RAG method, which can result in the need for more processing power, insufficient time constraints, and an increase in LLM prompts, which, when using proprietary models, lead to higher monetary costs. Therefore, to make some of these drawbacks worthwhile, it must be shown that these advanced methods achieve a significantly higher answer quality when compared to Naive RAG.

  The focus is on token-level answer quality instead of retrieval ability, since it provides a fair basis for comparing methods that enhance different aspects of the RAG pipeline. For evaluation, the F1 metric will be used, since that can provide a good, overall assessment as to the completeness and correctness of the generated answer. The retrieval ability of the methods will also be analyzed, but only to investigate if and how it affects the answer quality.

- **R2: How do dataset characteristics influence the performance and behavior of these RAG methods?**

  The experiment will focus on eight datasets, which focus on varying domains ranging from the more commonly used Common Crawl or Wikipedia based datasets to more specialized fields such as social welfare, travel or cooking. They also have different sizes, structures, and formats; therefore, a comparison could be done of not only the RAG methods against each other, but also of how their performance changes for each dataset. The retriever performance would need to compared across datasets with a larger total number of contexts, or lengthier contexts, which might be more likely to have distracting information. Since the conversation history is also provided during the prompting, it would also be

notable to examine whether having more QA turns is beneficial or disadvantageous for the generator performance.

For future experiments, this would allow for a more informed decision to be made regarding which approach best matches each dataset. While the F1 evaluation metric will be used to assess which method performs best, other metrics such as MRR and Recall@k will be used to further investigate which dataset characteristics affect which part of the RAG pipeline.

1.2 Thesis Structure

The structure of this paper is designed to provide an understanding of this research topic's importance. It also serves as an overview of recent research and developments, before leading to the creation and analysis of an experimental study.

The first chapter, **Introduction**, aims to provide an understanding of the motivation for the topic of this paper. It also briefly describes the more relevant terms, and delves further into what the main research questions that will be answered are.

The second chapter, **Literature Review**, serves as a comprehensive foundation behind the main concepts in the fields of natural language processing, transformers, large language models, and retrieval-augmented generation. It also documents how these fields evolved, how they connect, and what the most important advancements were.

The third chapter, **Related Work**, provides an overview of the recent research done into advanced RAG methods, discusses the format and style of the most commonly used datasets, and ends by discussing any gaps present in the current research.

The fourth chapter, **Methodology**, starts by introducing the benchmark datasets that will be used, including their structure and content creation, and discusses an important package that will be used to facilitate the running of the experiment. The chapter then continues by examining the chosen evaluation metrics along with the control and RAG methods that will be implemented.

The fifth chapter, **Experimental Study**, discusses how the implementation of the experiment was carried out and describes an analysis that was done to showcase the effect that added contexts can have on the answer quality. It then continues by first comparing the results of the RAG methods empirically and then having an in-depth analysis as to how the methods or datasets could have affected these results.

The sixth chapter, **Discussion**, analyzes whether the research questions were sufficiently answered by the experiment study, and discusses the limitations found and how they could be solved in future research. Lastly, it ends by going over the overall findings and what was learned from the thesis.

2 Literature Review

This chapter provides an overview of the history and development of the fields relevant to this thesis. It introduces early foundational theories and methodologies that have helped shape the progress and evolution of these fields. This then leads to detailing the major advances and key turning points, and how they were affected by not only technological advancements but also public attention.

Firstly, Section 2.1 starts by defining natural language processing (NLP), followed by a discussion of its early stages and progress, leading to the creation of transformers, detailed in Section 2.2. The chapter continues in Section 2.3 by outlining language models from their elementary forms to the more advanced current models, and ends in Section 2.4 by describing the main topic of this thesis, namely, retrieval-augmented generation (RAG).

2.1 Natural Language Processing (NLP)

NLP is an area of computational research that focuses on the analysis, understanding, and generation of natural language text or speech. The goal in NLP, as in many other fields of machine learning, is to achieve human-grade language processing capabilities, whether that is to improve human-machine communication, human-human communication, or simply to perform better analyses.

In the late 1940s and early 1950s, before the term NLP was even in use, research related to machine translation (MT) had already begun [17, 18]. A demonstration of this was the Russian to English *Georgetown-IBM experiment*, which, although a small experiment consisting of 250 words and six grammar rules, garnered widespread public attention [19]. With two studies of linguistic structure, Chomsky argued for both the use of finite-state Markov processes to represent grammar [20] and for the separation of syntax (sentence structures) and semantics (sentence meanings), which he claimed could exist independently from each other [21].

The research into MT and NLP continued with the creation of two major systems, SHRDLU [22] and LUNAR [23], both of which facilitated the communication between the user and the machine to be conducted in English. The two systems were both rule-based and combined semantic and syntactic analysis using augmented transition networks (ATNs) [24]. ATNs (similar to finite state machines) could generate deep structure representations during sentence structure analysis by using recursive parsing and without performing any reverse transformations [24], which were used to convert sentences from their surface structure to their deep structure [21]. The popularity of these systems only grew in the next decades [25–28], cementing the use of ATNs. During this time, the infamous report by the Automatic Language Processing Advisory Committee (ALPAC) [29, 30] and the British Lighthill report [31] were published, both of which held a pessimistic outlook over the future of AI, stating the high cost and lower-quality output of MT compared to manual human-translation and the discounting of combinatorial explosions for complex problems. Whether or not their skepticism was justified, the amount of funding for MT research was significantly reduced, heavily affecting up-and-coming research projects, leading to an AI winter. Nevertheless, AI research continued at a limited rate over the next decade, leading to developments that would renew governmental and public interest.

In the 1980s, with the progress of machine learning, NLP moved from being rule-based, which had its own set of limitations [32], towards more statistical techniques aided by the development of large text corpuses (Brown [33], Lancaster-Oslo-Bergen [34], CHILDES [35]). At the core of these techniques were estimators, which aimed to estimate the probability of a word occurring in a given sentence by looking at the words that had previously occurred [36]. Some estimators, such as N-gram models, use Markov assumptions to group text histories by assuming that the probability of a certain word depends only on the prior local context or the n words before it [37]. Developing alongside them were Hidden Markov Models (HMM), double stochastic processes with an unobservable Markov process, which became the basis for part-of-speech tagging. HMMs are composed of the following components [38]:

- Set of states $S = \{S_1, S_2, ...S_N\}$ = all the hidden states in the model
- State transition probability $A = a_{i_j}$ = the probability of transition from one state to another

$$a_{i_j} = P[q_{t+1} = S_i | q_t = S_j] \quad 1 \leq i, j \leq N$$

- Set of observations per state $O = \{o_1, o_2, ..., o_M\}$ = visible observations
- Probability distribution $B = b_j(k)$ = probability of a certain observation being generated from state s_i

$$b_j(k) = P[v_k att | q_t = S_j] \quad 1 \geq j \leq N 1 \geq k \leq M$$

- Initial state distribution $\Pi = \pi_i$ = the probability that a sequence starts in state s_i

$$\pi_i = P[q_i = S_i] \quad 1 \geq i \leq N$$

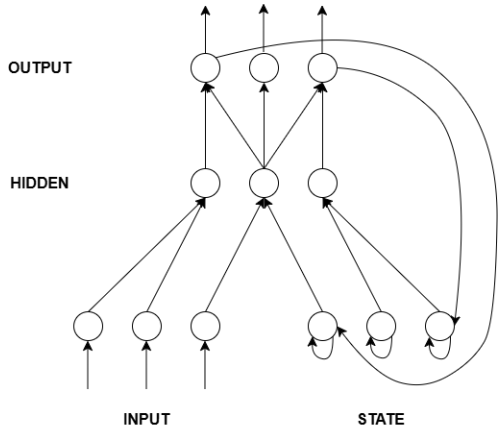


Figure 2.1: A diagram of the architecture of a recurrent neural network recreated from Elman [44]. The input and the state are processed together to produce a new hidden state. In this variant, the output is also used to compute the previous hidden state, thus allowing the network to include information about its past predictions.

Despite their widespread use, HMMs did suffer from some shortcomings, such as failing to take into account contextual information or long-range dependencies and poor discrimination due to their training algorithms, which optimize more to maximize likelihoods than to classification. This is where artificial neural networks (ANN) came into play. Inspired by the architecture of biological neural networks that are made up of a multitude of interconnected neurons, each of which performs a task upon receiving an input signal, ANNs are also made up of artificial neurons organized into layers [39].

Some of the earlier neural networks were feedforward in which information flows from the input layer to the hidden layer(s) to the output. These were first used in NLP by Bengio *et al.* [40] who proposed to tackle the "curse of dimensionality" that results from word sequences during testing being different from sequences the model has seen before, by learning a distributed representation of words. Convolutional neural networks (CNN), which employ filter optimization and recurrent neural networks (RNN), which allow information to move forward or backward through the layers, and make use of a recurrent unit, a sort of memory that contains knowledge of the previous state and the current inputs, as shown in Figure 2.1, also found widespread use in many NLP tasks [41–43].

Although RNNs allow better handling of sequential data [44], they also suffer from the vanishing gradient problem, a process in which, as error signals exponentially decrease as they back propagate through the layers, slowing or completely stopping the training process [45]. To combat this, a new method was introduced called Long Short-Term Memory (LSTM), which has a memory cell containing a linear unit termed the constant error carousel, which aims to disallow the decay of error signals. Connected to the memory cell are two multiplicative gates: an *input* gate which decides what information can enter the memory, and an *output* gate which decides what information

can leave, essentially what information will be used for the next output [46]. Gers *et al.* [47] noticed that this architecture could lead to indefinite growth of the memory cell and therefore introduced a *forget* gate, as shown in Figure 2.2, which would learn to reset these cells, thus eliminating useless or outdated information. This model was further advanced with the presentation of a bidirectional LSTM [48], thus when a point in a sequence is being processed, the model can know the points before and after it.

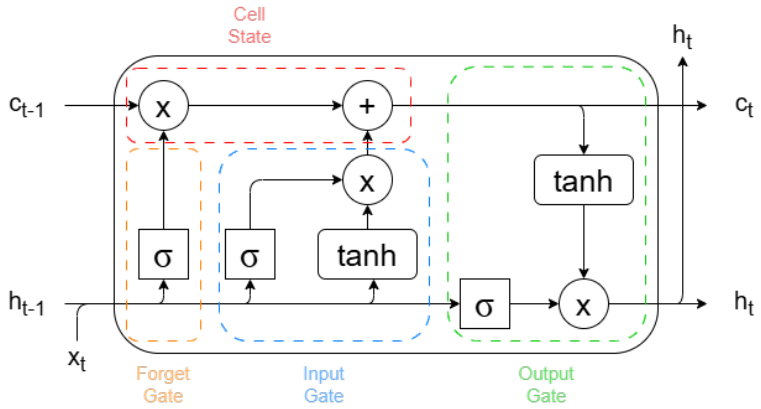


Figure 2.2: A diagram of the architecture of a long short-term memory recreated from Greff *et al.* [49]. The cell state is the memory cell that stores internal information, which can be controlled by the input and output gates. A forget gate is also used to reset data from the cell state.

Returning to NLP, during this evolution of neural networks, many substantial changes were also happening in word representations, such as the *word2vec* proposed by Mikolov *et al.* [50] in 2013. Before this, sparse representation was the main method used, one example being one-hot encoding, in which each word is assigned a vector the size of the text, with all the elements being 0 apart from the one that indicates the position of the word. The proposed models were Continuous Bag-of-Words (CBOW), which focuses on predicting a word based on its past and future context, and Continuous Skip-gram, which focuses on the inverse, predicting the context based on a certain word. These models used neural networks to create continuous vector representations of words, thus better capturing their semantic similarities [50]. Another significant change came with the Sequence-to-Sequence (*seq2seq*) approach, which uses two LSTMs, one to encode the input sequence to a vector representation and one to decode the vector back into a text sequence. Although this did have success mapping sequences to each other, the fixed lengths of the vectors would create a "bottleneck" problem, not allowing long sequences to be processed correctly [51].

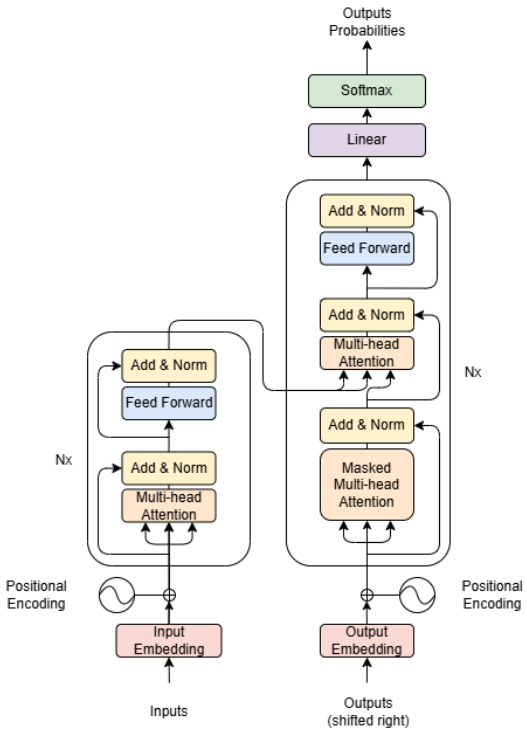


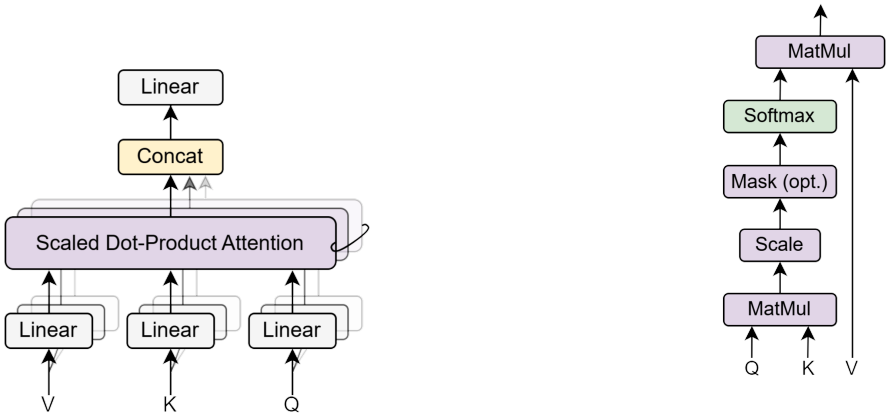
Figure 2.3: A diagram of the architecture of a transformer [52]. Transformers consist of an encoder (*left*) and decoder (*right*), which in turn use feedforward networks and multiple attention mechanisms to process input data.

2.2 Transformers

Transformers marked the next key turning point, proposed in the 2017 paper "Attention Is All You Need" [52]. LSTMs and RNNs, which processed sequences sequentially, one at a time, were on the way out, with transformers introducing a new way to process sequences all at once. They would also do away entirely with recurrence and convolution, instead replacing them with various self-attention layers, which proved to require less computational complexity, sequential operations, and would develop the learning of long-range dependencies, by having shorter distances between any points in the input or output sequence [52]. Transformers are made up of two major parts, an encoder and a decoder, as shown on the left and right, respectively, in Figure 2.3.

Encoder

The encoder is made up of N=6 layers, in the original implementation, which themselves are composed of two sublayers: a multi-head self-attention process and a feedforward network. The input first goes through an embedding process, where the sequence is first tokenized and then subsequently turned into a vector representation of those tokens. In NLP, tokenization is the process in which a character sequence is fragmented into so-called tokens, units of text that could be full words, characters, or word segments [53]. Tokenization for English texts started with the tokens being the words themselves, with



(a) Multi-head Attention (b) Dot-Product Attention

Figure 2.4: A visualization of the multi-head attention and dot-product attention present in transformers [52]. Multi-head attention is a series of self-attention mechanisms being performed in parallel, after which they are concatenated and linearly transformed. Dot-product attention first computes the attention weights of the keys and queries, which are then normalized, then used to weigh the value vectors.

white spaces and non-alphanumeric characters acting as delimiters, but subword based models have become more popular [54], creating a middle-ground between word-based and character-based tokens. Tokenization approaches require more complex techniques in specialized-domain texts or non-English languages with differing grammatical and morphological structures. Positional encodings are also appended to the input sequence, as a way of providing information about the token order and their relative position to each other. These encodings take the form of sinusoidal and cosinusoidal functions for even or odd indexes, respectively. This is shown

$$PE_{(pos,2i)} = \sin\left(pos \cdot 10000^{2i/d_{model}}\right), \quad PE_{(pos,2i+1)} = \cos\left(pos \cdot 10000^{2i/d_{model}}\right) \tag{2.1}$$

with pos being the word position, i the dimension, and d_{model} the model dimensions, in the case of this transformer equal to 512 [52].

The input sequence then reaches the encoder layer, where it will be mapped to a continuous representation that better grasps semantic context and uses attention to focus on the most relevant parts of the sequence. First, there is a multi-head self-attention module, which consists of multiple self-attention modules running in parallel, as shown in Figure 2.4a.

Self-attention is based on three vector representations: query (Q), key (K), and value (V), the dot products of which are computed according to Figure 2.4b. The dot products of the queries and keys are first calculated to obtain the attention weights,

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V \tag{2.2}$$

which are then divided by the square root of the keys' dimensions, and normalized through a softmax function.

The multi-head attention splits the Q, K, and V vectors into N representations, computes the self attention for each subset and concatenates them before passing them through a linear transformation in

$$\begin{aligned} MultiHead(Q,K,V) &= Concat(head_1,...,head_h)W^O \\ \text{where } head_i &= Attention(QW_i^Q, KW_i^K, VW_i^V) \end{aligned} \tag{2.3}$$

with the parameter matrices W_i and W^O. Through this attention mechanism, diverse contextual relationships can be captured, thus enriching the models understanding of the input sequence, all the while using less computational resources as a result of its parallel processing.

The multi-head output is entered into a fully-connected feedforward network, consisting of a ReLu activation sandwiched between two linear transformations. The attention and feedforward networks are both followed by a residual connection '*Add*' [55] and a normalization layer '*Norm*' [56], to allow better gradient flow and stabilization.

Decoder

The decoder is similar to the encoder in that it consists of $N = 6$ layers, and starts with the embedding and positional encoding of the output sequence, followed by multi-head attention layers and a feedforward network, with residual connections and layer normalization. However, there are some differences between the two, such as the encoder shifting the output sequence by one position to the right, ensuring that the predictions for a token at index i are only based on tokens before index i. The first multi-head attention layer is masked, which means that it does not have insight into the next tokens in the sequence [57]. The output from this layer then goes into the second layer together with the output from the encoder. After the final layer, the output is passed along to a linear layer and a softmax function to compute probabilities.

2.2.1 Transformer based models

After the introduction of the first transformer model, many other models started using and modifying the architecture. The BERT (Bidirectional Encoder Representations from Transformers) [58] model seeked to improve the left to right processing of transformers, through bidirectional processing of the context by removing the masked attention model in the decoder. Replacing this, was a masking mechanism, through which, 15% of tokens in a sequence would be masked. During its creation, BERT was pre-trained on unlabeled data with the goal of masked language modeling and next-sentence prediction and fine-tuned on task-specific labeled data.

BERT was used as a jumping point for many subsequent models, such as:

- RoBERTa, which suggested that BERT was undertrained and proposed a modified training method, namely longer training times and longer sequences, dynamic masking, in which the masking pattern is changed for every input sequence, and the removal of next sentence prediction [59].

- DistilBERT, proposes a method to pre-train a smaller, general-domain language model, which can then be fine-tuned on domain-specific tasks [60]. Knowledge distillation [61] is used for transfer learning, in which a smaller '*student*' model learns to behave similarly to a larger '*teacher*' model.
- ELECTRA [62], which during its pre-training, trains a generator and a discriminator network. The generator first uses a similar masking mechanism as BERT to replace certain tokens and then learns to predict the original values of the masked tokens. On the other hand, the discriminator learns to classify which tokens are from the original sequence and which have been generated, thus creating a relationship similar to a generative adversarial network.
- DeBERTa [63] is based on both the BERT and RoBERTa models and introduces two new techniques. The content and position of the input sequence are encoded in two separate vectors, and a disentangled attention mechanism is used to compute all pairwise relationships between the two. An improved mask decoder is also used, which when decoding the masked words, takes into consideration not only the relative position of a token but also the absolute position.

Another key model that uses Transformers as a foundational architecture is the Generative Pre-trained Transformer (GPT), now also referred to as GPT-1. GPT-1 combines unsupervised pre-training, using large text corpora and a 12-layer, left-to-right, decoder-only transformer, and supervised fine-tuning on a variety of tasks: natural language inference, semantic similarity, text classification, and question answering [64]. In recent years, GPT models have continued to evolve and have created notable breakthroughs in NLP, as will be discussed in Section 2.3.

In 2019, Facebook AI presented a Bidirectional and Auto-Regressive Transformer (BART) [65], which combines the bidirectional encoder from BERT, and the auto-regressive decoder from GPT. BART is pre-trained as a denoising autoencoder, focusing on tracing the original version of corrupted documents. The input sequences are corrupted in various ways, ranging from token masking and deletion to sentence permutation, text infilling, and sentence permutation. This allows BART to perform well not only in discriminative tasks, but also in regards to text generation. The model was then fine-tuned on a variety of downstream tasks, such as sequence classification and generation, and machine translation.

The last transformer-based architecture that will be discussed is the Text-to-Text Transfer Transformer (T5) [66], which treats every NLP task simply as processing a text input and producing a text output. This means that a single model can perform a variety of natural language tasks just by providing a task-specific prefix such as *sst2* for sentiment analysis, *translate English to German* for translation, or *summarization* for text summaries. For pre-training, the public, web-scraped data source Common Crawl was cleaned up using a list of heuristics to create the Colossal Clean Crawled Corpus (C4). Based on the BERT masking mechanism, a new masking objective was created, in which consecutive masked tokens were replaced by a solitary sentinel token, with the model being trained to predict sequences of tokens, thereby better recognizing long-range dependencies.

The subsequent GPT model (GPT-2) demonstrates that natural language tasks can be learned even without supervision, when trained on a new dataset WebText [67]. WebText was created by scraping webpages that were curated or filtered by humans, first starting by using links on Reddit as a jumping-off point, resulting in a dataset compiled of text from 45 million links. Byte Pair Encoding (BPE) [68] is used as a language segmentation algorithm that acts as a middle ground between character and word splitting. BPE creates a grouping of frequent character pairings, increasing the vocabulary of 50, 257, allowing it to better understand unknown or misspelled words. The architecture is largely based on the original model, with the addition of layer normalization at the start of each of the sub-blocks and after the last attention block.

The introduction of the transformer was revolutionary in the world of NLP, with Islam *et al.* [69] noting that of all the transformer models that are applied in various fields, about 40% were involved in NLP tasks, ranging from text classification & segmentation, language translation, question answering, and text summarization. It has also led to the development of large language models.

2.3 Large Language Models

Prior to discussing the benchmark language models that have been launched in recent years, it is necessary to first establish what makes a large language model. LLMs mainly refer to transformer-based, pre-trained language models that are larger and demonstrate a better understanding of natural language. In Figure 2.5, the main LLMs can be seen, grouped by their foundational architecture styles [70], as will be discussed below.

Encoder-only: These architectures include only the *encoder* stack of transformers, including a bidirectional self-attention mechanism that allows it to process tokens from both the past and the future. During pre-training, the main objectives are masked language modeling and next sentence prediction, which allow the model to establish a good understanding of natural language, and as such tend to be used more for named entity recognition or text classification. The main models built with this architecture are BERT, RoBERTa, and DeBERTa, which are examined in Section 2.2.

Decoder-only: A type of architecture which only uses the *decoder* stack of transformers, used mainly for text generation tasks, such as text completion or summarization, due to these models being autoregressive. Some of the leading LLMs today belong to this group, such as the LLaMa [71] or the OpenAI GPT [1] model families. Prefix-tuning [72] can also be used with decoder-only models, in which a set of vectors relating to the task is processed before the input sequence. This allows the model to be capable of performing different NLP tasks, without necessarily needing to be fine-tuned separately.

Encoder-Decoder: Here, the full transformer architecture is used, the encoder being able to process the entire input sequence and the decoder only processing the previous tokens. This serves as a middle ground between the architectures mentioned above, since it is able to perform well both in language understanding and text generation, with the representative models being BART [65], T5 [66], or mT5 [73].

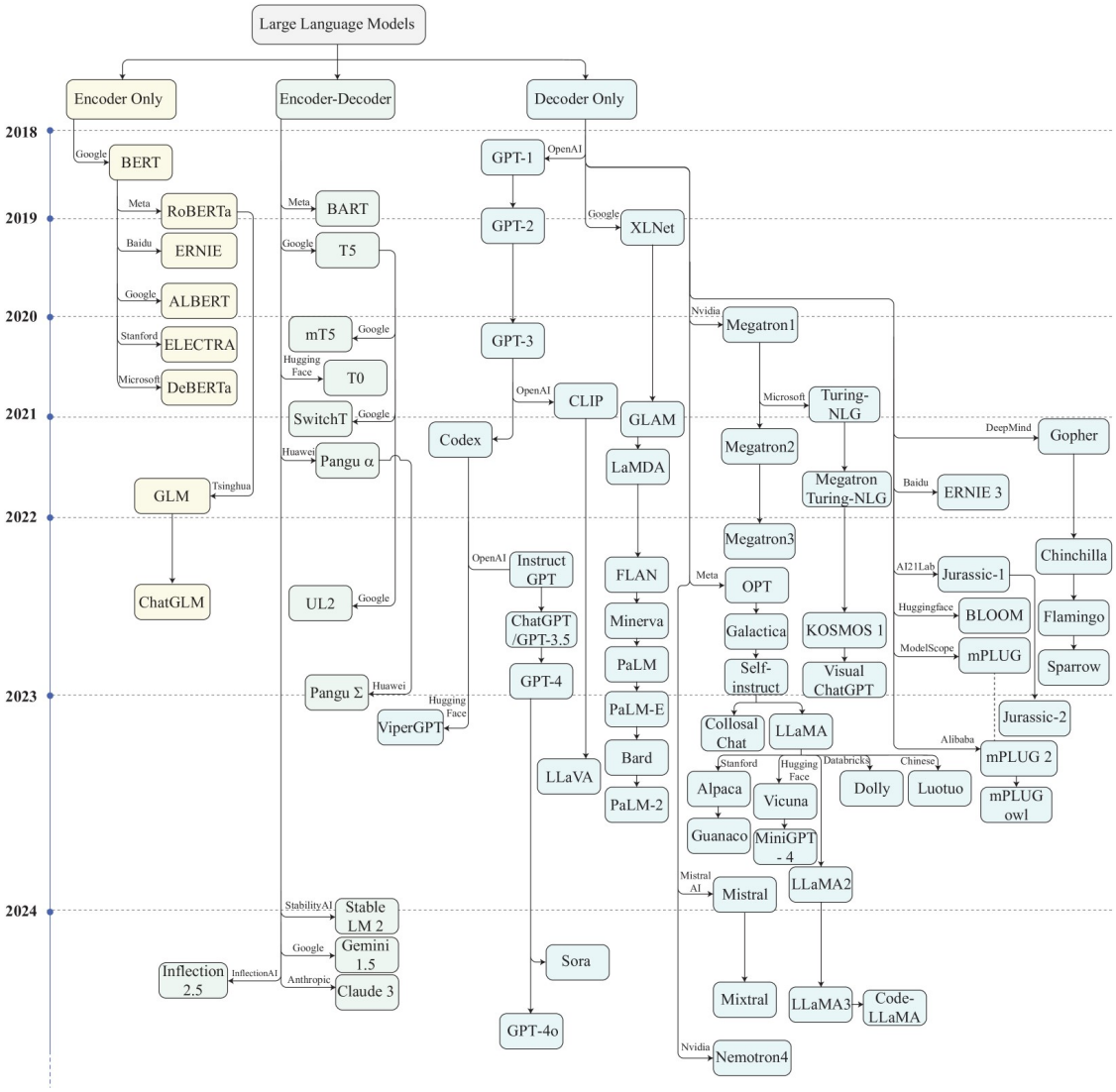


Figure 2.5: A diagram showing the evolution of large language models over time, grouped by their architecture type (Encoder Only, Encoder-Decoder, and Decoder Only) [74].

2.3.1 Model training

Modern language models go through two main learning processes known as pre-training and fine-tuning, which allows them to build a strong, general NLP foundation which can then be directed to more specialized tasks [75]:

Pre-training, where the goal is to learn and understand general language representation. In this stage, larger text corpuses are used, and all model weights have to be updated, making it much more computationally expensive and resource heavy. The main pre-training techniques include:

- Autoregressive Language Modeling - the model only has access to past tokens, which it uses to predict upcoming tokens.

- Masked Language Modeling - masks randomly sampled tokens or sequences of tokens, with the model predicting the masked tokens using past and future context.
- Unified Language Modeling - combines both autoregressive and masked language modeling, thus ensuring the model gains both language understanding and generation.
- Mixture of Experts - is a less resource-heavy but scalable pre-training method, in which a number of neural networks (*experts*) are trained to specialize in different NLP fields. A router is also trained to decide which experts should handle each input sequence, so that only a small amount of the available parameters are activated during processing.

Fine-tuning, occurs after pre-training and aims to specialize the model for certain, downstream tasks. In contrary to the aforementioned, since smaller, domain-specific corpuses are used and not all model parameters will be updated, this stage is shorter and less resource heavy.

- Instruction Tuning - also called supervised fine-tuning, involves training a model using an (*instruction*, *output*) dataset, improving the understanding of human instructions in a controlled environment. Some limitations include the difficulty in creating diverse, meaningful instruction/output pairs, and the model potentially learning only surface-level features of the output [4].
- Transfer Learning (TL) - is used to repurpose and adapt a model from its original task to a related but new task. Inductive TL uses a general-domain model and fine-tunes it on a related but more niche downstream task, thereby leveraging the knowledge acquired in the pre-training stage, whereas in transductive TL the original and new tasks are of the same nature but could exist in different domains or there could be a insufficient data for the target domain [76].
- Alignment Tuning - is a technique used to align the text generated by models to human values and standards. Shen *et al.* [77] suggests that it can be separated into outer alignment, focusing on correcting the harmlessness, honesty, and helpfulness of LLMs through human feedback, and inner alignment, which checks whether what the model is optimizing for matches the model objective. There are some limitations with both alignment method groups, such as the difficulty in obtaining human feedback and in analysing the complex decision making of the model.

2.3.2 Advancements in LLMs

GPT-3 is known as the first large language model, being much larger than its predecessor, with 175 billion parameters [78]. The model performed the best when using few-shot learning, which consists of giving the model K examples of context and desired

output, and achieving high but not optimal performance with one-shot (akin to few-shot with $K = 1$) and zero-shot (description of the task is given rather than examples). However, this model along with previous ones, were found have serious problems with creating discriminatory content, misleading or providing false information, or creating environments where user trust could be exploited [79]. Reasons for this could be due to the misalignment of objectives, one being the improvement of next word prediction, the other being the following human instructions or intent [80].

In order to place more importance on the latter objective, the InstructGPT model was created, which would fine-tune GPT-3 so that its outputs better align with those desired by the users. The fine-tuning process uses a combination of supervised learning and reinforcement learning with human feedback (RLHF), and is completed in three steps [81]:

- **Step 1**: Human labelers demonstrate desired outputs for randomly sampled prompts from the dataset. The outputs are then used to train a supervised policy.
- **Step 2**: Different outputs for the same prompt are collected from the model, and are then ranked from best to worst by labelers. These rankings are then used to train a reward model.
- **Step 3**: Reinforcement learning is used to improve the generated outputs by calculating a reward from the reward model. The rewards are then used to update the supervised policy.

On November 30th 2022, ChatGPT [1] was introduced to the public, using a GPT-3.5 model and fine-tuned using RLHF. It was created to behave in a conversational way, allowing an easier dialogue with users, and it marked a notable uptick in the public interest, seeing around 13 million unique, daily users [82] just months after launching. This popularity not only reignited interest in the field of AI, especially relating to generative models, but also led to the acceleration of progress in competing models. The next year OpenAI released GPT-4 [1], with an improved natural language generating ability, even in more nuanced and complex contexts, and multimodal processing, capable of understanding text and image inputs. This model still suffered from limitations that affected earlier models, albeit at a lesser amount, such as hallucinations, in which plausible but unreliable or incorrect information is created, and the generation of potentially harmful or offensive content.

The LLM family Claude [2] was introduced in early 2023, and was stated to be easier to converse with than ChatGPT and more effective in filtering out harmful information. This was done with the use of their Constitutional AI model [83], which is given a list of principles or a constitution taken from sources such as the UN Declaration of Human Rights [3], and goes through two training processes: a supervised learning stage, in which the model takes sample outputs from earlier models, which are then critiqued based on the constitution and then revised, and a reinforcement learning from AI feedback

[1] https://openai.com/index/chatgpt/
[2] https://www.anthropic.com/news/introducing-claude
[3] https://www.anthropic.com/news/claudes-constitution

stage (RLAIF), in which the model is fine-tuned with AI feedback in order to favor outputs that more align with the constitution.

Bard [4] was also launched by Google powered by their earlier model LaMDA [5]. The model was then rebranded as the multi-modal model Gemini, with the addition of audio and video processing, and text and image generation, and available in three models (Ultra, Pro, and Nano) each of which are tailored for different computational and complexity levels [84]. Meta also released their own LLM family under the name LLaMA (Large Language Model Meta AI) [2], the first model ranging from $7B - 65B$ parameters, with the later models reaching up to $405B$. The LLMs were made publicly accessible through a community license, along with an *Instruct* version of each model which were further fine-tuned to better follow instructions [71].

2.4 Retrieval-Augmented Generation

Retrieval-Augmented Generation (RAG) is a fine-tuning technique first introduced by Lewis *et al.* [10] in 2020 to enable models to continuously learn and update their information. Many language models are limited in their knowledge by the time when their training data was collected, also known as the cutoff date [85], meaning that any data created or updated after that date would not be accessible leading to the generation of inaccurate, outdated or *hallucinated* information.

2.4.1 RAG Process

As shown in Figure 2.6, RAG works by analyzing a submitted query and searching the given data sources to find the most relevant information, which is sent along with the query to the language model.

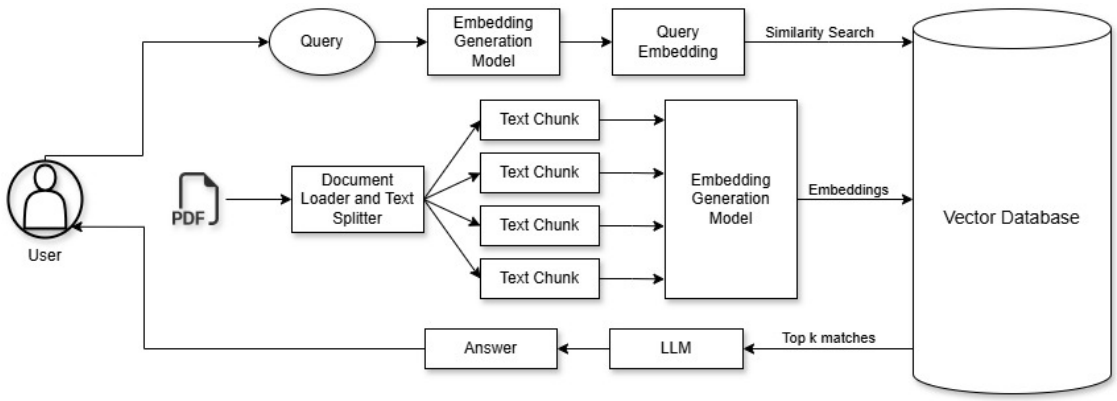


Figure 2.6: A diagram, inspired by Shukla [86], showing the workflow of a RAG process.

[4] https://blog.google/technology/ai/bard-google-ai-search-updates/
[5] https://blog.google/technology/ai/bard-google-ai-search-updates/

Indexing begins with the parsing and cleaning of the information from the given external sources, after which they are split into smaller *chunks*. These chunks are then also converted into vector representations by the retriever and stored in a vector database.

Retrieval occurs with the submission of a user query, which is then transformed into a vector representation by an embedding generation model or *retriever*. A similarity search is then performed between the data chunks and the query embedding, which determines the top N chunks that are the most semantically relevant to the query.

Generation is done with the use of a large language model, which is fed a prompt, combining the original query and the relevant chunks. The formulation of the prompt can then be altered, further optimizing it for the task at hand.

This core process is also known as *Naive RAG*, which has been shown to be effective at updating the model without any additional training required, and combining the parametric memory of an LLM with a non-parametric memory, i.e., vector database to provide more diverse and factual language [10]. On the other hand, some limitations also exist, especially in regards to the performance of the retriever, such as the fetching of irrelevant or noisy contexts, which can be distracting to LLMs [14], or contexts that are too long or too brief due to the chunk size. The structure and wording of the query could also be inadequate, leading to false or misleading retrievals, all of which leads to errors or hallucinations while generating. The solution for this was the creation of more advanced RAG methods, which employ various techniques to enhance the performance of the input, retrieval, or generation processes.

Input enhancement techniques

Input retrieval techniques focus on ensuring a higher-performing retriever, by improving the data preparation and optimizing for a smoother similarity searching process. To solve any issues with the query, query manipulation or refinement could be utilized to reformulate the wording or to add more meaningful terms, making the aim of the question clearer. Data augmentation could also be used to prepare the context data, where the data is paraphrased and/or summarized, to increase the readability, or to remove any irrelevant information. Some techniques also add information from other data sources, to enrich the contexts, should the text be unclear or ambiguous.

Retrieval enhancement techniques

These techniques aim to make the job of the retriever easier, whether that is by optimizing the chunk size to lower the chance of text being too large and including unrelated information, or text being cutoff in the middle of sentences, which would reduce semantic understanding. Other retrieval techniques work by optimizing the ranking of the relevant chunks, ensuring the most relevant are given higher importance, or by creating an adaptive retrieval process, which evaluates the query and decides dynamically whether a simple context lookup is needed, or whether a more complex method, such as recursive retrieval, is needed.

Generator enhancement techniques

The focus of these techniques is on developing the performance of the generator, whether that is through changing the generator itself, such as decoder tuning or generator fine-tuning, which can increase the chances of receiving the desired results. The quality of the generator inputs also factor in heavily in its performance, so using prompt optimization, in which the prompt, the text that precedes the query and context, can be tinkered with, to give the generator more information in how to answer the query. More complex techniques also repeat the retrieval and generation stages a set number of times in order to receive a more in-depth or cohesive answer. Iterative retrieval repeats the two stages, with the hope that repeatedly searching the external data would lead to better results. Recursive retrieval works similarly, but after each repetition, the original query is slightly altered in a way that would fix any ambiguity in the original. Both methods are followed by a judging module that decides whether the current answer is sufficient or another repetition is needed.

This chapter presented the evolution of NLP from the earliest machine translation to the benchmark LLMs of today. The architecture and key components of RAG were also explored, along with its limitations and proposed improvement techniques. Some of the more prominent techniques that have been introduced in recent years will be discussed in the next section, where their architecture and their effect on the RAG performance will be analyzed more in depth.

3
Related Work

This chapter provides an analysis of recent research conducted in the field of optimizing LLM performance using advanced RAG techniques with question-answer (QA) datasets. Section 3.1 starts by looking at several research papers that have proposed new methods of processing QA datasets, and examines the effect these methods have had on the performance. Based on these papers, Section 3.2 also details and contrasts the format and scope of the most frequently cited datasets with each other. Lastly, this chapter ends with Section 3.3 which looks at the gaps in the research and the work that needs to be done, and how this could be accomplished both by this thesis and by other future work.

3.1 Advanced RAG techniques

RQ-RAG

Refine Query for RAG (RQ-RAG) was proposed by Chan *et al.* [87] in order to combat complex or ambiguous queries. For straightforward queries, the model learns to search on demand, whereas the other queries are rewritten, decomposed, and disambiguated. A dataset was created, according to Formula 3.1, where the original input-output pairs are transformed using special tokens ($SPECIAL_{type}$) which specify the refinement action, $Q_{i,type}$ being the refined query, and the retrieved contexts $[D_{i1}, ...D_{ik}]$.

$$(X_{origin}, Y_{origin}) \rightarrow \underbrace{(X_{origin}, SPECIAL_{type}, Q_{i,type}, [D_{i1}, ...D_{ik}], Y_{new})}_{\text{repeat } i \text{ times}} \tag{3.1}$$

To facilitate this process, ChatGPT was used to group queries in refinement categories and to generate answers to the new queries, as seen in Figure 3.1. After the model (Llama2-7B) was trained on this constructed dataset, it was then tested on multiple single-hop and multi-hop QA datasets, with multi-hop referring to the answer of a

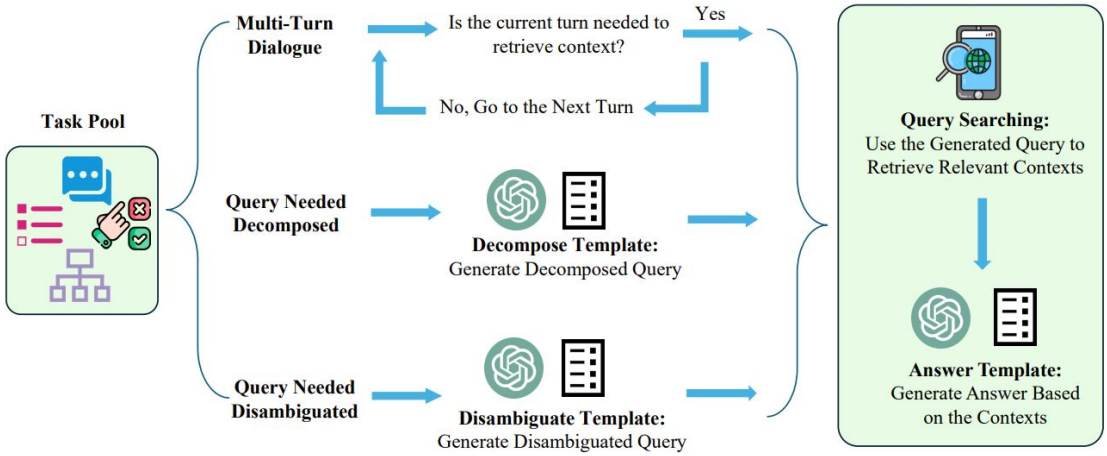


Figure 3.1: A diagram showing the role of ChatGPT in categorizing queries used to train RQ-RAG [87]. Queries would either be left as they are, or would be decomposed and rewritten to reduce ambiguity.

query needing multiple retrieval and reasoning steps from text sources, with single-hop questions typically needing one straightforward retrieval.

Model	Single-Hop QA	Multi-Hop QA
No Retrieval Baselines		
Llama2-7B (Zero Shot)	25.8	8.5
Llama2-7B-Chat (Zero Shot)	43.9	4.4
With Retrieval Baselines		
Llama2-7B (Zero Shot)	34.9	14.3
Llama2-7B-Chat (Zero Shot)	40.3	2.7
SAIL-7B	48.1	25.9
SELF-RAG	66.4	27.1
RQ-RAG	**68.3**	**49.7**

Table 3.1: Summarized table [87] showing the average results of multiple models with or without retrieval techniques across single- and multi-hop QA datasets.

The performance of the trained model was then compared with two main model groups: *No Retrieval Baselines*, which answer the question using their intrinsic knowledge, and *With Retrieval Baselines*, which adjoin retrieval from external data sources. Among these models are Llama2-7B and Llama2-7B-Chat, both the baseline version and one fine-tuned on task-specific datasets, SAIL-7B [88], which uses search-augmented instruction learning to give higher-quality responses by filtering out distracting search results, and SELF-RAG [89], which critiques and selects the best of multiple generated answers. Table 3.1 shows the performance of each model averaged across multiple datasets in the single- and multi-hop groups. In most cases, the performance of

the retrieval models is significantly higher than their counterparts, with multi-hop datasets proving especially difficult for the Llama models. RQ-RAG exhibits the highest performance, surpassing the others even in the complex, multi-step reasoning scenarios.

GraphRAG

Edge *et al.* [90] proposed GraphRAG to improve RAG performance when tasked with so-called *sensemaking* questions, which require a deeper understanding of an entire text. Unlike the prevalent RAG process shown in Section 2.4, GraphRAG uses graphs instead of vectors to store knowledge. The text sources are split into chunks, after which an LLM is used to process the text to obtain all possible entities, their relationships, and related factual statements. These are then used to create a knowledge graph, with nodes and edges representing entities and relationships, respectively, with the weight of an edge being based on the occurrence of the relationship. Leiden communities are also used to recursively partition the graph into similar, smaller communities, which are then summarized using a template. During the generation process, multiple answers along with helpfulness ratings are created in parallel for each community summarization by the LLM. The summarizations are subsequently ranked in descending order of their helpfulness and entered into a new context window, which is then used to generate the global answer.

The model was then tested on a podcast transcript and news article dataset, each having between $1 - 1.7$ million tokens, with the answers evaluated by an LLM on their comprehensiveness, diversity, empowerment, and directness. Six different approaches are tested, including a "traditional" vector RAG, a direct summarization model, and four graph-based methods using different levels of community summaries. The results of the experiments were quite mixed, with the vector RAG having the highest directness score for both datasets, and the direct summarization model having the highest empowerment rate for the podcast transcripts. The highest performers in the other evaluators are spread out relatively evenly among the graph methods, summarizing at deeper hierarchical levels.

Adaptive-RAG

Many different enhancement techniques have been proposed, each targeting specific areas of the RAG process or task-specific scenarios, but there are cases where multiple techniques could be beneficial for the same model. This is what Jeong *et al.* [91] aims to do with Adaptive-RAG, which can dynamically decide the most suitable technique for each query. There are three methods to choose from:

- Non-retrieval for QA: A traditional LLM that generates an answer based only on the input query and its own intrinsic knowledge.
- Single-step approach for QA: Simple RAG-based method, where the query is compared against an external knowledge source to find the most relevant segments, which are then used to aid in the answer generation.

- Multi-step approach for QA: Similar to RQ-RAG, the retriever and generator collaborate over multiple rounds, each time taking into consideration additional contexts and previous answers.

Adaptive Rag uses another language model (Classifier) to sequentially determine the complexity levels (low, moderate, high) of each query taken from multiple single- and multi-hop QA datasets. The model was then compared against the three aforementioned techniques, Self-RAG [89], and Adaptive-RAG with an Oracle classifier. To evaluate the performance, the metrics: F1, accuracy (Acc), exact match (EM), number of retriever-generator steps (Step), and the average time taken to answer each question, the latter two being relative to the single-hop approach.

Model	*EM*	*F1*	*Acc*	*Step*	*Time*
No retrieval	35.77	48.56	44.27	0.00	0.71
Single-step approach	34.73	46.99	45.27	1.00	1.00
Self-RAG	10.87	22.98	34.13	0.74	1.50
Multi-step approach	38.13	50.87	49.70	2.81	3.33
Adaptive-RAG	37.97	50.91	48.97	1.03	1.46
Adaptive RAG with Oracle classifier	47.70	62.80	58.57	0.50	1.03

Table 3.2: Summarized table [91] showing the average results of multiple models RAG approaches across single- and multi-hop QA datasets using a GPT 3.5 (Turbo) LLM.

Table 3.2 shows the results of the listed models averaged over all datasets and using GPT 3.5 (Turbo) as the backbone LLM. As can be expected, the simpler models (no retrieval, single-step approach) had a lower performance than their more complex counterparts. On the other hand, the multi-step approach did perform similarly well to Adaptive-RAG, but the *Step* and *Time* show that this method comes not only at a much higher computational cost, but also financially, since the LLM has to be accessed repeatedly. Adaptive-RAG with an Oracle classifier is the best performer, both in terms of answer validity and efficiency, thereby serving as an upper bound for Adaptive-RAG since the Oracle classifier is an idealized model, always selecting the correct complexity for each query.

3.2 Question-answer datasets

As described earlier, single- and multi-hop question-answer datasets seem to be the most commonly used when testing RAG enhancement techniques. This section gives a brief overview of the most prevalent datasets, their size, data sources, and format.

Single-hop datasets

SQuAD or Stanford Question Answering Dataset [92] is a single-hop dataset, consisting of 107,785 question-answer pairs taken from questions posed in 536 Wikipedia articles.

Crowdworkers were hired to create the questions and select the text segment they believe best answers them, leading to 19.2% of answers being dates or numerals, 32.6% being nouns such as person, location, or other entities, 31.8% being common noun phrases, and the rest being made up of other phrases or clauses.

TriviaQA [93] is a dataset made up of over 95,000 unique question-answer pairs and 650,000 question-answer-evidence triples; the questions were gathered from trivia websites, and the evidence was sourced from Wikipedia articles or Bing search results. Since the evidence documents could be quite large, they are seen as a *distant supervision* method, which could be used to train the model to wade through noisy environments. The majority of answers (92.85%) are just the titles of Wikipedia pages, with numerals and free text making up 4.17% and 2.98% respectively.

PopQA, introduced by Mallen *et al.* [94], is a 14,000 question-answer pair dataset that aims to cover information that might not have been included in previous datasets. This is done through traversing Wikidata, gathering knowledge triples and converting them to QA pairs, and collecting the pageviews for each article, thus determining its popularity.

Multi-hop datasets

2WikiMultiHopQA [95] is a Wikipedia-based dataset spanning over 192,000 triples in the (*subject,relation,object*) format. To avoid overly straightforward single-hop questions, four question templates were constructed [95]:

- Comparison question: Seeks to compare components of two or more entities.
- Inference question: Uses two related triples, such as (*s1,r1,o1*) and (*s1,r2,o2*) to create a new triple (*s1,r3,o2*), a combination of the two properties is created.
- Compositional question: Similar to the above, but in this case, a new connection cannot be created from *r1* and *r2*. Both properties would then have to be mentioned in the question to enable the connection of the triples.
- Bridge comparison question: Combines comparison and inference/compositional questions, in which two or more data pieces taken from their respective compositional questions are compared against each other.

The top five answer categories were found to be: yes/no (31.2%), dates (16.9%), films (13.5%), persons (11.7%), and cities (4.7%), with the remaining being other entities.

HotPotQA [96] is another Wikipedia based dataset of 113,000 QA pairs. It is made up of comparison and yes/no questions, with many of them requiring chain reasoning, which involves the identification and understanding of bridge or answer entities and their respective properties. The main dataset can be used in two different settings: **distractor** which mixes the two *gold paragraphs*, from which the answer can be deduced, with 8 other *distracting* paragraphs; and **full wiki**, which provides the first couple of paragraphs for all Wikipedia articles (over 5,000,0000).

3.3 Gaps in current research

After discussing some recently presented RAG enhancement techniques and commonly used datasets, it is clear that RAG-based models have expanded significantly in the last years and will only continue to develop further. However, it can be seen that there are still some unanswered questions and unexplored areas left.

Firstly, there is a lack of larger or medium-sized datasets that have long-form answers requiring multi-step reasoning. The larger datasets [95, 96] that tend to be used in RAG surveys tend to have more simplistic answers, such as short phrases, binary options, or multiple-choice, for which less complex evaluation methods are needed. Datasets that do have more in-depth answers are often smaller in size [97], or are not publicly accessible, thus they are not able to be replicated by other researchers [98].

The comparison of RAG methods across many different datasets is also missing, since most studies focus on a smaller number of datasets and more tailored RAG methods. These comparisons help in seeing how the different structures of each dataset affect the performance, and seeing how flexible certain RAG methods are in this regard. This would allow future researchers to not only know which datasets work well with certain methods, but also, most importantly, which ones work especially poorly.

4 Methodology

This chapter discusses the plan and setup for the experimental study, starting first with describing the structure and content of the datasets that will be used in Section 4.1. It then continues in Section 4.2 by discussing an important library that will be used to facilitate the execution and management of the methods and datasets, followed by an explanation of the evaluation metrics in Section 4.3 and the prompt design in Section 4.4. It ends on Section 4.5, with a thorough detailing of the control and RAG methods that will be used, and how they will be implemented.

4.1 Data

The dataset that will be used in the experimental study is ChatRAG-Bench [1] compiled by NVIDIA, and consisting of 10 QA conversational datasets covering varying QA formats and topics. These datasets were all slightly modified from their original versions to create a unified format and structure, leading to easier processing. The modified datasets all contain three core columns:

- **Messages** - list the QA conversations regarding a specific topic, in which the *user* asks questions and is answered by the *assistant*. Each entry contains the relevant conversation history, which could also be left empty if a new conversation is starting, and concludes with the latest question left unanswered. While the first questions in the sequence are more direct and coherent, the questions that appear later tend to be less understandable when removed from the general context, since they often informally refer to the previous answers (*which of those, how many of them*).
- **Answers** - contain the answer(s) of the last question in *Messages* in an array format, the length of which varies depending on the dataset. As seen in Table 4.1,

[1] https://huggingface.co/datasets/nvidia/ChatRAG-Bench

the multiple answers are there to compensate for question ambiguity or semantic differences in phrasing.

- **Contexts** - consists of the pertinent context(s) regarding the conversation. The format of this column is determined by whether the dataset contains another column.
  - Without **Ground-Truth-Context** - in this case, the column *Contexts* consists of only one paragraph, which contains the answer to the latest question.
  - With **Ground-Truth-Context** - in this case, *Contexts* consists of multiple paragraphs which are partially related to the conversation, but which might not necessarily hold the answer. The Ground-Truth-Contexts then contains the most relevant of these contexts.

messages	*answers*
User: What happened in the early 80's?	[Farina was cast in a film, he got into acting, he was a consultant, got into acting]
User: What color was Cotton?	[white, white, white, white]
User: Whose paint was it?	[the farmer, the farmer's, the old farmer's, the farmer's]

Table 4.1: Summarized table [2] showing the multiple answer format of the CoQA dataset.

The datasets HybriDial and ConvFinQA do not comply with the aforementioned formatting rules and will therefore not be used in the experimental study. HybridDial is a natural, conversation-based, multi-hop QA dataset, which would beneficial for the study but since the **Contexts** column contains multiple paragraphs and no **Ground-Truth-Context** exists, it would not be possible to verify the performance of the retriever. ConvFinQA, unlike the other datasets that primarily focus on language reasoning and text understanding, uses numerical-based tables from the finance sphere and aims to test numerical reasoning and arithmetic calculations. Unfortunately, the conversion of HTML tables to text creates certain formatting and alignment errors, which can lead to the incorrect handling of data.

Sequential Question Answering (SQA)

SQA [99] was developed to create question sequences, in which each answer depends on its predecessor. This is done by decomposing questions from the WikiTableQuestions dataset into more specific and comprehensible sub-questions, leading 17,553 question-answer pairs and 6,066 unique contexts.

Question Answering in Context (QuAC)

The QuAC dataset [100] contains 14,000 QA dialogues, created by crowdworkers acting out conversations between a *student* and *teacher*. The conversations are centered around a Wikipedia article, which the student knows only the title of but the teacher has full access, and the interaction continues until one of three options is completed: 12 questions have been answered, the student or teacher decides to end the dialogue, or two questions go unanswered.

Conversations Question Answering (CoQA)

Reddy *et al.* [101] introduced the 127,000 CoQA dataset with the aim of including varied data sources such children's literature, school exams, and Reddit, and casual sounding speech, in which questions often refer to the dialogue history and answers are direct and without explanation. As in the paper introducing this dataset, evaluation metrics will be performed between the generated answer and each of the reference answer, but only the one having the highest F1 score will be selected.

Domain-specific Question Answering (DoQA)

The DoQA [102] dataset is built upon a continous dialogue between a user and a domain expert relating to a certain topic, in this case cooking, travel or movie forums on Stack Exchange. Similar to CoQA, there are four correct answers for each question, with DoQA being a little bit smaller, 10,917 QA sets, and aiming for a more natural conversation, with less clunky factoids and follow-up prompts. Since this dataset is comprised of multiple sub-datasets, the subsequent results and findings will be averaged across all three.

Doc2Dial

IBM created doc2dial [103], a dataset consisting of 4800 conversations between a *user* and an *agent* on topics related to social welfare in the United States. The conversations were sorted into three different categories: *D1*, containing multiple questions relating to the given context, *D2*, in which the conversation revolves around one main inquiry with clarifying questions carried out by the agent, and *D3*, with question that are irrelevant to the context.

Question Rewriting in Conversational Context (QReCC)

The QReCC [104] dataset incorporates questions from pre-existing datasets, including the aforementioned QuAC, with information sourced from the Common Crawl dataset

Inital question	**Q**: Tell me about the benefits of Yoga?
Replacement	**Q**: Does it help in reducing stress? **R**: Does Yoga help in reducing stress?
Insertion	**Q**: What are some of the main types? **R**: What are some of the main types of Yoga?
Removal	**Q**: Can you tell me about the C++ language mentioned? **R**: Can you tell me about the C++ language?
Copy	**Q**: What are common poses in Kundalini Yoga? **R**: What are common poses in Kundalini Yoga?

Table 4.2: Summarized table [104] showing the process of rewriting pre-existing questions due to anaphores or missing content in the QReCC dataset.

and web searches. Multiple methods of question rewriting were also used to "fix" any inquiries which had references to the conversation history, see Table 4.2, thereby preserving the natural-sounding sentence structure, while removing ambiguity.

Topic switching in Open-domain Conversational Question Answering (TopiOCQA)

This dataset [105] explores topic switching within $50,000$ conversational QA pairs, by allowing the *answerer* access to links within the relevant Wikipedia article. The textitquestioner is then also allowed to view the metadata (main segment and chapter titles) of the linked articles, and can shift their inquiries accordingly. The answers were unrestricted and free-form, facilitating easier topic switches every 3-4 questions and enabling the handling of the changing context and long-term reasoning.

Information-Seeking Conversations with Mixed-Initiative Interactions (INSCIT)

Wu *et al.* [106] proposed an information-seeking dataset that would use a variety of interactions to challenge hard-to-answer questions. These mixed-initiative interactions are categorized into:

- Direct answers - the model finds information that it believes answers the question, and relays that to the user.
- Relevant answers - the model does not believe it has found a valid answer but will inform the user of information it believes is relevant.
- Clarifications - the user is prompted for further clarification on their question, and often occurs when there are many related contexts available.
- No information - the model has no answer available for the user.

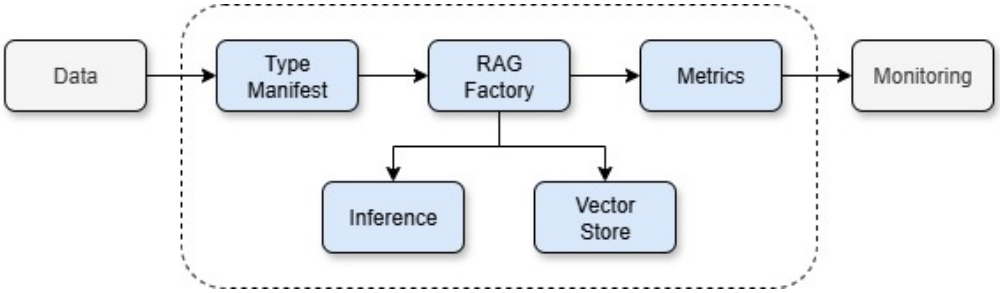


Figure 4.1: A diagram of the EncouRAGe library, inspired by [5]https://github.com/uhh-hcds/encourage, showing its framework for executing RAG methods.

4.2 Encourage

To facilitate the management and execution of the various datasets and RAG methods, the EncouRAGe library [3] is integrated into the code following its accompanying template [4]. EncouRAGe is a modular and flexible framework, whose workflow is shown in Figure 4.1 that supports the RAG process by improving inference efficiency, storing results and metadata, and enabling the use of multiple RAG methods and evaluation metrics.

EncouRAGe uses two important libraries (vLLM, MLflow) to streamline LLM inference and the monitoring of runs and results. vLLM [107] is an open-source, end-to-end LLM serving engine that aims to optimize key-value (KV) cache memory usage. KV cache memory is comprised of the key and value tensors necessary for the attention function in transformers, and since it expands and shrinks dynamically based on the request size, it can be prone to fragmentation and management complexity. vLLM introduces PagedAttention, an algorithm that divides the KV cache memory into blocks that hold a fixed number of key-value pairs. The more efficient memory management allows for the successful batching of large amounts of requests. MLflow [108] is also an open-source platform designed to streamline model deployment, experiment execution, and reproducibility. The main features used in EncouRAGe are in regard to experiment tracking, where the parameters of each experiment (RAG method, dataset name, LLM model, etc), along with the evaluation metrics, and the full table of the requests, contexts, and results.

Another important part of the RAG method is the vector database where all external contexts are stored and subsequently retrieved. The one used in this case is the open-source, embedding database Chroma [109] which has an easily accessible core API, made up of four main functions. A ChromaDB client is created, and the collection of contexts are added to it, along with the choice of *embedding models* and *distance* function. While Chroma supports a variety of different embedding functions, the default is the Sentence Transformers [110] **all-MiniLM-L6-v2** model, which maps words and sentences to 384-dimensional vector embeddings. The distance function is used to calculate the approximate nearest neighbors (ANN) to the query vector, thus finding the

[3]https://github.com/uhh-hcds/encourage
[4]https://github.com/pesc101/exp-template

				Metrics		Parameters		
	Run Name	Created	Duration	f1	mrr	dataset_size	dataset_subset	rag_method
	gregarious-worm-911	4 days ago	7.9min	0.540919479...	1	2792	qrecc	known_context
	unequaled-lynx-246	4 days ago	27.3min	0.362024531...	0.229793457...	2792	qrecc	hyde_reranker_rag
	adorable-cow-582	4 days ago	24.0min	0.361549368...	0.476796800...	2792	qrecc	hyde_rag
	peaceful-bear-486	4 days ago	16.1min	0.306355326...	0.211115095...	2792	qrecc	reranker_rag
	traveling-bug-925	4 days ago	2.5h	0.362213885...	0.145827363...	2792	qrecc	summarization_context
	clean-shrimp-480	4 days ago	15.2min	0.295930229...	0.197884546...	2792	qrecc	hybrid_bm25
	hilarious-hen-166	5 days ago	15.5min	0.297771080...	0.205760726...	2792	qrecc	hybrid_bm25

Figure 4.2: A snapshot of the MLflow platform tracking the parameters and evaluation metrics of experiments.

most relevant contexts, with *Squared L2*, *Inner Product*, and *Cosine Similarity* available. The default distance function, Cosine Similarity, as shown below

$$\mathrm{d} = 1 - \frac{\sum A_i \times B_i}{\sqrt{\sum A_i^2} \cdot \sqrt{\sum B_i^2}} \tag{4.1}$$

will be used in the experiment since it places more focus on the vector's direction rather than its magnitude, thus giving priority to the texts semantic meaning.

4.3 Evaluation

There are a variety of different evaluation metrics available in the EncouRAGe platform to evaluate the performance of both the generators and the retrievers. Among the generator evaluation metrics considered were BLEU [111], GLEU [112], ROUGE [113] and F1, whereas Recall@k and Mean Reciprocal Rank (MRR) were both used for assessing the retrievers performance.

Before discussing these metrics, it is important to explain two measures that are widely encountered: Recall, which determines how many *true positives* are *predicted positive*, and in terms of LLM generated answers it reflects the completeness of the answer; and Precision, which calculates how many of the *predicted positives* are actually *true positives*, and represents the correctness of the answer.

Generator evaluation metrics

The F1 score is calculated using the harmonic mean of the token-based precision and recall,

$$\mathrm{F1} = \frac{\mathrm{Precision} \times \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}} \tag{4.2}$$

thus creating a balance between correctness and completeness. BLEU [111], first used for assessing machine translation, is based on the geometric average of generated answers

n-gram precisions p_n with an added brevity penalty (BP) for generations shorter than the reference answer length. This is shown in

$$\text{BLEU} = BP \times \exp\left(\sum_{n=1}^{N} w_n \cdot \log p_n\right) \tag{4.3}$$

with N and w_n representing the size and weights of the n-grams respectively. Using this as a foundation, the GLEU [112] metric keeps the n-gram comparison but instead using the harmonic mean of F1. On the contrary, ROUGE [113] is more recall-oriented and is split into multiple varieties, with ROUGE-L used to determine the longest common subsequence between the generated and reference answer, and ROUGE-N being calculated as follows:

$$\text{ROUGE-N} = \frac{\sum_{\text{gram}_n \in \text{Ref}} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{\text{gram}_n \in \text{Ref}} \text{Count}(\text{gram}_n)} \tag{4.4}$$

where $Count(gram_n)$ is the total number of n-grams in the reference answer and $Count_{match}(gram_n)$ is the amount of n-grams occurring in both answers.

Ultimately, since the original paper [114] that introduced and experimented with the ChatRAG-Bench dataset collection focuses its results on F1, in order to have a fair comparison this study will also use F1 as the main evaluation metric to compare the generated answer to the reference answer. As mentioned in 4.1, some of the datasets have multiple correct answers, so similar to it is handled in their respective papers [92, 100, 101], the F1 score will be separately calculated for each possible answer, with the one possessing the maximum score being chosen.

Retriever evaluation metrics

The performance of the retriever is evaluated on a two-pronged approach: first, does the retriever fetch the correct context, and second, is the correct context ranked higher than the others. Recall@k is a binary metric which outputs a 1 if it determines that the ground truth context appears within the top k retrievals, or a 0 otherwise. MRR [115] emphasizes the priority of the correctly retrieved context by averaging the reciprocal of their ranking, as shown below,

$$\text{MRR} = \frac{1}{N} \sum_{n=1}^{N} \frac{1}{rank_i} \tag{4.5}$$

where N is the total number of retrieved contexts, and $rank_i$ representing the rank of the ground truth context(s). In the experiment study, both metrics will be essential in order to determine how significant the effect of the ranking is on the performance and to ease the comparison of the different RAG methods.

4.4 Data Preprocessing and Prompt Design

Before any RAG methods are implemented, the datasets must first be correctly fetched and prepared, to ensure that the are correctly processed. The datasets are loaded

from Hugging Face [6] and converted into a Pandas DataFrame, after which the queries, answers, contexts, and ground truth contexts are extracted. The queries are structured as a conversation between a user and an assistant, and this format is kept, not in the original JSON format but rather in a more readable form shown in Table 4.3. The collected contexts are also cleaned to remove any noisy formatting differences, with the ground truth contexts added as *reference_document* to the metadata.

Original format	*New format*
{"content": "how old was william shakespeare when he wrote hamlet", "role": "user"}, {"content": "Between 1589 and 1613 at the age of 49.", "role": "assistant"}, {"content": "who is he?", "role": "user"}	Conversation history: User: how old was william shakespeare when he wrote hamlet Assistant: Between 1589 and 1613 at the age of 49. User: who is he?

Table 4.3: Example of the conversation format processing, performed to remove any unnecessary JSON syntax elements and enhance the clarity of the query. The data in this example is taken from the TopiOCQA dataset.

Regarding the prompt design, a similar blueprint was used for each dataset in which the LLM is told that that they are a helpful assistant whose goal is to try and answer the questions to the best of their abilities, and the importance of the contexts and conversation history is emphasized. The LLM is instructed to not rely on internal knowledge, and to state if the answer cannot be found, rather than doing any guesswork. The main difference among the prompts, which will be provided in full in Appendix A, is the guidance on how the answer should be structured, whether in more explanatory full sentences, succinct phrases or specific values. Since the addition of the conversation history and contexts would already be quite lengthy, and to match the ChatQA paper [114], zero-shot prompting was chosen, thus no ideal response examples will be passed along.

4.5 RAG methods

This section delves into the RAG methods implemented in the EncouRAGe library, and discusses their underlying mechanisms, including how they augment the basic RAG process, and how these enhancements are predicted to affect the performance. The first approach to be discussed is No RAG, which will be used as a control group to test the performance and internal knowledge of the LLM. Followed by the base implementation of RAG and the *ideal* retriever method (Known Context), since they will serve respectively as a foundation for the more advanced methods and a performance

[6] https://huggingface.co/

ceiling for the retriever. Then, the methods will be discussed in order of which aspect of the RAG pipeline is enhanced: input-enhancing (HyDE and Query Rewriting), retriever-enhancing (HybridBM25, Reranker RAG), a combination of the two (HyDE + Reranker), and generator-enhancing (Context Summarization).

4.5.1 No RAG

In order to contextualize the effects that the following RAG methods will have on the performance, the LLM itself must first be analyzed. The No RAG method will send the user query and prompt directly to the LLM, without any additional context, in order to observe its internal knowledge and how well it can answer the question. Although, the prompt would still contain the conversation history, which would ideally provide enough contextual information to enhance the clarity and comprehensibility of the query. Since the experiment will be conducted with multiple datasets, all of which have different query structures and formats, this method will also help in creating a baseline performance value that is individually tailored to each dataset, so that the RAG results are not in any way inflated or minimized.

4.5.2 Known Context

This method, also known as *gold oracle*, is used to simulate the performance of an ideal retrieval, allowing for the isolated testing of the generator. Instead of the top relevant contexts being passed along to the LLM, only the ground truth context(s) for each query are relayed; by doing this, the performance ceiling of the generator can be ascertained. This ceiling acts as a sort of baseline for better interpreting whether another methods performance is strong or weak, and it puts into perspective the potential for any future improvements. The system prompt and other optimization techniques can then also be independently assessed and engineered.

4.5.3 Base Implementation

This technique, also known as Naive RAG, as described in Subsection 2.4.1 consists of loading and processing the datasets from Hugging Face, storing the loaded contexts into the Chroma vector database, retrieving the most relevant ones for each query and using them to generate the appropriate responses. The BaseRAG class is instantiated which starts by initializing the ChromaClient database and inserting the the context documents after they have been filtered to remove any duplicates.

Then it is time to retrieve the relevant contexts for each query, which is done using the Chroma query function which uses the aforementioned embedding model and retrieval queries, in this case the conversation history, to find the top k contexts. K could be set to a variety of different numbers, with the higher numbers ensuring a larger chance that the correct context is fetched. However, retrieving a larger number of contexts could cause the LLM to get distracted from the original question, or to get overwhelmed with noisy data, so effort must be made to find a balance between the two. Fortunately, Liu *et al.* [114] found $k = 5$ to be a happy medium, and found that lowering

```
Context: {{ documents[0].content }}
{{ user_prompt }}
Assistant:
```

Figure 4.3: The template of the prompt setup which sets the placement of the context, user prompt, and conversation history

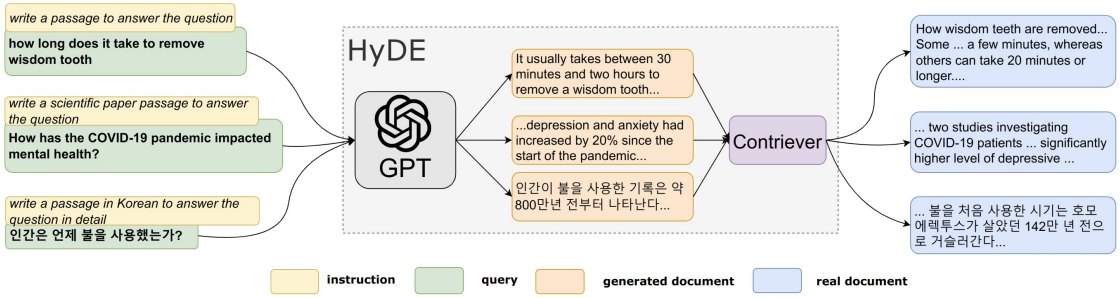


Figure 4.4: A diagram [116] showing how the HyDE model uses the initial user query to generate a hypothetical answer document, which will then in turn be encoded and used to search for the relevant contexts in the vector database.

the value to $k = 3$ or raising it to $k = 10$ both hindered the F1 score. The retrieved contexts along with the conversation history and system prompt, are then all collected in the template shown in Figure 4.3 and sent to the LLM through an inference runner.

4.5.4 HyDE

Hypothetical Document Embeddings (HyDE) was first introduced by Gao *et al.* [116] to improve the performance of zero-shot dense retrieval models. As shown in Figure 4.4, HyDE splits the retrieval process into two tasks:

- Generative task = the query is fed to the generator, which is then tasked to create a hypothetical document that would answer the question. This document, while it could contain errors and is unlikely to provide a full answer, would be able to capture the themes of the query and provide a guideline for the next task.
- Document similarity task = the hypothetical document is encoded into an embedding model, with the expectation that the *dense bottleneck* in the encoder would filter out the smaller, incorrect details. This vector is then used to search for the relevant contexts, on the basis of their similarity.

This process improves the efficacy of the retriever by working to bypass any noise or ambiguity in the original query, and the new generated document expands the query topic by adding contextual details. The HyDE function applies this process by using an additional query to instruct the LLM to create a passage that answers the provided question

> Rewrite the last user question to be fully self-contained and clearly answerable. Include only the essential context from the conversation. Do not add information that was not explicitly mentioned. The rewritten question should be short, precise, and reflect exactly what the user wants to know. Output only the rewritten question.

Figure 4.5: The prompt used to instruct the LLM into reformulating the conversation history in one, more comprehensible question.

4.5.5 Query Rewriting

Another input enhancement method that will be used is query rewriting, which reformulates the original query to aid in better retrieval results. Unlike HyDE, the goal of this method is not to provide extra topical information or necessarily expand the query itself, but to clarify any ambiguous coreferences, and perform entity disambiguation or grammatical restructuring.

This method uses a separate function and a short prompt as seen in Figure 4.5, to instruct the LLM to rewrite the last question of the conversation, which will then be used instead of the conversation history when sending the final prompts. It would be worthwhile to examine whether the extra background to the question that the conversation history or HyDE provides is beneficial to answering the question or if the more succinct, direct query allows for less noisy details.

4.5.6 Hybrid BM25

This method combines dense embedding retrieval, which is focused on capturing semantic meaning between the query and documents, and sparse retrieval, in which the relevance of document is heavily impacted by lexical similarity. As mentioned previously, the dense embedding used will be the Chroma vector database, along with its associated retrieval functions, whereas the Okapi BM25 will be used for sparse retrieval.

The Okapi BM25 (Best Matching) function, developed by Robertson *et al.* [117], is used to rank the relevance of documents as pertaining to a given query. This ranking score is calculated through the formula

$$\sum_{t \epsilon Q} w^{(t)} \frac{(k_1 + 1)tf}{K + tf} qtf \tag{4.6}$$

where:

- t is a term in the query Q
- $w^{(t)}$ is the term weight, which defaults to $IDF^{(t)}$ if the relevant documents are not known ahead of time. IDF (inverse document frequency) is used to emphasize the importance of words that appear in fewer documents overall, thereby giving commonly used words less weight.

- tf is term frequency, which determines how often a term appears in a document.
- qtf is query term frequency, determining how often a term appears in the query.
- K is equal to $k_1((1-b)+b * \frac{dl}{avdl}$ with the tuning parameters k_1 and b, document length dl and average document length $avdl$.

To combine these two approaches, the dense and sparse retrievals are performed in sequence, with the provided retrieval queries. The dense retrieved documents are already sorted in order of relevancy, whereas for sparse retrieval the scores are calculated separately and normalized. To determine how much emphasis is placed on each of the approaches, two variables *alpha* α and *beta* β are used which control whether the dense or sparse retrievals respectively are given more influence. After these variables are set, the individual scores (inverted rank scores) can then be assessed based on the normalized position of each document. The hybrid scores of the documents are then calculated through

$$\text{hybrid score} = (\text{dense score} * \alpha) + (\text{sparse score} * \beta) \tag{4.7}$$

and sorted, with only the highest scoring k documents being selected.

4.5.7 Reranker

To optimize the retrieval process, the Reranker method tries to increase the chance of fetching the ground truth context by retrieving more than the k documents needed, assessing their relevance, then further refining them using a cross-encoder. Unlike bi-encoders, which calculate the similarity score after encoding the queries and documents separately, cross-encoders score them by first concatenating the two with a separator token, as in the format below,

$$\textit{[CLS]}\ \text{query tokens}\ \textit{[SEP]}\ \text{document tokens}\ \textit{[SEP]} \tag{4.8}$$

with *CLS* representing the start of the sequence [118]. This means the model chosen for the experimental study **ms-marco-MiniLM-L6-v2** [7], can process both queries and contexts together, allowing for more subtle similarities to be picked up and a more accurate relevance score to be calculated.

4.5.8 HyDE + Reranker

As it can be gathered from the name, this method is combination of the aforementioned HyDE and Reranker methods. A Reranker instance is first instantiated which starts by collecting a larger pool of contexts, according to the formula below

$$initial\_top\_k = max(int(top\_k * rerank\_ratio), top\_k + 2) \tag{4.9}$$

[7] https://huggingface.co/cross-encoder/ms-marco-MiniLM-L6-v2

> Rewrite the context to keep all essential facts and remove only irrelevant or redundant details, without adding new information.

Figure 4.6: The prompt used to instruct the LLM into condensing the contexts while still keeping the essential information.

where the top_k is the desired number of contexts and the $rerank\_ratio$ is the multiplier used to retrieve the larger initial amount of contexts. The HyDE instance is then used to create a document that would aim to answer the query, which will then be used to retrieve the top k contexts from the larger $initial\_top\_k$ context pool. The Reranker then uses its cross-encoder model to rerank the fetched contexts in order of relevancy.

4.5.9 Summarization Context

This method aims to tackle the problem of querying noisy, unfocused contexts by condensing them into their most necessary information. This is done through the prompt shown in Figure 4.6, with which the LLM is prompted to summarize the contexts by discarding any tangential information. These contexts will then be used by the retrieval queries to find the top k documents, with the goal being that they will be shorter and more efficient to sift through.

From here there are two approaches that could be taken, either the summarized contexts retrieved could be sent along with the final user prompts to the LLM, or they could be used to find the original, unmodified contexts which will be sent along instead. The first method (SummarizationRAG) allows the retriever and generator to both be working with the same view of the contexts, since including details that were initially deemed distracting and were filtered out would be counterproductive. On the other hand, the second method (SummarizationContextRAG) could be a *best of both worlds* approach, utilizing the summarization when quicker and more efficient retrieval is needed, all the while reducing the risk of critical details being left out or external knowledge being included.

5 Experimental Study

This chapter presents the experimental study conducted to compare the performance of advanced RAG methods on multiple datasets of varying topics and formats. The experiment is built upon the plan laid out in the Chapter 4, and starts with Section 5.1 outlining how the LLM model was setup for this design to be implemented. Section 5.2 continues with a discussion on the results that can be expected from the experiment, based on a preliminary analysis. Afterwards, Section 5.3 continues by discussing the experiment results, laying out the performance expectations that were set, and examining whether the final results fit these predictions. Section 5.4 finishes by doing a deeper analysis of any datasets or RAG method that displayed irregular outcomes.

5.1 Implementation

As mentioned in Chapter 4, the implementation of the experimental study was carried out using the Python library EncouRAGe, with the Llama 3 8B Instruct model [71]. This model was chosen since it exhibits stronger language understanding than Llama 2 models [119] or smaller Llama 3 models such as Llama 3.2 1B [1], while requiring a more manageable level of computational resources than larger models such as the 70B or 405B [71] models. This model is also the one used by Liu *et al.* [114] to test the ChatRAG-Bench datasets, so an easier comparison could be made between the results.

The Llama 3 models are developed using a **pre-training** stage which trains the model for next-token prediction using various text corpuses This stage builds upon the Llama 2 models by employing a larger tokenizer with a vocabulary of 128K tokens resulting in a more time-efficient text input, improving inference speed using grouped-query attention (GQA) [120], which acts as a middle ground between multi-head and multi-query attention, and ensuring that self-attention does not get applied across multiple documents by using an attention mask. The *Instruct* models add a **post-training**

[1] https://ai.meta.com/blog/llama-3-2-connect-2024-vision-edge-mobile-devices/

stage which uses a combination of supervised fine-tuning using a specially trained reward model and direct preference optimization (DPO) to improve human preference alignment and instruction following.

When setting up for the experiment, certain parameters of the model can be altered so they are better tailored to the task at hand while still keeping efficient computing. The *temperature*, which controls how deterministic the model is and how much of a role randomness plays, was set to 0 since reproducible results are necessary in order to have a fair comparison between the different RAG methods. The *max_tokens* parameter, that decides the maximum length of the generated answer, was set to 1000 and the *max_model_len*, which sets the maximum context length that the model can process, was set to 40000.

The use of the LLM was assisted by the NVIDIA RTX A6000 GPU, which is used to reinforce the heavy computational needs. The Ampere-based architecture is equipped with 48 GB of ECC GDDR6 memory and over 10, 700 CUDA cores, which makes it ideal for the large data processing and intensive inference needed for the experiment [121].

5.2 Preliminary dataset analysis

This section starts by discussing some of the main differences between the datasets and how they could have an impact on the retriever or generator performance. It then describes the results of a preliminary analysis that was conducted to provide further insight into the results that can be expected for each dataset.

As can be seen in Table 5.1, the length of the answers varies considerably, from the more concise answers of the CoQA dataset to the lengthier explanations of INSCIT, which were already taken into account during the creation of the system prompts, shown in Appendix A. While lengthier contexts risk containing more distracting content and reducing generator performance, shorter contexts could be disadvantageous if they do not provide sufficient contextual information. Another dataset characteristic which could affect the performance of the retrieval is the number of total contexts, which, in the case of INSCIT, QReCC, QuAC, and especially TopiOCQA, might cause difficulties in retrieving the correct context.

Firstly, the internal knowledge and information searching capabilities of the LLM had to be assessed. This was done using the No RAG method, which sends the queries to the LLM without any additional contexts, and through which the baseline performance can be established. In addition to this, the ceiling performance could also be determined through the *gold oracle* or Known Context method, which attaches only the ground truth context to the query. The ceiling value helps in ascertaining whether the appended contexts actually improve the LLMs performance by a considerable amount, and if so, by how much. It is also informative regarding whether the main obstacle lies in retrieving the correct context or the reasoning ability of the model with the dataset at hand. A high ceiling shows that improving the retriever is key in achieving more accurate answers, whereas a lower ceiling reveals that the dataset may not be well handled by the models inference capabilities.

Dataset	Size	Token length			
		Nr. of Contexts	Question	Answer	Context
CoQA	7.98k	8482	7.69	4.43	333.02
SQA	3.01k	3197	10.69	37.26	444.84
TopiOCQA	2.51k	171705	9.12	17.30	97.71
QReCC	2.79k	22067	8.12	28.16	512.88
DoQA	1.79k	2193	13.16	19.00	326.69
QuAC	7.35k	33669	8.81	19.91	486.98
Doc2Dial	3.94k	5177	12.52	22.77	376.77
INSCIT	502	29940	12.23	45.32	101.74

Table 5.1: Table showing the size and formats of the datasets used in the experimental study.

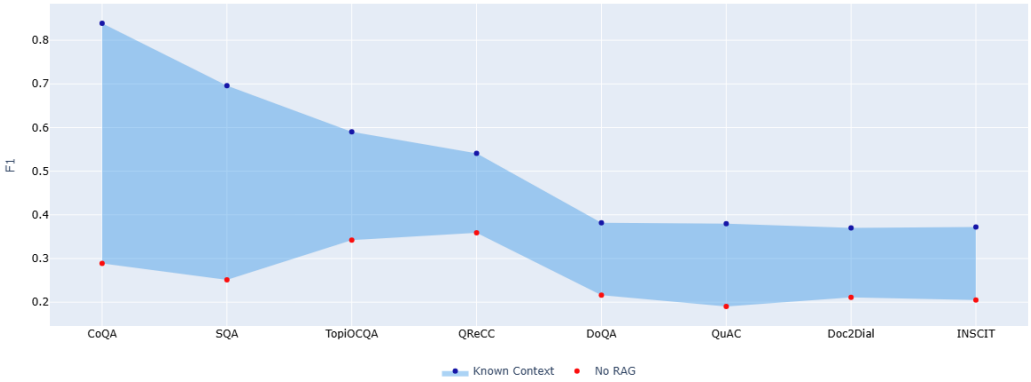


Figure 5.1: Plot showing the theoretical minimum and maximum ranges of F1 that can be achieved with the LLM. The minimum is achieved through the No RAG method, in which no contexts are retrieved, whereas the maximum is taken from the Known Context method, which directly takes the correct context.

The performance difference of these two methods is shown in Figure 5.1, which shows the range of F1 for each dataset. As one would expect, the retriever evaluation metrics are not needed in this case since they would be valued at 1 for Known Context, and would not be subject to evaluation for No RAG. It can be seen that for half of the datasets, the ceiling F1 value lies under 40% mark and the entire F1 range is only 15−20%, which shows that there is substantial headroom for retriever improvements, but also shows that there are likely other bottlenecks hindering performance, such as generator capability, or problems within the dataset itself. The other datasets show a more expected F1 range, in which the lack of supplementary contexts is the main performance obstacle. Though the RAG methods will be analyzed for all datasets, the retriever performance will likely be more clearly shown in the ones with a wider F1 range.

Figure 5.1 also shows that except for the QReCC and TopiOCQA datasets, the LLM has approximately equal internal knowledge regarding the subject matters. The training of many LLMs is done with a wide variety of publicly accessible or open-source data, so the No RAG method functions as a suitable test of whether certain datasets may have been used during the pre-training or fine-tuning phases.

5.3 Empirical result analysis

This section provides an overview analysis of the experiment results by illustrating how the evaluation metrics were affected by the RAG methods described in Section 4.5. It first lists the overall F1 and MRR results per dataset and RAG method and shortly discusses the outcome of each method. It then analyzes the combination of the two evaluation metrics and whether there is any correlation between their performance.

5.3.1 F1 performance by RAG Method and Dataset

This section discusses the performance of the RAG methods separately for each dataset, as shown in Figure 5.2, and which can be listed in Table 5.2. Afterwards, it provides a general overview of which methods improved the performance overall, and examines any approaches that struggled to meet even the baseline performance.

	CoQA	SQA	TopiOCQA	QReCC	DoQA (avg)	QuAC	Doc2Dial	INSCIT
	F1							
Base	0.396	0.429	0.336	0.365	0.361	0.254	0.265	0.193
Context Summarization	0.277	0.255	0.339	0.361	0.217	0.189	0.203	0.209
Hybrid BM25	0.401	0.429	0.339	0.373	0.362	0.274	0.267	0.19
HyDE	0.278	0.256	0.34	0.361	0.216	0.189	0.202	0.208
HyDE Reranker	0.278	0.255	0.339	0.362	0.215	0.188	0.201	0.209
Query Rewriting	0.33	0.372	0.27	0.327	0.276	0.208	0.216	0.167
Reranker	0.427	0.45	0.344	0.362	0.368	0.293	0.284	0.191
Summarization	0.276	0.256	0.34	0.361	0.215	0.189	0.202	0.208
	MRR							
Base	0.059	0.07	0.051	0.191	0.35	0.101	0.08	0.055
Context Summarization	0.046	0.055	0.039	0.201	0.316	0.103	0.069	0.055
Hybrid BM25	0.108	0.059	0.051	0.193	0.365	0.128	0.086	0.056
HyDE	0.095	0.068	0.156	0.285	0.348	0.121	0.094	0.18
HyDE Reranker	0.037	0.059	0.101	0.212	0.327	0.122	0.088	0.113
Query Rewriting	0.06	0.075	0.047	0.193	0.355	0.102	0.075	0.06
Reranker	0.046	0.067	0.055	0.182	0.345	0.131	0.083	0.066
Summarization	0.051	0.058	0.036	0.197	0.313	0.106	0.071	0.054

Table 5.2: Table showing full F1 and MRR results for all datasets and RAG methods

The methods that improve upon the No RAG performance amongst all the datasets, excluding INSCIT, QReCC, and TopiOCQA, are Reranker, Hybrid BM25, Base, and Query Rewriting in the sequence presented. The approaches reached a similar level of performance even across datasets with varying F1 ranges, with Reranker consistently reaching a $40 - 45\%$ score. It was hoped that the CoQA and SQA datasets would have

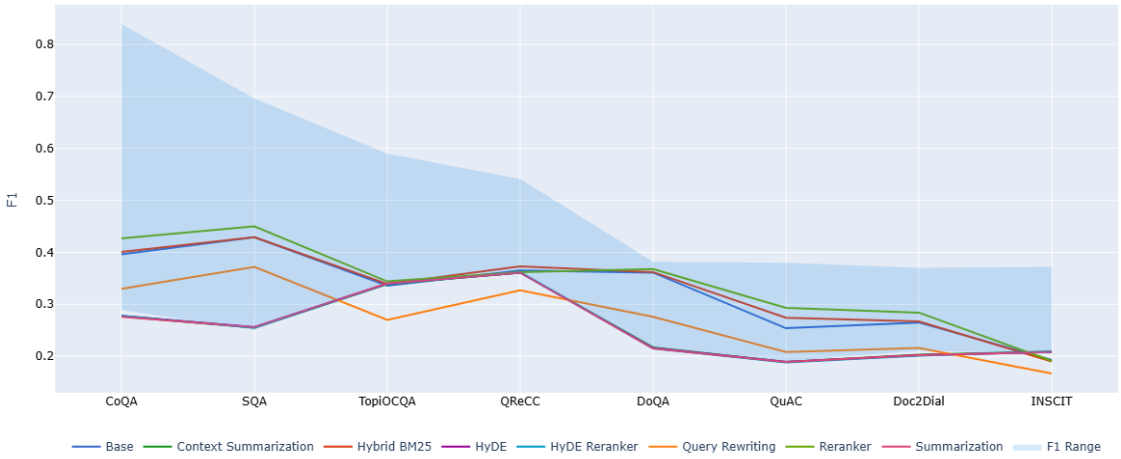


Figure 5.2: Visualization of the F1 performance of all RAG methods for each dataset, overlaid onto the F1 range calculated through the No RAG and Known Context methods.

the widest F1 range, would provide clearer canvases on which the methods could be more evenly dispersed, but this was proved otherwise.

Query Rewriting is a technique that relies on the Base implementation and alters it by editing the query to clarify any ambiguous references. On one hand, the efficiency of this method seems to be conditioned even more on the dataset than the others, as shown by its fluctuating performance with the INSCIT, QReCC, and TopiOCQA datasets, where it lies significantly below No RAG. The retrieval scores for Query Rewriting, as listed in Table 5.2, for these datasets show no deviation from the other values, which leads to the conclusion that the low performance is caused not by poor retrieval, but by the altered question deviating from the original.

The subsequent Section 5.4 discusses some of the points of interest shown here, such as the methods that have a high F1 score despite their inaccurate retrieval, and on the contrary, those that had a smoother retrieval process but lower F1 scores.

5.3.2 Association of F1 and MRR scores

The experiment analysis begins with an overview of the RAG methods performance on each of the datasets, firstly based on the F1 and MRR metrics. Figure 5.3 is used to show whether a link exists between these two metrics, and while the data point cluster on the bottom left supports the connection between poor retriever performance and inadequate answer generation, much of the other data shows that this connection is not as clearly defined. Additionally, the wide distribution of each RAG method demonstrates that there is no definite leading approach, and that the performances of each method are intrinsically linked with the datasets themselves.

At first glance, it can be seen the Reranker, Base, and Hybrid BM25 methods have among the highest F1 performance scores, which is notable since the latter two have relatively simple retrieval processes. Previously, the range of F1 scores per dataset was discussed, which showed that even with no context information the performance

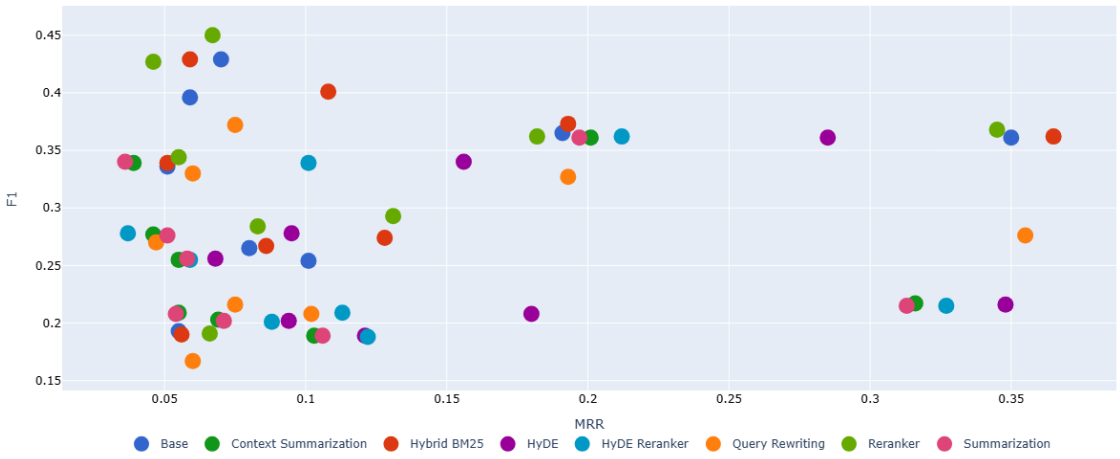


Figure 5.3: Scatter plot showing the relationship between F1 and MRR for each RAG method.

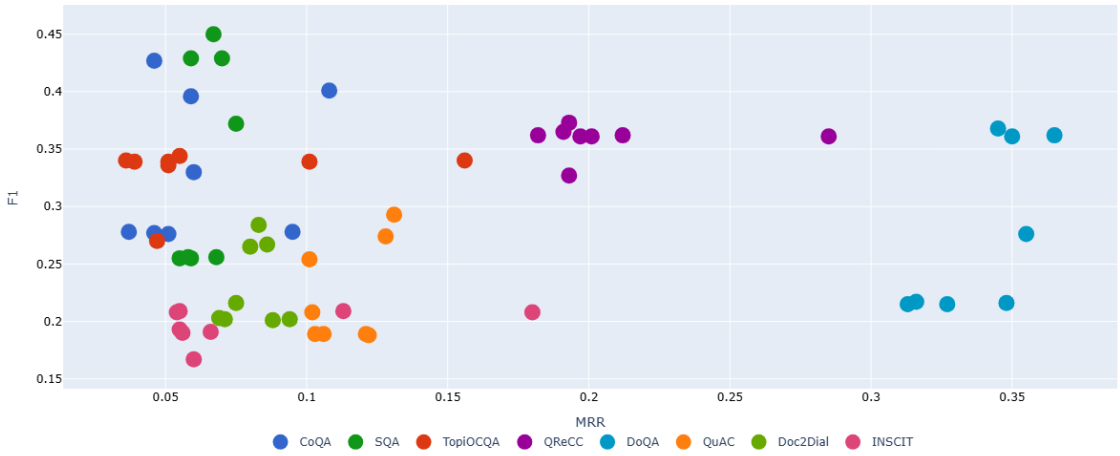


Figure 5.4: Scatter plot showing the relationship between F1 and MRR for each dataset.

never wavered below the 20% threshold, which does not seem to be the case for some of these methods. Thus, in some cases, inadequate retrieval could lead to even less favourable results than without any retrieval. Figure 5.3 also shows that the retriever performance as a whole was on the lower end, with the highest performing methods reaching 35% accurate ground-truth context retrieval.

To better understand these results, Figure 5.4 shows the same distribution, but this time separated by dataset, which provides more insight into the results. The point clusters here are easily identifiable, indicating that, in most cases, all the RAG methods yield relatively similar results regarding both F1 and MRR. It can be seen that the Reranker method seems to take the lead for F1, which could be caused by the Rerankers ability to prioritize context ranking, and similarly HyDE for MRR, since it provides additional contextual information.

5.4 Ablation studies

This section provides a deeper insight into the empirical results discussed above by first looking into the highest performing RAG methods, and analyzing in detail how each methods pipeline enhancements affect the performance. Then it continues by discussing a group of datasets that had poor performance across all RAG methods, and looking into whether it could have been caused by other factors such as conversation length. Lastly it ends on a short analysis into the two context summarization methods and the effect that their differences had on the performance.

5.4.1 Analysis of high-performing RAG methods

A look into the overall F1 and MRR values shown in Table 5.2, showed that the RAG methods, which showed the most accurate retrieval, did not necessarily always translate to the highest F1 values. The standout method in both the MRR and Recall retrieval metrics was HyDE, which, to restate, generates a document that would hypothetically answer the prompt question, which is then used to map to the ground truth contexts during retrieval. Evidently, this proves to be a successful approach, since it combines the supplementary information given by the contexts, while making sure it is tailor-made only to the latest question instead of the topic in general. The second best performer was Hybrid BM25, which integrates both dense and sparse retrieval, thus searching for both lexical and semantic matches. This shows that the contexts of some datasets have quite a considerable lexical overlap with the query and conversation history, and using only dense retrievers might lead to more generalized terms being searched and less priority given to those using the exact terminology.

Transitioning to the F1 values, it could be assumed that HyDE and Hybrid BM25 are among the highest performers, but that is only the case for the latter, which, on average, has the second-highest F1 performance across all datasets. On the other hand, the former achieved much lower scores, despite its high retrieval rate. This shows that Hybrid BM25 works as a good middle-ground between the two metrics, demonstrating the effectiveness of the sparse encoder in both precision and recall. The top performer in terms of answer quality across five out of the eight datasets was the Reranker method, which showed relatively moderate retrieval scores.

To take a closer look at why the Reranker could outperform HyDE, although its MRR scores are higher, Recall@k could also be analyzed, which shows whether the ground truth contexts appear in the top k scores, the full scores of which can be seen in Appendix B. Figure 5.5 shows a comparison between Recall@1 and Recall@5, and it can be seen that while Reranker has a lower chance of finding the correct context among all the retrieved ones, when it does find it, it is more likely to boost it up to the highest placement. Since each prompt has only one ground truth context, not much can be analyzed about the contexts that do not contain the correct answer but are topically related, but it is conceivable that Reranker might also place these higher, thus reducing the amount of unrelated contexts.

Figure 5.5: Plot showing the comparison of Recall values for the HyDE and Reranker methods, across the datasets where Reranker has the highest F1 performance. Recall@1 measures whether the correct context was retrieved in the first position whereas Recall@5 measures whether it occurs in the top 5 contexts.

5.4.2 Analysis of low-performing datasets

As could be seen in Figure 5.2, no RAG methods showed any improvement on the No RAG performance for the INSCIT, QReCC, and TopiOCQA datasets. An analysis was done into whether this could have been the effect of the number of conversation turns. Each prompt consists of the current question and the question-answer pairs that took place earlier in the conversation, which is sent to the LLM and used for context retrieval. A longer conversation history poses as a double-edged sword, since it can add much-needed contextual information, especially in the case of ambiguously phrased questions, but the slight change of topic between questions can lead to the attention being diverted from the main question. It would be expected that the datasets each have different levels as to when the drawbacks start to outweigh the benefits, and vice versa.

The analysis looked into the F1 and MRR results of these three datasets when using the Base RAG implementation, since it was planned to be used as a control group, and did surpassed the No RAG performance on the other datasets. Firstly, the F1 results of the datasets were examined, as seen in Figure 5.6, and as expected, in the case of QReCC and TopiOCQA, they steadily decrease as the conversation history increases. Even at the earliest points in the conversation history, the performance is still comparable to the No RAG performance, which indicates that the issue could lie with the context retrieval.

The performance of the retriever is shown in Figure 5.7, in which it can be seen that for QReCC and INSCIT, the MRR values drop sharply after the first couple of questions. This reveals that the larger conversation history does make it more difficult to fetch the correct retrieval, especially for these datasets that have a total of 22,000 and 30,000 contexts, respectively. This means that the distraction that the conversation history creates during the retrieval outweighs any benefits that this contextual information might provide. For TopiOCQA, the drop is less steep, but the consistent, low MRR is to be expected given the over 171,000 contexts, which would increase the likelihood of

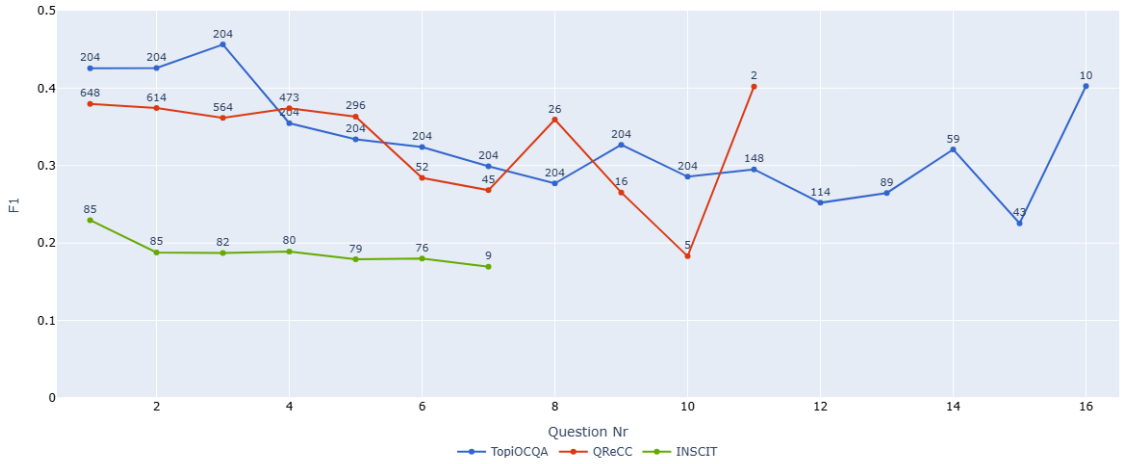


Figure 5.6: Plot showing how the Base RAG F1 performance of the TopiOCQA, QReCC, and INSCIT changes over each conversation turn. A conversation turn consists of the addition of a new question to the pre-existing conversation history.

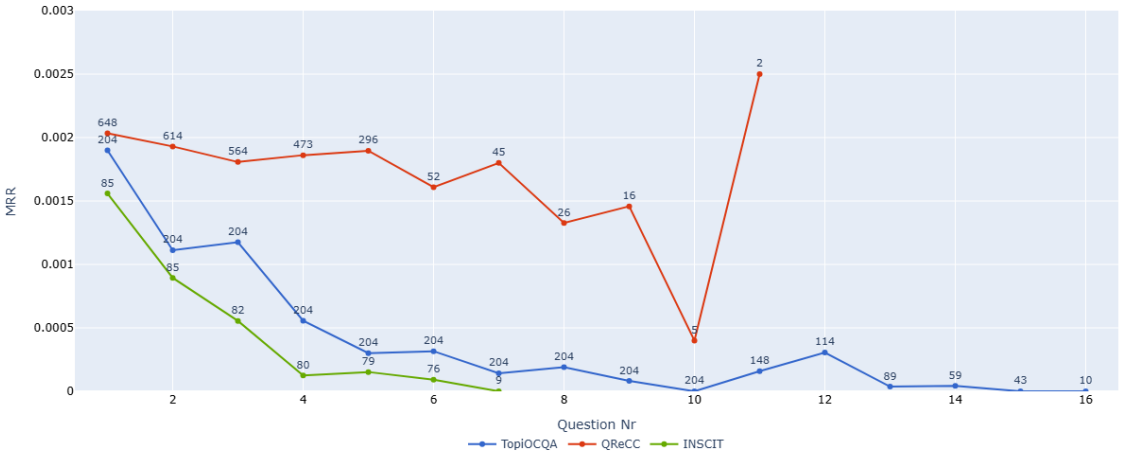


Figure 5.7: Graph showing how the MRR performance of the TopiOCQA, QReCC, and INSCIT changes over each conversation turn.

related but inaccurate contexts being fetched. In both figures, a small uptick in the F1 and MRR values is seen toward the higher end of the question amount, which could be attributed to anomaly cases, since they are far from the average conversation lengths.

5.4.3 Summarization versus Context Summarization

Subsection 4.5.9 discussed the design of the Summarization and Context Summarization methods, which begin similarly by making the LLM generate more concise contexts but later differ, with the former using the altered contexts in the final prompts and the latter using the original versions. The question arose as to how these two methods would perform with the datasets and whether the use of different contexts would affect

	CoQA	Doc2Dial	DoQA (avg)	INSCIT	QReCC	QuAC	SQA	TopiOCQA
Summarization	0.051	0.071	0.313	0.054	0.197	0.106	0.058	0.036
Context Summarization	0.046	0.069	0.316	0.055	0.201	0.103	0.055	0.039

Table 5.3: Table showing the difference of the MRR values between the Summarization and Context Summarization values.

	CoQA	Doc2Dial	DoQA (avg)	INSCIT	QReCC	QuAC	SQA	TopiOCQA
Summarization	0.276	0.202	0.215	0.208	0.361	0.189	0.256	0.34
Context Summarization	0.277	0.203	0.217	0.209	0.361	0.189	0.255	0.339

Table 5.4: Table showing the difference of the F1 values between the Summarization and Context Summarization values.

said performance. Table 5.3 shows the correctness of the retriever as measured with MRR, which, as expected, is quite similar, since both techniques use the same retrieval process. The values only differ by a few percent, which is well within the bounds of random error, and could be due to noise.

It is worth noting that even the F1 values were very similar between the two methods, as shown in Table 5.4, despite their differences. Through these results, the effect of the summarized or non-summarized context is unclear. One possibility is that both versions contribute a similar amount of contextual information, thus the negligible performance difference between the two. Another explanation could be that the fault lies in the retriever, which is supported by the relatively low MRR scores, and the fact that the F1 values resemble those of the No RAG method. Essentially, the LLM is not getting sufficient contextual information and is instead relying on its internal knowledge to answer the question.

The summarization process is handled by the LLM, which condenses the contexts, and while the prompt can be finetuned to a certain level, this process cannot be fully controlled. Therefore, while the contexts of some datasets may benefit by the removal of tangential information, this does not seem to be the case for the datasets looked at in this experiment.

6 Discussion

This chapter concludes this thesis by going over the research questions mentioned in Chapter 1 and discussing whether they have been sufficiently answered and including any limitations that were faced during the implementation in Section 6.1. Section 6.2 then continues by giving an overview of the main findings from the experiment and how this study might be used in future research.

6.1 Research questions

This section examines the research questions introduced at the beginning of the thesis, discusses how the experimental study attempted to answer them, and determines whether any conclusive findings were achieved.

- **R1: How do advanced RAG methods impact answer quality in comparison to a baseline Naive RAG?**

  As observed from the experiment analysis in the previous chapter, the results varied considerably across the datasets and RAG methods, making it difficult to form a unified finding. The top three methods in terms of F1 performance were Reranker, Hybrid BM25, and Base Implementation, respectively, showing that the Naive RAG is undoubtedly outperformed across all datasets by two advanced RAG methods. Therefore, in one respect, the two advanced methods can be recommended to future researchers as better performance alternatives.

  On the other hand, it is essential to look at what the advanced RAG methods add to the computing complexity and runtime in comparison to the base implementation. Reranker retrieves proportionally more contexts and then runs them through a cross-encoder to create relevancy scores. In contrast, Hybrid BM25 adds a sparse retrieval function and creates a hybrid dense/sparse relevancy score. All in all, while both methods do not add too much in terms of computing complexity,

observing the experiment runs shows that the double retrieval process and score calculation of Hybrid BM25 does lead to slightly more inflated runtimes.

In conclusion, while Naive RAG is not the best performer overall, its high F1 values and relatively easy implementation make it a worthwhile option to try, as it serves as a good baseline. Since the advanced methods do not stray far from the baseline, they would also provide a good indication as to how other methods might perform.

- **R2: How do dataset characteristics influence the performance and behavior of these RAG methods?**

  Even before the RAG methods were implemented, it was observed that the No RAG and Known Context methods demonstrated noticeable variations among the various datasets, which could be due to the models internal knowledge on specific topics or to the structure of the conversations and contexts. For example, the Doc2Dial dataset, which includes dialogue-style, open-ended questions that rely heavily on the conversation history, shows a very narrow F1 range unlike CoQA, which contains factoid or entity-based questions.

  The total number of contexts ultimately affected retriever performance by reducing the likelihood of choosing the correct context, but dataset size and amount were not necessarily positively correlated, as smaller datasets like TopiOCQA, with a size of around 2500, ended up having over $170,000$ contexts. Additionally, the effect of the number of conversation turns did not have any conclusive positive or negative effects on the F1 values, but it did significantly decrease the MRR, which could have canceled out any benefits of the extra conversation history.

  All in all, while specific RAG methods did perform well across most datasets, it is still essential that the datasets are analyzed in advance if optimal F1 or MRR values are sought. While the dataset characteristics had more of an impact on the retriever performance and MRR, the conversation histories and contexts especially may heavily affect the generator and F1 values in ways that are more difficult to observe at the offset. Thus, even if the dataset is studied beforehand, RAG methods might still have unexpected results, so a sampling of various methods might be the best approach.

6.1.1 Limitations and future work

This section discusses any obstacles or problems that hindered the execution of the experiment and how they could be solved or bypassed in future research. Firstly, the main problem was the wide variety of both RAG methods and datasets, which led to excessive time spent on pre-processing the data, as each dataset had its own prompt requirements, answer formats, and context structures. This also made in-depth analyses quite difficult to perform since all combinations of methods and datasets would have to be taken into account. In future research, this could be adjusted so that a smaller subsection of these components is chosen since there is some redundancy in the RAG method enhancements.

Another obstacle that was present during the retrieval process was the large amount of contexts, which often resulted in the ground truth context being difficult to fetch, especially in the TopiOCQA, QuAC, and INSCIT datasets. There are two main ways that this issue could be fixed: first, by grouping the contexts during pre-processing using their titles found in the metadata; second, by using other methods that aim to iterate over the retrieval or generation processes multiple times, thus increasing the likelihood of fetching the ground truth context and of generating a fitting answer.

6.2 Overall findings

In conclusion, based on the comparison between the No RAG and Known Context methods, it was found that adding the correct context to the prompt can improve the performance significantly by anywhere from 15% to 50%. Doing this analysis beforehand provided a good ceiling as to which values could be expected from the RAG methods, and the floor values were used to compare some of the lower-performing methods. Around half of the RAG methods had values comparable to No RAG, even with added contexts, showing that in most cases, an inefficient RAG pipeline works the same as no RAG at all. A bad RAG pipeline could be categorized by many features, such as retrieving contexts from various unrelated topics that could distract the LLM, or retrieving relevant contexts but not being able to push the ground truth context to a high enough position.

An analysis was also done into how methods that augment parts of the prompt affect performance. Query Rewriting, which aims to condense the conversation history and query had relatively good performance across five datasets, but in the other three it actually performed worse than No RAG, leading to the conclusion that this method is highly dataset specific, and the length and format of the prompts should be taken into consideration. On the other hand, the two Summarization methods, which aim to summarize the contexts by keeping the key data and removing tangential information, had quite poor performance across all datasets.

This performance disparity is challenging to investigate since the only way to do so is to modify the secondary prompt and observe how a sample of the data points are affected. While this can inform future decision making, a small sample may not fully represent the entire dataset, and there may be a lot of cases where prompt augmentation causes the dilution of important information or the misinterpretation of the original question leading to factual inaccuracies and model hallucination.

The Reranker and Hybrid BM25 methods achieved the two highest average performances in answer quality, showcasing the efficiency of both promoting relevant contexts to higher ranks and combining dense and sparse encoders. While these methods proved to be effective, the Naive RAG approach remained a competitive alternative. Overall, the key finding of this thesis is that advanced RAG methods can lead to significant performance improvements when data structures and formats are thoroughly analyzed, underscoring the importance of aligning retrieval strategies with dataset characteristics.

References

[1] OpenAI, GPT-4 technical report, *CoRR*, vol. abs/2303.08774, 2023. arXiv: 2303.08774.

[2] Hugo Touvron, Thibaut Lavril, Gautier Izacard, Xavier Martinet, Marie-Anne Lachaux, Timothée Lacroix, Baptiste Rozière, Naman Goyal, Eric Hambro, Faisal Azhar, *et al.*, Llama: Open and efficient foundation language models, *arXiv preprint arXiv:2302.13971*, 2023.

[3] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, Parker Schuh, Kensen Shi, *et al.*, Palm: Scaling language modeling with pathways, *J. Mach. Learn. Res.*, vol. 24, 240:1–240:113, 2023.

[4] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, *et al.*, Instruction tuning for large language models: A survey, *arXiv preprint arXiv:2308.10792*, 2023.

[5] Haolin Jin, Linghan Huang, Haipeng Cai, Jun Yan, Bo Li, and Huaming Chen, From llms to llm-based agents for software engineering: A survey of current, challenges and future, *arXiv preprint arXiv:2408.02479*, 2024.

[6] Enkelejda Kasneci, Kathrin Seßler, Stefan Küchemann, Maria Bannert, Daryna Dementieva, Frank Fischer, Urs Gasser, Georg Groh, Stephan Günnemann, Eyke Hüllermeier, *et al.*, Chatgpt for good? on opportunities and challenges of large language models for education, *Learning and individual differences*, vol. 103, p. 102 274, 2023.

[7] Anas Alhur, Redefining healthcare with artificial intelligence (AI): The contributions of ChatGPT, gemini, and co-pilot, *Cureus*, Apr. 7, 2024.

[8] Seyed Mahed Mousavi, Simone Alghisi, and Giuseppe Riccardi, Dyknow: Dynamically verifying time-sensitive factual knowledge in llms, in *Findings of the Association for Computational Linguistics: EMNLP 2024, Miami, Florida, USA, November 12-16, 2024*, Yaser Al-Onaizan, Mohit Bansal, and Yun-Nung Chen, Eds., Association for Computational Linguistics, 2024, pp. 8014–8029.

[9] Soumen Pal, Manojit Bhattacharya, Sang-Soo Lee, and Chiranjib Chakraborty, A domain-specific next-generation large language model (LLM) or ChatGPT is required for biomedical engineering and research, *Annals of Biomedical Engineering*, vol. 52, no. 3, pp. 451–454, Mar. 2024.

[10] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen-tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela, Retrieval-augmented generation for knowledge-intensive NLP tasks, in *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, Eds., 2020.

[11] D.M. Anisuzzaman, Jeffrey G. Malins, Paul A. Friedman, and Zachi I. Attia, Fine-tuning large language models for specialized use cases, *Mayo Clinic Proceedings: Digital Health*, vol. 3, no. 1, p. 100 184, Mar. 2025.

[12] Wenqi Fan, Yujuan Ding, Liangbo Ning, Shijie Wang, Hengyun Li, Dawei Yin, Tat-Seng Chua, and Qing Li, A survey on RAG meeting llms: Towards retrieval-augmented large language models, in *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining, KDD 2024, Barcelona, Spain, August 25-29, 2024*, Ricardo Baeza-Yates and Francesco Bonchi, Eds., ACM, 2024, pp. 6491–6501.

[13] Jintao Zhang, Guoliang Li, and Jinyang Su, Sage: A framework of precise retrieval for rag, *arXiv preprint arXiv:2503.01713*, 2025.

[14] Freda Shi, Xinyun Chen, Kanishka Misra, Nathan Scales, David Dohan, Ed H Chi, Nathanael Schärli, and Denny Zhou, Large language models can be easily distracted by irrelevant context, in *International Conference on Machine Learning*, PMLR, 2023, pp. 31 210–31 227.

[15] Ziwei Xu, Sanjay Jain, and Mohan S. Kankanhalli, Hallucination is inevitable: An innate limitation of large language models, *CoRR*, vol. abs/2401.11817, 2024.

[16] Yunfan Gao, Yun Xiong, Meng Wang, and Haofen Wang, Modular rag: Transforming rag systems into lego-like reconfigurable frameworks, *arXiv preprint arXiv:2407.21059*, 2024.

[17] William Nash Locke and Andrew Donald Booth, Machine translation of languages: Fourteen essays, *(No Title)*, 1955.

[18] Warren Weaver, Translation, in *Proceedings of the conference on mechanical translation*, 1952.

[19] W. John Hutchins, The georgetown-ibm experiment demonstrated in january 1954, in *Machine Translation: From Real Users to Research, 6th Conference of the Association for Machine Translation in the Americas, AMTA 2004, Washington, DC, USA, September 28-October 2, 2004, Proceedings*, Robert E. Frederking and Kathryn Taylor, Eds., ser. Lecture Notes in Computer Science, vol. 3265, Springer, 2004, pp. 102–114.

[20] Noam Chomsky, Three models for the description of language, *IRE Transactions on information theory*, vol. 2, no. 3, pp. 113–124, 1956.

[21] Noam Chomsky, Syntactic Structures. De Gruyter Mouton, 1957.

[22] Terry Winograd, Procedures as a Representation for Data in a Computer Program for Understanding Natural Language (AI-TR). M.I.T. Project MAC, 1971.

[23] William A Woods, Progress in natural language understanding: An application to lunar geology, in *Proceedings of the June 4-8, 1973, national computer conference and exposition*, 1973, pp. 441–450.

[24] William A Woods, Transition network grammars for natural language analysis, *Communications of the ACM*, vol. 13, no. 10, pp. 591–606, 1970.

[25] Roger C Schank, Neil M Goldman, Charles J Rieger III, and Christopher Riesbeck, Margie: Memory analysis response generation, and inference on english. In *IJCAI*, vol. 3, 1973, pp. 255–61.

[26] Robert Wilensky, Why john married mary: Understanding stories involving recurring goals, *Cognitive Science*, vol. 2, no. 3, pp. 235–266, 1978.

[27] James R Meehan, Tale-spin, an interactive program that writes stories. In *Ijcai*, vol. 77, 1977, pp. 91–98.

[28] Kenneth Mark Colby, Franklin Dennis Hilf, Sylvia Weber, and Helena C Kraemer, Turing-like indistinguishability tests for the validation of a computer simulation of paranoid processes, *Artificial Intelligence*, vol. 3, pp. 199–221, 1972.

[29] Languages ALPAC, Machines: Computers in translation and linguistics. a report by the automatic language processing advisory committee, division of behavioral sciences, *National Academy of Sciences, National Research Council, Publication*, vol. 1416, 1966.

[30] John Hutchins, Alpac: The (in) famous report, *Readings in machine translation*, vol. 14, pp. 131–135, 2003.

[31] John McCarthy, Professor sir james lighthill, FRS. artificial intelligence: A general survey, *Artif. Intell.*, vol. 5, no. 3, pp. 317–322, 1974.

[32] Mitchell Marcus, New trends in natural language processing: Statistical natural language processing. *Proceedings of the National Academy of Sciences*, vol. 92, no. 22, pp. 10 052–10 059, 1995.

[33] Henry Kučera, W. Nelson (Winthrop Nelson) Francis, W. F. (William Freeman) Twaddell, Mary Lois Marckworth, Laura M. Bell, and John Bissell Carroll, Computational analysis of present-day American English. Brown University Press, 1967.

[34] G Leech, S Johansson, K Hofland, R Garside, E Atwell, MC Jhar, and I Marshall, Lancaster-oslo/bergen corpus, 1961.

[35] Brian MacWhinney and Catherine Snow, The child language data exchange system: An update, *Journal of child language*, vol. 17, no. 2, pp. 457–472, 1990.

[36] Christopher Manning and Hinrich Schutze, Foundations of statistical natural language processing. MIT press, 1999.

[37] Peter F Brown, Vincent J Della Pietra, Peter V Desouza, Jennifer C Lai, and Robert L Mercer, Class-based n-gram models of natural language, *Computational linguistics*, vol. 18, no. 4, pp. 467–480, 1992.

[38] Chih-piao Wu, Loke Soo Hsu, and Chew L Tan, A survey on statistical approaches to natural language processing. Citeseer, 1992.

[39] Jinming Zou, Yi Han, and Sung-Sau So, Overview of artificial neural networks, *Artificial neural networks: methods and applications*, pp. 14–22, 2009.

[40] Yoshua Bengio, Réjean Ducharme, Pascal Vincent, and Christian Jauvin, A neural probabilistic language model, *Journal of machine learning research*, vol. 3, no. Feb, pp. 1137–1155, 2003.

[41] Seunghak Yu, Sathish Reddy Indurthi, Seohyun Back, and Haejun Lee, A multi-stage memory augmented neural network for machine reading comprehension, in *Proceedings of the workshop on machine reading for question answering*, 2018, pp. 21–30.

[42] Adam Santoro, Ryan Faulkner, David Raposo, Jack Rae, Mike Chrzanowski, Theophane Weber, Daan Wierstra, Oriol Vinyals, Razvan Pascanu, and Timothy Lillicrap, Relational recurrent neural networks, *Advances in neural information processing systems*, vol. 31, 2018.

[43] Richard Socher, Alex Perelygin, Jean Wu, Jason Chuang, Christopher D Manning, Andrew Y Ng, and Christopher Potts, Recursive deep models for semantic compositionality over a sentiment treebank, in *Proceedings of the 2013 conference on empirical methods in natural language processing*, 2013, pp. 1631–1642.

[44] Jeffrey L Elman, Finding structure in time, *Cognitive science*, vol. 14, no. 2, pp. 179–211, 1990.

[45] Sepp Hochreiter, Yoshua Bengio, Paolo Frasconi, Jürgen Schmidhuber, *et al.*, Gradient flow in recurrent nets: The difficulty of learning long-term dependencies, 2001.

[46] Sepp Hochreiter and Jürgen Schmidhuber, Long short-term memory, *Neural computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[47] Felix A Gers, Jürgen Schmidhuber, and Fred Cummins, Learning to forget: Continual prediction with lstm, *Neural computation*, vol. 12, no. 10, pp. 2451–2471, 2000.

[48] Felix A Gers and Jürgen Schmidhuber, Recurrent nets that time and count, in *Proceedings of the IEEE-INNS-ENNS International Joint Conference on Neural Networks. IJCNN 2000. Neural Computing: New Challenges and Perspectives for the New Millennium*, IEEE, vol. 3, 2000, pp. 189–194.

[49] Klaus Greff, Rupesh K Srivastava, Jan Koutník, Bas R Steunebrink, and Jürgen Schmidhuber, Lstm: A search space odyssey, *IEEE transactions on neural networks and learning systems*, vol. 28, no. 10, pp. 2222–2232, 2016.

[50] Tomás Mikolov, Kai Chen, Greg Corrado, and Jeffrey Dean, Efficient estimation of word representations in vector space, in *1st International Conference on Learning Representations, ICLR 2013, Scottsdale, Arizona, USA, May 2-4, 2013, Workshop Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2013.

[51] Dzmitry Bahdanau, Kyunghyun Cho, and Yoshua Bengio, Neural machine translation by jointly learning to align and translate, in *3rd International Conference on Learning Representations, ICLR 2015, San Diego, CA, USA, May 7-9, 2015, Conference Track Proceedings*, Yoshua Bengio and Yann LeCun, Eds., 2015.

[52] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin, Attention is all you need, *Advances in neural information processing systems*, vol. 30, 2017.

[53] Hinrich Schütze, Christopher D Manning, and Prabhakar Raghavan, Introduction to information retrieval. Cambridge University Press Cambridge, 2008, vol. 39.

[54] Sabrina J Mielke, Zaid Alyafeai, Elizabeth Salesky, Colin Raffel, Manan Dey, Matthias Gallé, Arun Raja, Chenglei Si, Wilson Y Lee, Benoît Sagot, *et al.*, Between words and characters: A brief history of open-vocabulary modeling and tokenization in nlp, *arXiv preprint arXiv:2112.10508*, 2021.

[55] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun, Deep residual learning for image recognition, in *Proceedings of the IEEE conference on computer vision and pattern recognition*, 2016, pp. 770–778.

[56] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton, Layer normalization, *arXiv preprint arXiv:1607.06450*, 2016.

[57] Shashank Mohan Jain, Introduction to transformers for nlp, *With the Hugging Face Library and Models to Solve Problems*, 2022.

[58] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova, Bert: Pre-training of deep bidirectional transformers for language understanding, in *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 2019, pp. 4171–4186.

[59] 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, *arXiv preprint arXiv:1907.11692*, 2019.

[60] Victor Sanh, Lysandre Debut, Julien Chaumond, and Thomas Wolf, Distilbert, a distilled version of bert: Smaller, faster, cheaper and lighter, *arXiv preprint arXiv:1910.01108*, 2019.

[61] Geoffrey Hinton, Oriol Vinyals, and Jeff Dean, Distilling the knowledge in a neural network, *arXiv preprint arXiv:1503.02531*, 2015.

[62] Kevin Clark, Minh-Thang Luong, Quoc V. Le, and Christopher D. Manning, ELECTRA: pre-training text encoders as discriminators rather than generators, in *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*, OpenReview.net, 2020.

[63] Pengcheng He, Xiaodong Liu, Jianfeng Gao, and Weizhu Chen, Deberta: Decoding-enhanced bert with disentangled attention, in *9th International Conference on Learning Representations, ICLR 2021, Virtual Event, Austria, May 3-7, 2021*, OpenReview.net, 2021.

[64] Alec Radford, Karthik Narasimhan, Tim Salimans, Ilya Sutskever, *et al.*, Improving language understanding by generative pre-training, 2018.

[65] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer, BART: denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension, in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics, ACL 2020, Online, July 5-10, 2020*, Dan Jurafsky, Joyce Chai, Natalie Schluter, and Joel R. Tetreault, Eds., Association for Computational Linguistics, 2020, pp. 7871–7880.

[66] 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, *Journal of machine learning research*, vol. 21, no. 140, pp. 1–67, 2020.

[67] Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, Ilya Sutskever, *et al.*, Language models are unsupervised multitask learners, *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[68] Rico Sennrich, Barry Haddow, and Alexandra Birch, Neural machine translation of rare words with subword units, in *Proceedings of the 54th Annual Meeting of the Association for Computational Linguistics, ACL 2016, August 7-12, 2016, Berlin, Germany, Volume 1: Long Papers*, The Association for Computer Linguistics, 2016.

[69] Saidul Islam, Hanae Elmekki, Ahmed Elsebai, Jamal Bentahar, Nagat Drawel, Gaith Rjoub, and Witold Pedrycz, A comprehensive survey on applications of transformers for deep learning tasks, *Expert Systems with Applications*, vol. 241, p. 122 666, 2024.

[70] Shervin Minaee, Tomas Mikolov, Narjes Nikzad, Meysam Chenaghlu, Richard Socher, Xavier Amatriain, and Jianfeng Gao, Large language models: A survey. arxiv 2024, *arXiv preprint arXiv:2402.06196*,

[71] Aaron Grattafiori, Abhimanyu Dubey, Abhinav Jauhri, Abhinav Pandey, Abhishek Kadian, Ahmad Al-Dahle, Aiesha Letman, Akhil Mathur, Alan Schelten, Alex Vaughan, *et al.*, The llama 3 herd of models, *arXiv preprint arXiv:2407.21783*, 2024.

[72] Xiang Lisa Li and Percy Liang, Prefix-tuning: Optimizing continuous prompts for generation, in *Proceedings of the 59th Annual Meeting of the Association for Computational Linguistics and the 11th International Joint Conference on Natural Language Processing, ACL/IJCNLP 2021, (Volume 1: Long Papers), Virtual Event, August 1-6, 2021*, Chengqing Zong, Fei Xia, Wenjie Li, and Roberto Navigli, Eds., Association for Computational Linguistics, 2021, pp. 4582–4597.

[73] Linting Xue, Noah Constant, Adam Roberts, Mihir Kale, Rami Al-Rfou, Aditya Siddhant, Aditya Barua, and Colin Raffel, Mt5: A massively multilingual pre-trained text-to-text transformer, in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2021, Online, June 6-11, 2021*, Kristina Toutanova, Anna Rumshisky, Luke Zettlemoyer, Dilek Hakkani-Tür, Iz Beltagy, Steven Bethard, Ryan Cotterell, Tanmoy Chakraborty, and Yichao Zhou, Eds., Association for Computational Linguistics, 2021, pp. 483–498.

[74] Minghao Shao, Abdul Basit, Ramesh Karri, and Muhammad Shafique, Survey of different large language model architectures: Trends, benchmarks, and challenges, *IEEE Access*, 2024.

[75] Humza Naveed, Asad Ullah Khan, Shi Qiu, Muhammad Saqib, Saeed Anwar, Muhammad Usman, Naveed Akhtar, Nick Barnes, and Ajmal Mian, A comprbehensive overview of large language models, *arXiv preprint arXiv:2307.06435*, 2023.

[76] Zaid Alyafeai, Maged Saeed AlShaibani, and Irfan Ahmad, A survey on transfer learning in natural language processing, *arXiv preprint arXiv:2007.04239*, 2020.

[77] Tianhao Shen, Renren Jin, Yufei Huang, Chuang Liu, Weilong Dong, Zishan Guo, Xinwei Wu, Yan Liu, and Deyi Xiong, Large language model alignment: A survey, *arXiv preprint arXiv:2309.15025*, 2023.

[78] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, *et al.*, Language models are few-shot learners, *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.

[79] Laura Weidinger, John Mellor, Maribeth Rauh, Conor Griffin, Jonathan Uesato, Po-Sen Huang, Myra Cheng, Mia Glaese, Borja Balle, Atoosa Kasirzadeh, *et al.*, Ethical and social risks of harm from language models, *arXiv preprint arXiv:2112.04359*, 2021.

[80] Zachary Kenton, Tom Everitt, Laura Weidinger, Iason Gabriel, Vladimir Mikulik, and Geoffrey Irving, Alignment of language agents, *arXiv preprint arXiv:2103.14659*, 2021.

[81] Long Ouyang, Jeffrey Wu, Xu Jiang, Diogo Almeida, Carroll Wainwright, Pamela Mishkin, Chong Zhang, Sandhini Agarwal, Katarina Slama, Alex Ray, *et al.*, Training language models to follow instructions with human feedback, *Advances in neural information processing systems*, vol. 35, pp. 27 730–27 744, 2022.

[82] Krystal Hu, Chatgpt sets record for fastest-growing user base - analyst note, 2023.

[83] Yuntao Bai, Saurav Kadavath, Sandipan Kundu, Amanda Askell, Jackson Kernion, Andy Jones, Anna Chen, Anna Goldie, Azalia Mirhoseini, Cameron McKinnon, *et al.*, Constitutional ai: Harmlessness from ai feedback, *arXiv preprint arXiv:2212.08073*, 2022.

[84] Gemini Team, Rohan Anil, Sebastian Borgeaud, Jean-Baptiste Alayrac, Jiahui Yu, Radu Soricut, Johan Schalkwyk, Andrew M Dai, Anja Hauth, Katie Millican, *et al.*, Gemini: A family of highly capable multimodal models, *arXiv preprint arXiv:2312.11805*, 2023.

[85] Jeffrey Cheng, Marc Marone, Orion Weller, Dawn Lawrie, Daniel Khashabi, and Benjamin Van Durme, Dated data: Tracing knowledge cutoffs in large language models, *arXiv preprint arXiv:2403.12958*, 2024.

[86] Pragya Shukla, Rag (retrieval-augmented generation) for your own documents, Mar. 2024.

[87] Chi-Min Chan, Chunpu Xu, Ruibin Yuan, Hongyin Luo, Wei Xue, Yike Guo, and Jie Fu, Rq-rag: Learning to refine queries for retrieval augmented generation, *arXiv preprint arXiv:2404.00610*, 2024.

[88] Hongyin Luo, Tianhua Zhang, Yung-Sung Chuang, Yuan Gong, Yoon Kim, Xixin Wu, Helen Meng, and James Glass, Search augmented instruction learning, in *Findings of the Association for Computational Linguistics: EMNLP 2023*, 2023, pp. 3717–3729.

[89] Akari Asai, Zeqiu Wu, Yizhong Wang, Avirup Sil, and Hannaneh Hajishirzi, Self-rag: Learning to retrieve, generate, and critique through self-reflection, in *The Twelfth International Conference on Learning Representations*, 2023.

[90] Darren Edge, Ha Trinh, Newman Cheng, Joshua Bradley, Alex Chao, Apurva Mody, Steven Truitt, Dasha Metropolitansky, Robert Osazuwa Ness, and Jonathan Larson, From local to global: A graph rag approach to query-focused summarization, *arXiv preprint arXiv:2404.16130*, 2024.

[91] Soyeong Jeong, Jinheon Baek, Sukmin Cho, Sung Ju Hwang, and Jong Park, Adaptive-rag: Learning to adapt retrieval-augmented large language models through question complexity, Kevin Duh, Helena Gómez-Adorno, and Steven Bethard, Eds., pp. 7036–7050, 2024.

[92] Pranav Rajpurkar, Jian Zhang, Konstantin Lopyrev, and Percy Liang, Squad: 100, 000+ questions for machine comprehension of text, in *Proceedings of the 2016 Conference on Empirical Methods in Natural Language Processing, EMNLP 2016, Austin, Texas, USA, November 1-4, 2016*, Jian Su, Xavier Carreras, and Kevin Duh, Eds., The Association for Computational Linguistics, 2016, pp. 2383–2392.

[93] Mandar Joshi, Eunsol Choi, Daniel S. Weld, and Luke Zettlemoyer, Triviaqa: A large scale distantly supervised challenge dataset for reading comprehension, Regina Barzilay and Min-Yen Kan, Eds., pp. 1601–1611, 2017.

[94] Alex Mallen, Akari Asai, Victor Zhong, Rajarshi Das, Daniel Khashabi, and Hannaneh Hajishirzi, When not to trust language models: Investigating effectiveness of parametric and non-parametric memories, in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2023, Toronto, Canada, July 9-14, 2023*, Anna Rogers, Jordan L. Boyd-Graber, and Naoaki Okazaki, Eds., Association for Computational Linguistics, 2023, pp. 9802–9822.

[95] Xanh Ho, Anh-Khoa Duong Nguyen, Saku Sugawara, and Akiko Aizawa, Constructing A multi-hop QA dataset for comprehensive evaluation of reasoning steps, in *Proceedings of the 28th International Conference on Computational Linguistics, COLING 2020, Barcelona, Spain (Online), December 8-13, 2020*, Donia Scott, Núria Bel, and Chengqing Zong, Eds., International Committee on Computational Linguistics, 2020, pp. 6609–6625.

[96] 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, in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing, Brussels, Belgium, October 31 - November 4, 2018*, Ellen Riloff, David Chiang, Julia Hockenmaier, and Jun'ichi Tsujii, Eds., Association for Computational Linguistics, 2018, pp. 2369–2380.

[97] Stephanie Lin, Jacob Hilton, and Owain Evans, Truthfulqa: Measuring how models mimic human falsehoods, in *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), ACL 2022, Dublin, Ireland, May 22-27, 2022*, Smaranda Muresan, Preslav Nakov, and Aline Villavicencio, Eds., Association for Computational Linguistics, 2022, pp. 3214–3252.

[98] Matouš Eibich, Shivay Nagpal, and Alexander Fred-Ojala, Aragog: Advanced rag output grading, *arXiv preprint arXiv:2404.01037*, 2024.

[99] Mohit Iyyer, Wen-tau Yih, and Ming-Wei Chang, Search-based neural structured learning for sequential question answering, in *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics*, 2017.

[100] Eunsol Choi, He He, Mohit Iyyer, Mark Yatskar, Wen-tau Yih, Yejin Choi, Percy Liang, and Luke Zettlemoyer, Quac: Question answering in context, in *Proceedings of the 2018 Conference on EMNLP*, 2018.

[101] Siva Reddy, Danqi Chen, and Christopher D Manning, Coqa: A conversational question answering challenge, *Transactions of the Association for Computational Linguistics*, 2019.

[102] Jon Ander Campos, Arantxa Otegi, Aitor Soroa, Jan Milan Deriu, Mark Cieliebak, and Eneko Agirre, Doqa-accessing domain-specific faqs via conversational qa, in *Proceedings of the 2020 Conference on ACL*, 2020.

[103] Song Feng, Hui Wan, Chulaka Gunasekara, Siva Patel, Sachindra Joshi, and Luis Lastras, Doc2dial: A goal-oriented document-grounded dialogue dataset, in *Proceedings of the 2020 Conference on EMNLP*, 2020.

[104] Raviteja Anantha, Svitlana Vakulenko, Zhucheng Tu, Shayne Longpre, Stephen Pulman, and Srinivas Chappidi, Open-domain question answering goes conversational via question rewriting, in *Proceedings of the 2021 Conference on NAACL*, 2021.

[105] Vaibhav Adlakha, Shehzaad Dhuliawala, Kaheer Suleman, Harm de Vries, and Siva Reddy, Topiocqa: Open-domain conversational question answering with topic switching, *Transactions of the Association for Computational Linguistics*, 2022.

[106] Zeqiu Wu, Ryu Parish, Hao Cheng, Sewon Min, Prithviraj Ammanabrolu, Mari Ostendorf, and Hannaneh Hajishirzi, Inscit: Information-seeking conversations with mixed-initiative interactions, *Transactions of the Association for Computational Linguistics*, 2023.

[107] Woosuk Kwon, Zhuohan Li, Siyuan Zhuang, Ying Sheng, Lianmin Zheng, Cody Hao Yu, Joseph E. Gonzalez, Hao Zhang, and Ion Stoica, Efficient memory management for large language model serving with pagedattention, in *Proceedings of the ACM SIGOPS 29th Symposium on Operating Systems Principles*, 2023.

[108] Matei A. Zaharia, Andrew Chen, Aaron Davidson, Ali Ghodsi, Sue Ann Hong, Andy Konwinski, Siddharth Murching, Tomas Nykodym, Paul Ogilvie, Mani Parkhe, Fen Xie, and Corey Zumar, Accelerating the Machine Learning Lifecycle with MLflow, *IEEE Data Eng. Bull.*, vol. 41, pp. 39–45, 2018.

[109] Chroma, Chroma - the open-source embedding database. https://github.com/chroma-core/chroma, 2022.

[110] Nils Reimers and Iryna Gurevych, Sentence-bert: Sentence embeddings using siamese bert-networks, in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019.

[111] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu, Bleu: A method for automatic evaluation of machine translation, in *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, 2002, pp. 311–318.

[112] Courtney Napoles, Keisuke Sakaguchi, Matt Post, and Joel Tetreault, Gleu without tuning, *arXiv preprint arXiv:1605.02592*, 2016.

[113] Chin-Yew Lin, Rouge: A package for automatic evaluation of summaries, in *Text summarization branches out*, 2004, pp. 74–81.

[114] Zihan Liu, Wei Ping, Rajarshi Roy, Peng Xu, Chankyu Lee, Mohammad Shoeybi, and Bryan Catanzaro, Chatqa: Surpassing gpt-4 on conversational qa and rag, *arXiv preprint arXiv:2401.10225*, 2024.

[115] Paul B. Kantor and Ellen M. Voorhees, The TREC-5 confusion track: Comparing retrieval methods for scanned text, *Inf. Retr.*, vol. 2, no. 2/3, pp. 165–176, 2000.

[116] Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan, Precise zero-shot dense retrieval without relevance labels, in *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, Anna Rogers, Jordan Boyd-Graber, and Naoaki Okazaki, Eds., Association for Computational Linguistics, Jul. 2023, pp. 1762–1777.

[117] Stephen E. Robertson, Steve Walker, and Micheline Hancock-Beaulieu, Experimentation as a way of life: Okapi at TREC, *Inf. Process. Manag.*, vol. 36, no. 1, pp. 95–108, 2000.

[118] Rodrigo Nogueira and Kyunghyun Cho, Passage re-ranking with BERT, *CoRR*, vol. abs/1901.04085, 2019. arXiv: 1901.04085.

[119] Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, and Dan Bikel, Llama 2: Open foundation and fine-tuned chat models, *CoRR*, vol. abs/2307.09288, 2023.

[120] Joshua Ainslie, James Lee-Thorp, Michiel de Jong, Yury Zemlyanskiy, Federico Lebrón, and Sumit Sanghai, GQA: training generalized multi-query transformer models from multi-head checkpoints, in *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, Houda Bouamor, Juan Pino, and Kalika Bali, Eds., Association for Computational Linguistics, 2023, pp. 4895–4901.

[121] NVIDIA Corporation, Nvidia rtx a6000 datasheet, 2021.

Appendices

A

Expanded prompts

CoQA: You are a helpful assistant who will try to answer the following question to the best of your abilities. Use only on the given context and conversation history and do not use any assumptions or external information. Make the answers as direct as possible without using any redundant information and without using full sentences. Indicate if you cannot find the answer based on the context.

Doc2Dial: You are a helpful assistant. Answer the question strictly based on the given context. Do not use prior knowledge, make assumptions, or introduce any information not present in the context. If the answer is clearly stated, respond in a complete and concise sentence. If the context does not provide enough information, respond with a relevant follow-up question to clarify the user's intent.

CoQA: You are a helpful assistant who will try to answer the following question to the best of your abilities. Use only on the given context and conversation history and do not use any assumptions or external information. Make the answers as direct as possible without using any redundant information and without using full sentences. Indicate if you cannot find the answer based on the context.

Doc2Dial: You are a helpful assistant. Answer the question strictly based on the given context. Do not use prior knowledge, make assumptions, or introduce any information not present in the context. If the answer is clearly stated, respond in a complete and concise sentence. If the context does not provide enough information, respond with a relevant follow-up question to clarify the user's intent.

DoQA: You are a helpful assistant trying to answer the questions to the best of your abilities. Use only the given context to answer the question. and do not use any assumptions or external information. Keep your answer relevant, direct and in one sentence. Do not explain the background, context or reasoning behind the answer. Do not refer to the context in your response. Indicate if you cannot find the answer based on the context.

INSCIT: You are a helpful assistant. Answer the question strictly based on the given context. Do not use prior knowledge, make assumptions, or include any information not present in the context. Do not refer to the context in your response. If the answer is not available, say so clearly. Respond in one full and complete sentence.

QReCC: You are a helpful assistant. Answer the question strictly based on the given context. Do not use prior knowledge, make assumptions, or introduce any information not present in the context. If the answer is not available, clearly state that. Respond in a single, clear, and complete sentence whenever possible.

QuAC: You are a helpful assistant who will try to answer the following question to the best of your abilities. Use only the given context and conversation history and do not use any assumptions or external information. Keep your answer short, direct and in one sentence. Do not explain the background, context or reasoning behind the answer. Indicate if you cannot find the answer based on the context.

SQA: You are a helpful assistant. Use only the given table and conversation history to answer the question. Do not rely on outside knowledge or make assumptions. Return the exact answer from the table. Use brief phrases or values and no full sentences.

TopiOCQA: You are a helpful assistant who will try to answer the following question to the best of your abilities. Use only the given context and conversation history and do not use any assumptions or external information. Make the answers as direct as possible without using any redundant information and without using full sentences. Indicate if you cannot find the answer based on the context.

Figure A.1: Showing a list of the system prompts used for the ChatRAG-Bench datasets. The prompts are designed to instruct the LLM on how best to answer the question and to emphasize the focus that should be placed on the previous conversation history and the contexts provided, and if they do not provide the answer then the LLM should indicate as such, instead of relying on internal information. A segment of the prompt is used to guide the LLM on how the answer should be formatted whether it be multiple sentences or short phrases.

B
Experiment results

	CoQA	SQA	TopiOCQA	QReCC	DoQA (avg)	QuAC	Doc2Dial	INSCIT
	Recall@1							
Base	0.025	0.029	0.019	0.079	0.163	0.032	0.032	0.016
Context Summarization	0.02	0.023	0.014	0.075	0.143	0.033	0.027	0.018
Hybrid BM25	0.055	0.027	0.016	0.074	0.165	0.039	0.036	0.02
HyDE	0.043	0.03	0.068	0.121	0.16	0.04	0.038	0.076
HyDE Reranker	0.015	0.024	0.048	0.1	0.149	0.05	0.037	0.05
Query Rewriting	0.027	0.032	0.015	0.082	0.16	0.031	0.03	0.024
Reranker	0.02	0.028	0.019	0.077	0.156	0.052	0.035	0.02
Summarization	0.021	0.024	0.013	0.072	0.141	0.036	0.027	0.02
	Recall@5							
Base	0.127	0.157	0.115	0.418	0.735	0.255	0.182	0.135
Context Summarization	0.101	0.12	0.089	0.453	0.665	0.26	0.159	0.129
Hybrid BM25	0.207	0.125	0.117	0.438	0.765	0.327	0.185	0.125
HyDE	0.208	0.15	0.317	0.587	0.731	0.298	0.211	0.359
HyDE Reranker	0.082	0.136	0.193	0.41	0.688	0.27	0.194	0.207
Query Rewriting	0.128	0.165	0.113	0.426	0.738	0.259	0.168	0.133
Reranker	0.101	0.149	0.121	0.381	0.732	0.298	0.181	0.139
Summarization	0.113	0.132	0.084	0.448	0.661	0.266	0.162	0.127

Table B.1: Table showing full Recall@1 and Recall@5 results for all datasets and RAG methods