Bauhaus-Universität Weimar
Faculty of Media
Degree Programme Digital Engineering

Retrieval Augmented Generation for the IR-Anthology

Master's Thesis

Islam Torky

1. Referee: Prof. Dr. Benno Stein
2. Referee: Prof. Dr. Volker Rodehorst

Submission date: March 22, 2024

Declaration

Unless otherwise indicated in the text or references, this thesis is entirely the product of my own scholarly work.

Weimar, Germany, March 22, 2024

. . . . . .
Islam Torky

Abstract

In recent years, Natural Language Processing (NLP) has seen a big leap forward with the emergence of pre-trained Large Language Models (LLMs). These models, like BERT, GPT-3, and Llama2, have excelled in various NLP tasks. While they've set new standards, they also face challenges, notably hallucinations, where they generate information that sounds plausible but is incorrect. Furthermore, they struggle with staying accurate and updated with new data called information cutoff. In addition to these two, they are general-purpose and not limited to a specific field; therefore, lack domain specificity.

Retrieval Augmented Generation (RAG) is a solution to these challenges aiming to combine the strengths of LLMs with external knowledge retrieval. RAG retrieves information during inference, reducing the risk of generating incorrect content and keeping information up-to-date. This thesis explores implementing RAG on the IR-Anthology, a vast collection of research papers on information retrieval. The goal is to make retrieval of information from the IR-Anthology more efficient, enabling easier access for researchers or students.

To set the stage, the thesis begins with an overview of recent NLP and RAG advancements, providing a foundation for understanding and introducing innovative methodologies using RAG. The thesis adopts a systematic approach, tailoring pipelines for the IR-Anthology dataset and evaluating their effectiveness in retrieval and generation stages. More specifically, this approach aims to not only assess overall efficacy but also understand variations in outputs across scenarios and data subsets.

The thesis explores how dividing documents into smaller segments (chunks) affects RAG pipelines, as well as different retrieval methods. The conducted experiments reveal that for retrieval of information from PDFs, larger chunks improve accuracy. However, for generating text, smaller chunks benefit the LLM by providing more focused information. Surprisingly, simpler retrieval methods outperform more complex ones.

Contents

List of Figures

List of Tables

Chapter 1

Introduction

In recent years, the field of natural language processing (NLP) has experienced a transformative shift with the advent of pre-trained large language models (LLM). These models, pre-trained on large corpora of text, have demonstrated remarkable performance across a wide spectrum of NLP tasks. LLMs, such as BERT (Devlin et al. [2019a]), GPT-3 (Patel et al. [2023]), and Llama2 (Touvron et al. [2023]), have achieved state-of-the-art results in tasks ranging from text classification to chat models.

While pre-trained LLMs have undeniably made significant strides in NLP, it is crucial to acknowledge their limitations. One notable challenge that LLMs face is the issue of hallucinations. Hallucinations refer to the generation of plausible-sounding but incorrect or misleading information in generated text. These models often rely on patterns in the training data and may produce responses that appear coherent but are factually incorrect or nonsensical. This is particularly problematic in applications where accuracy and reliability are paramount, such as medical diagnosis or legal document generation (Zhang et al. [2023]). Furthermore, LLMs tend to under perform when confronted with up-to-date information due to the information cutoff; once an LLM has been initially trained it can no longer process any new data without being finetuned. They are heavily reliant on the data distribution they were trained on, and their performance can degrade when applied to tasks involving recent developments. Another limitation of LLMs is being experts in a certain field, since they are heavily dependant on their data distribution and this data is not specific to a certain field they lack domain specificity. Understanding these limitations of LLMs is crucial when considering their application in practical scenarios. These shortcomings highlight the need for more specialized and robust approaches, such as Retrieval Augmented Generation (RAG), which aims to leverage the strengths of LLMs while addressing their weaknesses in handling factual accuracy, up-to-date information, and domain-specific knowl-

edge.

RAG, first introduced by Lewis et al. [2020] utilized a configuration where the parametric memory was implemented as a pre-trained sequence-to-sequence (seq2seq) encoder-decoder transformer, while the non-parametric memory consisted of a dense vector index of Wikipedia. This non-parametric memory was accessed using a pre-trained neural retriever, illustrating the incorporation of external knowledge retrieval in the RAG model. During inference, the models retrieve pertinent passages from Wikipedia, which are then employed in response generation. The high-level architecture of a standard RAG pipeline is depicted in Figure 1.1. This illustration outlines the operational flow wherein a user submits a query to the RAG pipeline. Subsequently, analogous information (referred to as a chunk) is retrieved from the designated knowledge base. This retrieved information is then fed into the generator (LLM), which generates a response characterized by its adherence to truth, up to date, and alignment with the user's domain. This capability enables RAG to access context specific to the query which has the potential to address the problem of hallucinations, information cutoff, and domain specificity (Siriwardhana et al. [2023]).

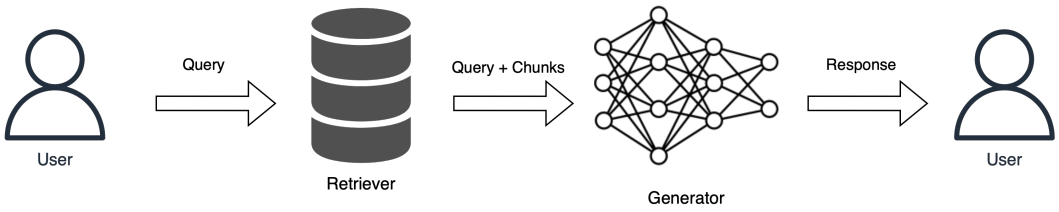

Figure 1.1: High Level RAG interaction between a knowledge base, LLM, and a user.

Lewis et al. [2020] emphasized that RAG exhibits strong performance in general question-answering datasets based on Wikipedia, such as Natural Questions (Kwiatkowski et al. [2019]). Recent research has also underscored that the generated outputs tend to be notably more factual. This is attributed to the pipeline being conditioned on the information retrieved from documents. Shuster et al. [2021] further draw attention to how effectively it reduces the occurrence of hallucinations in tasks involving knowledge-grounded conversational responses. In these tasks, the goal is to generate responses within a dialogue context using a vast knowledge base. Lewis et al. [2020] reported that RAG models are more factual and specific than BART for Jeopardy question generation.

While acknowledging the successes of NLP and RAG in their application across various domains, including general question-answering and knowledge-

grounded conversational responses, this thesis explores the practical implementation of RAG on the IR-Anthology, a collection of 62,846 papers on information retrieval meticulously compiled by Webis. This dataset offers a unique and challenging context for harnessing RAG's potential to improve content generation quality and accuracy. By leveraging these capabilities, such as real-time retrieval of external knowledge and adaptability to domain-specific information, this master's thesis main motivation is to implement it on the IR-Anthology dataset with a specific goal in mind: facilitating easy access to the wealth of knowledge contained within these papers for future researchers. The main objective of this thesis; however, is to investigate the optimal configuration of the RAG pipeline for the IR-Anthology. This investigation aims to identify a configuration that delivers both superior overall performance and exceptional value proposition. It also encompasses the mitigation of inherent limitations associated with standalone LLMs.

To lay the groundwork for this exploration, this thesis commences by presenting a comprehensive overview of recent advancements within the realm of NLP and RAG. This review serves a dual purpose: firstly, to shed light on the current landscape of methodologies and technologies in the field, offering a foundational understanding for subsequent discussions; and secondly, to establish a baseline for the innovative contributions within this work. The foundational explanation will provide crucial insights into the existing paradigms, their strengths, and limitations, thereby setting the stage for the introduction of novel methodologies utilizing RAG.

The systematic approach in this thesis involves crafting distinct pipelines for retrieval and generation stages. A comprehensive evaluation process employs diverse metrics, providing a unified assessment of efficacy and performance. Beyond overall effectiveness, the evaluation aims to reveal nuanced variations in outputs across scenarios, offering insights into adaptability and robustness. Within the experimentation's done for the RAG pipeline it was revealed that the bigger the chunk is the easier it is retrieved from the knowledge base; however, it comes at a tradeoff because the bigger the chunk is the more the LLM struggles in formulating an accurate response to the query based on the chunk. Leading the LLM to prefer smaller chunks retrieved from the knowledge base.

This introductory chapter has provided a foundational understanding of the research problem and its significance. Subsequent chapters will delve deeper into various aspects of the study:

- Chapter 2: An overview of the IR Anthology dataset and the original paper introducing RAG will be presented. This chapter will explore the potential applications of RAG and the importance of the IR Anthology

dataset in this context. As well as establishing the mains components that are within the pipeline such as transformers, LLMs, and embedding models.

- Chapter 3: We will explore the specific technological tools and methodologies employed in this research. This chapter will detail the process of building different RAG pipelines for further analysis. As well as the reasoning and selection of certain models, and retrieval methods.

- Chapter 4: Various evaluation methods will be explored, focusing on a two-step approach to assess the effectiveness of the individual components within the RAG pipeline.

- Chapter 5: The analysis chapter will present the findings of the research, including the generated results and any insights gleaned from the analysis.

- Chapter 6: The concluding chapter will give a brief review of all previous chapters as well as addressing the major challenges faced and finally detailing the final insights.

Chapter 2

Related Work

The previous chapter discussed the transformative influence of pre-trained LLM models in NLP. It explored their successes while acknowledging limitations like hallucination, information cutoff, and domain specificity. This chapter delves into potential solutions, exemplified by the RAG approach applied to the IR-Anthology dataset.

This chapter will first review the IR-Anthology dataset, a collection of text documents relevant to information retrieval research. Then, it will explore the original RAG approach (Lewis et al. [2020]), followed by a breakdown of the basic RAG pipeline. Next, it will delve into transformers, the key role in both retrieval and generation tasks within RAG. The chapter will also explain LLMs. Finally, it will discuss the embedding models.

2.1 IR-Anthology

The Information Retrieval Anthology (IR-Anthology) by Potthast et al. [2021] draws inspiration from and addresses challenges identified in existing research and projects within the domain of scholarly search and information retrieval. As an ongoing initiative, the IR-Anthology serves as a valuable repository for researchers and practitioners, providing access to a diverse range of literature related to information retrieval.

The ACL Anthology reference corpus proposed by Bird et al. [2008], a digital archive of conference and journal papers in natural language processing and computational linguistics, provides a curated collection of publications and serves as a reference repository of research results. It serves as a benchmark and inspiration for the IR Anthology, with its success attributed to a unified collection of bibliographic metadata and a comprehensive set of openly accessible full texts. The centralized web service architecture and search capabilities of the ACL Anthology form a foundational reference for the development of

the IR Anthology's analogous components.

The IR Anthology leverages the Digital Bilbiography & Library Project (DBLP) as a primary source for bibliographic metadata. Recognizing the labor-intensive nature of compiling such data, the integration with DBLP provides a starting point for the IR Anthology's metadata collection. The use of tailored scripts and automatic imports demonstrates an understanding of the challenges associated with maintaining a comprehensive and up-to-date bibliographic database.

In the context of RAG, the IR-Anthology holds particular relevance due to its extensive coverage of information retrieval techniques, corpora, and associated metadata. The inclusion of research papers spanning various subdomains within information retrieval offers a rich source of knowledge that can be leveraged to build a one stop point for researchers through RAG pipelines.

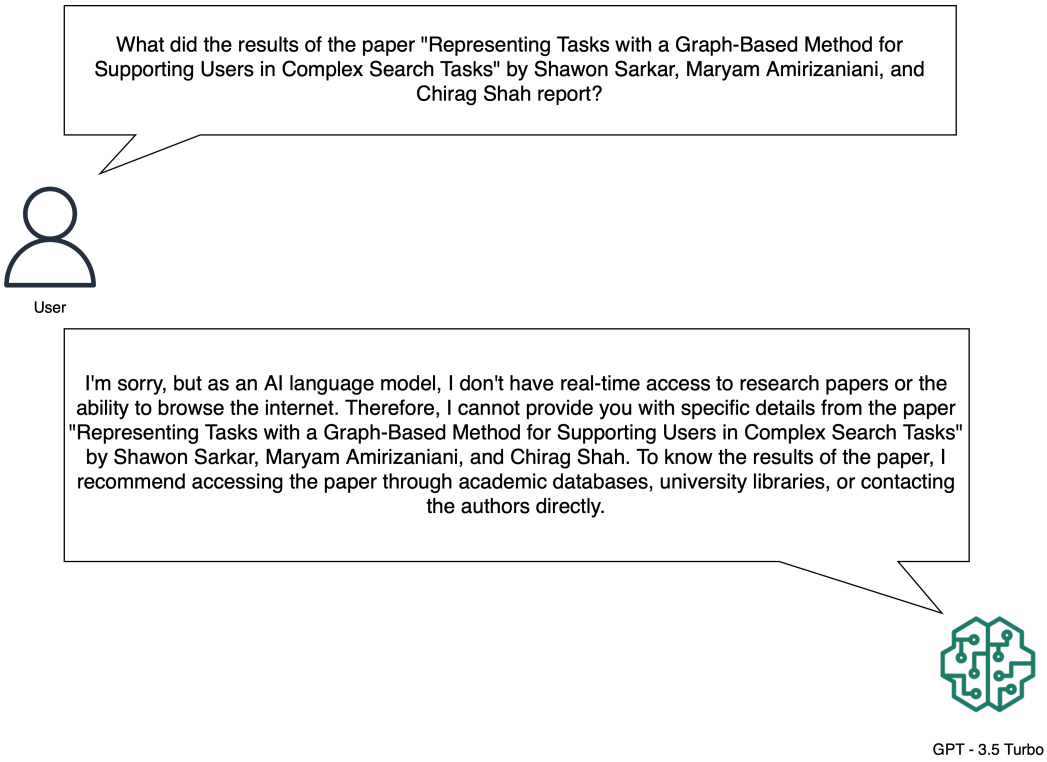

Figure 2.1: Querying GPT 3.5 to inquire about the research done by Sarkar et al. [2023]

Incorporating the IR-Anthology into a RAG pipeline not only enriches the breadth of information available but also addresses the issue of misinformation or incomplete data often encountered in LLMs. By integrating this domain-specific repository, RAG can establish a robust framework for fact-checking

and validation, ensuring that generated responses are grounded in accurate and up-to-date information. Furthermore, the dynamic nature of research in information retrieval necessitates a continuous update mechanism within RAG pipelines to accommodate new findings. With proper implementation, RAG can serve as a reliable resource for researchers, providing them with trustworthy insights and reducing the risk of relying on standalone LLMs that are not connected to any knowledge base. In the depicted illustration referenced as Figure 2.1, an inquiry directed towards GPT 3.5 regarding the findings delineated in the scholarly work authored by Sarkar et al. [2023] unveiled two notable observations: firstly, a deficiency in domain specificity, and secondly, information cutoff due to GPT 3.5 latest training date being January 2022.

2.2 Retrieval Augmented Generation

A recent study by Lewis et al. [2020] investigates advancements in RAG models. They detail the architecture and key strategies employed in RAG, which aim to leverage the strengths of LLMs while simultaneously mitigating their limitations. The proposed RAG framework integrates parametric and non-parametric memory structures. The parametric memory involves a pre-trained seq2seq encoder-decoder transformer, while the non-parametric memory utilizes a dense vector index sourced from Wikipedia. This integration empowers RAG to retrieve and integrate external knowledge during inference.

Prior studies have underscored the effectiveness of retrieval in enhancing performance across various NLP tasks when approached independently. Notable applications include open-domain question answering (Chen et al. 2017), and fact checking (Thorne et al. [2018]). This thesis builds upon these achievements, demonstrating that a unified retrieval-based architecture can yield robust performance using the IR-Anthology.

In the realm of general-purpose architectures for NLP, earlier investigations have demonstrated substantial success without resorting to retrieval mechanisms. Single, pre-trained language models have exhibited strong performance on diverse classification tasks in benchmark evaluations such as GLUE (Wang et al. [2021], Wang et al. [2019]). GPT-2 created by Rao et al. [2021] further illustrated the robustness of a single, left-to-right, pre-trained language model across discriminative and generative tasks. Recent advancements, exemplified by models like BART (Lewis et al. [2019]) and T5 (Raffel et al. [2020], Roberts et al. [2020]), advocate for a unified pre-trained encoder-decoder model utilizing bi-directional attention, thereby enhancing discriminative and generative task outcomes. RAG seeks to expand the scope of tasks addressable by a single, cohesive architecture by integrating a retrieval module with pre-trained

generative language models.

2.2.1 Retrieval and Generation

The RAG framework comprises of two primary components: a retriever in Equation 2.1 and a generator in Equation 2.2, parameterized by η and θ respectively. The retriever retrieves (top_k truncated) distributions over text passages given an input query x, while the generator generates the next token based on the context of preceding tokens, the input sequence, and a retrieved passage.

$$p_\eta(z|x) \tag{2.1}$$

- η: Non-parametric retriever.
- x: Sequence given by user.
- z: Text passages to retrieve.

$$p_\theta(y_i|x, z, y_{1:i-1}) \tag{2.2}$$

- θ: Parametric generator.
- y_i: Target sequence to generate.
- $y_{1:i-1}$: Sequences previously generated.

The retrieval component adopted from Dense Passage Retrieval (DPR), employs a bi-encoder architecture that utilizes dense representations of documents and query representations produced by BERT (Devlin et al. [2019b]). On the other hand, the generator component utilizes Bidirectional and Auto-Regressive Transformers (BART) (Lewis et al. [2019]) - a pre-trained seq2seq encoder-decoder transformer. BART's architecture effectively combines input sequences and retrieved content, leveraging its state-of-the-art performance across various generation tasks.

2.2.2 RAG Evaluation

Evaluating RAG models presents a unique challenge. Traditional metrics like BLEU and ROUGE scores, which focus on n-gram overlap, may not fully capture the effectiveness of RAG in incorporating retrieved knowledge. To address this, Lewis et al. [2020] employed a multifaceted evaluation strategy.

This approach combined quantitative metrics with human evaluation to assess RAG's performance across various tasks.

Specifically, the evaluation encompassed three distinct domains: open-domain question answering (Guu et al. [2020]), abstractive question answering (Nguyen et al. [2016]), and Jeopardy question generation based on a portion of the SearchQA dataset (Dunn et al. [2017]).

Open-domain Question Answering: This evaluation assesses the model's ability to answer a wide range of questions without relying on specific predefined domains. It examines how well the model comprehends and responds to diverse queries sourced from various subjects. RAG set new benchmarks, achieving state-of-the-art results without specialized pre-training, demonstrating its robustness. RAG was able to achieve an accuracy of 44.1% accuracy in natural questions compared to a T5 model which achieved an accruacy of 34.5%.

Abstractive Question Answering: In this context, the evaluation focuses on the model's capacity to generate responses that aren't directly present in the chunks but entail a comprehensive understanding of the retrieved chunks. It measures the model's ability to generate original and informative responses. RAG outperformed baseline models like BART, exhibiting an improvement of 2.6 Rouge-L points on Open MS-MARCO Natural Langage Generation tasks. Qualitatively, RAG displayed reduced hallucination tendencies and generated factually correct text more consistently than comparative models.

Jeopardy Question Generation: This evaluation specifically involves generating questions in the style of the popular game show "Jeopardy." The model must construct inquiries that encapsulate diverse pieces of information, often requiring synthesis from multiple sources. RAG excelled, outperforming BART and showcasing superior factual accuracy and specificity in human evaluations. Reviewers found BART to exhibit higher factual accuracy than RAG in merely 7.1% of cases. Conversely, RAG showcased superior factual correctness in 42.7% of cases. In an additional 17% of instances, both RAG and BART displayed factual accuracy. These findings distinctly underscore the efficacy of RAG, surpassing a cutting-edge generation model in factual precision across the evaluated cases.

2.3 RAG Pipeline

Building on the previous section, this section delves into the basic RAG pipeline setup. A concise overview of the relevant terminology and the fundamental RAG workflow will be explored to establish a foundation. An exploration will be done for each component separately and what the possible configurations

for it are.

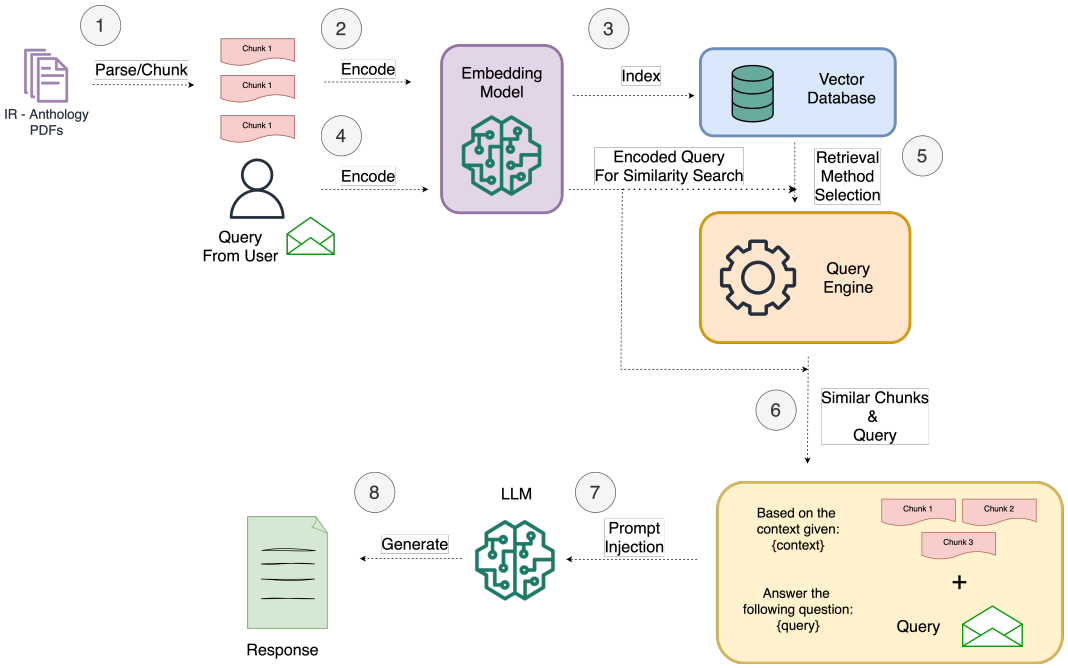

Figure 2.2: Basic flow of a RAG pipeline. Individual steps are illustrated with circled numbers.

A detailed step by step of Figure 2.2 is presented as follows:

1. Parsing & Chunking: PDFs are first parsed and split into smaller chunks.

2. Encode Chunks: The chunks are then processed into an embedding model which represents the chunks' textual content into numerical representations called embeddings.

3. Index: After the chunks have been encoded, they are then stored into the vector database and are ready to be retrieved.

4. Encode Query: Once the user gives in a query to the pipeline it is transformed into an embedding similarly as Step 2.

5. Retrieval Method: A similarity search is done between both the encoded query, and the encoded chunks through the query engine. This is done to determine which chunk is semantically similar to the query.

6. Similar Chunks & Query: After the similar chunks are retrieved they are fed into a prompt along with the query. This prompt contains an

instruction for the LLM on how to utilize the information it has been given.

7. Prompting: After the prompt has been built it is then fed to the LLM.
8. Generate: A response is generated and sent back to the user.

In Figure 2.2, numerous evaluation avenues are discernible. In the context of this thesis, the assessment will focus on Step 1, pertaining to the parsing method, Steps 2 and 4, encompassing the encoding method (embedding model), Step 5, evaluating the Retrieval Method, and Steps 7 and 8, scrutinizing the LLM model's generative capabilities. Prior to engaging in this evaluation, a succinct elucidation of these components will be provided.

2.4 Transformer Architecture

To establish a comprehensive understanding of the RAG pipeline and it's components as shown in Figure 2.2, this chapter commences with a review of the fundamental transformer architecture. This foundational review aims to equip the reader with a clear grasp of the core principles upon which various LLMs and embedding model, employed within this work. By providing this context, the discussion can then delve into the specific variations and modifications implemented in the utilized LLMs and embedding model, highlighting their unique characteristics in relation to the base transformer architecture.

Prior to 2017, recurrent neural networks (RNNs) dominated the field of machine translation. However, RNNs suffered from limitations in capturing long-range dependencies within sentences. The paper "Attention is All You Need" by Vaswani et al. [2017]) introduced the Transformer, a novel architecture that revolutionized machine translation.

The key innovation of Transformers lies in their reliance on an attention mechanism. Unlike RNNs that process information sequentially, Transformers can attend to all parts of the source sentence simultaneously. This allows them to capture complex relationships between words, even if they are far apart in the sequence.

The Transformer architecture consists of two sub-modules: an encoder block and a decoder block as shown in Figure 2.3. The encoder processes the source sentence, generating a contextual representation for each word. The decoder then leverages this representation and the attention mechanism to generate the target sentence word-by-word, focusing on the most relevant parts of the source sentence for each target word. Furthermore a break down of the transformer architecture into smaller parts will be useful later on to

address the differences between the vanilla architecture and the architectures utilized within this thesis.

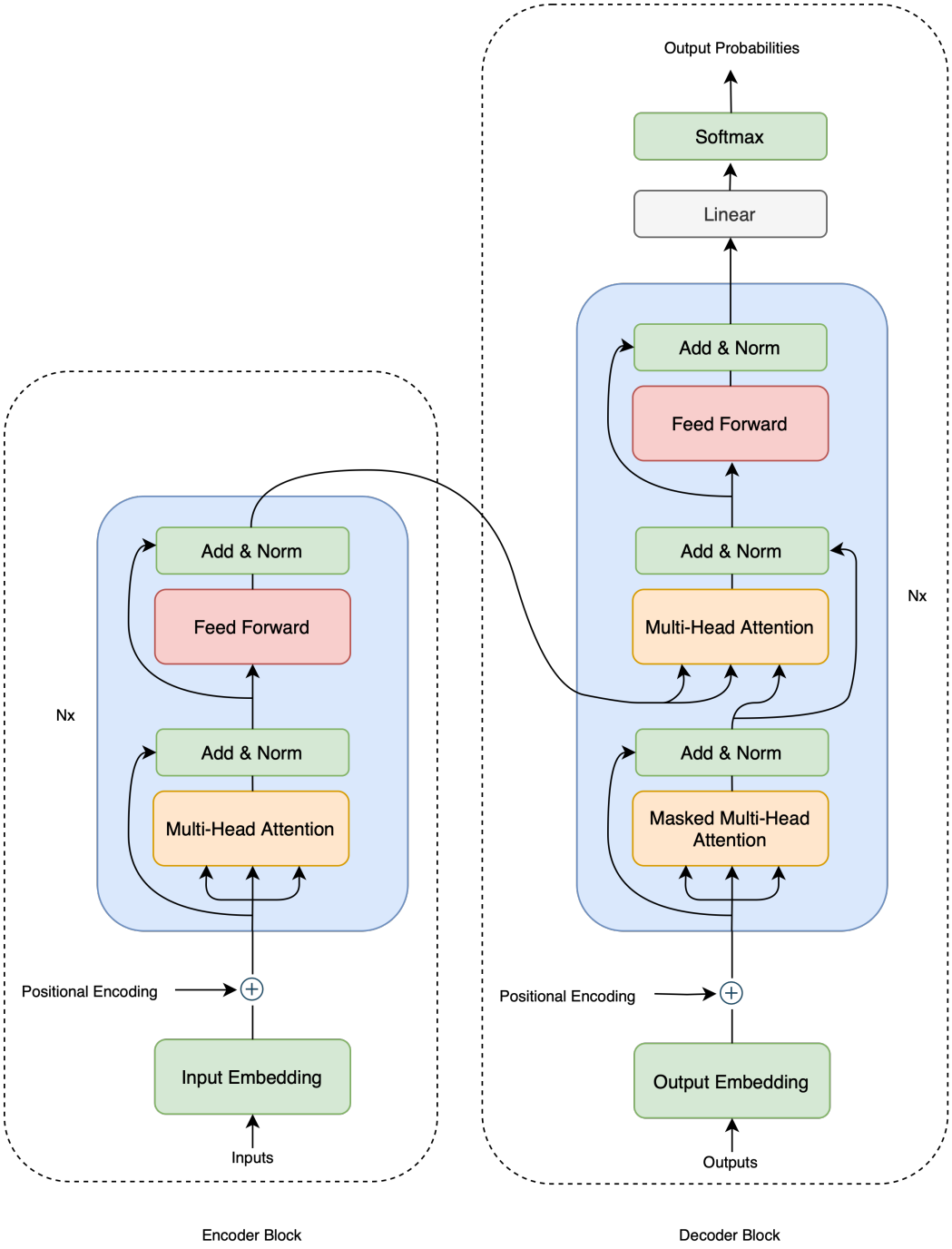

Figure 2.3: Vanilla transformer adapted from Vaswani et al. [2017]

Preprocessing:

- Input Embedding: The foundation for processing textual data in neural networks lies in the concept of input embedding. It involves transforming discrete symbolic elements (words, characters) into dense vector representations. This embedding process captures the semantic meaning and relationships between these elements. Each element in the input sequence is mapped to a vector in a high-dimensional space, where similar elements tend to reside closer together geometrically. The dimensionality (d) of this embedding vector is a hyperparameter independent of the context length.

- Tokenization: Breaks down the input text into individual units, like words or sub-words, called tokens. This is done through Byte Pair Encoding (BPE); which is a subword tokenization technique used in transformers. It iteratively merges frequent character pairs to create new subwords. This creates a smaller vocabulary while handling rare words (broken down into known subwords) and capturing some word context.

- Positional Encoding: Injects information about the relative position of each token within the sequence. This done through adding sine and cosine functions to the word embeddings, where the frequency of these functions depends on the token's position. This allows the model to learn the importance of word order in the sequence.

- Context Window: defined by the context length (L), acts like a sliding frame that focuses on a portion of the input sequence at a time. Its size determines how much information the model considers together, enabling it to capture contextual dependencies between elements. However, a larger window increases computational cost and may introduce irrelevant information, while a smaller window might miss important dependencies. The optimal choice depends on the task, data, and resources.

While a longer context length can theoretically capture more information, it also presents challenges. As L increases, the number of calculations required for the attention mechanism grows quadratically (L^2). This can significantly slow down training and inference. A long context might contain irrelevant information that can distract the model from the most important aspects of the sequence.

Encoder Block:

- Multi-head Attention: Allows the model to attend to different parts of the input sequence simultaneously, capturing relationships between tokens. This happens in multiple heads (N_x) in parallel, increasing the model's ability to learn diverse aspects of the input.
- Feed-forward Network: Introduces non-linearity into the model, allowing it to learn more complex relationships between the tokens. This non-linearity is done through the activation function known as rectified linear unit (ReLU).
- Residual Connections and Layer Normalization: Improve training stability and gradient flow through the network.

Decoder Block:

- Masked Multi-head Attention: Similar to the encoder's attention, but masks out future tokens in the output sequence to prevent information leakage during generation.
- Encoder-decoder Attention: Allows the decoder to attend to the encoded representation from the encoder, incorporating context from the input sequence into the generated output.
- Feed-forward Network: Similar to that of the encoder block.
- Output Layer: Converts the final decoder output into the desired format, like words in a translation task.

The vanilla transformer architecture revolves around two core functionalities: the encoder processing the input sequence and the decoder generating the output based on the encoded information. This versatility allows it to be employed in various configurations beyond its core encoder-decoder structure.

- Encoder-Decoder (Full Transformer): This is the original form and finds application in seq2seq tasks like neural machine translation. Both the encoder and decoder work together, with the encoder processing the input sequence and the decoder generating the corresponding output sequence.

- Encoder-Only (Embedding Model): This configuration utilizes only the encoder portion of the transformer. The encoded representation of the input sequence serves as a feature vector for various tasks like classification, sequence labeling, or creating embeddings.
- Decoder-Only (Large Language Model): In this setup, only the decoder is employed, with the encoder-decoder cross-attention module removed. This configuration is commonly used in sequence generation tasks like language modeling, where the model learns to generate text sequences (i.e., question answer format) based on the provided input.

This foundation is further strengthened by the powerful attention mechanisms, particularly self-attention, multi-head attention, and masked multi-head attention. These mechanisms enable the model to focus on specific, relevant parts of the input sequence.

Self-Attention:

This mechanism allows the model to attend to all elements within a single sequence, capturing relationships between tokens. It involves three key steps:
Linear Projections: Each token is projected into three different vector spaces given in Equations 2.3, 2.4, 2.5: Query (Q), Key (K), and Value (V). These projections are learned during training (W_q, W_k, W_v), then it is multiplied to the input sequence embedding (X).

$$Q = W_q X \tag{2.3}$$

$$K = W_k X \tag{2.4}$$

$$V = W_v X \tag{2.5}$$

Scaled Attention Scores: The model calculates a score for each pair of tokens, indicating how relevant one token (V) is to another token (Q). These scores are obtained by multiplying the Q vector of a token with the K vector of all other tokens in the sequence. They are then scaled by a factor of $1/\sqrt{d_k}$ where d_k is used to counteract the effect of the vanishing gradient problem (during training) if the softmax of the dot product of Q and K returns a very small gradient. The final scaled attention scored can be observed in Equation 2.6

$$Scaled\ Attention\ Score = \frac{QK^T}{\sqrt{d_k}} \tag{2.6}$$

Softmax and Weighted Sum: A softmax function is applied to the scaled attention scores, converting them into a probability distribution. This distribution indicates how much attention each token should receive. Finally, the model takes a weighted sum of the Value vectors based on the attention scores, resulting in a context vector in Equation 2.7 for each token that incorporates information from relevant parts of the sequence.

$$Attention(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V \tag{2.7}$$

Multi-Head Attention:

This extends self-attention by creating multiple heads (N_x) that learn different aspects of the relationships between tokens. The input sequence, Q, K, and V vectors are all linearly projected N_x times, resulting in N_x sets of Q, K, and V vectors for each head. Self-attention is performed independently on each head using their respective Q, K, and V vectors. The outputs from each head are then concatenated to form a final, richer representation of the sequence. The multi-head attention concatenation can be seen in Equation 2.8. While a visual representation can be seen in Figure 2.4

$$\text{Multi-Head Attention}(Q, K, V) = \text{Concat}(\text{head}_1, \ldots, \text{head}_{N_x})W^O \tag{2.8}$$

Masked Multi-Head Attention:

Used primarily in the decoder of a transformer, this mechanism is similar to multi-head attention with an added masking step. Since the decoder generates the output sequence one token at a time, it shouldn't attend to future tokens in the output to prevent information leakage. During attention score calculation, future tokens in the sequence are masked out by setting their attention scores to negative infinity before applying the softmax function. This ensures the model only attends to past tokens when generating the current output.

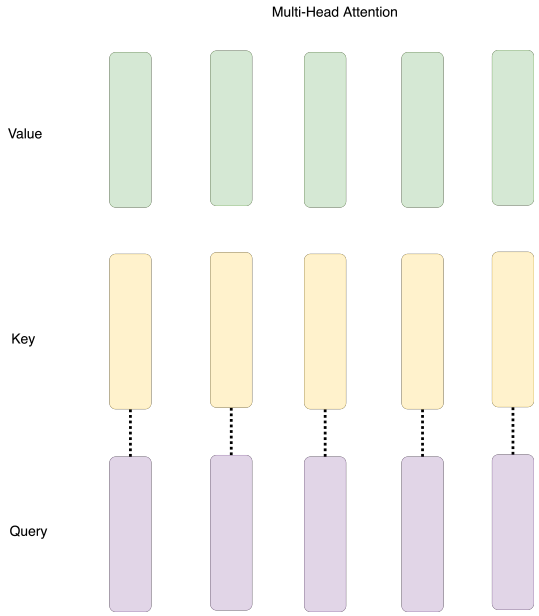

Figure 2.4: Multi-Head Attention Mechanism. Every head is represented by a Query, Key, and Value

2.5 Large Language Models

While the foundational principles of transformers, such as self-attention and multi-head attention, remain integral, modifications have been introduced to address specific limitations associated with Language Model limitations. Notably, the LLMs incorporate advanced memory optimization techniques, allowing them to efficiently process and retain information.

Building upon these foundational principles, most large language models have transitioned to decoder-only transformer architectures, streamlining the model for text generation tasks. This shift enables a more efficient focus on the core strength of LLMs: producing creative and informative text formats. Within this realm of decoder-only transformers, GPT-4 by OpenAI [2023] stands out for several key reasons. Firstly, its advanced memory optimization techniques allow it to process and retain information over longer contexts, leading to more coherent and factually accurate outputs. Secondly, GPT-4 exhibits superior performance on various benchmarks, such as Massive Multitask Language Understanding in 57 subjects (MMLU) introduced by Hendrycks et al. [2021] where it achieved a score of 86.4%. It also demonstrated its ability to excel in diverse tasks like code generation, question answering, and creative writing.

While acknowledging the advancements offered by closed-source models

like GPT-4, open-source LLMs offer distinct advantages for academic research. These advantages include fostering transparency and cost-effectiveness, along with enabling the reproducibility of results through readily available models. Additionally, open-source LLMs facilitate easier deployment on local systems, streamlining the research process.

2.5.1 Llama2 Architecture

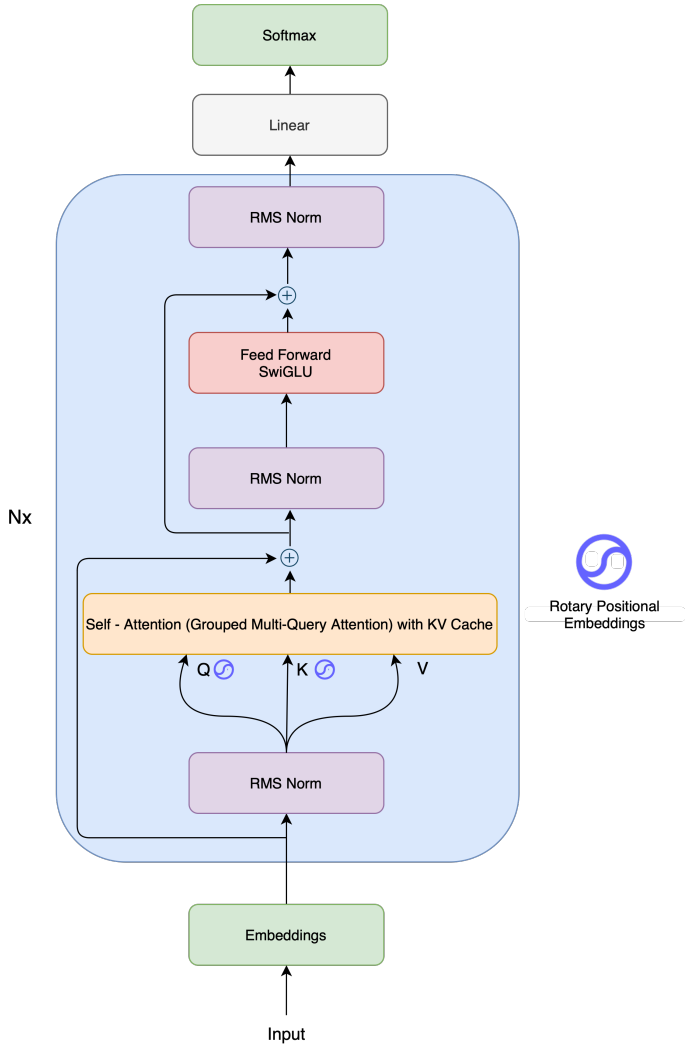

Figure 2.5: Llama2 decoder-only transformer architecture as introduced by Touvron et al. [2023].

This section delves into the architectural underpinnings of the evaluation LLM which will be later introduced in Chapter 3 by first examining the foundational

Llama2 13b architecture upon which it is built.

A significant recent advancement in large language models (LLMs) was the introduction of the Llama2 model presented by Touvron et al. [2023]. This work deviates from the standard transformer architecture by employing a decoder-only configuration. In Figure 2.5 a decoder-only transformer architecture of Llama 2 can better visualize the difference between it and a vanilla transformer.

Root Mean Square Layer Normalization (RMSNorm):

RMSNorm first introduced by Zhang and Sennrich [2019] is a simplified version of layer normalization used in vanilla transformers. It calculates the Root Mean Square (RMS) of activation's along specified dimensions, effectively normalizing the data. The key Differences between it and layer norm used in vanilla transformer architecture is that RMSNorm is computationally cheaper than layer normalization due to its potentially simpler calculations. As well as it incorporates learnable scaling factors (weights) for each dimension. This allows the model to dynamically adjust the normalization based on the specific task and data it encounters during training.

Swish Gated Linear Units (SwiGLU):

Shazeer [2020] introduced SwiGLU which is an activation function that combines the strengths of two existing techniques. Firstly the swish activation function which is a smooth, non-monotonic function addresses vanishing gradients and "dying ReLU" issues, commonly encountered with the ReLU activation. It offers a smooth transition between positive and negative values, improving gradient flow during backpropagation. Secondly the Gated Linear Unit (GLU) which incorporates a gating mechanism that controls the information flow within the network. A sigmoid function acts as a "gate" that modulates the output of a linear transformation, allowing the network to learn selectively which information to pass through. Switching out ReLU to SwiGLU allows the model to improve the gradient flow which avoids vanishing gradients, adaptive information flow offered by the gating mechanism which allows it to selectively learn and transmit relevant information, and finally potential efficiency.

Rotary Positional Embedding (RoPE):

This approach incorporates positional information more efficiently compared to traditional positional encoding. The model can process longer se-

quences of text, leading to improved understanding and generation capabilities. First introduced by Su et al. [2024] RoPE applies a rotation matrix to the token's embedding vector. The rotation angle is proportional to the token's absolute position in the sequence. For example, the first token might be rotated by a small angle, and subsequent tokens by progressively larger angles.

Grouped-query attention (GQA) & KV Cache:

GQA was proposed by Ainslie et al. [2023], which aimed to decrease the tension between performance and computational efficiency. Transformers rely heavily on the attention mechanism, which allows them to attend to specific parts of the input sequence. However, the standard attention mechanism involves extensive computations, especially for LLMs with a large number of parameters and attention heads. This high computational cost translates to slower training times and increased resource requirements for inference, hindering the practical application of LLMs. Multi - head attention traditionally has each query head within the attention layer attending to the entire input sequence independently. GQA as seen in Figure 2.6 groups these query heads into multiple smaller groups, each sharing a single key and value head.

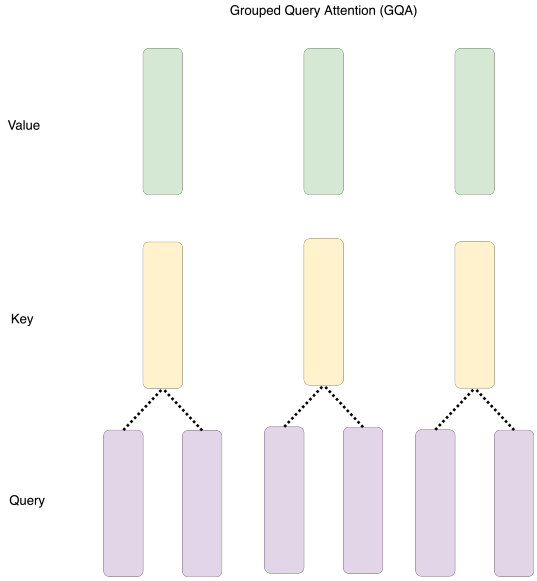

Figure 2.6: Grouped Query Attention as introduced by Ainslie et al. [2023]

KV Cache, also known as key-value cache, connects to GQA in the context of optimizing the efficiency of LLMs, specifically by reducing redundant computations during the attention mechanism. It acts as a temporary storage mechanism that stores frequently accessed key-value pairs. In the context of

GQA, the key could represent the input sequence embedding and the value could represent the intermediate attention scores calculated for specific groups of query heads. When a specific group of query heads needs to attend to the same input sequence, GQA can first check the KV Cache to see if the corresponding attention scores (value) have already been computed for the same key (input sequence embedding). If the scores are found in the cache, GQA can reuse them directly, avoiding redundant calculations. This significantly improves efficiency compared to recalculating the scores every time.

Training:

Touvron et al. [2023] trained their Llama2 model on a carefully curated dataset of 2 trillion tokens, sourced from publicly available resources. This emphasis on factual sources, achieved through up-sampling, aims to enhance the model's knowledge base and minimize the generation of factually inaccurate information.

Model Sizes:

Llama2 7B: This base model boasts 7 billion parameters, offering a balance between performance and resource efficiency.
Llama2 13B: With 13 billion parameters, this model exhibits improved capabilities compared to the 7B variant, particularly in tasks requiring deeper understanding.
Llama2 70B: The largest model in the series, featuring 70 billion parameters, pushing the boundaries of LLM performance.

Performance:

The primary distinction between the models lies in their parameter size and associated capabilities. As the parameter size increases, the model's capacity to learn complex relationships and generate nuanced text grows. The 70B model demonstrates the most impressive performance across various benchmarks, including MMLU scores and reasoning tasks. In Table 2.1 the different MMLU values for the different Llama2 sizes can be seen. The 13B model delivers competitive performance, while the 7B variant offers a more economical choice with respectable capabilities. The video random access memory (vRAM) requirement scales with the parameter size. The 7B model has the lowest requirement, followed by the 13B and 70B models, respectively.

This analysis of the Llama2 architecture provides a crucial foundation for

Llama2 Parameters	MMLU Score
7B	45.3%
13B	**54.8%**
70B	68.9%

Table 2.1: Llama2 Parameters and MMLU Scores as reported by Touvron et al. [2023].

understanding the discussion of the evaluation LLM in Chapter 3 and its specific adaptations built upon this innovative base.

2.6 Embedding Model

To facilitate a deeper understanding of the specific embedding model employed within this thesis, a preliminary discussion on the state-of-the-art Bidirectional Encoder Representations from Transformers (BERT) model is necessary. This is because a significant portion of contemporary embedding models leverage architectures similar to BERT.

Bidirectional Encoder Representations from Transformers:

BERT stands as a landmark achievement in NLP. Introduced by Devlin et al. [2019b], it leverages the Transformer architecture in a novel way to achieve state-of-the-art performance on a wide range of NLP tasks. BERT's architecture is an encoder - based transformer, a powerful neural network for sequence modeling. The Transformer encoder utilizes a self-attention mechanism, allowing each word in a sentence to attend to all other words, capturing their relationships. This is a significant improvement over previous models that processed text sequentially, left-to-right or right-to-left. BERT employs a deep encoder stack, typically with multiple layers of Transformer encoders, allowing it to learn complex contextual representations of words. BERT's training process is a two - step process (pre-training, and fine-tuning). The first being pre-trained on a massive dataset of text (e.g., books, articles) in an unsupervised manner.

Masked Language Modeling (MLM): Randomly masks words in the input sentence and trains the model to predict the masked words based on the surrounding context. This enforces the model to learn deep contextual representations for each word.

Next Sentence Prediction (NSP): Provides the model with two sentences and trains it to predict if the second sentence follows the first one in the

original text. This objective helps the model understand relationships between sentences.

Fine-tuning: The pre-trained BERT model is then fine-tuned for specific NLP tasks like question answering, sentiment analysis, or text summarization. This fine-tuning involves adding a task-specific output layer on top of the pre-trained BERT encoder and training the entire model on labeled data for the desired task.

This pre-training and fine-tuning approach allows BERT to acquire general-purpose knowledge from vast amounts of text data and then specialize in specific tasks through fine-tuning.

This analysis of the BERT architecture provides a crucial foundation for understanding the discussion of the embedding in Chapter 3.

A foundational understanding of the BERT architecture, as presented in this chapter, is essential for the understanding the embedding model used in RAG pipeline which will be explored in Chapter 3.

2.7 Chapter Conclusion

This chapter introduces the IR-Anthology and the rationale behind employing a RAG framework for its functionality. We discuss the potential benefits of RAG in addressing limitations encountered with standalone LLMs. Subsequently, the chapter delves into the core concepts of RAG, outlining the fundamental pipeline alongside its constituent components and their interaction flow. A key focus is placed on the Transformer architecture, which plays a pivotal role in both the retrieval (embedding model) and generation (LLM) aspects of RAG.

Chapter 3 will explore the specific LLMs, embedding model, and other components utilized within the RAG pipeline, as depicted in Figure 2.2. Additionally, detailed setups for these components will be provided.

Chapter 3

Approach

Chapters 1 and 2 established the limitations of LLMs in three key areas: hallucinations, information cutoff, and domain specificity. Chapter 2 explored RAG as a potential solution, demonstrating its efficacy and robustness in open-domain question answering, abstractive question answering, and Jeopardy question generation tasks. Followed by the exploration of the RAG pipeline, along with its individual components, and the scientific underpinnings of transformer architectures and its variations. Building upon this foundation, Chapter 3 will delve deeper into the selected components and variations that have been implemented within this thesis.

This chapter dives into the core components of the system. It starts by introducing the two LLMs used in this thesis, Mistral 7B and Prometheus 13B, highlighting their differences from the Llama2 architecture. The focus then turns to optimization techniques, including quantization and vLLMs. Next, the chapter explores the crucial role of the embedding model, which translates text into numerical vectors.

The narrative then introduces Llamaindex, a unifying platform that orchestrates the entire RAG logic. This framework seamlessly integrates all the previously discussed components: parsing methods, indexing, retrieval, and generation. Finally, the chapter delves into retrieval methods, explaining how the RAG pipeline finds relevant information from the vast knowledge base.

3.1 Mistral 7B

Mistral 7B, introduced by Jiang et al. [2023], is a decoder-only transformer language model serving as the backbone and primary RAG component for response generation in this thesis. This open-source model excels due to its robust architecture, surpassing many alternatives and revolutionizing the capabilities of open-source models upon its release. Building upon the existing

Llama2 architecture, Jiang et al. [2023] leveraged two established components: Sliding Window Attention (SWA), and Rolling Buffer Cache. These additions enhance the model's ability to handle longer sequences effectively while reducing computational cost. Additionally the model parameters for Mistral 7B can be observed in Table 3.1.

Parameter	Value
dim	4096
n-layers	32
head-dim	128
hidden-dim	14336
n-heads	32
n-kv-heads	8
window-size	4096
context-len	8192
vocab-size	32000

Table 3.1: Mistral 7B model architecture from Jiang et al. [2023].

Sliding Window Attention:

SWA (Sliding Window Attention) introduced by Beltagy et al. [2020] harnesses the hierarchical architecture of a transformer to broaden its information attention beyond the designated window size W. In the context of transformers, each layer consists of hidden states. Hidden states represent intermediate activations within the encoder/decoder stacks. They capture the evolving feature representation of the input sequence as it progresses through the network. The hidden state at position i within layer k, denoted as h_i, serves as a representative encoding of the input token at that specific position. The attention mechanism allows this hidden state to attend to all hidden states from the previous layer within a specified window, defined by the range between $i - W$ and i.

This recursive process empowers the hidden state h_i to access information from the input layer, encompassing tokens up to $W * k$ positions away. As illustrated in Figure 3.1, this mechanism facilitates a comprehensive understanding of the input sequence, even when it extends beyond the immediate window of attention.

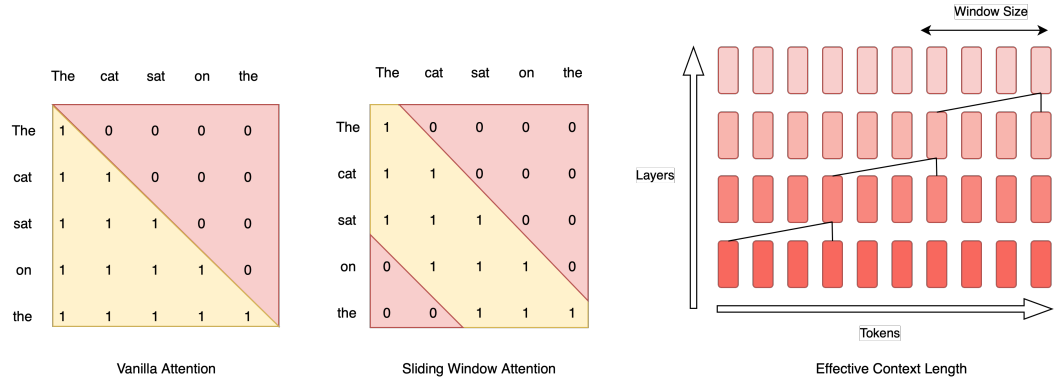

Figure 3.1: Sliding window attention adapted from Jiang et al. [2023].

Rolling Buffer Cache:

Jiang et al. [2023] proposed a technique called "rolling buffer cache" to limit the cache size and improve memory efficiency. This cache has a fixed size, denoted by W, and stores the information for each timestep i in a specific location within the cache based on a modular operation ($imodW$). This essentially creates a circular buffer where older entries are overwritten as newer ones come in, preventing the cache from growing indefinitely. They use an example with $W = 3$ in Figure 3.2 to illustrate this concept. Notably, this approach reduces the cache memory usage by $8x$ for a sequence length of 32k tokens, without sacrificing the model's performance.

Performance:

Jiang et al. [2023] offers two versions of its 7B parameter language model: a pre-trained base model and an instruction-tuned model specifically designed for chat applications. The instruction-tuned model[1] demonstrates superior performance in chat-like settings.

While the base model can be run with 14.4 GB of vRAM, Jiang et al. [2023] recommends using a system with at least 24 GB of vRAM for optimal performance. Additionally, the model achieves a reported MMLU score of 60.1%. Which outperforms the Llama2 13B model as found in Table 2.1. For this thesis the instruction-tuned model will be utilized.

[1] `https://huggingface.co/TheBloke/Mistral-7B-Instruct-v0.2-AWQ`

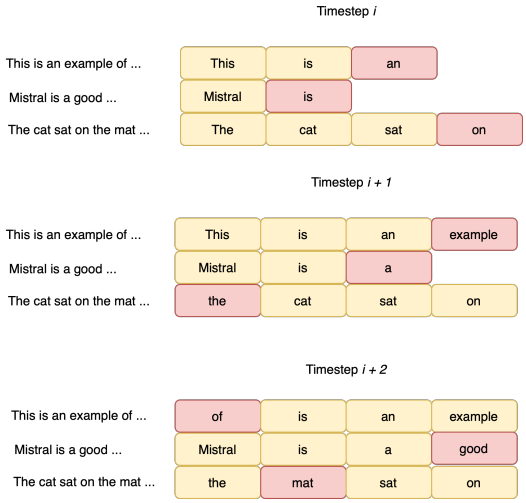

Figure 3.2: Rolling Buffer Cache adapted from Jiang et al. [2023]. The cache operates with a predefined capacity of W entries. Data is stored using a key-value structure, where each key-value pair is placed at a specific position determined by the modulo operation ($imodW$) on the key's index i. If the index i exceeds the cache's capacity W, the oldest entries are overwritten to accommodate new data. The most recently generated tokens and their corresponding internal representation are highlighted for easy identification.

3.2 Prometheus 13B

According to Kim et al. [2023], the recent trend in evaluating long-form responses involves using powerful, proprietary LLMs like GPT-4. However, this approach raises concerns for practical use due to the limitations inherent to closed-source models, such as lack of control over updates, potential bias, and high costs. Kim et al. [2023] reported that evaluation using GPT-4 for 1000 samples costs over $2000. To address these issues, the authors propose Prometheus, a fully open-source LLM based off the pre - training of Llama2 which is then specifically finetuned for evaluation tasks. When provided with relevant reference materials, Prometheus achieves evaluation performance comparable to GPT-4. Kim et al. [2023] developed a new dataset called FEEDBACK COLLECTION to train Prometheus, which includes various evaluation instructions, responses, and feedback generated by GPT-4. Compared to other open-source and commercial options, Prometheus demonstrates strong performance in evaluating text based on customized criteria. Additionally, it shows promise as a general-purpose reward model due to its high accuracy in benchmarks that measure human preference.

FEEDBACK COLLECTION is a method for gathering information about

how well a LLM performs a task. It involves creating a dataset of questions, instructions, and responses that are used to train an evaluation LLM. This evaluation LLM then scores the responses of other LLMs based on a set of criteria. The method is designed to be fair and unbiased. It uses a large number of examples with different scores so that the evaluation LLM doesn't learn to favor any particular type of response.

Here's a breakdown of the components involved:

Input:

- Instruction: This is a question or task that the LLM needs to complete.
- Response to Evaluate: This is the answer that the LLM gives to the instruction.
- Customized Score Rubric: This is a set of guidelines that tells the evaluation LLM how to score the response. It includes things like what the LLM should look for in a good answer and how to rate different aspects of the response.
- Reference Answer: This is an example of a perfect answer to the instruction. The evaluation LLM can use this to compare the response it is evaluating.

Output:

- Feedback: This is an explanation of why the response received a particular score. This helps to understand how the evaluation LLM arrived at its decision.
- Score: This is a number between 1 and 5 that indicates how well the response performed on the task.

Since Prometheus was built upon Llama2 it offers the same 7B and 13B finetuned models. To choose the evaluation model required for this thesis the Multi-Turn Benchmark (MT Bench) - Human Preference metric was the most suitable. The MT-Bench first introduced by Zheng et al. [2023] is a metric that measures the ability of LLMs to engage in coherent, informative, and engaging conversations. The authors hand-crafted multiple customized score rubrics and generated a reference answer using GPT-4 for each test prompt as well. Which in turn created a new evaluation benchmark called MT-Bench Human Preference. In Table 3.2 it can be observed how fine-tuning improved

and aligned more with human preferences. For this thesis the Prometheus 13B[2] will be utilized as the evaluator LLM.

Evaluation LM	MT - Bench (Human Preference)
Llama2 7B	51.78%
Llama2 13B	52.34%
Prometheus 7B	55.14%
Prometheus 13B	**57.72%**
GPT-4-0613	63.87%

Table 3.2: MT - Bench Human Preference for different models as reported by Kim et al. [2023].

3.3 Quantization & vLLM:

The remarkable capabilities of LLMs in tasks like text generation, translation, and question answering come at a significant computational cost. Their immense computational demands pose a significant challenge for widespread deployment. Training and running LLMs typically require expensive hardware with substantial memory resources, limiting their accessibility and practicality.

This thesis utilizes LLMs that have been quantized to address the computational complexity of LLMs. Quantization is a model compression strategy that reduces the memory footprint and computational cost of a model by converting its weights and activations from high-precision floating-point numbers (e.g., 32-bit) to lower-precision data types (e.g., 8-bit integers) with minimal impact on accuracy. This approach enables significant gains in model efficiency, facilitating the deployment of LLMs on resource-constrained devices and fostering broader adoption.

In this work, all LLMs employed have been quantized using the Activation Aware Weight Quantization (AWQ) technique introduced by Lin et al. [2023]. AWQ tailors the quantization process to consider both the weights and activations within each layer, leading to improved accuracy compared to standard quantization methods.

vLLM[3](Kwon et al. [2023]) is a high-throughput and memory-efficient inference and serving engine specifically designed for LLMs for both floating-point numbers, and their quantized counter parts (AWQ). It facilitates the implementation of efficient attention mechanisms like Flash Attention and Paged Attention.

[2] `https://huggingface.co/TheBloke/prometheus-13B-v1.0-AWQ`
[3] `https://github.com/vllm-project/vllm`

Activation Aware Weight Quantizaion:

AWQ (Lin et al. [2023]) recognizes that not all weights in a model are equally important. Some weights have a significant impact on the final output, while others have a minimal effect. AWQ uses a calibration step to identify these "salient weights." A small subset of the training data is passed through the model, and the activations are analyzed. Based on this analysis, AWQ determines which weights have a larger influence on the activations. Once identified, these crucial weights are protected during the quantization process. They are quantized with higher precision to minimize the introduction of errors. The remaining, less critical weights are quantized with lower precision. This approach significantly reduces the overall memory footprint of the model without compromising accuracy. In Figure 3.3 it can be observed how only the salient weights are left with their original weights, while the remaining weights are quantized. The red shading of the X matrix represents how salient the weights are.

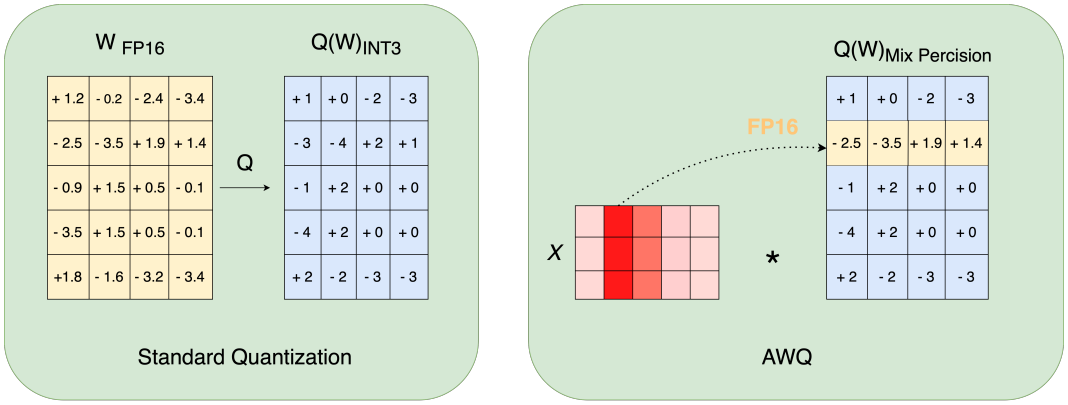

Figure 3.3: Standard Quantization vs. AWQ (Lin et al. [2023]).

Flash Attention:

Flash Attention (Dao et al. [2022]) leverages hardware accelerators, such as graphical processing units (GPU) or tensor processing unit (TPU), to significantly accelerate attention calculations within the LLM. Standard attention mechanisms, a core component in many LLMs, are computationally expensive and struggle with long sequences due to their quadratic memory complexity. This is where Flash Attention offers a significant performance improvements and efficient memory utilization.

In standard attention, the model calculates a compatibility score between

every element in the input sequence. This process requires storing a large attention matrix in memory, whose size scales quadratically with the sequence length. For long sequences, this matrix becomes enormous, exceeding the memory capacity of available hardware. Additionally, standard attention involves frequent data transfers between slower memory (high bandwith memory) and faster on-chip memory (static random-access memory) on GPUs, leading to performance bottlenecks.

Flash Attention tackles these challenges through two key innovations. First is tiling in which the attention matrix is cleverly divided into smaller, manageable tiles. This approach significantly reduces the memory footprint required to store the entire matrix at once. Secondly instead of repeatedly transferring data between memory and performing calculations step-by-step, Flash Attention performs all necessary operations (key, query, and value transformations) within the on-chip memory in one go. This eliminates the need for frequent data transfers and boosts performance resulting in a reduced memory footprint by utilizing tiling.

Flash Attention requires significantly less memory to process long sequences compared to standard attention. This enables processing of larger models and longer sequences on hardware with limited memory resources. Also resulting in Faster Inference by minimizing data transfers and performing fused operations in on-chip memory, Flash Attention significantly accelerates the attention calculations. This translates to faster model inference times and improved overall performance.

Flash Attention addresses the memory bottleneck and performance limitations of standard attention mechanisms. By leveraging tiling and fused operations, it offers a memory-efficient and high-performance solution for processing LLMs, particularly those dealing with long sequences. This technique paves the way for deploying powerful LLMs on resource-constrained devices.

Paged Attention:

The attention mechanism involves storing information about past words, like their embeddings, in the GPU's memory, which is referred to as the KV cache. The challenge is that for large models and long sequences, this KV cache can become very large, consuming a significant amount of GPU memory. This memory limitation restricts the length of sequences the LLM can handle and reduces the number of requests a system can process simultaneously.

PagedAttention (Kwon et al. [2023]) tackles this problem by introducing a smarter way to manage the KV cache. Instead of storing the entire KV cache as one contiguous block, PagedAttention divides it into smaller, fixed-size blocks. A separate lookup table keeps track of which block holds the information for

a particular word in the sequence. When the LLM needs information about a word, it uses the lookup table to find the corresponding block in memory and retrieves only the relevant data. This eliminates the need to keep the entire cache readily available.

The experiments in this thesis were conducted on an A100 GPU with 40 GB of vRAM. This hardware configuration provides a robust platform for evaluating the performance and efficiency of the quantized LLMs.

3.4 BGE Large:

In their work, Xiao et al. [2023] introduce Chinese Text Embedding Models (C-TEM) , a comprehensive set of well-trained multi - lingual embedding models. These models are based on a BERT-like architecture, where the last layer's hidden state of a special classify Token ([CLS]) is used as the embedding. They come in three sizes: large (326M parameters), base (102M parameters), and small (24M parameters). The large model achieves the best overall performance, significantly outperforming other publicly available models. The small model, while competitive with others in C-TEM, is also much faster and lighter, making it ideal for large knowledge bases and high-throughput applications. This range of model sizes allows users to choose between efficiency and representation quality based on their specific needs.

In the scope of this thesis, the utilization of the expansive C-TEM was employed, wherein textual information is encapsulated within a vector of dimensions 1024 (bge-large-en-v1.5[4]). The selection of this model was predicated upon its suitability for a task retrieval scenario, specifically geared towards identifying the most pertinent documents in response to a given query within an extensive text corpus. The primary objective involves retrieving the top-k documents that exhibit the highest similarity to the specified query.

To assess the efficacy of the C-TEM retrieval system, Xiao et al. [2023] assessed two fundamental criteria were taken into account: its proficiency in accurately ranking documents (ranking metric) and its capability to identify a comprehensive set of relevant documents (recall metric). The evaluative metric employed for this analysis was the Normalized Discounted Cumulative Gain for the top_k value of 10 (NDCG@10). The ensuing results, as presented in Table 3.3, delineate the performance outcomes of the three available models under consideration.

[4] `https://huggingface.co/BAAI/bge-large-en-v1.5`

Model	Dimension	Parameters	NDCG@10
BGE (small)	512	24M	63.07
BGE (base)	768	102M	69.53
BGE (large)	1024	326M	**71.53**

Table 3.3: Different embedding models used from Xiao et al. [2023]

3.5 Llamaindex

Llamaindex[5] is a software framework specifically designed to augment the capabilities of LLMs in the domain of RAG. Some of the core functionalities of Llamaindex are data integration, data preprocessing, and retrieval augmentation. Llamaindex establishes seamless connections between LLMs and diverse data repositories, including document databases, vector stores, and even other LLMs. This grants LLMs the ability to retrieve task-relevant information from these external sources. It can manipulate the retrieved data to render it suitable for consumption. This may encompass operations such as summarization, information extraction, and tokenization. During the text generation process, Llamaindex dynamically retrieves pertinent data based on the evolving context and injects it into the LLM.

The user furnishes a prompt or starting point for the text generation task, along with any pertinent contextual information. This is then provided through the flow with a provided prompt and context to execute a retrieval strategy, identifying and retrieving relevant information from the integrated data sources. The retrieved data undergoes processing to render it compatible with LLM consumption, and then it is fed back to the LLM as supplementary information. The LLM leverages both its internal knowledge repository and the retrieved data to generate text that is not only factually accurate but also informative and relevant to the prompt and context.

By furnishing LLMs with access to pertinent data, LlamaIndex facilitates the generation of text that is more accurate, informative, and factually sound. The retrieved data bolsters the LLM's ability to maintain coherence and consistency throughout the generated text. LlamaIndex's capacity to integrate diverse data sources and tools renders it adaptable to a broad spectrum of use cases. In essence, LlamaIndex empowers LLMs to move beyond their internal knowledge base (from pre-training) and leverage external information for a more robust and informative text generation process.

There are multiple software frameworks that offer this RAG logic, such

[5] `https://github.com/run-llama/llama_index`

as Langchain[6], and Haystack[7]. However, Llamaindex offers a more robust framework for evaluation and easy of change of techniques, and methods.

3.5.1 Parsing/Chunking Methods

In the course of this thesis, two distinct methodologies were employed: token-based parsing and sentence parsing. Prior to delving into the intricacies of these parsing methods, it is imperative to elucidate that the IR-Anthology under consideration constitutes a collection of PDF documents. The initial step in extracting textual content from these documents necessitates the use of a PDF reader. In the context of this study, the PyPDF [8] library was employed to facilitate the extraction of text from the aforementioned PDFs. Subsequently, a comprehensive discussion on the employed parsing methods will ensue.

Token Based Chunking:

This is the most straightforward approach. You define a desired number of tokens for each chunk (e.g., 512 tokens). The text is then divided into chunks of this size as much as possible. This is computationally efficient but may lead to context loss at chunk boundaries. Adjacent chunks have some overlap (e.g., the last few sentences of one chunk are included in the beginning of the next). This helps maintain context by providing the LLM with information from both sides of the chunk boundary. The overlap size is a parameter you can adjust based on your task and the LLM's requirements. The tokenization process to know which words, or sub-words to be selected hinges on the LLM's tokenizer.

Some tasks, like summarization, may benefit from larger chunks to capture broader context. Question answering may prefer smaller chunks to focus on relevant sections. Larger chunks with some overlap generally preserve context better than smaller, non-overlapping chunks.

The token chunking method hinges primarily on two pivotal parameters: namely, the chunk size and the chunk overlap. Figure 3.4 displays token based chunking method for three chunks.

[6] `https://github.com/langchain-ai/langchain`
[7] `https://github.com/deepset-ai/haystack`
[8] `https://github.com/py-pdf/pypdf`

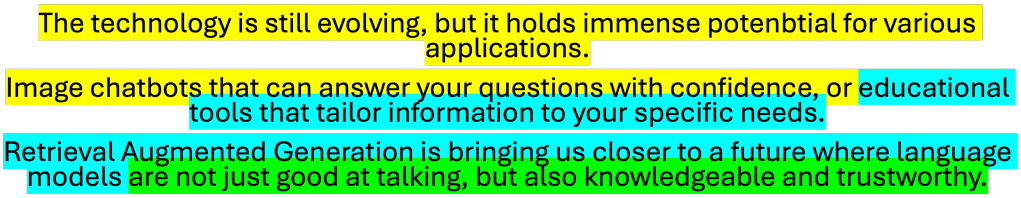

Figure 3.4: Token based chunking on three sentences.

Sentence Window Chunking:

In the context of SW chunking, the emphasis lies in partitioning the text into sentences, thereby forming discrete chunks that correspond to individual sentences. Given the inherent variability in sentence lengths, these chunks exhibit non-uniform sizes, distinguishing them from the standardized dimensions observed in token-based chunking. Regular expressions (regex) is used to split up the text into sentences. Following the segmentation of the PDF into sentence-based chunks, each chunk is augmented with metadata linking it to the adjacent sentences, thereby substantiating the inclusion of the term "window" in this methodological approach.

During retrieval and finding the most similar search to the query it is done on a singular sentence level; however, during synthesis namely step 6 in figure 2.2 the surrounding sentences are provided as the entire context. An example of what is passed into the prompt can be observed in Figure 3.5, provided a window of 3.

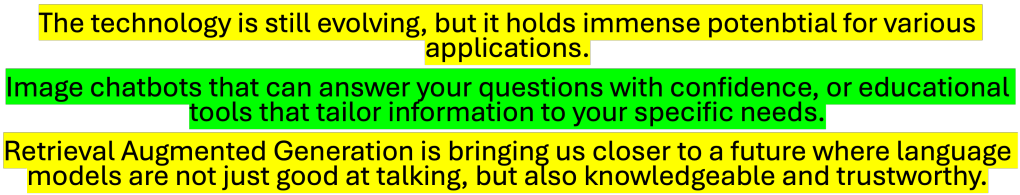

Figure 3.5: Sentence Window based chunking on three sentences.

3.5.2 Indexing/Embedding of Chunks

Following the application of the chunking method, the encoded chunks (embeddings) are then stored within a vector index. While various implementations exist for vector indexing, this work utilizes Qdrant for this purpose. To provide context before exploring Qdrant in detail, a general overview of the underlying algorithms used to construct vector indexes will be presented.

Hierarchical Navigable Small World (HNSW):

HNSW, as introduced by Malkov and Yashunin [2018], is an algorithm designed for approximate nearest neighbor search in high-dimensional vector embeddings. Several parameters influence the algorithm's performance, such as the number of neighbors connected to each data point in the graph (M). Increasing M enhances recall but at the cost of search speed. The search effort parameter (Ef) determines the number of neighbors explored during a search, where a higher Ef leads to improved search accuracy but longer search times. Additionally, the level parameter influences the graph's hierarchy, connecting distant clusters of points and enabling efficient "jumps" during search.

To set up the index, each data point, represented by its embedding, is transformed into a node in the graph. The process involves the following steps for each node:

- Randomly sample M other data points.
- Calculate the distance between the node and each sampled point.
- Connect the node to the M closest points, forming its immediate neighbors at Level 0.

Moving to the upper levels involves the following steps:

- Create a shortlist of candidate nodes for further connections, achieved by either selecting a random subset of nodes from the previous level (Level L-1) or choosing nodes that haven't reached their maximum number of connections yet.
- For each node in the shortlist: Sample a small number of points (e.g., $\sqrt{M}$) from the previous level (Level L-1). Calculate the distance to all points in the shortlist, and among these, connect to the furthest point. This connection acts as a "shortcut" to potentially distant clusters.

Navigability and efficiency in HNSW involve different levels with specific roles. In the lower levels (Level 0), the focus is on connecting very close neighbors to optimize precision for local searches. As one ascends to upper levels, the introduction of "shortcut" connections enables jumps to potentially distant but similar clusters in the data. This design establishes a navigable small world, facilitating efficient exploration of nearby points and the ability to jump to relevant areas when necessary.

The parameter M governs the density of connections and search speed. A higher M enhances recall but extends both the index build and search time. Ef acts as a balance between search accuracy and efficiency. A higher Ef results in more comparisons, potentially improving search results at the expense of increased search time. The introduction of levels establishes a hierarchy that facilitates exploration of both local neighborhoods and distant clusters. Through careful parameter selection, HNSW achieves a well-balanced trade-off between search speed and accuracy, positioning it as a suitable choice for approximate nearest neighbor searches in high-dimensional vector spaces. For this thesis when utilizing HNSW the parameters were as follows:

- Ef: 100
- M: 16

Qdrant:

The core of Qdrant [9] is written in Rust, a programming language known for its speed and memory safety. This makes Qdrant efficient androbust for handling large datasets and complex searches. Qdrant utilizes HNSW algorithm to build a navigable graph structure from the vector embeddings. To handle large datasets efficiently, Qdrant can store embeddings on disk while keeping frequently accessed data in memory. Qdrant provides a clean and convenient API for developers to interact with the database. This makes it easy to store, search, and manage vector embeddings within their applications. Overall, Qdrant offers a powerful and versatile open-source solution for building applications that leverage the power of vector similarity search.

Llamaindex provides a high-level interface (wrapper) for Qdrant databases, streamlining integration into the query engine.

3.5.3 Retrieval

Following the parsing and indexing of document chunks, the appropriate retrieval engine is selected. Subsequently, the retrieval method's parameters, including the top_k value, require careful consideration.

Top_k refers to the maximum number of most relevant chunks retrieved by the chosen method. Intuitively, a larger top_k value suggests a broader context retrieved from the index, potentially leading to a more comprehensive response to the query.

[9] `https://github.com/qdrant/qdrant`

However, this selection is not without limitations. The context length supported by the LLM must be factored in. For instance, Mistral 7B has a maximum context length of 8192 tokens. If top_k is set to 10 and each chunk comprises 1024 tokens, exceeding the LLM's capability would occur. Therefore, a smaller top_k value becomes necessary.

In this thesis work, a top_k value of 3 is deliberately chosen. This decision is driven by two factors. First, the evaluation process involves analyzing various chunk sizes. To ensure standardized evaluation across these different chunk sizes, maintaining a constant top_k value is crucial. Second, the chosen LLM's context length limitations must be considered. A top_k of 3 offers a balance between retrieving a sufficient context for informative responses and staying within the LLM's processing capabilities.

3.5.4 Generation

Once the retrieval stage has identified pertinent chunks, these chunks along with the user's query are used to construct a prompt that guides the LLM in response generation. This prompt essentially serves as a structured input for the LLM, incorporating the user's intent and the retrieved contextual information. The prompt typically combines the user's query with key elements from the retrieved documents. This may involve extractive techniques where relevant snippets from the retrieved chunks are directly incorporated into the prompt, or abstractive techniques where the key concepts and factual information are paraphrased and woven into a cohesive prompt. The constructed prompt is then fed into the LLM. The LLM's ability to process language and generate coherent text allows it to leverage the information within the prompt to formulate a response that addresses the user's query and incorporates insights from the retrieved documents.

As illustrated in Figure 3.6, a sample prompt can provide a concrete example of how the user's query and retrieved context are combined to guide the LLM's response generation.

```
"""
Based on the given context, answer the query given by the user.

Context
---------
1.  Information retrieval is the process of obtaining relevant information from a large
    collection of data.
2.  It involves searching and retrieveing specific information from databases or documents.
3.  Information retrieval focuses on efficiently finding and delivering relevant content in
    response to user queries.
---------


Query
---------
What is Information Retrieval?
---------
```

Figure 3.6: Prompting with top_k of 3.

3.6 Retrieval Methods

3.6.1 Vector Retrieval (HNSW)

Vector retrieval leverages the HNSW algorithm embedded within the vector index. The search process in HNSW follows a systematic approach. Given a new query embedding, the search begins from a random node in the graph. Exploration of the node's neighbors at Level 0 involves comparing distances (i.e., cosine similarity, euclidean distance) with the query. If the search effort (Ef) hasn't been exhausted, progression to upper levels occurs. In upper levels, "shortcut" connections are utilized to explore potentially relevant clusters located further away. The search concludes either after reaching the maximum number of comparisons specified by Ef or upon finding a sufficiently close neighbor.

3.6.2 Best Matching 25 (BM25)

The Okapi Best Matching 25 (BM25) algorithm (Robertson and Zaragoza [2009]) is a widely used ranking function in IR systems, particularly search engines. It estimates the relevance of documents to a given search query by considering various factors. In the context of this thesis it is chunks rather than the documents, but for the sake of simplicity the terminology of document will be used.

BM25 belongs to the family of "bag-of-words" retrieval functions, which means it ranks documents based on the presence of query terms within them,

without considering the order or proximity of those terms.

The BM25 score for a document D with respect to a query Q is calculated using Equation 3.1.

$$\text{score}(D,Q) = \sum_{w \in Q} \left\{ \text{IDF}(w) \cdot \frac{\text{tf}(w,D) \cdot (k_1+1)}{\text{tf}(w,D) + k_1 \cdot \left(1 - b + b \cdot \frac{|D|}{\text{avgdl}}\right)} \right\} \tag{3.1}$$

- w: A term from the query Q.
- $IDF(w)$: Inverse document frequency of term w. This reflects how rare or common the term is across the document collection.
- $tf(w, D)$: Term frequency of term w in document D. This indicates how often the term appears in the document.
- k_1: A tuning parameter that controls the influence of term frequency on the score.
- b: Another tuning parameter that affects the document length normalization.
- $|D|$: The length of document D (usually measured in terms of words).
- $avgdl$: The average document length in the collection.
- $IDF(w)$: This component ensures that terms that appear frequently across many documents (common words like "the" or "a") contribute less to the score compared to rare and potentially more informative terms.
- $tf(w, D)$: Documents containing a query term more often are intuitively considered more relevant. However, simply counting occurrences can be misleading. BM225 addresses this by incorporating a saturation factor through k_1.
- k_1: This parameter prevents scores from being dominated by documents with extremely high term frequencies. It essentially reduces the weight of additional occurrences of a term beyond a certain point.
- b: This parameter controls the document length normalization factor. A value of $b = 0$ assigns equal weight to documents of all lengths, while $b = 1$ penalizes longer documents that might contain the query terms more frequently just due to their size.

The BM25 algorithm is a powerful and well-established method for ranking documents based on their relevance to a search query. By considering both term frequency and inverse document frequency, it effectively balances the importance of terms within a document and their rarity across the entire collection. The use of tuning parameters allows for customization to different retrieval scenarios.

The BM25 algorithm in RAG context leverages textual metadata associated with the chunks for retrieval purposes. It does not directly incorporate the potential semantic richness captured within chunk embeddings.

3.6.3 Hybrid Retrieval

In the context of this thesis hybrid retrieval is the usage of both dense retrieval (i.e., vector retrieval) and sparse (i.e., BM25) are both used in the same retrieval method. Both retrieval methods are run simantenously, and then a union is performed on the retrieved chunks. The top_k retrieved chunks from each retrieval method is then ranked according to reciprocal rank fusion (RRF)(Cormack et al. [2009]).

RRF is a technique employed to consolidate ranked results retrieved from multiple search systems into a single, unified ranked list. It leverages the concept of reciprocal rank, which prioritizes documents positioned higher within individual rankings. For each chunk across all the ranked lists, RRF calculates the reciprocal of its rank within each individual list. The reciprocal rank essentially emphasizes chunks positioned higher in the rankings (closer to the first position) by assigning them a larger value. Finally, RRF aggregates the reciprocal rank values for each chunk across all the systems. Chunks consistently ranked highly across multiple systems will accumulate higher summed values. This cumulative score serves as the basis for the final re-ranked list, prioritizing chunks that received strong relevance signals from various search systems. Mathematically, for a chunk d, let $R_i(d)$ denote its rank in the ranking produced by system i. The RRF score, denoted as $RRF(d)$, is shown in Equation 3.2.

$$\mathrm{RRF}(d) = \sum_i \frac{1}{R_i(d)} \tag{3.2}$$

Equation 3.2 essentially sums the reciprocal ranks of chunk d across all the participating search systems. Chunks with a higher RRF score are considered more relevant and positioned accordingly in the final re-ranked list. RRF is an unsupervised method, requiring no training data or complex algorithms. It operates efficiently by leveraging readily available ranked lists. RRF capitalizes on the strengths of various search systems. Even if a chunk ranks highly in

only a few systems, it can still achieve a good RRF score, promoting diversity in the final results.

3.6.4 Hypothetical Document Embedding (HyDE)

Hypothetical Document Embeddings (HyDE) (Gao et al. [2022]) is a recent advancement in the field of information retrieval that leverages the power of LLMs to improve search accuracy, particularly when dealing with complex or nuanced queries. It is not a search method, rather it is a preprocessing step before it. In simple terms it is a reformation of the query to potentially enhance the similarity search. It can be used with any of the previously mentioned retrieval methods.

The process begins with a user query. An LLM (i.e., Mistral 7B - instruct), is tasked with generating a "hypothetical document" that captures the essence of the query. This document might not be factually accurate but reflects the LLM's understanding of the information need. The generation process can be repeated several times to create a diverse set of hypothetical documents. Each hypothetical document is then fed into a document embedding model. The resulting embedding vectors or text (depending on which retrieval method) are used to search the actual document corpus. By leveraging similarity techniques, the system retrieves documents in the corpus that reside closest (most similar) to the hypothetical document embeddings in the embedding space.

Gao et al. [2022] claim that HyDE can outperform traditional keyword-based retrieval methods, particularly for queries with ambiguous or metaphorical language. The hypothetical document acts as a bridge, capturing the user's intent more effectively.

3.6.5 Reranker

Rerankers are encoder - based transformer models that act as a post-processor to the retrieved chunks. They function as specialized models designed to re-evaluate and reorder the initial set of documents retrieved by a search engine in response to a user query. This process aims to elevate the most relevant and informative documents to the top of the search results, enhancing the user's experience by prioritizing the information they seek.

Following the initial retrieval of the top_k candidate documents by the retriever component, a reranker module takes center stage. This module meticulously evaluates the relevance of each retrieved document to the specific user query. This evaluation leverages a cross-encoder architecture, as introduced in Lee et al. [2023], which assesses the semantic similarity between a retrieved document chunk and the query. Unlike traditional vector similarity approaches,

where contextually chunked text and query text are independently processed by an embedding model before distance metrics (e.g., cosine similarity, Euclidean distance) are applied, cross-encoders encode both texts jointly within the embedding model. The resulting embeddings are then fed into a classifier layer, such as a neural network, to generate a final relevance score. After generating the final relevance score the chunks are then re - ordered with the highest score being placed at the first position.

This thesis leveraged the $BAAI/BGE - reranker - large$ (Xiao et al. [2023]) model[10] which has BERT-like architecture. In Figure 3.7 the differences of how the bi - encoder and cross - encoder handle the retrieved chunk, and query.

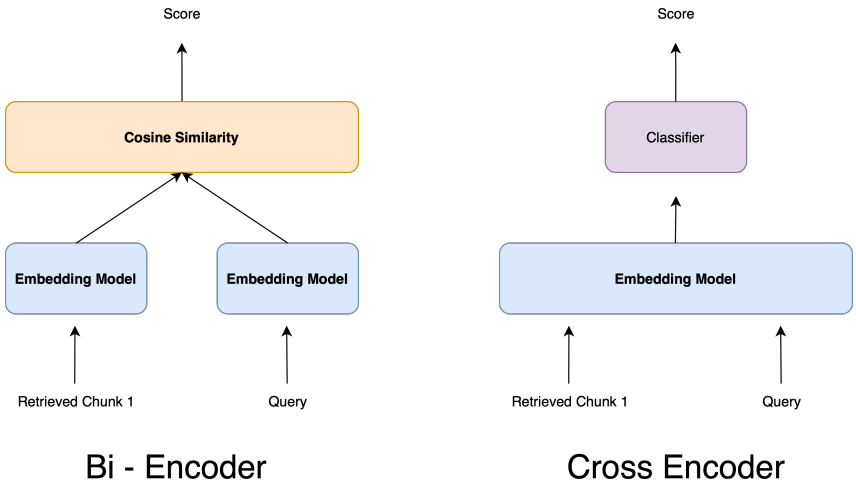

Figure 3.7: Bi - Encoder vs. Cross - Encoder

3.7 Chapter Conclusion

This chapter explores the intricacies of various methodologies employed within the RAG pipeline. It commences with an exploration of the LLMs used throughout the thesis (Mistral 7B, Prometheus 13B). Efficiency in LLM execution is then addressed, with a focus on vLLM, which leverages FlashAttention and PagedAttention techniques. Subsequently, the chapter explores the embedding model used (bge-large-en-v1.5). Followed by the functionality of Llamaindex and its integration within the RAG framework. Different variations that can occur within Llamaindex are then explained.

Finally, the chapter provides a comprehensive overview of the retrieval

[10] `https://huggingface.co/BAAI/bge-reranker-large`

methods utilized throughout this thesis. This includes the BM25 algorithm, vector retrieval using HNSW, and others.

Chapter 4, which follows, will explore the evaluation methodologies employed to identify the most suitable configuration for the IR-Anthology dataset. Additionally, it will delve into the various evaluations used and their connection to the RAG pipeline.

Chapter 4

Evaluation

The preceding chapters have meticulously dissected the inner workings of the RAG pipeline, a powerful tool designed to leverage the strengths of large LLMs and retrieval techniques. Chapters 1 and 2 unveiled the inherent limitations of LLMs, highlighting their short comings with hallucinations, information cut off, and domain-specificity. Chapter 2 introduced RAG as a potential remedy, showcasing its effectiveness across various tasks like open-domain question answering, abstractive question answering, and Jeopardy question generation. The explanation included a flow chart that explained the basic steps in a RAG pipeline. It also explained the key components, like transformers, which are the foundation of LLMs, and embedding models.

Chapter 3 built upon this foundation by delving into the intricate details of RAG's implementation. Chapter 3 meticulously examined the specific LLMs employed within the RAG pipeline, such as Mistral 7B, and Prometheus 13B, to identify their distinct advantages.

Furthermore, Chapter 3 investigated optimization techniques like quantization and vLLMs. Then an exploration of the unifying platform, the Llamaindex framework, that seamlessly orchestrates the execution of RAG logic by integrating all the previously discussed components. Finally, the chapter concluded with a comprehensive analysis of the retrieval methods utilized by RAG to extract relevant information from vast datasets.

Having established a thorough understanding of the RAG pipeline, Chapter 4 now shifts its focus towards the crucial process of evaluation. This chapter will delve into the methodologies employed to assess the suitability of various RAG configurations for the specific demands of the IR-Anthology dataset. An exploration of the diverse range of evaluation metrics and their intricate connection to the performance of the RAG pipeline.

This chapter outlines a two-stage evaluation framework for assessing retrieval performance. To address the cost and time constraints associated with

human-generated ground truth datasets, the framework leverages synthetic data. First, a discussion of the methodologies employed to generate synthetic data for various experimental setups, detailing the specific techniques used to create synthetic data tailored to each scenario. Subsequently, the chapter delves into the two-step evaluation process itself, which serves to assess the effectiveness of the retrieval methods using the synthetic data. In the context of this thesis the synthetic data will be a Q&A pair dataset which mimics the end user use case.

4.1 Generating Synthetic Data

Prior to generating the Q&A pairs, it is imperative to undertake a comprehensive dataset processing workflow involving parsing, encoding, and indexing. Given that each parsing technique possesses the potential to impact data retrieval and generation, a rigorous application of multiple parsing techniques is deemed necessary. The details of the parsed techniques, including the quantity of nodes, are presented in Table 4.1. Sentence Window (SW) 3 and 6 have the same amount of nodes since both of them are parsing sentence-wise but the only difference is the metadata of each respective window size. Token based (TB) method was used on three different sizes: 256, 512, 1024.

Parsing Method	**Chunks**
TB 1024	14920
TB 512	28586
TB 256	66446
SW 3 & 6	103940

Table 4.1: Chunks for each method in dataset preparation.

The Q&A generation process commences by initially addressing each individual node and formulating a corresponding question. Subsequently, this question is recorded in a JSON file, accompanied by a universally unique identifier (UUID). This strategic use of UUID facilitates the correlation between the specific chunk utilized for question generation and the resulting question. Following the generation of the question, an associated answer is produced and stored in a separate JSON file, encompassing information on the chunk, question, and answer.

4.1.1 LLM Parameters

Before using the LLMs to perform these generation tasks certain parameters must be set such as the *temperature*, $context_length$, and max_new_tokens.

In neural networks, particularly in the context of softmax activation, temperature is a parameter used to control the entropy or the smoothness of the probability distribution output by the softmax function as seen in Equation 4.1. After the softmax operation, the temperature parameter is applied to the logits before exponentiation, which affects the resulting probabilities.

$$\text{softmax}(z_i) = \frac{e^{z_i/T}}{\sum_j e^{z_j/T}} \tag{4.1}$$

- z_i is the i^{th} logit,
- T is the temperature parameter.

By adjusting the temperature parameter, we can control the smoothness of the probability distribution. Higher values of temperature result in a smoother distribution, while lower values lead to a sharper, more peaked distribution. In the context of LLMs increasing the temperature results in more creative answers, while keeping it at zero completely grounds it in the most probable output. In the case of this thesis the parameter is set to zero due to the IR-Anthology being grounded in scientific work and requiring it to be as precise as possible.

The parameter max new tokens plays a crucial role in controlling the length of text generated by the LLM. It determines the maximum number of new tokens the model will add to the provided context. This interaction between max new tokens and the length of the context is pivotal, with the latter establishing the starting point for text generation.

Higher values of max new tokens allow the LLM to produce longer and more detailed outputs, potentially offering comprehensive responses or narratives. Conversely, lower values result in concise summaries or shorter creative compositions.

In the Mistral 7B instructional LLM configuration, the parameter max new tokens was established at 128 for question generation and 512 for answer generation as seen in Table 4.2. Additionally, the context length was specified as 2048. For Prometheus 13B, a context length of 2048 was selected, along with a maximum new context length of 1024 (4.3), to facilitate a more detailed explanation for the evaluation to be discussed later in this chapter.

Table 4.2: Mistral 7B Instruct parameters.

Mistral 7B Instruct	Temperature: 0 Context Length: 2048 Max New Tokens: 128

Table 4.3: Prometheus 13B parameters.

Prometheus 13B	Temperature: 0 Context Length: 2048 Max New Tokens: 1024

4.1.2 Question Generation

The question generation process necessitates the utilization of both an index employing the requisite parsing method and an LLM. In this instance, Mistral 7B - instruct served as the selected LLM.

The procedure commences with an iterative traversal of the index. For each chunk contained within the index, the chunk is input into a prompt, subsequently fed into the LLM. This prompt functions as explicit guidance to the LLM, directing its actions based on the provided contextual information. The specific prompt employed is depicted in Figure 4.1.

```
"""
Context information is below.
---------------------
{context_str}
---------------------
Given the context information and not prior
knowledge.

Your task is to setup a question for an upcoming
quiz/examination. Restrict the question to the
context information Provided
"""
```

Figure 4.1: Question generation prompt.

Upon completion of the question generation process, the resulting inquiries are archived in a JSON file, characterized by three primary keys: "queries," "corpus," and "relevant chunks." Within this structure, the "queries" key encapsulates the questions generated by the LLM in response to the provided chunk within the prompt. Each query is uniquely identified by its UUID. The

"corpus" key represents the chunks stored within the index, with each individual chunk assigned its own UUID. The "relevant chunks" key establishes connections between the generated queries and the corresponding chunks through the utilization of UUIDs. A representative example of a singular question and its associated chunk pair is illustrated in Figure 4.2.

```
{
“query”: {
          “124b5fc3-813d”: QUERY
         }

”context”: {
          ”fba63e25-458a”: CHUNK
          },

“relevant_chunks”: {
                      “124b5fc3-813d”: ”fba63e25-458a”
                   },
}
```

Figure 4.2: Question and chunk pair in a JSON file.

4.1.3 Answer Generation

The initiation of answer generation involves the creation of a new JSON file, incorporating the preceding questions and chunks, excluding the UUIDs. Subsequently, this composite dataset undergoes processing through a prompt that incorporates both the question and chunk inputs, facilitating the generation of an answer. The resulting answer is then saved into a dedicated JSON file for further reference and utilization. The prompt used for the answer generation can be observed in Figure 4.3. In Figure 4.4 the JSON structure is presented.

```
"""
Context information is below.
---------------------
{context_str}
---------------------
Given the context information and not prior
knowledge.
generate only questions based on the below
query.
{query_str}
"""
```

Figure 4.3: Answer generation prompt.

```
”examples”: {

“query”: {
                QUERY
          },
“query_by”: {
                “model_name”: “TheBloke/Mistral-7B-Instruct-v0.2-AWQ”,
                “type”:”ai”
            },
”reference_contexts”: {
                          CHUNK
                        },
“answer”: {
                ANSWER
           },

”answer_by”: {
                “model_name”: “TheBloke/Mistral-7B-Instruct-v0.2-AWQ”,
                “type”: “ai”
             }
}
```

Figure 4.4: Question, context, and answer in a JSON file.

4.1.4 Document Selection

The challenge encountered in assessing various RAG pipelines across the IR-Anthology dataset primarily revolves around its considerable size. The dataset encompasses approximately 62,000 research papers, presenting a significant logistical hurdle in constructing a comprehensive index for evaluation purposes.

Consequently, a pragmatic approach was adopted, wherein a smaller subset of 1,000 documents was selected for analysis. This subset represents a diverse array of research papers sourced from multiple conferences and journals.

Analysis of the data, as depicted in Tables 4.4, 4.5, and 4.6, underscores the impact of chunk size on various pipeline operations. Notably, the time required for tasks such as vector index generation, question generation, and answer generation exhibits substantial escalation as the chunk size decreases. It is noteworthy that the indexing duration for SW 3 and SW 6 remains identical, as both processes entail parsing and encoding an equivalent number of chunks (sentences), albeit with differing metadata. However, divergence emerges during question and answer generation, owing to variations in the chunk sizes utilized for these operations, with SW 6 accommodating larger chunks compared to SW 3.

Parsing Method	**Duration (Hours)**
TB 1024	10:02
TB 512	17:49
TB 256	39:16
SW 3 & 6	48:32

Table 4.4: Duration of parsing, encoding, and indexing for each method for dataset preparation.

Method	**Question Generation Time (Hours)**
TB 1024	6:27
TB 512	12:42
TB 256	29:31
SW 3	42:21
SW 6	46:59

Table 4.5: Time required to generate questions.

Method	Answer Generation Time (Hours)
TB 1024	9:18
TB 512	16:35
TB 256	32:12
SW 3	44:25
SW 6	48:11

Table 4.6: Time required to generate answers.

4.2 Retrieval Evaluation

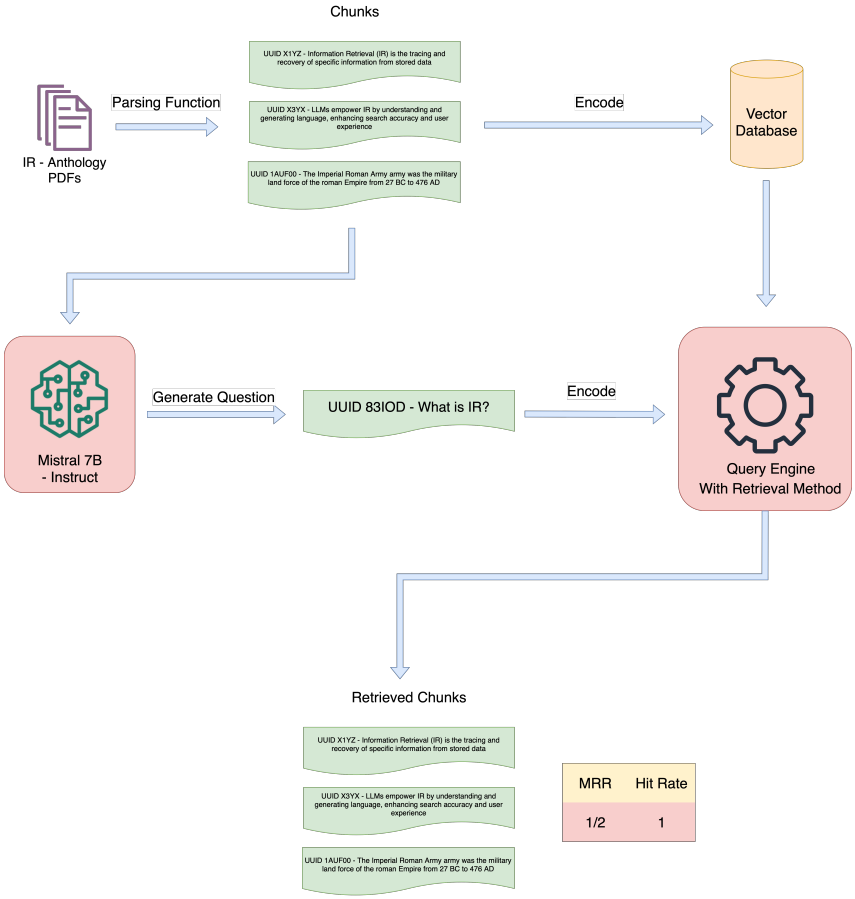

Figure 4.5: Question generation to evaluation

This section will delineate the retrieval evaluation process, elucidate the metrics employed for assessment, and detail the various configurations implemented.

4.2.1 Flow

Building upon the preceding section outlining the generation and storage of question evaluations in JSON format, this subsection delves into subsequent steps. Following the completion of the question generation setup, the queries undergo processing by the query engine to obtain the top_k results. As emphasized in the earlier chapter, our evaluations exclusively focus on the top_3 chunks. Once retrieved, these chunks undergo evaluation based on two metrics: Mean Reciprocal Rank (MRR) and Hit Rate. Figure 4.5 visualizes the end to end flow of retrieval evaluation.

4.2.2 Evaluation

Within the RAG pipeline, the objective is to assess two key elements: the parsing method, encoder (embedding model), and the retrieval method. This examination aims to scrutinize the impact that tuning each component will have on the retrieval setup.

Initiating the evaluation process commences with the scrutiny of three pivotal parsing components. Specifically, the token-based parsing is to be examined at three distinct sizes: 256, 512, and 1024.

Subsequently, an appraisal of retrieval methods is scheduled. This encompasses the evaluation of BM25 (sparse), HNSW (dense), a hybrid approach (combining sparse and dense methods), a reranker, and HyDE.

In the context of vector retrieval, the designated distance metric is the cosine similarity function. Equation 4.2, and 4.3 explicitly denotes the formulation of the cosine similarity function.

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\| \cdot \|\mathbf{B}\|} \tag{4.2}$$

$$\text{Cosine Similarity}(\mathbf{A}, \mathbf{B}) = \frac{\sum_{i=1}^{n} A_i \cdot B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \cdot \sqrt{\sum_{i=1}^{n} B_i^2}} \tag{4.3}$$

- $\mathbf{A}$ and $\mathbf{B}$ are vectors,
- $\|\mathbf{A}\|$ and $\|\mathbf{B}\|$ represent the Euclidean norms of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively.

The exploration of various configurations unfolded as follows: Initially, the fundamental retrieval methods—BM25, HNSW, and Hybrid—were individually assessed in their standard formats across distinct parsing methodologies.

Subsequently, a reranker was incorporated into each method. The entire process was reiterated, this time incorporating HyDE into the evaluation for a comprehensive analysis.

The schematic representation in Figure 4.6 illustrates the integration of HyDE within the evaluation flow as well as the prompt used in Figure 4.7. Prior to entering the query engine, the query undergoes a re-writing process facilitated by the LLM. Subsequently, the flow continues in a conventional manner.

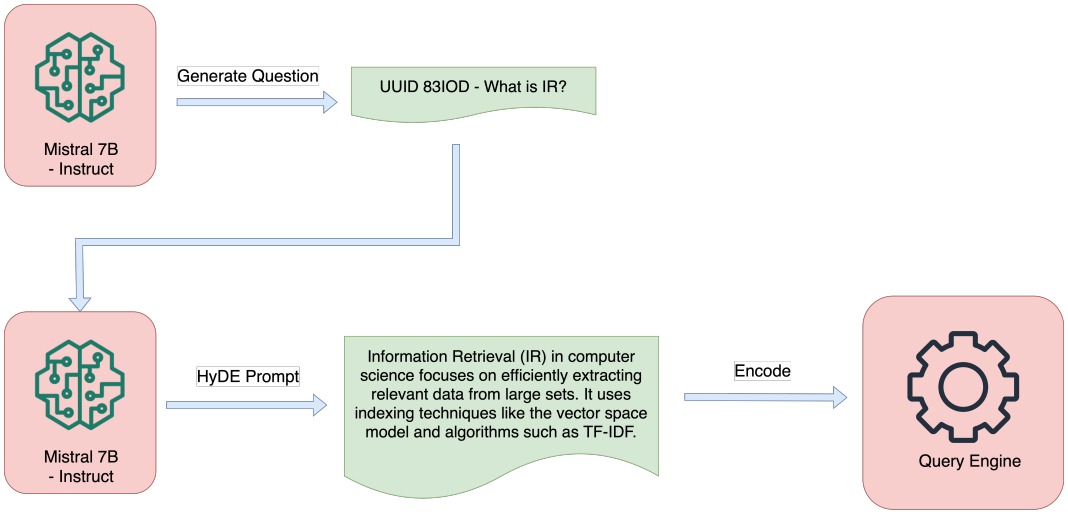

Figure 4.6: HyDE before flowing into the query engine.

```
"Please write a passage to answer the question\n"
"Try to include as many key details as
possible.\n"

"{context_str}\n"

'Passage:'
```

Figure 4.7: HyDE prompt.

4.2.3 Metrics

Mean Reciprocal Rank:

Mean Reciprocal Rank (MRR) is a metric used to evaluate the effectiveness of a ranking algorithm in information retrieval and recommendation systems. In Equation 4.4 it measures the quality of the ranking by considering the position of the first relevant item in the list. MRR is particularly useful when dealing with ranked lists, where the goal is to present the most relevant items at the top.

$$MRR = \frac{1}{|Q|} \sum_{i=1}^{|Q|} \frac{1}{R_i} \tag{4.4}$$

- $|Q|$ is the total number of queries,
- R_i is the rank of the first relevant item for the i-th query.

Hit Rate:

Hit Rate is a performance metric commonly used in information retrieval and recommendation systems to assess the effectiveness of a model or algorithm in finding relevant items. It measures the proportion of queries or instances for which at least one relevant item was found in the result set. The Hit Rate is expressed as the ratio of the number of relevant items found to the total number of queries.

4.2.4 Results

Table 4.7 illustrates noteworthy trends in Hit Rate and MRR across various Chunk Sizes (256, 512, and 1024) and Retrieval Methods, including Sparse with BM25 (S), Dense with HNSW (D), with Reranker (RR), and Hybrid (H). Remarkably, the combination of Chunk Size 1024 and Retrieval Method Hybrid consistently delivers superior performance, boasting the highest Hit Rate of 0.92 and commendable MRR values. Sparse with BM25, especially with chunk size 256, also demonstrates competitive outcomes. The incorporation of Reranker in certain scenarios proves beneficial, as evidenced by improved performance. Larger chunk sizes generally exhibit higher Hit Rates and MRR.

Table 4.7: Results of different retrieval methods.

Chunk Size	Retrieval Method	Hit Rate	Mean Reciprocal Rank
256	S	0.83	0.778
256	S, RR	0.83	0.731
256	D	0.79	0.741
256	D, RR	0.79	0.721
256	H	0.86	0.806
512	S	0.88	0.805
512	S, RR	0.88	**0.811**
512	D	0.79	0.708
512	D, RR	0.79	0.721
512	H	0.801	0.701
1024	S	0.89	0.806
1024	S, RR	0.89	0.725
1024	D	0.74	0.668
1024	D, RR	0.74	0.661
1024	H	**0.92**	0.798

In a parallel fashion, Table 4.8 depicts the outcomes derived from employing HyDE prior to submitting the query to the query engine. Nevertheless, discernibly, there is a conspicuous decline in performance associated with this methodology. In contrast, subsequent to the application of HyDE, HNSW retrieval exhibits a slight improvement over BM25, suggesting that BM25 performs slightly less favorably in this context. This improvement may be ascribed to the supplementary details and information incorporated into the query, thereby facilitating a closer alignment between the query and the target retrieval point within the vector space. During the evaluation of HyDE, the inclusion of a reranker was deemed unnecessary. This determination stemmed from the realization that integrating an additional reranker would necessitate four forward propagations of transformers leading a very long processing time (more than 20 seconds). In a finalized pipeline, the sequence of operations would manifest as follows:

1. Transformation of HyDE queries utilizing an **LLM**.
2. Encoding of queries employing an **Embedding Model**.
3. Retrieval and subsequent reranking of chunks facilitated by a **Cross-Encoder**.

4. Synthesis of queries and chunks accomplished through the utilization of an **LLM**.

Table 4.8: Results of different retrieval methods with HyDE.

Chunk Size	Retrieval Method	Hit Rate	Mean Reciprocal Rank
256	S	0.68	0.6
256	D	**0.84**	**0.71**
256	H	0.78	0.66
512	S	0.64	0.59
512	D	0.7	0.57
512	H	0.66	0.57
1024	S	0.6	0.53
1024	D	0.66	0.53
1024	H	0.65	0.54

4.3 Generation Evaluation

There are various evaluation frameworks[1] [2] available for the generation evaluation; however, most of them depend on the RAG pipeline LLM being used through GPT 4's API key. Since this work depends heavily on functioning on open - source models a custom evaluation generation pipeline was built.

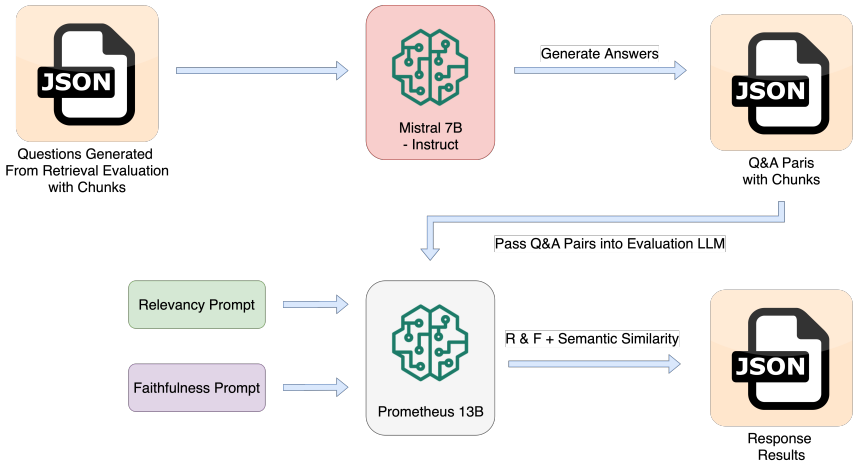

Figure 4.8: Generation and evaluation of Q&A pairs.

[1] https://github.com/explodinggradients/ragas
[2] https://github.com/truera/trulens

This section will systematically outline the process of evaluating the generation, explicate the metrics utilized for assessment, and provide comprehensive details on the diverse configurations implemented.

4.3.1 Flow

In Figure 4.8, the procedural outline of the generation evaluation process commences subsequent to the generation and compilation of question and chunk pairs into a JSON file during the retrieval evaluation phase. Following the acquisition of these question-chunk pairs, they undergo processing wherein they are inputted into the LLM (Mistral 7B - instruct) for answer generation. The resultant answers are subsequently stored within a JSON file along with the preceding contextual information.

Upon the completion of JSON file formulation, the evaluation phase ensues. Sequentially, each question-answer pair, along with its associated chunk, is subjected to assessment using the evaluating LLM (Prometheus 13B). This evaluation is conducted with regard to two primary metrics: Relevancy and Faithfulness. Furthermore, the semantic similarity between the generated answer and the corresponding chunk is determined by encoding both elements via an embedding model, followed by cosine similarity analysis on the resultant vectors.

The resulting JSON file contains feedback on the metrics of faithfulness and relevancy, alongside their respective scores. This feedback elucidates the rationale behind the assigned scores, providing insight into the evaluation process.

4.3.2 Evaluation

The primary objectives of the generation evaluation encompass assessing the LLMs, particularly Mistral 7B - instruct, in their capacity to synthesize contextual information, maintain fidelity to the provided chunk, and furnish pertinent responses to inquiries. Additionally, the evaluation aims to scrutinize how different parsing methods impact these aforementioned aspects.

In contrast to the preceding retrieval evaluation, the current evaluation extends beyond the evaluation of the conventional token-based parsing method to include an examination of the SW parsing method. Notably, the consideration of the SW parsing method is exclusive to the generation evaluation owing to its characteristic of accommodating an average token size for a SW of 3 and 6, closely approximating token sizes of 256 and 512 respectively. This decision is further justified by the observations outlined in Table 4.1, wherein the significant volume of nodes necessitates substantial computational resources for

evaluation, rendering insights gleaned from approaches closely aligned with token sizes of 256 or 512 less impactful.

4.3.3 Metrics

Three main metrics were utilized for the evaluation. Relevancy, faithfulness, and semantic similarity.

Relevancy:

The assessment of relevancy pertains to the degree to which the provided response aligns with both the query and the given context. This evaluation process involves inputting the prompt instruction into the evaluation LLM, as depicted in Figure 4.9, along with necessary inputs. Upon generating a response, the evaluation LLM yields two outcomes: firstly, a rationale explaining the determination of whether the text is relevant to the query or not, and secondly, a scoring mechanism ranging from 0 (indicating irrelevance) to 1 (indicating relevance), which is derived from the initial rationale and converted into an integer. This score is subsequently averaged across the entire dataset for comprehensive assessment.

The prompt in Figure 4.9 leverages zero-shot prompting, a technique where a LLM is instructed to perform a task without explicit training or in-prompt examples of how to complete it.

```
“””
Your task is to evaluate if the response for the query
    is in line with the context information provided.
     You have two options to answer. Either YES/ NO.
       Answer - YES, if the response for the query
    is in line with context information otherwise NO.
            Query and Response:  {query_str}
                 Context: {context_str}
                        Answer:

”””
```

Figure 4.9: Relevancy prompt.

Faithfulness:

Faithfulness serves as the primary metric in the evaluation of generation, aiming to identify any instances of inaccurate information within the generated responses. It assesses the degree to which the generated answer aligns with the provided context chunks. This evaluation process involves the provision of a few-shot prompt, which guides the evaluation LLM in determining whether the generated answer faithfully represents the content provided and effectively addresses the query. Similar to the relevancy metric, faithfulness provides both an explanatory rationale for the decision made by the evaluation LLM and an outcome, quantified as either 0 (indicating a lack of faithfulness) or 1 (indicating fidelity), derived from that explanation.

The prompt in Figure 4.10 incorporates the few-shot prompting principle. In this approach, the LLM is tasked to adapt to new tasks with minimal exemplars provided within the prompt itself.

```
“””
Please tell if a given piece of information
is supported by the context.
You need to answer with either YES or NO.
Answer YES if any of the context supports the information, even
if most of the context is unrelated.
Some examples are provided below.
Information: Apple pie is generally double-crusted.
Context: An apple pie is a fruit pie in which the principal filling
ingredient is apples.
Apple pie is often served with whipped cream, ice cream
('apple pie à la mode'), custard or cheddar cheese.
It is generally double-crusted, with pastry both above
and below the filling; the upper crust may be solid or
latticed (woven of crosswise strips).
Answer: YES\n
Information: Apple pies tastes bad.
Context: An apple pie is a fruit pie in which the principal filling
ingredient is apples.
Apple pie is often served with whipped cream, ice cream
('apple pie à la mode'), custard or cheddar cheese.
It is generally double-crusted, with pastry both above
and below the filling; the upper crust may be solid or
latticed (woven of crosswise strips).
Answer: NO
Information: {query_str}
Context: {context_str}
Answer:
”””
```

Figure 4.10: Faithfulness prompt.

Semantic Similarity:

Semantic similarity is established through the calculation of cosine similarity between the generated answer and the corresponding chunk. This process primarily aims to gauge the proximity of the generated answer to the chunk while ensuring semantic coherence is maintained. The encoding of both the chunk, and the generated answer is done through the BGE Large embedding model that was used previously for the retrieval evaluation.

4.3.4 Results

Table 4.9 presents the results of various evaluations across different chunk sizes, namely 256, 512, and 1024, as well as sentence window 3 and 6. In Figure 4.11 the resulting JSON file for a single Q&A example can be seen.

In terms of faithfulness, the chunk size of 256 exhibits the highest performance, with a score of 0.933, indicating a strong alignment between the generated answers and the provided chunks. Conversely, the chunk size of 1024 demonstrates the lowest faithfulness score of 0.585.

Regarding relevancy, the chunk size of SW 6 performs the best, achieving a score of 0.714. In contrast, the chunk size of 1024 displays the lowest relevancy score of 0.541.

In semantic similarity, the chunk size of SW 6 also emerges as the top performer, with a score of 0.547, indicating a high degree of similarity between the generated answers and the provided chunks. Conversely, the chunk size of 512 exhibits the lowest semantic similarity score of 0.480.

These preliminary findings provide insights into the varying performance of different chunk sizes across the metrics of faithfulness, relevancy, and semantic similarity. Further exploration and detailed analysis of these results will be discussed in Chapter 5.

Table 4.9: Results of different evaluations.

Chunk Size	Relevancy	Faithfulness	Semantic Similarity
256	0.653	**0.933**	0.545
512	0.674	0.879	0.511
1024	0.541	0.585	0.48
SW 3	0.689	0.91	0.53
SW 6	**0.714**	0.883	**0.547**

```
{
"relevancy": {
    "score": 1.0,
    "query": QUESTION,
    "contexts": CHUNKS,
    "response": ANSWER,
    "passing": true,
    "feedback": The text is relevant because of.."
        },
"faithfulness":{
        "score": 1.0,
        "query": QUERY,
        "contexts": CHUNKS,
        "response": ANSWER,
        "passing": true,
        "feedback": "The text is faithful because of.."
        },
"semantic_similarity": {
        "score": 0.47,
        "query": QUESTION,
        "contexts": CHUNKS,
        "response": ANSWER,
        "feedback": "Similarity score: 0.47
        },
}
```


Figure 4.11: Evaluation response JSON.

4.4 Chapter Conclusion

This chapter delves into the comprehensive evaluation process of the RAG pipeline. It begins by outlining the rationale behind synthetic data generation and subsequently establishes a two-step evaluation approach.

The initial step meticulously evaluates the retrieval component across various parsing methods and retrieval algorithms. To assess retrieval effectiveness, two key metrics were employed: MRR and Hit Rate. The evaluation revealed that the Hybrid retrieval method emerged as the most suitable choice for the utilized IR-Anthology dataset.

The second step focuses on evaluating the generation component, again employing different parsing methods. This section establishes the evaluation metrics by detailing the utilization of Prometheus 13B for generating Faithfulness and Relevancy metrics. Furthermore, it explains the setup of the Semantic Similarity metric as a numerical value through cosine similarity. The evaluation yielded compelling results, indicating that smaller chunk sizes demonstrably improve both faithfulness and relevancy in the generated responses.

The subsequent Chapter 5 will delve deeper into potential explanations for the observed results and the rationale behind the chosen evaluation methods.

Chapter 5

Discussion & Analysis

As explored in the preceding chapters, the RAG framework offers promising solutions to the limitations encountered with standalone LLMs in NLP. Chapter 2 laid the groundwork by discussing the potential of RAG in addressing issues like hallucination, information cutoff, and domain specificity, while Chapter 3 delved into the implementation details of the RAG pipeline, highlighting key components such as the LLMs used, the quantization technique, embedding models, and the orchestration framework, Llamaindex. Furthermore, Chapter 4 provided insights into the evaluation methodologies employed to assess the performance of the RAG pipeline, particularly focusing on the two-step evaluation process for the retrieval and generation components.

This chapter aims to analyze the findings from the evaluation process and propose avenues for further exploration and improvement within the RAG framework. Following the two-step evaluation process introduced in Chapter 4, which served as an ablation study to isolate the impact of each component, this section analyzes the insights gained from each step. This analysis informs the selection of the configuration for the IR-Anthology and possible trade-offs. The discussion will first explore the findings from each evaluation stage, followed by a comprehensive overview of the potential final setup.

5.1 Analysis of Retrieval Results

For the retrieval component the results of Table 4.7 clearly shows that sparse methods (BM25), are outperforming dense methods (HNSW) throughout all the metrics (MRR, and Hit Rate). This could be attributed to the IR-Anthology having more factual and use specific terminology compared to general text. BM25 is a term-frequency based method that rewards documents with a high frequency of query terms. BM25 prioritizes documents with matching keywords, which can be effective for retrieving relevant chunks, especially for

factual queries. Dense methods like HNSW might capture more semantic relationships between chunks, but these might not be as crucial for factual searches in scientific research. However, a combination of both BM25 and HNSW resulted also in the best performing hit rate which could be attributed of getting the best of both worlds from both the algorithms.

Chunk size exhibited a significant influence on retrieval performance, with opposing effects on sparse and dense retrieval models. Sparse retrieval algorithms, like BM25, demonstrated improved performance with increasing chunk size. This can be attributed to the term frequency weighting mechanism within BM25, which benefits from a wider range of terms to assess relevance. Conversely, dense retrieval models experienced performance degradation with larger chunk sizes.

The reranker does not offer improvement to the hit rate metric as the required chunk is still within the top_k, but its ranking withint the top_k impacts the MRR metric.

The introduction of HyDE resulted as seen in Table 4.8 shows a decrease in performance across all evaluated configurations. This decline is likely attributable to the query rewriting process, which may introduce irrelevant terms ("hallucinations") during expansion. For retrieval models like BM25, which rely heavily on the presence of original query terms, this can be particularly detrimental as relevant keywords might be removed or altered. Conversely, HNSW exhibited a smaller performance drop with HyDE. This suggests that the transformed queries may have remained within a similar semantic space to the original, potentially aligning with the intended behavior of HyDE.

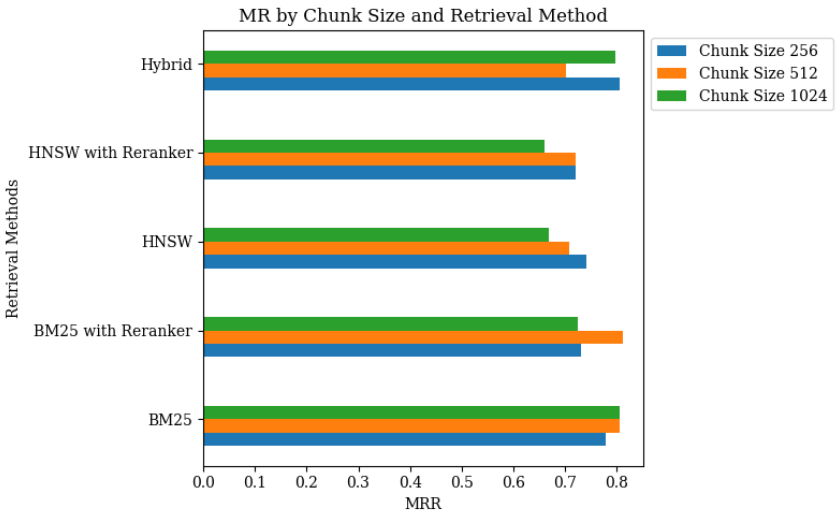

Figure 5.1: Evaluation results grouped by different chunk sizes, for each retrieval method.

In Figure 5.1 the plotting of all the grouped chunk sizes with their respective retrieval algorithms can be observed. The following are the best performing findings in terms of the retrieval method for each chunk size:

- 256 - Sparse: Hit Rate 0.83, MRR 0.778
- 512 - Sparse, with Reranker: Hit Rate 0.88, MRR 0.811
- 1024 - Hybrid: Hit Rate 0.92, MRR 0.798

5.2 Analysis of Generation Results

The results in Table 4.9 offer valuable insights into the challenges LLMs face with longer inputs. This observation can be explained by limitations in the self-attention mechanism.

As the input length increases, self-attention's effectiveness diminishes. The model might struggle to efficiently distribute its focus across the entire sequence.

By breaking down the input into smaller chunks, it essentially alleviates the limitations of the self-attention mechanism. With shorter sequences to process, the model can dedicate its focus more effectively, resulting in outputs that are both faithful to the original content and relevant to the overall context. This aligns with the findings in Table 4.9, where smaller chunk sizes yielded better generation results.

In Figure 5.2 it is observable that as soon as the chunk size increases the performance drops in all metrics.

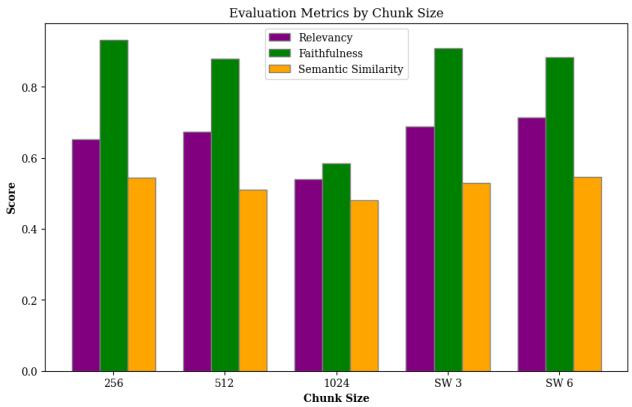

Figure 5.2: Evaluation results for the generation. Comparison bar plot for the different chunk sizes. (Sentence Window - SW)

Chapter 6

Conclusion

The objective of this thesis was to utilize RAG to develop efficient interaction between users and the IR-Anthology, this was done through experimenting multiple RAG configurations to reach the most optimal setup. Through doing that it mitigated issues faced within using standalone LLMs that suffer from hallucinations, information cutoff, and domain specificity.

Chapter Review

- Chapter 1 & Chapter 2: The groundwork was laid for understanding the potential of RAG in mitigating issues such as hallucination, information cutoff, and domain specificity. The fundamental concepts of RAG were introduced (e.g., transformers, LLMs, embedding models), emphasizing its potential to combine the strengths of RAG with the IR-Anthology to create a robust tool for researchers within the domain of IR to refer to.
- Chapter 3: Within this chapter it provided detailed insights into the implementation aspects of the RAG pipeline, including the selection of LLMs, embedding model, and retrieval methods. The explored methodologies for LLMs highlighted the importance of optimization techniques such as FlashAttention and PagedAttention.
- Chapter 4: The evaluation process discussed offered valuable insights into the performance of the RAG pipeline, particularly in terms of retrieval and generation components. Through meticulous evaluation methodologies, optimal configurations were identified, showcasing the effectiveness of techniques such as hybrid retrieval methods and smaller chunk sizes in enhancing both retrieval and generation performance.
- Chapter 5: This chapter further deepened the analysis by dissecting the retrieval and generation results and exploring explanations for observed

trends. The analysis of retrieval outcomes highlighted the significance of sparse retrieval methods like BM25, especially in the context of factual queries within scientific research. Additionally, the impact of chunk size on retrieval performance was highlighted, emphasizing the need for a balanced approach to chunk parsing.

Challenges

The thesis encountered numerous challenges primarily related to framework compatibility, largely stemming from their reliance on closed source APIs. Given the open source nature of the thesis, significant modifications were necessary to adapt these frameworks to the open source setup employed.

- Evaluation Framework: The evaluation framework had taken a substantial amount of time to setup. This is due to issues faced with closed and open source models. Most of the evaluation components had to be built from scratch.
- Computational Time: It is imperative to acknowledge the substantial time investment required for the evaluation and establishment of these evaluation pipelines as noted in chapter 4. Notably, executing these pipelines for both retrieval and generation components consumed a considerable duration, totaling 21 days of constant runtime on A100 GPUs. This prolonged computational effort underscores the complexity and resource-intensive nature of comprehensively assessing the efficacy of the RAG framework.
- Metrics: The evaluation metrics for the retriever adhere to standardized practices within the information retrieval domain, such as MRR and Hit Rate. However, assessing the generative capabilities of the LLM presents considerable challenges. Despite the utilization of feedback loops provided by the evaluation LLM to measure specific metrics like Faithfulness, determinism remains elusive.

Concluding Remarks

The analysis of generation results underscores the challenges LLMs face with longer inputs and the effectiveness of smaller chunk sizes in improving generation quality. By mitigating the limitations of the self-attention mechanism, smaller sequences enabled more focused generation, resulting in outputs that were both faithful and contextually relevant. However, this analysis

also leads to the inference that a discernible trade-off exists between retrieval efficiency and generation quality. Specifically, increasing the size of chunks enhances the performance of the retrieval process, albeit at the expense of compromising the generation capabilities of the LLM. This phenomenon can be analogized to presenting an individual with a sizable textbook and tasking them with extracting information to answer a query. While the required information may indeed be present within the extensive text, the individual faces challenges in processing the entirety of the textbook simultaneously, possibly necessitating additional research endeavors, such as fine-tuning, to adapt more effectively to its contents.

In essence, the journey of implementing RAG within the IR-Anthology framework illuminated not only the technical hurdles but also the inherent trade-offs between different aspects of the system. These challenges underscore the need for further exploration and refinement, particularly in devising strategies that strike a delicate balance between retrieval efficiency and generation quality. Future endeavors may focus on innovative approaches to optimize this trade-off. By addressing these challenges head-on, we can pave the way for more robust and versatile systems that redefine the landscape of information retrieval and generation in research domains.

Moving forward, the insights gleaned from this comprehensive analysis pave the way for further exploration and refinement of the RAG framework. Future research endeavors could focus on finetuning retrieval and generation strategies, optimizing chunk segmentation techniques, and exploring novel approaches to enhance overall performance and scalability.

Bibliography

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 Houda Bouamor, Juan Pino, and Kalika Bali, editors, *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, EMNLP 2023, Singapore, December 6-10, 2023*, pages 4895–4901. Association for Computational Linguistics, 2023. URL https://aclanthology.org/2023.emnlp-main.298.

Iz Beltagy, Matthew E. Peters, and Arman Cohan. Longformer: The long-document transformer, 2020.

Steven Bird, Robert Dale, Bonnie Dorr, Bryan Gibson, Mark Joseph, Min-Yen Kan, Dongwon Lee, Brett Powley, Dragomir Radev, and Yee Fan Tan. The ACL Anthology reference corpus: A reference dataset for bibliographic research in computational linguistics. In Nicoletta Calzolari, Khalid Choukri, Bente Maegaard, Joseph Mariani, Jan Odijk, Stelios Piperidis, and Daniel Tapias, editors, *Proceedings of the Sixth International Conference on Language Resources and Evaluation (LREC'08)*, Marrakech, Morocco, May 2008. European Language Resources Association (ELRA). URL http://www.lrec-conf.org/proceedings/lrec2008/pdf/445_paper.pdf.

Danqi Chen, Adam Fisch, Jason Weston, and Antoine Bordes. Reading wikipedia to answer open-domain questions. In Regina Barzilay and Min-Yen Kan, editors, *Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics, ACL 2017, Vancouver, Canada, July 30 - August 4, Volume 1: Long Papers*, pages 1870–1879. Association for Computational Linguistics, 2017. doi: 10.18653/V1/P17-1171. URL https://doi.org/10.18653/v1/P17-1171.

Gordon V. Cormack, Charles L. A. Clarke, and Stefan Büttcher. Reciprocal rank fusion outperforms condorcet and individual rank learning methods. In James Allan, Javed A. Aslam, Mark Sanderson, ChengXiang Zhai, and Justin Zobel, editors, *Proceedings of the 32nd Annual International ACM*

SIGIR Conference on Research and Development in Information Retrieval, SIGIR 2009, Boston, MA, USA, July 19-23, 2009, pages 758–759. ACM, 2009. doi: 10.1145/1571941.1572114. URL https://doi.org/10.1145/1571941.1572114.

Tri Dao, Daniel Y. Fu, Stefano Ermon, Atri Rudra, and Christopher Ré. Flashattention: Fast and memory-efficient exact attention with io-awareness, 2022.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. In Jill Burstein, Christy Doran, and Thamar Solorio, editors, *Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2019, Minneapolis, MN, USA, June 2-7, 2019, Volume 1 (Long and Short Papers)*, pages 4171–4186. Association for Computational Linguistics, 2019a. doi: 10.18653/v1/n19-1423. URL https://doi.org/10.18653/v1/n19-1423.

Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding, 2019b.

Matthew Dunn, Levent Sagun, Mike Higgins, V. Ugur Güney, Volkan Cirik, and Kyunghyun Cho. Searchqa: A new q&a dataset augmented with context from a search engine. *CoRR*, abs/1704.05179, 2017. URL http://arxiv.org/abs/1704.05179.

Luyu Gao, Xueguang Ma, Jimmy Lin, and Jamie Callan. Precise zero-shot dense retrieval without relevance labels, 2022.

Kelvin Guu, Kenton Lee, Zora Tung, Panupong Pasupat, and Ming-Wei Chang. REALM: retrieval-augmented language model pre-training. *CoRR*, abs/2002.08909, 2020. URL https://arxiv.org/abs/2002.08909.

Dan Hendrycks, Collin Burns, Steven Basart, Andy Zou, Mantas Mazeika, Dawn Song, and Jacob Steinhardt. Measuring massive multitask language understanding, 2021.

Albert Q. Jiang, Alexandre Sablayrolles, Arthur Mensch, Chris Bamford, Devendra Singh Chaplot, Diego de las Casas, Florian Bressand, Gianna Lengyel, Guillaume Lample, Lucile Saulnier, Lélio Renard Lavaud, Marie-Anne Lachaux, Pierre Stock, Teven Le Scao, Thibaut Lavril, Thomas Wang, Timothée Lacroix, and William El Sayed. Mistral 7b, 2023.

Seungone Kim, Jamin Shin, Yejin Cho, Joel Jang, Shayne Longpre, Hwaran Lee, Sangdoo Yun, Seongjin Shin, Sungdong Kim, James Thorne, and Minjoon Seo. Prometheus: Inducing fine-grained evaluation capability in language models, 2023.

Tom Kwiatkowski, Jennimaria Palomaki, Olivia Redfield, Michael Collins, Ankur P. Parikh, Chris Alberti, Danielle Epstein, Illia Polosukhin, Jacob Devlin, Kenton Lee, Kristina Toutanova, Llion Jones, Matthew Kelcey, Ming-Wei Chang, Andrew M. Dai, Jakob Uszkoreit, Quoc Le, and Slav Petrov. Natural questions: a benchmark for question answering research. *Trans. Assoc. Comput. Linguistics*, 7:452–466, 2019. doi: 10.1162/tacl_a_00276. URL `https://doi.org/10.1162/tacl_a_00276`.

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, 2023.

Hyun Seung Lee, Seungtaek Choi, Yunsung Lee, Hyeongdon Moon, Shinhyeok Oh, Myeongho Jeong, Hyojun Go, and Christian Wallraven. Cross encoding as augmentation: Towards effective educational text classification, 2023.

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. *CoRR*, abs/1910.13461, 2019. URL `http://arxiv.org/abs/1910.13461`.

Patrick S. H. 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 Hugo Larochelle, Marc'Aurelio Ranzato, Raia Hadsell, Maria-Florina Balcan, and Hsuan-Tien Lin, editors, *Advances in Neural Information Processing Systems 33: Annual Conference on Neural Information Processing Systems 2020, NeurIPS 2020, December 6-12, 2020, virtual*, 2020. URL `https://proceedings.neurips.cc/paper/2020/hash/6b493230205f780e1bc26945df7481e5-Abstract.html`.

Ji Lin, Jiaming Tang, Haotian Tang, Shang Yang, Xingyu Dang, Chuang Gan, and Song Han. Awq: Activation-aware weight quantization for llm compression and acceleration, 2023.

Yu. A. Malkov and D. A. Yashunin. Efficient and robust approximate nearest neighbor search using hierarchical navigable small world graphs, 2018.

Tri Nguyen, Mir Rosenberg, Xia Song, Jianfeng Gao, Saurabh Tiwary, Rangan Majumder, and Li Deng. MS MARCO: A human generated machine reading comprehension dataset. In Tarek Richard Besold, Antoine Bordes, Artur S. d'Avila Garcez, and Greg Wayne, editors, *Proceedings of the Workshop on Cognitive Computation: Integrating neural and symbolic approaches 2016 co-located with the 30th Annual Conference on Neural Information Processing Systems (NIPS 2016), Barcelona, Spain, December 9, 2016*, volume 1773 of *CEUR Workshop Proceedings*. CEUR-WS.org, 2016. URL `https://ceur-ws.org/Vol-1773/CoCoNIPS_2016_paper9.pdf`.

OpenAI. GPT-4 technical report. *CoRR*, abs/2303.08774, 2023. doi: 10.48550/ARXIV.2303.08774. URL `https://doi.org/10.48550/arXiv.2303.08774`.

Ajay Patel, Bryan Li, Mohammad Sadegh Rasooli, Noah Constant, Colin Raffel, and Chris Callison-Burch. Bidirectional language models are also few-shot learners. In *The Eleventh International Conference on Learning Representations, ICLR 2023, Kigali, Rwanda, May 1-5, 2023*. OpenReview.net, 2023. URL `https://openreview.net/pdf?id=wCFB37bzud4`.

Martin Potthast, Sebastian Günther, Janek Bevendorff, Jan Philipp Bittner, Alexander Bondarenko, Maik Fröbe, Christian Kahmann, Andreas Niekler, Michael Völske, Benno Stein, and Matthias Hagen. The information retrieval anthology. In Fernando Diaz, Chirag Shah, Torsten Suel, Pablo Castells, Rosie Jones, and Tetsuya Sakai, editors, *SIGIR '21: The 44th International ACM SIGIR Conference on Research and Development in Information Retrieval, Virtual Event, Canada, July 11-15, 2021*, pages 2550–2555. ACM, 2021. doi: 10.1145/3404835.3462798. URL `https://doi.org/10.1145/3404835.3462798`.

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. *J. Mach. Learn. Res.*, 21:140:1–140:67, 2020. URL `http://jmlr.org/papers/v21/20-074.html`.

Roshan Rao, Joshua Meier, Tom Sercu, Sergey Ovchinnikov, and Alexander Rives. Transformer protein language models are unsupervised structure learners. In *9th International Conference on Learning Representations,*

ICLR 2021, Virtual Event, Austria, May 3-7, 2021. OpenReview.net, 2021. URL https://openreview.net/forum?id=fylclEqgvgd.

Adam Roberts, Colin Raffel, and Noam Shazeer. How much knowledge can you pack into the parameters of a language model? In Bonnie Webber, Trevor Cohn, Yulan He, and Yang Liu, editors, *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing, EMNLP 2020, Online, November 16-20, 2020*, pages 5418–5426. Association for Computational Linguistics, 2020. doi: 10.18653/V1/2020.EMNLP-MAIN.437. URL https://doi.org/10.18653/v1/2020.emnlp-main.437.

Stephen E. Robertson and Hugo Zaragoza. The probabilistic relevance framework: Bm25 and beyond. *Foundations and Trends in Information Retrieval*, 3(4):333–389, 2009.

Shawon Sarkar, Maryam Amirizaniani, and Chirag Shah. Representing tasks with a graph-based method for supporting users in complex search tasks. In Jacek Gwizdka and Soo Young Rieh, editors, *Proceedings of the 2023 Conference on Human Information Interaction and Retrieval, CHIIR 2023, Austin, TX, USA, March 19-23, 2023*, pages 378–382. ACM, 2023. doi: 10.1145/3576840.3578279. URL https://doi.org/10.1145/3576840.3578279.

Noam Shazeer. GLU variants improve transformer. *CoRR*, abs/2002.05202, 2020. URL https://arxiv.org/abs/2002.05202.

Kurt Shuster, Spencer Poff, Moya Chen, Douwe Kiela, and Jason Weston. Retrieval augmentation reduces hallucination in conversation. In Marie-Francine Moens, Xuanjing Huang, Lucia Specia, and Scott Wen-tau Yih, editors, *Findings of the Association for Computational Linguistics: EMNLP 2021, Virtual Event / Punta Cana, Dominican Republic, 16-20 November, 2021*, pages 3784–3803. Association for Computational Linguistics, 2021. doi: 10.18653/v1/2021.findings-emnlp.320. URL https://doi.org/10.18653/v1/2021.findings-emnlp.320.

Shamane Siriwardhana, Rivindu Weerasekera, Tharindu Kaluarachchi, Elliott Wen, Rajib Rana, and Suranga Nanayakkara. Improving the domain adaptation of retrieval augmented generation (RAG) models for open domain question answering. *Trans. Assoc. Comput. Linguistics*, 11:1–17, 2023. URL https://transacl.org/ojs/index.php/tacl/article/view/4029.

Jianlin Su, Murtadha H. M. Ahmed, Yu Lu, Shengfeng Pan, Wen Bo, and Yunfeng Liu. Roformer: Enhanced transformer with rotary position embedding. *Neurocomputing*, 568:127063, 2024. doi: 10.1016/J.NEUCOM.2023.127063. URL https://doi.org/10.1016/j.neucom.2023.127063.

James Thorne, Andreas Vlachos, Christos Christodoulopoulos, and Arpit Mittal. FEVER: a large-scale dataset for fact extraction and verification. In Marilyn A. Walker, Heng Ji, and Amanda Stent, editors, *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, NAACL-HLT 2018, New Orleans, Louisiana, USA, June 1-6, 2018, Volume 1 (Long Papers)*, pages 809–819. Association for Computational Linguistics, 2018. doi: 10.18653/V1/N18-1074. URL https://doi.org/10.18653/v1/n18-1074.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton-Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurélien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models. *CoRR*, abs/2307.09288, 2023. doi: 10.48550/arXiv.2307.09288. URL https://doi.org/10.48550/arXiv.2307.09288.

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. *CoRR*, abs/1706.03762, 2017. URL http://arxiv.org/abs/1706.03762.

Alex Wang, Yada Pruksachatkun, Nikita Nangia, Amanpreet Singh, Julian Michael, Felix Hill, Omer Levy, and Samuel R. Bowman. Superglue: A stickier benchmark for general-purpose language understanding systems. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 3261–

3275, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/4496bf24afe7fab6f046bf4923da8de6-Abstract.html.

Boxin Wang, Chejian Xu, Shuohang Wang, Zhe Gan, Yu Cheng, Jianfeng Gao, Ahmed Hassan Awadallah, and Bo Li. Adversarial GLUE: A multi-task benchmark for robustness evaluation of language models. In Joaquin Vanschoren and Sai-Kit Yeung, editors, *Proceedings of the Neural Information Processing Systems Track on Datasets and Benchmarks 1, NeurIPS Datasets and Benchmarks 2021, December 2021, virtual*, 2021. URL https://datasets-benchmarks-proceedings.neurips.cc/paper/2021/hash/335f5352088d7d9bf74191e006d8e24c-Abstract-round2.html.

Shitao Xiao, Zheng Liu, Peitian Zhang, and Niklas Muennighoff. C-pack: Packaged resources to advance general chinese embedding, 2023.

Biao Zhang and Rico Sennrich. Root mean square layer normalization. In Hanna M. Wallach, Hugo Larochelle, Alina Beygelzimer, Florence d'Alché-Buc, Emily B. Fox, and Roman Garnett, editors, *Advances in Neural Information Processing Systems 32: Annual Conference on Neural Information Processing Systems 2019, NeurIPS 2019, December 8-14, 2019, Vancouver, BC, Canada*, pages 12360–12371, 2019. URL https://proceedings.neurips.cc/paper/2019/hash/1e8a19426224ca89e83cef47f1e7f53b-Abstract.html.

Yue Zhang, Yafu Li, Leyang Cui, Deng Cai, Lemao Liu, Tingchen Fu, Xinting Huang, Enbo Zhao, Yu Zhang, Yulong Chen, Longyue Wang, Anh Tuan Luu, Wei Bi, Freda Shi, and Shuming Shi. Siren's song in the AI ocean: A survey on hallucination in large language models. *CoRR*, abs/2309.01219, 2023. doi: 10.48550/arXiv.2309.01219. URL https://doi.org/10.48550/arXiv.2309.01219.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric P. Xing, Hao Zhang, Joseph E. Gonzalez, and Ion Stoica. Judging llm-as-a-judge with mt-bench and chatbot arena, 2023.