Alma Mater Studiorum · University of Bologna

DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING
Master's Degree in Computer Science

Proposal for industry RAG evaluation: Generative Universal Evaluation of LLMs and Information retrieval

Supervisor:
Chiar.mo Prof.
Fabio Tamburini

Co-Supervisor:
Simone Sensidoni

Presented by:
Gianluca Gueli

Session III

Academic Year 2023/2024

"La coscienza non può essere spiegata in termini fisici

e nei termini di nessun'altra cosa."

Erwin Schrödinger

Contents

List of Figures

Sommario

Questa tesi rappresenta il risultato della mia esperienza di stage presso Bitapp, un'azienda di sviluppo software con sede a Bologna. Il progetto si è focalizzato sulla progettazione e implementazione di un chatbot basato sulla Retrieval-Augmented Generation (RAG), con l'obiettivo di sviluppare un insegnante virtuale per una piattaforma di e-learning destinata a uno dei clienti dell'azienda. La metodologia RAG consente di fornire un contesto pertinente in risposta alle domande degli utenti, utilizzando tale contesto come input per un modello di linguaggio di grandi dimensioni (LLM), il quale genera risposte appropriate.

Questa ricerca analizza i limiti intrinseci dei modelli linguistici di grandi dimensioni, in particolare nel contesto della loro capacità di rispondere a query che non rientrano nei set di dati utilizzati per il loro addestramento. Questo fenomeno, noto come "allucinazione", si verifica quando i LLM producono informazioni errate o completamente inventate. Inoltre, la presente tesi affronta la sfida di valutare il sistema RAG in assenza di benchmark preesistenti.

Lo sviluppo di sistemi RAG richiede un processo di valutazione rigoroso e sistematico. Tale processo implica la progettazione meticolosa di esperimenti, che comprende la selezione attenta di set di dati, modelli e metriche di valutazione adeguati, nonché l'esecuzione di valutazioni delle prestazioni e l'analisi dei dati risultanti. Per le aziende che intendono implementare un sistema RAG, è fondamentale condurre un'analisi approfondita dei requisiti specifici, superando i limiti dei convenzionali benchmark accademici e includendo un'analisi comparativa di vari progetti e architetture.

Per affrontare questa problematica, è stato sviluppato un approccio per generare un benchmark nelle condizioni critiche per l'azienda. La ricerca presenta una valutazione del sistema di recupero e un'indagine sulla relazione tra la dimensione dei chunk e il modello di embedding, con il calcolo della metrica del tasso di successo. Inoltre, è stata condotta un'analisi empirica per valutare la qualità delle query generate nei benchmark. Per facilitare l'automazione delle fasi di generazione e valutazione dei benchmark, è stata creata una repository su GitHub.

Introduction

This thesis represents the work of my internship experience at Bitapp, a software development company based in Bologna. The work undertaken centred on the design and implementation of a chatbot based on Retrieval-Augmented Generation (RAG). The objective was to develop a virtual teacher on an e-learning platform for one of the company's customers.

The RAG methodology provides a pertinent context in response to user queries, use this context as a prompt for an LLM, which generates the appropriate responses. This thesis examines the inherent limitations of large language models in responding to queries that fall outside the scope of their training datasets. This phenomenon, known as *hallucination*, occurs when LLMs are capable of generating incorrect or entirely invented information.

Moreover, this research addresses the challenge of evaluating the RAG system without the availability of pre-existing benchmarks. The development of RAG systems requires a comprehensive and systematic evaluation process. This process encompasses the meticulous design of experiments, which includes the careful selection of suitable datasets, models, and evaluation metrics, as well as the execution of performance assessments and the subsequent analysis of the resulting data. For companies seeking to implement a RAG system, it is imperative to conduct a thorough assessment of their specific requirements. This assessment must go beyond the constraints of conventional academic benchmarks and involve a comparative analysis of various architectural designs and models [1].

In order to address this issue, a generational approach was developed for the purpose of benchmarking under business-critical conditions. The research presents an evaluation of the retrieval system and an investigation into the relationship between chunk size and embedding model, with the calculation of the hit rate metric. Additionally, an empirical evaluation was conducted to assess the quality of the queries generated in the benchmarks. To facilitate the automation of the benchmark generation and evaluation phase, a GitHub repository was established.

This thesis work is organized as follows. Section 1 provides a quick brief of LLM and their architecture. Section 2 presents a detailed examination of the concept of Retrieval-Augmented Generation (RAG), offering an in-depth analysis of its constituent elements, including five distinct levels of chunking strategies and the function of word embeddings and vector storage in RAG. Section 3 examines the evaluation aspects of RAG systems, outlining the current frameworks available for conducting such evaluations and the associated metrics. Section 4 presents the generation approach developed during the internship, along with the results and commentary on the performance of the embedding model and query generation evaluation. Finally, section 5 concludes this work.

Chapter 1

Large Language Models

1.1 Use of LLM

Language represents a fundamental aspect of communication for humans and their interaction with machines [2]. As Weiznenbaum highlight just in 1975 a machine specialized in a specific task is a machine that does this task and only this task. The human, instead, understand complex instruction that are expressed in natural language. The instructions are less precise and ambiguous than those typically found in programming languages [3]. So, the increasing demand for machines capable of handling complex language tasks, such as translation, summarization, information retrieval, conversational interactions, and others, has given rise to the necessity for the development and utilization of generalized models. The recent advances in language models can be attributed primarily to three factors: firstly, the advent of transformers; secondly, the enhancement of computational capabilities; and thirdly, the accessibility of extensive training data. These developments have contributed to notable advancements in the field of language models [2].

1.1.1 Definition and brief history

The concept of a language model is based on the statistical modeling of word sequences, with the objective of predicting the probability of their occurrence in the future. Fundamentally, a language model is a **statistical tool** which assigns probabilities to a range of possible sequences of words, thus providing an assessment of the likelihood of a given sequence being used in a specific language [4].

Since their introduction in 2017, transformer language models have significantly transformed the field of natural language processing (NLP). Recent research and increased public interest have demonstrated that large language models are capable of achieving exceptional performance on both standard NLP benchmarks and open-ended natural language generation tasks. Such models are already being utilised in a number of industrial contexts, including web search, chatbots and the analysis of medical and financial documents [5].

Research in language modelling (LM) has attracted considerable attention and has progressed through four distinct phases of development. The initial stage of LM development was characterized by the emergence of statistical language models (SLMs), including n-gram models. These models estimate the likelihood of the subsequent word in a sequence by analysing the frequency of preceding n-grams. For instance, a bigram model is based on the frequency of word pairs and is used to predict the probability of the subsequent word.

The second stage involved the introduction of neural language models (NLMs), which employ neural networks to predict the probability distribution of the next word based on the preceding words in a sequence. This approach frequently incorporates Recurrent Neural Networks (RNNs) and their variants, including Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs). However, recurrent neural networks (RNNs) present a number of challenges, including slow computational times for long sequences, the occurrence of vanishing or exploding gradients, and difficulties in accessing information from points in time that are temporally distant [6].

To overcome these issues, the third stage of LM development saw the advent of contextualised word embeddings, which are referred to as pre-trained language models (PLMs). These models employ neural networks to generate vector representations of words, reflecting their meanings and contexts within sentences or texts. The fourth and final stage comprises the development of Large Language Models (LLMs), which are large-scale pre-trained models capable of executing a variety of natural language processing (NLP) tasks with remarkable efficacy. Models such as GPT-3 and GPT-4 are trained on extensive datasets and can be fine-tuned for specific downstream applications, including language translation and question answering [4].

1.2 Transformers

The Transformer is a deep learning model based on an attention mechanism for processing sequence data, which has been demonstrated to be an effective approach for solving complex natural language processing problems. The model was first proposed in 2017 and replaced the traditional recurrent neural network architecture in machine translation tasks, becoming the state-of-the-art model at that time. Given its compatibility with parallel computing and sophisticated design, the Transformer model has demonstrated superior performance compared to the previously prevalent recurrent neural networks, particularly in terms of accuracy and performance [7].

The LLM operates on discrete units of textual data; thus, a tokenization procedure is required. This represents a fundamental stage in the preprocessing of text, whereby the text is parsed into non-decomposable units, termed tokens. The specific nature of these tokens may vary, encompassing characters, subwords, symbols, or words, contingent on the particular tokenisation process employed. The most commonly employed tokenisation schemes in LLMs are wordpiece, byte pair encoding (BPE), and unigramLM [2].

1.2.1 Word Embeddings and position encodings

The next step is to map the tokens to numerical values, thereby converting the discrete linguistic units into continuous vector representations. This transformation enables the model to more effectively discern intricate semantic relationships between words. It is important to note that these numerical representations are not fixed; rather, they serve as parameters that the model learns to adjust, in order to accurately reflect the meaning of words [6].

The lack of recurrent connections inherent to the transformer architecture precludes its ability to represent the notion of token order that is intrinsic to recurrent neural networks. To address this issue, positional embedding is employed, whereby a unique vector is assigned to each token position within the input sequence. The positional embeddings are then appended to the word embedding. The absolute Positional Encoding method entails the generation of distinctive positional embeddings for each position and dimension through the incorporation of sine and cosine functions, which are subsequently integrated with word embeddings. This approach enables the model to identify the sequential information of words within a sentence. In contrast, relative positional encoding method is oriented towards the relative distances between words, thereby effectively capturing positional relationships, especially in long sequences. The selection of an appropriate encoding method is dependent upon the specific application and the design of the model in question [7].

Overall, the input embeddings capture the meaning of the word, while the positional encoding provides information about the word's position within the sentence. The following step involves the utilization of self-attention, which enables the model to establish relationships between words.

1.2.2 Self-attention

Indeed, the objective of the attention mechanism is to identify a small subset of relevant data from a vast corpus of information, while filtering out the vast majority of inconsequential data. As variant of the attention mechanism, it reduces the dependency on external information, thereby facilitating more effective identification of internal relationships within data or features. When applied to text, the self-attention mechanism is used to calculate the mutual influence between words, thus effectively addressing the challenge of long-range dependencies [7].

The fundamental equation for key-value attention is as follows:

$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right) V \tag{1.1}$$

The input consist of queries and keys, each characterized by a dimensionality of d_k. The computation entails the determination of the dot products between the queries and all corresponding keys, followed by the normalisation of these products through division by the square root of the dimensionality of the keys, denoted by d_k. Subsequently, a softmax function is applied in order to derive the weights associated with the values. In practical applications, the attention function is calculated in parallel for a set of queries, which are organised into a matrix denoted as Q. Similarly, the keys and values are consolidated into matrices, designated as K and V, respectively [8].

So, self-attention allows the model to **weigh** the importance of different words in a sentence when predicting a particular word. It calculates a weighted sum of the values of all words in the sentence, where the weights are determined by the relevance of each word to the target word.

Rather than employing a single attention function using keys, values, and queries, the researchers identified advantages in linearly projecting the queries, keys, and values h times using distinct, learned linear projections to dimensions d_k, d_k, and d_v, respectively. Subsequently, the attention function is applied in parallel to each of these projected representations of queries, keys, and values, resulting in the generation of output values of dimensionality d_v. The implementation of this multi-head attention mechanism enables the model to simultaneously attend to information from diverse representation sub-spaces across different positions [8].

The different types of attention heads are capable of focusing attention on different aspects of the input, thus enabling the capture of even the most complex and diverse dependencies and patterns [7].

1.2.3 Architectures

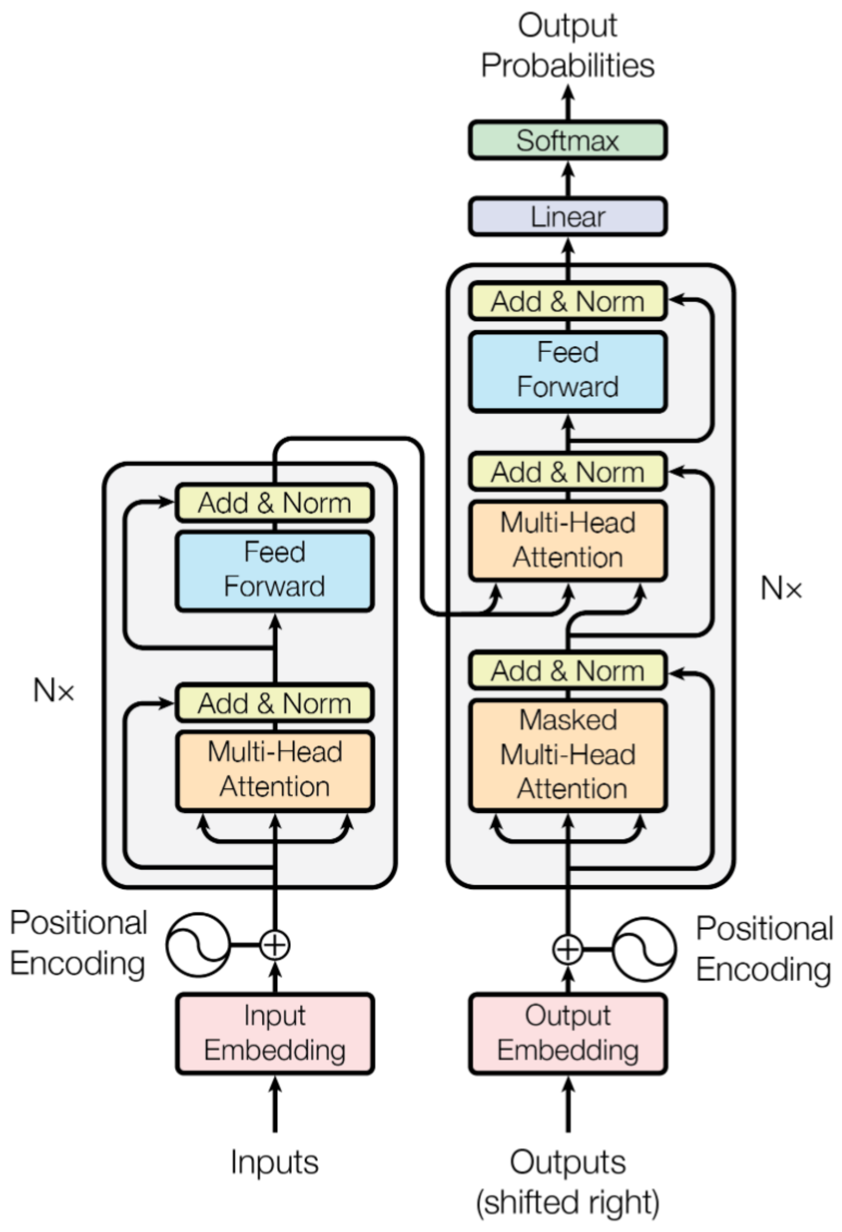


Figure 1.1: Transformer architecture proposal by vaswani et al. [8]

The transformer architecture is based on an encoder-decoder structure. The encoder transforms an input sequence of symbols, represented by the vector $x = (x_1, x_2, \ldots, x_n)$, into a sequence of continuous representations, represented by the vector $z = (z_1, z_2, \ldots, z_n)$. Subsequently, the decoder utilises these continuous representations, denoted as z, to generate an output sequence of symbols, represented as $y_1, y_2, \ldots, y_m$, one symbol at a time. At each step, the model generates the next symbol based on the symbols it has already produced, thereby employing its previous outputs to inform the next generation [8].

Typically, LLM architectures fall into three categories [7]: Encoder-only, Encoder-decoder and Decoder-only.

- **Encoder-decoder:** Each component of the encoder is constituted by multiple layers of Transformer's Multi-Head Self-Attention layers, the function of which is to encode the input sequence. In contrast, the Decoder employs cross-attention over the output representation of the Encoder with the objective of generating the target sequence in an autoregressive manner. The encoder-decoder architectural approach provides the basis for a number of prominent LLMs, including T5, Flan-T5 and BART.

- **Decoder-only:** In this configuration, the model generates tokens sequentially, attending to the preceding tokens within the sequence. This architecture has been employed in a range of language generation tasks, demonstrating its efficacy across various applications, including text generation, without necessitating an explicit encoding phase. The decoder-only architecture can be further categorized into two distinct types: the *causal decoder* architecture and the *prefix decoder* architecture.

  - **Causal Decoder Architecture:** each element of the model input sequence is capable of attending only to preceding input elements and the current element itself during the decoding process. The representative LLM models are the GPT series.
  - **Prefix Decoder Architecture:** it employs a distinctive configuration of masks that enables bidirectional attention for tokens in the prefix, while maintaining unidirectional attention for the generation of subsequent tokens. This design enables the autoregressive generation of the output sequence while allowing for bidirectional

attention to be paid to the prefix tokens. Notable LLMs that employ the prefix decoder architecture include PaLM and GLM.

- **Encoder-only:** the fundamental objective of an encoder-only model is to extract meaningful context from input sequences. This context-rich representation provides a valuable resource for downstream tasks that require a comprehensive understanding of the input's semantics and nuances. In contrast to traditional encoder-decoder models, which perform both understanding and generation, encoder-only models are designed to focus exclusively on contextual encoding. This specialization is particularly advantageous in tasks such as sentiment analysis, named entity recognition, and others. An example is BERT (Bidirectional Encoder Representations from Transformers) [9].

Chapter 2

What is RAG?

2.1 Background

Large language models (LLMs) have made a substantial impact on the development of natural language processing, facilitating a diverse range of applications including text generation and question answering. They are capable of capturing a vast amount of knowledge about the world, which enables them to respond to queries without the need to access external sources [10]. However, they can exhibit "hallucinations", where they generate responses that, while sounding plausible, are incorrect or don't align with the provided information [2]. Also the integration of dynamic, external and up-to-date information remains a challenge for these models once the training process has completed. Another challenge faced by LLMs is that they lack the capacity to respond to queries regarding private business data, due to the absence of training on this subject matter. Therefore, the model lacks the requisite knowledge to respond to a highly specific question within the domain of business. In addition, the integration of dynamic, external, and current information remains a challenge for these models after the training process is complete. To overcome these challenges, Lewis et. al. [11] introduced Retrieval-Augmented Generation (RAG) that enhances LLMs by retrieving relevant document chunks from external knowledge base through semantic similarity calculation. By referencing external knowledge, RAG effectively reduces the problem of generating factually incorrect content. Its integration into LLMs has resulted in widespread adoption, establishing RAG as a key technology in advancing chatbots and enhancing the suitability of LLMs for real-world applications [12] [13].

2.2 RAG Component

A RAG system is composed primarily of two elements: an **information retrieval** and a **text generator model**. The first component is typically a dense vector index. The second component may be a pre-trained seq2seq model, such as the Bidirectional and Auto-Regressive Transformers (BART) model. The benefits of this architectural approach are that the internal knowledge of the RAG system can be rapidly modified or even updated in real-time. This enables a control over the system's knowledge base, avoiding the wastage of time and computational resources that would otherwise be required for the retraining of the entire model [14]. Therefore, a combination of parametric and non-parametric memory is utilized: a input sequence x is employed for the retrieval of text documents z and use them as additional context when generating the target sequence y. In particular a retriever $p_\eta(z|x)$ with parameters η that returns (top-K truncated) distributions over text passages given a query x and a generator $p_\theta(y_i|x, z, y_{1:i\text{-}1})$ parametrized by θ that generates a current token based on a context of the previous $i-1$ tokens $y_{1:i\text{-}1}$, the original input x and a retrieved passage z [11].

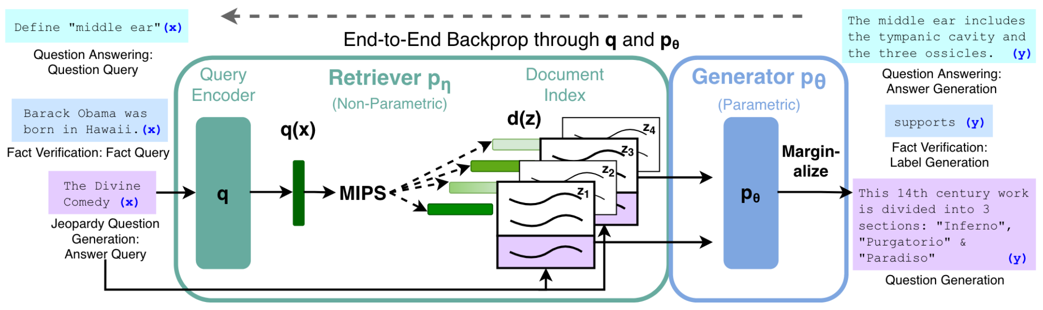

Define "middle ear" (x)
Question Answering: Question Query
Barack Obama was born in Hawaii. (x)
Fact Verification: Fact Query
The Divine Comedy (x)
Jeopardy Question Generation: Answer Query
End-to-End Backprop through q and pθ
Query Encoder
q
q(x)
Retriever pη
(Non-Parametric)
MIPS
Document Index
d(z)
z4
z3
z2
z1
Generator pθ
(Parametric)
pθ
Margin-alize
The middle ear includes the tympanic cavity and the three ossicles. (y)
Question Answering: Answer Generation
supports (y)
Fact Verification: Label Generation
This 14th century work is divided into 3 sections: "Inferno", "Purgatorio" & "Paradiso" (y)
Question Generation

Figure 2.1: RAG overview main component [11]

In other words, when a user submits a query, the Retrieval-Augmented Generation (RAG) system initiates a search within an external knowledge base with the objective of identifying the most pertinent information chunks. Subsequently, these are integrated into the input prompt of the Large Language Model (LLM), thus enabling the model to utilize the contextualized information in its response generation process.

There is also a augmentation component in RAG pipeline, that consist on iterate the process of retriever to find the lacked information. This approach has been shown to increase the robustness of subsequent response generation by providing additional contextual references through multiple retrieval iterations.

2.2.1 Retriever

The aim of this component is to conduct a search in a heterogeneous collection of data in order to identify the most pertinent information for the query. The research is performed in a vector database because all the texts are transformed into a vectorial space using an embedding model. To create this component the documents will be processed, segmented and transformed into vectors. The following section delineates the methodology employed in the creation of this component, providing a comprehensive description of the steps and roles of the various components.

2.2.1.1 Retrieval source

The RAG system start by extracting data from a knowledge base in diverse format like PDF, HTML, Word, Markdown and others. From a business perspective, it is essential to consider the entirety of the information an organisation possesses. The documents in question may be any textual sources, including PDFs, CVS file, articles, books, web pages, MD file, data from different messaging platforms, web page and others. In my internship experience, the data source was also non-textual sources (such as audio and video files), which can be transcribed into text format through the use of an automatic speech recognition (ASR) system, such as Whisper. So one of the RAG system's most notable features is that data can be sourced from any format and source because the data may be stored in a variety of formats to address specific requirements. However the use of semi-structured data, which is characterised by a combination of text and table information (e.g. PDF), presents a number of challenges for conventional RAG systems. First of all, during text splitting processes may inadvertently separate tables, resulting in data corruption in the phase of retrieval. Secondly, incorporating tables into the data can complicate semantic similarity searches. To address these challenges, a viable approach involves leveraging the coding capabilities of LLMs to execute Text-2-SQL queries on tables within databases, such as TableGPT. This approach enables the efficient processing and retrieval of semi-structured data, ultimately enhancing the overall performance of RAG systems [13].

Recently, the creation of frameworks, such as Langchain, has provided built-in tools for the development of retrieval-based pipelines, thereby facilitating the management of multiple external data sources[1]. However, during my internship, I was required to work with the XAPI format, which is the industry standard for e-learning platforms. In accordance with this standard, educational organisations and companies are able to create courses for their respective audiences.

xAPI Standard

xAPI is a data and interface standard that enables software applications to capture and share (big) data about human performance, together with associated contextual information (i.e. "experience" data). Used in conjunction with learning analytics, xAPI promises to transform the way education and training is delivered, managed and evaluated. xAPI can be integrated into

[1] The DocumentLoaders load data into the standard LangChain Document format and here all the loaders implemented in this framework.

almost any (new or existing) learning technology and is agnostic to the type of learning content being provided. "xAPI" stands for Experience Application Programming Interface. The "x" stands for experience, because xAPI enables the detailed capture and transfer of "learning experience" data, whether that data comes from an e-learning experience, a simulation-based training experience, a tablet-based training experience, or even an operational (on-the-job) experience. In software, an Application Programming Interface (API) allows two or more applications to exchange data with each other (or "talk" to each other). In the context of education and training technologies, APIs can be used to exchange data about learners and learning activities. xAPI is therefore a specific standard designed to enable the interoperable exchange of data about learner behaviour and performance. This means that a person's output is encoded into a standard format, and that the data is transported to a data repository or between applications according to standard transport rules [15].

2.2.1.2 Chunking

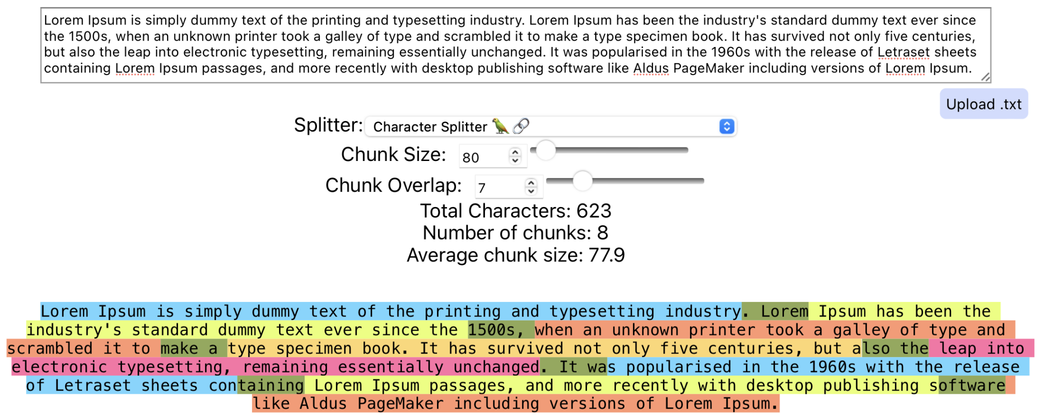
The next phase in developing RAG system is split large text into manageable chunks because of context limitation of language models but also for embedding model's capability to deal with limited input size. In fact, increase chunk size allows for more context to be captured, but also results in greater noise generation, necessitating longer processing times and higher costs [13].

The process of splitting large texts into smaller, meaningful pieces of information so that the LLM's non-parametric memory can be used more effectively is called chunking [16]. A variety of chunking strategies exist, including fixed methods based on characters, recursive approaches that balance fixed sizes and natural language structures, and advanced techniques that consider semantic topic changes. A preliminary definition of the concepts that form the basis of this discussion is a prerequisite to any analysis of the techniques of chunking:

- **Chunk Size:** The number of characters you would like in your chunks. 50, 100, 100,000, etc.

- **Chunk Overlap:** The amount you would like your sequential chunks to overlap. This is to try to avoid cutting a single piece of context into multiple pieces. This will create duplicate data across chunks.

- **Context length:** The maximum number of tokens that a Large Language Model can process in a single request.



Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software like Aldus PageMaker including versions of Lorem Ipsum.

Splitter: Character Splitter 🦜🔗

Chunk Size: 80

Chunk Overlap: 7

Total Characters: 623
Number of chunks: 8
Average chunk size: 77.9

```
Lorem Ipsum is simply dummy text of the printing and typesetting industry. Lorem Ipsum has been the
industry's standard dummy text ever since the 1500s, when an unknown printer took a galley of type and
scrambled it to make a type specimen book. It has survived not only five centuries, but also the leap into
electronic typesetting, remaining essentially unchanged. It was popularised in the 1960s with the release
of Letraset sheets containing Lorem Ipsum passages, and more recently with desktop publishing software
like Aldus PageMaker including versions of Lorem Ipsum.
```

Figure 2.2: A visual tool to better understand chunking strategies.

Chunking methods can be classified into two principal categories. The first category, **rule-based methods**, relies on the use of explicit separators, such as punctuation or space characters, or the application of sophisticated systems, like regular expressions, to partition text into chunks. The second category, **semantic clustering methods**, employs the inherent meaning embedded in the text to guide the chunking process. These may utilise machine learning algorithms to discern context and infer natural divisions within the text [17].

Level1: Character Splitting

The most common method is to split the document into chunks on a fixed number of tokens (e.g., 100, 256, 512) regardless of their content or form. The pro is then this method is semple but it's very rigid and doesn't take into account the structure of your text.

Level2: Recursive Character Text Splitting

The problem with Level1 is that we don't take into account the structure of our document at all. The Recursive Character Text Splitter tries to split based on special characters[2] until the chunks are small enough. The splitter first looks for double new lines (paragraph break). Once paragraphs are split, then it looks at the chunk size, if a chunk is too big, then it'll split by the next separator. If the chunk is still too big, then it'll move onto the next one and so forth. This has the effect of trying to keep all paragraphs (and then sentences, and then words) together as long as possible, as those would generically seem to be the strongest semantically related pieces of text.

Level 3: Document Specific Chunking

Document-specific chunking creates chunks by considering the document's inherent structure, such as paragraphs or sections. It's like the recursive method but changing the special characters list in order to fit with the document format: for example MD file, Python and JS file.

PDF documents

A specif consideration require the PDF documents, in particular when table are present. A common practice to deal with table is to summarize the raw table using a LLM and embedding the summaries. If the summary embedding matches the query, then pass the raw table to LLM. It is also necessary to consider the images that are present within the file. In this case, it would be possible to employ a multimodal LLM to generate textual summaries from images and subsequently incorporate this data into the vector database.

Level 4: Semantic Chunking

So far, we've only considered where to split our data, whether it's at end of a paragraph or a new line or a tab or other separators. But we haven't thought about when to split, that is, how to better capture a meaningful chunk rather than just a chunk of some length. This approach is known as semantic chunking [16]. As Todeschini show in medium article [17], we can use KMeans Cluster-

[2] The default separtor list is ["\n\n", "\n", " ", ""]

ing tecnique to perform this chunking. However, a significant drawback of the K-Means algorithm is that it does not maintain sentence order, resulting in the loss of the original sequence of sentences and potential distortion of the text's narrative flow. This is a particularly crucial issue, as the coherence and contextual integrity of the text may be at risk of being compromised. Furthermore, K-Means clustering is a computationally intensive process, particularly when applied to large text corpora or when managing a greater number of clusters. This limitation presents a significant challenge for real-time applications and the processing of big data, potentially impeding the efficiency and responsiveness of such systems. To overcome some of the limitations of KMeans clustering, especially the loss of sentence order, an alternative approach could be clustering adjacent sentences based on their semantic similarity. The fundamental premise of this approach is that two sentences that appear consecutively in a text are more likely to be semantically related than two sentences that are farther apart. The basic idea of this algorithm is to exploit emebedding to group similar chunks together, because embeddings represent the semantic meaning of a string. So considering sentence per sentence compare the current with the next one and calculate the cosine similary: if they distance is smaller than a threeshold or maximum chunk size is not reached, keep the sentence into the same group.

Level 5: Agentic Chunking

An alternative method of text segmentation is the application of agentive chunking. The objective is to emulate the cognitive processes that humans employ when segmenting documents into coherent units of meaning.

As the reader progresses through the text, it continuously assess whether each subsequent sentence or segment should be incorporated into the existing chunk or if it necessitates the formation of a new one. This iterative process continues until the document is fully analyzed, resulting in discrete chunks that maintain semantic similarity. This cognitive approach can be conceptualized as pseudo code, providing a foundational framework for implementing Agentic Chunking within an LLM.

Metadata Enrichment

The addition of metadata, including information such as page number, file name, author, category, and timestamp, can enhance the value of data chunks. Subsequently, retrieval can be filtered based on the specified metadata, thereby limiting the scope of the retrieval[3]. In addition to the extraction of metadata from the source documents, it is also possible to construct metadata artificially. One example of this is the addition of summaries of paragraphs, as well as the introduction of hypothetical questions. This method is also known as Reverse HyDE. In particular, an LLM can be used to generate questions that can be answered by the document. The similarity between the original question and the hypothetical question can then be calculated during retrieval, with the aim of reducing the semantic gap between the question and the answer.

2.2.1.3 Embedding

In the RAG system, the retrieval of relevant information is achieved by computing the similarity between the embeddings of the question and the document chunks, using metric such as cosine similarity. The ability of embedding models to capture semantic representations plays a key role in this process, as highlighted in recent surveys on RAG models [13].

Embedding models achieve this by generating vector representations of textual inputs, effectively encoding their semantic meaning in a numerical space [18]. This facilitates the retrieval of relevant information by allowing the model to compare and contrast the semantic similarity between different pieces of text. In the following section, we will outline the approach to embedding solutions.

[3] This approach is implemented in Langchain through "Self-querying" retrieval

Distrubutional Semantic Model

Distributional Semantic Models (DSMs) [19], also known as "word space" or **"distributional similarity" models**, are a class of models in the field of Natural Language Processing (NLP) that **quantify** and **categorize semantic similarities** between linguistic items based on their distributional properties in large samples of language data. The defining feature of any distributional semantics model is the assumption that the concept of semantic similarity can be defined in terms of linguistic distributions. This hypothesis is known as the Distributional Hypothesis (DH), which can be expressed as follows:

> The degree of semantic similarity between two linguistic expressions A and B is a function of the similarity of the linguistic contexts in which A and B can appear.

To fully understand the meaning of a word, it is essential to consider the context in which it is used. This is where the self-attention mechanism in the transformer model becomes crucial. By employing this mechanism, the transformer is able to capture the contextual information associated with each token, effectively relating them to all other tokens in the sentence [20].

Word Embedding: the idea

Vector semantics and distributional representations formalize the notion of contextual meaning by employing mathematical techniques to encode the statistical association strength between a word and its context. By associating a word with a contextual representation and formalizing it as a vector, words can be conceived as points in a high-dimensional "distributional space", where their position is determined by their statistical distribution in each context. Adopting the Distributional Hypothesis, this distributional vector space can be interpreted as a **semantic space**, where geometric distances between points correspond to semantic distances between words, enabling the quantification of semantic relationships between words [19].

Human vocabulary comes in free text. In order to make a machine learning model understand and process the natural language, we need to transform the free-text words into numeric values. One of the simplest transformation approaches is to do a one-hot encoding in which each distinct word stands for one dimension of the resulting vector and a binary value indicates whether the word presents (1) or not (0). However, one-hot encoding is impractical computationally when dealing with the entire vocabulary, as the representation demands hundreds of thousands of dimensions [21]. While one-hot encoding provides a straightforward way of representing words, it has several notable disadvantages. First, the high dimensionality of the vectors leads to computational inefficiencies, as the size of the representation increases linearly with the size of the vocabulary. This results in sparse vectors, where most elements are zero, making storage and processing even more difficult. In addition, one-hot encoding fails to capture semantic relationships between words; for example, the vectors for "cat" and "dog" are equidistant from "apple", ignoring their contextual similarities.

To overcome the limitations of one-hot encoding, researchers began exploring techniques that incorporate contextual knowledge. These techniques can be categorized into count-based and context-based methods. Count-based methods, such as Latent Semantic Analysis (LSA) [22], utilize statistical information derived from word co-occurrences within a corpus. By analyzing how frequently words appear together, these methods generate dense vector representations that capture some semantic relationships. However, count-based techniques produce static embeddings that do not account for polysemy, meaning that a word with multiple meanings will have the same representation regardless of context. In contrast, context-based techniques, such as the Skip-Gram model and Continuous Bag-of-Words (CBOW), represent a significant advancement in embedding methodologies. These models, part of the Word2Vec[4] framework, leverage neural networks to learn word representations based on their context within a given window of text. The Skip-Gram model predicts surrounding context words given a target word, effectively capturing semantic relationships by focusing on the context in which words appear. In contrast, CBOW predicts a target word based on its surrounding context words, highlighting the importance of context in understanding word meanings. Despite their improvements over static representations, context-based techniques still face challenges. They often struggle to capture long-range dependencies and complex meanings in large sentences, and their effectiveness depends highly on the quantity and quality of the training data. Insufficient or biased data can lead to suboptimal representations [23]. The introduction of transformer models, such as BERT (Bidirectional Encoder Representations from Transformers) and GPT (Generative Pre-trained Transformer), has marked a revolutionary shift in the field of NLP. Transformers utilize self-attention mechanisms to process words in relation to all other words in a sentence, allowing for a more nuanced understanding of context. This dynamic contextualization enables transformers to generate embeddings that are context-dependent, meaning the representation of a word can change based on its surrounding words, effectively addressing the issue of polysemy. Additionally, the self-attention mechanism allows transformers to capture relationships between words that are far apart in a sentence, improving the model's understanding of complex structures.

The mechanism plays, also, a vital role in the comprehension of semantic meaning at the sentence level within natural language processing models. By feeding a sentence into BERT without including the linear layer, the result is a vector that encapsulates the meaning of the sentence in question. It is important to note, however, that BERT is not designed to guarantee that the

[4] Word2Vec, a technique for natural language processing (NLP) that uses a neural network model to learn word associations from a large corpus of text.

embeddings it generates can be compared using cosine similarity. In other words, similar sentences should ideally be represented by vectors that point in the same direction within the vector space. This gives raise to the question of how one might effectively train BERT to produce embeddings that can be evaluated using a chosen similarity function.

To address this challenge, they employ Sentence-BERT [24], a model that has been specifically designed to enhance BERT's ability to generate meaningful sentence embeddings. The training process involves the input of two comparable sentences into BERT, which generates vector representations for each. Subsequently, the cosine similarity between the two vectors is calculated and compared to a predefined target cosine similarity. By calculating the loss based on this comparison, the model parameters can be adjusted through backpropagation, allowing for the modification of the model's weights. This iterative process enables the model to learn to produce similar vector representations for sentences that are semantically similar [20].

2.2.1.4 Vector store index

The most commonly utilized methodology for archiving and retrieving unstructured information involves the embedding of the data and the generation of corresponding embedding vectors. During the query process, it is possible to embed the query in order to retrieve the embedding vectors that present the closest match to the embedded query. A vector store is responsible for storing embedded data and performing vector searches using a distance metric, which the most common are [25]:

- **Euclidean Distance:** This metric represents the shortest possible straight-line distance between two points in a multidimensional space, analogous to measuring the physical distance in a geometric context.
- **Cosine Similarity:** This metric concerns the angle, not the length. It quantifies the degree to which two vectors point in the same direction, which is particularly useful for text or documents where the focus is on

meaning. For instance, it can be employed to determine whether two concepts are similar, opposite, or unrelated.

- **Dot Product:** This metric examines the degree to which two vectors exhibit alignment.

So a Vector Database is a specialized system designed to efficiently handle high-dimensional vector data. It excels at indexing, querying, and retrieving this data, enabling advanced analysis and similarity searches that traditional databases cannot easily perform.

The definition of a vector in a vector database is dependent upon three key elements: the ID, the dimensions, and the payload [25]. These components operate in conjunction to ensure the effective representation of a vector within the system. Together, they form a point, which represents the fundamental unit of data stored and retrieved in a vector database. In some cases, it may be necessary to consider additional information beyond the numerical data to gain a comprehensive understanding or to refine a search. While the dimensions provide an overview of the data, the payload contains metadata for structured information. This could include textual data such as descriptions, tags, or categories, or it could be numerical values such as dates or prices. This supplementary information is crucial when attempting to filter or rank search results based on criteria that are not directly encoded in the vector [26]. For example, during my traineeship, it was advantageous to integrate additional data regarding the courses themselves into the vector database. This enabled a more precise search, whereby users could restrict their results to a particular course.

How indexing vectors

The process of indexing vectors is analogous to the creation of an entry in a conventional database. However, it is essential that vectors are indexed in a manner that facilitates efficient searching in subsequent operations. A simplistic approach might be to compare the query against all vectors, sort them by distance, and retain the top K results. Although this method offers high precision due to its exhaustive nature, it is computationally expensive, particularly as the number of vectors in the database increases. This trade-off between precision and computational efficiency demonstrates a discordance in the indexing process, whereby the achievement of accurate results frequently entails a compromise in speed. To improve the speed of the search process, it is necessary to reduce the number of comparisons that are made during the search. One effective solution is the use of the Hierarchical Navigable Small World (HNSW) algorithm [27], which is a powerful indexing technique that is employed by many vector databases. It is an evolution of the Navigable Small Worlds algorithm for Approximate Nearest Neighbors, which is based on the concept of Six Degrees of Separation. The HNSW algorithm constructs a multi-layered graph, wherein each vector is represented as a node, and the connections between nodes signify their similarity. In this structure, the upper layers establish connections between vectors that are broadly similar, while the lower layers link vectors that are more closely related. This enables the execution of searches that become more and more refined as one progresses deeper into the graph. Upon executing a search, HNSW initiates the search process at the top layer, rapidly narrowing the search space by traversing between layers. The hierarchical approach allows the algorithm to focus exclusively on the most relevant vectors as it delves deeper, thereby refining the search with each successive step.

2.2.1.5 Advanced Retrieval Types

In the previous sections, we have examined the importance of breaking down raw text into manageable chunks, a vital step in preparing data for various applications. However, it is equally crucial to evaluate whether the raw text serves as the most effective representation of the data for the specific task at hand. For example, when performing semantic searches on chat messages, the unprocessed text may lack the necessary context to generate meaningful embeddings. In such cases, it may be more advantageous to conduct a semantic search on a summary of the conversation or even on hypothetical questions that the chat could address [28]. This consideration naturally introduces the topic of indexing, where the emphasis shifts from merely categorizing data to making informed decisions about how to represent that data within a database or knowledge base. Indexing is not merely a technical necessity; it is a strategic choice that significantly influences the efficiency and effectiveness of data retrieval. Depending on the use case, various indexing methods can be implemented, each offering distinct advantages tailored to specific requirements. The most used approach are [29]:

- **Smaller chunks:** split a document into smaller chunks, and embed those (this is ParentDocumentRetriever). It is frequently beneficial to retrieve larger units of information and embed smaller units within them. This approach enables embeddings to capture the semantic meaning with the greatest possible accuracy while allowing for the transfer of as much context as possible to subsequent processes.

- **Summary:** create a summary for each document, embed that along with (or instead of) the document.

- **Hypothetical questions:** create hypothetical questions that each document would be appropriate to answer, embed those along with (or instead of) the document.

Another one techniques that required attention is the Self-querying retriever. As the name suggests, has the ability to query itself. Specifically, given any natural language query, the retriever uses a query-constructing LLM chain to write a structured query and then applies that structured query to its underlying VectorStore. This allows the retriever to not only use the user-input query for semantic similarity comparison with the contents of stored documents but to also extract filters from the user query on the metadata of stored documents and to execute those filters.

2.2.1.6 Pre-retrieval module: Query Optimization

A particularly noteworthy approach is the use of LLMs in the manipulation of input queries. LLMs have the potential to enhance the indexing process and facilitate a more profound comprehension of the context and intent underlying user queries. By leveraging their capacity to comprehend the complexities of natural language, LLMs can transform user queries into more meaningful representations that are more closely aligned with the indexed data. This transformation not only improves the accuracy of search results, but also allows more sophisticated filtering and classification mechanisms to be applied, thereby enhancing the overall search experience. The primary applications of query analysis include the rewriting of the query itself to facilitate semantic or lexical searches, or the construction of structured queries for inter-search databases.

Query re-writing

It is desirable for retrieval systems to be capable of handling a wide range of user inputs, from simple and poorly worded queries to complex, multi-faceted questions. In order to achieve this versatility, a popular approach is to use models to transform raw user queries into more effective search queries. This transformation can range from simple keyword extraction to more sophisticated techniques such as query expansion and reformulation.

One prominent approach is **multi-query**, which is particularly beneficial when the objective is to ensure high recall in retrieval by providing multiple phrasings of a question. This method entails rephrasing the user's query in a number of alternative formats. By generating multiple versions of the original query, the system is able to retrieve documents corresponding to each rewritten question. The results are then aggregated in order to return a unique set of documents that encompass the information relevant to all variations of the query. This approach is particularly beneficial in scenarios where users may express the same intent in different ways, thereby increasing the likelihood of retrieving pertinent documents that might otherwise be overlooked.

An additional strategy that has been demonstrated to be effective is **decomposition**. This can be applied when a question can be divided into smaller, more manageable sub-problems. This technique requires the analysis of the user's question in order to identify its constituent sub questions or subproblem. The sub questions may be addressed either sequentially or in parallel. In a sequential approach, the response to the initial sub question is employed to inform the retrieval process for the subsequent sub question. In contrast, a parallel approach involves addressing each sub question independently, with the responses subsequently consolidated to provide a comprehensive answer to the original query. This method is particularly efficacious for complex inquiries that necessitate a sophisticated comprehension of the multifaceted aspects of a given topic.

The **step-back technique** is employed when a higher-level conceptual understanding is required to address the user's question effectively. In this approach, the system initially prompts a large language model (LLM) to generate a generic step-back question that focuses on higher-level concepts or principles related to the user's inquiry. By retrieving pertinent facts and information about these overarching concepts, the system is able to provide a more informed and contextually grounded answer to the user's original question. This approach is beneficial in cases where the user's query may be overly specific or lacking sufficient context. It enables the system to construct a more comprehensive understanding before addressing the precise inquiry.

Lastly, the **HyDE (Hypothetical Document Embedding)** technique is particularly useful in instances where challenges are encountered in the retrieval of relevant documents using the raw user inputs. This innovative approach employs an LLM to transform user queries into hypothetical documents that can effectively address the underlying issue. By embedding these hypothetical documents, the system can perform a document-to-document similarity search, aiming to retrieve real documents that align closely with the generated hypothetical content. The underlying premise of this method is that a similarity search can yield more relevant matches, as it allows the system to explore the conceptual space surrounding the user's query rather than relying solely on the original phrasing of the question.

Query construction

Another significant aspect of query optimisation is the translation of natural language queries into specialised query languages or filters. It is crucial to translate natural language queries into structured and semi-structured data formats in order to facilitate the efficient and accurate retrieval of relevant information by users. In the context of structured data, which is commonly found in relational and graph databases, domain-specific languages (DSLs) are employed to facilitate data querying. To take one example, the text-to-SQL approach converts natural language queries into SQL statements, thus enabling users to interact with relational databases in an efficacious and non-intrusive manner. Similarly, the Text-to-Cypher method translates natural language into Cypher queries, which are specifically designed for querying graph databases. These techniques demonstrate how natural language processing (NLP) can facilitate the alignment between user intent and the technical requirements of structured data retrieval.

In addition to structured data, semi-structured data, such as that stored in vector stores, necessitates a distinct methodology for querying. In this context, queries can be integrated with semantic search and metadata filtering. The technique of Natural Language to Metadata Filters converts user queries into appropriate metadata filters, thereby enabling users to refine their searches based on specific attributes rather than solely relying on textual similarity. This is particularly advantageous when users seek information that pertains to the metadata of documents, as opposed to their content.

2.2.2 Generation

Once the information has been retrieved, it is recommended that it is not directly input into the LLM for the purpose of answering questions. The following section will introduce adjustments from two perspectives: firstly, adjustments to the retrieved content and secondly, adjustments to the LLM [13].

2.2.2.1 Post-retriever techniques: Context curation

The presence of redundant information and long contexts presents a significant challenge to effective information processing and generation. The LLM, which exhibit cognitive tendencies similar to those observed in humans, demonstrate a tendency to focus predominantly on the beginning and end of long texts, while forgetting the middle portion. This phenomenon, which is known as the 'lost in the middle' problem, can be solved by implementing more advanced processing techniques to enhance the quality of the retrieved information, prior to its utilisation in LLM generation phase.

One prominent strategy is **reranking**, which involves the systematic reordering of document chunks to prioritise the most relevant results. This approach serves two distinct functions, acting both as an enhancer and a filter in the information retrieval process. Reranking can be executed through a variety of methodologies, including rule-based approaches that rely on predefined metrics such as diversity, relevance, and mean reciprocal rank (MRR), as well as model-based techniques utilising sophisticated architectures, such as encoder-decoder models from the BERT series and general LLMs, including GPT.

Another critical aspect of optimising LLM performance is the selection and **compression of context**. A common misconception within the RAG framework is the belief that aggregating a large number of relevant documents into a single, extensive retrieval prompt is advantageous. In practice, an excess of context can introduce noise, thereby impairing the LLM's ability to discern essential information. To address this challenge, small language models (SLMs) have been utilized to identify and eliminate superfluous tokens, resulting in a compressed format that, while potentially challenging for human comprehension, is more conducive to LLM processing. The utilisation of techniques such as training information extractors and condensers through contrastive learning serves to further enhance this compression process, thereby improving the accuracy of the model's outputs.

The **"Filter-Reranker"** paradigm represents an innovative approach that effectively combines the strengths of both LLMs and SLMs. In this approach, SLMs act as filters to identify challenging samples, and LLMs are tasked with reordering these samples for optimal information extraction. Empirical evidence indicates that this collaborative framework results in notable improvements across a range of information extraction tasks. Furthermore, enabling LLMs to assess retrieved content before generating final responses represents a practical approach to enhancing relevance and accuracy. This is exemplified by systems such as Chatlaw, where LLMs are prompted to evaluate the relevance of referenced legal provisions. The combination of these strategies demonstrates the necessity of refining the retrieval and processing stages in RAG systems to enhance LLM performance and guarantee the generation of high-quality outputs.

2.2.2.2 Prompt engineering

Prompt Engineering, also referred to as In-Context Prompting, represents a fundamental methodology in the interaction with Large Language Models. It enables users to influence the model's behaviour towards desired outcomes without the necessity for modifications to the model's weights [30]. This empirical science is distinguished by its reliance on experimentation and heuristics, given that the efficacy of different prompting techniques can vary quite significantly across different models. Among the foundational approaches to prompt engineering are **zero-shot** and **few-shot learning**, both of which have been extensively explored in the literature and serve as benchmarks for evaluating LLM performance. The concept of zero-shot learning involves presenting a model with text that is task-specific, and then requesting the results without providing examples. In contrast, few-shot learning provides a curated set of high-quality demonstrations, each comprising both the input and the desired output for the target task. The utilization of exemplar cases in few-shot learning facilitates an enhanced comprehension of human intentions and the criteria for acceptable responses, frequently resulting in a superior performance compared to that observed in zero-shot learning. However, this approach results in a greater consumption of tokens and may encounter limitations related to context length, particularly when dealing with lengthy input and output texts.

A number of studies have examined the construction of in-context examples with the objective of optimising performance. These studies have identified that the choice of prompt format, the inclusion of training examples and the sequence in which the examples are presented can have a significant impact on the resulting performance.

For example, the selection of effective examples is of crucial importance in few-shot learning. Liu et al. [31] proposed the selection of examples that are semantically similar to the test example through the use of nearest-neighbour clustering in the embedding space. To achieve a diverse and representative set of examples, Su et al. [32] introduced a graph-based approach that constructs a directed graph based on cosine similarity among embeddings, thereby facilitating the selection of diverse samples.

It is recommended that the selection of examples be diverse, pertinent to the test sample, and presented in a randomized order to mitigate potential issues related to majority label bias and recency bias. It has been demonstrated that simply increasing the model size or incorporating additional training examples does not reduce the variance observed across different permutations of in-context examples. The efficacy of a given order may vary depending on the model in question [30].

Instruction Prompting

Another approach is to provide the LLM with direct instructions. Instruction prompting has emerged as a pivotal technique within the field of prompt engineering, characterized by the direct provision of explicit instructions to

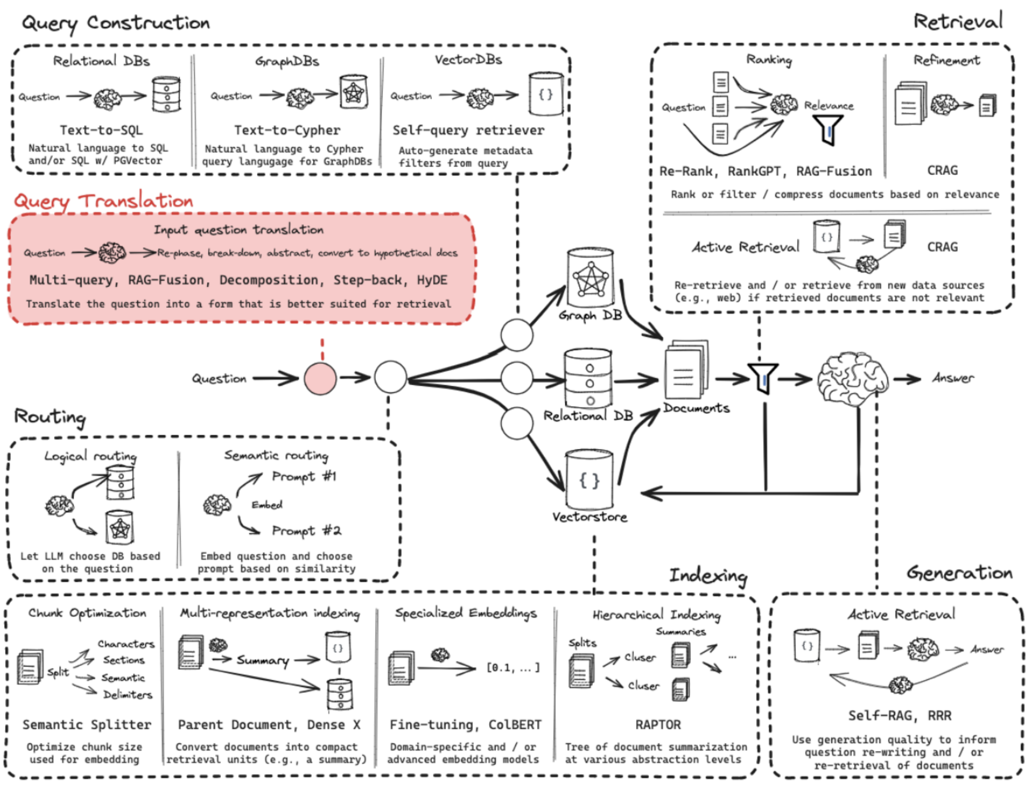
LLMs. This approach is exemplified by instructed language models, such as InstructGPT and Natural Instruction, which are fine-tuned using high-quality tuples that consist of task instructions, inputs, and corresponding ground truth outputs. The primary objective of this fine-tuning process is to enhance the model's ability to comprehend user intentions and adhere to specific instructions. A common methodology employed in the fine-tuning of instructed language models is reinforcement learning from human feedback (RLHF). This technique uses human evaluators to provide feedback on the model's outputs, which is subsequently employed to refine the model's behaviour in accordance with human expectations. The benefit of instruction-following style fine-tuning is the improvement of the model's alignment with human intention and the reduction of communication costs.

In the context of interaction with instruction-based models, it is essential to articulate the task requirements with the necessary level of specificity and precision. By providing a comprehensive and precise delineation of the desired outcome, it is possible to significantly enhance the performance of the model. For example, it is preferable to avoid the use of negative phrasing, such as "do not do X," and instead focus on specifying the desired actions.

In their proposal, Ye et al. [33] further refine the instruction prompting technique by integrating few-shot learning with instruction-based prompts, thus introducing the concept of in-context instruction learning. This method involves the integration of multiple demonstration examples across a range of tasks within a single prompt where each demonstration is made by an instruction, a task input, and an expected output.

Furthermore, the concept of chain-of-thought (CoT) prompting has emerged as a valuable supplementary approach within the domain of instruction prompting. As articulated by Wei et al. [34], CoT prompting involves the generation of a sequence of short sentences that describe the reasoning process step by step. These sentences are known as reasoning chains or rationales and ultimately lead to the final answer. The advantages of CoT prompting are especially pronounced in the context of complex reasoning tasks, particularly when employing large models with over 50 billion parameters. In contrast, the benefits of CoT prompting may be less pronounced for simpler tasks.

CoT prompting can be classified into two principal categories: "few-shot" and "zero-shot". Few-shot CoT involves prompting the model with a limited number of demonstrations, each containing manually crafted or model-generated high-quality reasoning chains. In contrast, zero-shot CoT employs natural language statements, such as "Let's think step by step" to explicitly prompt the model to generate reasoning chains before reaching a conclusion.



Query Construction
Relational DBs
Question
Text-to-SQL
Natural language to SQL and/or SQL w/ PGVector
GraphDBs
Question
Text-to-Cypher
Natural language to Cypher query langugage for GraphDBs
VectorDBs
Question
Self-query retriever
Auto-generate metadata filters from query
Retrieval
Ranking
Question
Relevance
Re-Rank, RankGPT, RAG-Fusion
Refinement
CRAG
Rank or filter / compress documents based on relevance
Active Retrieval
CRAG
Re-retrieve and / or retrieve from new data sources (e.g., web) if retrieved documents are not relevant
Query Translation
Input question translation
Question
Re-phase, break-down, abstract, convert to hypothetical docs
Multi-query, RAG-Fusion, Decomposition, Step-back, HyDE
Translate the question into a form that is better suited for retrieval
Question
Graph DB
Relational DB
Vectorstore
Documents
Answer
Routing
Logical routing
Let LLM choose DB based on the question
Semantic routing
Prompt #1
Embed
Prompt #2
Embed question and choose prompt based on similarity
Indexing
Chunk Optimization
Split
Characters
Sections
Semantic
Delimiters
Semantic Splitter
Optimize chunk size used for embedding
Multi-representation indexing
Summary
Parent Document, Dense X
Convert documents into compact retrieval units (e.g., a summary)
Specialized Embeddings
[0.1, ...]
Fine-tuning, ColBERT
Domain-specific and / or advanced embedding models
Hierarchical Indexing
Splits
Summaries
Cluser
Cluser
...
RAPTOR
Tree of document summarization at various abstraction levels
Generation
Active Retrieval
Answer
Self-RAG, RRR
Use generation quality to inform question re-writing and / or re-retrieval of documents

Figure 2.3: A complete overview of all the system RAG part involved [35]

2.2.3 Argumentation process

In the field of Retrieval-Augmented Generation, conventional methodologies typically entail a single retrieval phase, followed by the generation of responses. Although this approach is relatively simple, it is often inefficient and inadequate for addressing complex problems that require multi-step reasoning. This is because it limits the scope of information available to the language model. In order to enhance the efficacy of information retrieval and generation, a number of advanced strategies have emerged, including iterative retrieval, recursive retrieval and adaptive retrieval [13]. Three types of retrieval augmentation processes have been included.

Iterative retrieval involves alternating between retrieval and generation, allowing for richer and more targeted context from the knowledge base at each step. However, challenges such as semantic discontinuity and the accumulation of irrelevant information may emerge.

Another significant technique is **recursive retrieval**. It involves gradually refining the user query and breaking down the problem into sub-problems, then continuously solving complex problems through retrieval and generation.

The objective of this method is to enhance the search experience through the implementation of a feedback loop that gradually converges on the most pertinent information. For example, the IRCoT [36] uses a chain-of-thought methodology to direct the retrieval process, whereby the chain of thought is refined in accordance with the results obtained. Similarly, Tree of Clarifications (ToC) [37] framework employs a clarification tree to systematically optimize ambiguous aspects of the query. This recursive nature is particularly advantageous in complex search scenarios where the user's needs may not be fully articulated.

The objective of **adaptive retrieval** methods is to facilitate the autonomous determination by the RAG system of the necessity for external knowledge retrieval and the optimal point at which to terminate the retrieval and generation processes. This is frequently achieved through the use of LLM-generated special tokens for control.

This proactive approach enables LLMs to discern when to seek necessary information, thereby emulating the behaviour of agents that utilise a variety of tools. Similarly, WebGPT incorporates a reinforcement learning framework to train the GPT-3 model to autonomously use a search engine during text generation. It navigates this process through specialized tokens, which facilitate actions such as the execution of search engine queries, the navigation of results and the referencing of sources. This integration of external search engines serves to augment the capabilities of GPT-3. In contrast, Self-RAG introduces "reflection tokens" which permit the model to engage in introspective evaluation of its outputs. The tokens are classified into two categories: "retrieve" and "critic." The model autonomously determines the optimal point in time to activate the retrieval process, or alternatively, this process may be triggered by a predefined threshold.

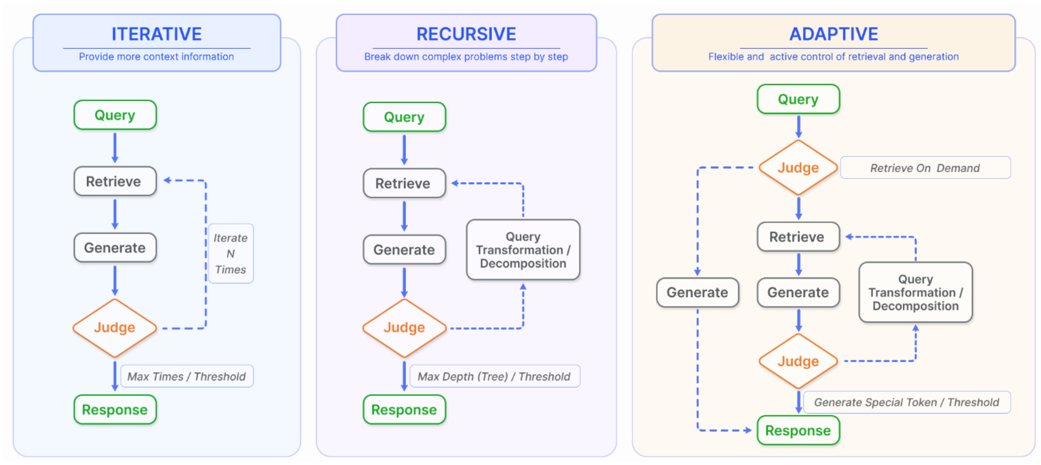

ITERATIVE
Provide more context information
Query
Retrieve
Generate
Iterate N Times
Judge
Max Times / Threshold
Response
RECURSIVE
Break down complex problems step by step
Query
Retrieve
Generate
Query Transformation / Decomposition
Judge
Max Depth (Tree) / Threshold
Response
ADAPTIVE
Flexible and active control of retrieval and generation
Query
Judge
Retrieve On Demand
Retrieve
Generate
Generate
Query Transformation / Decomposition
Judge
Generate Special Token / Threshold
Response

Figure 2.4: Argumentation process in RAG [13]

2.3 RAG Paradigms

Since its initial formulation, the RAG paradigm has undergone a substantial degree of evolution, leading to the emergence of three discrete methodologies. These include Naive RAG, Advanced RAG and Modular RAG [13]. Each of these paradigms reflects a progression in the sophistication and effectiveness of retrieval and generation processes, addressing the limitations of its predecessors while enhancing overall performance. The **naive RAG** approach represents the fundamental methodology underlying RAG, achieving prominence shortly after the widespread adoption of models such as ChatGPT. This methodology is based on a traditional framework comprising a sequential process of indexing, retrieval and generation, which is often referred to as the "Retrieve-Read" paradigm. Although Naive RAG offers a simple method for integrating retrieval with generation, it is constrained in its ability to process complex queries that necessitate sophisticated comprehension and multi-stage reasoning.

In response to the limitations of Naive RAG, **Advanced RAG** incorporates a series of improvements with the objective of enhancing retrieval quality. This paradigm employs both pre-retrieval and post-retrieval strategies to refine the indexing process, using techniques such as sliding window approaches, fine-grained segmentation, and the incorporation of metadata. These optimizations provide a more efficient retrieval process, thereby enabling access to information that is both more relevant and contextually appropriate.

Finally, the **Modular RAG** architecture represents a significant advancement beyond both the naive and advanced RAG paradigms. It offers enhanced adaptability and versatility through the incorporation of specialized modules designed to improve retrieval and processing capabilities. This framework introduces new components, including a *Search module* that direct searches across diverse data sources, such as search engines and/or a knowledge graphs through the utilisation of LLM-generated code and query languages. The *Memory module* facilitates enhanced retrieval by creating an unbounded memory pool that aligns text with data distribution through iterative self-enhancement.

The *Routing component*, meanwhile, enables optimal navigation through various data sources, enabling the selection of the most relevant pathways for queries. Furthermore, the *Predict module* minimizes redundancy and noise by generating context directly through the LLM, thereby ensuring the relevance and accuracy of the information retrieved. The *Task Adapter module* serves to further adapt the RAG framework to accommodate a variety of downstream tasks. It automates the retrieval of prompts for inputs that are not previously observed, and it generates task-specific retrievers through the generation of few-shot queries. This comprehensive approach not only streamlines the retrieval process but also significantly enhances the quality and relevance of the information sourced, thereby catering to a wide array of tasks and queries with improved precision and flexibility.

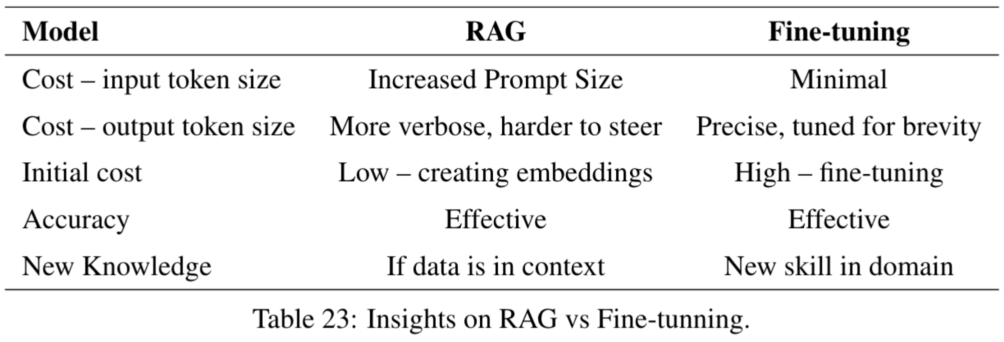
2.3.1 RAG vs fine-tuning

As research advanced, RAG strategy covers not only inference stage but also fine-tuning techniques.

It has been demonstrated that the application of fine-tuning, when specifically adapted to align with the characteristics of a given scenario and data set, can lead to enhanced outcomes in the performance of large language models. Such an approach allows the introduction of supplementary knowledge into the model, thereby enhancing its performance in specialized domains. Moreover, fine-tuning provides the additional benefit of enabling the model to adapt both its input and output, thus allowing it to conform to specific data formats and generate responses that align with with human or retriever preferences [13]. Fine-tuning is distinguished by its relatively static nature, necessitating retraining for updates. Although this process allows for profound customisation of the model's behaviour and stylistic output, it also necessitates the allocation of considerable computational resources for the preparation of the dataset and the training process. In contrast, RAG demonstrates optimal efficacy in dynamic environments, wherein there is a critical need for real-time knowledge updates and the effective utilization of external knowledge sources. The architectural design of RAG is designed in such a way as to allow for high levels of interpretability, thereby enabling users to trace the origins of the information presented.

Empirical evaluations of RAG and fine-tuning in a variety of knowledge-intensive tasks have demonstrated that, while unsupervised fine-tuning can result in some improvements, RAG consistently outperforms it in both known and novel knowledge contexts. Moreover, studies have demonstrated that LLMs frequently encounter challenges in assimilating new factual information through unsupervised fine-tuning alone. However, the decision to use RAG or fine-tuning is dependent on the specific requirements of the data dynamics, the desired level of customisation, and the computational capabilities available in the application context [13].

It is important to note that RAG and fine-tuning are not mutually exclusive; rather, they can be employed in conjunction with one another, enhancing a model's capabilities at various levels. In certain scenarios, the combination of both strategies may result in optimal performance outcomes.



Model	RAG	Fine-tuning
Cost – input token size	Increased Prompt Size	Minimal
Cost – output token size	More verbose, harder to steer	Precise, tuned for brevity
Initial cost	Low – creating embeddings	High – fine-tuning
Accuracy	Effective	Effective
New Knowledge	If data is in context	New skill in domain

Table 23: Insights on RAG vs Fine-tunning.

Figure 2.5: Comparison table from Balaguer et al. [38]

2.4 RAG flow pattern

The entire RAG system can be conceptualized as a permutation and combination of multiple modules and corresponding operators, forming what we refer to as a RAG Flow. Each module type represents a fundamental process within the RAG system, incorporating a multitude of functional modules [39]. This section aims to explore the typical RAG flow patterns, detailing the specific implementations and best practices observed in the industry.

2.4.1 RAG flow patterns

In the context of RAG, we delineate four distinct flow patterns during the inference stage: Sequential, Conditional, Branching, and Loop structures.

Inference Stage

The inference stage of RAG systems is characterized by four primary flow patterns:

- **Sequential Flow:** This structure is designed to organise the RAG modules and operators in a linear sequence, frequently incorporating both pre-retrieval and post-retrieval modules. The sequential flow is a prominent feature of advanced RAG paradigms, as evidenced by pipelines such as Query Rewrite or HyDE, which precede retrieval and are followed by a Rerank operator. The Rewrite-Retrieve-Read (RRR) [40] model is a representative example of this flow, in which a smaller trainable language model optimises the query through reinforcement learning, utilising the LLM's output as a reward signal.
- **Conditional Flow:** In this structure, the RAG pathway is defined by specific conditions, typically facilitated by a routing module that selects different routes based on the semantics of the query. This enables the provision of customised responses in accordance with the characteristics of the query, with the utilisation of diverse retrieval sources and procedures for disparate categories of questions.
- **Branching Flow:** In contrast to the conditional approach, branching involves the concurrent execution of multiple parallel paths. In the case

of pre-retrieval branching, the original query is expanded into a series of sub-queries, each of which undergoes a separate retrieval process. In contrast, post-retrieval branching involves maintaining the original query and retrieving multiple document chunks for concurrent generation. The REPLUG framework [41] provides an illustrative example of this structure, employing a weighted ensemble of predictions from various branches in order to refine the final output.

- **Loop Flow:** The loop structure is an essential component of the modularity of RAG systems, enabling interdependent retrieval and reasoning steps. This flow can be categorised according to the retrieval approach employed, which may be iterative, recursive or adaptive. Iterative retrieval, as exemplified by ITER-RETGEN [42], comprises multiple cycles of retrieval and generation, whereby outputs from previous iterations are leveraged to enhance relevance. Recursive retrieval deepens the retrieval process through query transformation. In contrast, adaptive retrieval, such as Self-RAG, proactively determines when to engage in retrieval based on the model's assessment of knowledge sufficiency.

2.4.2 Best-industrial case flow patterns

In the context of modern artificial intelligence, Retrieval-Augmented Generation (RAG) systems have emerged as a crucial technology, particularly within industrial applications. This section will provide insight into several industry-leading RAG practices from the perspective of RAG Flow, offering an understanding of how to effectively combine and construct the flow of RAG in real-world application scenarios by examining the cases of OpenAI and Baichuan.

The OpenAI flow was primarily designed to facilitate basic query transformation and retrieval. It utilised techniques such as cosine similarity to identify relevant information, with a 45% accuracy rate in its retrieval tasks. They investigated several techniques, including hypothetical document embeddings

(HyDE), fine-tuning embeddings, and other methods; however, the results were not satisfactory. Instead, through the implementation of an experimental approach involving the manipulation of information chunks and the embedding of diverse content sections, the team was able to achieve an enhanced accuracy rate of 65%. By employing reranking and classification techniques to identify the most relevant retrieved chunks, the accuracy rate was enhanced to 85%. In conclusion, the integration of prompt engineering and query expansion resulted in an accuracy rate of 98%.

The OpenAI teams highlighted the significant potential of model fine-tuning and the integration of RAG, particularly in terms of achieving industry-leading levels without the use of complex techniques. This is made possible through the implementation of model fine-tuning and prompt engineering [43].

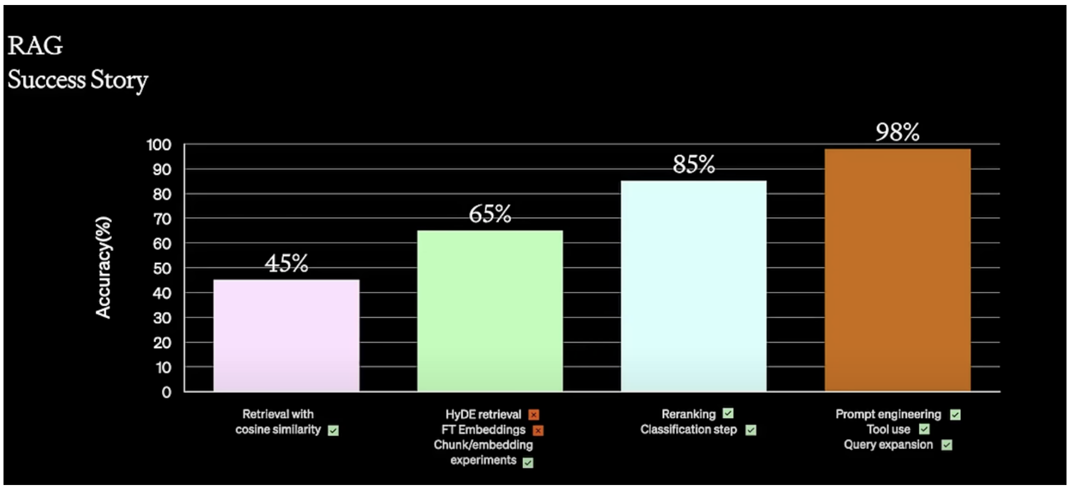


Figure 2.6: OpenAI RAG use case

Baichuan provides another illustrative case of RAG implementation with reference to Meta's CoVe [5]. The organisation has developed a methodology for decomposing complex queries into a variety of discrete and concurrent re-

[5] Chain-of-Verification (CoVe) is a method designed to reduce hallucinations in large language models through a structured four-step process: first, the model drafts an initial response; second, it formulates verification questions to fact-check this draft; third, it independently answers these questions to avoid bias; and finally, it generates a final, verified response based on the independent answers. This approach enhances the factual accuracy of the model's outputs across various tasks.

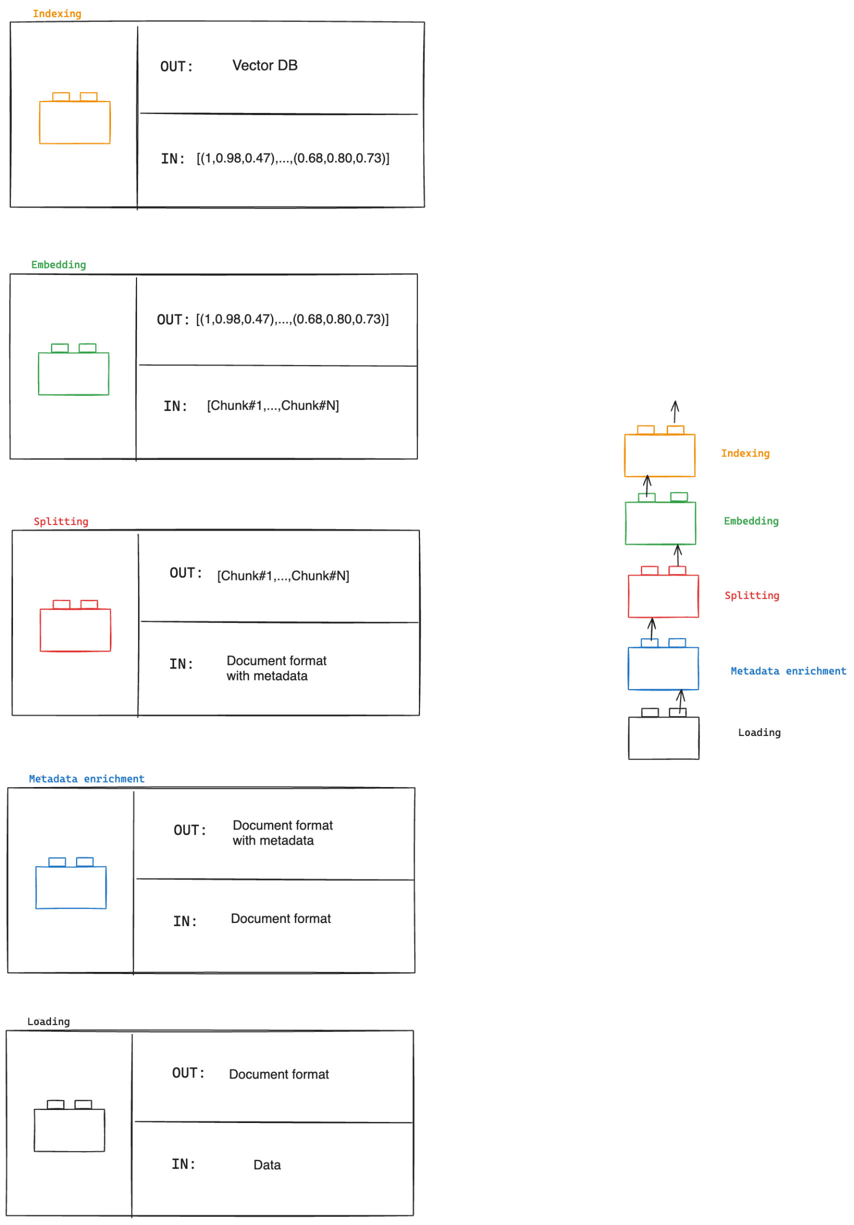
trievable queries. This innovative approach allows large models to perform targeted searches within their knowledge base for each sub-query, thereby increasing the accuracy and detail of responses while reducing the occurrence of incorrect response.

A key element of Baichuan's strategic approach is its proprietary Think-Step Further (TSF) method, which is designed to complement the step-back prompting technique. This method enables the system to infer deeper underlying questions from user input, thereby facilitating a more sophisticated understanding of user intent. In the retrieval phase, Baichuan has introduced the Baichuan text embedding vector model, which has been pre-trained on an extensive corpus of over 1.5 trillion tokens in Chinese.

In addition, Baichuan has implemented a hybrid retrieval approach that combines vector retrieval with sparse retrieval, resulting in a significant increase in the recall rate to 95%. Baichuan's complex RAG flow, characterized by numerous branches and advanced reranking processes, exemplifies the potential of sophisticated retrieval strategies to enhance the efficacy of RAG systems.

2.4.3 Internship use case flow pattern

In order to simplify the development of the RAG system, it would be beneficial to adopt a modular approach. This modular design draws inspiration from the concept of building with LEGO bricks. By systematically disassembling the complex architecture of the system into well-defined, discrete functional modules, each characterized by its specific operational functions, a highly reconfigurable framework is established [44].



Indexing
OUT: Vector DB
IN: [(1,0.98,0.47),....,(0.68,0.80,0.73)]
Embedding
OUT: [(1,0.98,0.47),....,(0.68,0.80,0.73)]
IN: [Chunk#1,...,Chunk#N]
Splitting
OUT: [Chunk#1,...,Chunk#N]
IN: Document format with metadata
Metadata enrichment
OUT: Document format with metadata
IN: Document format
Loading
OUT: Document format
IN: Data
Indexing
Embedding
Splitting
Metadata enrichment
Loading

Figure 2.7: Overview lego-like RAG modules

In accordance with this concept, a Python-based implementation was developed in which each module is responsible for performing a specific task, with the output of one module serving as the input for the next. The document structure, representing a single source, is based on the Langchain document structure:

```
{
    "page_content": "Content of the document",
    "metadata":
        "summary": "..."
        //other metadata
}
```

The modules that have been implemented are as follows:

- **Loading Module:** This module ingests the raw data and transforms it into a standardized document format.
- **Metadata Enrichment Module:** This module enhances the document with relevant metadata, which can be crucial for retrieval and context understanding.
- **Splitting Module:** This module segments the document into smaller chunks, suitable for embedding and efficient retrieval. Various splitting strategies can be implemented within this module, offering flexibility in chunk size and method.
- **Embedding Module:** This module generates vector representations (embeddings) for each chunk using a chosen embedding model.
- **Indexing Module:** This module stores the embeddings in a vector database, enabling efficient similarity search during retrieval.

The system was designed with the objective of reducing the complexity of the evaluation stage. Although the implementation focuses on a limited set of parameters—primarily chunk size and embedding model — to establish a benchmark using private data, the code has been written in such a way that it can accommodate future expansion and exploration of a broader range of configurations and functionalities.

Chapter 3

RAG evaluation

This chapter is inspired by the study conducted by Yu et al. [45]. This survey work provides a comprehensive overview of the evaluation and benchmarks of RAG systems.

3.1 Challenges

The complexity of evaluating RAG systems is derived from the integration of the retrieval component with the generative component. This requires the use of a range of methodologies, metrics and assessment tools. The intrinsic opacity of LLMs further complicate this landscape, as the black-box nature of these models obscures the processes by which outputs are generated from inputs. It is therefore crucial that the evaluation framework takes into account the efficacy of the retrieval component in sourcing pertinent information from external, dynamic databases, in addition to the generative component's capacity to synthesize this information into coherent and contextually appropriate outputs [10].

So, in general, the evaluation of RAG system must consider *three key aspects*: the capacity of the retrieval system to **identify relevant and contextually focused passages**. The proficiency of the **LLM in use these passages** while maintaining fidelity to the original content and the **overall quality of the generated output**. It is important to note that the quality of the information retrieved is crucial to the integrity of the final output, as it has a direct impact on the coherence and relevance of the generated text [46]. Therefore, a comprehensive evaluation framework must consider these interrelated components in order to ensure the effectiveness of RAG systems in real-world applications.

Once the components of a RAG system have been outlined, it is essential to establish a robust evaluation framework to address the critical question of which system will ultimately prove to be the most successful. Given the myriad of RAG configurations and workflows, it is important to identify the specific elements of the system that require evaluation and then pursue the optimal configuration. However, this evaluation process often relies on manual annotation of input queries, retrieved passages and generated responses. In fact, business company seeking to implement RAG systems need to conduct evaluations tailored to their unique use cases, which transcend traditional academic benchmarks and often involve comparisons between different architectures and models [1].

3.2 Evaluation Component

The two main components to be evaluated are the **retrieval** and **generation** component, as well as the RAG system in its totality.

The evaluation of the retrieval component presents a significant challenge due to the vast and diverse landscape of potential information sources, which encompass a range of formats, including structured databases and unstructured web content. While traditional metrics such as precision and recall remain foundational, they are insufficient for capturing the complexities of RAG systems. Consequently, the development of more sophisticated, task-specific evaluation metrics is necessary to account for temporal relevance and quality of retrieved information.

The generation component, driven by LLMs, is responsible for producing coherent responses that are contextually appropriate and derived from the retrieved content. A significant challenge in this context is evaluating the faithfulness and accuracy of the generated content in relation to the input data. This evaluation involves assessing not only the factual correctness of the responses but also their relevance to the original query and the overall coherence of the generated text. Furthermore, the subjective nature of certain tasks, such as creative content generation and open-ended question answering, introduces additional complexity to the evaluation process, as it allows for variability in the criteria that define a "correct" or "high-quality" response.

The integration of the two components introduces a greater degree of complexity to the evaluation process, as the overall performance of the system cannot be fully understood by evaluating each component in isolation. It is crucial to assess the system's capability to effectively use retrieved information to enhance response quality. This necessitates the measurement of the added value that the retrieval component contributes to the generative process. Furthermore, practical considerations, such as response latency and the system's capacity to manage ambiguous or complex queries, are essential for evaluating the overall effectiveness and usability of the system.

3.2.1 Evaluation aspects

In order to facilitate comprehension of the RAG evaluation process we will address three key questions: *What to Evaluate?*, *How to Evaluate?*, *How to Measure?*.

The first question concerns the evaluation direction and which aspects of the system need to be evaluated. The RAG system must be **relevant**, **accurate**, **faithful** and **correct**. For the retrieval component, *relevance* evaluates how well the retrieved documents match the information needed expressed in the query, measuring the precision and specificity of the retrieval process. *Accuracy* assesses how accurate the retrieved documents are in comparison to a set of candidate documents. It is a measure of the system's ability to identify and score relevant documents higher than less relevant or irrelevant ones. For the generation component, *relevance* measures how well the generated response aligns with the intent and content of the initial query. It ensures that the response is related to the query topic and meets the query's specific requirements. The *faithful* of the system evaluates if the generated response accurately reflects the information contained within the relevant documents and measures the consistency between generated content and the source documents. In the end, the correctness refers to the accuracy of the generated response against a sample response, which serves as a ground truth. It checks if the response is correct in terms of factual information and appropriate in the context of the query.

Current evaluation practices delineate **three primary quality metrics** - *context relevance*, *answer fidelity*, and *answer relevance* - that serve as critical metrics for assessing the dual goals of retrieval and generation within RAG frameworks [13]. **Context relevance** refers to the accuracy and specificity of the retrieved information, ensuring that the content is not only relevant, but also minimizes the cognitive load associated with extraneous or irrelevant data. This aspect is crucial for improving the efficiency of the retrieval process, as it directly influences the quality of the subsequent generation phase.

Answer faithfulness quality metric evaluates the integrity of the generated answers in relation to the retrieved context. It is imperative that the answers generated by the RAG model remain consistent with the retrieved information, avoiding hallucinations that could compromise the credibility of the system. This fidelity to the retrieved context is essential to maintain user confidence and ensure that the content generated is not only informative but also reliable. The **answer relevance** metric assesses the direct relevance of the generated answer to the queries posed. This metric is critical in determining the effectiveness of the RAG model in addressing the core questions, thereby increasing user satisfaction and engagement.

In addition to these quality scores, the evaluation of RAG system includes

four essential **abilities** [13] that reflect the adaptability and operational efficiency of the system: *noise robustness*, *negative rejection*, *information integration* and *counterfactual robustness.*

Noise robustness assesses the ability of the model to handle documents that may be related to the query, but may contain superfluous or misleading information. This ability is particularly important in real-world applications where the quality of retrieved documents can vary significantly.

Negative Rejection, on the other hand, assesses the model's ability to avoid generating answers when the retrieved documents lack the necessary knowledge to adequately answer a question. This ability is essential to ensure that the generated content is accurate and reflects critical engagement with the source material.

Information Integration evaluates the model's proficiency in synthesizing information from multiple documents to address complex questions.

Counterfactual Robustness tests the model's ability to recognize and disregard known inaccuracies within documents, even when instructed about potential misinformation.

Thus, contextual relevance and noise robustness are integral to the evaluation of retrieval quality, while response answer faithfulness, answer relevance, negative rejection, information integration and counterfactual robustness are paramount to the evaluation of generation quality.

It is important to consider that an effective information retrieval system will avoid the retrieval of documents that lack the necessary information to provide an answer, as well as documents that are not relevant to the search query. Consequently, during my internship, I focus on the evaluation of the information retrieval system during the synthetic data creation phase, as will be discussed in chapter 4.

3.3 Evaluation Framework

This section presents an overview of methodologies proposed in recent literature, emphasizing the necessity of establishing private benchmarks for effective assessment. Meanwhile, it will attempt to address the question of *how to evaluate.*

A number of studies have proposed methodologies for assessing the performance of RAG architectures, with each approach emphasizing a different dimension of evaluation and employ varying strategies for dataset construction, ranging from leveraging existing resources to generating entirely new data tailored for specific evaluation aspects.

One significant framework is RAGAS [10], which introduces a reference-free evaluation approach specifically designed for RAG pipelines. The framework identifies three fundamental quality aspects: faithfulness, answer relevance, and context relevance. Faithfulness ensures that the generated answers are based on the retrieved context, thereby reducing the risk of hallucinations, which is a significant concern in domains where factual accuracy is paramount, such as law [10]. By prompting a language model to measure these aspects, RAGAS provides a self-contained evaluation mechanism that does not rely on human-annotated datasets, which are often scarce in practical scenarios. Similarly, ARES (An Automated Evaluation Framework for Retrieval - Augmented Generation Systems) [47] offers a systematic approach to RAG evaluation by creating synthetic training data. ARES fine-tunes lightweight language model judges to assess the quality of individual RAG components across various tasks, including those in the KILT and SuperGLUE benchmarks [47].

This framework also incorporates a small set of human-annotated data points to enhance the accuracy of its predictions through a method known as prediction-powered inference (PPI) [48]. By focusing on context relevance, answer faithfulness, and answer relevance, ARES provides a robust evaluation mechanism that minimizes reliance on extensive human annotations, thus streamlining the evaluation process.

In contrast, INSPECTORRAGET [1] introduces an introspective platform for RAG evaluation, allowing users to analyze both aggregate and instance-level performance using a combination of human and algorithmic metrics. This tool addresses the need for rigorous evaluation by providing insights into annotator quality and the overall performance of RAG systems. Such introspective capabilities are essential for understanding the nuances of RAG performance and for identifying areas for improvement. The complexity of RAG systems is further compounded by the integration of additional modules, such as query rewriting and prompt compression.

AutoRAG-HP [49] addresses the challenges associated with hyper-parameter tuning in this context, proposing an automatic online tuning mechanism that adapts to user feedback. This approach emphasizes the importance of continuous evaluation and optimization, particularly in dynamic environments where user interactions can inform system performance.

Moreover, the IRSC [46] benchmark introduces a zero-shot evaluation framework specifically designed for information retrieval in RAG scenarios. The proposed benchmark addresses a significant gap in the existing evaluation landscape by providing a comprehensive assessment of RAG systems across a range of retrieval tasks.

Despite the multitude of proposed evaluation methodologies, a common thread can be identified: the **necessity for a private benchmark** to initiate the evaluation process. Each of these approaches is based on the assumption of a controlled environment in which synthetic data can be generated or specific retrieval tasks can be defined. In this context, my thesis will focus on the creation of synthetic private benchmarks and the evaluation of retrieval scores within RAG components, thereby making a contribution to the ongoing discourse concerning effective RAG evaluation methodologies. In fact, the creation and selection of datasets represent a crucial step in the evaluation of RAG systems. The use of datasets designed for specific metrics or tasks improves the accuracy of the evaluation process and provides guidance for the development of adaptable RAG systems that can meet the needs of real-world information requirements.

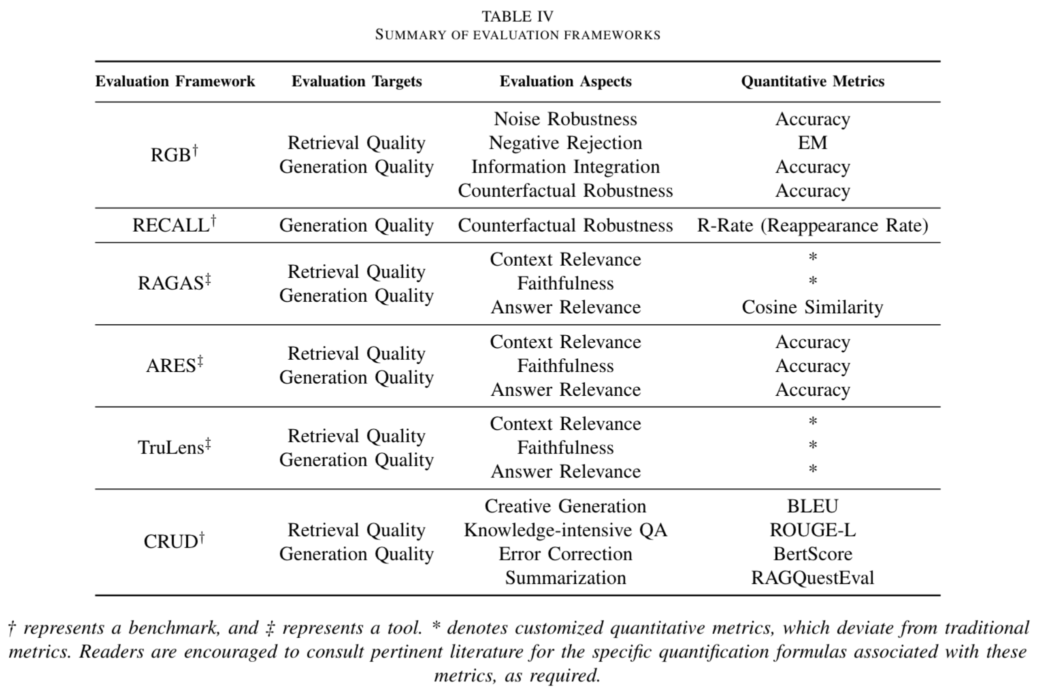


TABLE IV
SUMMARY OF EVALUATION FRAMEWORKS

Evaluation Framework	Evaluation Targets	Evaluation Aspects	Quantitative Metrics
RGB†	Retrieval Quality	Noise Robustness	Accuracy
	Generation Quality	Negative Rejection	EM
		Information Integration	Accuracy
		Counterfactual Robustness	Accuracy
RECALL†	Generation Quality	Counterfactual Robustness	R-Rate (Reappearance Rate)
RAGAS‡	Retrieval Quality	Context Relevance	*
	Generation Quality	Faithfulness	*
		Answer Relevance	Cosine Similarity
ARES‡	Retrieval Quality	Context Relevance	Accuracy
	Generation Quality	Faithfulness	Accuracy
		Answer Relevance	Accuracy
TruLens‡	Retrieval Quality	Context Relevance	*
	Generation Quality	Faithfulness	*
		Answer Relevance	*
CRUD†	Retrieval Quality	Creative Generation	BLEU
	Generation Quality	Knowledge-intensive QA	ROUGE-L
		Error Correction	BertScore
		Summarization	RAGQuestEval

*† represents a benchmark, and ‡ represents a tool. * denotes customized quantitative metrics, which deviate from traditional metrics. Readers are encouraged to consult pertinent literature for the specific quantification formulas associated with these metrics, as required.*

Figure 3.1: Table of evaluation frameworks from [13]

3.4 Metrics

This section will introduce several commonly used metrics for retrieval and generation components and will attempt to answer the question of *how to evaluate* and quantify.

Evaluation metrics are used to figure out how well RAG systems are doing. They give a consistent way to check how effectively a system pulls up relevant info and gives accurate answers. These metrics can also highlight if the system is retrieving irrelevant data or producing confusing answers, and identify potential weaknesses or areas for improvement. RAG systems are frequently tested with human evaluators feedback or automated metrics that compare the output to predefined correct responses in order to determine relevance. This makes it easier to assess how closely the content that was generated adheres to the original task or question.

However, establishing evaluative criteria that correspond with human preferences and address practical considerations presents a significant challenge. The RAG systems are composed of multiple components, each of which necessitates a distinct, tailored approach to evaluative criteria in accordance with its unique functionalities and objectives.

Retrieval Metrics

The most common metrics used for *retrieval component* can be rank or not rank based.

The *non-rank* based often assess binary outcomes (whether an item is relevant or not) without considering the position of the item in a ranked list.

- **Accuracy:** is the proportion of true results (both true positives and true negatives) among the total number of cases examined.

- **Precision:** is the fraction of relevant instances among the retrieved instances

$$\text{Precision} = \frac{TP}{TP + FP} \tag{3.1}$$

- **Recall** *at k:(Recall@k)* is the fraction of relevant instances that have been retrieved over the total amount of relevant cases, considering only the top k results.

$$\text{Recall@}k = \frac{|\text{Relevant Instances Retrieved in Top } k|}{|\text{Total Relevant Instances}|} \tag{3.2}$$

- **Hit Rate:** This metric is used to measure the percentage of times the system retrieves the correct context. A hit rate of 0.33 indicates that 33% of the time the system returns the correct chunk. This metric is used to measure context relevance.

$$\text{Hit Rate} = \frac{\text{Number of Correct Contexts}}{\text{Total Number of Contexts}}$$

Rank-Based Metrics evaluate the order in which relevant items are presented, with higher importance placed on the positioning of relevant items at the ranking list.

- **Mean Reciprocal Rank (MRR):** is the average of the reciprocal ranks of the first correct answer for a set of queries.

$$\text{MRR} = \frac{1}{|Q|}\sum_{i=1}^{|Q|}\frac{1}{\text{rank}_i} \tag{3.3}$$

  $|Q|$ is the number of queries. rank_i is the rank position of the first relevant document for the i-th query.

  It measure how well the system retriever the correnct chunks considering the position. The higher the MRR, the better the system is at putting the right context in upfront. Suppose we have three different query and single correct chunk

Query	Retrieval Order results	Correct chunk	Rank	Reciprocal Rank
Q1	chunk4, chunk2, chunk3	chunk3	3	$\frac{1}{3}$
Q2	chunk6, chunk7, chunk8	chunk6	1	1
Q3	chunk10, chunk11, chunk12	chunk12	2	$\frac{1}{2}$

  the mean reciprocal rank is:

$$\text{Mean Reciprocal Rank} = \frac{\left(\frac{1}{3}+1+\frac{1}{2}\right)}{3} = \frac{\left(\frac{1}{3}+\frac{3}{3}+\frac{1.5}{3}\right)}{3} = \frac{\frac{11}{6}}{3} = \frac{11}{18} \approx 0.61$$

- **Mean Average Precision (MAP)** is defined as the mean of the average precision scores for each query. It can be mathematically expressed as:

$$\text{MAP} = \frac{1}{|Q|}\sum_{q=1}^{|Q|}\frac{1}{|\text{relevant documents}_q|}\sum_{k=1}^{n}(P(k)\times\text{rel}(k)) \tag{3.4}$$

  where P(k) is the precision at cutoff k in the list, rel(k) is an indicator function equaling 1 if the item at rank k is a relevant document, 0 otherwise, and n is the number of retrieved documents.

Generation Metrics

In the context of text generation, the evaluation of generated responses extends beyond the mere accuracy of the text produced. Instead, it encompasses a range of linguistic qualities, including coherence, relevance, fluency, and alignment with human judgement. This requires the development of metrics that can assess the nuances of language production, including factual correctness, readability, and user satisfaction with the generated content.

The metrics are:

- **BLUE:** (Bilingual Evaluation Understudy): It is a metric for the evaluation of the quality of machine-translated text in comparison to one or more reference translations. BLEU assesses the precision of n-grams in the generated text in comparison to the reference text, and subsequently applies a brevity penalty with the objective of preventing translations that are excessively concise. It should be noted that BLEU has certain limitations, including an inability to account for aspects such as fluency or grammatical correctness in the generated text.

- **ROUGE:** (Recall-Oriented Understudy for Gisting Evaluation). It measures the overlap between the generated output and the ground truth response. It is a set of metrics designed to evaluate the quality of summaries by comparing them to human-generated reference summaries. The variants of ROUGEs measure the overlap of n-grams (ROUGE-N, ROUGGE-W), word subsequences (ROUGE-L, ROUGGE-S), and word pairs between the system-generated summary and the reference summaries.

- **BertScore:** The method is based on the contextual embedding from pre-trained transformers, such as BERT, which is used to evaluate the semantic similarity between generated text and reference text. BertScore employs a token-level approach to similarity calculation using contextual embedding, and produces precision, recall, and F1 scores. In comparison to n-gram-based metrics, BertScore is more robust to paraphrasing and more sensitive to semantic equivalence, as it captures the meaning of words in context.

- **LLM as a Judge [50]:** it represents a more recent methodology. In this approach, LLMs assess generated text based on criteria such as coherence, relevance, and fluency. LLMs may be fine-tuned on human judgments to enhance predictions of unseen text quality or employed in zero-shot or few-shot contexts. This strategy capitalizes on the LLM's linguistic and contextual comprehension to deliver a nuanced assessment of text quality. For example, Balanguer et al. [38] demonstrates that providing LLM judges with explicit scoring guidelines, such as a 1 to 5 scale, can standardize evaluations.

The incorporation of LLM as evaluative judges marks a significant advancement in the methodology of quality evaluation in the field of natural language processing. This methodology is particularly advantageous in scenarios where traditional ground truths are challenging to establish, thereby offering a versatile and automated solution for evaluating generated text. The strategic implementation of detailed prompt templates serves to further enhance this evaluative process, ensuring that assessments are aligned with human preferences and effectively standardizing evaluations across various content dimensions.

Despite the considerable promise of LLMs as automatic evaluative judges, a number of challenges remain. The alignment of automated assessments with human judgment remains a significant challenge, as differences in determining the correctness, clarity and quality of a given text may arise between automated and human evaluations. Moreover, the efficacy of example-based scoring is dependent on a number of factors, and there is no universally applicable scale for grading and prompting text, which presents a challenge to the standardisation of the LLM as a judge qualification. Also, the high computational demands of using LLMs for data generation and validation require a meticulous consideration of the associated constraints. In addition, Fu et al. [51] didn't find any significant correlations to human evaluations of factuality for GPT-4, PaLM-2, and Claude-2. It is therefore desirable to develop evaluation methodologies that can effectively assess RAG systems using smaller amounts of data while maintaining the validity and reliability of the results.

3.5 Benchmark and Synthetic data generation

A tipical RAG systems evaluation is done using fixed benchmark datasets. These datasets consist of (source(s), question, answer) tuples that enable separate evaluation of the retriever and the QA model [52].

The construction of distinct benchmarks for evaluating RAG systems employs a variety of strategies, ranging from the utilization of existing resources to the generation of entirely new datasets tailored for specific evaluative aspects. A number of benchmarks, including those derived from the KILT (Knowledge Intensive Language Tasks) benchmark [53], incorporate established datasets such as Natural Questions [54], HotpotQA [55], and FEVER [56], as well as components from SuperGLUE [57], including MultiRC [58] and ReCoRD [59]. However, the usage of these datasets is not without constraints, particularly with regard to the challenges posed by dynamic real-world and private business scenarios.

The advent of sophisticated LLMs has significantly impacted the process of dataset construction. The capacity to generate specific queries and ground truths aligned with evaluation targets enables the generation of datasets in desired formats with relative ease. It is notable that the RGB [60], MultiHop-RAG [61], CRUD-RAG [62], and CDQA [63] benchmarks exemplify this innovative approach, utilizing online news articles to rigorously assess the capabilities of RAG systems. These benchmarks extend beyond the scope of traditional training data, thereby enabling a more comprehensive evaluation of LLMs' proficiency in processing and synthesising real-world information.

This was also the approach developed in my internship use case: we use LLaMAntino-3-ANITA-8B-Inst[6] as LLM in order to generate a query from each chunks. The necessity for the creation of a private benchmark was also

[6] The model is an instruction-tuned version of Meta-Llama-3-8b-instruct (a fine-tuned LLaMA 3 model).

driven by the fact that the existing benchmarks lacked the inclusion of private data, which made them unsuitable for evaluating the use case of the chatbot.

The generation and selection of data sets represent a critical step in the evaluation of RAG systems. The initial challenge, therefore, is to establish a benchmark based on one's own private data, followed by the identification of optimal configurations. Consequently, my efforts have been focused on the creation of a benchmark and the simultaneous comparison and assessment of the RAG system.

Chapter 4

Proposal for industry RAG evaluation: ***G***enerative ***U***niversal ***E***valuation ***L***LM and ***I***nformation retrieval

4.1 Problem

A RAG system consists of a retriever and a large language model. In response to a user query, the retriever identifies most relevant content from a corpus, which the LLM then use to generate a response. This formulation allows for a multitude of design choices, including the selection of an appropriate retrieval method, chunk size, splitting technique and embedding model. It is not always the case that a RAG system will perform optimally in all contexts. Its efficacy may vary depending on the specific data domain, corpus size and cost budget in question. Therefore, the creation of a universal architecture for all use cases is not a viable solution; rather, it is essential to analyze each case individually in order to identify the most suitable parameters.

In order to optimize RAG systems, traditional methodologies require human annotators for query generation, finding relevant context for evaluating the performance of the retriever, and providing domain-specific answers. Although these strategies require considerable investment in annotation, the use of a model-based evaluation approach can help to mitigate this cost [47]. For example, the open-source RAGAS framework [10] prompts a language model to assess the relevance of retrieved information and the fidelity of generated responses. Furthermore, LLMs can be employed to construct a benchmark dataset from scratch [47]. The ultimate objective is to evaluate the performance of the embedding model and retrieval capabilities [46] in order to identify the most effective options. Some studies [12] have concentrated on identifying the most effective RAG techniques, including Maximum Marginal Relevance, sentence-window retrieval, document summary indexing, HyDe, and multi-query approaches. In the context of business applications, however, the creation of a benchmark based on proprietary data is an essential process. For example, in the development of an AI chatbot, organizations must construct a benchmark from scratch, relying exclusively on their internal domain documents if they lack a preexisting collection of question-answer pairs.

The generated dataset composed of **questions, contexts, and answers** derived from private data can then be used to evaluate the performance of the vector database retriever and also serve as a basis for fine-tuning.

4.2 Proposal

To achieve this efficiently, we recognize that the manual production of hundreds of question-context-answer examples from documents is a time-consuming and labour-intensive process. Accordingly, the generation of synthetic test data using a LLM represents an efficient approach that minimizes both time and effort [46].

As RAGAS highlight [64] the type of query in RAG system can be categorize into two single-hop and multi-hop. The first one is a straightforward question that requires retrieving information from a single document or source to provide a relevant answer. The context can be reached with a single step. In contrast, a multi-hop query necessitates the utilization of information from two or more sources, and is characterized by a series of reasoning steps. It is necessary for the system to retrieve information from a variety of documents and to establish connections between them in order to generate an accurate answer.

However, the assessment of the quality of the generated dataset by a human evaluator remains a necessary step. Consequently, in this case, an LLM-based evaluation method is not employed.

The objective of this evaluation is to assess the performance of the embedding model representation. This requires determining whether the chosen embedding model is capable of capturing and representing the numerical characteristics of the domain data in an appropriate manner. This is a similar objective to that proposed by Lin et al. [46].

4.2.1 Available framework for data generation

At the time of writing, various frameworks exist for creating synthetic training data. The main one is RAGAS, followed by ARES. In particular, RAGAS has recently changed its approach to generating synthetic data, using a Knowledge Graph to ask abstract questions, for those that require more context [64]. Below we illustrate the approach employed into ARES, RAGAS.

4.2.1.1 RAGAS in synthetic dataset

An ideal evaluation dataset should include a representative sample of the types of questions encountered in production, including questions of varying difficulty levels. It is a characteristic of LLM models that they are not able to create diverse samples, as they tend to follow common paths. In order to achieve this, RAGAS employs an evolutionary generation paradigm, whereby questions with varying characteristics, including those requiring reasoning, conditioning, and multi-context, are systematically generated from the provided set of documents. This approach (version 0.1.21 [65]) served as the primary source of inspiration for this thesis study. In contrast, the new paradigm of version 0.2.5 [64] employs a knowledge graph during the splitting phase.

Old approach

Two LLMs are used: the *generator LLM*, which aims to generate questions and potential answers, and the *critic LLM*, which evaluates the quality of these generated outputs. In addition, an embedding model is used to represent textual data into numerical vectors, enabling similarity calculations.

The generation of synthetic QA data begins with the application of a series of transformations, which are categorised into three different types:

- **Reasoning:** Rewrite the question in a way that enhances the need for reasoning to answer it effectively.
- **Conditioning:** Modify the question to introduce a conditional element, which adds complexity to the question.
- **Multi-Context:** Rephrase the question in a manner that necessitates information from multiple related sections or chunks to formulate an answer.

With this approach, not only diversifies the question set, but also enriches the complexity and depth of the questions generated.

The LLM validator is then used to check if the generated questions maintained their relevance and answerability. At the same time, a validation step assesses the quality of the answers generated by the generator LLM, evaluating their accuracy, coherence and relevance to the corresponding questions. This two-layer validation process is essential to filter out poor quality output and ensure that only robust QA pairs are retained.

Upon successful validation, the questions that meet the established criteria are classified as valid QA samples, culminating in the creation of a synthetic QA dataset.

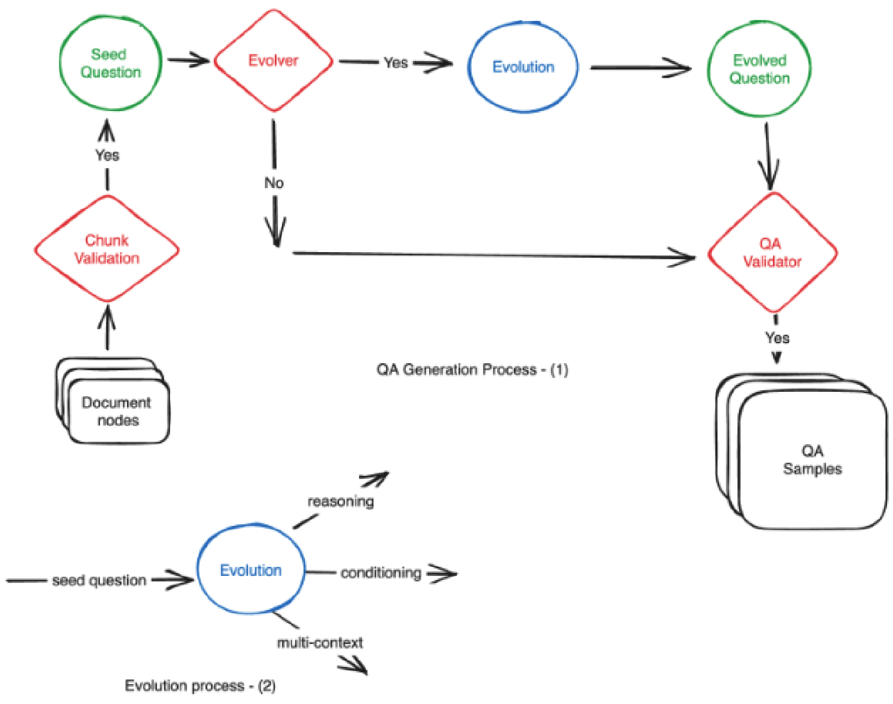


Figure 4.1: Generation step in RAG synthetic Test Data generation [65]

Recent approach

In order to generate different types of queries from a given corpus of documents, a main obstacle is the identification of appropriate segments or documents that allow large language models to formulate these queries. RAGAS addresses this problem using a Knowledge Graph for test-set generation.

The initial phase of the RAGAS framework involves the systematic segmentation of documents into hierarchical nodes. This chunking process is flexible, using different splitters adapted to the specific document type. For example, in the case of financial documents, chunking can be performed using a splitter that delineates sections such as the income statement, balance sheet and cash flow statement. This hierarchical organization of information facilitates more detailed data extraction, which is critical for the following steps of query generation.

To extract relevant information from these nodes, RAGAS uses a combination of extractors, which can be divided into LLM-based extractors and rule-based extractors. The relationships between the nodes are established based on the extracted information, which is essential to maintain the integrity of the knowledge graph.

Once the knowledge graph is constructed, it serves as a robust query generation tool. By traversing the graph, RAGAS can generate a variety of queries informed by the relationships and data encapsulated within the nodes. For example, to formulate a query such as "Compare the revenue growth of Company X and Company Y from FY2020 to FY2023", the graph traversal mechanism identifies the relevant nodes that contain the necessary information about the revenue growth of the specified companies over the specified fiscal years. This capability highlights the effectiveness of the knowledge graph approach in generating contextually relevant queries that are highly correlated with the data. This not only improves the quality of the queries generated, but also helps to provide a more complete understanding of the information contained in private documents [64].

4.2.1.2 ARES in synthetic dataset

Starting from the corpus of in-domain passages, the ARES [47] framework is implemented in three stages. First, an LM is used to generate a synthetic dataset of question-answer pairs derived from the passages within the corpus. Secondly, three different judge models are defined, each with a specific classification task: context relevance, answer faithfulness and answer relevance. The judges are lightweight models fine-tuned to a contrastive learning objective. Third, ARES uses prediction-powered inference (PPI; Angelopoulos et al., 2023) to evaluate the different RAG systems, thereby improving the accuracy of model-based evaluation and providing statistical confidence intervals for RAG scoring. PPI uses a limited number of human-annotated data points to compute its confidence intervals. This annotated set is referred to as the Human Preference Validation Set and comprises approximately 150 data points or more, providing both positive and negative examples of context relevance, response fidelity and response relevance.

LLM Generation of Synthetic Dataset

The method devolved in ARES generate synthetic queries and corresponding answers from the corpus passages using generative LLMs. The resulting data includes both positive and negative instances of query-passage-answer triples, including relevant and irrelevant passages as well as correct and incorrect answers. For the generation process, the LLM uses an input set of few-shot examples consisting of in-domain passages aligned with in-domain queries and answers. The prompt used is:

> Example#1
>
> Document: <few-shot example here>
>
> Query: <few-shot example here>
>
> Example#2
>
> Document: <few-shot example here>
>
> Query: <few-shot example here>
>
> Example#3
>
> Document: <few-shot example here>
>
> Query: <few-shot example here>
>
> Example#4 Document: <in-domain passage>
>
> Query:

The model then generates a synthetic question and answer based on a given in-domain passage, allowing the generation of both positive and negative training examples. In developing their synthetic data, they primarily use the FLAN-T5 XXL model, but their system is designed to work also with other high quality models for generating synthetic queries and answers. To ensure the quality of the generated queries, they implement a filtering mechanism that assesses whether a given query can successfully retrieve its original passage as a top result using its retriever. This filtering strategy has been used in previous research to isolate high-quality synthetic queries (Dai et al., 2022; Saad-Falcon et al., 2023).

4.3 Benchmark Creation: Synthetic data generation

As the capabilities of LLMs have increased, particularly in their ability to follow instructions [2], the component that we have selected for evaluation is the retrieval system. In particular, we are investigating the relationship between chunk size and embedding model.

The principal challenge encountered during the internship was the utilisation of tools that operated without the integration of closed LLMs. This was due to the fact that the company in question was reluctant to share personal data with third-party organisations, preferring to maintain the integrity of its internal ecosystem by retaining all data within its own system. For example, RAGAS is not the optimal choice for smaller LLMs, as they are not always able to generate the requisite output in the desired JSON format. In order to overcome these limitations and establish an internal benchmark that could be refined in the future, it was necessary to select an LLM that could be run locally and did not rely on external APIs. The decision was made to use llama3.1 8b-instruct-q4_K_M and gemma2 9b-instruct-q3_K_M models, which were deployed on the Ollama service running on local machine (see A for specifications).

The selection of these models was based on a number of factors. The primary consideration was the maximum capacity that the available graphics card could accommodate without exceeding the memory buffer limit. For this reason, a quantized model was selected, offering an appropriate balance between performance and computational cost.

Furthermore, in view of the fact that the use case was the RAG, it was essential for the LLM to comply with the instructions set forth in the prompt. Consequently, an instruction model was identified as the most appropriate approach, and this was subsequently employed to generate the questions related to the provided context.

The following section will provide a detailed account of this implementation.

Algoritm

We first split the text document into a given *chunk_ size* using the recursive text splitter method and then we store these in in-memory qdrantDB vector store. Now, for each context that is larger than 100 characters, we create a question using an LLM with the following prompt:

> Il tuo compito è quello di formulare una domanda a partire da un contesto dato, rispettando le regole indicate di seguito:
> 1. La domanda deve avere una risposta completa a partire dal contesto dato.
> 2. La domanda deve essere formulata a partire da una parte che contiene informazioni non banali.
> 3. La risposta non deve contenere collegamenti.
> 4. La domanda deve essere di difficoltà moderata.
> 5. La domanda deve essere ragionevole e deve poter essere compresa e risolta dagli esseri umani.
> 6. Non usare frasi come "ha fornito il contesto", ecc. nel contesto della domanda.
> Contesto:
>
> ———
>
> contesto
>
> ———
>
> Rispondi con solo la domanda riformulata.

Once all the questions are created, we then filter out low-quality queries by testing if a given query can retrieve its original passage as the top result using its retriever, as done by Saad-Falcon et al. in [47].

If the retriever returns the original context where the question came from, we assign a score of 1, 0 otherwise. So, we perform the *hit rate* metrics by sum up the score and dividing by the total number of context.

To sum up the steps are:

1. Load raw text data from source;
2. Split the texts into chunks using RecursiveCharacterTextSplitter;
3. Create a DataFrame with the follow strucuture:
    - *ID:* each rows rappresent a context with a unique id, in order to check if the retrieved content if the same as the contenxt.
    - *question*: the generated query by the LLM starting from the context;
    - *context*: the spitted document is considered as the context provided if the lenght of the chunks is grater than a treeshold;
    - *retrieved_ content*: the content retrieved from the system given the query;
    - *retrieved_ id*: the chunk ID retrieved by the retriever system;
    - *score*: 1 if the retrieved context is equals to the context, 0 otherwise;
4. Create a comparison table for all combinations found

The generation of benchmark files is a process that requires a significant amount of time. This is due to the fact that a single question is generated for each individual chunk. Consequently, the process of testing the 30 combinations, as part of my internship, required approximately five hours of runtime.

In particular, the configuration that was chosen for testing was as follows:

- Three different chunk size: 500, 1000, 1500;
- Two LLMs: gemma2 and llama3.1;
- Five embedding model available from sentence Transformes [66]: msmarco-MiniLM-L-6-v3 , paraphrase-multilingual-mpnet-base-v2 , multi-qa-mpnet-base-dot-v1, all-mpnet-base-v2, all-MiniLM-L6-v2;

Based on different chunk size of 500, 1000 and 1500 we obtain respectably 473, 231, 142 questions. It was decided to use the sentence transformer models as they could be integrated into the Python environment without the need for external application programming interfaces (APIs). The decision to utilise an embedding model was based on the available benchmark [67], with consideration given to general-purpose and multilingual models.

In particular, multi-QA models are selected due to their training for semantic search. In response to a query, these models are capable of identifying relevant text passages [67]. The MSMARCO passage models have been selected due to their training on the MSMARCO Passage Ranking Dataset [68], comprising 500,000 real queries from Bing search and the pertinent passages from a range of web sources. The diversity of the MSMARCO dataset enables models to perform well in other domains. For further information regarding the embedding of parameters, see reference B.1.

4.4 Internship use case: results and comment

In section 2.4.3, we delineated the Lego-like architectural framework of RAG system, which enables the simple combination of diverse parameters, including chunk size, LLMs and embedding models. To implement this flexibility, a python script has been developed and made publicly available on GitHub page. This script has been developed with the specific objective of establishing benchmarks for all possible combinations.

The subjects addressed in the documents were diverse, encompassing ESG topics, the Italian tax system, and the code of conduct. Therefore, the private data also included information that the LLM was unlikely to observe during the training phase, such as internal regulations and specific rules of conduct.

Following the generation of the benchmarks, an empirical analysis was conducted to evaluate the type and quality of the questions generated. An alternative approach would have been to use a ReAct-style LLM agent [69] to evaluate the questions and then conduct a human evaluation. However, it was established that the evaluation would be conducted manually. As Yi Zhang [70] proposes in his practical guide to RAG pipeline evaluation, the optimal methodology for reliable evaluation is to have a human-verified contextual dataset. The advantage of curating this dataset is that the process of verification is not repeated, as is necessary for LLM-based metrics without ground truth.

The analysis demonstrated that, in general, the questions generated predominantly began with the phrases "What is" or "What are" when the chunk size was set to 500, regardless of the LLM utilized. When the context was expanded to 1000, the majority of the questions continued to exhibit the previous structure. However, a subset of these questions start with "Based on" and incorporated part of the context. This formulation is unlikely to be formulated by the end user, who, in contrast, tends to anthropomorphize the chatbot, making a number of assumptions about the information provided. In the case of chunk sizes of 1500, some questions were not contextualized adequately or were phrased in a way that did not align with the information provided. This indicate a tendency of LLM to focus on the final part of the prompt, rather than considering the context as a whole. At the same time, some questions were both pertinent and accurately phrased.

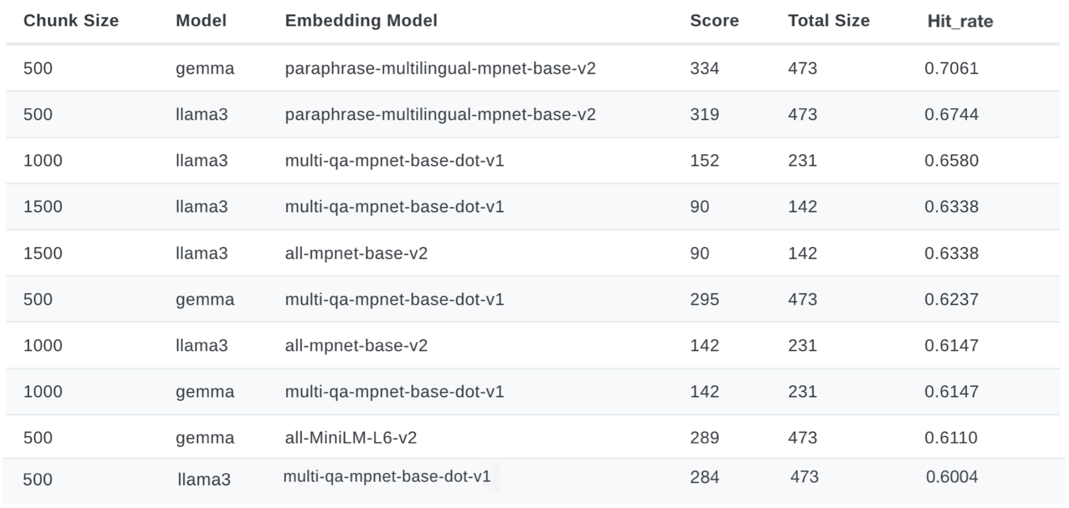
In general, it was found that questions assigned a score of zero by the retrieval system were more general in nature and did not pertain specifically to the context provided.

It can be observed that the lowest-performing LLM in question generation was gemma2 with 1500 as chunk size. So, an excessive amount of contextual information is incompatible with the generation of questions by this model.

In contrast, the Llama3.1 model is capable of formulating a highly specific question in response to the given context, even when the context is of a greater length.

Result

The result are presented in the follows image:



Chunk Size	Model	Embedding Model	Score	Total Size	Hit_rate
500	gemma	paraphrase-multilingual-mpnet-base-v2	334	473	0.7061
500	llama3	paraphrase-multilingual-mpnet-base-v2	319	473	0.6744
1000	llama3	multi-qa-mpnet-base-dot-v1	152	231	0.6580
1500	llama3	multi-qa-mpnet-base-dot-v1	90	142	0.6338
1500	llama3	all-mpnet-base-v2	90	142	0.6338
500	gemma	multi-qa-mpnet-base-dot-v1	295	473	0.6237
1000	llama3	all-mpnet-base-v2	142	231	0.6147
1000	gemma	multi-qa-mpnet-base-dot-v1	142	231	0.6147
500	gemma	all-MiniLM-L6-v2	289	473	0.6110
500	llama3	multi-qa-mpnet-base-dot-v1	284	473	0.6004

Top combinations result

We find that the combination with the high hit rate is a chunk size of 500 using parapharase-multilingual-mpnet-based-v2 and gemma2-inst as LLM. What we can notice in this top 10 configuration is that based on the granularity we choose we have a different embedding model capable of capturing the meaning. With regard to the private data on which the experiment was conducted, it can be observed that the embedding model (paraphrase-multilingual-mpnet-based-v2) appears to be an excellent candidate capable of mapping sentences and paragraphs in a dense vector space of 768 dimensions. In contrast with the observations made in the embedding model benchmark [67], the all-mpnet-base-v2 model is not positioned within the top ranks, indicating that it may not be the optimal choice for our semantic search task.

We perform also an empirical evaluation to assess the undesired behaviour of retrieving chunks when the query is outside the data content. So in the retriever system we set up the score_thresholds[7] parametrize to 0.6, to get the desired behavior regardless of the embedding model used. What we notice is

[7] In this way we return chunks only if the similiraty is high to 0.6

that the only embedding model that doesn't return anything, even with the score_thresholds param set to 0.3, is paraphrase-multilingual-mpnet-base-v2. So this enforce the result and the choose of this emebdding model for the in question RAG system.

4.5 Limitation and future work

One limitation of this approach is that, in the absence of human intervention, the question generated by the LLM from the context is assumed to be accurate and reasonable. Consequently, once the dataset has been refined through human evaluation, the calculation of the metrics must be conducted again. Despite this, it constitutes an excellent point of departure for the creation of a benchmark.

Further work could include the calculation of the Mean Reciprocal Rank (MRR) as a metric for the retrieval system, with an assessment of whether the initial context is ranked within the top three positions of the returned chunks.

Another aspect of this approach is that it required a comprehensive examination of all potential parameter combinations. An alternative strategy would be to implement hyperparameter tuning on a selected set of RAG parameters, similar to the methodology employed by RagBuilder[8].

[8] It is a toolkit that helps to automatically create an optimal RAG (Retrieval-Augmented-Generation) configuration ready for production for one's own data

Chapter 5

Conclusion

During the course of my internship thesis, I was able to investigate a novel methodology for the generation of benchmarks. The use case in question, namely the chatbot trained to provide assistance on a specific training course, was an additional service that the company wished to offer. For this reason the company did not already possess a database from which to create a benchmark. It was thus necessary to create a benchmark from scratch, one that would require minimal human intervention initially. Given these circumstances, the approach of using LLMs was the optimal solution for our needs. In particular, they had to be used locally due to the reluctance to share training information outside the company perimeter. The work then focused more on finding the best embedding model for the use case, and we also asked which chunk size was most appropriate. The final aim was to create a RAG system that would be able to answer the specific question about the course content, because if we had asked the model directly, it would certainly have come up with a plausible answer. Indeed, one of the problems that the RAG technique seeks to address is hallucination in LLMs. However, it is not possible to completely eliminate this phenomenon. The results demonstrate that LLMs are unable to learn all of the computable functions, which inevitably leads to the occurrence of hallucinations. Given that the formal world is a subset of the real world, which is inherently more complex, it follows that hallucinations are an inevitable con-sequence for real-world LLMs. In the absence of appropriate safeguards, it is not recommended to utilise LLMs for critical decision-making processes. Furthermore, in the absence of human supervision, it is not appropriate to utilise LLMs in situations where the outcome could have significant implications for human wellbeing or safety. The process of making decisions that have a significant impact on safety is one that requires the application of rational and humane judgement, which encompasses elements such as comprehension, empathy, and ethical reasoning. It is challenging, if not impossible, for algorithms to compute these types of judgements. However, it would be inaccurate to view hallucination as entirely negative. Firstly, the decision to utilise an LLM is fundamentally a trade-off between precision and efficiency, which is largely determined by the specific application in question. LLMs are particularly effective at processing and structuring information at scales and speeds that are unachievable by humans, thereby facilitating rapid decision-making and idea generation. In scenarios where speed and volume of information processing are significant, the occasional inaccuracies of LLMs can be viewed as acceptable compromises. Conversely, in situations where precision is critical, the outputs of LLMs must be verified and supplemented with human supervision [71].

Machines are designed and constructed to execute a specific set of functions in a predictable manner. Their capacity for predictable functioning is a prerequisite for their utility; if machines were to operate in an unpredictable manner, their efficacy would be compromised. This reliability is particularly beneficial, as machines are capable of repetitively performing the same tasks without fatigue, in contrast to humans, who are prone to boredom and distraction, which can lead to errors. The advent of robotics and artificial intelligence based on neural networks has led to computers becoming increasingly involved in processing data derived from the real world. This data may include information generated by quantum events and human decisions. This evolution enables computers to function beyond the confines of classical deterministic systems, presenting both challenges and opportunities that remain, even for experts in the field, to be fully understood. The variability of complex patterns, such as faces, words or urban traffic situations, is considerable. As a result, the recognition rate cannot be perfect, particularly in unfavourable environmental conditions. It follows that the performance of robots or computers cannot be guaranteed with the same degree of confidence as that afforded by deterministic algorithms. It is therefore evident that if computers are allowed to make significant decisions based on the presumed reliability of their pattern recognition, the potential for disastrous consequences is significant. This issue is particularly pressing given that computers lack comprehension of the contexts in which they operate. It is therefore evident that ethical decision-making cannot be relegated to algorithms. There is an insurmountable gap between the capabilities of artificial intelligence and those of the human mind. This is exemplified by the unique attribute of *understanding*, which is frequently undermined and challenging for computers to replicate. Another crucial concept that holds precise meaning for a machine but not for a living organism is that of "completion". Upon leaving the factory, a machine is considered complete; once finished, it remains essentially unchanged. In contrast, living organisms are in a constant state of change, transformation, and evolution. They are never complete, as they are always in a state of becoming. The dynamism of a living organism is **irreducible** and remarkable, whether it manifests as a bacterium or a human being. Life can be conceptualized as a flame that produces other flames through division [72].

The current trend is to utilise an alternative LLM to assess or quantify the quality of the generation output of a preceding LLM. This process can continue indefinitely, and it is up to us to determine the extent to which we rely on these stochastic generators. This is particularly important for activities where the consequences of their choices have a significant impact on the lives of others. It is therefore beneficial and useful to employ these artificial intelligence tools, but they require caution in their use.

Back in 1975, computer scientist Joseph Weizenbaum made a prescient observation while studying interactions with ELIZA, an early natural language processing programme. He noticed the deep emotional bond that users established with the computer and their tendency to anthropomorphise it, despite the relatively simple programming. In his work, 'The Power of Computers and Human Reason', he wrote:

> Decisions made by the general public about emerging technologies depend much more on what the public attributes to those technologies than on what they actually are or can or cannot do.

As we navigate the era of increasingly sophisticated AI, Weizenbaum's early warnings are a timely warning. While embracing the potential of these tools, it is essential to remain aware of their inherent limitations and the human tendency to attribute capabilities to them that go beyond their actual functions.

Appendix A

Local Machine Configuration for Experimental Setup

The experiments were performed using a machine equipped with the following specifications:

- **Hardware Specifications**
  - **Graphics Processing Unit (GPU)**:
    - Model: NVIDIA GeForce RTX 3070
    - GPU Architecture: GA104 [GeForce RTX 3070 Lite Hash Rate]
  - **Central Processing Unit (CPU)**:
    - Model: 12th Gen Intel(R) Core(TM) i7-12700F
  - **Memory**:
    - Configuration: 2x16 GB
    - Type: DIMM DDR4 Synchronous
    - Speed: 3200 MHz
    - Latency: 0.3 ns
- **Operating System**
  - **Kernel Version**:

```
Linux 5.15.0-117-generic x86_64
```

- **Software Environment** The experiments were conducted using the following software environment:
  - **Large Language Model (LLM) Framework**:
    - Version: Ollama (version 0.1.29)
  - **Containerization Platform**:
    - Version: Docker (version 26.0.1)

Appendix B

Embedding models used

The table below provides an overview of the embedding models chosen.

Table B.1: Embedding Model Parameters

Description	**Base Model**	**Max Sequence Length**	**Dimension**	**Suitable Score Functions**	**Size**	**Params**
msmarco-MiniLM-L-6-v3	MSMARCO	256	384	Cosine Similarity, Dot Product	90.9 MB	27.7M
paraphrase-multilingual-mpnet-base-v2	Teacher:paraphrase-mpnet-base-v2; Student:xlm-roberta-base	128	768	Cosine Similarity	970 MB	278M
multi-qa-mpnet-base-dot-v1	-	512	768	Dot Product	420 MB	109M
all-mpnet-base-v2	microsoft/mpnet-base	384	768	Cosine Similarity, Euclidean Distance, Dot product	420 MB	109M
all-MiniLM-L12-v2	microsoft/MiniLM-L12-H384-uncased	256	384	Cosine Similarity, Euclidean Distance, Dot product	120 MB	33.4M

Bibliography

[1] K. Fadnis, S. S. Patel, O. Boni, *et al.* "InspectorRAGet: An Introspection Platform for RAG Evaluation." arXiv: `2404.17347 [cs]`. (Apr. 26, 2024), pre-published.

[2] H. Naveed, A. U. Khan, S. Qiu, *et al.* "A Comprehensive Overview of Large Language Models." arXiv: `2307.06435 [cs]`. (Feb. 20, 2024), pre-published.

[3] J. Weizenbaum, *Computer Power and Human Reason: From Judgment to Calculation*, 1st. USA: W. H. Freeman & Co., 1977, ISBN: 0716704633.

[4] M. U. Hadi, R. Qureshi, A. Shah, *et al.*, "A survey on large language models: Applications, challenges, limitations, and practical usage," *Authorea Preprints*, 2023.

[5] T. A. Chang and B. K. Bergen, "Language Model Behavior: A Comprehensive Survey," *Computational Linguistics*, vol. 50, no. 1, pp. 293–350, Mar. 1, 2024, ISSN: 0891-2017. DOI: `10.1162/coli_a_00492`. [Online]. Available: `https://doi.org/10.1162/coli_a_00492`.

[6] Umar Jamil, director, *Attention is all you need (Transformer) - Model explanation (including math), Inference and Training*, May 28, 2023.

[7] Y. Liu, H. He, T. Han, *et al.* "Understanding LLMs: A Comprehensive Overview from Training to Inference." arXiv: `2401.02038 [cs]`. (Jan. 5, 2024), pre-published.

[8] A. Vaswani, N. Shazeer, N. Parmar, *et al.* "Attention Is All You Need." arXiv: `1706.03762 [cs]`. (Aug. 2, 2023), pre-published.

[9] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, *Bert: Pre-training of deep bidirectional transformers for language understanding*, 2019. arXiv: 1810.04805 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.04805.

[10] S. Es, J. James, L. Espinosa Anke, and S. Schockaert, "RAGAs: Automated Evaluation of Retrieval Augmented Generation," in *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations*, N. Aletras and O. De Clercq, Eds., St. Julians, Malta: Association for Computational Linguistics, Mar. 2024, pp. 150–158.

[11] P. Lewis, E. Perez, A. Piktus, *et al.* "Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks." arXiv: 2005.11401 [cs]. (Apr. 12, 2021), pre-published.

[12] M. Eibich, S. Nagpal, and A. Fred-Ojala. "ARAGOG: Advanced RAG Output Grading." arXiv: 2404.01037 [cs]. (Apr. 1, 2024), pre-published.

[13] Y. Gao, Y. Xiong, X. Gao, *et al.* "Retrieval-Augmented Generation for Large Language Models: A Survey." arXiv: 2312.10997 [cs]. (Mar. 27, 2024), pre-published.

[14] "Retrieval Augmented Generation: Streamlining the creation of intelligent natural language processing models." (), [Online]. Available: https://ai.meta.com/blog/retrieval-augmented-generation-streamlining-the-creation-of-intelligent-natural-language-processing-models/.

[15] "Experience API (xAPI) Standard," ADL Initiative. (), [Online]. Available: https://adlnet.gov/projects/xapi/.

[16] S. Ranganath. "RAG 101: Chunking Strategies," Medium. (), [Online]. Available: https://towardsdatascience.com/rag-101-chunking-strategies-fdc6f6c2aaec.

[17] S. Todeschini. "How to Chunk Text Data — A Comparative Analysis," Medium. (), [Online]. Available: https://towardsdatascience.com/how-to-chunk-text-data-a-comparative-analysis-3858c4a0997a (visited on 10/18/2024).

[18] "Conceptual guide | LangChain." (), [Online]. Available: https://python.langchain.com/docs/concepts/.

[19] A. Lenci and M. Sahlgren, "Distributional semantics," 2023. [Online]. Available: https://api.semanticscholar.org/CorpusID:261635041.

[20] Umar Jamil, director. "Retrieval Augmented Generation (RAG) Explained: Embedding, Sentence BERT, Vector Database (HNSW)." (Nov. 27, 2023).

[21] M. Sahlgren, "The word-space model: Using distributional analysis to represent syntagmatic and paradigmatic relations between words in high-dimensional vector spaces," 2006. [Online]. Available: https://api.semanticscholar.org/CorpusID:11917163.

[22] T. K. Landauer and S. T. Dumais, "A solution to plato's problem: The latent semantic analysis theory of acquisition, induction, and representation of knowledge.," *Psychological Review*, vol. 104, pp. 211–240, 1997. [Online]. Available: https://api.semanticscholar.org/CorpusID:1144461.

[23] T. Mikolov, K. Chen, G. Corrado, and J. Dean, *Efficient estimation of word representations in vector space*, 2013. arXiv: 1301.3781 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1301.3781.

[24] N. Reimers and I. Gurevych, *Sentence-bert: Sentence embeddings using siamese bert-networks*, 2019. arXiv: 1908.10084 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1908.10084.

[25] S. Aquino. "An Introduction to Vector Databases - Qdrant." (), [Online]. Available: https://qdrant.tech/articles/what-is-a-vector-database/.

[26] S. Aquino. “An Introduction to Vector Databases - Qdrant.” (), [Online]. Available: https://qdrant.tech/articles/what-is-a-vector-database/.

[27] “Indexing - Qdrant.” (), [Online]. Available: https://qdrant.tech/documentation/concepts/indexing/.

[28] G. Kamradt. “5_Levels_Of_Text_Splitting.ipynb at main · FullStackRetrieval-com/RetrievalTutorials,” GitHub. (), [Online]. Available: https://shorturl.at/OM8QR.

[29] “MultiVector Retriever |LangChain.” (), [Online]. Available: https://python.langchain.com/docs/how_to/multi_vector/.

[30] L. Weng. “Prompt Engineering.” (), [Online]. Available: https://lilianweng.github.io/posts/2023-03-15-prompt-engineering/.

[31] J. Liu, D. Shen, Y. Zhang, B. Dolan, L. Carin, and W. Chen. “What Makes Good In-Context Examples for GPT?” arXiv: 2101.06804. (Jan. 17, 2021), pre-published.

[32] H. Su, J. Kasai, C. H. Wu, *et al.* “Selective Annotation Makes Language Models Better Few-Shot Learners.” arXiv: 2209.01975. (Sep. 5, 2022), pre-published.

[33] S. Ye, H. Hwang, S. Yang, H. Yun, Y. Kim, and M. Seo. “Investigating the Effectiveness of Task-Agnostic Prefix Prompt for Instruction Following.” arXiv: 2302.14691. (Dec. 24, 2023), pre-published.

[34] J. Wei, X. Wang, D. Schuurmans, *et al.* “Chain-of-Thought Prompting Elicits Reasoning in Large Language Models.” arXiv: 2201.11903. (Jan. 10, 2023), pre-published.

[35] langchain ai, *Rag from scratch*, https://github.com/langchain-ai/rag-from-scratch.

[36] H. Trivedi, N. Balasubramanian, T. Khot, and A. Sabharwal. "Interleaving Retrieval with Chain-of-Thought Reasoning for Knowledge-Intensive Multi-Step Questions." arXiv: 2212.10509. (Jun. 23, 2023), pre-published.

[37] G. Kim, S. Kim, B. Jeon, J. Park, and J. Kang. "Tree of Clarifications: Answering Ambiguous Questions with Retrieval-Augmented Large Language Models." arXiv: 2310.14696. (Oct. 23, 2023), pre-published.

[38] A. Balaguer, V. Benara, R. L. d. F. Cunha, *et al.* "RAG vs Fine-tuning: Pipelines, Tradeoffs, and a Case Study on Agriculture." arXiv: 2401.08406. (Jan. 30, 2024), pre-published.

[39] Y. Gao, Y. Xiong, M. Wang, and H. Wang. "Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks." arXiv: 2407.21059. (Jul. 26, 2024), pre-published.

[40] X. Ma, Y. Gong, P. He, H. Zhao, and N. Duan. "Query Rewriting for Retrieval-Augmented Large Language Models." arXiv: 2305.14283. (Oct. 23, 2023), pre-published.

[41] W. Shi, S. Min, M. Yasunaga, *et al.* "REPLUG: Retrieval-Augmented Black-Box Language Models." arXiv: 2301.12652. (May 24, 2023), pre-published.

[42] Z. Shao, Y. Gong, Y. Shen, M. Huang, N. Duan, and W. Chen. "Enhancing Retrieval-Augmented Large Language Models with Iterative Retrieval-Generation Synergy." arXiv: 2305.15294 [cs]. (Oct. 23, 2023), pre-published.

[43] Y. Gao. "Modular RAG and RAG Flow: Part II," Medium. (), [Online]. Available: https://medium.com/@yufan1602/modular-rag-and-rag-flow-part-ii-77b62bf8a5d3.

[44] Y. Gao, Y. Xiong, M. Wang, and H. Wang. "Modular RAG: Transforming RAG Systems into LEGO-like Reconfigurable Frameworks." arXiv: 2407.21059. (Jul. 26, 2024), pre-published.

[45] H. Yu, A. Gan, K. Zhang, S. Tong, Q. Liu, and Z. Liu, *Evaluation of retrieval-augmented generation: A survey*, 2024. arXiv: 2405.07437 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2405.07437.

[46] H. Lin, S. Zhan, J. Su, H. Zheng, and H. Wang. "IRSC: A Zero-shot Evaluation Benchmark for Information Retrieval through Semantic Comprehension in Retrieval-Augmented Generation Scenarios." arXiv: 2409.15763 [cs]. (Sep. 26, 2024), pre-published.

[47] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia. "ARES: An Automated Evaluation Framework for Retrieval-Augmented Generation Systems." arXiv: 2311.09476 [cs]. (Mar. 31, 2024), pre-published.

[48] A. N. Angelopoulos, S. Bates, C. Fannjiang, M. I. Jordan, and T. Zrnic. "Prediction-Powered Inference." arXiv: 2301.09633. (Nov. 9, 2023), pre-published.

[49] J. Fu, X. Qin, F. Yang, *et al.* "AutoRAG-HP: Automatic Online Hyper-Parameter Tuning for Retrieval-Augmented Generation." arXiv: 2406.19251 [cs]. (Jun. 27, 2024), pre-published.

[50] L. Zheng, W.-L. Chiang, Y. Sheng, *et al.* "Judging LLM-as-a-Judge with MT-Bench and Chatbot Arena." arXiv: 2306.05685. (Dec. 24, 2023), pre-published.

[51] X.-Y. Fu, M. T. R. Laskar, C. Chen, and S. B. Tn, "Are large language models reliable judges? a study on the factuality evaluation capabilities of LLMs," in *Proceedings of the Third Workshop on Natural Language Generation, Evaluation, and Metrics (GEM)*, S. Gehrmann, A. Wang, J. Sedoc, *et al.*, Eds., Singapore: Association for Computational Linguistics, Dec. 2023, pp. 310–316. [Online]. Available: https://aclanthology.org/2023.gem-1.25.

[52] I. Wu, S. Jayanthi, V. Viswanathan, *et al.*, "Synthetic Multimodal Question Generation," in *Findings of the Association for Computational Linguistics: EMNLP 2024*, Y. Al-Onaizan, M. Bansal, and Y.-N. Chen, Eds.,

Miami, Florida, USA: Association for Computational Linguistics, Nov. 2024, pp. 12 960–12 993.

[53] F. Petroni, A. Piktus, A. Fan, *et al.*, "KILT: A benchmark for knowledge intensive language tasks," in *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, K. Toutanova, A. Rumshisky, L. Zettlemoyer, *et al.*, Eds., Online: Association for Computational Linguistics, Jun. 2021, pp. 2523–2544. DOI: `10.18653/v1/2021.naacl-main.200`.

[54] T. Kwiatkowski, J. Palomaki, O. Redfield, *et al.*, "Natural questions: A benchmark for question answering research," *Transactions of the Association for Computational Linguistics*, vol. 7, pp. 453–466, Aug. 2019, ISSN: 2307-387X. DOI: `10.1162/tacl_a_00276`.

[55] Z. Yang, P. Qi, S. Zhang, *et al.*, "HotpotQA: A dataset for diverse, explainable multi-hop question answering," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, E. Riloff, D. Chiang, J. Hockenmaier, and J. Tsujii, Eds., Brussels, Belgium: Association for Computational Linguistics, 2018, pp. 2369–2380. DOI: `10.18653/v1/D18-1259`. [Online]. Available: `https://aclanthology.org/D18-1259`.

[56] J. Thorne, A. Vlachos, C. Christodoulopoulos, and A. Mittal, "FEVER: A large-scale dataset for fact extraction and VERification," in *Proceedings of the 2018 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, Volume 1 (Long Papers)*, M. Walker, H. Ji, and A. Stent, Eds., New Orleans, Louisiana: Association for Computational Linguistics, Jun. 2018, pp. 809–819. DOI: `10.18653/v1/N18-1074`. [Online]. Available: `https://aclanthology.org/N18-1074`.

[57] A. Wang, Y. Pruksachatkun, N. Nangia, *et al.*, *Superglue: A stickier benchmark for general-purpose language understanding systems*, 2020. arXiv: 1905.00537 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1905.00537.

[58] J. DeYoung, S. Jain, N. F. Rajani, *et al.*, "ERASER: A benchmark to evaluate rationalized NLP models," in *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, D. Jurafsky, J. Chai, N. Schluter, and J. Tetreault, Eds., Online: Association for Computational Linguistics, Jul. 2020, pp. 4443–4458. DOI: 10.18653/v1/2020.acl-main.408. [Online]. Available: https://aclanthology.org/2020.acl-main.408.

[59] S. Zhang, X. Liu, J. Liu, J. Gao, K. Duh, and B. V. Durme, *Record: Bridging the gap between human and machine commonsense reading comprehension*, 2018. arXiv: 1810.12885 [cs.CL]. [Online]. Available: https://arxiv.org/abs/1810.12885.

[60] J. Chen, H. Lin, X. Han, and L. Sun, *Benchmarking large language models in retrieval-augmented generation*, 2023. arXiv: 2309.01431 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2309.01431.

[61] Y. Tang and Y. Yang, *Multihop-rag: Benchmarking retrieval-augmented generation for multi-hop queries*, 2024. arXiv: 2401.15391 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2401.15391.

[62] Y. Lyu, Z. Li, S. Niu, *et al.*, *Crud-rag: A comprehensive chinese benchmark for retrieval-augmented generation of large language models*, 2024. arXiv: 2401.17043 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2401.17043.

[63] Z. Xu, Y. Li, R. Ding, *et al.*, *Let llms take on the latest challenges! a chinese dynamic question answering benchmark*, 2024. arXiv: 2402.19248 [cs.CL]. [Online]. Available: https://arxiv.org/abs/2402.19248.

[64] "Testset Generation for RAG - Ragas." (), [Online]. Available: `https://docs.ragas.io/en/latest/concepts/test_data_generation/rag/#query-types-in-rag`.

[65] "Synthetic Test Data generation | Ragas." (), [Online]. Available: `https://docs.ragas.io/en/v0.1.21/concepts/testset_generation.html`.

[66] N. Reimers and I. Gurevych, "Sentence-bert: Sentence embeddings using siamese bert-networks," in *Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing*, Association for Computational Linguistics, Nov. 2019. [Online]. Available: `https://arxiv.org/abs/1908.10084`.

[67] "Pretrained Models — Sentence Transformers documentation." (), [Online]. Available: `https://www.sbert.net/docs/sentence_transformer/pretrained_models.html#original-models`.

[68] *Microsoft/MSMARCO-Passage-Ranking*, Microsoft. [Online]. Available: `https://github.com/microsoft/MSMARCO-Passage-Ranking`.

[69] R. Aksitov, S. Miryoosefi, Z. Li, *et al.* "ReST meets ReAct: Self-Improvement for Multi-Step Reasoning LLM Agent." arXiv: `2312.10003`. (Dec. 15, 2023), pre-published.

[70] Y. Zhang. "A Practical Guide to RAG Pipeline Evaluation (part 1)," Medium. (), [Online]. Available: `https://blog.relari.ai/a-practical-guide-to-rag-pipeline-evaluation-part-1-27a472b09893`.

[71] Z. Xu, S. Jain, and M. Kankanhalli. "Hallucination is Inevitable: An Innate Limitation of Large Language Models." arXiv: `2401.11817 [cs]`. (Jan. 22, 2024), pre-published.

[72] F. Faggin, *Irriducibile.* Mondadori, 2022, ISBN: 978-88-357-2035-5.