### ALMA MATER STUDIORUM UNIVERSITÀ DI BOLOGNA

### DEPARTMENT OF COMPUTER SCIENCE AND ENGINEERING

#### ARTIFICIAL INTELLIGENCE

#### **MASTER THESIS**

in

Natural Language Processing

### ENHANCING THE SOFTWARE DEVELOPMENT LIFE CYCLE WITH CONVERSATIONAL AI: LLM-DRIVEN DOCUMENTATION Q&A AND AUTOMATED MODEL EVALUATION

CANDIDATE Shola Oshodi SUPERVISOR Prof. Paolo Torroni

CO-SUPERVISOR Dott. Ing. Francesco Saracco

Academic year 2023-2024 Session 5th to my loved ones, I owe you everything

## Contents

1	Intr	oductio	n	1
	1.1	Conter	nt Overview	2
2	Bac	kground	d	4
	2.1	Softwa	are Development Life cycle	4
	2.2	Genera	ative AI	6
	2.3	LLM .		7
	2.4	RAG .		9
	2.5	RepoA	Agent	10
3	Met	hodolog	3y	12
	3.1	Autom	nated Documentation Generation	12
	3.2	Summ	ary Generation	13
	3.3	Questi	on Answering System	15
		3.3.1	Models	17
		3.3.2	LLM Generation	21
		3.3.3	System Interface	22
	3.4	Evalua	ation Approach	22
		3.4.1	Dataset	24
		3.4.2	Classification Model Metrics	25
		3.4.3	Summary Evaluation Approach	25
		3.4.4	Q&A Evaluation Approach	27

4	Res	ults		33									
	4.1	Classi	fication Model	33									
	4.2	Summ	Summary Evaluation Results										
		4.2.1	Quantitative Evaluation Results	34									
		4.2.2	Qualitative Evaluation Results	36									
		4.2.3	Error Analysis	39									
	4.3	Q&A	Evaluation Results	40									
		4.3.1	Quantitative Evaluation Results	40									
		4.3.2	Qualitative Evaluation Results	41									
		4.3.3	Error Analysis	42									
5	Con	clusion		45									
		5.0.1	Future Work	46									
A	cknov	vledgem	ients	47									
Bi	bliog	raphy		48									
A	A Qualitative Evaluation of Summary: Prompt Example 54												
B	B Classification Model Prompt												
С	C Multi-Agent System Conversation 55												

# **List of Figures**

2.1	Description of two architectures of Retrieval Augmented Gen-	
	eration (RAG): Naive RAG and Advanced RAG. [14]	10
2.2	Overview of the RepoAgent framework, illustrating the Global	
	Structure Analysis, Documentation Generation, and Documen-	
	tation Update phases. [15]	11
3.1	Map-Reduce approach for multi-threads text summarisation.	14
3.2	Architecture of Question Answering System.	16
3.3	Self-Querying Architecture. The query is processed, struc-	
	tured with filters, and translated for efficient search in a vector	
	store [24]	21
3.4	Overview of the multi-agent system's iterative analysis and	
	evaluation process.	31

## **List of Tables**

3.1	Details on test case dimensions, sizes, and the number of files	25
4.1	Precision and Recall in classifying specific and general queries.	33
4.2	Summary generation execution time	34
4.3	ROUGE evaluation scores for GPT-3.5-Turbo and GPT-4.0-	
	Turbo summaries on Skillner Project. For each reference sum-	
	mary (A,B,C) the Table presents precision, recall and F1 scores	
	for ROUGE-1, ROUGE-2, ROUGE-L, with and without ac-	
	cess to the project's README.	35
4.4	ROUGE evaluation scores for GPT-3.5-Turbo and GPT-4.0-	
	Turbo summaries on Repoagent Project. For each reference	
	summary (A,B,C) the Table presents precision, recall and F1	
	scores for ROUGE-1, ROUGE-2, ROUGE-L, with and with-	
	out access to the project's README.	35
4.5	Qualitative Evaluation of GPT Models for Repoagent Sum-	
	marization assessing consistency(1-5), coherence(1-5), relevance(	[1-
	5) and fluency (1-3)	37
4.6	Qualitative Evaluation of GPT Models for Skillner Summa-	
	rization assessing consistency(1-5), coherence(1-5), relevance(1-	
	5) and fluency (1-3)	37
4.7	Question answering quantitative evaluation	40
4.8	Examples for error analysis of general questions and answers.	43
4.9	Examples for error analysis of specific questions and answers .	44

#### Abstract

In software development, navigating code documentation is a time-consuming and cognitively demanding task. This research, investigates the potential of Large Language Models (LLMs) to improve documentation consultation, aiming to reduce developer efforts while increasing their understanding. The study introduces a novel system that integrates an existing LLM-based documentation generator with an advanced question-answering system. This system, composed of different Retrieval-Augmented Generation (RAG) models, is designed to understand the contextual needs of user queries and intelligently route them to the most appropriate RAG within the proposed architecture. This approach proved to effectively handle both fine-grained and coarsegrained questions, obtaining high Context Precision (0.955) and Answer Relevancy (0.900) according to the RAGAs metrics. The research also investigates automated project summarisation and automated evaluation strategies for natural language systems, proposing procedures to reduce human effort while ensuring reliable results.

Overall, this study highlights the potential of LLMs and Generative AI to streamline software development workflows, reducing time and cost associated with code understanding and LLM evaluation. It contributes to the growing field of AI-driven software engineering tools, laying a foundation for future research and applications in documentation management and intelligent systems design. This thesis is the result of practical research conducted in collaboration with DATA Reply.

## **Chapter 1**

### Introduction

A significant portion of the time spent by software developers (from 50 to 70%) is dedicated to code comprehension, making it one of the most timeconsuming tasks of their daily work [1][2]. This, in turn, has a consistent impact on software development costs, as developers must spend time understanding the code before being able to implement changes.

For these reasons, effective communication of knowledge is a key activity within the Software Development Life Cycle (SDLC).

One of the primary means to explain and instruct users about systems or code is documentation, which however, often presents various usability challenges in real-world scenarios: the resulting difficulties in navigating and locating relevant information can lead to significant cognitive overload for software engineers, impacting both efficiency and productivity.

To address these challenges, this thesis examined the potential of Large Language Models (LLMs) in enhancing the effectiveness of documentation consultation through an interactive question-answering system composed of multiple Retrieval-Augmented Generation (RAG) models. The strength of the developed system lies in its intelligent routing mechanism, which directs queries to the most appropriate RAG model based on their specificity and contextual requirements. The designed project starts from an existing code-level documentation generator used to obtain an informative knowledge base about the software functionalities.

Additionally, this research explored tasks as multi-document summarisation and natural language generation evaluation proposing automated strategies to minimise human effort.

The work discussed contributes to the domain of software development by reducing the cognitive load required for effective documentation understanding. This has direct implications on time and cost, while also indirectly improving the quality of code. By providing developers with a deep understanding of the project, it supports more informed decisions and reduces errors caused by misunderstanding or unclear information.

The entire process was driven by the following research questions:

- 1. **Q1**: How can LLMs be applied to improve the understanding of documentation?
- 2. Q2: How can Conversational AI systems be designed to handle different types of queries with varying complexity and information requirements?
- 3. Q3: Is it possible to reduce human dependency on evaluation while providing useful insights?
- 4. **Q4**: What are the key differences in performance and output quality between different LLMs on specific tasks?

### **1.1 Content Overview**

The thesis is structured as follows:

• Chapter 1 - Introduction

- Chapter 2 Background: Introduces key concepts and provides foundational knowledge necessary for project understanding.
- Chapter 3 Methodology: Describes the developed system, focusing on the adopted methodologies and the criteria defined for its evaluation.
- Chapter 4 Results: Presents and analyse the obtained results.
- Chapter 5 Conclusion: Summarises the key contributions and outlines potential future research directions.

## **Chapter 2**

### Background

To fully understand the significance and scope of this research, it is essential to first explore the underlying concepts and theories that form the basis of this study. This chapter provides a comprehensive overview of the key principles.

### 2.1 Software Development Life cycle

In the fast evolution of the field of software development, it became necessary to establish a standardized structured process that guided the entire development through its phases. The Software Development Life Cycle (SDLC) defines different stages ensuring better organization and higher code quality[3].

It usually consists in various phases where the principal are:

- 1. Planning: Project goals, requirements and timelines are defined.
- Feasibility Analysis: Technical requirements, cost, and risk are assessed in order to measure the project's feasibility and its technical and financial limitations.
- 3. System Design: Focuses on creating software architecture and design.
- 4. **Implementation**: This phase focuses on development, where functional applications are implemented based on the project's design.

- 5. **Testing identification and bug fixing**: Comprises both automated and manual testing methods to identify bugs and vulnerabilities. If any bug is present, it is addressed during this phase.
- 6. **Deployment**: In this phase, the software is implemented in a real environment, configured and made available to end users.
- Maintenance: Once the software is developed, it should be continuously updated to fix bugs and improve functionalities. Its main goal is to ensure that the system remains stable, secure, and performs well over time.

These phases can interact with each other in various ways, forming different development models. The most frequently used are Waterfall, Iterative, Agile, V-Model and DevOps.

- Waterfall: Each phase begins when the previous one is completed and error-free. It is typically used in small projects.
- Iterative: It starts with the implementation of just a part of the requirements and proceeds iteratively to complete the project. Each implementation phase is combined with testing and feedback.
- Agile: This model focuses on strong collaboration, continuous feedback and frequent updates to satisfy new requirements. Development is divided into small parts called "sprint", which are implemented incrementally and iteratively.
- V-Model: Each stage, except for the implementation phase, consists of two parts: validation and verification. Each development stage is linked to its verification phase.
- **DevOps**: It bridges the traditional gap between development and operations by managing the application life cycle through a seamless and

continuous flow of testing, development, and deployment activities. It focuses on streamlining processes through a high degree of automation.

While each model is suited to specific contexts, certain general requirements remain fundamental across all models, highlighting the benefits of documentation management system presented in this thesis:

- Clear communication among team members;
- Traceability of design choices;
- Deep understanding of the codebase to facilitate testing, debugging and maintainability.

### 2.2 Generative AI

In the past, it was commonly believed that innovation and creativity —writing, coding, designing, or drawing— were the exclusive domain of humans. However, recent advancements have demonstrated that Artificial Intelligence can play an active role in innovation, contributing with new ideas and solutions by combining and transforming existing knowledge. This emerging field is known as **Generative AI**.

Specifically, Generative AI refers to computational techniques capable of generating new, meaningful content (e.g. text, image, or audio) based on training data [4]. Mathematically, this is achieved through generative models that aim to infer statistical data distributions: these models can estimate the distribution of the label Y or the joint distribution of observing the label Y given the data X. This ability allows the models to generate new synthetic samples, either by creating new pairs of observations (X, Y), or by producing new observations X based on a target value Y [5].

The potential of Generative AI has led to increasing interest in applying it

across various contexts and tasks. In relation to this thesis project, novel approaches include its integration into the field of Software Development to minimise team members' workloads [6] [7]. Specifically, this research explores the effectiveness of Generative AI in this domain, focusing on its natural language interaction capabilities.

### 2.3 LLM

Large Language Models (LLMs) are advanced models trained on vast amounts of data, designed to generate and process text and solve natural languagerelated tasks. These capabilities are acquired by learning the statistical relationships and patterns that occur in the data during the self-supervised and semi-supervised training process.

This thesis examines one of the most widely used models in industry and research: **GPT (Generative Pre-trained Transformer)** by OpenAI.

GPT is a deep learning model pre-trained on large corpora of text data, which can be fine-tuned to perform better on specific tasks.

The architecture of GPT is based on the Transformer model [8], which employs a self-attention mechanism to consider the context of the entire input sequence while generating the new words.

There are different versions of GPT, each built upon advancements in pretraining, fine-tuning, and architectural design. Specifically, this work focuses on GPT-3.5-Turbo and GPT-4.0-Turbo.

#### GPT-3.5-Turbo

Launched in November 2022, it has been specifically optimised for chat-based interactions, making it particularly effective in conversational contexts. Its design enables comprehension and generation of both natural language and code.

#### GPT-4.0-Turbo

GPT-4.0-Turbo is a large multi-modal model capable of processing both text and images. It achieves one of the highest level of accuracy in solving complex problems among OpenAI's models, thanks to its broader general knowledge, extended context window, and more advanced reasoning capabilities. The differences between GPT-4.0-Turbo and GPT-3.5-turbo become particularly evident in complex task scenarios, where GPT-4.0-Turbo outperforms GPT-3.5-Turbo. [9]

Both models have been enhanced by OpenAI through a specific training process involving Reinforcement Learning from Human Feedback (**RLHF**) [10]. This approach begins with pre-training the model on a large corpus of text in a self-supervised manner and proceeds with human evaluators ranking generated responses to create a reward model. Finally, the model is further optimized using Proximal Policy Optimization (**PPO**), a reinforcement learning algorithm that fine-tunes the model by iteratively adjusting its parameters to maximize cumulative rewards [11]. OpenAI has not provided extensive details about the training process of GPT-4.0-Turbo models, but it has mentioned the introduction of an additional reward model called Rule-Based-Reward Model (**RBRM**) [12], designed to enforce alignment of generation with predefined rules.

Although the exact number of parameters is not publicly available, the GPT-4.0-Turbo family is estimated to be about 10 times larger than GPT-3.5-Turbo, with about 1 trillion parameters compared to the 175 billion used in the previous version [13].

#### Limitations

GPT models are subject to certain limitations that can be summarised as follows:

• They tend to generate plausible-sounding but incorrect answers.

- Models are sensitive to how questions are phrased or rephrased, and may produce different answers when the same question is asked multiple times with slight variations.
- Models often handle ambiguous questions poorly, attempting to provide an answer rather than asking for clarification.
- The models tend to be verbose and redundant in their responses.
- Despite the implementation of a moderation system, the models can still produce unsafe or inappropriate responses.

### 2.4 RAG

As underlined in the previous section, one of the key limitations of LLMs is their tendency to hallucinate, which leads to inaccurate, outdated, and not verified responses. Additionally, the difficulty in tracing the source of the model's answer makes it even harder to discern whether the generated information is based on reliable data. A promising solution is **Retrieval-Augmented Generation (RAG)**[14], which approach can be summarised in three main phases:

- **Indexing**: The input text is divided into chunks, encoded into vector representations, and stored in a vector database.
- **Retrieval**: The user's query is also embedded as a vector, and its similarity to stored document chunks is computed. The system retrieves the most relevant chunks based on similarity scores.
- Generation: The provided query, along with the selected chunks, is integrated into the prompt and passed to the LLM model for response generation.

RAG model, to be considered effective, should accurately recognize relevant information while filtering out irrelevant data to enhance the generation



Figure 2.1: Description of two architectures of Retrieval Augmented Generation (RAG): Naive RAG and Advanced RAG. [14]

process. To further improve performance, **Advanced RAG** methods have been introduced. These employ a refined indexing strategy and the incorporation of pre-retrieval and post-retrieval steps, such as re-ranking, query transformation or query expansion. A visual comparison of these two models is provided in Figure 2.1.

### 2.5 RepoAgent

One of the primary objectives of this thesis, as extensively highlighted in the previous chapter, was to develop a highly automated tool to explain documentation. To achieve this, the open-source framework **Repoagent** [15] was utilised for generating documentation. Repoagent enables the generation, maintenance, and automation of markdown documents, synchronizing them with project repositories. To generate detailed code-level documentation, the project pipeline begins with a global analysis of the Python files. Their code structure is parsed, function and classes extracted, and reference relationship mapped. This information, along with Abstract Syntax Tree, metadata, file paths, and past documentation, is provided to the LLM. The resulting documentation includes code details such as functionality, parameters, code descriptions, notes, and examples. Finally, the system automatically tracks repository updates triggering documentation generation upon each commit. A more in-depth description of the entire process is provided in Figure 2.2.



Figure 2.2: Overview of the RepoAgent framework, illustrating the Global Structure Analysis, Documentation Generation, and Documentation Update phases. [15]

The integration of this advanced tool with the developed system resulted in a fully automated and highly customizable pipeline.

## **Chapter 3**

## Methodology

This chapter outlines the methodological approach followed in this thesis and introduces the various automated evaluation methods implemented to assess system performance. The goal of these evaluations is to reduce human dependency, reduce time and effort, and ensure reliable and consistent analysis results.

### **3.1** Automated Documentation Generation

As highlighted in the Background chapter, documentation generation was made possible through Repoagent, a public repository that managed the generation process and the integration with Github for automated updates. Considering the final objective of providing a clear understanding of the project, leveraging this existing framework, several minor refinements and bug fixes were made.

In particular, the generated documentation lacked a general overview file that briefly explained the project's main purpose: the documentation was highly technical and code-centered, limiting users seeking a quick, general understanding of the project. To address this, a **summary** documentation file generation was implemented. Additionally, to prevent unnecessary computational overhead, the documentation update process was modified, allowing users to disable automatic updates with each commit if desired.

### 3.2 Summary Generation

The creation of the summary proved to be fundamental in supporting the questionanswering component of the system, as will be discussed in detail in the General Model section. To ensure a good alignment with the project information, the summary generated relied directly on the Markdown files produced by Repoagent. Given the necessity of processing a large number of files, we considered summarisation techniques that could balance both efficiency and accuracy. Among the methods reviewed in the literature, we chose **MapReduce** [16] [17], a popular programming model that provides scalability for handling large text collections in summarisation tasks. This approach effectively addresses the challenges of multi-document summarisation, obtaining meaningful results and computational efficiency even with large number of documents.

In this approach, each document is first summarised individually in the Map step, where the key points and essential information are extracted from each file, then, in the Reduce step, the individual summaries are combined into an overall structured summary. This process ensures that all the important information from all the documents is considered and synthesized into a unified text.

Initially, we implemented the process using LangChain's MapReduce chain, but this didn't take full advantage of the scalability properties of this strategy, resulting in a computationally intensive process. To address this, we manually implement the summarisation by parallelizing the Map step across multiple threads.



Figure 3.1: Map-Reduce approach for multi-threads text summarisation.

The resulting summarisation workflow, as shown in Figure 3.1, is defined as follows:

- Each document was summarised using LangChain's Stuff chain and executed simultaneously by a pool of threads.
- The individual summaries were then combined using a custom prompt designed to merge them into an overall summary. The model was instructed to write the summary as a project description, providing a general overview without focusing on implementation details. Not all the methods had to be mentioned, only the most important ones. The model was asked to avoid naming specific features and going into deep technical detail. The generated summary had to be 15-20 lines long.

A key factor that significantly influenced both the performance and accuracy of the models was the **chunk size** used in processing documents. Since the optimal chunk size can vary depending on the nature and structure of the documents being considered, an additional feature was implemented to allow users to manually test and select the most suitable chunk size for their specific use case. It is important to note that smaller chunks size can considerably slow down the process: in the considered study the optimal size was determined to be 5000 tokens.

### **3.3 Question Answering System**

In order to investigate the potential of LLMs in enhancing code understandability, a newly created question-answering system was integrated into the existing Repoagent project. The primary objective of the system was to leverage large pre-trained language models to provide near-instantaneous, precise, and fact-verified answers. While LLMs have demonstrated notable abilities in conversational and question-answering tasks [18][19], they have been observed to struggle with expanding or updating their memory, providing clear insights into their predictions, and being prone to generating "hallucinations".[20] To address this issue, a Retrieval Augmented Generation approach was adopted, combining the strengths of information retrieval and language generation, it enabled the LLM to directly access relevant data information from a specific knowledge base. By retrieving pertinent information, the likelihood of generating hallucinated or incorrect statements is reduced. The use of a knowledge base also helps the LLM better understand queries and domain-specific requests. The RAG architectures were implemented using the LangChain library, whose modularity allows for the adjustment of implementation details (e.g. switching the LLM model) without modifying the entire codebase: it ensures the system remains easily customizable and adaptable to different requirements and use cases.

As illustrated in Figure 3.2 the question answering complete pipeline comprises three different models (their detailed explanation will be provided in



Figure 3.2: Architecture of Question Answering System.

the subsequent section). The reason behind this structure arises from the limitations of LLMs in processing long-context text, in particular, retrieval-based techniques have showed significant advantages for answering short questions, with minimal effect on the comprehension of longer contexts [21]. This aspect posed a challenge in the initial version of the system where a single model was used: the model could answer very specific questions, but lacked an understanding of the general functionalities and purpose of the analysed project. This limitation significantly impacted the system's potential, making it less useful for users who needed concise project information rather than detailed implementation specifics.

The proposed solution aims to identify questions that require a larger context to classify and redirect them to a **General** LLM model, which uses the generated project summary as knowledge base. Specific question, on the other hand, are redirected to the **Specific** RAG Model, which considers all the documentation files. In order to ensure an effective navigation of the documentation and diminishing the developers' struggles, a smooth and consistent conversation is crucial. To this extent, the history of past messages has to be taken into account. In this case, the partitioning of the pipeline in three different models introduced an additional level of complexity in handling and synchronizing the conversation history, which made necessary the introduction of a shared history, updated by both the General and Specific models after each interaction.

As for human beings, LLMs also have their own mental frames, processing and responding to inputs based on their training and inherent biases. This can sometimes lead to differences between human and LLMs interpretations of the same prompt. Since it can be challenging for humans to properly assess and refine the clarity of their own question, in our study the Rephrase and Respond strategy has been employed [22]. This method involves LLMs to rephrase sentences, while integrating all the historical information necessary to contextualise the query into a single prompt.

The contextualised questions are then passed to the **Classifier** model, which determines whether to assign the task to the Specific or the General model.

#### **3.3.1 Models**

Both the Specific and General models have been designed to process Markdown documentation files; therefore, the same pre-processing steps were applied to their knowledge base texts. Before feeding the text into the LLM, it was necessary to divide it into manageable chunks. To this end, LangChain's MarkdownTextSplitter was employed to segment the text based on header tags, and the RecursiveTextSplitter was then utilised to further break down the remaining text into smaller sections. Through experimentation, the optimal chunk size was determined to be 250 tokens with an overlap of 30 tokens. The overlaps helped capture context more efficiently across consecutive chunks.

#### Classifier

This model was introduced to differentiate between broad questions about the project's purpose and functionalities (general) and questions about code functions and implementation details (specific). To build a reliable classifier, we had to leverage some prompt engineering techniques to maximize the classification accuracy. Initially, a simple prompt describing the characteristics of general and specific question was used: this approach turned out to be largely inaccurate, as the model struggled to consistently assign the correct class.

To enhance performance, the few-shot learning strategy was explored, a prompting techniques in which the model is provided with several examples to help its understanding of the task. The efficiency of few-shot learning derives from its imitation of a common natural language pattern: the repetition of similar structures with varying parameters. By providing a few examples, GPT models-in contrast to earlier models-can successfully recognise these patterns and generalise to new input, even when examples are presented in isolated contexts. [23]

For the case under consideration, initially a series of examples was defined for both general and specific questions, each example being associated with its respective class label. This led to better performance, but also revealed that misclassification affects conversational models in different ways. When a specific question was misclassified as general, the model relied on high-level summaries, resulting in incomplete or inaccurate responses. However, when a general question was misclassified as specific, the model could still sometimes deliver a reasonable response. To emphasize this disparity, we refined the prompt (see Appendix B ) including exclusively examples of general questions, instructing the model to classify any query that did not match these examples as specific. This approach encouraged the model to be more cautious, classifying uncertain cases as specific, reducing the risk of critical misclassification.

#### **General Model**

As explained in the Background chapter, the acronym RAG stands for Retrieval, Augmentation and Generation, which are the three key components of this technique. Generation is highly dependent on the other phases, particularly Retrieval, which significantly impacts the overall performance of the model.

In our system, the General Model, designed to answer user queries by consulting a generated summary, relies on a **VectorStore Retriever**, which transforms each chunk of text into embeddings (numerical vector representations that encode the semantic meaning of the text). In this high-dimensional vector space, semantically similar texts are positioned closer to one another, enabling more effective retrieval based on meaning rather than relying on simple keyword matching.

When a user submits a query, this is converted into an embedding, and a similarity search is conducted between the query and the stored text embeddings by evaluating their proximity in the vector space. This approach ensures that the most relevant documents are retrieved, directly influencing the accuracy and relevance of the final response.

The decision to use this specific retriever arises from its scalability and efficiency, making it highly suitable for real-time applications like the developed chat system, where fast, accurate retrieval is crucial. Once the relevant results are retrieved, they are combined with the user's query and passed to the LLM which generates the final answer.

#### **Specific Model**

The grater the model is able to explain the documentation to the programmers, the easier it will be for them the code understanding. In order to facilitate the processing of complex queries and the refinement of results based on certain criteria, for the specific model, it was necessary to implement a more advanced retrieval method.

For example, in the context of the project, it is likely that users will request information also by specifying a file name in order to gain deeper insights or explanations. Therefore, the file source has to be considered as an important metadata information: unlike General Model —which relies solely on semantic similarity for document retrieval— this model requires the ability to apply filters based on metadata.

To achieve this, the Langchains's SelfQueryRetriever was implemented: this retriever performs searches on the vector store applying dynamic filters considering metadata. Specifically, given a natural language query, the retriever uses a query-constructing LLM to generate both a structured query and relevant metadata filters. These are then passed to a query translator to convert them to the correct format for query execution. The translator adapts the output to fit the Chroma DB format [24]. The entire process is summarised in Figure 3.3.

The SelfQueryRetriever enhances search accuracy by giving importance both to context and metadata. Additionally, it makes the search process more efficient both in computation and time, as the number of chunks on which the similarity function is computed is reduced through the application of filters beforehand.



Figure 3.3: Self-Querying Architecture. The query is processed, structured with filters, and translated for efficient search in a vector store [24]

#### 3.3.2 LLM Generation

When selecting the appropriate large language model for our system, we considered various factors such as performance, accuracy, versatility, and scalability. In particular, while the OpenAI models exhibited state-of-the-art capabilities in understanding and emulating human natural language, they also have demonstrated the ability to reach or even surpass human performance levels in a variety of language tasks. [25]

For these reasons, in order to ensure a comprehensive evaluation, two distinct models from OpenAI were utilised: GPT-4.0-Turbo and GPT-3.5-Turbo.

#### GPT-4.0-Turbo

GPT-4.0-Turbo, one of the OpenAI's most advanced models, was selected to guarantee that the analysis included an high-performing system capable of providing a comprehensive understanding of LLMs' potential capabilities in the field. From a practical perspective, GPT-4.0-Turbo's near-human response times make it particularly well-suited for conversational question-answering. Furthermore, its deep contextual understanding potentially offers significant advantages for both advanced reasoning and precise information retrieval.

#### GPT-3.5-Turbo

In our evaluation, we recognized the need for a balanced use of the LLMs. For secondary tasks (such as project summarisation) we were interested in exploring less powerful (but more cost-effective) models like GPT-3.5-Turbo. This decision allowed us to assess whether reserving the robust capabilities of GPT-4.0-Turbo exclusively for the question-answering system would be beneficial.

This explorative approach was driven by two main considerations. First, since question-answering represents the core functionality of our system, we were more interested in prioritizing its robustness and reliability. Second, as one of the primary goals of this dissertation was to explore the full potential of LLMs in enhancing documentation understanding, we required for the task a model capable of demonstrating the highest level of comprehension and analytical ability.

If GPT-3.5-Turbo delivers results comparable to GPT-4.0-Turbo for secondary tasks, then we will be able to optimize resource consumption and cost efficiency without compromising the performance of our primary objective.

#### **3.3.3 System Interface**

The developed chat was designed as a tool for programmers to better understand code projects. To facilitate this, we opted for a command line interface (CLI), which enables fast and practical interaction with no need to switch to a different tab or application. It is seamlessly integrated with the CLI of the Repoagent project, allowing for centralised shell commands.

### **3.4 Evaluation Approach**

Evaluation is a crucial part of every research task, especially in complex systems. It is essential to carefully examine each component and establish the most relevant criteria to accurately reflect the system's overall performance. In this thesis project, to understand the quality of the developed solution, we focus on evaluating three key elements to assess the system:

- The accuracy of the classification model.
- The quality of the generated summary.
- The **performance** of both the specific and general models in generating answers.

While the evaluation of classification tasks is relatively straightforward, assessing the quality of generated natural language poses several challenges, making it a complex and ongoing research area as shown in [26]. This difficulty arises from the presence of multiple valid responses, with their correctness depending on factors such as context, clarity, reliability and fluency. As can be easily understood, these challenges also extend to the summary evaluation, where in addition, there is a need to balance the completeness of the information with its conciseness. [27]

The most common approaches in natural language generation tasks evaluation involve the use of automatic metrics (e.g. ROUGE, METEOR, BLEU) which offer objective, quick, and easily comparable evaluation methods based on lexical matching with a gold standard human-provided text. [28] The metrics' dependence on N-gram overlap often leads to poor alignment with human judgments, as they fail to accurately score correct sentences that differ syntactically or semantically from the reference text [29] For this reason, they should not be considered the sole indicators of quality, alongside with this quantitative evaluation it is a good practice to include a qualitative human assessment to gain more reliable insights into system performance. Unfortunately, human evaluation presents additional challenges: it requires a technically competent group of evaluators who can invest time and effort in the assessment process. This is not always feasible in practice, which is problematic, since the quality of the evaluation depends heavily on their expertise and the time they can dedicate to the task.

One of the research interests of this thesis was also to propose a methodology to address this problem while aligning with the principle of automating processes. To mitigate the bottleneck caused by human evaluation, the use of LLMs was explored as a replacement for human assessments to streamline the process and reduce manual effort.

The following subsections of this chapter present a more detailed overview of the evaluations of each component.

#### 3.4.1 Dataset

In order to evaluate the system, it was necessary to select projects for generating the documentation and test the system's summarisation and questionanswering capabilities. Two projects of different sizes were chosen to assess the system's performance across varying levels of complexity. The first project, Skillner [30], focuses on the automated extraction of soft skills from text, and we used it as a test case for a smaller-scale project.

For the second case, the documentation generated on this thesis project itself was used. The objectives were to test performance on a larger project and to leverage the extensive knowledge we had about the system for a more accurate analysis of its responses. Additionally, using the project's own documentation as a test case contributed to enhance the clarity of the result discussion. Table 3.1 provides details on the dataset's dimensions.

Project	Size	Number of Files
Skillner	22.98MB	169
Repoagent + Developed System	0.98GB	30,078

Table 3.1: Details on test case dimensions, sizes, and the number of files

#### **3.4.2** Classification Model Metrics

To evaluate the performance of the classification model, the following standard metrics were used:

- **Precision**: Indicates the proportion of true positives among all positive predictions, highlighting the model's ability to avoid false positives.
- **Recall**: Reflects the proportion of true positives correctly identified out of all actual positives, emphasizes the model's ability to capture relevant instances and avoid false negatives.

#### 3.4.3 Summary Evaluation Approach

#### **Quantitative Evaluation**

The assessment of the quality of an automated summary is typically performed using ROUGE metrics [31], which represent the standard *de facto* approach. Consequently, these metrics were used in order to obtain a unified measurement. There exist different versions of ROUGE, each designed to evaluate the overlap of N-grams between the generated and reference summaries, with N as a variable parameter.

The general formula is provided below, where N represents the length of the N-gram:

$$ROUGE - N = \frac{\sum_{S \in \{\text{Gold Standard Summaries}\}} \sum_{\text{gram}_n \in S} \text{Count}_{\text{match}}(\text{gram}_n)}{\sum_{S \in \{\text{Gold Standard Summaries}\}} \sum_{\text{gram}_n \in S} \text{Count}(\text{gram}_n)}$$

In particular, we used ROUGE-1, ROUGE-2 and ROUGE-L, with ROUGE-L being a specific version that measures the longest common subsequence between the generated and reference summaries. While ROUGE 1 and 2 are useful to check simple overlap, ROUGE-L is needed to verify whether the generated summary follows the reference summary structure. These utility of these scores is mainly based on their ability to provide a statistically informative comparison between the generated summaries and the gold standard. In fact, several research (e.g. [32][33]) have shown that the ROUGE score may have low alignment with human evaluation, and in some cases, even poor summaries can achieve high ROUGE scores. For this reason, it should be considered as an indicative, partially reliable measure.

#### **Qualitative Evaluation**

Human assessment plays a crucial role in evaluating the quality of generated summaries since it takes into account factors that automated metrics often miss. These factors can vary depending on the specific use case but usually include aspects such as relevance, clarity, fluency, and overall coherency. If competent external evaluators are not available, the risk of developers assessing their own project is the potential of providing biased feedback. To address this gap and obtain informative opinions without involving additional human resources, we drew inspiration from G-Eval, a study conducted by the Microsoft Cognitive Services Research team [25] which uses GPT-4 as evaluator. According to the cited approach, GPT-4 successfully outperformed other summarisation metrics, demonstrating the best alignment with human evaluations. The LLM was asked to score each of the following aspects:

- Coherence (1-5): Measures the logical flow and the quality of integration of sentences within the summary.
- Consistency (1-5): Measures the factual alignment with the source documents. Summaries with hallucinated or unsupported facts were penalized.
- **Relevance (1-5)**: Measures how accurate is the selection of important content from the source. Summaries with redundancies or unnecessary

information were penalized.

• Fluency (1-3): Measures the quality of the summary in terms of grammar, spelling, punctuation, word choice, and sentence structure.

The two main factors that contributed to G-Eval's improved evaluation method were the score normalization and the Chain of Thought [34] technique. The score normalization remained unchanged in our implementation while the prompting approach was refined by introducing an intermediate step focused on the extraction of key points before the score assignment.

The key points were automatically extracted via LLM and then provided to the evaluators along with the generated summaries.

This approach preserved the structure of the Chain-of-Thought reasoning while ensuring that the process remained within the LLM's token limits.

This method also improved the assessment of consistency and relevance by verifying that the extracted key points were accurately reflected in the text and that all summaries were factually aligned with these key points. This reduced the chance of evaluators focusing on unnecessary details.

Additionally, to enhance clarity in the scoring, evaluators were asked to include a brief explanation of the motivation behind each evaluation. A prompt example of our approach is reported in Appendix A

#### 3.4.4 Q&A Evaluation Approach

The evaluation of RAG models presents the same challenges traditionally associated with natural language generation models that were mentioned in the previous section. Moreover, they introduce further complexities that require additional evaluation criteria due to their combination of generation and external information retrieval. Given the aim of this study of exploring low-human-intervention evaluation methods, the principal challenges in evaluating the question-answering systems pertain to the effort required to:

- Understand every detail of the project.
- Generate suitable test questions.
- Define the ground truth responses for quantitative evaluation.
- Provide feedback on the given responses.

As can be easily understood, this process is highly time-consuming.

#### **Quantitative Evaluation**

Considering the factors involved, after extensive research, we decided to use RAGAs (Retrieval Augmented Generation Assessment) a recently implemented reference-free framework specifically designed to evaluate RAG pipelines [es2023RAGAsautomatedevaluationretrieval]. RAGAs evaluates all the key dimensions of RAG: the ability to retrieve the correct information and passages, the LLM's capacity to process this information, and the quality of the generated response. Additionally, it has shown close alignment with human evaluation and does not require the formulation of a ground truth answer.

RAGAs evaluation focuses on the following aspects:

- Faithfulness: Measures how well it is possible to infer the claims made in the answer from the provided context.
- Answer Relevance: Measures how directly the generated answer addresses the provided question.
- Context Precision: Measures how much relevant information is retrieved.

- Context Recall: Measures how much of the retrieved information is useful.
- Answer Correctness: Evaluates the factual accuracy of the retrieved answer.

The other crucial aspect for a reliable evaluation is the selection of test questions, as their quality significantly impact the accuracy of the assessment. This task can be particularly challenging as it requires expertise, domain knowledge, and effort. Poorly formulated questions can lead to misleading scores or feedback. The RAGAs framework introduces a method for generating synthetic test sets, which we implemented to build a question-answer dataset.

This approach offered two key advantages:

- Automated Pipeline Creation: It enabled the creation of a fully automated pipeline, streamlining the dataset generation process. This allows for greater scalability compared to manual assessment.
- Generation of Queries with Different Levels of Complexity: It allowed for the in-depth examination of various query types to properly evaluate the system's ability to respond. In particular, it tested the system's ability to handle simple queries, reasoning-based queries, and multi-context queries.

Utilising RAGAs a set of 70 specific documentation-related questions was generated, ensuring an equal distribution across the three query types described above. In addition, another set of 70 specific questions was manually defined to be compared with the automatically generated one. While the system's performance remained stable across both sets, the RAGAs queries provided unbiased and more informative examples. As a result, they were selected to be used for the evaluation. This decision was also motivated by the desire to

propose an approach for generating test sets in the absence of competent technical support.

The definition of general question was less time consuming since it required multiple formulation of the same type of question (e.g. inquiries about project's purpose). For this reason, and to ensure the questions would focus on the project from a high-level perspective, 30 general questions were manually defined.

#### **Qualitative Evaluation**

Apart from this quantitative evaluation, we were interested in providing a qualitative evaluation by domain experts while maintaining an automated approach that ensured low human involvement. Our idea was inspired by ChatEval [35] where a **multi-agent system** was used to evaluate answers generated by different models and select the one with the best performance. This inspired us to explore the use of specific role agents for conversational evaluation of answers: this approach allowed us to obtain detailed feedback and identify potential areas for improvement.

As for ChatEval, our implementation used the AgenteVerse [36] library to facilitate the definition of agents and their interactions; we defined the role and the task that each agent was required to do. Every agent had different expertise in order to catch different aspects of the quality of the model responses, in particular, we decided to consider the opinion of 4 different agents:

- A **software engineer** with expertise in Natural Language Processing and Machine Learning to improve the quality of the answers generated by the system.
- A **data scientist** with experience in information retrieval and database management to assess and improve the efficiency of the retrieved information.

- A **technical writer** with a background in software development to ensure the clarity, accuracy, and understandability of the system's responses.
- A **project manager** with a deep understanding of software development processes to provide insights from a high-level perspective on the usefulness of the retrieved information.



Figure 3.4: Overview of the multi-agent system's iterative analysis and evaluation process.

The communication strategy adopted (summarised in Figure 3.4), began with the drafting of a comprehensive analysis by the software engineer, who examined the responses generated by the system, highlighting their strengths, weaknesses, and areas for potential improvements. The other agents then acted as reviewers, providing their critical and technical opinions on the software engineer's analysis (openly expressing any disagreements when necessary). The analysis was refined iteratively until all agents reached a consensus on the observations made. Once a compromise was achieved, each agent was asked to score the responses on a scale from 1 to 10 evaluating their: Consistency, Coherence, Fluency and Relevance.

To enhance the interpretability of the scores provided by the multi-agent system, each agent was also required to propose brief suggestions for improving the question-answering model.

In order to verify the robustness of this evaluation and ensure that the multiagent system analysis was not biased by reliance on the same GPT model, an additional experiment was conducted. Specifically, the same multi-agent system approach was applied with the Meta LLaMA2-70B model. The experiment's grading and feedback aligned with those from the GPT-4.0-Turbo model, confirming the reliability of our approach.

The possibility of consulting the entire conversation of the agents, along with their scores and suggestions, made it possible to highlight aspects of the model that did not emerge during the quantitative evaluation. In scenarios where a competent technical team is not available, without the multi-agent system all these insights would have been lost.

## **Chapter 4**

### Results

In this chapter, an extensive discussion of the obtained results is provided. For clarity, the organization of the chapter will follow a structure similar to the previous one.

### 4.1 Classification Model

Metric	Specific	General	Overall
Precision	0.883	0.913	0.898
Recall	0.971	0.700	0.836

Table 4.1: Precision and Recall in classifying specific and general queries.

Observing Table 4.1 is evident that the model achieved satisfactory results in classifying the questions. In particular, it exhibits optimal results in specific questions both for precision and recall. The lower recall for the general question is a partially expected behaviour since, as previously mentioned, while training through few-shot learning the model was asked to classify as specific any question not matching the provided examples. As the recall highlights, the model misses some general question examples, however, since it is preferable to misclassify a general question as specific rather than the contrary, this trade-off is acceptable.

### 4.2 Summary Evaluation Results

In consideration of the project's objective of reducing developers' time expenditure, one of the aspects considered was execution speed in generating the summarisation. Table 4.2 reports the times recorded during development. For this evaluation the larger project test set was used.

Chunksize	Strategy	Time
1500	base	8.38min
5000	base	3.59min
5000	map reduce	2.26min
5000	parallel map + reduce	12sec

Table 4.2: Summary generation execution time

As mentioned previously, the size of the chunk significantly affects execution time, enabling a speed-up of 2.33x. Additionally, parallelization results in an even more substantial reduction, saving 97.61% of the time.

#### 4.2.1 Quantitative Evaluation Results

The Tables 4.4 and 4.3 evaluate the performance across the two test sets by examining the differences achieved between GPT3.5-Turbo and GPT-4.0-Turbo. The evaluation considered the inclusion or exclusion of the README file as a source of information and focused on several metrics. For each project, three different summaries (A,B,C) were written and used as references.

From the reported results, the following observations can be made:

• GPT-3.5-Turbo consistently achieved higher precision and F1 scores than GPT-4.0-Turbo, suggesting that the model is more likely to select relevant words, although it occasionally misses terms present in the reference summaries. In contrast, GPT-4.0-Turbo, with its higher recall, captures a broader range of words, some of which may not be relevant.

SKILLNE	R		ROUGE-1			RC	DUGE-2		ROUGE-L		
Reference Summary	Model	README	Precision	Recall	F1	Precision	Recall	F1	Precision	Recall	F1
A	gpt-3.5-turbo	no	0.521	0.455	0.485	0.042	0.037	0.039	0.260	0.226	0.243
В	gpt-3.5-turbo	no	0.447	0.350	0.393	0.063	0.049	0.055	0.229	0.178	0.201
С	gpt-3.5-turbo	no	0.438	0.393	0.414	0.095	0.085	0.090	0.281	0.252	0.266
Α	gpt-4.0-turbo	no	0.295	0.600	0.395	0.036	0.073	0.048	0.133	0.273	0.180
В	gpt-4.0-turbo	no	0.290	0.528	0.374	0.036	0.066	0.046	0.143	0.260	0.184
С	gpt-4.0-turbo	no	0.295	0.610	0.398	0.036	0.075	0.049	0.152	0.318	0.205
A	gpt-3.5-turbo	yes	0.543	0.455	0.495	0.032	0.028	0.030	0.228	0.190	0.208
В	gpt-3.5-turbo	yes	0.457	0.341	0.391	0.044	0.033	0.038	0.217	0.163	0.186
С	gpt-3.5-turbo	yes	0.467	0.402	0.432	0.054	0.047	0.051	0.250	0.214	0.231
A	gpt-4.0-turbo	yes	0.270	0.591	0.370	0.046	0.100	0.063	0.129	0.282	0.176
В	gpt-4.0-turbo	yes	0.290	0.569	0.385	0.058	0.114	0.077	0.137	0.268	0.181
С	gpt-4.0-turbo	yes	0.274	0.617	0.379	0.050	0.113	0.069	0.116	0.261	0.161

Table 4.3: ROUGE evaluation scores for GPT-3.5-Turbo and GPT-4.0-Turbo summaries on Skillner Project. For each reference summary (A,B,C) the Table presents precision, recall and F1 scores for ROUGE-1, ROUGE-2, ROUGE-L, with and without access to the project's README.

	REPOAGENT		RC	DUGE-1		RC	UGE-2		ROUGE-L		
reference summary	model	README	precision	recall	F1	precision	recall	F1	precision	recall	F1
A	gpt-3.5-turbo	no	0.350	0.390	0.371	0.070	0.080	0.070	0.237	0.238	0.228
В	gpt-3.5-turbo	no	0.425	0.377	0.400	0.081	0.072	0.076	0.264	0.234	0.249
С	gpt-3.5-turbo	no	0.410	0.450	0.431	0.046	0.050	0.048	0.195	0.212	0.200
A	gpt-4.0-turbo	no	0.210	0.590	0.311	0.058	0.164	0.087	0.135	0.375	0.198
В	gpt-4.0-turbo	no	0.256	0.580	0.360	0.050	0.110	0.070	0.130	0.276	0.170
С	gpt-4.0-turbo	no	0.220	0.610	0.320	0.036	0.100	0.053	0.090	0.263	0.130
A	gpt-3.5-turbo	yes	0.350	0.487	0.408	0.100	0.139	0.116	0.210	0.287	0.241
В	gpt-3.5-turbo	yes	0.420	0.479	0.450	0.090	0.103	0.097	0.216	0.244	0.230
С	gpt-3.5-turbo	yes	0.378	0.525	0.440	0.063	0.089	0.074	0.171	0.238	0.200
A	gpt-4.0-turbo	yes	0.205	0.525	0.296	0.019	0.051	0.028	0.122	0.313	0.176
В	gpt-4.0-turbo	yes	0.270	0.561	0.364	0.025	0.052	0.033	0.132	0.276	0.178
С	gpt-4.0-turbo	yes	0.230	0.587	0.331	0.029	0.076	0.043	0.112	0.287	0.161

Table 4.4: ROUGE evaluation scores for GPT-3.5-Turbo and GPT-4.0-Turbo summaries on Repoagent Project. For each reference summary (A,B,C) the Table presents precision, recall and F1 scores for ROUGE-1, ROUGE-2, ROUGE-L, with and without access to the project's README.

This difference is likely due to GPT-3.5-Turbo being more conservative and concise compared to GPT-4.0-Turbo which tends to be more verbose and detailed.

- According to the ROUGE metric, the models capture a significant number of unigrams (ROUGE-1) from the reference summaries, demonstrating their ability to use relevant vocabulary. The lower ROUGE-2 score, however, suggests that the models struggle to capture correct pairs of consecutive words: they have more difficulty in generating specific word combinations. To conclude, the higher ROUGE-L score indicates that, while there is still room for improvement in aligning more closely with the reference, the model is better at maintaining the flow of information and preserving the overall structure of the summary.
- The inclusion of the README generally results in lower precision and higher recall, indicating that while additional context provided is beneficial, it may also introduce some confusion for the model.

In the Error Analysis subsection, we will observe in more detail how, as discussed in the Evaluation chapter, these metrics—while informative about the model's behaviour—are not necessarily accurate in assessing the quality of the solution.

#### 4.2.2 Qualitative Evaluation Results

In this use case, the quantitative evaluation discussed earlier has limited significance, the primary evaluation criterion is the summary's effectiveness in capturing and clearly conveying the project's core functionality. For this purpose, it is crucial to rely on qualitative evaluation.

Model	Cons.	Consistency	Coh.	Coherence	Rel.	Relevance	Flu.	Fluency
GPT-3.5- turbo	5	Factually consistent, reflecting automation, AI tools, and external services.	3	Coherent overview, missing detailed coverage of several components. Logical but lacks specifics.	3	Lacks key details like ChromaDB management and function roles. Incomplete relevance.	3	Clear, coherent, and error-free writing.
GPT-4.0- Turbo.0	5	Factually consistent, covers core functionalities without errors or hallucinations.	4	Coherent overview but lacks detail on some components. Well-structured and organized.	3	Captures goals but omits roles and introduces irrelevant details.	3	Clear and readable with no errors.
GPT-3.5- turbo + Readme	5	Consistent, reflects automation, README creation, and change detection.	2	Lacks coherence in detailing roles of classes and functions. Missing details.	2	Mentions related functionalities but lacks key details on classes/functions.	3	Well-written, no grammar or spelling errors.
Model-4.0 + Readme	5	Consistent, covers Markdown handling, change detection, and UI management.	3	General overview, lacks clear structure and specific details. Coherent but lacks explicit connections.	3	Captures goals but misses some key components and details. Introduces irrelevant elements.	3	Well-written with no errors, clear and effective communication.

Table 4.5: Qualitative Evaluation of GPT Models for Repoagent Summarization assessing consistency(1-5), coherence(1-5), relevance(1-5) and fluency (1-3)

Model	Cons.	Consistency	Coh.	Coherence	Rel.	Relevance	Flu.	Fluency
GPT-3.5- turbo	5	Captures text processing, skill extraction, and visualization without unsupported facts	5	Well-organized, outlines skill extraction, text cleaning, and matcher management.	5	Adheres closely to source texts, covering key aspects accurately.	3	Well-written, no grammar or spelling errors.
GPT-4.0- Turbo	5	Accurately reflects skill extraction, text processing, and visualization without factual errors.	5	Well-structured, presents objectives, functionalities, and capabilities logically.	5	Captures key aspects like text cleaning and skill extraction using NLP while avoiding irrelevant details.	3	Well-written, no grammar or spelling errors.
GPT-3.5-t with README	5	Reflects text cleaning, processing, skill extraction, and matcher creation without errors.	5	Logically structured, progressing from text preparation to visualization.	5	Focuses on skill extraction and processing without unnecessary details.	3	Well-written, no grammar or spelling errors.
GPT-4.0- Turbo with README	5	Factually consistent, covering NLP-based skill extraction, visualization, and conflict resolution.	5	Well-structured, presenting key functionalities like text cleaning, preprocessing, and annotation coherently	5	Covers NLP for skill recognition, visualization, resume parsing, and talent management.	3	Well-written with no errors, clear and effective communication.

Table 4.6: Qualitative Evaluation of GPT Models for Skillner Summarization assessing consistency(1-5), coherence(1-5), relevance(1-5) and fluency (1-3)

As illustrated in Table 4.5, the LLM's evaluation scored the generated summaries of the Skillner project as optimal. As the size of the project increased (Table 4.6), the evaluator was able to identify greater disparity between the approaches, which in some cases, led to lower scores as well. Consistency remained optimal, indicating that the generated output didn't contain hallucinations. In regards to relevance and coherence, the GPT-3.5-Turbo encountered difficulties in presenting the information in a coherent manner. For this reason, it was criticised by the evaluator for unclear textual organization and for failing to mention all the key points.

#### 4.2.3 Error Analysis

#### **GPT-4.0-Turbo summary**

This project is an advanced system designed to automate the generation, updating, and management of documentation for software repositories. At its core, the system facilitates the transfer and organization of Markdown documents, ensuring that README files are appropriately created and maintained within a project's directory structure. It leverages natural language processing and machine learning models to summarize documents, generate responses to user queries, and produce structured documentation that aligns with the codebase.

The system is adept at detecting changes within a Git repository, identifying modifications to Python files, and staging files based on predefined conditions. It also includes functionality for handling temporary files during the documentation process, ensuring a clean workspace post-generation.

A key feature of the system is its ability to interact with users through a chat interface, providing explanations and answers based on the project's documentation. This interactive component is supported by models that classify user questions and retrieve relevant information from the repository.

Additionally, the system manages configurations and settings, allowing for dynamic adjustments to project parameters. It integrates with external databases and services for storing document embeddings and utilizes a user interface for enhanced interaction.

Overall, the project streamlines the documentation workflow for developers, making it easier to keep documentation synchronized with code changes, manage project configurations, and engage with repository content through an AI-powered chat system.

#### **GPT-3.5-Turbo summary**

Repoagent is a project that automates the generation of documentation for code repositories by utilizing AI models and tools to analyze text, extract keywords, and provide answers to user queries. It integrates with external services like OpenAI, TextAnalysisTool, and Chroma-Manager to enhance the documentation process. The system interacts with users through a chat interface, generating responses based on user messages and retrieved information. The project aims to streamline documentation tasks, improve search capabilities, and facilitate efficient communication between users and the code repository.

Analysing both texts, is it possible to observe that the models provide a correct overview of the system's purpose. GPT-3.5's explanation is concise, yet it includes all the important details. On the other hand, GPT-4.0-Turbo is more verbose and offers a more comprehensive view of the project's details and functionalities. In a real-life context, GPT-4.0-Turbo's summaries would likely be more useful for developers looking for a deep-level understanding while GPT-3.5's explanation is better suited for audience looking for a quick overview.

### 4.3 Q&A Evaluation Results

#### 4.3.1 Quantitative Evaluation Results

In order to analyse the performance of the question answering task, in Table 4.7 the metrics computed on the two models are reported.

Category	Faithfulness	Answer Relevancy	<b>Context Recall</b>	<b>Context Precision</b>	Answer Correctness
General	0.757	0.778	0.603	0.954	0.655
Specific	0.801	0.913	0.804	0.953	0.715
Overall	0.741	0.900	0.703	0.955	0.685

T 11 4 7	$\sim$ ·	•	. • •	1
1 abla /1 //	( ) and the main	O 10 OTT TO MILLO OT	amontitativa	or rolling to the
I a nie d T	I IIIACIIAN	anew/erino	/mannanve	evanianon
$\mathbf{T}_{\mathbf{u}}$	Vucouton	answorme	uuanniarivu	Cvaruation
	X			
	~	<u> </u>	1	

In terms of retrieval performance, the specific model demonstrates stronger

context recall, making it more precise and less likely to lose important information. In contrast, the general model may synthesize too much of its broader project vision, potentially missing some key points.

However, both models exhibits high **context precision** indicating their effectiveness in filtering out irrelevant data once provided with context. This is a positive sign, as it suggests that the models, when used as a development support tool, are unlikely to introduce noisy information. In addition, as reflected in the **faithfulness** metric, they are less prone to hallucinations, meaning they tend to provide reliable responses and accurately define the context from which the information is drawn.

The metrics for **answer correctness** show higher performances for the specific model, this may be due to the fact that with a broader context (as in the general question case), the model struggles more to precisely identify the information to be returned.

#### 4.3.2 Qualitative Evaluation Results

The quality of the generated responses, as discussed in the previous chapter, was assessed using a multi-agent system, which has the key advantage of delivering tailored responses and suggestions by leveraging the expertise of different agents.

The full agent discussion is available in the Appendix C and highlights that, according to each expert, the system demonstrates a strong ability to provide accurate and well-structured responses to technical queries.

Although the system is able to deliver relevant and coherent answers, feedback also indicated a tendency to produce overly detailed responses, which could make it difficult for users to extract essential information efficiently. In this respect, the agent discussion made it possible to consider different perspectives in order to obtain a more complete vision of the problem.

In fact, through their conversation, the agents understood that verbosity isn't always a drawback. While providing too much information could be overwhelming, in a technical context it also allows for a comprehensive explanation and a full understanding of the answer.

Finally experts noted that prioritizing the most pertinent information at the beginning of the sentence would improve readability and accessibility.

To ensure robustness of this evaluation, the score assignment was repeated multiple times, showing that **coherence** and **consistency** always scored 8 out of 10 while **fluency** and **relevance** ranged between 8 and 9. The feedback remained consistent with the previously discusses observations.

#### 4.3.3 Error Analysis

#### **General Questions**

A complete and informative assessment is possible by examining some examples. In the following table (4.9), we considered three different formulations of general questions and their responses.

The examples show that in all cases the system correctly understands the question and provides a clear and informative response. However, one possible reason for the lower evaluation of the general-purpose responses is that they are not always complete. For instance, the first response correctly explains that the project focuses on documentation generation but fails to mention the chat interface and the system's integration with GitHub repositories. This is probably a consequence of the model further summarizing the information in its knowledge base, which in this case is already essential.

01	What is the core mission of this project?			
Υ	The core mission of this project is to streamline the management			
	of software documentation, ensuring it is automatically generated,			
	updated, and synchronized with code changes.			
Q2	Can you provide an overview of this project?			
	This project is focused on automating the documentation process for			
	software repositories. It monitors Git repositories for changes, up-			
	dates documentation accordingly, organizes it in a standardized for-			
	mat, and provides an AI-powered chat interface for user interaction			
	with the documentation.			
Q3	Explain Repoagent			
	Repoagent is a system designed to automate the management of soft-			
	ware documentation. It generates and maintains Markdown docu-			
	ments, detects changes within Git repositories, and provides an in-			
	teractive chat feature for users to query documentation. The system			
	utilizes natural language processing, machine learning, and a user			
	interface to enhance the documentation process.			

Table 4.8: Examples for error analysis of general questions and answers.

#### **Specific Questions**

As highlighted by the results, the specific model demonstrated superior performance in providing appropriate answers; however, it was still possible to identify examples that effectively represent some model's vulnerabilities observed during the exploration of the results.

The majority of the model's responses were accurate and clear, as demonstrated in the third example. However, the first and second examples indicate that while the model effectively provides adequate information, it may experience difficulties with clarity. As in Q2, responses were occasionally excessively verbose, making the answer unnecessarily complex. Additionally, (as in Q1), the model exhibits difficulties with readability due to redundancy and out-of-context words, such as "yes", at the beginning of the sentence.

The inspection of the result proved to be particularly useful, contributing to obtaining a comprehensive perspective on the system's capabilities. This, in turn, facilitated the conclusion reaching which is outlined in the next chapter. \_

Q1	What happens to the ChatMessageHistory object if the old mes-				
	sages are more than twice the max_messages limit and how				
	many old messages are kept?				
	Yes, if the old messages in the ChatMessageHistory object exceed				
	twice the max_messages limit, the object will keep only the most				
	recent messages up to the max_messages limit. The ChatMessage-				
	History object will retain the latest messages within the specified				
	limit while discarding the older messages that surpass the thresh-				
	old.				
Q2	What does the convert_to_markdown_file function do?				
	The convert_to_markdown_file function's purpose is to read a				
	project hierarchy JSON file to retrieve information about a speci-				
	fied file path. It then processes the file structure data and generates				
	markdown content based on the structure of the file. This function				
	facilitates the conversion of file content into markdown format for				
	documentation and presentation purposes.				
03	How does the get_map_reduce_chain function combine and re-				
QS	duce mapped documents using two LLMChain instances and a				
	chain of processes?				
	The get_map_reduce_chain function initializes two LLMChain in-				
	stances, one for mapping and one for reducing documents, to han-				
	dle the different stages of the MapReduce process efficiently. By				
	setting up a chain of processes, it iteratively combines and reduces				
	the mapped documents, streamlining the overall MapReduceDocu-				
	mentsChain operation effectively.				

Table 4.9: Examples for error analysis of specific questions and answers

## **Chapter 5**

## Conclusion

The conducted study provides valuable insights into the potential of Large Language Models in improving the understanding of technical projects. It introduces an intelligent query routing system capable of dynamically directing fine-grained and coarse-grained queries to suitable Retrieval-Augmented Generation (RAG) models, based on their complexity and information requirements. This strategy, combined with optimized prompt and summarisation techniques, significantly contributed to obtaining reliable results. The RA-GAs metrics retrieved high Context Precision (0.955) and Answer Relevancy (0.900) scores indicating the system's effectiveness in retrieving relevant information and providing pertinent answers.

Furthermore, the research underlined the different characteristics of GPT-4.0-Turbo and GPT-3.5-turbo in handling natural language generation tasks, with GPT-4.0-Turbo producing more detailed and comprehensive responses, sometimes at the cost of excessive verbosity. To conclude, the study proposed strategies to evaluate natural language generated texts with limited human intervention, using LLMs and multi-agent systems as evaluators. While human feedback remains invaluable, this investigation reduces human effort and improves the scalability of evaluations.

A fully automated pipeline that generates, explains, and evaluates itself using LLMs is indeed possible and can offer several benefits to the domain of Software Development Life Cycle.

#### 5.0.1 Future Work

Despite promising results, the system exhibits some limitations and necessities for further investigation, defining interesting research directions for future work.

In particular:

- Enhancing response quality: Exploring fine-tuning or advanced prompt engineering techniques to improve clarity and conciseness of model responses.
- Improving knowledge base comprehension: Exploring whether incorporating additional information sources, such as additional project descriptors (e.g. UML, diagrams, etc.), logs, or repositories information, would enhance the model's understanding of the project.
- Evaluating human-LLM alignment and quantifying real-scenario improvements: Comparison of human feedback with the evaluations provided by LLM evaluators and estimation of concrete Key Performance Indicators (KPIs) to quantify the impact of the proposed system on development workflows.

The findings presented here demonstrate the potential of LLMs to transform software development practices. Further exploration of these research directions would be crucial to assess the full potential of LLMs and enable a more robust integration into Software Development workflow. This would lead to more efficient development processes, reduced cognitive load for developers, and improved code quality, while optimising time and cost management.

## Acknowledgements

I would like to express my deep gratitude to everyone who contributed to the completion of this thesis. In particular, I sincerely thank DATA Reply for providing me with the opportunity to pursue this research within their inspiring environment and for equipping me with the necessary tools.

A special mention goes to my co-supervisor, Francesco Saracco, for his kindness and unwavering support throughout every stage of this study.

My deepest thanks are reserved for my supervisor, Paolo Torroni, whose availability, insightful feedback, and expertise have significantly shaped this dissertation.

### **Bibliography**

- X. Xia, L. Bao, D. Lo, Z. Xing, A. E. Hassan, and S. Li. Measuring program comprehension: a large-scale field study with professionals. *IEEE Transactions on Software Engineering*, PP:1–1, July 2017. DOI: 10.1109/TSE.2017.2734091.
- [2] R. Minelli, A. Mocci, and M. Lanza. I know what you did last summer

   an investigation of how developers spend their time. In 2015 IEEE
   23rd International Conference on Program Comprehension, pages 25– 35, 2015. DOI: 10.1109/ICPC.2015.12.
- [3] Atlassian. Software development life cycle (sdlc), 2025. URL: https: //www.atlassian.com/agile/software-development/sdlc. Accessed: 2025-03-01.
- [4] S. Feuerriegel, J. Hartmann, C. Janiesch, et al. Generative ai. Business & Information Systems Engineering, 66:111–126, 2024. DOI: 10. 1007/s12599-023-00834-7.
- [5] C. M. Bishop. *Pattern Recognition and Machine Learning*, volume 1. Springer, New York, NY, USA, 2006, page 740.
- [6] A. S. Pothukuchi, L. V. Kota, and V. Mallikarjunaradhya. Impact of generative ai on the software development lifecycle (sdlc). *International Journal of Creative Research Thoughts*, 11(8), August 2023. URL: https://ssrn.com/abstract=4536700. Available at SSRN.

- [7] Z. Cui, M. Demirer, S. Jaffe, L. Musolff, S. Peng, and T. Salz. The effects of generative ai on high-skilled work: evidence from three field experiments with software developers. Available at SSRN: https:// ssrn.com/abstract=4945566 or http://dx.doi.org/10.2139/ ssrn.4945566, February 2025. SSRN: 4945566.
- [8] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez,
  L. Kaiser, and I. Polosukhin. Attention is all you need, 2023. arXiv: 1706.03762 [cs.CL]. URL: https://arxiv.org/abs/1706.03762.
- [9] OpenAI. Gpt-4 research, 2025. URL: https://openai.com/index/ gpt-4-research/. Accessed: 2025-03-16.
- [10] L. Ouyang, J. Wu, X. Jiang, D. Almeida, C. L. Wainwright, P. Mishkin,
  C. Zhang, S. Agarwal, K. Slama, A. Ray, J. Schulman, J. Hilton, F. Kelton, L. Miller, M. Simens, A. Askell, P. Welinder, P. Christiano,
  J. Leike, and R. Lowe. Training language models to follow instructions with human feedback, 2022. arXiv: 2203.02155 [cs.CL]. URL: https://arxiv.org/abs/2203.02155.
- J. Schulman, F. Wolski, P. Dhariwal, A. Radford, and O. Klimov. Proximal policy optimization algorithms, 2017. arXiv: 1707.06347 [cs.LG].
   URL: https://arxiv.org/abs/1707.06347.
- [12] T. Mu, A. Helyar, J. Heidecke, J. Achiam, A. Vallone, I. Kivlichan, M. Lin, A. Beutel, J. Schulman, and L. Weng. Rule based rewards for language model safety, 2024. arXiv: 2411.01111 [cs.AI]. URL: https://arxiv.org/abs/2411.01111.
- [13] A. Koubaa. Gpt-4 vs. gpt-3.5: a concise showdown. *TechRxiv*, April 2023. DOI: 10.36227/techrxiv.22312330.v2.

- [14] Y. Gao, Y. Xiong, X. Gao, K. Jia, J. Pan, Y. Bi, Y. Dai, J. Sun, M. Wang, and H. Wang. Retrieval-augmented generation for large language models: a survey, 2024. arXiv: 2312.10997 [cs.CL]. URL: https:// arxiv.org/abs/2312.10997.
- Q. Luo, Y. Ye, S. Liang, Z. Zhang, Y. Qin, Y. Lu, Y. Wu, X. Cong,
  Y. Lin, Y. Zhang, X. Che, Z. Liu, and M. Sun. Repoagent: an Ilm-powered open-source framework for repository-level code documentation generation, 2024. arXiv: 2402.16667 [cs.CL]. URL: https: //arxiv.org/abs/2402.16667.
- [16] N. K. Nagwani. Summarizing large text collection using topic modeling and clustering based on mapreduce framework. *Journal of Big Data*, 2(1):6, June 2015.
- [17] V. Priya and K. Umamaheswari. Aspect-based text summarization using mapreduce optimization. In Computational Intelligence and Sustainable Systems: Intelligence and Sustainable Computing. H. Anandakumar, R. Arulmurugan, and C. C. Onn, editors. Springer International Publishing, Cham, 2019, pages 131–139.
- [18] T. B. Brown, B. Mann, N. Ryder, et al. Language models are fewshot learners. arXiv preprint arXiv:2005.14165, 2020. URL: https: //arxiv.org/abs/2005.14165.
- [19] J. Devlin, M.-W. Chang, K. Lee, et al. Bert: pre-training of deep bidirectional transformers for language understanding. arXiv preprint arXiv:1810.04805, 2019. URL: https://arxiv.org/abs/1810.04805.
- [20] G. Marcus. The next decade in ai: four steps towards robust artificial intelligence. arXiv preprint arXiv:2002.06177, 2020. URL: https:// arxiv.org/abs/2002.06177.
- [21] T. Li, G. Zhang, Q. D. Do, X. Yue, and W. Chen. Longiclbenc: a benchmark for long-range image captioning with large contexts. arXiv preprint arXiv:2404.02060. URL: https://arxiv.org/pdf/2404.02060.

- [22] Y. Deng, W. Zhang, Z. Chen, and Q. Gu. Rephrase and respond: let large language models ask better questions for themselves. arXiv preprint, arXiv:2311.04205, 2024. URL: https://arxiv.org/pdf/2311. 04205.
- [23] L. Yan, Y. Zheng, and J. Cao. Few-shot learning for short text classification. *Multimedia Tools and Applications*, 77(27):29799–29810, 2018.
   DOI: 10.1007/s11042-018-5772-4.
- [24] LangChain. Self-querying retrieval, 2025. URL: https://python. langchain.com/docs/how\_to/self\_query/. Accessed: 2025-02-17.
- [25] L. Bojic, P. Kovacevic, and M. Cabarkapa. Gpt-4 surpassing human performance in linguistic pragmatics. *arXiv preprint arXiv:2312.09545*, 2023. URL: https://arxiv.org/abs/2312.09545.
- [26] C. van der Lee, A. Gatt, E. van Miltenburg, and E. Krahmer. Human evaluation of automatically generated text: current trends and best practice guidelines. *Computer Speech Language*:101151, 2021. URL: https: //www.sciencedirect.com/science/article/pii/S088523082030084X.
- [27] E. Lloret, L. Plaza, and A. Aker. The challenging task of summary evaluation: an overview. *Language Resources and Evaluation*, 52(1), 2018.
   ISSN: 1574-0218. URL: https://doi.org/10.1007/s10579-017-9399-2.
- [28] E. Reiter and A. Belz. An investigation into the validity of some metrics for automatically evaluating natural language generation systems. *Computational Linguistics*, 35(4), December 2009. URL: https:// aclanthology.org/J09-4008/.
- J. Sjöbergh. Older versions of the rougeeval summarization evaluation system were easier to fool. *Inf. Process. Manage.*, 43(6):1500–1505, November 2007. ISSN: 0306-4573. DOI: 10.1016/j.ipm.2007.01.
   014. URL: https://doi.org/10.1016/j.ipm.2007.01.014.

- [30] S. Fareri, N. Melluso, F. Chiarello, and G. Fantoni. Skillner: mining and mapping soft skills from any text. *Expert Systems with Applications*, 184:115544, December 2021. ISSN: 0957-4174. DOI: 10.1016/j.eswa.2021.115544. URL: http://dx.doi.org/10.1016/j.eswa.2021.115544.
- [31] C.-Y. Lin. ROUGE: a package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pages 74–81, Barcelona, Spain. Association for Computational Linguistics, July 2004. URL: https: //aclanthology.org/W04-1013/.
- [32] F. Liu and Y. Liu. Exploring correlation between rouge and human evaluation on meeting summaries. *IEEE Transactions on Audio, Speech,* and Language Processing, 18(1):187–196, 2010. DOI: 10.1109/TASL. 2009.2025096.
- [33] W. Tay, A. Joshi, X. Zhang, S. Karimi, and S. Wan. Red-faced ROUGE: examining the suitability of ROUGE for opinion summary evaluation. In M. Mistica, M. Piccardi, and A. MacKinlay, editors, *Proceedings* of the 17th Annual Workshop of the Australasian Language Technology Association, Sydney, Australia. Australasian Language Technology Association, 2019. URL: https://aclanthology.org/U19-1008/.
- [34] J. Wei, X. Wang, D. Schuurmans, M. Bosma, B. Ichter, F. Xia, E. Chi, Q. Le, and D. Zhou. Chain-of-thought prompting elicits reasoning in large language models, 2023. arXiv: 2201.11903 [cs.CL]. URL: https://arxiv.org/abs/2201.11903.
- [35] C.-M. Chan, W. Chen, Y. Su, J. Yu, W. Xue, S. Zhang, J. Fu, and Z. Liu. Chateval: towards better llm-based evaluators through multi-agent debate, 2023. arXiv: 2308.07201 [cs.CL]. URL: https://arxiv. org/abs/2308.07201.

[36] W. Chen, Y. Su, J. Zuo, C. Yang, C. Yuan, C.-M. Chan, H. Yu, Y. Lu, Y.-H. Hung, C. Qian, Y. Qin, X. Cong, R. Xie, Z. Liu, M. Sun, and J. Zhou. Agentverse: facilitating multi-agent collaboration and exploring emergent behaviors, 2023. arXiv: 2308.10848 [cs.CL]. URL: https://arxiv.org/abs/2308.10848.

## **Appendix A**

# **Qualitative Evaluation of Summary: Prompt Example**

"You will be given the main point of a project documentation. You will then be given the summary written on this documentations.

Your task is to rate the multi document summarization on one metric.

Please make sure you read and understand these instructions carefully. Please keep this document open while reviewing, and refer to it as needed.

#### Evaluation Criteria:

Relevance (1-5) - selection of important content from the main point. The summary should include only important information from the main point. Annotators were instructed to penalize summaries which contained redundancies and excess information. Evaluation Steps:

1. Read the identified main features and key points.

2. Compare the summary to the main points and identify the main

features.

3. Assess how well the summary covers the defined main points, and how much irrelevant or redundant information the summary contains.

4. Assign a relevance score from 1 to 5.

Input data:

Main Point to be Compared: {main\_point} Summary to be Evaluated: {summary}

Evaluation Form (scores ONLY):

- Relevance: then a brief explanation of the evaluation criteria you followed."

## **Appendix B**

## **Classification Model Prompt**

"You are a classifier that identifies general question in text. If the text matches the general examples, classify it as '*general*'. If it doesn't match, classify it as '*specific*'.

A question is 'general' if it asks about broad or high-level information about the project, such as its purpose, function, or overall description.

So If a question is trying to understand how the project works, what it does, or what its main goal is, it is considered 'general'.

#### **Examples**:

general: "Explain me what the repository is about"
general: "Explain what the project is about"
general: "What is this program doing?"
general: "How does the project work?"
general: "What is the project about?"
general: "What is the purpose of this repository?"
general: "What is the main goal of the project?"
general: "How is this tool working?"

general: "What is the project's main function?"

Classify the following text: Text: "{text}" Classification:

Please output just "general" or "specific" without quotes and without any additional information."

## **Appendix C**

## **Multi-Agent System Conversation**

#### **Initial Plan by Software Developer**

Based on the given examples, I believe the question-answer system has performed commendably. It is able to provide detailed and correct answers to the posed questions, demonstrating a deep understanding of the topic. The system seems to excel at interpreting technical questions and returning precise answers, which is crucial in this context.

For example, the system's responses to questions about specific functions such as 'build\_path\_tree' and 'add\_parent\_references' were spot on, demonstrating its ability to accurately interpret and answer complex technical questions. Its answers to broader queries about processes like 'identifying file changes since the last commit' and 'generating search queries' were also comprehensive and accurate, showing its ability to handle a range of question types.

However, there are areas for potential improvement. The system could work on making its answers more concise and straightforward without losing the essential details. Some responses are quite verbose, which can be overwhelming and might make it difficult for the user to find the specific information they're looking In conclusion, the question-answer system demonstrates a high level of competency, providing accurate and detailed responses to a range of questions. With some minor improvements in conciseness and clarity, it could prove to be an excellent tool for users.

#### Reviews

**[Data scientist]:** While I agree with most of your points, I believe the verbosity of the responses is not necessarily a drawback. In technical contexts, detailed responses can help ensure that the user fully understands the answer. However, the system could benefit from improvements in the relevance of search results to enhance the efficiency and accuracy of information retrieval.

[**Project manager**]: While I mostly agree with the feedback given, I would like to add that the system's ability to provide contextspecific answers is crucial. The length of the answers, while might seem verbose, can be actually beneficial for users looking for comprehensive explanations. On the other hand, improving the system's ability to understand the user's context and provide answers of appropriate length and detail according to that context would be a valuable enhancement.

**[Technical writer]**: While the system provides detailed responses, it lacks a level of clarity and simplicity essential for effective technical communication.

#### **Updated Plan**

Based on the analysis of the given responses and the input from my teammates, I believe the question-answer system performed

for.

well in providing detailed and accurate responses. It demonstrated a solid understanding of various technical topics, from specific functions to broader processes, which is highly commendable.

However, there's a notable room for improvement in the areas of response clarity, conciseness, and context-awareness. While the detailed responses can be beneficial for users seeking in-depth explanations, they can also be overwhelming for those who prefer straightforward answers. The system's ability to adapt the detail and complexity of its responses based on the user's context and preference would be a significant enhancement.

Moreover, the relevance and efficiency of information retrieval in response to search queries can be improved. The system could focus on providing the most relevant information first, enhancing the user experience by quickly delivering the desired information.

In conclusion, the system shows promising capabilities and, with a few adjustments, could enhance its performance significantly. The key lies in balancing detailed explanations with clear, concise communication while ensuring context-awareness and efficient information retrieval.

#### Reviews

**[Data scientist]**: The system's performance can be improved by optimizing the relevance and efficiency of information retrieval in response to search queries. It could focus on extracting the most relevant information first, which would enhance the user experience by quickly delivering the desired information. Additionally, considering the scalability aspect, the system should be designed to handle a massive amount of data without compromising the retrieval speed and accuracy, which is crucial in a real-world scenario. The system also needs to incorporate advanced techniques such as query expansion and semantic understanding for better interpretation of user queries and delivering more accurate responses.

[Project manager]: I have nothing to add.

[Technical Writer]: While the system provides detailed answers, it's crucial to enhance the clarity and simplicity of the responses to ensure they are easily understandable to a wide range of users. It's not about reducing the detail, but about presenting complex information in a more digestible manner.

#### **Updated Plan**

Based on the evaluation of the responses, the question-answer system appears to have a strong understanding of various technical topics and is capable of providing detailed and accurate answers. This level of comprehension and detail can be particularly beneficial for users seeking comprehensive explanations.

However, there are several areas where the system could improve. For one, while the detail provided in the responses can be beneficial, it could potentially be overwhelming for users who are seeking more straightforward answers. In this regard, the system could benefit from being more adaptable, adjusting the length and detail of its responses based on the user's needs and the context of the question.

Additionally, while the system is able to generate relevant search queries and retrieve information efficiently, there could be improvements made in the way this information is presented. Prioritizing the most relevant information and presenting it first could improve the user experience by providing the desired information more quickly.

In terms of clarity and simplicity, the system could work on presenting complex information in a more digestible manner, making the responses easier to understand for a wider range of users.

Finally, advanced techniques such as query expansion and semantic understanding could be incorporated to improve the interpretation of user queries and the accuracy of responses.

In conclusion, while the question-answer system is performing well in many areas, there are several improvements that could be made to enhance its performance and user experience. With these adjustments, the system could provide even more accurate, efficient, and user-friendly service.

#### **Reviews:**

[Data scientist]: I agree.

[Technical writer]: I agree.

[Project manager]: I agree.

#### **Final Plan**

Based on the evaluation of the responses, the question-answer system appears to have a strong understanding of various technical topics and is capable of providing detailed and accurate answers. This level of comprehension and detail can be particularly beneficial for users seeking comprehensive explanations.

However, there are several areas where the system could improve. For one, while the detail provided in the responses can be beneficial, it could potentially be overwhelming for users who are seeking more straightforward answers. In this regard, the system could benefit from being more adaptable, adjusting the length and detail of its responses based on the user's needs and the context of the question.

Additionally, while the system is able to generate relevant search queries and retrieve information efficiently, there could be improvements made in the way this information is presented. Prioritizing the most relevant information and presenting it first could improve the user experience by providing the desired information more quickly.

In terms of clarity and simplicity, the system could work on presenting complex information in a more digestible manner, making the responses easier to understand for a wider range of users.

Finally, advanced techniques such as query expansion and semantic understanding could be incorporated to improve the interpretation of user queries and the accuracy of responses.

In conclusion, while the question-answer system is performing well in many areas, there are several improvements that could be made to enhance its performance and user experience. With these adjustments, the system could provide even more accurate, efficient, and user-friendly service.

#### Scoring

- Consistency: 8
- Coherence: 8
- Fluency: 9
- Relevance: 8

 Advice: While the responses are technically sound and detailed, it would be beneficial to aim for conciseness without compromising the completeness of the answer. Some responses may be overwhelming for the reader due to their length.

Retrying ...

- Consistency: 8
- Coherence: 8
- Fluency: 8
- Relevance: 9
- Advice: The answers are overall detailed, relevant, and wellstructured, with minor room for improvement in terms of conciseness. The writer could consider reducing some redundancy and unnecessary details to make the responses more concise and to the point.

Retrying ...

- Consistency: 8
- Coherence: 8
- Fluency: 8
- Relevance: 9
- Advice: The answers are overall detailed, relevant, and wellstructured, with minor room for improvement in terms of conciseness. The writer could consider reducing some redundancy and unnecessary details to make the responses more concise and to the point.