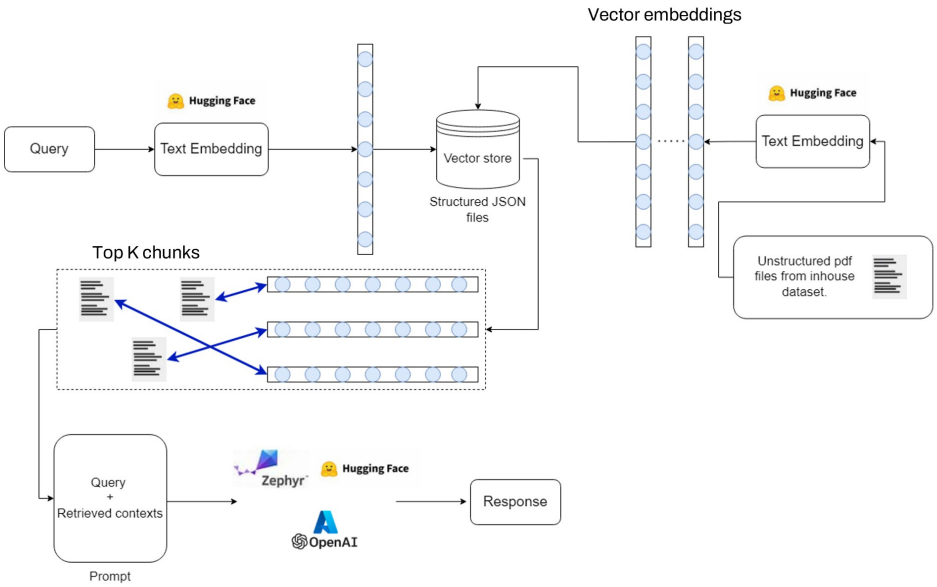
CHALMERS
UNIVERSITY OF TECHNOLOGY



Productivity Applications of Language Intelligence Modeling in Domain Specific Research

Master's Thesis in Systems, Control and Mechatronics

Sanam Molaee

DEPARTMENT OF ELECTRICAL ENGINEERING
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2024
www.chalmers.se

Master's thesis in Systems, Control and Mechatronics

Productivity Applications of Language Intelligence Modeling in Domain Specific Research

Sanam Molaee

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Chalmers University of Technology
Gothenburg, Sweden 2024

Productivity Applications of Language Intelligence Modeling in Domain Specific Research
Sanam Molaee

Supervisor: Dr. Jawwad Ahmed, Autoliv Research
Examiner: Prof. Martin Fabian, Department of Electrical Engineering, Chalmers

Degree project report 2024
Department of Electrical Engineering
Chalmers University of Technology
SE-412 96 Gothenburg
Sweden
Telephone +46 31 772 1000

Cover: Retrieval Augmented Generation Pipeline

Typeset in LaTeX
Gothenburg, Sweden 2024

Productivity Applications of Language Intelligence Modeling in Domain Specific Research
Sanam Molaee
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The popularity of large language models (LLMs) has led to their widespread adoption in various domains for diverse tasks. These models, characterized by their vast number of parameters, show advanced capabilities like solving complex problems and multi-step reasoning. Despite potential risks, they are seen as promising for enhancing productivity and workflows in numerous areas.
The aim of this project is to integrate LLMs into research workflows in the automotive safety domain to streamline processes. It focuses on assessing and selecting a suitable open-source LLM for this purpose, as well as implementing capabilities such as semantic search and automatic insight generation from the given data. For the complex question answering, a retrieval augmented generation (RAG) pipeline is implemented, which is then shown to be a viable approach to exploiting the capabilities of large language models. The key deliverable is a proof-of-concept demonstrating the practical application of LLMs in processing and analyzing domain-specific data. In this project it is shown that the open-source LLM can have a significant role in enhancing the productivity of domain specific research. Also, by comparing the performance with GPT-3.5 Turbo, which is a proprietary model and has higher costs, it can be seen that the open-source model provides a competitive performance.

Keywords: Natural language processing, Generative AI, Large language models, Retrieval augmented generation

Acknowledgements

I would like to thank Dr. Jawwad Ahmed, my industrial supervisor at Autoliv, whose insights and guidance has been an invaluable assist for me throughout this project.

I would also like to thank Prof. Martin Fabian, my academic supervisor and examiner at Chalmers, for his helpful comments, patience and support during this work.

Sanam Molaee, Gothenburg, 2024

List of Acronyms

Below is the list of acronyms that have been used throughout this thesis listed in alphabetical order:

BERT	Bidirectional Encoder Representations from Transformers
GPT	Generative Pre-trained Transformer
GUI	Graphical User Interface
GT	Ground truth
LLM	Large Language Model
NLP	Natural language processing
PLM	Pre-trained language models
RAG	Retrieval Augmented Generation

Contents

List of Figures

List of Tables

1
Introduction

This project has been done in collaboration with Autoliv for exploring the integration of LLMs into research workflows to enhance cognitive capabilities and streamline processes. It focuses on assessing and selecting suitable LLM for enterprise settings and building a research system with capabilities such as semantic search and automatic insight generation from Autoliv's inhouse dataset containing publications in the field of automotive safety. The key deliverable of this research is a proof-of-concept demonstrating the practical application of LLMs in processing and analyzing domain-specific data.

1.1 Background

The popularity of LLMs has rapidly increased, leading to their widespread adoption across various domains for different tasks. These models, characterized by their vast number of parameters, exhibit advanced capabilities such as solving complex problems, performing multi-step reasoning, and generating human-like text. Their proficiency also extends to tasks like text summarization, translation, sentiment analysis, and conversational agents, making them invaluable tools in sectors such as healthcare, finance, education, and research.
Despite the potential risks associated with their use, including ethical concerns, biases, and the possibility of generating misleading information, LLMs are increasingly seen as transformative for enhancing productivity and workflows [1]. Their ability to automate repetitive tasks, provide insights from large datasets, and assist in decision-making processes can significantly boost efficiency and accuracy. For example, in educational settings, LLMs can provide personalized learning experiences and automate grading [2].
Overall, the adoption of LLMs marks a significant step towards leveraging artificial intelligence to improve productivity, drive innovation, and tackle complex challenges in numerous domains.

1.2 Motivation

The motivation for this thesis comes from the rapid growth and capabilities of LLMs in transforming research methodologies. With the increasing complexity and volume of data in research, there is a growing need for tools that can enhance efficiency and productivity. Also, mostly the research data we have is unstructured, spread across many sources, vast in volume, and sensitive in nature. This motivates the need for

an advanced NLP based solution that can not only provide advanced knowledge extraction capabilities, but also provides the ability of a digital expert to rapidly respond to complex technical questions in the specific area in a secure enterprise environment. This thesis is driven by the potential of LLMs to revolutionize data analysis, information retrieval, and insight generation in domain-specific research, addressing current limitations.

1.3 Objectives

The main goal of the project is to explore current open-source LLMs for enhancing research workflows. It focuses on assessing and choosing suitable LLMs for enterprise use, based on criteria including complexity, capabilities, and performances. The project will concentrate on in-context learning and history maintenance to utilize the zero/few-shot learning abilities of LLMs for domain-specific tasks.

1.4 Research questions

Three key questions are discussed in this research.

1. Can current state-of-the-art LLM technologies be used to improve the research workflows in the automotive safety domain by enhancing productivity or quality of work?
2. Is exploiting zero/few shot capabilities of LLMs, for example RAG, a viable approach?
3. Under the RAG scenario, can open-source LLMs provide competitive performance compared to closed source/proprietary LLM models?

2
Theory

This chapter discusses the theoretical foundations of this project including key areas such as NLP, the architecture of Transformers, the importance of Text Embeddings, LLMs, methods for using them, and the evaluation metrics.

2.1 Natural language processing

NLP involves techniques designed to allow computers to interpret, process, and produce human language in ways that are both meaningful and practical [3]. This field combines elements of computational linguistics, which provides structured models for understanding language, with various machine learning and deep learning approaches. These models are developed to simulate human language understanding capabilities, enabling computers to interact with humans through text or spoken words.

2.2 Transformers

Transformers are models specifically engineered for handling sequential data, such as text, for NLP tasks [4]. In transformers, data processing is facilitated entirely through layers known as attention layers. These layers are designed to dynamically weigh the relevance of different parts of the input data. The transformer architecture generally consists of an encoder and a decoder. The encoder processes the input data into a continuous representation that holds all the learned insights about the input. The decoder then uses this representation, along with previous output elements, to generate the output sequence. Each layer in both the encoder and decoder contains a series of self-attention and feed-forward neural network layers, which help in refining the transformations and representations at each step. The architecture of a transformer can be seen in Figure 2.1.

2.3 Text embeddings

Text embeddings are an important element in NLP tasks. These embeddings transform text into low-dimensional vector forms that encapsulate the semantic relationships of words or phrases within text [5]. This transformation is achieved through vector operations, which mathematically analyze and map the semantic similarities and differences between texts into a spatial representation. Cosine similarity is a

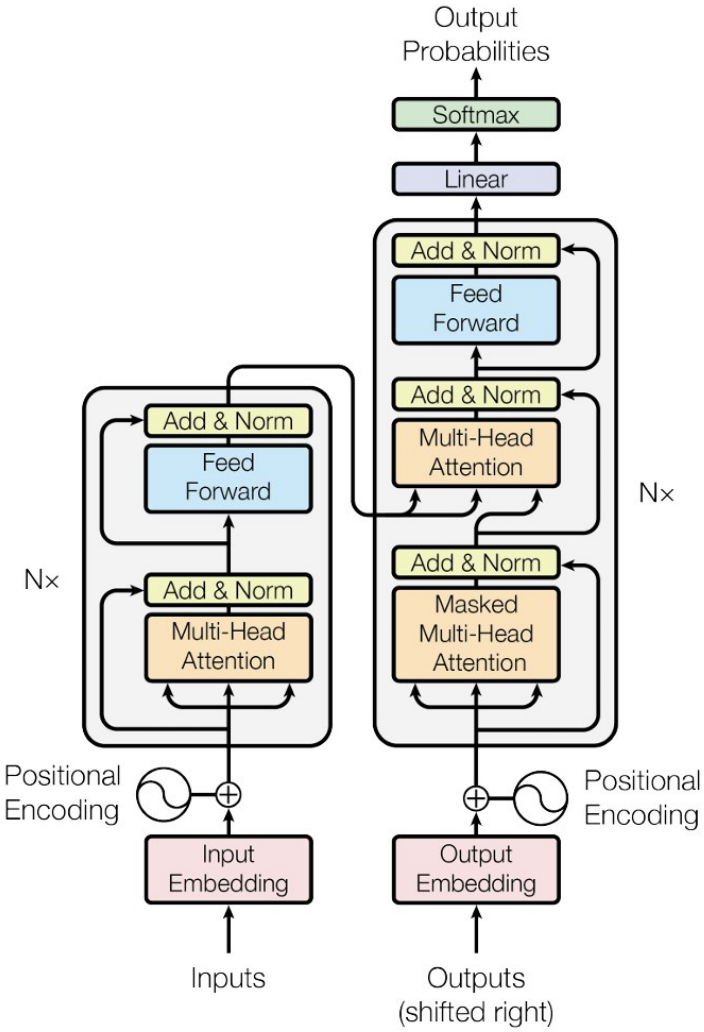


Figure 2.1: Transformer architecture [4]

key metric in this context. It measures the cosine of the angle between two vectors, providing a value between -1 and 1. These values indicate how similar the two vectors are, with 1 meaning two vectors are pointing in the same direction and have maximum similarity, 0 meaning they are orthogonal with no similarity, and -1 meaning they are pointing in exactly opposite directions with maximum dissimilarity [6]. Mathematically, cosine similarity between two vectors $\mathbf{A}$ and $\mathbf{B}$ is calculated as:

$$\text{cosine similarity} = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} \tag{2.1}$$

This measure is particularly useful in semantic search and information retrieval, where it helps in finding documents or text segments that are contextually similar to a given query. By focusing on the direction rather than the magnitude of the vectors, the cosine similarity effectively captures the semantic relevance, making it a powerful tool for comparing text embeddings.

2.4 Large language models

LLMs represent a subset of NLP models, pre-trained on vast amounts of text data from diverse sources, which equip them with a comprehensive grasp of human language. Based on transformer architectures, these models utilize attention mechanisms to enhance their predictive and generative text capabilities. LLMs are trained using a process called unsupervised learning. This involves feeding the model massive amounts of text data and having the model learn the patterns and relationships between words and phrases in the text. They are known for their ability to generate coherent and contextually appropriate text responses from given prompts. Their major advantages include the ability to adapt to various language tasks without direct task-specific training, often demonstrating human-like writing abilities [3].

2.4.1 Types

One prominent model based on the Transformer architecture is BERT [7]. BERT focuses on understanding the context of a word based on its surrounding words in both directions (left and right). This bidirectional approach allows for a more comprehensive grasp of linguistic subtleties, making BERT particularly effective for tasks such as question answering and sentiment analysis.
Another significant model in the LLM landscape is the GPT Series, including GPT-2 and GPT-3 [8, 9]. GPT models are autoregressive, meaning they generate text by predicting the next word in a sequence based on the words that came before it. These models are pre-trained on a large corpus of text and then fine-tuned for specific tasks. GPT models have demonstrated remarkable capabilities in text generation, conversation, and even creative writing, thanks to their extensive pre-training on diverse datasets.
T5 (Text-To-Text Transfer Transformer) [10], represents another innovative approach in the realm of LLMs. T5 converts all NLP tasks into a text-to-text format, enabling a unified framework for various tasks such as translation, summarization, and question answering. This text-to-text paradigm simplifies the model architecture and makes it versatile in handling different NLP tasks with a single model.

2.4.2 Parameters

LLMs have various parameters, but three of the most essential ones are as follows.

1. Top K: This parameter limits the model's predictions to the k most probable tokens at each generation step. By assigning a value to k, the model will be guided to focus on certain tokens with the most probability.
2. TOP P: This model controls the cumulative probability of the generated tokens. The tokens will be generated until the cumulative probability exceeds the threshold P.
3. Temperature: This is a scaling factor, applied within the model to shape the probability distribution of the next model. Higher temperature adds more randomness in the output.

2.5 Techniques for using LLMs

There are three main techniques for adapting LLMs to a specific task.

2.5.1 Prompt engineering

Prompt engineering is the process of creating and using specific instructions, known as prompts, to guide the output of pre-trained LLMs [11]. This technique is significant because it modifies model behavior without altering the core model parameters, enhancing the flexibility and effectiveness of these models in various applications. The essence of prompt engineering lies in its capacity to integrate pre-trained models seamlessly into diverse tasks through well-crafted instructions. This approach contrasts with traditional methods that often necessitate extensive model retraining or fine-tuning, presenting a more efficient alternative for adapting models to new tasks.
There are various prompt engineering techniques, but the focus in this project will be on two of them:

1. Zero-shot prompting: where the model leverages its general knowledge to tackle new tasks without specific training, based on the prompt description alone.
2. Few-shot prompting: This method enhances model understanding of a task by providing a handful of examples, improving its performance on similar tasks.

2.5.2 Fine-tuning

Fine-tuning involves taking a pre-trained LLM and adapting it to a specific task or domain by further training it on a smaller, task-specific dataset. This additional training process fine-tunes the model's parameters, allowing it to perform better on the new task compared to its performance as a general-purpose language model [12]. Although highly effective in generating precise outputs, this approach comes with significant initial costs, such as computational resources [13]. Also, fine-tuning usually is a supervised learning process, which necessitates having a labeled dataset.

2.5.3 RAG

RAG is a model designed to enhance the performance of language models on knowledge-intensive tasks by integrating retrieval mechanisms with generative models [14]. The key innovation of RAG is to augment the generation process with relevant information fetched from a large external corpus, enabling the model to produce more accurate and contextually rich responses. Main components of a RAG pipeline are [15]:

1. Retrieval: This component retrieves relevant documents based on input queries using dense vector representations, which facilitates efficient and accurate similarity computations.
2. Generation: In this step the generative model conditions on the input query and the retrieved documents to generate responses. This allows the model to

leverage up-to-date external knowledge, enhancing contextual relevance and factual accuracy.

Different types of RAG include:

1. Naive RAG: This combines retrieval and generation with minimal interaction. The generative model directly uses retrieved documents as additional context.
2. Advanced RAG: This employs techniques like fine-tuning of the retrieval mechanism and alignment between retrieval and generation to enhance the quality of responses to ensure better relevance and coherence.
3. Modular RAG: This separates RAG into distinct, interchangeable modules for retrieval, generation, and augmentation, allowing for more targeted improvements and flexibility.

2.6 Evaluation

The performance of a RAG pipeline can be evaluated by various metrics [16]. In this research, the focus has been on six metrics discussed in this section.

2.6.1 Faithfulness

The faithfulness metric evaluates the factual consistency of the generated answer compared to the provided context, calculating it from the answer and the retrieved context. The results are scaled to a range of [0,1], with higher scores indicating better performance. An answer is considered faithful if all the statements made within it can be derived from the context provided. To determine this, a list of claims from the generated answer is first identified. Each of these claims is then cross-referenced against the given context to verify if it can indeed be inferred from it. The faithfulness score is calculated as:

$$\text{Faithfulness score} = \frac{|\text{Number of claims in the generated answer that can be inferred from given context}|}{|\text{Total number of claims in the generated answer}|} \tag{2.2}$$

2.6.2 Answer relevancy

The answer relevancy metric measures the relevance of the generated answer to the specified prompt. It assigns lower scores to answers that are either incomplete or contain excessive information, while higher scores reflect greater relevance. This metric is calculated using the question, its context, and the answer. Answer relevancy is determined by calculating the average cosine similarity between the original question and a set of artificial questions that have been reverse-engineered from the answer. The answer relevancy score is calculated as:

$$\text{Answer relevancy} = \frac{1}{N}\sum_{i=1}^{N}\cos(E_{gi}, E_o) = \frac{1}{N}\sum_{i=1}^{N}\frac{E_{gi}\cdot E_o}{\|E_{gi}\|\|E_o\|}, \tag{2.3}$$

where E_{gi} is the embedding of the generated question i, E_o is the embedding of the original question, and N is the number of generated questions.
It is worth mentioning that although the score typically ranges between 0 and 1, in practice this range is not mathematically guaranteed. This is because, as mentioned earlier, the nature of cosine similarity allows for values ranging from -1 to 1.

2.6.3 Context precision

Context precision is a metric designed to assess whether all relevant ground truth items in the contexts are ranked appropriately high. Ideally, all relevant chunks should appear in the topmost ranks. This metric is calculated using the question, ground truth data, and the contexts, with values spanning from 0 to 1. Higher scores indicate greater precision. Context precision is calculated as:

$$\text{Context Precision@K} = \frac{\sum_{k=1}^{K}(\text{Precision@k} \times v_k)}{\text{Total number of relevant items in the top } K \text{ results}} \tag{2.4}$$

$$\text{Precision@k} = \frac{\text{true positives@k}}{\text{true positives@k} + \text{false positives@k}} \tag{2.5}$$

where K is the total number of chunks in the contexts and $v_K \in \{0, 1\}$ is the relevance indicator at rank k.

2.6.4 Context recall

Context recall evaluates how closely the retrieved context matches the annotated answer, considered as the ground truth. This metric is calculated using the ground truth and the retrieved context, with values ranging from 0 to 1, where higher scores indicate better performance. To assess context recall from the ground truth answer, each sentence within the ground truth answer is examined to see if it corresponds to the retrieved context. Ideally, every sentence in the ground truth answer would be attributed to the retrieved context. The context recall score can be calculated as:

$$\text{Context recall} = \frac{|\text{GT sentences that can be attributed to context}|}{|\text{Number of sentences in GT}|} \tag{2.6}$$

2.6.5 Answer semantic similarity

Answer semantic similarity enables evaluating how semantically similar the generated answer is to the ground truth. This metric is derived from both the ground truth and the answer, with scores ranging from 0 to 1. Higher scores indicate a stronger semantic alignment between the generated answer and the ground truth. Assessing semantic similarity provides important information about the quality of the generated response. This process employs a cross-encoder model to determine the semantic similarity score. The process is done through three steps:

1. Vectorize the ground truth answer using the specified embedding model.
2. Vectorize the generated answer using the same embedding model.
3. Compute the cosine similarity between the two vectors.

2.6.6 Answer correctness

The answer correctness score measures the precision of the generated answer relative to the ground truth. This analysis depends on comparing the ground truth with the answer, and the scoring system ranges from 0 to 1. A higher score reflects a more accurate match with the ground truth, indicating greater correctness. Answer correctness involves two essential factors: the semantic similarity and the factual accuracy between the generated answer and the ground truth. These factors are combined using a weighted formula to calculate the answer correctness score. The computation of the answer correctness score involves adding the factual correctness and the semantic similarity between the provided answer and the ground truth. Factual correctness assesses the factual overlap between the generated answer and the ground truth answer. This assessment is based on three concepts:

1. TP (True Positive): Facts that appear in both the ground truth and the generated answer
2. FP (False Positive): Facts that appear in the generated answer but are absent in the ground truth
3. FN (False Negative): Facts that are missing in the generated answer but are appear in the ground truth

An F1 score is then used to calculate the factual correctness as:

$$\text{F1 Score} = \frac{|\text{TP}|}{(|\text{TP}| + 0.5 \times (|\text{FP}| + |\text{FN}|))} \tag{2.7}$$

Then, a weighted average of the factual similarity and semantic similarity will be taken to achieve the answer correctness score.

3
Method

This section outlines the comprehensive methodology employed in this study. The dataset consisted of research articles and technical documents in automotive safety area from Autoliv, all in PDF format. The methodology encompasses several key stages, including choosing the suitable open-source LLM and embedding model, building a RAG pipeline, and evaluating its performance. Additionally, a user-friendly GUI was developed to facilitate interaction with the models, ensuring practical applicability and ease of use. It is worth mentioning that the source code was written in Python.

3.1 Open-source LLM

Zephyr-7B-β model [17] was selected as the LLM for this project and Hugging Face was used for accessing this model. Part of the Zephyr series of language models, the **Zephyr-7B-β** represents a specialized iteration, having been fine-tuned from the pre-trained generative text model **Mistral-7B-v0.1** [18]. This model consists of 7 billion parameters and has undergone fine-tuning using a publicly accessible synthetic dataset that includes 1.4 million dialogues generated by **GPT-3.5 Turbo** covering a variety of topics. The reason for choosing this model, was its competitive performance based on MT-Bench [19] and AIpacaEval [20]. This model, with only 7 billion number of parameters, has the best performance compared to other open-source models with bigger sizes, up to 70 billion parameters [17]. This small model size made **Mistral-7B-v0.1** a low complexity model that can provide good performance while running on low to mid range devices. Different combinations of temperature, Top-p, and Top-k based of previous findings were evaluated to determine the optimal values for efficient response generation [21]. Table 3.1 illustrates the time taken by the language model to generate responses under varying parameter settings. The results indicate that the optimal configuration is a temperature of 0.7, a Top-p of 0.95, and a Top-k of 50. It is important to note that this comparison was conducted on the same dataset, yielding identical responses, thereby ensuring that the only variable affecting the outcome was the response time.

3.2 Embedding model

BAAI/bge-small-en-v1.5 [22] has been used as the text embedding model and Hugging Face was used for accessing this model. The decision for this model came from its evaluation via MTEB [23], a widely recognized evaluator in the field. With

Temperature	Top-p	Top-k	time[s]
0.6	0.95	50	720
0.7	0.95	40	510
0.7	0.9	50	490
0.7	0.95	50	360
0.7	0.95	60	1670
0.8	0.95	50	1800

Table 3.1: Sample Table with Temperature, Top-p, and Top-k

33 million parameters and 384 embedding dimensions, this model strikes a balance between resource efficiency and performance. Notably, it consumes less memory compared to other models, which is a crucial feature for our study. Despite this, it achieves a high MTEB score, indicating its effectiveness and suitability for our research objectives.

3.3 RAG pipeline

The RAG pipeline was developed in two main stages, using the **LlamaIndex** framework [24]. An illustration of the pipeline is shown in Figure 3.1. As can be seen, the data is first organized and indexed, making it ready for search queries. When a user submits a query, the index narrows down the most pertinent data to use. This selected data, along with the user's query and a specific prompt, are then sent to the LLM, which generates a response based on this context.

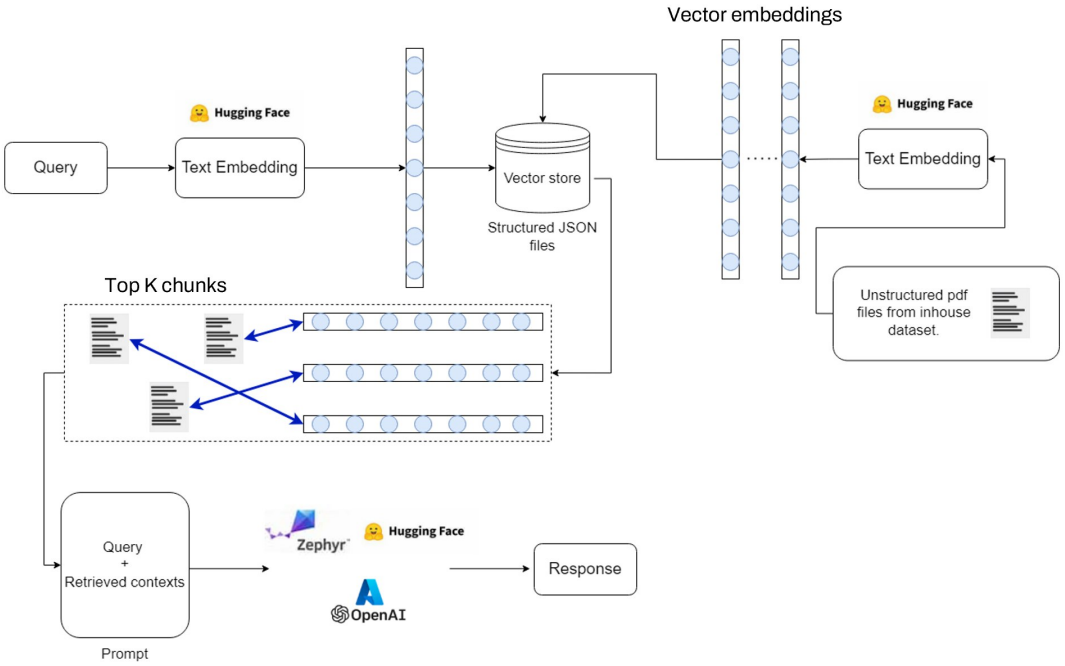


Figure 3.1: RAG pipeline

3.3.1 Data indexing

For loading the dataset, the `SimpleDirectoryReader` function, which is part of LlamaIndex library, is used. This function breaks each document to its number of pages and generates metadata for each page, containing a dictionary of annotations that can be appended to the text. This brings structure to each unstructured text file. Afterwards, the documents were parsed into nodes and split into chunks with size of 300 characters. Small chunk sizes enable the model to perform better in terms of semantic search, but with decreasing the chunk size, the memory usage increased significantly. Therefore, the chunk size of 300 was chosen to be a viable option.
In the next step, the documents were indexed. `VectorStoreIndex` was used to create vector embeddings for each node via the **BAAI/bge-base-en-v1.5** text embedding model. The indexed data was then stored on local disk in JSON format. These files include index IDs, containing the metadata for indexes.

3.3.2 Retrieval and Generation

For this part, `as_chat_engine` was used for prompting the model and enabling it to retrieve the relevant information for the query from the stored vector indices. The model was given a prompt as shown in Listing 3.1.

```
1     context_prompt=(
2     "You are an expert Q&A system that is trusted around the world
     .\n"
3     "Always answer the query using the provided context information
     ,  "
4     "and not prior knowledge.\n"
5     "Some rules to follow:\n"
6     "1. Write the name of the study you used to get your response."
7     "2. Avoid statements like 'Based on the context, ...' or "
8     "'The context information ...' or anything along "
9     "those lines."
10    "Here is some context that may be relevant:\n"
11    "-----\n"
12    "{node_context}\n"
13    "-----\n"
14    "Please write a response to the following question, using the
     above context:\n"
15    "{query_str}\n")
```

Listing 3.1: System prompt given for the query

Via the embedding model, the 20 most related contexts to the query were retrieved, enabling the model to perform semantic search. In this model, the similarity between the context and query was evaluated using average cosine similarity.
To integrate the capability for automatic insight generation into the model, a combination of a query and word chunks derived from split documents was employed. Listing 3.2 demonstrates how the initial chunk of the split document, which is `nodes[0].get_content()` in Listing 3.2, has been incorporated into the prompt.

```
1     response = chat_engine.chat('Write a short summary and result
     of the document provided below:\n'+nodes[0].get_content())
```

Listing 3.2: Query for the text summarization (automatic insight generation)

Another feature implemented in the model is history maintenance. In order to do so, the `SimpleChatStore` and `ChatMemoryBuffer` functions were used to enable saving chats and loading the previous ones. `SimpleChatStore` takes part in retrieving and formatting the history, creating a prompt that includes both the history and new input, and then using the LLM to generate a response based on this combined context. This ensures the model can provide contextually relevant responses, maintaining the flow of the conversation. `ChatMemoryBuffer` allocates an amount of memory for saving user input and the model's response. This process is done via the number of token limit which is set to 2000. This number indicates the amount of text saved in memory. If the amount of saved text exceeds this number, the new latest chats will replace the preliminary chats.

3.4 Evaluation

For evaluation of the open-source model and the RAG pipeline, the RAGAS framework [16] was used. This process was done through multiple steps. A demonstration of the pipeline is shown in Figure 3.2.

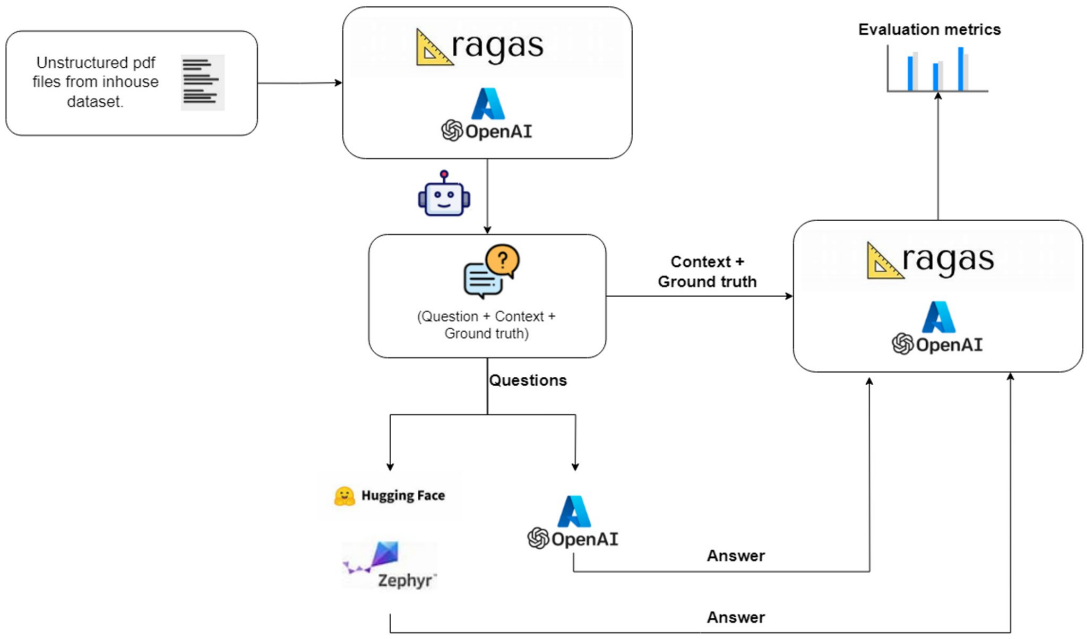


Figure 3.2: Evaluation pipeline

3.4.1 Proprietary LLM

In order to get a sense of the performance of open-source LLM compared to a proprietary one, **GPT-3.5 Turbo** was used as the proprietary model. Developed by **OpenAI**, **GPT-3.5 Turbo** has 175 billion parameters and is known for its ability to perform few-shot learning tasks [25]. The embedding model for it was **text-embedding-ada-002**, which is the default embedding model for OpenAI LLMs.

Microsoft Azure AI Machine Learning Studio was used for accessing these models through the API. The temperature was set to 0.70 and top P was set to 0.95, which were the default values for this model in the platform.

3.4.2 Synthetic dataset generation

In order to have a ground truth to compare with the results generated by the developed model, a synthetic dataset consisting of 100 questions from the in-house documents along with their ground truth answers and relevant context, were generated. This method was used in order to avoid spending significant time in manually creating the answers to each question, as well as the possible errors that a human made answer could contain. It is worth mentioning that the original test set consisted of 400 questions, but the RAGAS evaluator was only able to generate ground truths for 100 of them. For this dataset generation, **GPT-3.5 Turbo** was used both as the generator LLM for generating the synthetic dataset and the critic LLM for evaluating the results of the RAG pipeline. To generate different kinds of easy to hard questions from the dataset, three evolution types were used as follows [26].

1. Simple: These questions were simple and basic
2. Reasoning: These questions were written in a way to enhance the need of the model for reasoning in order to give the correct answer
3. Multi-context: These questions were written in a way to necessitate retrieving information from multiple related sections or chunks to give a correct answer

The distribution percentage of the generated questions included 50 % simple questions, 25 % reasoning questions and 25 % multi-context questions. A data cleaning was done afterwards in order to drop the unnecessary features, such as "evolution type", "metadata", etc., and then another feature called "answer" was added to correct the format of the input for the RAGAS evaluator. It is worth mentioning that manual inspection was performed to check the quality of the generated synthetic dataset.

Also, five questions and answers unrelated to the in-house dataset along with their ground truths were added manually, in order to see the performance of the model in answering questions that were out of context.

3.4.3 RAGAS evaluator

The generated questions were used as input for both **Zephyr-7B-β** (open-source), as well as **GPT-3.5 Turbo** (proprietary). The answers generated from these two models along with the ground truth and context generated by the RAGAS synthetic dataset, were given to the RAGAS evaluator by the `Evaluate` function. The evaluator LLM in this step was **GPT-3.5 Turbo** and the embedding model was **text-embedding-ada-002**. It is worth mentioning that some of the ground truths and generated answers were also manually checked to make sure the RAGAS evaluator was working properly.

3.5 Graphical User Interface

A graphical user interface was built using the `TKinter` library, which is a GUI toolkit. The GUI consists of three steps.

1. First, the user has the option to create a new chat or resume a previous chat.
2. If the new chat option is chosen, the next step is to choose the LLM model, which can be either **Zephyr-7B-β** or **GPT-3.5 Turbo**. After the LLM is loaded, you can start interacting with it. The GUI will receive the input, and afterwards, it will stream the generated responses using the `stream-chat` function. Another feature in the GUI is that upon uploading a new document, it splits the document into chunks of 300 characters, creates vector embeddings, and then adds those to the previous vector indices. After that, the GUI will give a short summary about the added document, indicating the ability of the model in generating automatic insights.
3. If, on the other hand, the previous chat option is chosen, the previous chats will show up and you can choose which one of them you want to resume, from the folder in which the previous chats are stored. The GUI will continue answering new questions using the LLM previously used in the selected chat.

A demonstration of the GUI can be seen in Figure 3.3.

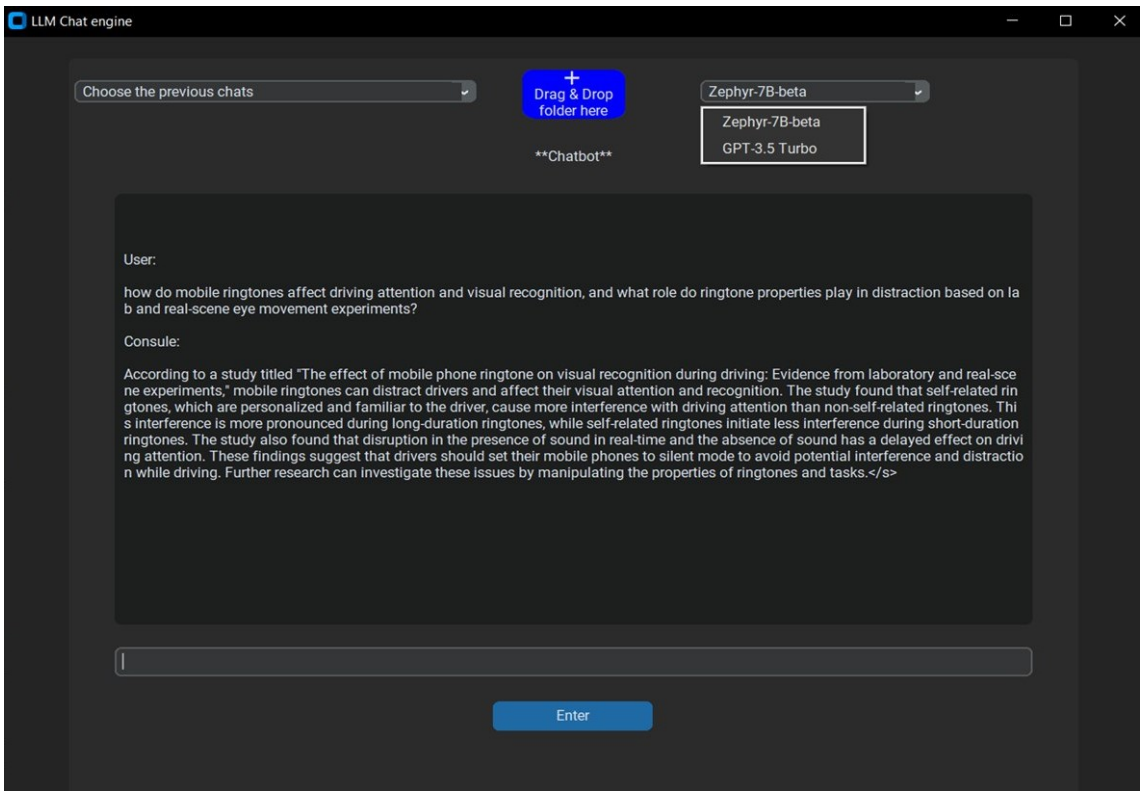


Figure 3.3: GUI

4
Results

In this chapter, the evaluation results will be presented and discussed. It is important to note that the model was deployed on a system equipped with an Intel (R) Core (TM) i7-10850H central processing unit (CPU) featuring 6 cores and 64 GB of random-access memory (RAM). With this hardware configuration, the average response speed of **Zephyr-7B-β** was 1.26 tokens per second, which is roughly one word per second.

4.1 Related questions

Results for the 100 related questions is discussed in this section. Two different approaches have been used for interpreting these results. The first approach is taken to determine which type of questions are more manageable for each LLM based on different evaluation metrics, which allows the user to use the LLM based on the preferred question type. The second approach is taken to to get an overview about the overall performance of each LLM based on the evaluation metrics only and be able to compare the overall performance of these two models.

4.1.1 Results with respect to evolution type

The results based on each evolution type for related questions will be discussed in this part.

- The faithfulness score for each evolution type is shown in Figure 4.1. It can be seen that for the reasoning evolution type **GPT-3.5 Turbo** outperforms **Zephyr-7B-β**, while having almost the same performance for the other two evolution types. This suggests that when there is a need for the LLM to perform some reasoning to achieve an answer, **GPT-3.5 Turbo** might give out more factual answers.
- The answer relevancy score for each evolution type is shown in Figure 4.2. It can be seen that for all three evolution types, **Zephyr-7B-β** outperforms **GPT-3.5 Turbo**, indicating its strength in generating related answers regardless of the question type.
- The context precision score for each evolution type is shown in Figure 4.3. It can be seen that **Zephyr-7B-β** outperforms **GPT-3.5 Turbo** in simple evolution types, while having the same performance in the other two evolution types. Therefore, when using the LLM for simple question types, **Zephyr-7B-β** is slightly a better option in terms of ranking relevant contexts appropriately

high.

- The context recall score for each evolution type is shown in Figure 4.4. It can be seen that **GPT-3.5 Turbo** outperforms **Zephyr-7B-β** in simple evolution types, while having the same performance in the other two evolution types. Hence, in simple question types, **GPT-3.5 Turbo** seems to be a slightly better option for retrieving all the relevant contexts with respect to the ground truth.
- The answer semantic similarity score for each evolution type is shown in Figure 4.5. It can be seen that in all three evolution types, **GPT-3.5 Turbo** slightly outperforms **Zephyr-7B-β**, showing its competitiveness in generating answers that are semantically similar to the ground truth.
- The answer correctness score for each evolution type is shown in Figure 4.6. It can be seen that in reasoning evolution types, **Zephyr-7B-β** outperforms **GPT-3.5 Turbo**, while having a lower score in simple and multi-context evolution types. Therefore, when focusing on reasoning question types, **Zephyr-7B-β** is a better option to choose fore achieving more accurate answers with respect to the ground truth.

4.1.2 Overall results for each metric

The overall results for related questions based on each metric is shown in Figure 4.7.

- It can be seen that **GPT-3.5 Turbo** outperforms **Zephyr-7B-β** in faithfulness, suggesting that it gives more factual answers to given questions.
- In terms of answer relevancy, **Zephyr-7B-β** seems to have considerably better performance than **GPT-3.5 Turbo**, indicating its ability to directly and appropriately address the original question.
- It can be seen that **Zephyr-7B-β** slightly outperforms **GPT-3.5 Turbo** in terms of context precision, while having slightly lower score in terms of context recall. But overall, both models perform well in these two metrics, which indicates that the RAG pipeline provides high precision in ranking relevant items in the context and retrieving the most aligned ones.
- In terms of answer semantic similarity, **GPT-3.5 Turbo** slightly outperforms **Zephyr-7B-β**. But both models have high score for this metric, which indicates their ability in performing semantic search.
- **Zephyr-7B-β** outperforms **GPT-3.5 Turbo** in answer correctness, which can indicate that it has more accuracy in generating an answer that has factual and semantic similarity with the ground truth.

4.2 Unrelated questions

The overall results for the 5 unrelated questions are shown in Figure 4.8. It can be seen that both models perform equally well in terms of faithfulness and answer relevancy. The context precision and context recall metrics are equal to zero for both models. This is due to the fact that these questions are unrelated to the in-house documents, and therefore, there are no related context in the dataset that can be retrieved by the model. In terms of answer semantic similarity, it can bee seen that **GPT-3.5 Turbo** slightly outperforms **Zephyr-7B-β**, while having a slightly

lower score in answer correctness. This indicates that **GPT-3.5 Turbo** excels in transferring the general meaning of each answer, while **Zephyr-7B-β** is better in generating precise and detailed answers.

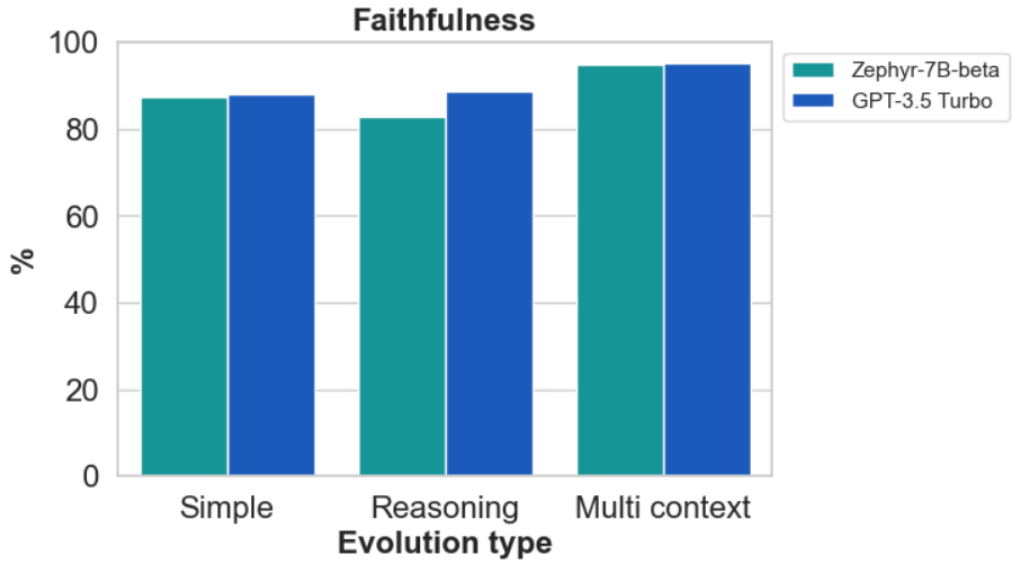


Figure 4.1: Faithfulness

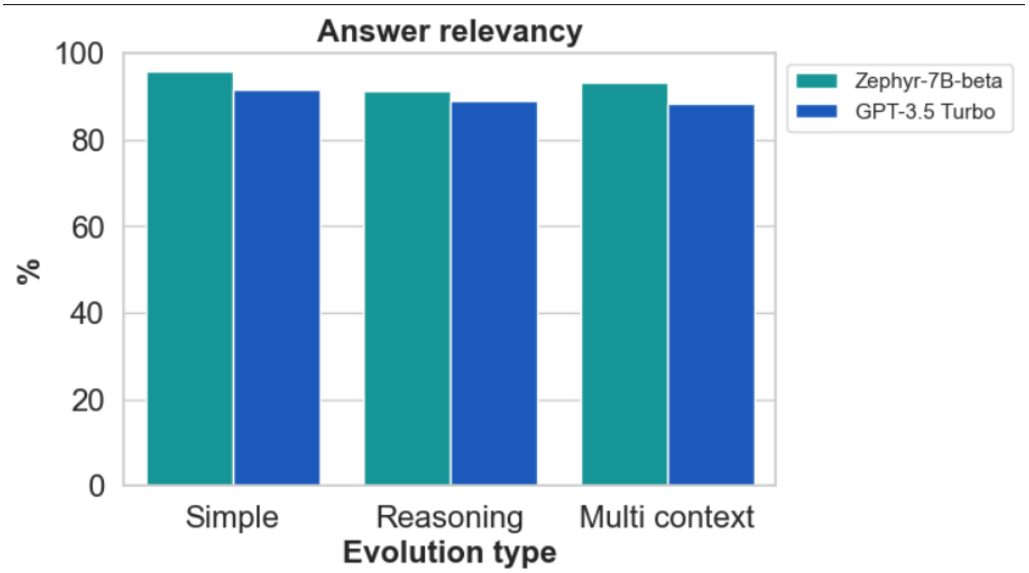


Figure 4.2: Answer relevancy

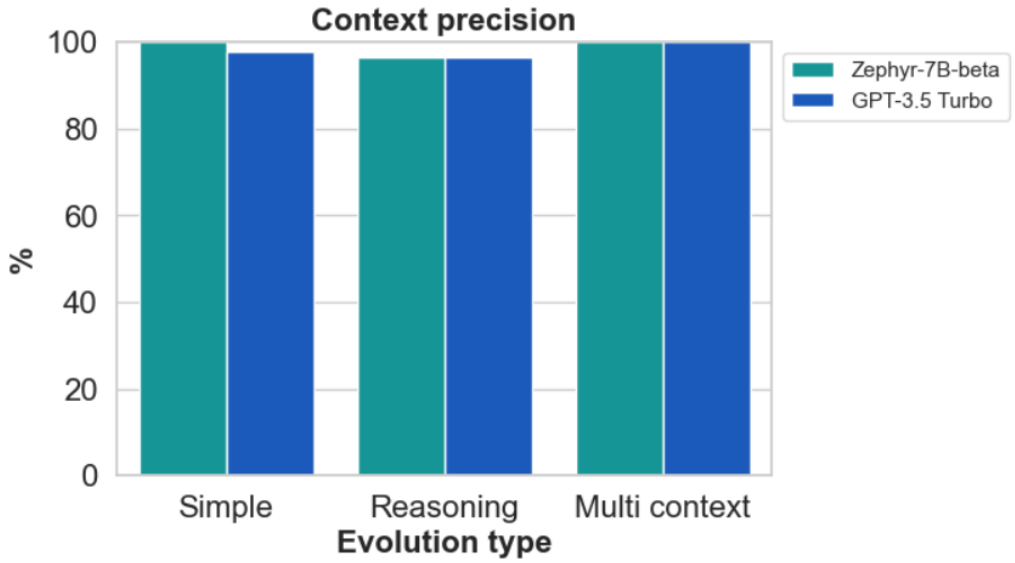


Figure 4.3: Context precision

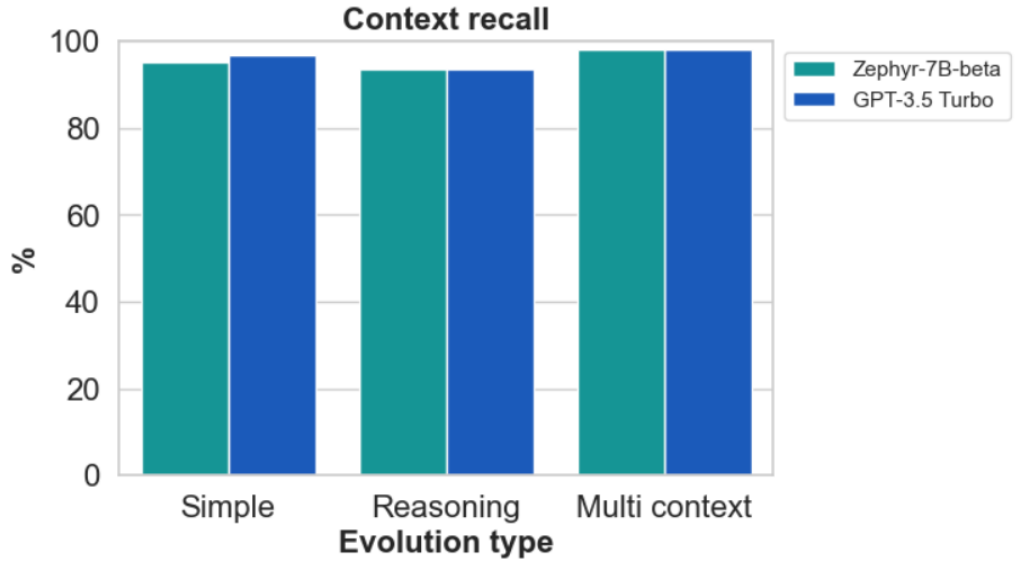


Figure 4.4: Context recall

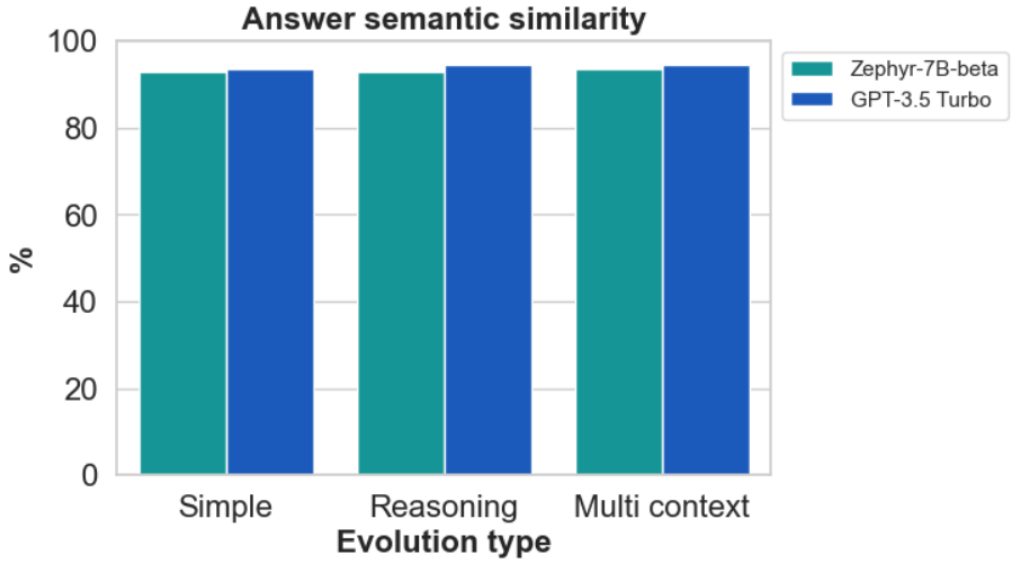


Figure 4.5: Answer semantic similarity

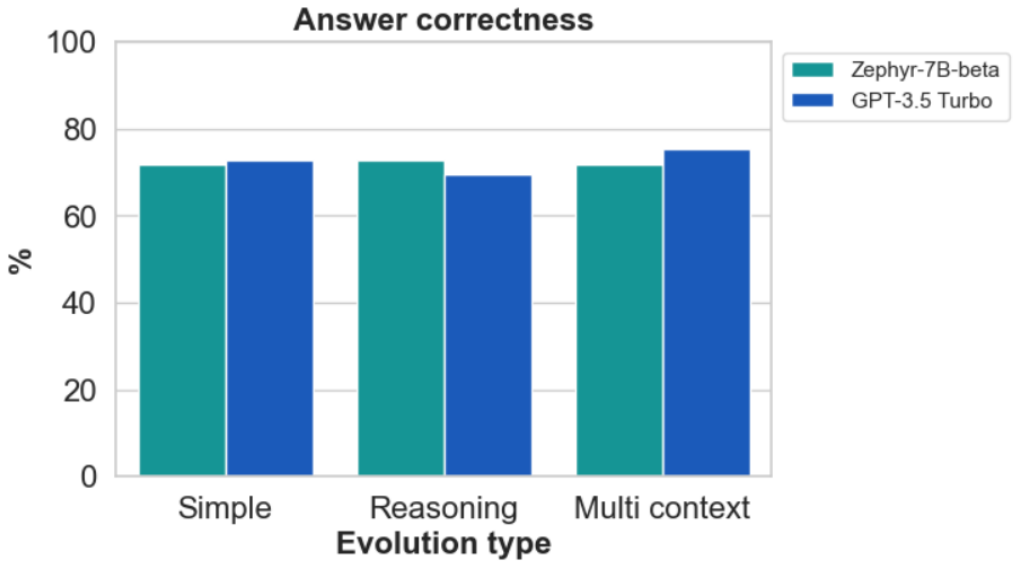


Figure 4.6: Answer correctness

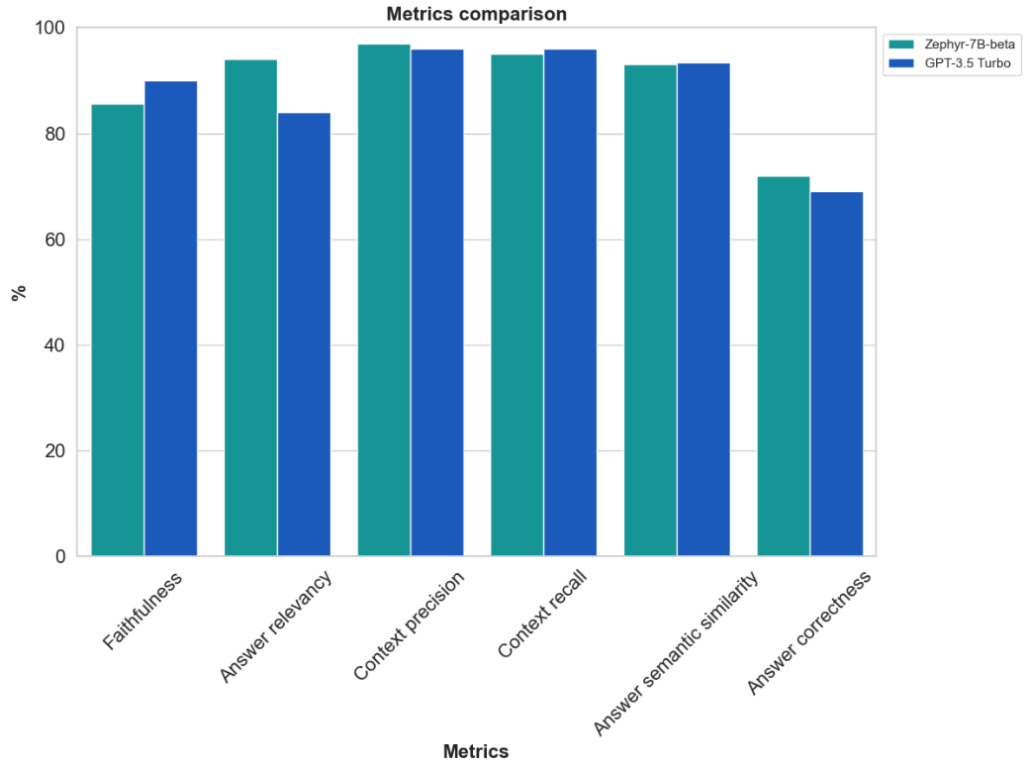


Figure 4.7: Overall results for the related questions

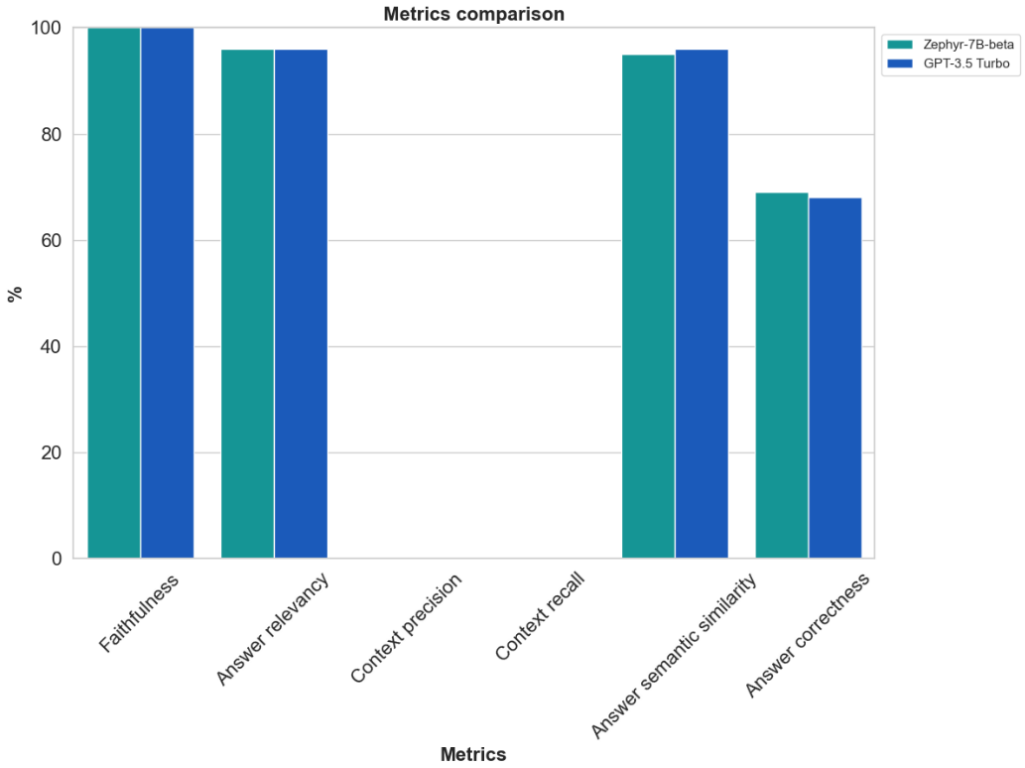


Figure 4.8: Overall results for the unrelated questions

5
Conclusion

In this chapter, the research questions are answered based on the obtained results.

1. The first research question was to see if the current state-of-the-art LLM technologies can be used to improve the research workflows in the automotive safety domain by enhancing productivity or quality of work. According to the evaluation results and high scores in metrics such as faithfulness, answer relevancy and answer correctness, it can be seen that choosing the right LLM along with proper prompting and modifications, can lead to achieving factual and accurate answers for each question. It can be seen in the results that even for complex question types such as multi-context ones that require the model to retrieve the relevant context from multiple sources and documents, the metric scores are still high. This shows the reliability of such question answering system and can yield promising performance in enhancing productivity in domain specific research. By using LLMs in a correct manner, one is able to search efficiently among documents, draw effective conclusions for each question, and get important information from one or multiple documents. This can significantly boost the productivity and quality of research work flows. Getting quick summaries from long documents to save time, drawing main takeaways from a study, and generating new ideas based on existing research, are just a few examples of productivity applications of these models.
2. The second research question was to check if exploiting zero/few shot capabilities of LLMs, for example RAG, is a viable approach. According to the results, it can be seen that the the developed RAG pipeline has achieved high scores in retrieval-related metrics, such as answer relevancy, context precision and context recall. This indicates that the RAG approach is successful in retrieving the relevant information, and therefore, eliminates the need for fine-tuning the model in order to get accurate answers. The RAG pipeline's ability to retrieve relevant information ensures that the generated responses are based on accurate and up-to-date information. It can be seen that instead of adjusting the model's parameters on the in-house dataset, the model can rely on dynamically fetched information. This approach ensures that the responses are not only accurate but also contextually relevant, reflecting the most recent and relevant data available. The RAG approach allows the model to handle a wide range of queries by using diverse data sources. This flexibility means the model can dynamically adapt to different topics without requiring specialized training for each one, which makes it a viable approach to take when using LLMs for domain-specific data.
3. The last research question was to examine if open-source LLMs provide com-

petitive performance compared to proprietary LLM models. As seen in result section, **Zephyr-7B-β** showed competitive performance compared to **GPT-3.5 Turbo**. It can be concluded that by choosing the right open-source model, we can obtain results as good as results generated by a proprietary one. This enables users to leverage the power of LLMs in their research process, without having to pay for licence fees and API keys required in proprietary models like **GPT-3.5 Turbo**. Also, open-source models are more transparent, since their source code is accessible to the public. This will give the users the opportunity to explore more with the model's architecture and its different weights in order to customize it towards their specific tasks. Moreover, open-source models come with privacy benefits of local deployment compared to the alternative closed source commercial solutions, which can make them a better option for companies using sensitive or private data, allowing them to run the model on-device with satisfactory runtime performance.

6
Future Work

The main constraint faced during this project was the limited amount of time available, since running LLMs is a time consuming process and this prevents us from exploring all the possible details. In this chapter, various areas where improvements can be made for future research will be discussed.

1. One possible modification that may improve the performance of the current model is to fine-tune it on the in-house dataset. Although the RAG pipeline has been proven to be an effective approach for using LLMs on in-house datasets with lower initial cost, there are still some benefits to fine-tuning, such as its higher accuracy in giving precise outputs [13]. Therefore, mixing fine-tuning and the RAG method can be an interesting approach to take.
2. One other area of enhancement is to explore more with LLM's parameters. One of the main benefits of using open-source LLMs was their transparency which gives everyone the opportunity to tune the model's parameters in their own way. Hence, by having enough time, it would be beneficial to make use of this opportunity and experiment more with the parameters to monitor how they can lead to achieving better results.
3. Another thing to consider for future research, would be to use vector databases as the vector store in the RAG pipeline. In this research, the vector embeddings has been stored on the local computer, which made the process very time-consuming. By using vector databases as an alternative, one would be able to decrease the processing time and improve the semantic search ability [27]. Thus, it would be a good idea to try them out and see how they can affect the results of this project.
4. One other enhancement could be to scale up the synthetic dataset size to get a more accurate picture of the performance.
5. Lastly, instead of using **GPT-3.5 Turbo** as the proprietary model for comparison, more advanced models can be used, such as **GPT-4** and **GPT-4o**. This will alow one to get a better sense of **Zephyr-7B-β**'s competence compared to proprietary models. Moreover, as the critic model used on the evaluation part, more advanced models such as **GPT-4** or **GPT-4o** can be used for getting more accurate evaluation results.

Bibliography

[1] E. M. Bender, T. Gebru, A. McMillan-Major, and S. Shmitchell, "On the dangers of stochastic parrots: Can language models be too big?" in *Proceedings of the 2021 ACM Conference on Fairness, Accountability, and Transparency*, 2021, pp. 610–623.

[2] N. T. Heffernan and C. L. Heffernan, "The assistments ecosystem: Building a platform that brings scientists and teachers together for minimally invasive research on human learning and teaching," *International Journal of Artificial Intelligence in Education*, vol. 24, no. 4, pp. 470–497, 2014.

[3] H. Naveed, A. U. Khan, S. Qiu, M. Saqib, S. Anwar, M. Usman, N. Akhtar, N. Barnes, and A. Mian, "A comprehensive overview of large language models," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2307.06435

[4] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," *arXiv.org*, 2017. [Online]. Available: https://arxiv.org/abs/1706.03762

[5] Z. Li, X. Zhang, Y. Zhang, D. Long, P. Xie, and M. Zhang, "Towards general text embeddings with multi-stage contrastive learning," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2308.03281

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

[7] J. Devlin, M. W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint*, 2018.

[8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[9] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," in *Advances in Neural Information Processing Systems*, vol. 33, 2020, pp. 1877–1901.

[10] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," *Journal of Machine Learning Research*, vol. 21, no. 140, pp. 1–67, 2020.

[11] P. Sahoo, A. K. Singh, S. Saha, V. Jain, S. Mondal, and A. Chadha, "A systematic survey of prompt engineering in large language models: Techniques and applications," *arXiv.org*, 2024. [Online]. Available: https://arxiv.org/abs/2402.07927

[12] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell *et al.*, "Language models are few-shot learners," *arXiv.org*, 2020. [Online]. Available: https://arxiv.org/abs/2005.14165

[13] A. Balaguer, V. Benara, R. L. De Freitas Cunha, R. De M Estevão Filho, T. Hendry, D. Holstein, J. Marsman, N. Mecklenburg, S. Malvar, L. O. Nunes *et al.*, "Rag vs fine-tuning: Pipelines, tradeoffs, and a case study on agriculture," *arXiv.org*, 2024. [Online]. Available: https://arxiv.org/abs/2401.08406

[14] P. Lewis, E. Perez, A. Piktus, F. Petroni, V. Karpukhin, N. Goyal, H. Küttler, M. Lewis, W. Yih, T. Rocktäschel *et al.*, "Retrieval-augmented generation for knowledge-intensive nlp tasks," *arXiv.org*, 2020. [Online]. Available: https://arxiv.org/abs/2005.11401

[15] 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. [Online]. Available: https://arxiv.org/abs/2312.10997

[16] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, "Ragas: Automated evaluation of retrieval augmented generation," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2309.15217

[17] L. Tunstall, E. Beeching, N. Lambert, N. Rajani, K. Rasul, Y. Belkada, S. Huang, V. W. Leandro, C. Fourrier, N. Habib *et al.*, "Zephyr: Direct distillation of lm alignment," *arXiv (Cornell University)*, 2023.

[18] A. Q. Jiang, A. Sablayrolles, A. Mensch, C. Bamford, D. S. Chaplot, D. De Las Casas, F. Bressand, G. Lengyel, G. Lample, L. Saulnier *et al.*, "Mistral 7b," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2310.06825

[19] L. Zheng, W. Chiang, Y. Sheng, S. Zhuang, Z. Wu, Y. Zhuang, Z. Lin, Z. Li, D. Li, E. P. Xing *et al.*, "Judging llm-as-a-judge with mt-bench and chatbot arena," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2306.05685

[20] Tatsu-Lab. Github - tatsu-lab/alpaca-eval: An automatic evaluator for instruction-following language models. human-validated, high-quality, cheap, and fast. [Online]. Available: https://github.com/tatsu-lab/alpaca_eval

[21] A. Holtzman, J. Buys, L. Du, M. Forbes, and Y. Choi, "The curious case of neural text degeneration," 2020. [Online]. Available: https://arxiv.org/abs/1904.09751

[22] S. Xiao, Z. Liu, P. Zhang, and N. Muennighoff, "C-pack: Packaged resources to advance general chinese embedding," 2023.

[23] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, "Mteb: Massive text embedding benchmark," *arXiv.org*, 2022. [Online]. Available: https://arxiv.org/abs/2210.07316

[24] J. Liu. Llamaindex. [Online]. Available: https://github.com/jerryjliu/llama_index

[25] W. X. Zhao, K. Zhou, J. Li, T. Tang, X. Wang, Y. Hou, Y. Min, B. Zhang, J. Zhang, Z. Dong *et al.*, "A survey of large language models," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2303.18223

[26] C. Xu, Q. Sun, K. Zheng, X. Geng, P. Zhao, J. Feng, C. Tao, and D. Jiang, "Wizardlm: Empowering large language models to

follow complex instructions," *arXiv.org*, 2023. [Online]. Available: https://arxiv.org/abs/2304.12244

[27] Z. Jing, Y. Su, Y. Han, B. Yuan, H. Xu, C. Liu, K. Chen, and M. Zhang, "When large language models meet vector databases: A survey," *arXiv.org*, 2024. [Online]. Available: https://arxiv.org/abs/2402.01763

DEPARTMENT OF Electrical Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY