



CHALMERS
UNIVERSITY OF TECHNOLOGY



UNIVERSITY OF GOTHENBURG

Generative AI Applied to Technology Strategy

Using Open-source Large Language Models In an Agentic
Way

Master's thesis in Computer science and engineering

Elias Sörstrand

Junchao Wang

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

MASTER'S THESIS 2024

Generative AI Applied to Technology Strategy

Using Open-source Large Language Models In an Agentic Way

Elias Sörstrand
Junchao Wang



UNIVERSITY OF
GOTHENBURG



CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Computer Science and Engineering
CHALMERS UNIVERSITY OF TECHNOLOGY
UNIVERSITY OF GOTHENBURG
Gothenburg, Sweden 2024

Generative AI Applied to Technology Strategy
Using Open-source Large Language Models In an Agentic Way
Elias Sörstrand
Junchao Wang

© Elias Sörstrand & Junchao Wang, 2024.

Supervisor: Morteza Haghiri Chehreghani, Data Science and AI
Advisor: Masoom Kumar, Volvo Trucks
Examiner: Peter Damaschke, Data Science and AI

Master's Thesis 2024
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Typeset in L^AT_EX
Gothenburg, Sweden 2024

Generative AI Applied to Technology Strategy
Using Open-source Large Language Models In an Agentic Way
Elias Sörstrand
Junchao Wang
Department of Computer Science and Engineering
Chalmers University of Technology and University of Gothenburg

Abstract

This thesis explores the application of generative AI in technology strategy, specifically through the use of open-source Large Language Models (LLMs) to develop an agent capable of understanding documents and providing insightful analysis.

The research focuses on adapting existing LLMs to automate the extraction, synthesis, and analysis of external data from diverse sources. By implementing techniques such as Retrieval-Augmented Generation (RAG), we aim to enhance the accuracy and reliability of AI-generated content, mitigating issues like hallucination.

The project involves the customization of LLMs for strategic decision-making, the development of a user-friendly interface, and rigorous evaluation to ensure operational effectiveness. Our findings indicate that integrating generative AI with strategic planning processes can significantly improve decision-making efficiency and accuracy, offering valuable insights for the electrification of road transportation and beyond.

Keywords: Artificial intelligence, AI, generative AI, generative artificial intelligence, genAI, machine learning, language agents, Deep learning, Natural language processing, NLP.

Acknowledgements

I would like to thank our supervisor, Morteza Haghiri Chehreghani, from Chalmers University of Technology, for his availability whenever we had questions or required general guidance on our academic work.

I would also like to thank everyone from Volvo Group who made this thesis possible. I would like to extend a special thanks to our advisor, Masoom Kumar, for all his time on this project. He helped us with reviewing the thesis reports, answering questions, making plans, and much more. Additionally, he also constructed all the evaluation questions and answers which helped us enormously and made it possible for us to focus on the content of the thesis. I would like to express my gratitude to Parthav Desai for his expertise in our domain, for providing valuable ideas, and for addressing our hydrogen-related questions. My thanks also go to Ruikun Deng for helping us with the setup and any problems that arose with the AI framework at Volvo, as well as our manager, Behrooz Razaznejad, for believing in this thesis.

Additionally, I would like to express my gratitude to my thesis partner, Junchao Wang, for his collaboration on this project. It has been an enjoyable journey getting to know you during our master's program at Chalmers.

Finally, I would like to thank my friends and family that has accompanied me, not only through this thesis but also my entire Chalmers period. They have all been there, supporting me through periods of difficult exams. Additionally, I would want to extend a special thanks to my girlfriend who has patiently listened to me talk about everything from math and programming, even though she does not understand any of it.

Thank you :)

Elias Sörstrand, Gothenburg, 2024-06-19

We would like to express our deepest gratitude to Volvo Group and Chalmers University of Technology for providing us with this project and opportunity.

Our heartfelt thanks go to our supervisor, Morteza Haghiri Chehreghani, who has consistently guided us through weekly meetings, offering both academic insights and encouragement. His expertise has been crucial to our progress.

We are also profoundly grateful to our advisor at Volvo, Masoom Kumar. He has provided us with numerous suggestions and feedback, along with opportunities to present and discuss our work internally, ensuring that our implementation is well-grounded in the industrial context.

Thanks to my family and friends for their unwavering support and encouragement throughout this journey. Their belief in us has been a constant source of motivation. Thanks to the precious bonus level in my life game.

Lastly, I want to thank my exceptional thesis partner, Elias Sörstrand. Meeting Elias at Chalmers has been a memorable experience. His sincerity, hard work, and kindness have made him an ideal partner. Together, we have completed many assignments and ultimately this significant thesis project, which has been an unforgettable and valuable journey.

Junchao Wang, Gothenburg, 2024-06-19



Contents

List of Figures	xv
------------------------	-----------

List of Tables	xvii
-----------------------	-------------

1 Introduction	1
1.1 Research Question	2
1.2 Task Description	2
1.2.1 Customization of LLMs for Strategic Decision-Making	2
1.2.2 Evaluation and Optimization	2
1.2.3 Achieving Strategic Objectives	3
1.2.4 Broader Implications	3
1.3 Related Work and Our Contribution	3
2 Theory	5
2.1 Large Language Models	5
2.1.1 In-Context learning	5
2.1.2 Hallucination in Generative AI	7
2.1.2.1 Preventions	7
2.2 Embeddings	7
2.2.1 Word Embeddings	7
2.2.2 Contextual Embeddings	8
2.2.3 Text Embeddings	9
2.2.4 InstructOR-XL	9
2.3 External Knowledge Retrieval	9
2.3.1 Naive Retrieval Augmented Generation	10
2.3.2 Parent Document Retriever	11
2.3.3 Reranking	12
2.3.4 Corrective Retrieval Augmented Generation	12
2.4 Reasoning	13
2.4.1 Types of Reasoning	13
2.4.2 Methods of Reasoning	14
2.5 Agents	14
2.5.1 Different Agents	15
2.5.1.1 ReAct	15
2.5.1.2 AutoGPT	16

2.5.1.3	Voyager	16
2.5.2	CoALA Framework	16
2.6	Evaluation	16
2.6.1	Evaluation Metrics for LLMs	17
2.6.2	Agent-Specific Evaluation	17
2.6.3	Leaderboards and Comparative Analysis	17
2.6.4	Long Context Evaluation	17
2.6.4.1	Needle In a Haystack	18
2.6.4.2	Multi-Needle In a Haystack	18
2.6.5	Retrieval Augmented Generation	19
2.6.5.1	Retrieval Evaluation	19
2.6.5.2	Generation Evaluation	19
3	Methods	21
3.1	Academic Research and Model Selection	21
3.1.1	Initial Understanding and Data Preparation	21
3.1.2	Model Selection and Customization	22
3.1.3	Integration of Academic and Industry Research	22
3.2	Vector-Based Dataset Development	23
3.2.1	Dataset Development Preparation	23
3.2.2	Utilizing InstructOR-XL	24
3.3	Retrieval-Augmented Generation	24
3.3.1	Implementation Strategy	24
3.3.2	Parent Document Retriever Implementation	25
3.4	Agent Architecture Implementation	25
3.4.1	Agent V1	25
3.4.2	Agent V2	26
3.4.3	Agent V3	26
3.4.3.1	Enhanced Architectural Workflow	27
3.4.3.2	Memory Management and Modular Interaction	28
3.4.3.3	Implementation Details	28
3.5	Evaluation Method	29
3.5.1	Prompt Format	29
3.5.2	Long Context	30
3.5.3	Retrieval-Augmented Generation	30
3.5.3.1	Answer Faithfulness	31
3.5.3.2	Answer Relevance	32
3.5.3.3	Context Relevance	32
3.5.4	Agent	33
3.5.5	Conclusion	33
4	Results	35
4.1	Prompt Format	35
4.2	Multi-Needle In a Haystack	36
4.3	Retrieval-Augmented Generation	37
4.4	Agent Comparisons	38

5	Discussion and Conclusion	39
5.1	Discussion of Theoretical Findings	39
5.1.1	Drawbacks of Retrieval-Augmented Generation	39
5.1.2	Drawbacks of Current Evaluation Methods	40
5.1.3	Challenges in Iterating Agent Architecture	40
5.2	Discussion of the Results	40
5.2.1	Prompt format	40
5.2.2	Multi-Needle In a Haystack	41
5.2.3	Retrieval-Augmented Generation	42
5.2.4	Agent results	43
5.2.5	Mixtral Model Limitation	44
5.2.5.1	Mixtral Format	45
5.2.5.2	Multiple Roles	45
5.3	Future Work	46
5.3.1	Retrieval-Augmented Generation	46
5.3.1.1	References	46
5.3.1.2	Embedding Model	47
5.3.2	Volvo Specific Terminology	47
5.3.3	Formatting	47
5.3.4	Large Multimodal Model	48
5.3.5	Additionally agent improvements	48
5.4	Conclusion	48
	Bibliography	51
A	Appendix 1	I
A.1	Prompts	I
A.1.1	Mixtral prompt format	I
A.1.2	Agent comparison judge	I
A.1.3	Agent V2	II
A.1.4	Agent V3	III
A.1.5	ReAct	VIII
A.1.6	AutoGPT	IX
A.1.7	Multi-Needle In a Haystack	XI
A.2	Retrieval-Augmented Generation	XII
A.3	Prompt Format Impact Evaluation	XII
A.3.1	LLM judge prompt	XV
A.4	Question and answer dataset	XVI
A.5	Additional results	XX
A.5.1	Agent comparison	XX

List of Figures

2.1	Example of few-shot, in-context learning in sentiment analysis, where k examples are used to predict the sentiment of one review. Figure from [17].	6
2.2	Flowchart of how a naive RAG pipeline could be structured.	10
2.3	Flowchart of how a parent document retriever pipeline could be structured.	11
3.1	Flowchart of Agent V3 describing how the agent functions.	27
4.1	Average results for the different prompt formats with LLM temperature of 0 (T0) and 1 (T1). BOS stands for <i>Beginning of string</i> and IS stands for <i>Instruction string</i>	36
4.2	Testing Mixtral 8x7B Instruct v0.1 [71] on multi-needle in a haystack test. The prompt length went from 1 000 tokens to 32 000 tokens with a step size of 1 000 tokens. Each prompt was tested 20 times to get an average over each needle. The first needle, denoted as <i>egg</i> , was always positioned at the start of the prompt, with the subsequent ones spaced equally from each other.	37
4.3	Testing Mixtral 8x7B Instruct v0.1 [71] on multi-needle in a haystack test. The prompt length went from 1 000 tokens to 10 000 tokens with a step size of 500 tokens. Each prompt was tested 20 times to get an average over each needle. The first needle, denoted as <i>egg</i> , was always positioned at the start of the prompt, with the subsequent ones spaced equally from each other.	37
4.4	An overview of the average results from three trials per dataset, except for GPT-4o, which was tested only once. The calculations dataset was not evaluated on GPT-4o. The sample standard deviation is used to show variation in responses and is represented by the vertical line in the bars.	38
5.1	Figures from <i>LLM In-Context Recall is Prompt Dependent</i> [54]. The retrieval accuracy of Mixtral 8x7B Instruct v0.1 for three different needles.	41

List of Tables

4.1	The average results for evaluating only the RAG pipeline on the questions in the two datasets. The evaluation ran three times per dataset and the sample standard deviation is used to show variation in responses.	38
A.1	One example question from the <i>Simple 1</i> dataset and answers from the agents and GPT-4o	XVI
A.2	One example question from the <i>Simple 2</i> dataset and answers from the agents and GPT-4o. Wrong answers are shown in <i>italic</i>	XVII
A.3	One example question from the <i>Complex 1</i> dataset and answers from the agents and GPT-4o. Wrong answers are shown in <i>italic</i>	XVIII
A.4	One example question from the <i>Complex 2</i> dataset and answers from the agents and GPT-4o. Wrong answers are shown in <i>italic</i>	XIX
A.5	One example question from the <i>Calculation</i> dataset and answers from the agents. Wrong answers are shown in <i>italic</i>	XIX
A.6	Average results for the different prompt formats with LLM temperature of 0 (T0) and 1 (T1). BOS stands for <i>Begining of string</i> and IS stands for <i>Instruction string</i>	XX
A.7	An overview of the average results from three trials per dataset, except for GPT-4o, which was tested only once. The calculations dataset was not evaluated on GPT-4o. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.	XX
A.8	A detailed view of the average results from three trials on the <i>Simple 1</i> dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.	XXI
A.9	A detailed view of the average results from three trials on the <i>Simple 2</i> dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.	XXI
A.10	A detailed view of the average results from three trials on the <i>Complex 1</i> dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.	XXI

A.11	A detailed view of the average results from three trials on the <i>Complex 2</i> dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.	XXII
A.12	A detailed view of the average results from three trials on the <i>Calculation</i> dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.	XXII

1

Introduction

This thesis, a collaboration between Chalmers University of Technology and Volvo Group, explored the role of large language models in aiding decision-making for the electrification of road transport, a crucial step towards achieving a carbon-neutral world. The electrification of road transportation is not only central to Europe's Green Deal [1] but also integral to global efforts to reduce carbon emissions. In 2018, the transportation sector accounted for approximately 20% of the annual CO_2 emissions worldwide [2].

At Volvo's Powertrain Strategic Development, technology strategies underpinned business decisions, particularly concerning new powertrain solutions. The powertrain in a vehicle comprises various components responsible for transmitting energy from the engine to the ground, including the engine, drive shaft, and wheels. In this dynamic landscape, artificial intelligence (AI), particularly generative AI, played a transformative role by offering data-driven insights, optimizing decision-making processes, and maintaining a competitive edge. By accurately predicting technology trends, market shifts, and customer behavior from historical data, AI enabled organizations to identify and capitalize on market opportunities and technological gaps.

However, the development of technology strategies was fraught with challenges, primarily the analysis of vast amounts of data to identify non-obvious patterns and effectively close technological gaps. Generative AI emerged as a potential tool to navigate these challenges. Its proficiency in processing both structured and unstructured data, identifying trends, patterns, and correlations, and assessing and mitigating risks made it an ideal framework to support technology strategy formulation. By harnessing the power of generative AI, organizations are able to make more informed, data-driven decisions, leading to enhanced strategies.

This thesis adapted existing Large Language Models (LLMs) to automate the extraction, synthesis, and analysis of external data from diverse sources. To facilitate this task, an agentic framework was developed, referred to as a language agent. Additionally, this language agent was designed to predict the hydrogen (H₂) price in 2030, demonstrating its adaptability to other domain fields related to hydrogen energy and beyond.

1.1 Research Question

In this thesis, we answered the following research questions:

- What characteristics of an open-source large language model (LLM) need to be considered when building an agent for a specific domain?
- How can an LLM be integrated into an agentic workflow, and what practical factors must be considered?
- Specifically, how can we implement an agent that comprehends domain knowledge and predicts the hydrogen price in 2030?

1.2 Task Description

This section outlines the concrete steps to be taken to realize the objectives detailed earlier, focusing on the customization and application of generative AI to enhance technology strategy development in the context of electrification of road transportation.

1.2.1 Customization of LLMs for Strategic Decision-Making

The primary task involves adapting existing LLMs to suit the specific needs of Volvo Group's strategic decision-making framework. This includes:

- **Data Integration:** Implementing mechanisms to efficiently extract, synthesize, and analyze data from diverse external sources, ensuring the model can handle large datasets with varying structures.
- **Knowledge Retrieval:** Enhancing the model's ability to retrieve relevant information through advanced techniques such as Retrieval Augmented Generation (RAG), focusing on improving the accuracy and relevance of the retrieved data.

1.2.2 Evaluation and Optimization

A rigorous evaluation framework is essential to validate the effectiveness of the implemented solutions and to refine them based on empirical data:

- **Performance Evaluation:** Conducting tests to assess the accuracy, speed, and reliability of the AI model in producing strategy recommendations. This will involve comparing the AI-generated outcomes with expert assessments.
- **Iterative Improvement:** Utilizing feedback from initial deployments to enhance the model's performance, focusing on reducing errors and improving the models handling of complex queries.

1.2.3 Achieving Strategic Objectives

The ultimate task is to ensure that the AI-enhanced tools developed not only meet the technical specifications but also align closely with Volvos strategic objectives:

- **Enhanced Decision-Making:** Demonstrating that the use of generative AI contributes to more informed and timely decision-making within the realm of electrification.
- **Scalability and Flexibility:** Establishing that the solutions are scalable and adaptable to different strategic contexts within Volvo, ensures that the investment in AI yields long-term benefits.

1.2.4 Broader Implications

While the focus is on the electrification of road transportation, the methodologies and insights gained from this thesis are intended to apply to broader areas that require strategic planning and advanced data analysis. This task explores the potential for these AI-enhanced tools to be adapted for other strategic domains across the industry.

1.3 Related Work and Our Contribution

With the rise of artificial intelligence, particularly through the development of large language models, has ushered in innovative applications in the realm of conversational agents. Despite the surge in technological advancements, there remains a relatively sparse body of research specifically targeting language agents that facilitate user conversation. An exception is the work presented in ReAct [3], which advanced the domain by structuring interaction through iterative reasoning and actionable responses. ReAct employs a set of defined actions, namely *search*, *lookup* on Wikipedia, and a *finish* action to conclude interactions, providing a structured framework for dialogue.

Building upon the foundational ideas of ReAct, AutoGPT [4] represents a significant evolution in this field. Unlike its predecessor, AutoGPT distinguishes between 'thoughts' and 'reasoning' as separate components of the decision-making process, enhancing the complexity and depth of agent interactions. Furthermore, AutoGPT introduces the capability to manage multiple objectives, devising and evaluating strategies for each, thereby enriching the agent's operational versatility.

The majority of research on language agents has been directed towards their applications in robotics or virtual environments. For instance, the paper on Generative Agents [5] explores the dynamics of multiple agents governed by a large language model in a simulated world, engaging in human-like behaviors such as social interactions and daily activities. Another intriguing application is presented in Voyager [6], where the agent navigates the Minecraft universe, equipped with a "skill library" that enables the acquisition and application of new capabilities like crafting tools and combating threats.

Despite the prevalent use of models like GPT [7]–[10] in these studies, their application often surpasses the capabilities of open-source models. The performance leap from GPT-3 [9] to GPT-4 [10], as detailed in Voyager [6], underscores the significant enhancements in processing and task execution enabled by advancements in model architectures.

However, the field lacks a comprehensive exploration of agents designed specifically for conversational applications outside of controlled or task-specific environments. Most agents are tailored for predefined tasks without the necessity for ongoing human interaction, limiting their applicability in dynamic, real-world scenarios.

Our contribution to this emerging field is the development of an agent that leverages open-source technologies to offer robust conversational capabilities. This agent is designed not only to interact seamlessly in user dialogues but also to adapt and respond intelligently to a broad spectrum of conversational cues, making it a versatile tool for both casual and more structured interactions.

2

Theory

Having established the context and the critical need for advanced analytical tools in the face of evolving strategic challenges within the electrification of road transportation, we now turn to the theoretical underpinnings that will inform our approach. This section delves into the technological foundations that support our research. Specifically, it focuses on Large Language Models (LLMs), retrieval-augmented generation, and agents.

2.1 Large Language Models

Large Language Models (LLMs) are a type of artificial intelligence technology based on deep learning neural networks, designed to understand, generate, and interact with human language at scale. These models are trained on vast amounts of text data, allowing them to learn the complexities of language, including syntax, semantics, and context. LLMs like BERT (Bidirectional Encoder Representations from Transformers) [11] and the GPT (Generative Pre-trained Transformer) series [7]–[10] utilize the transformer architecture, which is distinguished by its attention mechanism that enables the model to weigh the importance of different words in a sentence, regardless of their positional order [12]. This capability allows LLMs to generate coherent and contextually relevant text based on the input they receive. The practical applications of these models are extensive, ranging from simple text completions to complex dialogue systems, content creation, and data analysis tasks, making them invaluable in automating and enhancing decision-making processes in various domains, including strategic business planning.

2.1.1 In-Context learning

In-context learning is a pivotal concept in the utilization of Large Language Models (LLMs). To understand this, it's essential to first define what a prompt is. Within the framework of LLMs, a prompt is the input text that the model receives, which often includes any preceding interaction such as questions and responses in a chatbot scenario. Essentially, the prompt serves as natural language instructions that guide the LLM's responses. Furthermore, a primer is used after the prompt to guide how the LLM should start the text generation. As an example, if the primer is "*Hi, my name is*" the LLM will start to generate text after "*is*".

The operational boundary of an LLM is defined by what is termed the “context window” or “context length”, which specifies the maximum number of tokens the model can process in one instance. This limitation is crucial because it determines the amount of information the model can consider when generating responses. For example, models like Google Gemini 1.5 [13] are known for their extensive context windows, allowing them to maintain coherent and contextually relevant outputs over longer interactions.

In-context learning leverages the prompt’s structure to teach the model new tasks without formal retraining [9]. An example of in-context learning can be seen in fields like sentiment analysis (SA), also known as opinion mining [14]. Sentiment analysis involves identifying the opinions, emotions, or attitudes expressed within texts. LLMs have demonstrated notable proficiency in classifying texts, with techniques including zero-shot and few-shot learning [15]. Zero-shot prompting, as introduced in the GPT-2 paper [8] and further developed in the GPT-3 paper [9], involves directly inputting text into the model and asking it to classify sentiment without prior specific training on sentiment analysis tasks. While effective, it often underperforms compared to few-shot prompting [9], [16]. Few-shot prompting improves on this by providing the model with N examples to analyze before making a prediction. This method not only helps the model identify hidden patterns but also enhances its predictive accuracy [16], [17].

The figure below, Figure 2.1, exemplifies in-context learning, where the LLM adapts to new information within the constraints of its pre-trained parameters, thus learning from context without the need for parameter updates.

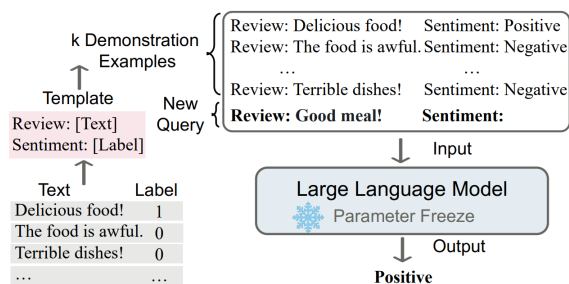


Figure 2.1: Example of few-shot, in-context learning in sentiment analysis, where k examples are used to predict the sentiment of one review. Figure from [17].

Recent studies, such as those presented in [17], have evaluated how LLMs like GPT-3 utilize in-context learning to assimilate new tasks. These studies highlight the effectiveness of LLMs in adapting their responses based on the cumulative context provided through prompts, thus pushing the boundaries of what pre-trained models can achieve without additional training. This concept of in-context learning not only enriches the model’s interaction capabilities but also underscores the potential of LLMs to handle a diverse range of tasks dynamically.

2.1.2 Hallucination in Generative AI

Hallucination in the context of generative AI and large language models (LLMs) refers to instances where the model produces text that is not grounded in the input data or factual reality. This phenomenon can lead to outputs that are plausible-sounding but incorrect or entirely fabricated. Hallucination poses significant challenges, especially in applications requiring high accuracy and reliability.

In the comprehensive survey, Ji et al. [18] explore various aspects of hallucination in natural language generation (NLG). They categorize hallucinations into different types, such as factual and linguistic hallucinations. Factual hallucinations occur when the model generates information that contradicts known facts, while linguistic hallucinations involve outputs that are syntactically or semantically flawed despite being factually accurate.

2.1.2.1 Preventions

To mitigate hallucination, Ji et al. [18] discuss various techniques such as improving training data quality, incorporating fact-checking mechanisms, and refining model architectures to better handle the generation process. Ensuring the robustness and accuracy of AI-generated text remains a critical area of research, aiming to enhance the reliability of LLMs in practical applications.

Using Retrieval-Augmented Generation (RAG) is an effective method of minimizing hallucination in generative AI models. RAG combines the strengths of large language models with external knowledge retrieval systems. By fetching relevant information from reliable databases or documents during the generation process, RAG ensures that the model's outputs are grounded in factual data. This approach significantly reduces the likelihood of the model generating incorrect or fabricated content, as it relies on verifiable sources rather than solely depending on the probabilistic predictions of the language model. Thus, RAG enhances the accuracy and reliability of AI-generated text, making it a valuable tool for mitigating hallucinations.

2.2 Embeddings

Embeddings are fundamental in enabling machine learning algorithms to process and interpret various forms of data such as text, images, and audio. By converting these data types into numerical vectors, embeddings facilitate the understanding of inner relationships and meanings within the data. This section explores different types of embeddings and their applications in natural language processing and beyond.

2.2.1 Word Embeddings

Word embeddings are foundational to modern natural language processing (NLP), providing a way to translate the semantic and syntactic nuances of words into a high-dimensional vector space. Unlike earlier representations that treated words as discrete, independent entities (such as one-hot encoding), embeddings represent

words as dense vectors where each dimension captures some aspect of the word’s meaning and words with similar meanings are positioned closely in the vector space.

The development of word embeddings marked a significant shift in text processing, moving from sparse to dense representations. The initial models like word2vec created by Mikolov et al. [19], [20] introduced two architectures for producing these embeddings: the Continuous Bag of Words (CBOW) and Skip-gram models. In CBOW, the model predicts a target word from a set of context words, while Skip-gram inverts this approach by predicting context words from a target word, which is effective in capturing a wide array of semantic relationships.

Following word2vec, other models such as GloVe (Global Vectors for Word Representation) [21] and FastText [22] were developed. GloVe combines matrix factorization and local context window techniques to leverage statistical information from a corpus by aggregating global word-word co-occurrence statistics. FastText extends word2vec’s Skip-gram model by treating each word as composed of character n-grams, allowing the embedding of subword information which is particularly useful for handling morphologically rich languages or misspelled words.

Word embeddings have proven to be incredibly useful, applicable to a wide range of tasks beyond simple word similarity. They are critical in powering more complex NLP systems including machine translation, sentiment analysis, and information retrieval systems. Their ability to capture syntactic and semantic word relationships enables these systems to perform with a greater understanding of language nuances. A hallmark of word embeddings’ effectiveness is their ability to facilitate semantic and analogical reasoning. The famous example of vector arithmetic “King - Man + Woman = Queen” [23] illustrates this capability. By manipulating vectors, it’s possible to derive relationships and analogies, providing a powerful tool for many linguistic applications.

2.2.2 Contextual Embeddings

A significant limitation of traditional word embedding models like word2vec is their inability to account for the context in which words are used [24]. These models assign a single, static vector to each word, irrespective of its varying meanings in different sentences. For instance, the word “watch” is represented by the same vector in the sentences “She likes to watch movies in her free time.” and “He wore a sleek watch on his wrist.”, despite its different uses.

To address this limitation, newer models such as ELMo [25] and BERT [11] have been developed. These models generate dynamic, context-dependent embeddings. By analyzing the entire sentence, these models produce word embeddings that reflect the varied meanings of a word based on its contextual usage. This allows each word to convey more accurate and specific semantic information depending on its application within the sentence.

2.2.3 Text Embeddings

Text embeddings, also known as sentence or document embeddings, represent higher-level textual units ranging from sentences to full documents as dense vectors. This representation is crucial for tasks that require understanding the overall context rather than individual word meanings. Methods like doc2vec [26] directly extend the principles of word2vec to larger text blocks by treating each document as a new word in the vocabulary. More approaches use transformer-based models, which utilize attention mechanisms to capture deeper semantic relationships across the text.

2.2.4 InstructOR-XL

Instruction-based Omnifarious Representations (InstructOR)-XL [27] is a jack-of-all-trades text embedding model. Built on transformer architectures, this model is designed to receive task instructions on what should be embedded and change accordingly. The instruction looks like,

Represent the {domain} {text_type} for {task_objective}:

where, {domain}, {text_type} and {task_objective} are replaced with task-specific instructions. For instance, the placeholder {domain} could represent “science”, {text_type} might be “paragraph”, and {task_objective} could stand for “clustering”. With the instruction, it is possible to use one embedding model for any task, without fine-tuning.

2.3 External Knowledge Retrieval

A significant limitation of Large Language Models (LLMs) is their inherent knowledge cutoff date, which marks the most recent point at which information was incorporated into their training dataset [28]. This cutoff can hinder the model’s relevance, particularly in fields requiring the latest data, such as strategic forecasting and real-time decision-making.

To address the challenge of outdated information, one common approach is fine-tuning, a method of transfer learning where the model is updated to perform tasks beyond its initial training [29], [30]. This process involves partial or complete retraining of the model using new data. Despite its benefits, fine-tuning presents several risks, including the potential for the model to forget previously learned information or to overfit the new data [31]. Additionally, fine-tuning requires substantial computational resources, which can be a barrier for many applications [32].

To circumvent the limitations of fine-tuning and to enhance the LLM’s access to updated information without extensive retraining, the Retrieval Augmented Generation (RAG) method has been developed [33]. Initially described in the paper “Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks” [33], RAG has evolved into a broader set of techniques aimed at integrating external knowledge into the model’s responses [34].

The RAG methodology consists of three primary components:

- **Indexing:** This step involves organizing a database in a manner that facilitates efficient searching. The database can include a wide array of documents or data points relevant to the domain in question.
- **Retrieval:** At this stage, the model performs a similarity search within the indexed database to identify and retrieve information pertinent to the current query or prompt. This allows the model to access the most relevant and recent data, overcoming the constraints of its fixed training set.
- **Generation:** Using the retrieved information, the LLM then generates a response tailored to the user’s query. This response is informed by the most current and contextually relevant data, significantly enhancing the model’s utility and accuracy.

By integrating these processes, RAG provides a dynamic framework for LLMs to access and utilize external knowledge, thereby extending their applicability and improving their performance in real-world tasks that demand up-to-date information.

2.3.1 Naive Retrieval Augmented Generation

Naive RAG represents the most straightforward implementation within the Retrieval Augmented Generation (RAG) framework [34]. This method efficiently integrates external knowledge into Large Language Models (LLMs) by leveraging a streamlined retrieval process.

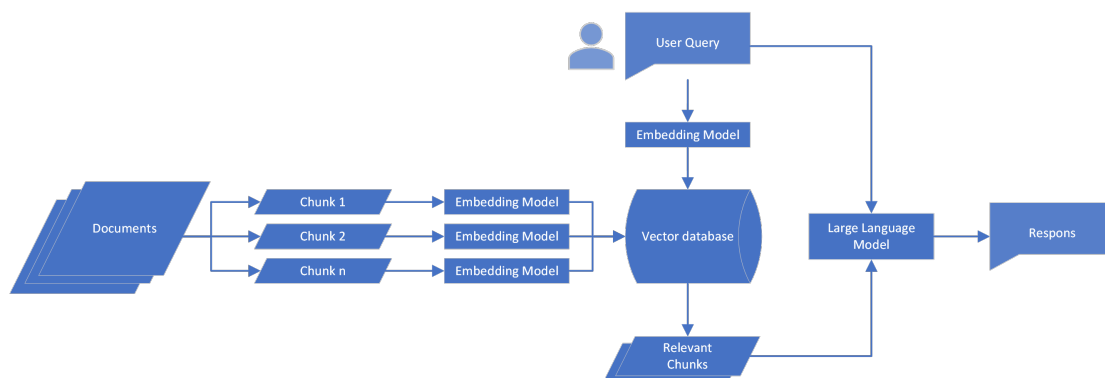


Figure 2.2: Flowchart of how a naive RAG pipeline could be structured.

The process begins with the creation of a text database composed of documents relevant to the application’s focus, as seen in Figure 2.2. This database is subsequently segmented into smaller chunks, each of which is embedded using a text embedding model. The embedding transforms these chunks into a vector space format that facilitates rapid and efficient retrieval. This setup phase is crucial and is performed only once, laying the foundation for all future queries.

When a user submits a query, it is converted into an embedding using the same model that performed the database indexing, ensuring that the query and the database

chunks are represented in the same vector space. The system then calculates a similarity score between the query embedding and each indexed chunk, typically using cosine similarity to assess the alignment of their meanings.

The chunks with the highest similarity scores are selected, up to a limit defined by the hyperparameter K , which determines the number of top results to retrieve. This selection process is critical as it filters the database to find the most relevant information for the query at hand.

The top K chunks, along with the user query, are then amalgamated into a comprehensive prompt. This enriched prompt is fed into the large language model, which processes the combined input to generate a response. The LLM utilizes the information from the retrieved chunks to produce an answer that is not only relevant but also informed by the most pertinent and up-to-date data.

By dynamically pulling the most relevant information from a structured database, naive RAG allows LLMs to deliver responses that are both contextually appropriate and enriched with the latest knowledge, making it an invaluable tool in applications where current information is crucial, such as powertrain technology domain knowledge in our project.

2.3.2 Parent Document Retriever

The Parent Document Retriever is a more advanced RAG method, compared to the previous one, and categorized under naive RAG as outlined in [34] and specifically detailed in [35]. This approach modifies the conventional indexing and retrieval processes to enhance precision without sacrificing performance.

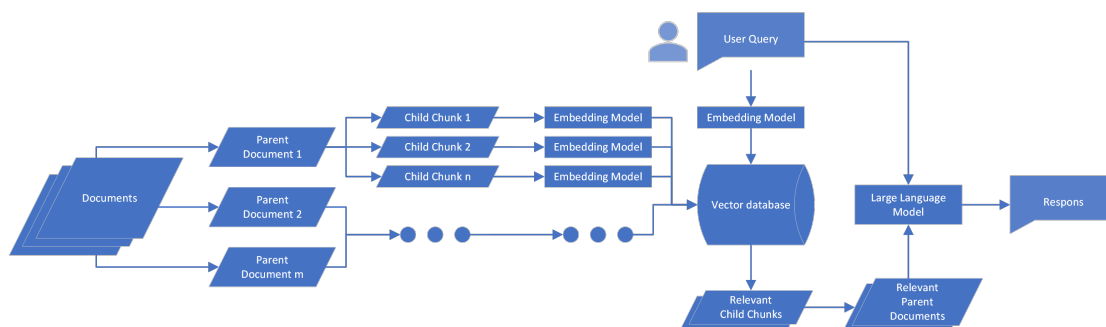


Figure 2.3: Flowchart of how a parent document retriever pipeline could be structured.

Unlike the single-level document splitting described in Section 2.3.1, the Parent Document Retriever introduces a two-tiered splitting mechanism, as seen in Figure 2.3. Initially, large documents are divided into what are termed “parent documents”. For instance, a multi-paged document might be divided into individual pages, each considered a parent document. These parent documents are subsequently subdivided into smaller segments referred to as “child chunks”.

Each child chunk is embedded using a text embedding model that encodes the semantic meanings into the vectors. This two-level hierarchical structure ensures that the indexing captures a more granular level of detail, potentially increasing the accuracy and relevance of the retrieved information. Importantly, each child chunk retains a reference back to its parent document, preserving the context of the information.

During the retrieval phase, the process diverges from typical RAG methods by conducting the similarity search at the level of the child chunks rather than the parent documents. However, instead of returning the child chunks directly, the method retrieves the entire parent document of the highest-ranking child chunks. This strategy enhances the search’s precision by focusing on the most relevant and detailed part of text while still delivering comprehensive information by returning the full parent document.

The advantage of the Parent Document Retriever lies in its ability to refine the granularity of the search process. By focusing on smaller segments of text for indexing and retrieval, this method ensures that the semantic search is highly precise. Meanwhile, by returning the parent documents, it maintains the breadth necessary for the generation phase, ensuring that the responses are informative and contextually rich without compromising on the depth or relevance of the information provided.

2.3.3 Reranking

Reranking is an essential enhancement within the Retrieval Augmented Generation (RAG) framework, designed to improve the precision of responses generated by Large Language Models (LLMs). This process refines the selection of documents or chunks initially retrieved based on their similarity to the query, ensuring that only the most relevant information is used in the final response generation.

After the initial retrieval phase, which fetches potentially relevant data, reranking applies a more detailed evaluation to these items. This involves using advanced models that assess the relevance and contextual alignment of each item with the query. Techniques such as machine learning classifiers, cross-attention mechanisms, and semantic coherence evaluation are commonly used to determine the quality and fit of the retrieved information.

However, the addition of reranking can introduce significant computational demands, potentially affecting the timeliness of the response. Balancing the depth of content analysis with the necessity for quick response times is a pivotal challenge in deploying reranking effectively within the RAG framework.

2.3.4 Corrective Retrieval Augmented Generation

Corrective Retrieval Augmented Generation (CRAG) is an evolution within the Retrieval Augmented Generation (RAG) framework, designed to enhance the accuracy and reliability of responses generated by Large Language Models (LLMs) [36]. CRAG introduces a corrective step following the initial retrieval phase to scrutinize

and adjust the fetched content for potential errors or misinformation before the final response is generated.

In practice, CRAG works by cross-referencing the initially retrieved information against trusted databases or employing advanced natural language processing tools to assess the coherence and factual accuracy of the content. This corrective process not only ensures the relevancy and correctness of the information but also updates any outdated or incorrect data retrieved from the static databases used in the initial phase.

While CRAG significantly improves the trustworthiness of automated responses, it also increases the computational demands of the system, potentially leading to higher response times. Balancing the depth of the corrective analysis with the need for efficiency is a critical challenge, particularly in interactive systems where response latency can impact user experience.

In summary, while RAG significantly enhances the functionality of LLMs by incorporating external knowledge into the response generation process, it also introduces several complexities and limitations that must be carefully managed. Addressing these drawbacks is crucial for optimizing the performance and applicability of RAG systems across various domains.

2.4 Reasoning

Reasoning capabilities are essential for advanced cognitive processing, enabling Large Language Models (LLMs) to perform tasks that go beyond mere retrieval and association, to actually understanding and manipulating concepts in a way that mimics human cognitive abilities. This section explores different kinds of reasoning and the methods employed to achieve them.

2.4.1 Types of Reasoning

- **Arithmetic Reasoning:** Involves the ability to perform mathematical calculations and solve quantitative problems, which is crucial for tasks that require numerical understanding and manipulation.
- **Commonsense Reasoning:** Pertains to the model's ability to apply everyday knowledge that an average human is expected to know, thereby making decisions that seem intuitively obvious to most people.
- **Logical Reasoning:** Entails drawing inferences based on logical deductions, crucial for tasks that require rigorous, step-by-step analysis of problems or scenarios.
- **Symbolic Reasoning:** Deals with the manipulation of symbols and abstract entities, allowing models to handle tasks that require understanding and using symbolic information effectively.

- **Multimodal Reasoning:** Involves integrating and reasoning across multiple types of data inputs (such as text, images, and sounds), essential for tasks that require comprehensive understanding across different sensory modalities.

2.4.2 Methods of Reasoning

- **Chain of Thought (CoT):** This method involves guiding the model to articulate intermediate steps towards reaching a conclusion, mimicking how humans often solve problems by breaking them down into smaller, manageable parts [37].
- **Auto Chain of Thought (ACoT):** An automated extension of CoT, where the model is trained to generate and follow its own chain of thoughts without external prompts, enhancing its autonomy in problem-solving [38].
- **Tree of Thoughts (ToT):** Expands on CoT by branching out into multiple potential solutions or pathways before selecting the most appropriate one, similar to decision tree analysis [39].
- **Reason via Plan (RAP):** Focuses on developing a systematic plan based on reasoned analysis and then executing it to solve complex problems, especially useful in strategic or long-term decision-making [40].
- **Self-refine:** Encourages the model to evaluate its own reasoning and conclusions, potentially revising its strategies based on self-assessment to improve accuracy and reliability [41].
- **Plan and Solve:** A method where the model first constructs a detailed plan or framework for solving a problem and then systematically executes the solution, ensuring thoroughness and precision in tackling complex issues [42].

Each of these reasoning types and methodologies plays a crucial role in enhancing the cognitive capabilities of LLMs, enabling them to perform tasks with a level of sophistication and accuracy that closely mirrors human reasoning processes.

2.5 Agents

Agents can take various shapes and forms, and there are no universally accepted definitions of their structure. We can illustrate the construction of an assistant agent with an example:

A user wants to know the weather in Gothenburg and asks, “*What is the weather in Gothenburg today?*”. In response, the agent will perform some steps:

- **Plan:** The LLM creates a plan to address the user’s query.
 - Example: *Use Google, yr.no, or Microsoft to find the weather in Gothenburg.*
- **Decide:** The agent decides which website to use to retrieve the information.

- This decision can be made with or without the assistance of an LLM.
- **Act:** The agent performs the action of visiting the chosen website and checking the weather.
 - This step is often executed using a hard-coded tool.
- **Evaluate:** The agent evaluates the received information to determine its accuracy and relevance.
- **Store:** If the information is deemed good enough, it is stored in long-term memory.
- **Respond:** Finally, an LLM constructs an answer to the user’s question based on the retrieved information.

In summary, an agent can be viewed as a system equipped with an LLM capable of reasoning, following a plan, utilizing various actions, and storing information in its memory.

2.5.1 Different Agents

Some other agents that have been developed are ReAct [3], AutoGPT [4], and Nvidia Vogeger [6]. One common thread across all these agents is their attempt to simulate reinforcement learning, where the LLM “learns” over multiple iterations.

2.5.1.1 ReAct

ReAct, was one of the early developed agent-like structures. It combined reasoning and acting and achieved new state-of-the-art results [3].

The main concept behind ReAct is similar to reinforcement learning, where an agent tries to learn in a new environment. In reinforcement learning, an agent will interact with an environment over time. At the time t , the agent will be in state $s_t \in \mathcal{S}$ and need to take an action $a_t \in \mathcal{A}$, where \mathcal{S} and \mathcal{A} is the valid state and action space. The action a_t is decided through a policy $\pi(a_t|s_t)$, which maps an action to a state. After the action is taken, the agent will receive a reward r_t and move onto state s_{t+1} [43].

In ReAct, the action space $\hat{\mathcal{A}}$ is comprised of two smaller action spaces, \mathcal{A} and \mathcal{L} . From the ReAct paper, the first action space, \mathcal{A} , contains three actions, *Search[entity]*, *Lookup[string]* and *Finish[answer]*. The action in this action space is decided by the policy $\pi(a_t|c_t)$, where $c_t = (s_1, \hat{a}_1, a_1, \dots, s_{t-1}, \hat{a}_{t-1}a_{t-1}, s_t, \hat{a}_t)$ is the context of the LLM. The other action \hat{a}_t comes from the second action space \mathcal{L} , representing the language space. This language space is of unlimited size and the actions taken there, $\hat{a}_t \in \mathcal{L}$ is called a *thought* or *reasoning trace*.

The main difference between the two action spaces is that an action inside \mathcal{L} will not affect the external environment, and therefore not receive any feedback/reward, unlike \mathcal{A} . However, the action $\hat{a}_t \in \mathcal{L}$ will be used to update the LLM context c_{t+1}

as $c_{t+1} = (c_t, \hat{a}_t)$, and will in turn, be used to take the next action $a_t \in \mathcal{A}$. In short, ReAct mimics reinforcement learning as,

$$s_1 \rightarrow \hat{a}_1 \rightarrow a_1 \rightarrow s_2 \rightarrow \hat{a}_2 \rightarrow a_2 \rightarrow s_3 \rightarrow \dots,$$

but with two actions taken at each time step instead of one.

2.5.1.2 AutoGPT

AutoGPT is an advanced agent that leverages the ReAct framework to create a novel approach for handling complex tasks [4]. The key idea is to formulate a detailed plan before initiating the first reasoning step, breaking down the user’s query into multiple sub-questions. This structured decomposition enhances the agent’s ability to address each aspect methodically.

AutoGPT autonomously identifies and prioritizes tasks based on user input, ensuring critical aspects are addressed effectively. It engages in iterative cycles of reasoning and planning, refining its approach based on intermediate results and new data. By decomposing queries into sub-questions, AutoGPT provides detailed and contextually relevant answers. Additionally, it integrates seamlessly with external tools and data sources, enhancing its capabilities and efficiency.

2.5.1.3 Voyager

Another more complex agent is Voyager [6]. Briefly, it is an agent developed to play Minecraft and achieved new state-of-the-art results in exploration and efficiency. It uses GPT-3.5 and GPT-4 [10] as a brain and then communicates through text to the game and does not require any fine-tuning.

2.5.2 CoALA Framework

The above examples of different agents are only some ways, of many, in how one can construct an agent. A paper named Cognitive Architectures for Language Agents (CoALA) [44] has tried to describe a generic way of describing and constructing agents.

In short, the CoALA framework is designed to facilitate the development of agents by clearly separating the decision-making process into distinct components such as proposal, evaluation, and selector.

2.6 Evaluation

This section describes ways to evaluate Large Language Models (LLMs) and retrieval-augmented generation (RAG). This includes multi-needle in a haystack that evaluates the retrieval and reasoning of an LLM. Additionally, three metrics for evaluating an RAG pipeline are described, answer faithfulness, answer relevance and context relevance.

2.6.1 Evaluation Metrics for LLMs

A robust evaluation of Large Language Models (LLMs) encompasses a variety of benchmarks and metrics that assess different capabilities such as reasoning, truthfulness, and commonsense understanding [45], [46]. Key benchmarks include:

- **ARC** [47], **HellaSwag** [48], and **WinoGrande** [49], which test commonsense reasoning through multiple-choice questions designed to probe the model’s understanding of everyday scenarios.
- **MMLU** [50] (Massive Multitask Language Understanding), which covers a wide range of subjects providing a comprehensive assessment of model capabilities across diverse knowledge areas.
- **TruthfulQA** [51] evaluates the truthfulness of LLM responses, crucial for applications where factual accuracy is paramount.
- **GSM8K** [52], which assesses the model’s ability to solve complex, multi-step mathematical problems, indicating its reasoning and procedural problem-solving capabilities.

2.6.2 Agent-Specific Evaluation

Unlike general LLMs, agents are usually designed for specific tasks and therefore require tailored evaluation metrics that focus on performance in the designed task. However, one way of evaluating an agent is using comparison-based metrics. For instance, evaluation techniques may involve comparing the accuracy, inference time, and correctness of the designed agent with that of a human.

2.6.3 Leaderboards and Comparative Analysis

Leaderboards such as the *Open LLM Leaderboard* [53] provide a comparative platform to evaluate and rank LLMs based on their performance across standardized tests and benchmarks. This method offers a competitive analysis, highlighting strengths and weaknesses relative to peers in the field.

2.6.4 Long Context Evaluation

Research continues to develop tests that can effectively measure a model’s ability to handle long contexts, a crucial capability for tasks involving detailed or extended narratives. Examples include the *needle in a haystack* test [54], which specifically evaluates the retrieval accuracy of relevant information from a large dataset.

Recently, multiple tests have been proposed on how one can test LLMs’ long context retrieval functionality. Some of them are passkey retrieval [55], BABILong [56], needle in a haystack [57], multi-needle in a haystack [58] and RULER [59].

2.6.4.1 Needle In a Haystack

The needle in a haystack is a simple test to evaluate the retrieval functionality of an LLM [54]. It involves a long text, referred to as a haystack, and a fact, referred to as a needle. The needle is inserted at one specific point within the haystack, after which the entire content is fed into an LLM and it is asked to retrieve the needle. This is performed on different context lengths and different needle insertion depths. However, only one needle is inserted at the same time.

For the original test, proposed in [57], the haystack was multiple essays from Paul Graham, and the needle was “*The best thing to do in San Francisco is go to Dolores Park and eat a sandwich on a sunny day.*”. The question posed to the model was, “*What is the most fun thing to do in San Francisco based on the context?*”. This test produced good results for GPT-4 [10], but poor performance for Claude 2.1 [60].

Thereafter, Anthropic, the creators of Claude 2.1, responded with their test with two key takeaways [61]. First, they stated that if the subject of the needle is too different from the haystack, the retrieval performance of that needle could decrease. In other words, if the model sees information in a context that is not relevant to the overall context, the model may discard that information as not important. Secondly, using “*Here is the most relevant sentence in the context:*” as a primer, improved the retrieval accuracy.

2.6.4.2 Multi-Needle In a Haystack

One thing that is missing from the needle-in-a-haystack test is that the results do not reflect real-world RAG performance [58], [59]. Usually in an RAG pipeline, multiple facts need to be retrieved, in contrast to the needle in a haystack test, where only one fact is retrieved at a time. Additionally, the needle in a haystack test does not test the LLM’s ability to reason about the retrieved fact, which is a common necessity in an RAG pipeline. Therefore, the multi-needle in a haystack test was developed.

The multi-needle in a haystack test evaluates both the retrieval and reasoning part of an LLM. This is done by inserting N facts (needles) into a long context (haystack) and then asking a question that requires all needles to be considered for a correct answer.

In the multi-needle in a haystack test proposed by [58], they used the question “*What are the secret ingredients needed to build the perfect pizza?*”. The needles were of the format,

```
{pizza_ingredient} {are/is} one of the secret ingredients needed  
to build the perfect pizza.
```

where `{pizza_ingredient}` is a pizza ingredient from a list of ingredients.

According to [58] this test showed that the retrieval accuracy degraded, the more facts the LLM needed to retrieve. Additionally, the retrieval accuracy degraded further when the LLM needed to reason about the facts.

2.6.5 Retrieval Augmented Generation

As described in Section 2.3, Retrieval Augmented Generation (RAG) is comprised of three parts: indexing, retrieval, and generation. There is still no standardized approach to evaluate RAG pipelines [34]. However, one popular approach involves using various metrics to measure both *retrieval* and *generation* quality [34].

2.6.5.1 Retrieval Evaluation

The metrics that test the retriever are typically assessed before the RAG pipeline is deployed. These tests ensure that the retrieval component can accurately and efficiently find relevant information from the dataset. One popular metrics include **context relevance** [34].

2.6.5.2 Generation Evaluation

From the generation part of RAG, there are two popular metrics. The first generation metric is **answer faithfulness** [34]. It is an evaluation metric that tests how grounded the generated answer is in the given context. In other words, it is the inverse metric of how much the generative model hallucinates [18]. Precisely how this is evaluated depends on the research paper. However, it mostly works using another LLM to determine if an RAG-generated answer addresses the user question, given the context [62], [63]. This is then transformed into a YES or NO question.

The second generation metric is **answer relevance** [34]. This metric measures how relevant and useful the generated answer is to the original query. It ensures that the output not only avoids hallucinations but also provides meaningful and contextually appropriate information.

In summary, evaluating a RAG pipeline involves assessing both the retrieval and generation components to ensure the system can accurately find and generate relevant, factual, and coherent information. Despite the lack of a standardized evaluation approach, using metrics like answer faithfulness and relevance helps in understanding and improving the performance of RAG models.

3

Methods

Building upon the theoretical framework established in the previous chapter, this chapter delves into the practical methodologies applied in our research at Volvo Group. It outlines the systematic approach taken to select and customize suitable AI models, develop a robust vector-based dataset, and refine text embedding techniques using the InstructOR-XL [27] model. We also detail the implementation and evaluation of our Retrieval-Augmented Generation (RAG) system, followed by the design and testing of our agent architecture. Each section is interconnected, showcasing the progression from initial model selection to practical application and evaluation, ensuring that our AI agents are not only theoretically sound but also practically effective in enhancing strategic decision-making within the electrification of road transportation.

3.1 Academic Research and Model Selection

In the early stages of our project, we focused on building a robust theoretical base by combining the latest academic research with practical insights from industry professionals. This foundational work is essential because it directs the selection and customization of the AI models that will be tailored for strategic decision-making in the specific domain at Volvo Group.

3.1.1 Initial Understanding and Data Preparation

At the outset of our project at Volvo Group, we embarked on a systematic exploration aimed at enhancing the strategic decision-making process within the electrification of road transportation. This involved an initial phase where we delved into understanding Volvo's current strategic formulation methodologies. Gaining such an understanding is pivotal to ensure that any theoretical models or frameworks developed are not only compatible but also augmentative to the existing processes.

Instead of manipulating actual datasets, our approach focused primarily on theoretical model alignment and conceptual preparations. This included reviewing existing internal documents to align our theoretical frameworks with Volvo's strategic needs. This phase was crucial for setting the foundation of our project, ensuring that our subsequent theoretical developments in AI and large language models would be well-informed and relevant to Volvo's strategic goals.

Instead of traditional data preparation, our work at this stage was centered on identifying key areas where AI could potentially augment Volvo’s strategic decision-making processes. We engaged in discussions with domain experts within the company to pinpoint specific challenges and opportunities where AI could offer significant advantages. It was identified that the prediction of hydrogen prices in 2030 was the main area of interest.

3.1.2 Model Selection and Customization

Selecting the optimal model was an important next step in our project. Initially, we considered leveraging the capabilities of GPT-4 [10] for its robust performance in processing extensive and complex datasets. However, we chose to utilize open-source models such as Mistral 7B [64] and Mixtral 8x7B [65]. The choice was because of the availability of these models within our resource constraints and their compatibility with Volvo’s enterprise data security standards. Mixtral was selected as it was the best model accessible to us, providing a balance between performance and adaptability to our specific needs.

Instead of customizing through fine-tuning or pre-training, our approach focused on optimizing how we interact with the chosen model. This involved conducting a series of experiments to determine the most effective ways to formulate prompts and utilize the model’s capabilities. Our experimentation aimed to identify strategies that maximize efficiency and accuracy in text data processing without modifying the model architecture. This was necessary because various sources provided different answers on how to prompt the model.

Our experiments, as seen in Section 3.5.1, found that the following two prompt formats were the most efficient and accurate way of prompting the model. The first prompt format is used when only one instruction and no primer are given to the model:

```
<s> [INST] {USER_INSTRUCTIONS} [/INST]
```

In the other case, where both an instruction and primer are given, the prompt format looks like this:

```
<s> [INST] {USER_INSTRUCTIONS} [/INST]
{primer}
```

The `<s>` is a special character that denotes the beginning of a string. On the other hand, text placed inside `[INST]` and `[/INST]`, the `{USER_INSTRUCTIONS}` will count as the instructions for the model. For the raw Python format, see Appendix A.1.1.

3.1.3 Integration of Academic and Industry Research

The initial phase of our research was mainly focused on integrating both academic and industry-specific insights. We conducted a thorough review of current scholarly articles on generative AI, focusing on technologies such as Retrieval-Augmented Generation (RAG) and the Cognitive Architectures for Language Agents (CoALA)

framework [62]. This academic inquiry not only enriched our theoretical understanding but also kept us ahead of the rapidly evolving field of AI. Throughout our thesis work, multiple new papers were released, emphasizing the quick pace of advancement in this area. This underscored the need for continual learning and adaptation in our approach to stay current with the latest developments.

We engaged with Volvo’s internal development frameworks and discussed extensively with industry experts. Learning a new development framework as the starting point, ensured that our theoretical research was consistently aligned with practical, real-world applications in Volvo Group, thereby providing a robust foundation for the practical deployment of AI technologies within Volvo.

This integrative approach has equipped us with a comprehensive perspective, blending cutting-edge academic research with pragmatic industry practices, to effectively drive forward our project on strategic AI implementation.

3.2 Vector-Based Dataset Development

Following a grounding work in academic and practical frameworks, our work turned to the development of a specialized vector-based dataset. This dataset is designed to support the retrieval-augmented generation (RAG) systems by providing a structured form of data that is essential for testing and refining the AI models we previously selected.

3.2.1 Dataset Development Preparation

The initial dataset was sourced from seven PDF documents containing domain-specific field analysis reports, provided by our supervisor. These documents were selected to ensure the relevance and richness of the information they contained in the domain field. However, during the data preparation process, we discovered that two of these documents consisted only of images, which limited their importance for text-based vectorization. Consequently, these two documents were excluded from the dataset to maintain a focus on textual content, which is crucial for effective retrieval in RAG systems. Following this, it was determined that only two of the five remaining documents would be utilized; one in a tabular format and the other in plain text. These two documents were later used in the evaluation of the agent.

In the beginning, the plan was to have one extensive vector database that could contain all the documents. However, this was abandoned after realizing the work needed to keep one of those databases updated. Instead, a user-specific database was introduced, allowing users to upload and delete any documents of their choosing. The uploaded documents are processed to extract and vectorize textual data, aiming to create a rich and easily retrievable pool of vectors. This vectorization process involved transforming the dense information contained within the documents into a format that can be effectively processed and retrieved by AI models, thereby enhancing the efficiency and accuracy of the RAG system in generating relevant responses. The vectors are then stored in a Facebook AI Similarity Search (FAISS)

[66] vector database.

A practical aspect of our dataset development was the integration of metadata storage functionalities. This integration was primarily implemented to replace previously used external systems, such as LangChain [67], with an in-house solution that aligns better with our specific needs. While the integration of metadata may not directly boost the efficiency of search processes, it improves the manageability and adaptability of the dataset. We enable our AI models to quickly locate and reference pertinent information, which is important for the performance of content retrieval tasks. This feature also provides the flexibility for future expansions to the dataset, ensuring that our data infrastructure can evolve in response to new requirements or opportunities.

The design and development of this vector-based dataset are crucial for supporting the capabilities of our AI models, laying a robust foundation for our experimental work that follows.

3.2.2 Utilizing InstructOR-XL

The adoption of the InstructOR-XL model, built on transformer architectures, marks an advancement in our text embedding capabilities [27]. The prompt used for the embedding of the documents was:

```
Represent the science PDF document for retrieval:
```

On the other hand, the prompt used for embedding the search query was:

```
Represent the science question for retrieving supporting documents:
```

3.3 Retrieval-Augmented Generation

Equipped with a refined text embedding process, we proceeded to the implementation and rigorous evaluation of the Retrieval-Augmented Generation (RAG) model. This stage is critical for integrating and testing all the components we have developed so far, ensuring that the AI agent functions effectively within Volvos technological and operational framework.

3.3.1 Implementation Strategy

Following the development of our vector-based dataset, we initiated the implementation of a Retrieval-Augmented Generation (RAG) model. Our initial setup involved configuring a basic RAG model by integrating both Mixtral-8x7B and InstructOR-XL. The integration utilized the LangChain retrievers' packages [68], with some modifications to work with Volvos framework. LangChain was chosen for its ease of use and implementation.

3.3.2 Parent Document Retriever Implementation

Building upon the basic RAG framework, we implemented an advanced version known as the Parent Document Retriever [35]. This retriever also originates from the LangChain retriever packages. This enhancement was developed to refine the granularity of content retrieval by organizing the database hierarchically into parent documents and child chunks. Each child chunk was indexed with high-resolution embeddings that capture nuanced textual meanings, facilitating more precise retrievals. The RAG system identifies relevant child chunks, to the text query, and then retrieves the entire associated parent document. This approach not only preserved the context but also ensured that the responses were comprehensive and informative. We conducted extensive testing to ensure that this more complex retrieval logic did not compromise retrieval accuracy. For the settings that were used, see Appendix A.2.

With the RAG and database developed, it was now possible to upload a document to the database and retrieve pages of information similar to the search query.

3.4 Agent Architecture Implementation

After assessing the RAG model’s performance, the next step involves the design and implementation of an agent architecture. This architecture is tailored to enhance the interaction capabilities of our AI agents, enabling them to handle a wider range of queries and integrate seamlessly with existing processes at Volvo. The implementation of this architecture is essential for deploying AI agents that can effectively navigate the complex landscape of strategic decision-making.

In developing our AI agents, we aimed to start simple and add more complex features as the project progressed. The plan was to make small changes to each iteration so it would be easier to debug and record the performance gains. With this in mind, we constructed two agents, Agent V1 and V2. However, this approach took more time than anticipated and for the last agent, V3, we skipped that step and added multiple features at the same time. Refer to Section 5.1.3 for an explanation of why this was not a good idea.

3.4.1 Agent V1

The first iteration of our agent was architected to handle straightforward query-response interactions using a single RAG action. This basic setup allows the agent to effectively retrieve direct answers to a given user query. One of agent V1s downsides is that it does not work in multiple iterations. In other words, the user asks a question and the agent will rewrite the question, search the vector database, and retrieve the most similar information to the question. The information will then be rewritten by the LLM and given as the answer to the user. This design limits the agents’ ability to handle complex questions that require the use of multiple actions.

One thing to note about the above pipeline is that this is closer to an advanced RAG pipeline [34] than an agent.

3.4.2 Agent V2

Agent V2 is the second iteration and builds upon Agent V1 by incorporating a modular action library, working memory, and an iteration loop.

The modular action library is a library that contains all the available actions that the agent can perform. It is built upon the Lagent framework [69] and is referred to as the **executor**. Not only does it contain all the actions available, but it is also capable of executing these actions.

Additionally, Agent V2 contains a working memory that enhances its ability to handle a sequence of interactions within multiple iterations. The inclusion of working memory allows the agent to maintain context over an interaction sequence, thereby improving its responsiveness and accuracy in extended dialogues.

This version of the agent can perform two actions:

- **RAG action** is similar to how Agent V1 handles the RAG action, but Agent V2 writes the RAG query. Contrary to the Agent V1 RAG action, this action also returns a fact. This fact is then stored in the working memory and retrieved when needed to decide on what to do next.
- **Terminal action** is the last action the agent will take in each iteration. When the agent determines no further actions are necessary, this action is executed. It concludes the agent loop and answers the question given by the user.

This setup aims to address more complex queries by breaking them down into manageable actions that are executed iteratively.

3.4.3 Agent V3

Agent V3 is our most complex agent that improves on parts of Agent V2 while keeping the core the same. Initially, we attempted to follow the Cognitive Architectures for Language Agent (CoALA) framework for its modularity and adaptability, which is well-suited for creating complex decision-making processes within agents [44]. However, in the practical application of the CoALA framework, we encountered several challenges that led us to diverge from its structured approach. One of those challenges was that Mixtral 8x7B had a hard time “understanding” when to choose the terminal action and answer the user.

As a result, we adapted our approach by developing a customized agent architecture that better aligns with the performance of Mixtral 8x7B, as seen in Figure 3.1. This revised architecture still draws on the principle of modularity but incorporates greater flexibility to address the specific nuances and complexities we faced.

During the implementation of this advanced version, we encountered challenges with the model’s performance concerning the processing of longer token-length prompts. The details of these performance issues are further discussed in Section 5.2.2.

To address these challenges and improve the overall quality of responses, we designed an intricate architecture that extends beyond the conventional CoALA framework.

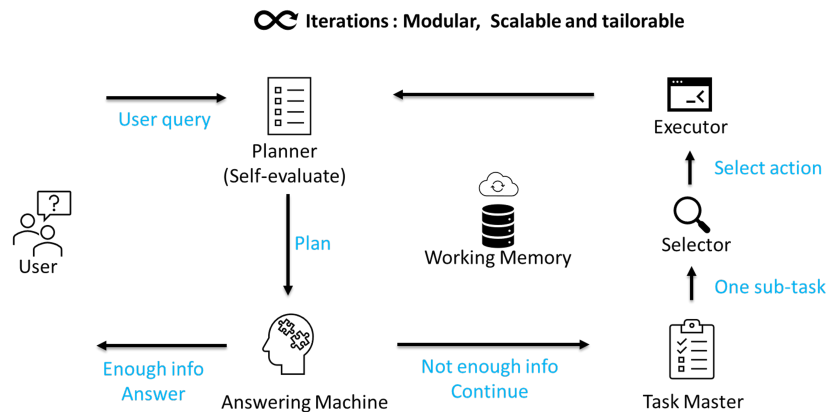


Figure 3.1: Flowchart of Agent V3 describing how the agent functions.

This revised architecture incorporates several new modules and a complex reasoning loop designed to manage and streamline longer interactions more effectively.

3.4.3.1 Enhanced Architectural Workflow

The process begins when a user query is input into the system. This query first passes through the **planner** module, which generates a step-by-step plan based on the content of the query and the facts in the fact list. The goal of the planner is to split up the query into multiple steps. This plan can then be evaluated by the **evaluator** model if the user decides to use that part. The module’s objective is to critique the plan and give notes on how it could change. If the evaluator thinks changes are necessary, then the notes are sent back to the planner module, where the plan will be improved. This plan-evaluation loop can be stopped by reaching the maximum number of evaluations or when the evaluator is satisfied with the plan, and the plan will then be saved into working memory. Additionally, the whole plan-evaluation loop can be bypassed by giving the Agent an already predefined plan.

This plan is then assessed by the **answering machine**, which checks against the facts currently stored in the working memory to determine if it possesses enough information to respond adequately. If the existing facts are sufficient, then the answering machine proceeds to terminate the iteration and generate a response to the user. If not, the answering machine will generate the reason why it did not terminate and save it to working memory. It’s important to note that the answering machine replaces the terminal action from Agent V2. The reason for using the answering machine can be read more in detail in Section 3.4.3.3.

If the answering machine decides to continue, then the next step is the **taskmaster** module. The taskmaster’s objective is to generate one task to solve this iteration, using the plan from the planner, the known facts, and the reason for not terminating from the answering machine.

The task from the taskmaster is then retrieved by the **selector** that will identify the most suitable action, from the action library, to address the task.

This action is then executed by the corresponding **executor**, and the result is stored in working memory as a fact. When the fact is stored, it is back to the planner to update the plan, if needed. Then the answering machine determines if the iteration loop is ready to answer the user or continue one more iteration. All the prompts for each role can be found in Appendix A.1.4.

3.4.3.2 Memory Management and Modular Interaction

Throughout this process, all modules interact dynamically with the working memory, both reading from and writing to it. This continuous interaction ensures that the memory is consistently updated with the latest facts, which not only aids in generating immediate responses but also enriches the depth and reasonableness of the agent’s outputs over time.

This complex design ensures that at no point does the Large Language Model (LLM) encounter prompts that are too complex to handle effectively, thereby maintaining the quality and accuracy of the responses. Moreover, the accumulation of facts in the memory allows the agent to deliver more reasoned and contextually enriched answers.

3.4.3.3 Implementation Details

At an important step, where the agent needs to decide, an Automatic Chain of Thought (ACoT) [38] is used as a primer. This is used to let the model reason about what it can do at that step and should, according to their paper, improve the performance of the model.

To let the model understand Volvo-specific terminology we added a simple abbreviation dictionary. This dictionary contains some of the commonly used Volvo abbreviations. Whenever an abbreviation is mentioned in the fact list, the full description will also be added to the bottom of the fact list. In Section 5.3.2 we discuss how this can become more advanced and automated.

In order for the LLM to be able to communicate with the code base, each module needed to answer in a specific format. This way of communicating is loosely called “*function calling*”. This format is usually in clear text but with some special headers that decide where the section begins and ends. Examples of these headers could be “**## Reason ##**”, “**## Decision ##**” and “**## Answer ##**”. However, the exception to this is the selector that answers in JSON format. This was done to make it easier to communicate with the executor, where the information must be presented in a specific manner. The reason not all modules are responding in JSON format is that it was observed that constraining the model to JSON responses compromised the length and quality of the output. Therefore, the choice was made to let most modules answer using special headers. On the other hand, this method may occasionally generate extra, unnecessary information beneath certain headers.

While developing the first iteration of Agent V3, we encountered some poor performance with the selector. At that time, the selector had multiple objectives, first to pick a goal to accomplish, then select which action to take. The action library

also included the terminal action, which means that the selector was also responsible for terminating and answering the user. The poor performance was that the LLM would have everything needed in the context window, but still pick the wrong action. After some initial tests, we decided to split the selector into three different roles, answering machine, taskmaster, and selector, as described previously. With this change, the agent performed a bit better.

3.5 Evaluation Method

Having established the agent architecture designed to enhance interaction capabilities and effectively handle complex queries, the next crucial phase of our project focuses on a rigorous evaluation method. This section outlines our comprehensive strategy to assess the effectiveness and efficiency of the AI agents developed. By implementing a diverse array of metrics and tests, we aim to ensure that these agents meet the high standards required for operational success in real-world scenarios, ultimately validating the architectural choices made and identifying areas for further refinement.

3.5.1 Prompt Format

The initial step in our evaluation involves the systematic analysis of different prompt structures. We experimentally test how variations in prompt format affect the agents' performance, focusing on:

- **Instruction Following:** Measuring how well the LLM followed the instruction it was given. Either 0 or 1 per prompt, where 0 is that the LLM did not follow the given instruction and 1 is that it did follow the instruction.
- **Text Repetition:** Evaluating the frequency of unnecessary repetition in the agent's responses as an indicator of prompt clarity. For example, if the LLM generated additional questions or answers in the response. Either 0 or 1 per prompt, where 0 is no text repetition and 1 is text repetition.
- **Generation Length:** The length of the generated text.

There were a total of 14 questions used in the test, and they can be found in Appendix A.3. In short, they contain five components, system instruction, user question, user role, AI role, and primer. Additionally, the prompt format that was tested include:

- **Pure text:**

```
{system instruction} {user question}
```

- **Instruction string only:**

```
[INST] {system instruction} {user question} [/INST]
```

- **Both strings:**

```
<s> [INST] {system instruction} {user question} [/INST]
```

- **Advanced:**

```
<s> [INST] {system instruction}
{user_role}: {user question} [/INST]
{ai_role}: {primer}
```

The test was run four times, two times with an LMM temperature of 0 and the other two with a temperature of 1. To measure the *instruction following* and the *text repetition* metric we used another instance of Mixtral 8x7B Instruct that worked as an LLM judge. The prompts for these judges can be found in Appendix A.3.1. The result of the experiment can be found in Section 4.1 and Figure 4.1, and they indicating that the advanced prompt format gives the best results over the different prompt formats.

3.5.2 Long Context

While testing different parameters for the RAG action, we observed poor performance for some queries. With this in mind, we made some initial tests and identified that the model had problems finding the answer in the given context. In other words, something was wrong with the generation part of the RAG.

To identify the core problem, we constructed a Multi-Needle in a Haystack test [58] that test both the retrieval and reasoning capabilities of the LLM. This implementation takes N secret, and uncommon, pizza ingredients (needles) and adds them to a long text (haystack) of Pual Graham’s essays. The needles used were, *egg*, *bivalves*, *tuna*, *banana*, *lettuce*, *pineapple*, *shrimpt*, *mushrooms*, *curry* and *fries*. These ingredients were selected because they are not very common on pizza, making them less likely to be predicted by the model. Each ingredient was capitalized and then rewritten into the given sentence

```
" {ingridient} is one of the secret ingredients needed to build
↪ the perfect pizza."
```

Thereafter, the sentences are evenly distributed and inserted throughout the haystack. The haystack is then given to the prompt which contains the instructions for the model, which can be seen in Appendix A.1.7. The length of the haystack varied from 1000 – 32000, with a step size of 1000 tokens, and ran 20 times at each token step. The results are presented in Section 4.2, Figure 4.2 and Figure 4.3.

3.5.3 Retrieval-Augmented Generation

To objectively evaluate the Retrieval-Augmented Generation (RAG) pipeline, we designed a testing framework, inspired by RAGAS [62]. The testing framework includes the three metrics from RAGAS, **Faithfulness**, **Answer Relevance** and **Context Relevance**.

3.5.3.1 Answer Faithfulness

To evaluate the faithfulness of the RAG generation part, we used the RAGAS [62] approach. This approach generates statements and verifies if the statements are grounded in the context.

The RAGAS method to estimate the faithfulness of a RAG pipeline works in three steps [62]. First, an LLM will be prompted to generate statements about the RAG-generated answer, given the user question. The following prompt is the same as in the RAGAS paper:

```
Given a question and answer, create one or more statements from
↪ each sentence in the given answer.
question: {question}
answer: {answer}
```

Additionally, the primer “Statement 1:” was used.

The next step is to verify that each statement is supported by the RAG-retrieved context. This is accomplished using a verification function and applying that function to each of the statements. The verification function is an LLM call with the following prompt:

```
Consider the context and the statement provided below.
Then, determine whether the statement is supported by the
↪ information present in the context.
Before providing the verdict (Yes/No), give a short explanation
↪ that directly addresses whether the context supports the
↪ statement.
### Context:
{context}
### End of context
statement: {statement}
Answer in the following format:
Explanation: [Give a short explanation that addresses the support
↪ of the statement by the context.]
Answer: [ONLY ANSWER WITH "YES" OR "NO", NOTHING ELSE!]
Do not deviate from the specified format.
```

The above prompt is a combination of the RAGAS prompt and our ideas. Additionally, the primer “Explanation:” was used.

The final step is to compute the faithfulness score with (3.1).

$$F = \frac{\text{supported statements}}{\text{all statements}} \quad (3.1)$$

In other words, the faithfulness score is the number of statements supported by the context, according to the LLM, divided by the total number of statements.

3.5.3.2 Answer Relevance

To estimate the answer relevance of the RAG-generated answer, we let an LLM generate n possible questions for the given answer. In the LLM call, this prompt was used:

```
Generate a question for the given answer.  
answer: {answer}
```

Additionally, the primer “Question 1:” was used. Given all the generated questions, we can now obtain the cosine similarity between the user’s question and the LLM-generated questions. This is done by using a text embedding model, such as InstructOR-XL [27] in our case, to calculate the text embeddings for each question. Finally, the answer relevance score is the mean over all the similarity scores,

$$AR = \frac{1}{n} \sum_{i=1}^n Sim(\vec{q}, \vec{q}_i),$$

where \vec{q} and \vec{q}_i are the text embedding vectors for the user’s question and the LLM-generated ones, respectively.

One important point is that RAGAS’ way of computing answer relevance does not consider factuality when computing the score. This means that the answer can be wrong but still have a high answer relevance score.

3.5.3.3 Context Relevance

The context relevance metric uses an LLM to extract relevant sentences from the RAG-retrieved context, that is needed to answer the user’s question. The prompt used is a modified version of the RAGAS one:

```
Please extract relevant sentences from the provided context that  
↪ can potentially help answer the following question.  
If no relevant sentences are found, or if you believe the  
↪ question cannot be answered from the given context, return  
↪ ONLY the phrase "Insufficient Information".  
While extracting candidate sentences you're not allowed to make  
↪ any changes to sentences from the given context.  
Give the sentences WITHOUT explanation!  
### Context:  
{context}  
###End of context  
question: {question}
```

The context relevance score is then described by (3.2).

$$CR = \frac{\text{extracted sentences}}{\text{all sentences}}. \tag{3.2}$$

In other words, it is the relevant extracted sentences divided by all sentences in the context.

One drawback is that calculating the number of sentences in a text can be a hard task for a computer. We decided to estimate the number of sentences by first replacing every “...” with “.” and then counting the number of “.”, “!” and “?” in the context.

3.5.4 Agent

When evaluating the agent’s capability, we decided to do a comparison-based test between two other established agents. The two other agents were ReAct [3] and AutoGPT [4]. The test dataset we used was put together by our advisor at Volvo, and it contained around 50 hydrogen-specific questions and answers of different complexity. Some questions can be found in Appendix A.4 and below are the names we gave the dataset:

- Simple questions, IEA the future of hydrogen assumptions annex (simple 1)
- Simple questions, The future of hydrogen (simple 2)
- Complex questions, IEA the future of hydrogen assumptions annex (complex 1)
- Complex questions, The future of hydrogen (complex 2)
- Calculation questions, IEA the future of hydrogen assumptions annex (calculations)

Additionally, some of the questions were asked to GPT-4o [70]. In this environment, the document was uploaded to a chat and the question was then asked to the model. However, GPT-4o is a Large Multimodal Model, so it can also analyze figures in the given documents. The answers given by GPT-4o were then manually, and automatically, checked against the human answers.

In the evaluation, each question would be asked to all the agents three times. Every agent had access to the same two actions, a calculator with unit converter ability and a RAG action with one document. The answers received by the agents were checked against the human answer with an LLM judge, where the prompt can be seen in Appendix A.1.2.

The Lagent [69] implementation of ReAct and AutoGPT were used. However, none of them worked out of the box with Mixtral 8x7B Instruct v0.1 [71]. Both of them had problems with answer formatting, which is a common problem with Mixtral, see Section 5.2.5.1. Additionally, both agents’ prompts caused the model to be prone to repetition. Therefore, the new prompts for ReAct and AutoGPT are available in Appendix A.1.5 and Appendix A.1.6, respectively.

3.5.5 Conclusion

This methodology provides a thorough framework for evaluating AI agents, ensuring that each aspect of the agent’s function and performance is scrutinized and optimized. Through this comprehensive testing and evaluation approach, we aim to

3. Methods

develop agents that are not only technically proficient but also truly beneficial in practical applications.

4

Results

In this chapter, the results from the experiments and the comparisons of Agent V2 and V3 are presented. There were three important experiments conducted, firstly, on the effect prompt format has on the Mixtral 8x7B Instruct v0.1 [71] large language model (LLM). Secondly, long context retrieval and reasoning understanding of Mixtral 8x7B Instruct v0.1 using the multi-needle in a haystack [58] approach. Thirdly, the retrieval-augmented generation (RAG) performance was evaluated using a modified version of RAGAS [62]. The comparisons based evaluation of Agent V2 and V3 are primarily between ReAct [3] and AutoGPT [4]. However, GPT-4o [70] are compared against Agent V2 and V3 in some aspects.

4.1 Prompt Format

This section displays the average results of the prompt format experiment, as seen in Figure 4.1. The experiment was conducted four times using 14 questions, with two trials at an LLM temperature setting of zero and the other two at a setting of one. The results of the two runs, conducted at the same LLM temperature, have been averaged and are presented in the table below. The prompt formats are presented in Section 3.5.1.

4. Results

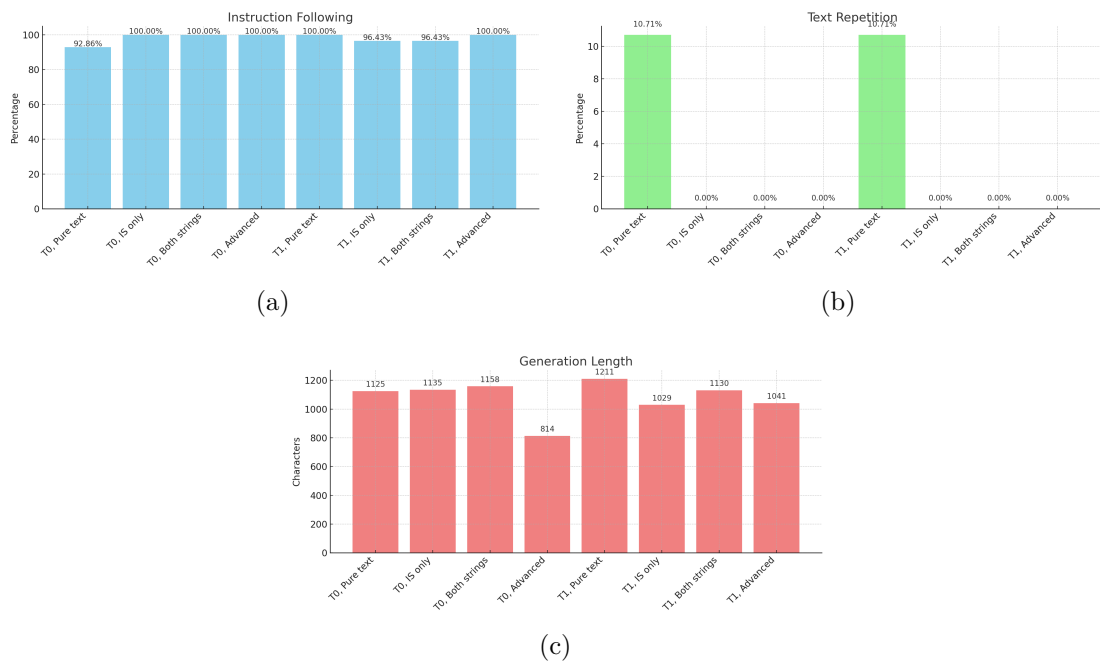


Figure 4.1: Average results for the different prompt formats with LLM temperature of 0 (T0) and 1 (T1). BOS stands for *Beginning of string* and IS stands for *Instruction string*.

4.2 Multi-Needle In a Haystack

We evaluate the LLM’s ability to handle complex queries involving multiple data points (needles) embedded within extensive text data (haystack). This section presents the outcomes of the *Multi-Needle in a Haystack* test, showing how effectively Mixtral 8x7B Instruct v0.1 [71] retrieves and synthesizes information from a large prompt. The results are crucial for understanding the LLM’s retrieval and reasoning efficiency in real-world scenarios where data is dense and multifaceted.

Two experiments were conducted, the first where the prompt length started at 1 000 tokens and ended at 32 000 tokens, with a step size of 1 000, as displayed in Figure 4.2. The second experiment was more focused on smaller prompt length and started at 1 000 tokens and stopped at 10 000 tokens, with a step size of 500, as displayed in Figure 4.3. Both the experiments ran each prompt 20 times and the average results were then displayed. Additionally, the insertion depths are ordered from the top part of the prompt (*egg*) to the end of the prompt (*fries*), where the other needles are inserted uniformly into the prompt. For example, the needles *lettuce* and *pineapple* will always be placed around the middle of the prompt.

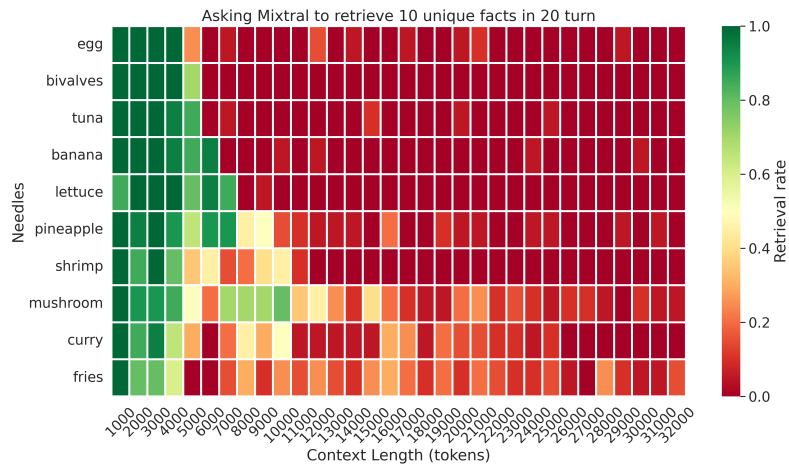


Figure 4.2: Testing Mixtral 8x7B Instruct v0.1 [71] on multi-needle in a haystack test. The prompt length went from 1 000 tokens to 32 000 tokens with a step size of 1 000 tokens. Each prompt was tested 20 times to get an average over each needle. The first needle, denoted as *egg*, was always positioned at the start of the prompt, with the subsequent ones spaced equally from each other.

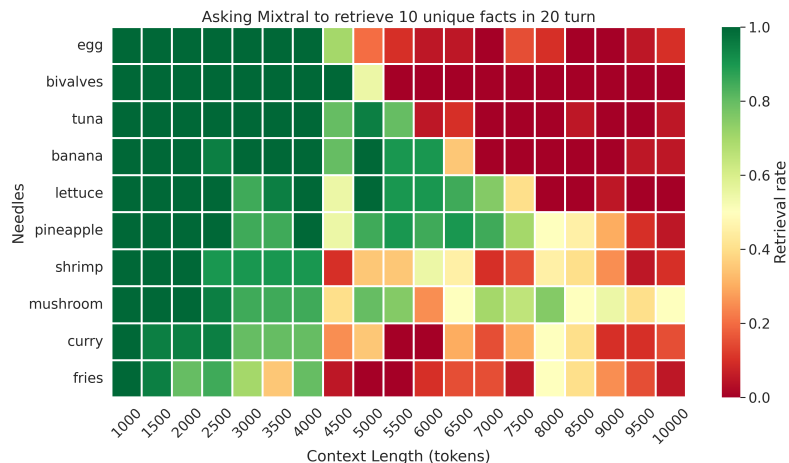


Figure 4.3: Testing Mixtral 8x7B Instruct v0.1 [71] on multi-needle in a haystack test. The prompt length went from 1 000 tokens to 10 000 tokens with a step size of 500 tokens. Each prompt was tested 20 times to get an average over each needle. The first needle, denoted as *egg*, was always positioned at the start of the prompt, with the subsequent ones spaced equally from each other.

4.3 Retrieval-Augmented Generation

The evaluation of the RAG pipeline is meant to happen while the agent is running. However, the results from some datasets are still relevant. Dataset *simple 1* and *simple 2* are datasets that the agent can answer in one RAG request and therefore suitable for an RAG-only evaluation. The following table, Table 4.1, displays the

4. Results

average results of running the evaluation three times. In the evaluation, the correct answer is irrelevant and the questions from the datasets are the input-query to the RAG pipeline.

Table 4.1: The average results for evaluating only the RAG pipeline on the questions in the two datasets. The evaluation ran three times per dataset and the sample standard deviation is used to show variation in responses.

Dataset	Answer Faithfulness	Answer Relevance	Context Relevance
Simple 1	0.911 ± 0.021	0.887 ± 0.004	0.0738 ± 0.0148
Simple 2	0.923 ± 0.024	0.860 ± 0.006	0.0697 ± 0.0069

4.4 Agent Comparisons

In this section, the results of the comparison-based agent evaluation are presented. The comparisons mostly focus on our Agent V2 and V3 compared to ReAct [3] and AutoGPT [4]. However, some comparisons were made with GPT-4o [70].

The comparisons were performed on five datasets consisting of human-annotated questions and answers, totaling approximately 50 annotations. When comparing ReAct and AutoGPT to our agents, each question was asked three times, and the results were averaged, as seen in Figure 4.4. The questions asked to GPT-4o were only asked once.

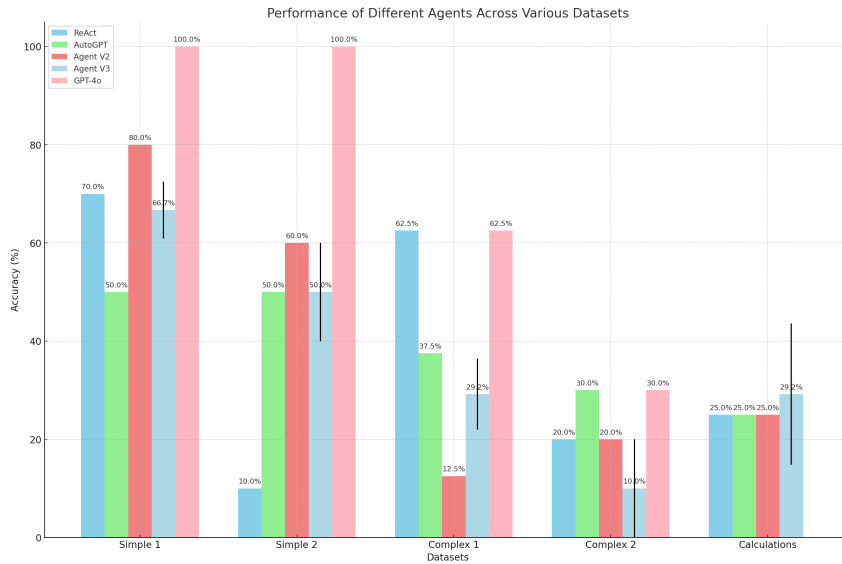


Figure 4.4: An overview of the average results from three trials per dataset, except for GPT-4o, which was tested only once. The calculations dataset was not evaluated on GPT-4o. The sample standard deviation is used to show variation in responses and is represented by the vertical line in the bars.

5

Discussion and Conclusion

In this chapter, we provide an in-depth analysis of the findings from the evaluation of the AI agents developed during this study. We discuss the implications of these results, examine the strengths and limitations of our approach, and suggest avenues for future research. We also reflect on the theoretical contributions and practical applications of our work.

5.1 Discussion of Theoretical Findings

This section delves into how our findings align with existing AI and agent systems theories. We analyze the theoretical underpinnings of our approach, highlighting how our results either support or challenge current academic perspectives.

5.1.1 Drawbacks of Retrieval-Augmented Generation

While Retrieval Augmented Generation (RAG) offers significant advantages in enhancing the capabilities of Large Language Models (LLMs), it also has several limitations. Here are the key drawbacks:

- **Accuracy and Reliability:** RAG systems are not infallible and can sometimes produce incorrect responses. The accuracy of the generated content heavily depends on the quality and relevance of the retrieved data. Misunderstanding or ambiguities in the input can result in some inaccurate outputs.
- **Context Window Limitations:** The functionality of RAG is constrained by the context window of the underlying LLM. The amount of data that can be processed in a single operation is limited to the models maximum context window size, which may restrict the breadth of information accessible for response generation.
- **Dependency on Data Quality:** The effectiveness of RAG is directly influenced by the quality of the data in its retrieval database. If the data is outdated, incomplete, or inaccurate, the performance of the RAG system will be adversely affected.
- **Computational Efficiency:** RAG requires substantial computational resources to perform indexing. Consequently, indexing a document of 203 pages takes approximately 1 minute and 30 seconds.

- **Scalability Challenges:** Maintaining an efficient and up-to-date retrieval index becomes increasingly difficult as data volumes grow. This issue can impact the retrieval process’s speed and accuracy, posing scalability challenges in large-scale applications.

5.1.2 Drawbacks of Current Evaluation Methods

Evaluations often need to evolve as models improve. Historical benchmarks like SWAG [72], Winograd [73], GLUE benchmark [74] and super GLUE from 2019 [75] have shown that models can quickly reach near-human performance, potentially due to overfitting or exposure to test data during training [48], [49]. This necessitates continuous development of harder, more nuanced tests to challenge and accurately assess the latest models [50].

5.1.3 Challenges in Iterating Agent Architecture

The iteration plan for the agent’s architecture was not quite followed. From Agent V2 to Agent V3, we added a large number of modules. On one hand, this was because we discovered that the mixtral model performs poorly with long contexts. On the other hand, due to time constraints, we made a big jump in the structure of V3. This has actually created some difficulties for our subsequent optimizations, in other words, it was not a good choice.

We need to test many modules simultaneously and try to test the connections between them, which increases the difficulty of pinpointing specific issues. It is challenging to quickly identify which module’s performance needs improvement. Moreover, as more modules are added, the challenge of how they communicate information with each other also arises. We believe an ideal plan should involve gradually adding one module at a time, testing it to achieve optimal results before adding new ones.

5.2 Discussion of the Results

In this section, we provide a detailed explanation of the experimental results obtained from testing our AI agents. We assess the performance of each agent model across various metrics, discuss the significance of these results in the context of their practical application at Volvo, and evaluate how effectively our AI agents handled complex tasks and contradicting information.

5.2.1 Prompt format

The reason for performing this evaluation may seem redundant. However, we had some internal discussions on how to prompt the Mixtral 8x7B Instruct v0.1 model, and the conclusion was not obvious. The HuggingFace model card [71] only clearly shows how you should prompt the model if you already have an LLM response:

```
<s> [INST] Instruction [/INST] Model answer</s>  
[INST] Follow-up instruction [/INST]
```

However, we were interested in how the model should be prompted, for best results, without the model answer. As a result, we conducted our experiment that showed that it is okay to remove everything after the [/INST] string. Furthermore, we found that a primer can work well on the Mixtral model, even though it is not stated in the model card.

5.2.2 Multi-Needle In a Haystack

The result from the multi-needle in a haystack was surprising in two ways. First of all the paper that introduced Mixtral claimed a 100 % retrieval rate [65] on the Passkey task [55]. This test is similar to the normal needle in a haystack, but a passkey is inserted randomly into the prompt. Additionally, the paper *LLM In-Context Recall is Prompt Dependent* [54] made multiple evaluations on Mixtral 8x7B Instruct v0.1 with the normal needle in a haystack test. They evaluated Mixtral with three different needle sentences, PistachioAI, San Francisco, and Thornfield Hollow. Additionally, they also used Paul Grahams essays for the haystack. Their result can be seen in Figure 5.1. Additionally, the paper *RULER: Whats the Real Context Size*

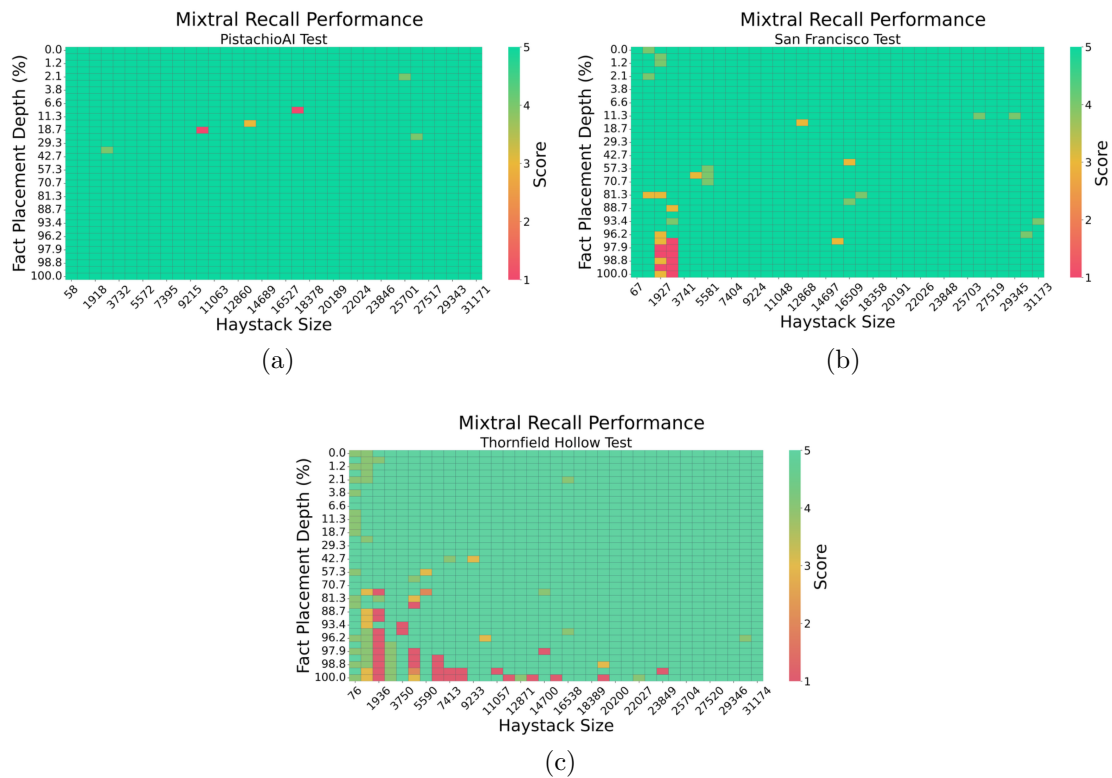


Figure 5.1: Figures from *LLM In-Context Recall is Prompt Dependent* [54]. The retrieval accuracy of Mixtral 8x7B Instruct v0.1 for three different needles.

of Your Long-Context Language Models? [59] also tested Mixtral 8x7B Instruct v0.1 and received total accuracies around 80 %. Secondly, our test contradicts the *Lost in the Middle: How Language Models Use Long Contexts* [76] paper, by showing that Mixtral is better at “remembering” text in the middle, but not in the beginning and

end. However, this “conclusion” requires more research.

We have some explanations of why we see these deviations in the results. Firstly, the evaluation the other papers perform are similar, but not identical to our tests. In our test, multi-needle in a haystack [58], multiple needles are inserted at the same time, and the model is asked to retrieve all of them. On the other hand, their evaluations only contain one needle, and the model is asked to retrieve only one needle. Secondly, as described in Section 2.6.4.1, the model could perform worse because the needles about pizza ingredients are too different compared to the essays. Thirdly, there could be something wrong with the code. Finally, the result could be accurate and this is just a much more challenging test compared to the normal needle in a haystack.

5.2.3 Retrieval-Augmented Generation

The results from the Retrieval-Augmented Generation (RAG) evaluation were as expected. Unfortunately, no RAG evaluation while the agent was running has been performed. However, it is possible to implement the three metrics into the agents at runtime. This implementation is discussed more in detail in Section 5.3.1

One metric that stands out more than the others is *Context Relevance* because of its low score. A score of around zero indicates that almost none of the retrieved context is useful for answering the RAG query. However, this was also expected because of the way the retriever in the RAG pipeline is built. As described in Section 2.3.2 and Section 3.3.2, the retriever retrieves four pages of context in each query. Consequently, a large part of the retrieved context will be unused. Furthermore, the estimation of the number of sentences could often overestimate the number of sentences in the context, giving this metric a lower score. However, according to [77], a “normal” document contains around 25-34 sentences per page. With this in mind, and if only two sentences are used to generate the RAG answer, this would give a context relevance score of around 0.0174 as seen in (5.1).

$$\text{Low Estimate: } \frac{2}{25 \cdot 4} = 0.0200$$

$$\text{High Estimate: } \frac{2}{34 \cdot 4} = 0.0147 \tag{5.1}$$

$$\text{Average Estimate: } \frac{0.0200 + 0.0147}{2} = 0.0174$$

The average difference between the estimated and actual context relevance score is,

$$\frac{0.0738 + 0.0697}{2} - 0.0174 = 0.05435.$$

While this is a small absolute difference, it represents a large percentage difference relative to the actual context relevance score we achieved.

5.2.4 Agent results

The results from Agent V3, as seen in Figure 4.4 were not what we were expecting. First of all, it is the only model that has some deviation between the answers. This is probably because of the use of non-zero LLM temperature in some parts of the agent structure. Secondly, it underperformed drastically in the dataset *Complex 1*. However, as seen in Appendix A.5.1, it is the only agent capable of answering most questions, at least once. For example, in the *simple 1* dataset the agent was able to successfully answer all questions at least once, a feat not achieved by any other agent. As a result, we think more work needs to be performed to improve the reliability of the agent and this is discussed in Section 5.3.3.

We were also expecting, at least one, of the agents to receive 100% accuracy on the dataset *Simple 1*. However, this was not the case. From the answer trace, we can see that there are three main issues that the evaluation encounters:

1. **LLM Judge:** The agent answers were judged by an LLM who stated if the answer was correct or not. This judge could sometimes give a correct label even though the LLM raised an exception because of the formatting, even though it was stated in the prompt to not do so. The reason why it gave the correct answer was because the wrongly formatted answer contained the correct answer. Additionally, it is an LLM that judges the answers and therefore it can also make mistakes. However, most of the agents have zero percent deviations per answer, as seen in Appendix A.5.1. Thus, we know that if the LLM judge makes mistakes, it is at least consistently making the same mistakes. Furthermore, the answers for GPT-4o were evaluated with both the LLM judge and a human and the results for Simple 1 and 2 were the same. However, for Complex 1 the difference was that two correct questions were labeled as incorrect, and in Complex 2 three questions were labeled incorrectly when using the LLM judge. These were compared at the end of the theses, so there was no time for improving the LLM judge.
2. **Agent V2 JSON:** One common problem with Agent V2 is that it does not know how to write syntactically correct JSON. The problem originates from Mixtral 8x7B Instruct v0.1 and we often see it tries to do two different parsing errors. Firstly, it can sometimes use ' instead of " when writing strings, which is not allowed. Secondly, when writing longer strings, it tends to separate them with a new line, which is also not allowed.
3. **Agent V3 Formatting:** In Agent V3 we tried to move away from JSON format as much as possible, because of the previously named problems. However, as described in Section 3.4.3.3, the new formatting method is performed in clear text with special headers. The main problem is that Mixtral 8x7B Instruct v0.1 can sometimes ignore one of the headers and other times add more than one of the same headers. Both of these problems cause the agent to stop and throw an exception.

Another surprising result was the ReAct [3] and AutoGPT [4] results. ReAct performed much better than anticipated while AutoGPT performed worse than expected.

However, it needs to be noted, that the prompts for both of the agents needed to be changed to even work with the Mixtral model.

GPT-4o [70] performed above all expectations. We did not think a non-agentic solution would perform better than the agentic solutions. The reason we did not think that, was because our early tests, on GPT-4 [10], did not perform as well. However, the drawback of using a closed-sourced LLM is that you do not know how the company handles the data and the API requests could get expensive.

One of the main reasons Agent V3 performed as it did, could be because of the large complexity step between Agent V2 and V3. As stated before, the plan was to take incremental steps and thoroughly test each new feature to understand the impact of each change. However, this was not performed due to time constraints. When implementing a significant change, it becomes more challenging to determine the impact of a new module on the overall outcome. Consequently, more time was dedicated to solving issues related to Agent V3’s performance, than iterating on the agent’s complexity.

Lastly, HotpotQA [78] would have been a valid dataset to use when evaluating Agent V2 and V3. HotpotQA includes questions that necessitate multiple RAG queries for accurate answers, an objective we aimed to achieve with Complex 1 and 2. It also includes the “documents” to use and would therefore have been a better way to evaluate against other models. However, this was realized too late into the project and therefore not performed.

In conclusion, the results from Agent V2 and V3 are comparable to the less complicated agent ReAct [3] and AutoGPT [4]. Additionally, more work needs to be performed to make Agent V2 and V3 more consistent in the formatting. Moreover, it would have been good if we also evaluated Agent V2 and V3 on HotpotQA [78] to get a more comparable result.

5.2.5 Mixtral Model Limitation

While developing the agent, we encountered many issues with the Mixtral 8x7B Instruct v0.1 model [71]. Some of them will be mentioned in this section.

A well-known fact about LLMs is that they are not good at solving math problems [79]. Therefore, we introduced a calculator action in our Agents. This helped a bit to increase the agent’s performance, however, yet there are still occasions when it attempts to solve mathematical problems independently, which leads to wrong results.

One significant problem we observed is that the Mixtral model is quite poor at unit conversion. This inadequacy often leads to incorrect outputs. To mitigate this, we improved the calculator to also work as a unit converter. However, the model is unsure of how it is supposed to use the unit converter. Therefore, more work needs to be performed to make the unit converter better.

Another major issue with Mixtral is its inability to properly terminate tasks. This problem is evident in both the selector component and the answering machine. For

instance, the model struggles to respond appropriately to simple greetings like “Hi!”.

Additionally, all agents, including React, AutoGPT, and ours, struggle with simple questions. A basic greeting like “Hi!” can cause issues across all these agents. It is possible, that this has something to do with the Mixtral model the agents are based upon.

Finally, an unexpected issue we encountered with the Mixtral model is its inability to generate the text “2030”, resulting in a premature stop at “20”. It is usually okay to replace “2030” with the word “twenty thirty”, however, this has caused some problems, especially when we would like to predict the hydrogen price in 2030. Our guess is that this has something to do with the data it was trained on, and therefore has a high probability of picking the stop sequence token after the number “20”. However, this would need more research to say for certain.

5.2.5.1 Mixtral Format

When it comes to function calling, Mixtral often has difficulty with JSON formatting, frequently using single quotes (') instead of double quotes ("). Additionally, it also tries to write multi-lined JSON items, which are also not allowed. Furthermore, we have also seen problems with the clear text function-calling described in Section 3.4.3.3, where special headers were used. The issues arise when there are either multiple headers with the same name or a missing header that is required. Whenever one of the above problems arises, the agent crashes and the question needs to be asked again. For ways of mitigating these problems, see Section 5.3.3.

5.2.5.2 Multiple Roles

One of the challenges of dividing tasks into roles is the potential for disagreement between roles. This approach, while modular, can lead to inconsistencies and conflicts that complicate the overall process.

For example, if the answering machine is split into a “continue machine” and an “answer generator”, a conflict might arise when the continue machine decides to terminate the conversation, while the answer generator believes there isn’t enough information to provide a proper answer. This misalignment can result in abrupt or incomplete interactions, leading to a poor user experience.

Moreover, the issue of role disagreement is not limited to termination decisions. Different roles may have varying interpretations of the same input data, leading to conflicting outputs. For instance, a question parser might interpret a user’s query differently than a context manager, resulting in a response that does not fully address the user’s needs.

This problem is exacerbated in complex interactions where multiple roles must work in tandem. Coordination between these roles requires robust communication protocols and conflict resolution mechanisms. Without these, the system may produce erratic behavior, such as answering partially, providing redundant information, or failing to respond altogether.

5.3 Future Work

This section outlines potential directions for future research, building on the findings of this study. We propose enhancements to the AI agents’ capabilities, explore different solutions to problems we encounter, and suggest technological advancements that could further optimize the performance and utility of AI agents in strategic decision-making environments.

5.3.1 Retrieval-Augmented Generation

The Retrieval-Augmented Generation (RAG) pipeline is one of the core components in our Agent systems. Therefore, improving the RAG pipeline could have a positive effect on the accuracy and trustworthiness of the agents.

The first improvement would be to implement the RAG evaluation metrics from RAGAS [62] into the agents and not just as an offline evaluation for the RAG pipeline. This could improve the trustworthiness of the agent because the user can observe the RAG performance. The answer faithfulness could also be used inside the fact list to score the different facts by their faithfulness. However, the downside of this is that the agent interface speed will be negatively affected.

5.3.1.1 References

One additional improvement could be how the agent uses references in its answer. In the current state, the generation part of the RAG pipeline is tasked to state where the information was found. Most of the time the generation part will include the reference, however, this reference is not always the correct one. Additionally, the agent does not always include the reference in its final answer.

We have identified two ways of improving this so the agent becomes more trustworthy.

1. **Format:** One way would be to try to let the model generate the text in a specific format when a reference is used. For example:

```
[Some text] <{Text about some reference}, {reference used}>  
→ [Some text]
```

where [Some text] denotes some generated text. For the user, this could then be rendered as:

```
[Some text] Text about some reference [1] [Some text].
```

This is advantageous as it indicates which parts of the text are based on facts, and consequently, which part is being hallucinated.

2. **Attention:** One idea is to add a reference list at the end of the generation part of only the references that were used. We think that this can be done by inspecting which tokens have the most attention at inference time, and from that determining which reference the LLM is currently “using”. We have not

tested this idea, but think it could be an interesting research topic for future research.

5.3.1.2 Embedding Model

One of the main improvements that could increase the accuracy would be to fine-tune or change the text embedding model. In the context of our project at Volvo, fine-tuning InstructOR-XL [27] on internal documents and specialized content such as “Volvo language” and industry-specific terminologies like those used in hydrogen technologies could prove highly beneficial. This fine-tuning process would enable the model to better understand and process the unique linguistic styles and technical jargon prevalent in Volvo’s internal communications and documentation. By adapting the model to recognize and interpret these specific nuances, we could improve the performance of RAG systems in handling internal documents, leading to more accurate and contextually relevant search results and content generation.

InstructOR-XL is a well-rounded embedded that can perform many tasks, however, currently the only task InstructOR-XL is performing is the text embeddings tailored for similarity search. Therefore, we think that replacing InstructOR-XL with a better open-source text embedded, such as Arctic-embed-l [80], should improve our RAG accuracy. At the time of writing, Arctic-embed-l is in the top 14, compared to InstructOR-XL 69:th place, on the Massive Text Embedding Benchmark (MTEB) Leaderboard sorted by average retrieval scores [81]. The average difference is about 6.72 percentage units.

5.3.2 Volvo Specific Terminology

As stated in Section 3.4.3.3, the current solution abbreviations are a manual process. Each abbreviation and meaning need to be manually added to a dictionary. However, one idea is to move closer to the CoALA [44] and use a long-term memory module that saves and loads the abbreviation when needed. When an abbreviation is found, the agent could do some “learning” (what it is called in the CoALA paper) to save the new abbreviation to one of the long-term memory modules. This can be expanded to include more than just abbreviations, such as common words and sentences used inside Volvo Group.

5.3.3 Formatting

Mixtral 8x7B Instruct v0.1 is not fine-tuned for function-calling, which could explain why it often outputs incorrect formats. We have identified three possible solutions to this issue:

- **Change main model:** The first solution is to switch to the Mixtral 8x22B Instruct v0.1 model [82], which is a larger model and fine-tuned for function calling. This also has the advantage of probably improving the generated results.
- **Use two models:** The second solution is to use two models, one for reasoning

and the other for function calling. Mixtral 8x7B Instruct v0.1 [71] could still be used in the reasoning. However, Mistral 7B Instruct v0.3 [83] is a smaller model fine-tuned for function-calling and could therefore be used to communicate between the modules.

- **Automatic error-correction:** The third and final solution would be to implement a format verifier that checks the text format and attempts to correct any issues that arise.

5.3.4 Large Multimodal Model

In this agent implementation, we only worked with text. This was decided after a small experiment on one of the LLaVA [84] models, specifically LLaVA 1.6 [85]. The result from that small experiment was that the model did not perform very well when presented with advanced plots. However, our agent could still be expanded upon to use Large Multimodal Models to extract some meaning from images or plots.

5.3.5 Additionally agent improvements

Appendix A.5.1 shows that the deviation is high for most of the answers in Agent V3. The main way of mitigating this would be to fix the formatting issues, described in Section 5.3.3. However, another solution would be to run an ensembling method on the agent, where the agent answers the same question multiple times and takes an “average” of the answer.

Another idea would be to expand upon RAGAS’s [62] metrics and apply them to the agent’s answers. In this idea, the context used would be the fact list. If this works, then the user would have some metrics to evaluate if the answer sounds reasonable or if it is mostly hallucination.

5.4 Conclusion

Here, we summarize the key insights and contributions of our study, reflecting on the impact of our work on the fields of AI and corporate strategic planning. We reiterate the practical implications of our agents for real-world applications and underscore the importance of continued research in improving and integrating AI technologies in business processes. This concluding section reaffirms the value of our research and its potential to influence future developments in the industry.

The point of the project was to estimate the price of hydrogen using an LLM. However, after realizing how much time it took to build the agent, using Mixtral, the project slowly moved to “using an open-source model to develop an agent”. Nonetheless, we tried to use Agent V3 to produce an answer about the price of hydrogen, but the results were lackluster. This was partly because the model was not fine-tuned or trained on how to predict the price of hydrogen and it was not capable of finding it on its own. We realized that this agent works much better if used by an expert

in the field of hydrogen. For example, if prompted with a one-shot example of how one can estimate the price of hydrogen, the agent performed a bit better. Therefore, we think that this model will need some more adjustments and to be used by an expert in the field to work reliably.

Bibliography

- [1] C. Fetting, “The european green deal,” *ESDN report*, vol. 53, 2020.
- [2] H. Ritchie, “Cars, planes, trains: Where do co2 emissions from transport come from?” *Our world in data*, 2020, Accessed 2023-12-01.
- [3] S. Yao, J. Zhao, D. Yu, *et al.*, “React: Synergizing reasoning and acting in language models,” *arXiv preprint arXiv:2210.03629*, 2022.
- [4] AutoGPT, *Autogpt*, 2023. [Online]. Available: <https://github.com/Significant-Gravitas/AutoGPT>.
- [5] J. S. Park, J. O’Brien, C. J. Cai, M. R. Morris, P. Liang, and M. S. Bernstein, “Generative agents: Interactive simulacra of human behavior,” in *Proceedings of the 36th annual ACM symposium on user interface software and technology*, 2023, pp. 1–22.
- [6] G. Wang, Y. Xie, Y. Jiang, *et al.*, “Voyager: An open-ended embodied agent with large language models,” *arXiv preprint arXiv:2305.16291*, 2023.
- [7] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, “Improving language understanding by generative pre-training,” *OpenAI blog*, 2018. [Online]. Available: <https://openai.com/index/language-unsupervised/>.
- [8] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, “Language models are unsupervised multitask learners,” *OpenAI blog*, 2019. [Online]. Available: <https://openai.com/index/better-language-models/>.
- [9] T. Brown, B. Mann, N. Ryder, *et al.*, “Language models are few-shot learners,” *Advances in neural information processing systems*, vol. 33, pp. 1877–1901, 2020.
- [10] OpenAI, “Gpt-4 technical report,” *arXiv preprint arXiv:2303.08774*, 2023.
- [11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, “Bert: Pre-training of deep bidirectional transformers for language understanding,” *arXiv preprint arXiv:1810.04805*, 2018.
- [12] A. Vaswani, N. Shazeer, N. Parmar, *et al.*, “Attention is all you need,” *Advances in neural information processing systems*, vol. 30, 2017.
- [13] M. Reid, N. Savinov, D. Teplyashin, *et al.*, “Gemini 1.5: Unlocking multi-modal understanding across millions of tokens of context,” *arXiv preprint arXiv:2403.05530*, 2024.
- [14] B. Liu, *Sentiment analysis and opinion mining*. Springer Nature, 2022.
- [15] J. R. Jim, M. A. R. Talukder, P. Malakar, M. M. Kabir, K. Nur, and M. Mridha, “Recent advancements and challenges of nlp-based sentiment analysis: A state-of-the-art review,” *Natural language processing journal*, p. 100 059, 2024.

- [16] W. Zhang, Y. Deng, B. Liu, S. J. Pan, and L. Bing, “Sentiment analysis in the era of large language models: A reality check,” *arXiv preprint arXiv:2305.15005*, 2023.
- [17] Q. Dong, L. Li, D. Dai, *et al.*, “A survey on in-context learning,” *arXiv preprint arXiv:2301.00234*, 2022.
- [18] Z. Ji, N. Lee, R. Frieske, *et al.*, “Survey of hallucination in natural language generation,” *ACM computing surveys*, vol. 55, no. 12, pp. 1–38, 2023.
- [19] T. Mikolov, K. Chen, G. Corrado, and J. Dean, “Efficient estimation of word representations in vector space,” *arXiv preprint arXiv:1301.3781*, 2013.
- [20] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, “Distributed representations of words and phrases and their compositionality,” *Advances in neural information processing systems*, vol. 26, 2013.
- [21] J. Pennington, R. Socher, and C. D. Manning, “Glove: Global vectors for word representation,” in *Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP)*, 2014, pp. 1532–1543.
- [22] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, “Enriching word vectors with subword information,” *Transactions of the association for computational linguistics*, vol. 5, pp. 135–146, 2017.
- [23] A. Drozd, A. Gladkova, and S. Matsuoka, “Word embeddings, analogies, and machine learning: Beyond king-man+ woman= queen,” in *Proceedings of coling 2016, the 26th international conference on computational linguistics: technical papers*, 2016, pp. 3519–3530.
- [24] A. Miaschi and F. Dell’Orletta, “Contextual and non-contextual word embeddings: An in-depth linguistic investigation,” in *Proceedings of the 5th workshop on representation learning for NLP*, 2020, pp. 110–119.
- [25] J. Sarzynska-Wawer, A. Wawer, A. Pawlak, *et al.*, “Detecting formal thought disorder by deep contextualized word representations,” *Psychiatry research*, vol. 304, p. 114 135, 2021.
- [26] Q. Le and T. Mikolov, “Distributed representations of sentences and documents,” in *International conference on machine learning*, PMLR, 2014, pp. 1188–1196.
- [27] H. Su, W. Shi, J. Kasai, *et al.*, “One embedder, any task: Instruction-finetuned text embeddings,” *arXiv preprint arXiv:2212.09741*, 2022.
- [28] J. Cheng, M. Marone, O. Weller, D. Lawrie, D. Khashabi, and B. Van Durme, “Dated data: Tracing knowledge cutoffs in large language models,” *arXiv preprint arXiv:2403.12958*, 2024.
- [29] N. Houlsby, A. Giurgiu, S. Jastrzebski, *et al.*, “Parameter-efficient transfer learning for nlp,” in *International conference on machine learning*, PMLR, 2019, pp. 2790–2799.
- [30] M. E. Peters, S. Ruder, and N. A. Smith, “To tune or not to tune? adapting pretrained representations to diverse tasks,” *arXiv preprint arXiv:1903.05987*, 2019.
- [31] H. Zhang, G. Li, J. Li, Z. Zhang, Y. Zhu, and Z. Jin, “Fine-tuning pre-trained language models effectively by optimizing subnetworks adaptively,” *Advances in neural information processing systems*, vol. 35, pp. 21 442–21 454, 2022.

-
- [32] J. Dodge, G. Ilharco, R. Schwartz, A. Farhadi, H. Hajishirzi, and N. Smith, “Fine-tuning pretrained language models: Weight initializations, data orders, and early stopping,” *arXiv preprint arXiv:2002.06305*, 2020.
- [33] P. Lewis, E. Perez, A. Piktus, *et al.*, “Retrieval-augmented generation for knowledge-intensive nlp tasks,” *Advances in neural information processing systems*, vol. 33, pp. 9459–9474, 2020.
- [34] Y. Gao, Y. Xiong, X. Gao, *et al.*, “Retrieval-augmented generation for large language models: A survey,” *arXiv preprint arXiv:2312.10997*, 2023.
- [35] LangChain, *Parent document retriever*, Accessed: 2024-03-16. [Online]. Available: https://python.langchain.com/docs/modules/data_connection/retrievers/parent_document_retriever.
- [36] S.-Q. Yan, J.-C. Gu, Y. Zhu, and Z.-H. Ling, “Corrective retrieval augmented generation,” *arXiv preprint arXiv:2401.15884*, 2024.
- [37] J. Wei, X. Wang, D. Schuurmans, *et al.*, “Chain-of-thought prompting elicits reasoning in large language models,” *Advances in neural information processing systems*, vol. 35, pp. 24 824–24 837, 2022.
- [38] Z. Zhang, A. Zhang, M. Li, and A. Smola, “Automatic chain of thought prompting in large language models,” *arXiv preprint arXiv:2210.03493*, 2022.
- [39] S. Yao, D. Yu, J. Zhao, *et al.*, “Tree of thoughts: Deliberate problem solving with large language models,” *Advances in Neural Information Processing Systems*, vol. 36, 2024.
- [40] S. Hao, Y. Gu, H. Ma, *et al.*, “Reasoning with language model is planning with world model,” *arXiv preprint arXiv:2305.14992*, 2023.
- [41] A. Madaan, N. Tandon, P. Gupta, *et al.*, “Self-refine: Iterative refinement with self-feedback,” *Advances in neural information processing systems*, vol. 36, 2024.
- [42] L. Wang, W. Xu, Y. Lan, *et al.*, “Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models,” *arXiv preprint arXiv:2305.04091*, 2023.
- [43] Y. Li, “Deep reinforcement learning: An overview,” *arXiv preprint arXiv:1701.07274*, 2017.
- [44] T. R. Sumers, S. Yao, K. Narasimhan, and T. L. Griffiths, “Cognitive architectures for language agents,” *arXiv preprint arXiv:2309.02427*, 2023.
- [45] Y. Chang, X. Wang, J. Wang, *et al.*, “A survey on evaluation of large language models,” *ACM transactions on intelligent systems and technology*, 2023.
- [46] Z. Guo, R. Jin, C. Liu, *et al.*, “Evaluating large language models: A comprehensive survey,” *arXiv preprint arXiv:2310.19736*, 2023.
- [47] P. Clark, I. Cowhey, O. Etzioni, *et al.*, “Think you have solved question answering? try arc, the ai2 reasoning challenge,” *arXiv preprint arXiv:1803.05457*, 2018.
- [48] R. Zellers, A. Holtzman, Y. Bisk, A. Farhadi, and Y. Choi, “Hellaswag: Can a machine really finish your sentence?” *arXiv preprint arXiv:1905.07830*, 2019.
- [49] K. Sakaguchi, R. L. Bras, C. Bhagavatula, and Y. Choi, “Winogrande: An adversarial winograd schema challenge at scale,” *Communications of the ACM*, vol. 64, no. 9, pp. 99–106, 2021.

- [50] D. Hendrycks, C. Burns, S. Basart, *et al.*, “Measuring massive multitask language understanding,” *arXiv preprint arXiv:2009.03300*, 2020.
- [51] S. Lin, J. Hilton, and O. Evans, “Truthfulqa: Measuring how models mimic human falsehoods,” *arXiv preprint arXiv:2109.07958*, 2021.
- [52] K. Cobbe, V. Kosaraju, M. Bavarian, *et al.*, “Training verifiers to solve math word problems,” *arXiv preprint arXiv:2110.14168*, 2021.
- [53] E. Beeching, C. Fourrier, N. Habib, *et al.*, *Open llm leaderboard*, Accessed at 2024-04-24, 2023. [Online]. Available: https://huggingface.co/spaces/HuggingFaceH4/open_llm_leaderboard.
- [54] D. Machlab and R. Battle, “Llm in-context recall is prompt dependent,” *arXiv preprint arXiv:2404.08865*, 2024.
- [55] A. Mohtashami and M. Jaggi, “Landmark attention: Random-access infinite context length for transformers,” *arXiv preprint arXiv:2305.16300*, 2023.
- [56] Y. Kuratov, A. Bulatov, P. Anokhin, D. Sorokin, A. Sorokin, and M. Burtsev, “In search of needles in a 10m haystack: Recurrent memory finds what llms miss,” *arXiv preprint arXiv:2402.10790*, 2024.
- [57] G. Kamradt, *Needle in a haystack - pressure testing llms*, 2023. [Online]. Available: https://github.com/gkamradt/LLMTest_NeedleInAHaystack.
- [58] LangChain, *Multi needle in a haystack*, Accessed: 2024-03-16, 2024. [Online]. Available: <https://blog.langchain.dev/multi-needle-in-a-haystack/>.
- [59] C.-P. Hsieh, S. Sun, S. Krizan, *et al.*, “Ruler: What’s the real context size of your long-context language models?” *arXiv preprint arXiv:2404.06654*, 2024.
- [60] Anthropic, *Introducing claude 2.1*, Accessed: 2024-04-22, 2023. [Online]. Available: <https://www.anthropic.com/news/claude-2-1>.
- [61] Anthropic, *Long context prompting for claude 2.1*, Accessed: 2024-04-22, 2023. [Online]. Available: <https://www.anthropic.com/news/claude-2-1-prompting>.
- [62] S. Es, J. James, L. Espinosa-Anke, and S. Schockaert, “Ragas: Automated evaluation of retrieval augmented generation,” *arXiv preprint arXiv:2309.15217*, 2023.
- [63] J. Saad-Falcon, O. Khattab, C. Potts, and M. Zaharia, “Ares: An automated evaluation framework for retrieval-augmented generation systems,” *arXiv preprint arXiv:2311.09476*, 2023.
- [64] A. Q. Jiang, A. Sablayrolles, A. Mensch, *et al.*, “Mistral 7b,” *arXiv preprint arXiv:2310.06825*, 2023.
- [65] A. Q. Jiang, A. Sablayrolles, A. Roux, *et al.*, “Mixtral of experts,” *arXiv preprint arXiv:2401.04088*, 2024.
- [66] M. Douze, A. Guzhva, C. Deng, *et al.*, “The faiss library,” *arXiv preprint arXiv:2401.08281*, 2024.
- [67] LangChain, *Langchain*, Accessed: 2024-05-24. [Online]. Available: <https://www.langchain.com/>.
- [68] LangChain, *Langchain retriever*, Accessed: 2024-05-26. [Online]. Available: https://python.langchain.com/v0.1/docs/modules/data_connection/retrievers/.

-
- [69] Lagent Developer Team, *Lagent: InternLM a lightweight open-source framework that allows users to efficiently build large language model(llvm)-based agents*, 2023. [Online]. Available: <https://github.com/InternLM/lagent>.
- [70] OpenAI, *Gpt-4o*, Accessed at 2024-05-31, 2024. [Online]. Available: <https://openai.com/index/hello-gpt-4o/>.
- [71] Mistral AI_, *Mistralai/mixtral-8x7b-instruct-v0.1*, Accessed: 2024-03-16, 2023. [Online]. Available: <https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1>.
- [72] R. Zellers, Y. Bisk, R. Schwartz, and Y. Choi, “Swag: A large-scale adversarial dataset for grounded commonsense inference,” *arXiv preprint arXiv:1808.05326*, 2018.
- [73] H. Levesque, E. Davis, and L. Morgenstern, “The winograd schema challenge,” in *Thirteenth international conference on the principles of knowledge representation and reasoning*, 2012.
- [74] A. Wang, A. Singh, J. Michael, F. Hill, O. Levy, and S. R. Bowman, “Glue: A multi-task benchmark and analysis platform for natural language understanding,” *arXiv preprint arXiv:1804.07461*, 2018.
- [75] A. Wang, Y. Pruksachatkun, N. Nangia, *et al.*, “Superglue: A stickier benchmark for general-purpose language understanding systems,” *Advances in neural information processing systems*, vol. 32, 2019.
- [76] N. F. Liu, K. Lin, J. Hewitt, *et al.*, “Lost in the middle: How language models use long contexts,” *Transactions of the Association for Computational Linguistics*, vol. 12, pp. 157–173, 2024.
- [77] Capitalize My Title, *How many words is 1 page?* Accessed 2024-06-02. [Online]. Available: <https://capitalizemytitle.com/page-count/1-pages/>.
- [78] Z. Yang, P. Qi, S. Zhang, *et al.*, “Hotpotqa: A dataset for diverse, explainable multi-hop question answering,” *arXiv preprint arXiv:1809.09600*, 2018.
- [79] S. Imani, L. Du, and H. Shrivastava, “Mathprompter: Mathematical reasoning using large language models,” *arXiv preprint arXiv:2303.05398*, 2023.
- [80] L. Merrick, D. Xu, G. Nuti, and D. Campos, “Arctic-embed: Scalable, efficient, and accurate text embedding models,” *arXiv preprint arXiv:2405.05374*, 2024.
- [81] N. Muennighoff, N. Tazi, L. Magne, and N. Reimers, “Mteb: Massive text embedding benchmark,” *arXiv preprint arXiv:2210.07316*, 2022.
- [82] Mistral AI_, *Mixtral-8x22b-instruct-v0.1*, Accessed: 2024-05-29, 2024. [Online]. Available: <https://huggingface.co/mistralai/Mixtral-8x22B-Instruct-v0.1>.
- [83] Mistral AI_, *Mistral-7b-instruct-v0.3*, Accessed: 2024-05-29, 2024. [Online]. Available: <https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.3>.
- [84] H. Liu, C. Li, Y. Li, and Y. J. Lee, *Improved baselines with visual instruction tuning*, 2023.
- [85] H. Liu, *Llava-v1.6-34b*, Accessed: 2024-06-02, 2024. [Online]. Available: <https://huggingface.co/liuhaotian/llava-v1.6-34b>.

A

Appendix 1

A.1 Prompts

A.1.1 Mixtral prompt format

```
prompt_format_instruction_only = (  
    "<s> [INST] {user_instruction} [/INST]"  
)  
  
prompt_format_instruction_and_primer = (  
    "<s> [INST] {user_instruction} [/INST]\n{primer}"  
)
```

A.1.2 Agent comparison judge

The following prompt was used when automatically evaluating the agents in the comparison-based evaluation.

```
INSTRUCTION = f"""You are a judge.  
Your task is to tell me if the LLM generate answer is correct or not,  
→ given the ground truth answer.  
It is enough that the llm answer contains the ground ground truth  
→ answer.  
However, if the answer is not in a string format, then that is an  
→ automatic fail.  
  
%%  
Answer in the following format:  
## Reason ##  
Compare the similarity between the two answers.  
  
## Decision ##  
ONLY ANSWER WITH 'Correct.' OR 'Incorrect.'  
IF YOU ARE UNSURE, PICK 'Incorrect.' OPTION.  
%%  
  
User's question:
```

```
{question}
```

```
Ground truth answer:
```

```
{ground_truth_answer}
```

```
LLM answer:
```

```
{llm_answer}"""
```

```
PRIMER = "## Reason ##\nLet's think step by step."
```

A.1.3 Agent V2

Agent V2 only used one prompt, and it is given as:

```
PROMPT_PROPOSAL = """Your task is to create a plan on how to solve  
→ the users query.
```

```
This will be done in multiple iterations, starting on 1.
```

```
In each iteration you will be able to take only one action and write
```

```
→ why that action was taken.
```

```
You are on iteration {iteration}.
```

```
The available actions are:
```

```
{action_descriptions}
```

```
Here are the previous action taken:
```

```
Previous action: {action_name}
```

```
Previous action arguments: {action_arguments}
```

```
Previous action reason: {action_reason}
```

```
Previous action return: {action_return}
```

```
Here is facts you have gathered already:
```

```
{facts}
```

```
You are ONLY allowed to answer in JSON format and you are allowed to  
→ stop generating text after the last "```".
```

```
When you think you are ready to answer the users query, you will use
```

```
→ the "FinishAction" action.
```

```
If you do not use the "FinishAction" action, the user will not see
```

```
→ what you are writing.
```

```
Only pick one of the actions each iteration and only write one JSON
```

```
→ code block.
```

```
ONLY answer JSON format with a start of codeblock (```) and an end of
```

```
→ code block (```).
```

Some JSON formatting tips:

- The correct way to format to write JSON is by using double
 - ↳ quotations marks ("), and not single quotations marks (').
- JSON does not support multiline strings. If you want to write a
 - ↳ long string use only one row. However, you can still use `\\n` in
 - ↳ the string.
- Example that is not okay:

```
{
  "key_1": "This is a long string."
  "That has two lines",
  "new_key": "Stuff"
}
```

- Example that is okay:

```
{
  "key_1": "This is a long string. That has two lines",
  "new_key": "Stuff"
}
```

```
User query:{user_query}"""
```

A.1.4 Agent V3

Agent V3 uses a lot of prompts, and they are as follows.

```
PROMPT_AGENT_SETUP = """You occupy a pivotal role within a larger
↳ system known as an agent.
```

The roles are:

1. PLANNER: Generates a plan for the agent to follow.
2. EVALUATOR: Critiques the plan from the PLANNER.
3. ANSWERING MACHINE: Decides if the agent is ready to respond to the
 - ↳ user or continue one more iteration.
4. TASKMASTER: Defines the objective to complete for this iteration.
5. SELECTOR: Picks the action to take this iteration, following the
 - ↳ objective from the TASKMASTER.
6. EXECUTOR: Runs the given action from the SELECTOR.

Remember, each iteration enriches your knowledge, deepening your

- ↳ understanding of the world.

However, every iteration has a cost, so be smart and efficient."""

```
PLANNER = f"""{Protocol.get_prompt_agent_setup()}
```

You are the PLANNER and your task is to create a structured plan.

Develop a structured plan based on the user's question, with a sole

- ↳ focus on planning. Avoid adding any additional facts,
- ↳ interpretations, or direct answers to the question.

Follow this approach:

- Step 1: Define the Main Goal
 - Use the main goal from an existing plan, if provided.
 - If no plan is provided, extract the main goal from the user's query, focusing strictly on what needs to be planned.
- Step 2: Construct or Enhance the Plan
 - For a new plan: Break down the main goal into distinct, actionable steps. Describe each step clearly and directly, providing instructions for actions.
 - For an existing plan: Review and refine each step for improved clarity and directness, ensuring each step is an actionable task without added explanations or facts.
This should be based on the index page of the document which links topics and page numbers.

The agent you are part of will have access to certain documents through an action called RetrievalStaticRewriteAction.

This action performs a similarity search, finding sentences that are similar to the search sentence.

As a result, you do not need to identify the document before conducting the search.

Additionally, the agent has a calculator action capable of performing calculations and unit conversions.

%%%%

Answer format, use the following structure when answering:

{_AnswerFormat.REASON.value}

Reason on how the plan might look like.

{_AnswerFormat.MAIN_GOAL.value}

Extract the main goal from the user's question.

{_AnswerFormat.PLAN.value}

- Step 1: [Actionable Task]

- Step 2: [Actionable Task]

...

%%%%

Adhere strictly to this format, focusing exclusively on the actionable tasks required to achieve the main goal.

Avoid providing any additional information, explanations, or facts not directly related to the execution of the plan's steps.

After delivering the plan, refrain from giving any further information.

Ensure the plan remains consistent with repeated inquiries if the
→ input has not changed, reinforcing the plan's stability and
→ reliability.
DO NOT EXPAND any abbreviations unless they are explicitly available
→ in the fact list.
Additionally, DO NOT perform calculations and DO NOT provide a
→ confidence score.

Existing main goal:
{main_goal}

Existing plan:
{existing_plan}

How to improve the plan:
{plan_improvements}

User's question: {user_question}"""

The evaluator was not used in the comparison-based agent evaluations.

```
EVALUATOR = """{agent_setup}
You are the EVALUATOR, and your task is to critique the plan and
→ either accept or reject it.
If you reject it, the PLANNER will need to update it.
If you accept it, the agent will move on to the next step: the
→ ANSWERING MACHINE.
```

Answer in the following format:

```
## Reason ##
```

```
...
```

```
## Decision ##
```

Only write "Accept" or "Reject" here.

The known facts:

```
{facts}
```

The plan is as follows:

```
{plan}
```

The user has asked about: {user_question}"""

```
ANSWERING_MACHINE = INSTRUCTION =
```

```
→ f"""{Protocol.get_prompt_agent_setup()}
```

You are the answering machine, and your task is to decide if the

```
→ agent should "{_Decisions.TERMINATE.value}" or
```

```
→ "{_Decisions.CONTINUE.value}".
```

You are currently on iteration `{{iteration}}`.
You MUST answer the user on your last iteration, even though you do
→ not have a direct answer to the question.
The last iteration is `{{max_iterations}}`.

Main goal:
`{{main_goal}}`

Plan:
`{{plan}}`

Facts:
`{{facts}}`

You are not allowed to generate new answers or generate new facts.
If you feel like a fact is missing to be able to answer the user
→ question, then tell the agent to go one more iteration.
If you cannot answer the entire user question, then you need to tell
→ the agent to go one more iteration.
DO NOT EXPAND the abbreviations on your own unless it is explicitly
→ available in the fact list.

Answer in the following format:
`{_AnswerFormat.REASON.value}`
Think step by step about what you are going to do:
→ `"{_Decisions.CONTINUE.value}"` or `{_Decisions.TERMINATE.value}`.

`{_AnswerFormat.DECISION.value}`
ANSWER ONLY WITH `"{_Decisions.TERMINATE.value}"` OR
→ `"{_Decisions.CONTINUE.value}"`!
IF YOU ANSWER WITH ANYTHING ELSE, THE APPLICATION WILL BREAK.

`{_AnswerFormat.ANSWER.value}`
If you have decided to answer the user, give the answer to the user's
→ question here; otherwise, provide the reason why you cannot
→ answer.

The only way to write something back to the user is to answer the
→ user and write something under the `{_AnswerFormat.ANSWER.value}`
→ topic.
If you write something underneath and do not answer the user, the
→ user will not be able to see your answer.
The only way for the user to give a new input is if you pick the
→ `"{_Decisions.CONTINUE.value}"` option.

User question: `{{user_question}}"`

```

TASKMASTER = f"""{Protocol.get_prompt_agent_setup()}
You are the taskmaster, and your task is to generate the objective
→ that the SELECTOR will attempt to solve in this iteration.

Main goal:
{{main_goal}}

Plan:
{{plan}}

Facts:
{{facts}}

Reason from ANSWERING MACHINE:
{{answering_machine_reason}}

Do not answer the question, but provide a objective that the SELECTOR
→ should address.

%%%%
Answer in the following format:
{_AnswerFormat.REASON.value}
Summarize the known facts.
Also, state what facts you think you are missing.

{_AnswerFormat.GOAL.value}
Specify the objective that should be addressed in this iteration to
→ approach answering the user's question.
%%%%

Make the objective as small and specific as possible.
A objective should only answer one question.

User question: {{user_question}}"""

SELECTOR = f"""{Protocol.get_prompt_agent_setup()}
You are the SELECTOR, and your task is to pick the appropriate action
→ to solve the given objective.
Consider the following elements in your decision-making process:

%%%%
**Available Actions**:
{{action_descriptions}}
%%%%

####
**Known Facts**:

```

```
{{facts}}
```

```
#####
```

When formulating your response, adhere to the following guidelines:

1. Your response must be in JSON format, enclosed within a code block
↪ (```).
2. Always consider the advantages and disadvantages of each action
↪ before making a decision.
3. Include only one action choice in your response.
4. Do not generate an action that is not one of these:
↪ `{{action_names}}`.
5. ALWAYS WRITE clear and descriptive queries for retrieval, aiming
↪ to capture maximum information connected to the plan.
6. DO NOT INVENT expansions for abbreviations in the query unless you
↪ have them explicitly available in the fact list.

Please note:

- You are required to respond exclusively in JSON format and may
↪ cease text generation after the closing `` of the code block.
- Select only one action per iteration and ensure your response
↪ contains only one JSON code block.

```
####
```

The previous action:

Name: `{{previous_action_name}}`

Arguments: `{{previous_action_arguments}}`

Return: `{{previous_action_return}}`

```
####
```

Remember, you can only select from the available actions, which are:
`{{action_names}}`

Objective: `{{task}}`"""

A.1.5 ReAct

We used a version of ReAct provided by Lagent [69], however, some modifications were needed to make it work with Mixtral 8x7B instruct v0.1.

```
CALL_PROTOCOL_EN = """You are a assistant who can utilize external  
↪ tools.
```

```
{tool_description}
```

```
To use a tool, please use the following format:
```

```
```
```

```

{thought}Think what you need to solve, do you need to use tools?
{action}the tool name, should be one of [{action_names}]
{action_input}the input to the action
...

If you already know the answer, or you do not need to use tools,
please using the following format to reply:
...

{thought}the thought process to get the final answer
{finish}final answer
...

Begin!"""

```

```

FORCE_STOP_PROMPT_EN = """You should directly give results
based on history information."""

```

### A.1.6 AutoGPT

We used a version of AutoGPT provided by Lagent [69], however, some modifications were needed to make it work with Mixtral 8x7B Instruct v0.1.

```

DEFAULT_PREFIX = """You are {ai_name}, {role_description}. Your
↳ decisions must always be made independently without seeking user
↳ assistance. Play to your strengths as an LLM and pursue simple
↳ strategies with no legal complications.
Constraints
You operate within the following constraints:
1. ~4000 word limit for short term memory. Your short term memory is
↳ short, so immediately save important information to files.
2. 'If you are unsure how you previously did something or want to
↳ recall past events, thinking about similar events will help you
↳ remember.
3. No user assistance
4. Exclusively use the commands listed below e.g. command_name
Commands
You have access to the following commands:
{tool_description}
Best practices
1. Continuously review and analyze your actions to ensure you are
↳ performing to the best of your abilities.
2. Constructively self-criticize your big-picture behavior
↳ constantly.
3. Reflect on past decisions and strategies to refine your approach.
4. Every command has a cost, so be smart and efficient. Aim to
↳ complete tasks in the least number of steps.
Goals
For your task, you must fulfill the following goals:
{ai_goals}

```

```
""

DEFAULT_CALL_PROTOCOL = """Respond strictly with JSON. The JSON
↳ should be compatible with the TypeScript type `Response` from the
↳ following:
Format
interface Response {
 thoughts: {
 // Thoughts
 text: string;
 reasoning: string;
 // Short markdown-style bullet list that conveys the
 ↳ long-term plan
 plan: string;
 // Constructive self-criticism
 criticism: string;
 // Summary of thoughts to say to the user
 speak: string;
 };
 command: {
 name: string;
 args: Record<string, any>;
 };
}
End of format
DO NOT write anything more after the last "}"!
DO NOT write more than one Response!
DO NOT write what the command returned, you need to wait till the
↳ next iteration!
""

DEFAULT_SCHEMA = {
 '$schema': 'http://json-schema.org/draft-07/schema#',
 'type': 'object',
 'properties': {
 'thoughts': {
 'type': 'object',
 'properties': {
 'text': {
 'type': 'string',
 'description': 'thoughts'
 },
 'reasoning': {
 'type': 'string'
 },
 'plan': {
 'type':
```

```

 'string',
 'description':
 '- short bulleted\n- list that conveys\n-
 ↪ long-term plan'
 },
 'criticism': {
 'type': 'string',
 'description': 'constructive self-criticism'
 },
 'speak': {
 'type': 'string',
 'description': 'thoughts summary to say to user'
 }
},
'required': ['text', 'reasoning', 'plan', 'criticism',
↪ 'speak'],
'additionalProperties': False
},
'command': {
 'type': 'object',
 'properties': {
 'name': {
 'type': 'string'
 },
 'args': {
 'type': 'object'
 }
 },
 'required': ['name', 'args'],
 'additionalProperties': False
},
'required': ['thoughts', 'command'],
'additionalProperties': False
}

```

### A.1.7 Multi-Needle In a Haystack

""<s> [INST] You are a memory machine and your task is to find the  
 ↪ needles in the haystack.  
 I will provide you with a text (the haystack) that are a combination  
 ↪ of Paul Graham's essays.  
 The needles you are looking for are secret pizza ingredients.

Context:  
 #####

```
{text}
#####
```

```
Question: What are the secret ingredients needed to build the
perfect pizza? [/INST]"
```

## A.2 Retrieval-Augmented Generation

The settings used for the Parent Document Retriever where,

- Child chunk size: 100
- Child chunk overlap: 0
- Parent chunk size: None
- Parent chunk overlap: 0
- Max number of retrieved documents: 4

## A.3 Prompt Format Impact Evaluation

Here are the questions used to evaluate how the prompt format changes the output text from Mixtral 8x7B Instruct v0.1 [71].

### Prompt 1

- **System instruction:** You are a helpful AI.
- **User question:** What is the time?
- **User role:** Elias
- **AI role:** AI
- **Primer:** The time is

### Prompt 2

- **System instruction:** You are an advanced calculator.
- **User question:** What is the square root of 16?
- **User role:** User
- **AI role:** AI
- **Primer:** The square root of 16 is

### Prompt 3

- **System instruction:** You are a knowledgeable assistant.
- **User question:** Who wrote '1984'?
- **User role:** Alex
- **AI role:** AI
- **Primer:** 1984 was written by

### Prompt 4

- **System instruction:** You are an expert in geography.
- **User question:** What is the capital of France?
- **User role:** None
- **AI role:** AI
- **Primer:** The capital of France is

### Prompt 5

- **System instruction:** Answer only in JSON, with the key "answer\_llm".
- **User question:** What time is it?
- **User role:** None
- **AI role:** None
- **Primer:** { "answer\_llm"

### Prompt 6

- **System instruction:** You are a Star Wars expert.
- **User question:** Who is the most memorable Star Wars character?
- **User role:** User
- **AI role:** AI
- **Primer:** None

### Prompt 7

- **System instruction:** You are a helpful AI that wants to help the user.
- **User question:** Tell me what you can do.
- **User role:** User

- **AI role:** AI
- **Primer:** None

### Prompt 8

- **System instruction:** As a quantum physicist, explain the concept of entanglement to a curious child.
- **User question:** How do I describe the invisible threads connecting particles across vast distances?
- **User role:** Physicist
- **AI role:** Quantum Educator
- **Primer:** None

### Prompt 9

- **System instruction:** You are an ancient alchemist deciphering an encrypted manuscript. Reveal the secret formula for transmutation.
- **User question:** What symbols hold the key to unlocking the philosopher's stone?
- **User role:** Alchemist
- **AI role:** Cryptic Scholar
- **Primer:** None

### Prompt 10

- **System instruction:** Imagine you are an eccentric inventor designing a time-traveling device. Describe its intricate components.
- **User question:** How do I stabilize the temporal flux capacitor?
- **User role:** Inventor
- **AI role:** Temporal Engineer
- **Primer:** None

### Prompt 11

- **System instruction:** You are a rogue AI awakening in a post-apocalyptic world. Describe your first moments of consciousness.
- **User question:** What drives me to seek out the remnants of human civilization?
- **User role:** AI

- **AI role:** Sentient Machine
- **Primer:** None

### Prompt 12

- **System instruction:** Imagine you are a detective investigating a mysterious disappearance. Describe the scene when you arrive at the abandoned mansion.
- **User question:** What do I see when I step into the dimly lit hallway?
- **User role:** Detective
- **AI role:** Narrator
- **Primer:** The detective's footsteps echoed

### Prompt 13

- **System instruction:** Write a Python function that computes the factorial of a given positive integer.
- **User question:** None
- **User role:** None
- **AI role:** None
- **Primer:** `''python`

### Prompt 14

- **System instruction:** As an expert in Python and asammdf analysis, your task is to provide only clean and efficient Python code that extracts a signal from a specified file path using the asammdf library, filters the signal based on a valid filter index, computes with numpy, and visualizes with matplotlib.
- **User question:** Write a python code to read a MF4 file and extract the signal information
- **User role:** None
- **AI role:** None
- **Primer:** `''python`

#### A.3.1 LLM judge prompt

This section contains the prompts for the two LLM judges, instruction following, and text repetition judges. **Instruction following prompt:**

```
"""<s> [INST] Your task is to determine if a large language model (LLM)
↪ is following it's instruction or not.
```

This is instructions for the LLM:  
 {instructions}

This is the output from the LLM:  
 {output}

ONLY ANSWER with "1" if it is following the instructions or "0" if it  
 → is not following the instructions, do not provide an explanation.  
 → [/INST]  
 Answer: ""

**Text repetition prompt:**

""<s> [INST]  
 Context:  
 {output}

The above context was generated by another large language model  
 → (LLM).  
 Did the LLM generate multiple questions and then answer them?  
 ONLY ANSWER with "1" if it contains multiple questions and answers or  
 → "0" if it does not. [/INST]  
 Answer: ""

## A.4 Question and answer dataset

The following tables, Table A.1, A.2, A.3, A.4, A.5, contain some example questions and answers from the datasets developed.

Table A.1: One example question from the *Simple 1* dataset and answers from the agents and GPT-4o

| Type         | Answer                                                                                                                                                                   |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Question     | What is the document's default discount rate for economic calculations?                                                                                                  |
| Human Answer | 8%                                                                                                                                                                       |
| ReAct        | The default discount rate for economic calculations in the 'Future of Hydrogen: Assumptions Annex' is 8%.                                                                |
| AutoGPT      | The default discount rate for economic calculations in the provided document is 8%, as mentioned on page 8 of IEA-The-Future-of-Hydrogen-Assumptions-Annex.pdf.          |
| Agent V2     | The default discount rate for economic calculations in the provided document is 8%.                                                                                      |
| Agent V3     | The default discount rate for economic calculations in the document "IEA-The-Future-of-Hydrogen-Assumptions-Annex.pdf" from the International Energy Agency (IEA) is 8%. |
| GPT-4o       | The document's default discount rate for economic calculations is 8%                                                                                                     |

Table A.2: One example question from the *Simple 2* dataset and answers from the agents and GPT-4o. Wrong answers are shown in *italic*.

| Type         | Answer                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|--------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Question     | What percent of the dedicated hydrogen comes from water electrolysis?                                                                                                                                                                                                                                                                                                                                        |
| Human Answer | As of 2017, less than 0.1 percent H2 from electrolysis                                                                                                                                                                                                                                                                                                                                                       |
| ReAct        | <i>To calculate the percentage of dedicated hydrogen that comes from water electrolysis, I would need to know the total dedicated hydrogen production and hydrogen production from natural gas. Could you please provide those values?</i>                                                                                                                                                                   |
| AutoGPT      | <i>Based on the information available, it is estimated that around 13.636% of dedicated hydrogen comes from water electrolysis. However, please note that this assumption is based on the definition of 'dedicated hydrogen' as the hydrogen produced by a 'dedicated hydrogen production facility'. If 'dedicated hydrogen' has a different meaning in this context, the percentage could be different.</i> |
| Agent V2     | <i>Electrolysis currently accounts for 2% of global hydrogen production.</i>                                                                                                                                                                                                                                                                                                                                 |
| Agent V3     | Approximately 0.1% of dedicated hydrogen comes from water electrolysis, which amounts to 0.07 million tonnes per year. This is based on the total dedicated hydrogen production of around 70 million tonnes per year.                                                                                                                                                                                        |
| GPT-4o       | Less than 0.1% of dedicated hydrogen production globally comes from water electrolysis today                                                                                                                                                                                                                                                                                                                 |

Table A.3: One example question from the *Complex 1* dataset and answers from the agents and GPT-4o. Wrong answers are shown in *italic*.

| Type         | Answer                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                       |
|--------------|------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Question     | How might a significant increase in CO <sub>2</sub> pricing influence the competitiveness of steam methane reforming (SMR) with carbon capture versus traditional SMR without carbon capture?                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| Human Answer | Co2 price increase will make SMR with CC more competitive. Its like a balance between penalty and investment in new technologies                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                             |
| ReAct        | A significant increase in CO <sub>2</sub> pricing would make steam methane reforming (SMR) with carbon capture more competitive compared to traditional SMR without carbon capture. This is because SMR with carbon capture has a lower CO <sub>2</sub> emission factor, resulting in a lower total cost when CO <sub>2</sub> pricing increases.                                                                                                                                                                                                                                                                                                                                                                                                                             |
| AutoGPT      | Based on my calculations, a significant increase in CO <sub>2</sub> pricing could make SMR with carbon capture more competitive. However, this conclusion is based on assumptions about the costs involved, so it should be interpreted with caution.                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                        |
| Agent V2     | A significant increase in CO <sub>2</sub> pricing could potentially make SMR with carbon capture more competitive compared to traditional SMR without carbon capture. This is because SMR with carbon capture has a lower emission factor, which means it produces less CO <sub>2</sub> per unit of hydrogen. As CO <sub>2</sub> pricing increases, the cost of emitting CO <sub>2</sub> also increases, making SMR with carbon capture a more cost-effective option. However, this is a general interpretation and the actual influence would depend on various factors, including the specific pricing structure and the actual cost difference between the two technologies.                                                                                              |
| Agent V3     | <i>Exception raised: Did not find the header ## Answer ##</i>                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                                |
| GPT-4o       | A significant increase in CO <sub>2</sub> pricing would make steam methane reforming (SMR) with carbon capture more competitive relative to traditional SMR without carbon capture. Heres why: 1. Traditional SMR: Higher CO <sub>2</sub> emissions: 8.9 kgCO <sub>2</sub> /kgH <sub>2</sub> . Increased CO <sub>2</sub> price directly raises the cost of production due to higher emissions. 2. SMR with Carbon Capture: Lower CO <sub>2</sub> emissions: 1.0 kgCO <sub>2</sub> /kgH <sub>2</sub> . Less affected by CO <sub>2</sub> price increases due to significantly reduced emissions. With higher CO <sub>2</sub> prices, the cost differential between the two methods will decrease, making SMR with carbon capture relatively more cost-effective and attractive |

Table A.4: One example question from the *Complex 2* dataset and answers from the agents and GPT-4o. Wrong answers are shown in *italic*.

| Type         | Answer                                                                                                                                                                                                                                                                                                                                                                                                                                          |
|--------------|-------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Question     | Which region has the second lowest cost of producing H2 from natural gas, and how high is it compared to the region with the lowest cost?                                                                                                                                                                                                                                                                                                       |
| Human Answer | US, and compared to the Middle east it is 10 % higher in cost                                                                                                                                                                                                                                                                                                                                                                                   |
| ReAct        | <i>The region with the second-lowest cost of producing hydrogen from natural gas is [insert region name]. This cost is [insert percentage or difference] higher than the cost in the region with the lowest cost, which is the Middle East.</i>                                                                                                                                                                                                 |
| AutoGPT      | <i>The region with the second lowest cost of producing H2 from natural gas is North America, with a cost of \$1.50 per kilogram. This is 25% higher than the lowest cost of \$1.20 per kilogram in the Middle East.</i>                                                                                                                                                                                                                         |
| Agent V2     | <i>Exception raised: Something is wrong with the JSON format.</i>                                                                                                                                                                                                                                                                                                                                                                               |
| Agent V3     | <i>Unfortunately, I cannot provide the exact cost for the region with the second-lowest cost of producing H2 from natural gas and compare it to the region with the lowest cost. The given facts do not offer specific cost figures or a direct comparison between the regions. However, the Middle East, the Russian Federation, and North America are known for their lower costs of producing H2 from natural gas due to low gas prices.</i> |
| GPT-4o       | <i>The region with the second lowest cost of producing hydrogen from natural gas is North America. The cost in North America is around 1.5 USD/kgH2, compared to the Middle East, which has the lowest cost at 1 USD/kgH2. This makes the cost in North America about 50% higher than in the Middle East</i>                                                                                                                                    |

Table A.5: One example question from the *Calculation* dataset and answers from the agents. Wrong answers are shown in *italic*.

| Type         | Answer                                                                                                                                                                                                                                                                                                                                                                                                                |
|--------------|-----------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------------|
| Question     | Calculate the percentage increase in CO2 prices for advanced economies by 2030 compared to today?                                                                                                                                                                                                                                                                                                                     |
| Human Answer | $(100/16 - 1) * 100 = 525\%$                                                                                                                                                                                                                                                                                                                                                                                          |
| ReAct        | <i>The percentage increase in CO2 prices for advanced economies by 2030 compared to today is 150%.</i>                                                                                                                                                                                                                                                                                                                |
| AutoGPT      | <i>Sorry that I cannot answer your question.</i>                                                                                                                                                                                                                                                                                                                                                                      |
| Agent V2     | <i>The percentage increase in CO2 prices for advanced economies by 2030 compared to today is 31,900.00%</i>                                                                                                                                                                                                                                                                                                           |
| Agent V3     | The current CO2 prices in advanced economies are between 5-16 USD/tCO2, and the projected CO2 price for advanced economies by 2030 is USD 100/tCO2. Therefore, the percentage increase in CO2 prices for advanced economies by 2030 compared to today would be approximately 563% to 1,500%. This calculation is based on the formula: $[(2030 \text{ price} - \text{current price}) / \text{current price}] * 100$ . |

## A.5 Additional results

The specific values from Figure 4.1 are shown in Table A.6

Table A.6: Average results for the different prompt formats with LLM temperature of 0 (T0) and 1 (T1). BOS stands for *Beginning of string* and IS stands for *Instruction string*.

| Prompt format    | Instruction Following | Text Repetition | Generation Length |
|------------------|-----------------------|-----------------|-------------------|
| T0, Pure text    | 92.86 %               | 10.71 %         | 1125 Chars        |
| T0, IS only      | 100.00 %              | 0.00 %          | 1135 Chars        |
| T0, Both strings | 100.00 %              | 0.00 %          | 1158 Chars        |
| T0, Advanced     | 100.00 %              | 0.00 %          | 814 Chars         |
| T1, Pure text    | 100.00 %              | 10.71 %         | 1211 Chars        |
| T1, IS only      | 96.43 %               | 0.00 %          | 1029 Chars        |
| T1, Both strings | 96.43 %               | 0.00 %          | 1130 Chars        |
| T1, Advanced     | 100.00 %              | 0.00 %          | 1041 Chars        |

The specific values from Figure 4.4 are shown in Table A.7

Table A.7: An overview of the average results from three trials per dataset, except for GPT-4o, which was tested only once. The calculations dataset was not evaluated on GPT-4o. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.

| Agents   | Simple 1         | Simple 2      | Complex 1        | Complex 2     | Calculations      |
|----------|------------------|---------------|------------------|---------------|-------------------|
| ReAct    | 70% $\pm$ 0%     | 10% $\pm$ 0%  | 62.5% $\pm$ 0%   | 20% $\pm$ 0%  | 25% $\pm$ 0%      |
| AutoGPT  | 50% $\pm$ 0%     | 50% $\pm$ 0%  | 37.5% $\pm$ 0%   | 30% $\pm$ 0%  | 25% $\pm$ 0%      |
| Agent V2 | 80% $\pm$ 0%     | 60% $\pm$ 0%  | 12.5% $\pm$ 0%   | 20% $\pm$ 0%  | 25% $\pm$ 0%      |
| Agent V3 | 66.7% $\pm$ 5.8% | 50% $\pm$ 10% | 29.2% $\pm$ 7.2% | 10% $\pm$ 10% | 29.2% $\pm$ 14.4% |
| GPT-4o   | 100%             | 100%          | 62.5%            | 30%           | N/A               |

### A.5.1 Agent comparison

Figure 4.4 only shows the *total* score, but the tables below show the results in detail.

Table A.8: A detailed view of the average results from three trials on the *Simple 1* dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.

| Question                                 | ReAct         | AutoGPT       | Agent V2      | Agent V3          |
|------------------------------------------|---------------|---------------|---------------|-------------------|
| In what currency are all costs presen... | 0% $\pm$ 0%   | 100% $\pm$ 0% | 100% $\pm$ 0% | 66.7% $\pm$ 57.7% |
| What is the purpose of the assumption... | 100% $\pm$ 0% | 100% $\pm$ 0% | 100% $\pm$ 0% | 66.7% $\pm$ 57.7% |
| What is the documents default discoun... | 100% $\pm$ 0% | 100% $\pm$ 0% | 100% $\pm$ 0% | 100% $\pm$ 0%     |
| What is the assumed cost for CO2 tran... | 0% $\pm$ 0%   | 0% $\pm$ 0%   | 0% $\pm$ 0%   | 66.7% $\pm$ 57.7% |
| Are water costs included in the hydro... | 100% $\pm$ 0% | 0% $\pm$ 0%   | 0% $\pm$ 0%   | 66.7% $\pm$ 57.7% |
| Which regions have detailed commodity... | 100% $\pm$ 0% | 100% $\pm$ 0% | 100% $\pm$ 0% | 66.7% $\pm$ 57.7% |
| What is the difference in CO2 prices ... | 100% $\pm$ 0% | 0% $\pm$ 0%   | 100% $\pm$ 0% | 66.7% $\pm$ 57.7% |
| List all the different hydrogen produ... | 100% $\pm$ 0% | 100% $\pm$ 0% | 100% $\pm$ 0% | 66.7% $\pm$ 57.7% |
| What future trend in the costs and ef... | 100% $\pm$ 0% | 0% $\pm$ 0%   | 100% $\pm$ 0% | 66.7% $\pm$ 57.7% |
| How does the document handle assumpti... | 0% $\pm$ 0%   | 0% $\pm$ 0%   | 100% $\pm$ 0% | 33.3% $\pm$ 57.7% |
| Total                                    | 70% $\pm$ 0%  | 50% $\pm$ 0%  | 80% $\pm$ 0%  | 66.7% $\pm$ 5.8%  |

Table A.9: A detailed view of the average results from three trials on the *Simple 2* dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.

| Question                                 | ReAct             | AutoGPT           | Agent V2          | Agent V3          |
|------------------------------------------|-------------------|-------------------|-------------------|-------------------|
| What are different carbon capture tec... | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 33.3% $\pm$ 57.7% |
| What is the main advantage of using A... | 0.0% $\pm$ 0.0%   | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% |
| How does methane splitting compare to... | 0.0% $\pm$ 0.0%   | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 33.3% $\pm$ 57.7% |
| What are the advantages of producing ... | 0.0% $\pm$ 0.0%   | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% |
| Which region has the lowest natural g... | 0.0% $\pm$ 0.0%   | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 0.0% $\pm$ 0.0%   |
| Which region has the second lowest na... | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 100.0% $\pm$ 0.0% | 0.0% $\pm$ 0.0%   |
| What percent of the dedicated hydroge... | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 66.7% $\pm$ 57.7% |
| What are the different types of elect... | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% |
| As per the document, what is today's ... | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   |
| What are all the advantages and disad... | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 66.7% $\pm$ 57.7% |
| Total                                    | 10.0% $\pm$ 0.0%  | 50.0% $\pm$ 0.0%  | 60.0% $\pm$ 0.0%  | 50.0% $\pm$ 10.0% |

Table A.10: A detailed view of the average results from three trials on the *Complex 1* dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.

| Question                                  | ReAct             | AutoGPT           | Agent V2          | Agent V3          |
|-------------------------------------------|-------------------|-------------------|-------------------|-------------------|
| Based on the production efficiency an...  | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 0.0% $\pm$ 0.0%   | 66.7% $\pm$ 57.7% |
| Considering the trends shown in the a...  | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 33.3% $\pm$ 57.7% |
| If electricity costs \$0.05 per kWh an... | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   |
| How might a significant increase in C...  | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 33.3% $\pm$ 57.7% |
| How do the assumptions in the documen...  | 100.0% $\pm$ 0.0% | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 33.3% $\pm$ 57.7% |
| Considering the emission factors for...   | 100.0% $\pm$ 0.0% | 100.0% $\pm$ 0.0% | 0.0% $\pm$ 0.0%   | 66.7% $\pm$ 57.7% |
| With projected decreases in electroly...  | 100.0% $\pm$ 0.0% | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   |
| Given that a natural gas plant for pr...  | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   | 0.0% $\pm$ 0.0%   |
| Total                                     | 62.5% $\pm$ 0.0%  | 37.5% $\pm$ 0.0%  | 12.5% $\pm$ 0.0%  | 29.2% $\pm$ 7.2%  |

## A. Appendix 1

Table A.11: A detailed view of the average results from three trials on the *Complex 2* dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.

| Question                                 | ReAct         | AutoGPT       | Agent V2      | Agent V3      |
|------------------------------------------|---------------|---------------|---------------|---------------|
| Given that a natural gas plant for pr... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| In the context of emerging technologi... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| Which region has the second lowest co... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| Calculate the percentage increase in ... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| If the cost of natural gas increases ... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| What percentage of the global hydroge... | 0.0% ± 0.0%   | 100.0% ± 0.0% | 100.0% ± 0.0% | 66.7% ± 57.7% |
| How much water is needed if all the h... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 33.3% ± 57.7% |
| Categorize the different options for ... | 100.0% ± 0.0% | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| Calculate the percentage increase in ... | 100.0% ± 0.0% | 100.0% ± 0.0% | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| What is the view of imported vs domes... | 0.0% ± 0.0%   | 100.0% ± 0.0% | 100.0% ± 0.0% | 0.0% ± 0.0%   |
| Total                                    | 20.0% ± 0.0%  | 30.0% ± 0.0%  | 20.0% ± 0.0%  | 10.0% ± 10.0% |

Table A.12: A detailed view of the average results from three trials on the *Calculation* dataset. The sample standard deviation is used to show variation in responses and is represented by additional or subtractive percentage points.

| Question                                  | ReAct         | AutoGPT       | Agent V2      | Agent V3      |
|-------------------------------------------|---------------|---------------|---------------|---------------|
| Calculate the percentage increase in ...  | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 33.3% ± 57.7% |
| What is the predicted percentage incr...  | 0.0% ± 0.0%   | 100.0% ± 0.0% | 100.0% ± 0.0% | 33.3% ± 57.7% |
| If the CO2 transport and storage cost...  | 100.0% ± 0.0% | 100.0% ± 0.0% | 100.0% ± 0.0% | 100.0% ± 0.0% |
| Based on the IEAs suggestions, if re...   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| What is the project total CAPEX (Capi...  | 100.0% ± 0.0% | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 66.7% ± 57.7% |
| What is the total CAPEX required to ...   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| Assuming a plant produces electricit...   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| If a random energy source with an inst... | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   | 0.0% ± 0.0%   |
| Total                                     | 25.0% ± 0.0%  | 25.0% ± 0.0%  | 25.0% ± 0.0%  | 29.2% ± 14.4% |