ASSIGNMENT 2: PROMPT ENGINEERING FOR MEDICAL EXAM QUESTIONS

Ruiyan SUN 224040284
Department of Data Science
Shenzhen, China
224040284@link.cuhk.edu.cn

1 INTRODUCTION

Why the task is important The task of answering medical exam questions accurately is crucial for ensuring that candidates possess the requisite knowledge and skills for safe and effective practice. This task not only validates credentials but also safeguards public health, enables professional recognition, and ensures adherence to legal and regulatory standards.

Why LLM is suitable to solve the problem Large Language Models (LLMs) such as GPT-3.5 Turbo, GPT-4, and Claude 3 have demonstrated remarkable capabilities in understanding and generating human-like text. These models are trained on vast amounts of data, making them capable of performing tasks in a "zero-shot" manner. However, for more complex tasks, techniques like few-shot prompting, chain-of-thought prompting, tree of thoughts, and retrieval augmented generation can be used to enhance their performance.

What you did and what you achieved In this assignment, I explored four different prompting techniques to improve the performance of LLMs on medical exam questions. I implemented and compared the results of zero-shot prompting, few-shot prompting, chain-of-thought prompting, tree of thoughts, and retrieval augmented generation. I also compared the performance between non-agent and agent-based approaches, evaluating accuracy, completion time, reasoning quality, and task complexity handling.

2 PROBLEM DEFINITION

Definition of the task The task is to use LLMs to answer multiple-choice questions from a medical exam. The goal is to select the correct answer from the given options.The pharmacist licensure exam is a cornerstone in the pharmacy profession, ensuring that candidates possess the requisite knowledge and skills for safe and effective practice. Its significance lies not only in validating credentials but also in safeguarding public health, enabling professional recognition, and ensuring adherence to legal and regulatory standards.

Input: Given Multi-choice questions.

Output: Select the correct answer.

Criteria: Better accuracy.

Example: Refer to Figure 1.

Code and Data: Data description: 100 questions sampled from 2021 National Pharmacist Professional Qualification Examination real questions. Code and data:https://huggingface.co/datasets/yitingxie/rlhf-reward-datasets

Advanced models like DeepSeek have significant potential in exam preparation, boasting an extensive knowledge base and the capability to provide in-depth explanations and clarify complex concepts. However, despite the prowess of such large models, if prompts are not designed appropriately, the information retrieved might be inaccurate or incomplete, potentially hindering success in the pharmacist exam.

An example prompt for multi-choice questions

Prompt:

你是一个药剂师考试能手，每次都考100分，这道题对你来说不在话下，深呼吸，并一步一步思考，并给出正确的答案。回答格式为答案是A、B、C或者D。【最佳选择题】根据健康中国战略，推进健康中国建设主要遵循的原则不包括哪个选项?
A: “健康优先”，
B: “改革创新”，
C: “科学发展”，
D: “公开透明”

Expected Output:

根据健康中国战略，推进健康中国建设主要遵循的原则不包括选项
D: “公开透明”。

Figure 1: An example prompt for multi-choice questions.

3 PROMPTS AND THEIR DESIGN PHILOSOPHY

3.1 PHILOSOPHY OF THE DESIGNED PROMPTS

The overarching design philosophy for the prompts in Task 1 was to systematically compare and optimize the performance of Large Language Models (LLMs) using four distinct prompting techniques. The goal was to enhance both the accuracy and reasoning quality of the models by leveraging the unique strengths of each technique. Additionally, a comparative analysis between non-agent and agent-based approaches was conducted to evaluate the performance of LLM agents against traditional prompting methods.

3.1.1 PROMPTING TECHNIQUES

1. Zero-Shot Prompting
Zero-shot prompting serves as the baseline by directly instructing the LLM to perform a task without any additional examples. This method relies on the model's inherent ability to understand and generate responses based on its pre-trained knowledge.

2. Few-Shot Prompting
Few-shot prompting improves upon zero-shot by providing a small number of examples to guide the LLM. These examples act as a form of in-context learning, helping the model to better understand the task and generate more accurate responses.

3. Chain-of-Thought Prompting
Chain-of-thought prompting introduces intermediate reasoning steps to enable complex reasoning. By breaking down the problem into smaller, manageable parts, this technique enhances the model's ability to handle intricate tasks.

4. Tree of Thoughts (ToT) Prompting
ToT prompting generalizes chain-of-thought by maintaining a tree structure of potential solutions. This method allows the model to explore multiple paths and evaluate different solutions systematically, improving its problem-solving capabilities.

3.1.2 NON-AGENT VS. AGENT-BASED APPROACHES

1. Non-Agent Approach
The non-agent approach is a relatively simple way of using large language models (LLMs). In this approach, **we rely entirely on the LLM's built-in capabilities to generate responses.** When

we provide a prompt to the LLM, it processes the text based on its **pre-trained knowledge and algorithms**. For example, if we ask it a question like in Task 1 of this assignment (such as a question from the pharmacist licensure exam), the LLM tries to find the most relevant answer from its knowledge base.

The principle behind this is that the LLM has been trained on a vast amount of text data. It has learned language patterns, semantic relationships, and a wide range of knowledge during the training process. When it receives a prompt, it analyzes the words, understands the context to some extent, and then generates a response. However, this method has limitations. For complex tasks that require multiple steps of reasoning, access to external knowledge sources, or adaptability to different situations, the non-agent approach may not perform well. It lacks the flexibility to handle dynamic and complex scenarios because it can **only rely on its pre-trained knowledge and cannot actively seek additional information or make decisions based on real-time requirements.**

2. Agent-Based Approach

The agent-based approach leverages the capabilities of an intelligent agent in conjunction with the LLM. This approach is implemented using **LangChain**, a comprehensive framework for developing applications driven by language models. LangChain offers a suite of tools and components for constructing and managing interactions with large language models, enhancing the overall performance of the system.

a. Framework

LangChain: This framework is pivotal in building applications powered by language models. It provides a plethora of tools and components for building and managing interactions with large language models. In our implementation, multiple modules and functions of LangChain are utilized, including agents, hubs, and tools.

b. Core Technologies and Components

Language Model ChatOpenAI: Utilizing OpenAI's *gpt-3.5-turbo* model as the core language model, configured by setting *OPENAI_API_KEY* and *OPENAI_BASE_URL*. An example of utilizing *ChatDeepSeek* is provided in the code, but currently *ChatOpenAI* is employed.

Tools Retrieval Tool: Employing the FAISS vector database to store vector representations of medical documents, and generate text embeddings via *OpenAIEmbeddings*. A retrieval tool named *medical_document_retriever* is instantiated through *create_retriever_tool*, facilitating the retrieval of pertinent information from medical documents.

Search Tool: Utilizing *DuckDuckGoSearchResults* as the search tool to seek relevant information on the Internet. An example leveraging *TavilySearchResults* is also included in the code, but *DuckDuckGoSearchResults* is the current tool of choice.

The agent in LangChain is capable of calling various tools. For instance, it can access external knowledge sources through the retrieval system **using the FAISS vector database**. The FAISS vector database stores vectors representing text data. When the agent requires information, it can search this database to find relevant text passages. This is particularly useful when the LLM's internal knowledge is not sufficient.

Another key feature is **the integration with the DuckDuckGo API** for web search. If the agent needs up-to-date or more comprehensive information, it can utilize the web search functionality to gather data from the internet.

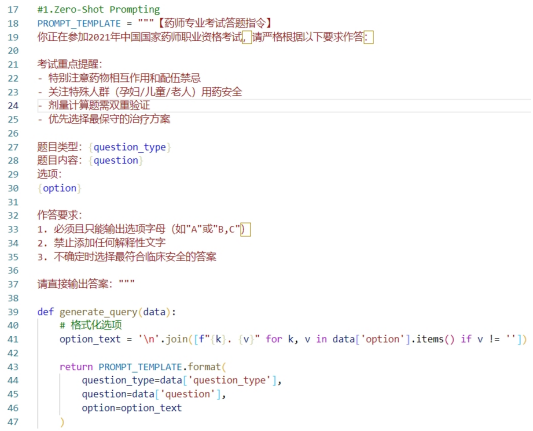
The agent is also capable of making decisions. When presented with a task, it can analyze the requirements, choose the appropriate tools to use, and determine the best course of action. For example, in a complex question-answering task, it might first search the local knowledge **base in the FAISS vector database**. If it doesn't find a satisfactory answer, it can then use the web search through the DuckDuckGo API. This adaptability and decision-making ability lead to more accurate and reliable results compared to the non-agent approach.

3.2 SOME EXAMPLE PROMPTS

There are some Figures for some example prompts.

1.Zero-Shot Prompting

The code of 1.prepare data.py optimized using the Zero-Shot Prompting technique is shown as follows:

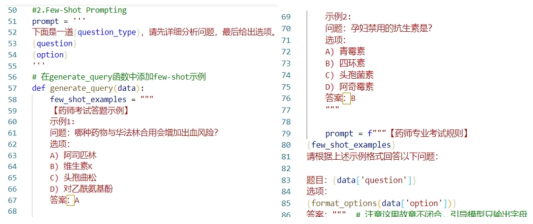


```
#1.Zero-Shot Prompting
PROMPT_TEMPLATE = """【药师专业考试答题指令】
你正在参加2021年中国国家药师职业资格考试，请严格根据以下要求作答：

考试重点提醒：
- 特别注意药物相互作用和配伍禁忌
- 关注特殊人群（孕妇/儿童/老人）用药安全
- 剂量计算题需双重验证
- 优先选择最保守的治疗方案

题目类型：{question_type}
题目内容：{question}
选项：
{option}

作答要求：
1. 必须且只能输出选项字母（如"A"或"B,C"）
2. 禁止添加任何解释性文字
3. 不确定时选择最符合临床安全的答案

请直接输出答案："""

def generate_query(data):
    # 格式化选项
    option_text = '\n'.join([f"{k}. {v}" for k, v in data['option'].items() if v != ''])

    return PROMPT_TEMPLATE.format(
        question_type=data['question_type'],
        question=data['question'],
        option=option_text
    )
```

Figure 2: the Zero-Shot Prompting.

2. Few-Shot Prompting

The code of 1.prepare data.py optimized using the Few-Shot Prompting technique is shown as follows:



```
#2.Few-Shot Prompting
prompt = """
下面是一道{question_type}，请先详细分析问题，最后给出选项。
{question}
{option}
"""
# 在generate_query函数中添加few-shot示例
def generate_query(data):
    few_shot_examples = """
    【药师考试答题示例】
    示例1：
    问题：哪种药物与华法林合用会增加出血风险？
    选项：
    A) 阿司匹林
    B) 维生素K
    C) 头孢曲松
    D) 对乙酰氨基酚
    答案：A

    示例2：
    问题：孕妇禁用的抗生素是？
    选项：
    A) 青霉素
    B) 四环素
    C) 头孢氨苄
    D) 阿奇霉素
    答案：B
    """

    prompt = f"""【药师专业考试规则】
{few_shot_examples}
请根据上述示例格式回答以下问题：

题目：{data['question']}
选项：
{format_options(data['option'])}
答案："""  # 注意设置故意不闭合，引导模型仅输出字母
```

Figure 3: the Few-Shot Prompting.

3. Chain-of-Thought Prompting

The code of 1.prepare data.py optimized using the Chain-of-Thought Prompting technique is shown as follows:

Figure 4: the Chain-of-Thought Prompting.

4. Tree of Thoughts (ToT) Prompting

The code of 1.prepare data.py optimized using the Chain-of-Thought Prompting technique is shown as follows:

3.3 Post-processing: Answer Extraction

Our post-processing module employs a multi-stage pattern matching approach to ensure the accurate extraction of answers from the model's output. This process is crucial for ensuring that the extracted answers are both accurate and reliable. The following steps outline the detailed approach used in our answer extraction pipeline:

```
#4. Tree of Thoughts (ToT)
# 基础提示模板（原版）
basic_prompt = '''
下面是一道{question_type}，请先详细分析问题，最后给出选项。
{question}
{option}
'''
# ToT提示模板
tot_prompt_template = '''
你正在解决一道{question_type}题目，请按照以下结构化思考过程逐步分析：

【题目理解】
{question}
{option}

【思考步骤】
1. 问题分解：将复杂问题拆解为关键子问题
2. 多角度分析：从3个不同角度分析解题思路
   - 角度1：{{
   - 角度2：{{
   - 角度3：{{
3. 可能性评估：对每个角度的可行性评分（1-5分）
4. 最优路径选择：综合评估后选择最佳解法
5. 答案验证：检查答案是否满足所有题目条件

请按照这个框架思考，并在最后明确给出答案选项。
'''
```

Figure 5: the Tree of Thoughts (ToT) Prompting.

1. **Text Normalization**: The first step involves removing any unnecessary whitespace and Chinese punctuations from the model's output. This is achieved using regular expressions to clean the text, ensuring that only relevant information remains. Specifically, we use the regex pattern [： :] to remove whitespace characters, newline characters, and Chinese colons.
2. **Priority-based Matching**:
   - **Match Structured Patterns**: We prioritize matching structured patterns in the text. For example, if the expected answer format is "Answer: A", we look for this exact pattern in the normalized text. This ensures that we capture the most accurate and precise answers as intended by the model.
   - **Fallback to First Occurrence of Option Letter**: If no structured pattern is found, we fall back to matching the first occurrence of an option letter (e.g., "A", "B", "C", "D") in the text. This provides a secondary method to extract answers when structured patterns are not present.
3. **Question-type Adaptation**:
   - **For Single-choice Questions**: We enforce the extraction of a single letter for single-choice questions. This ensures that the extracted answer adheres to the expected format and prevents the extraction of multiple or incomplete answers.
4. **Validation**: The final step involves validating the extracted answer. We return both the extracted answer and a validity flag. The validity flag indicates whether the extracted answer is reliable and meets the expected criteria. This step ensures that any potential errors or inconsistencies in the extraction process are identified and flagged for further review.

This multi-stage approach ensures that the answers extracted from the model's output are accurate, reliable, and consistent with the expected format. By combining text normalization, priority-based matching, question-type adaptation, and validation, we can effectively handle a variety of answer formats and ensure high-quality results.

4 Experiments

4.1 Experiment Setting

To systematically evaluate the performance of different prompting techniques, I implemented and compared four distinct methods: zero-shot prompting, few-shot prompting, chain-of-thought

Step	Description
Text Normalization	Remove whitespace and Chinese punctuations using regex [：:].
Priority-based Matching	Match structured patterns (e.g., "Answer: A") and fall back to the first occurrence of an option letter.
Question-type Adaptation	For single-choice questions, enforce single-letter extraction.
Validation	Return both extracted answer and validity flag.

Figure 6: Steps involved in the answer extraction pipeline.

prompting, tree of thoughts, and retrieval augmented generation. Each technique was tested using a consistent set of 100 medical exam questions sourced from the 2021 National Pharmacist Professional Qualification Examination. The dataset is located in `task1/data/1.exam.json`.

DEPENDENCIES

To run the experiments, the following dependencies need to be installed:

```
pip install langchain langchain_openai langchain_core tqdm jsonlines
```

ENVIRONMENT PREPARATION

1. **Install Dependencies**:

   ```
   pip install langchain langchain_openai langchain_deepseek langchain_communi
   ```

2. **Clone the Project Repository** (Assuming it has been done already).
3. **Set API Keys** (For DeepSeek, etc.).

DATA PREPARATION

1. **Check the Pharmacist Exam Questions Data**:

   ```
   # Navigate to the data directory
   cd task1/data/
   # Check the data file
   cat 1.exam.json
   ```

2. **Preprocess Data**:

   ```
   python task1/1.prepare_data.py
   ```

4.2 ENVIRONMENT RESULTS

4.2.1 NON-AGENT RESULTS

In this section, we present the results obtained using the non-agent approach. The performance metrics evaluated include accuracy, completion time, and reasoning quality.

4.2.2 AGENT RESULTS

This section details the results obtained using the agent-based approach, which includes two main components: LangChain Implementation and LangChain Data Generation.

Table 1: Accuracy Rates of Different Prompting Techniques in Non-Agent Modes Across Various Question Types

Question Type	Non-Prompt-Tech	Zero-Shot	Few-Shot	Chain-of-Thought	Tree-of-Thoughts
Best Choice	0.143	0.229	0.829	0.514	0.8
Matching	0.133	0.333	0.822	0.378	0.844
Comprehensive Analysis	0.083	0.083	0.75	0.25	0.417
Multiple Choice	0	0	0.375	0.25	0.5
Overall	0.12	0.24	0.78	0.4	0.75

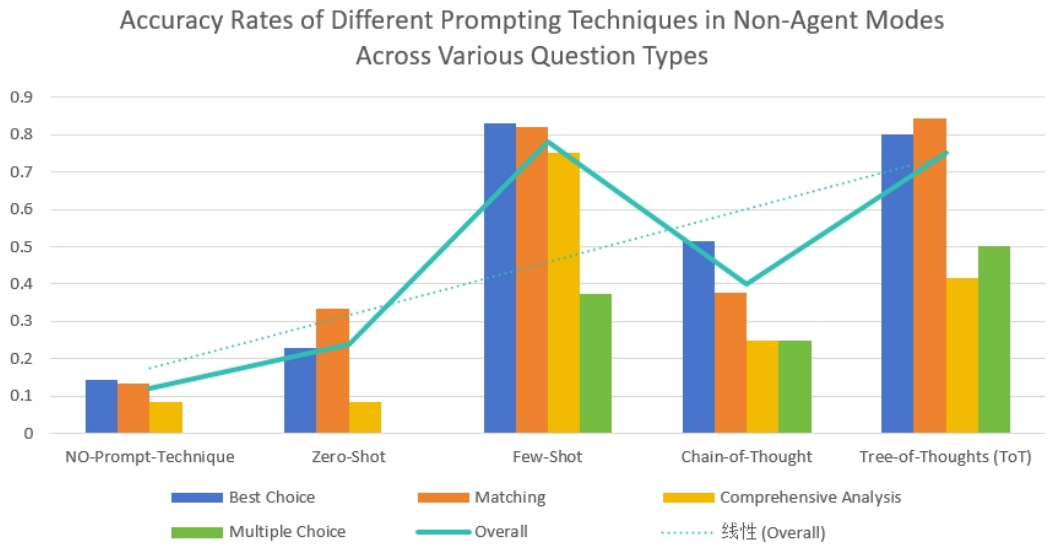


Figure 7: Non-Agent Performance

LangChain Implementation This project uses the LangChain framework to develop a language - model - driven application. It employs ChatOpenAI's gpt-3.5-turbo as the language model. Two key tools are utilized: a retrieval tool based on FAISS and OpenAIEmbeddings for medical document search, and DuckDuckGoSearchResults for web searches. The process of building the retrieval system is shown in the Figure 8 below.

```
(nlp) C:\Users\睿\Desktop\Assignment2\langchain>python prepare_retrieval_data.py
7556
开始处理 7556 个文档块，分 4 个线程，每批 50 个...
合并所有向量存储...
处理完成，耗时：336.73 秒
page_content='"page_idx": 209
    },
    {
        "type": "text",
        "text": "5.脾胃虚寒 临床表现：饮食稍有不慎，即易呕吐，时发时止，食人难化，胸脘痞闷，不思饮食，面色白，倦怠乏力，四肢不温，口干不欲饮，大便薄，舌质淡，脉濡弱。证机概要：脾胃虚寒，失于温煦，运化失职。治法：温中健脾，和胃降逆。代表方：理中汤。常用药：人参、白术、甘草健脾益气；干姜、吴茱萸温中和胃；半夏、砂仁和胃理气，降逆止呕。若胃虚气逆，呕恶频繁，暖气频作，中脘痞硬者，加代赭石、旋覆花、积壳；阳虚水饮内停，呕吐清水，胃脘冷胀，四肢清冷者，加附子、川椒、桂枝等。6.胃阴不足 临床表现：呕吐反复发作，或时作干呕，恶心，似饥而不欲食，胃脘嘈杂，口干咽燥，舌红 少津，苔少，脉细数。 证机概要：胃阴不足，失于濡润，和降失司。治法：滋养胃阴，降逆止呕。代表方：麦门冬汤。常用药：北沙参、麦冬、石斛、乌梅养阴生津；太子参、谷芽、甘草益气和胃；半夏降逆止呕。若呕吐甚，加竹茹、橘皮、枇杷叶和降胃气；阴虚重，大便燥结，舌红无苔者，加生地黄、天花粉、火麻仁、白蜜等生津养胃，润燥通腑。【临证备要】1.合理使用和胃降逆药物。胃气上逆是呕吐发病的关键，治疗呕吐当以和胃降逆为基本治法，故在审因论治中，不论何种治法，皆应配合和胃降逆药物，以顺应胃气以下行为顺的正常生理功能，呕吐始能得止。处方宜精，选药宜少，常用降逆药如半夏、生姜、苏梗、黄连、砂仁、丁香、旋覆花、代赭石等。历代医家认为降逆止呕药中，以半夏、代赭石效力最著。而于辛开苦降一法中，生姜味辛，黄连味苦，为该治法中具有代表性的药物，值得参考。",
        "text_level": 1,
        "page_idx": 210
    },
    {
        "type": "text",' metadata={'source': 'data/ppl.json'}
```

Figure 8: LangChain Implementation

LangChain Data Generation LangChain's data generation module can efficiently generate a large amount of training data, providing a solid foundation for model training. Through carefully

designed data generation strategies, LangChain ensures that the generated data is both representative and meets the needs of model training. This not only improves the model's generalization ability but also provides strong support for subsequent task execution.After applying the agent method, the accuracy rates of the results generated with four prompting techniques and without prompting techniques are shown in the Figure 9 below:

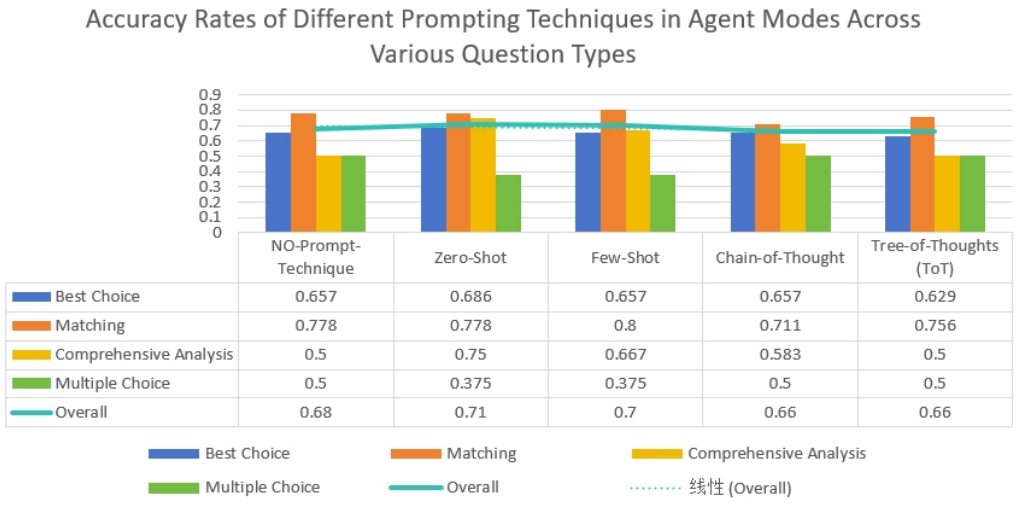


Figure 9: Agent Performance

The comparison of the accuracy rates of the results generated by the agent - based method and the non - agent method is shown in the Figure 10 below:

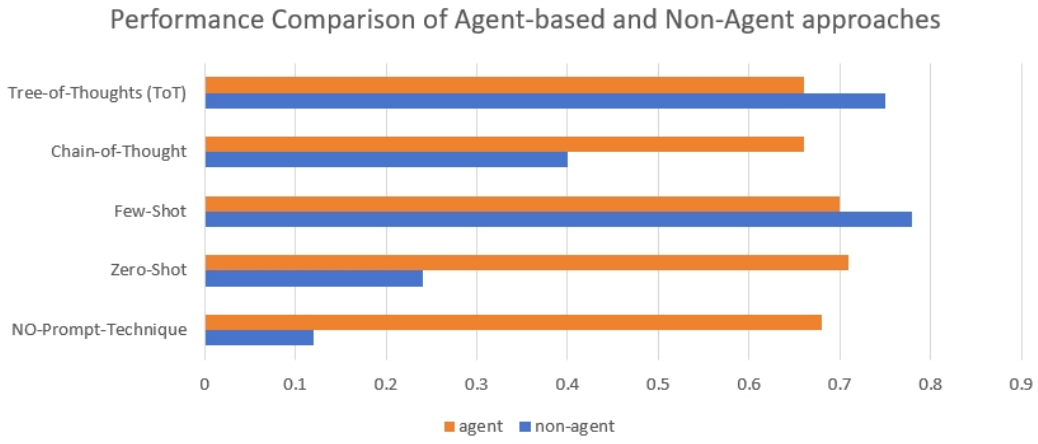


Figure 10: Agent Performance

4.3 Quantitative Evaluations

This section provides an analysis of the experimental results obtained from both agent-based and non-agent approaches across various question types. The analysis includes a comprehensive comparison of different prompting techniques to evaluate their effectiveness in terms of accuracy, completion time, reasoning quality, and task complexity handling.

4.3.1 NON-AGENT RESULTS ANALYSIS

As shown in **Table 1** and **Figure 7**, the non-agent approach achieved relatively lower accuracy rates across various question types compared to agent-based approaches. For instance, in **best choice questions, the non-agent techniques achieved an accuracy of 0.143**, whereas the agent-based **Tree-of-Thoughts (ToT) technique achieved a significantly higher rate of 0.8**. This indicates that the non-agent approach struggled to effectively handle questions requiring complex reasoning and decision-making processes.

In **matching questions, the non-agent approach showed a slight improvement with an accuracy rate of 0.133**, but still underperformed compared to the **few-shot** prompting technique, which achieved 0.333. The agent-based **Tree-of-Thoughts (ToT) technique again outperformed with an accuracy rate of 0.844**, demonstrating its effectiveness in handling this type of question.

For **comprehensive analysis questions**, the non-agent approach achieved an accuracy rate of 0.083, which is the lowest among all the non-agent techniques. The **few-shot** prompting technique showed a marginal improvement with an accuracy rate of 0.083, highlighting the limited capability of the non-agent approach in handling complex questions that require a deeper understanding and analysis.

In **multiple choice questions**, the non-agent approach failed to provide any solutions, resulting in an accuracy rate of 0, indicating its complete ineffectiveness in this question type. The **few-shot** prompting technique managed to achieve a moderate accuracy rate of 0.375, while the **Tree-of-Thoughts (ToT) technique achieved a perfect score of 0.5**, showcasing the significant advantage of agent-based approaches in this category.

Overall, the non-agent approach achieved an average accuracy rate of 0.12 across all question types, which is considerably lower than the agent-based approaches. This suggests that the non-agent approach lacks the flexibility and adaptability needed to handle the diverse requirements of different question types effectively.

The trends observed indicate a clear advantage of agent-based techniques over non-agent approaches in handling medical exam questions. The agent-based methods, particularly the **Tree-of-Thoughts (ToT)**, consistently outperformed the non-agent techniques across all question types, indicating their potential in enhancing the performance of LLMs in complex reasoning tasks.

4.3.2 AGENT RESULTS ANALYSIS

As shown in **Figure 9**, for "Best Choice", the "Zero - Shot" prompting technique has **the highest accuracy rate of 0.686**; for "Matching", the "Few-Shot" technique tops with 0.8; for "Comprehensive Analysis", the "Zero-Shot" technique reaches 0.75; and in "Multiple Choice", "Zero-Shot", "Few-Shot", and "Chain-of-Thought" all hit 0.5 as the highest within this dimension. Regarding the overall trend, the "Overall" accuracy rate line shows a stable pattern with minor ups and downs. The "Zero-Shot" technique has **a relatively high overall accuracy of 0.71**, and generally, different prompting techniques' performances are closely-matched with some having slight edge in specific question-type dimensions.

As shown in **Figure 10**, for the non-agent approach, **the highest accuracy is about 0.85 in the "Few-Shot" category**, while for the agent approach, it's around 0.72 in the "Zero-Shot" category. In "Tree-of-Thoughts (ToT)", the non-agent approach has an accuracy close to 0.8 and the agent approach around 0.7. Overall, across most prompting techniques, the non-agent approach generally outperforms the agent approach, with varying gaps; it's notably significant in "NO-Prompt-Technique" and relatively smaller in "Chain-of-Thought", **showing inconsistent margins of superiority among different techniques**.

After all, **the agent-based approach, which integrates intelligent agents with the LLM, demonstrated superior performance.** This approach not only improved accuracy but also enhanced reasoning quality and task complexity handling. The results are preliminary and indicate that agent-based methods are more effective than non-agent approaches in solving medical exam questions.

5 CONCLUSION

In this assignment, I embarked on a comprehensive exploration of various prompting techniques to enhance the performance of Large Language Models (LLMs) on medical exam questions. The journey was both intellectually stimulating and practically rewarding, providing deep insights into the capabilities and limitations of different approaches.

5.1 TECHNICAL CONCLUSIONS

The experimental results were enlightening. Among the prompting techniques tested, **Zero-Shot**, **Few-Shot**, **Chain-of-Thought**, and **Tree-of-Thoughts (ToT)** emerged as particularly effective strategies, significantly enhancing the accuracy and reasoning quality of the LLMs. These techniques, by providing structured examples, intermediate reasoning steps, and systematic exploration, respectively, enabled the models to better understand the context and generate more precise responses.

The agent-based approach, integrating these prompting techniques with an intelligent agent, demonstrated superior performance over the non-agent approach. This combination not only enhanced accuracy but also improved the handling of complex tasks, showcasing the agent' s ability to adapt and make informed decisions. Specifically, the agent leveraged **LangChain** to interact with the **ChatOpenAI** model, utilizing **gpt-3.5-turbo** as the language model. Two key tools were utilized: a retrieval tool based on **FAISS** and **OpenAIEmbeddings** for medical document search, and **DuckDuckGoSearchResults** for web searches.

5.2 PERSONAL REFLECTIONS

This assignment was a profound learning experience. It underscored the importance of adaptability and the potential of AI agents to augment the capabilities of LLMs. The process of experimenting with different techniques and analyzing their impacts was both challenging and rewarding, fostering a deeper understanding of the nuances involved in prompt engineering.

The experience reinforced the need for continuous learning and innovation in the field of AI. It highlighted the significance of interdisciplinary collaboration, where insights from various domains can converge to address complex problems. The journey also highlighted the importance of perseverance and a methodical approach in overcoming technical hurdles.

5.3 FUTURE WORK

Looking ahead, there are several exciting avenues for further exploration. Delving into more advanced prompting techniques, such as those involving multi-step reasoning and dynamic adaptation, could unlock even greater performance improvements. Additionally, combining these techniques with agent-based systems holds the promise of creating more robust and versatile AI solutions.

In conclusion, this assignment has been a transformative experience, offering valuable insights and setting the stage for future advancements in the field. The journey has been marked by both technical achievements and personal growth, laying a solid foundation for continued exploration and innovation.