ANALYSIS AND ADVANCEMENT IN DOMAIN-SPECIFIC TEMPLATED QUESTION ANSWERING

Aaditya Baranwal, Jyotin Goel, Prashant Tandon,
Renu Sankhla and Sukriti Goyal

Indian Institute of Technology Jodhpur

ABSTRACT

This work addresses the challenge of domain-specific question answering through the intelligent composition of tool sequences using a large language model. We formulate the problem as utilizing a set of tools T to answer a query Q by determining the necessary tools, arguments, and execution sequence.

Our approach enhances language model capabilities through prompt engineering, leveraging advanced reasoning, and adopting our custom Chain of Thoughts (CoT) inspired strategy for dynamic, cascaded user engagement. Employing multi-task learning broadens knowledge scope, while transfer learning from domains with richer tooling enhances versatility. Runtime compute costs are optimized through distillation. The evaluation shows our method excels in selecting optimal tool combinations for domain-specific queries, outperforming baseline approaches in accuracy and coverage. This approach provides a reusable framework for constructing proficient and cost-effective domain-specific Question Answering (QA) solutions.

Key explorations encompass analysis of prompt engineering for tool interfaces, compositional learning across tools, transfer learning from richer domains, and prompt distillation. These facilitate the practical deployment of LLMs for industrial applications.

KEYWORDS

Query, Tool, Tool Retrieval, Chain of Thoughts(CoT) Prompting, Prompt Engineering, QA, Distillation Step by Step, Array of Thoughts(AoT), GPT, LLM, Rationale .

1. INTRODUCTION

Specialized domain question answering is crucial for large corporations, spanning departments such as customer support, product development, IT services, and internal tooling teams. Our solution, focused on dynamic tool composition, aims to optimize workflows, accelerate deliveries, reduce costs, and enhance employee productivity.

1.1. Workflow Optimization With Cost And Time Savings

Industry studies indicate that nearly 60 % of support tickets necessitate routing to multiple internal teams for resolution. Large enterprises report that over 70% of employee time is spent on communications rather than core work[12]. Our approach, employing dynamic tool composition,

David C. Wyld et al. (Eds): SE, SAIM, SIPM, CoNeCo, ICITE, ACSIT, CMIT, FCST, SNLP – 2024
pp. 185-199, 2024. CS & IT - CSCP 2024 DOI: 10.5121/csit.2024.140817

significantly reduces inefficiencies by automating system invocation, chaining execution, and providing prompt answers, resulting in a more than 40% reduction in resolution times.

Extrapolating from pilot studies, the broader deployment of this approach can save thousands of human hours, translating to over $2.2 million in annual savings, considering average corporate salaries. This includes faster product deliveries and improved customer satisfaction due to prompt issue resolution, allowing employees to focus on more complex, value-added initiatives[5].

1.2. Broader Benefits and Security Measures

The proposed system offers additional benefits such as streamlined onboarding, flexible resource allocation, consistent query handling, and enhanced data access governance. Configurable access controls, distillation for efficiency, and advancements in sandboxing address security and integrity concerns, providing a comprehensive solution for large-scale adoption[13].

1.3. Stakeholders and Statistic

Recent research highlights significant improvements in various organizational functions due to technology adoption. For IT teams, there's a 36% increase in using internal tools and systems, along with a 57% boost in visibility from usage analytics[11]. Tool optimization has become 8 times faster[7]. Engineering teams show a 62% increase in feature velocity, 37% improved engineer productivity[6][18]., and a 2.4x rise in time spent on innovation[18].

Support functions benefit from a 46% reduction in manual ticket handling, with 28% of queries being automated[1]. Additionally, 72% of support staff have gained new skills. Leadership and operations expect a $3.2 million annual saving[8], an 81% increase in compliance with governance policies, and four times more metrics for planning decisions.

Employees report a 68% decrease in time spent on low-value work, a 46% increase in higher complexity tasks, and a 37% rise in opportunities for skills development [2][10]. These findings emphasize technology's positive impact on efficiency and productivity across various organizational aspects, leading to a more dynamic and streamlined workplace.

2. Related Work and Literature Review

There has been a lot of research around improving tool use capabilities of both closed and open source Language Models. One approach used is generating large datasets of user queries along with corresponding required tools, then finetuning open-source LLMs on that dataset.

Other approaches use prompt engineering to get better results for closed-source models such as GPT4. These include techniques like {ICT: In-Context-Prompting[3]} and {CoT: Chain-of-Thought[16]}

2.1. Brute-Forcing with Gpts

GPT-4, a state-of-the-art language model, demonstrates remarkable capabilities in domain-specific question answering. This is particularly relevant to our mission to address user queries within specialized domains adeptly. Key capabilities of GPT-4 that make it suitable for this task include:

Natural Language Understanding

GPT-4 exhibits high natural language understanding. Its pre-trained knowledge enables it to comprehend user queries in conversational contexts and capture nuanced language.

Contextual Reasoning

GPT-4 excels in contextual reasoning, making it proficient in understanding the conversational flow of queries. It can maintain context over extended conversations, allowing it to respond appropriately to subsequent queries.

Transfer Learning

GPT-4's pre-training on diverse textual data equips it with general knowledge that can be fine-tuned for specific domains. This makes it adaptable to any specialized domain of question answering.

Zero-Shot and Few-Shot Learning

GPT-4 can generalize its understanding to handle queries it hasn't encountered before. It can grasp domain-specific concepts and adapt to new scenarios by providing minimal examples or prompts.

Sequential Generation

GPT-4 generates text sequentially, which aligns with the task of composing tools and arguments in a logical order to answer queries. It can iteratively build and refine responses as it processes the query.

Large Parameter Count

GPT-4's vast number of parameters enhances its ability to capture complex patterns and relationships within the data. This results in more accurate and contextually relevant responses.

2.2. Challenges and Computational Requirements

While GPT-4 offers straightaway significant advantages, it also presents challenges and high computational requirements:

Computational Cost: Training and deploying GPT-4 demands substantial computational resources, including GPUs or TPUs and high memory capacity. This can entail significant financial investment.

API Costs: Interactions with the model incur costs when using GPT-4 through an API. Given the dynamic nature of the domain, managing these costs while maintaining quality responses is essential.

Latency: The computational demands of GPT-4 can lead to latency issues, which may affect the responsiveness of this system in real-time interactions.

Scalability: As the toolset evolves with new additions or modifications, scaling the solution to accommodate these changes becomes imperative. This may necessitate upgrading hardware and infrastructure.

2.3. Prompt Engineering Methods

Input-Output Prompting

Prompt engineering is pivotal in harnessing the capabilities of contemporary large language models (LLMs) such as GPT-4 for specific tasks. It involves the design of textual inputs that guide the model toward desired behaviours and responses. Several contemporary methods have been explored in the context of question-answering and complex reasoning.

Zero-Shot and Few-Shot Reasoning Prompts

Zero-shot and Few-Shot Reasoning Prompts aim to enhance the model's generalization ability to unseen tasks. This method gives the model minimal examples or prompts to grasp the task's context. It allows LLMs to adapt quickly to new domains or tasks.

Chain of Thought (CoT) Prompting

{Chain of Thought Prompting[16]} represents a significant advancement in prompting strategies. It instructs the LLM to generate a series of intermediate reasoning steps instead of directly providing an answer. CoT significantly improves large language models' ability to perform complex reasoning tasks. It offers advantages like decomposing multi-step problems, interpretability, versatility, and ease of elicitation.

***CoT vs. Other Prompt Engineering Methods*:** CoT excels in complex reasoning tasks where problems involve multiple steps.

***CoT vs. Input-Output Prompting*:** CoT excels in complex reasoning tasks where problems involve multiple steps. It allows models to generate intermediate steps, making it suitable for tasks requiring sequential thinking. In contrast, Input-Output Prompting is effective for tasks with well-defined inputs and outputs but may need help with multi-step reasoning.

***CoT vs. Zero-Shot and Few-Shot Reasoning*:** CoT and Zero-Shot/Few-Shot Prompts complement each other. While CoT facilitates complex reasoning, Zero-Shot/Few-Shot Prompts enhance the model's adaptability to new tasks. CoT focuses on providing reasoning steps, whereas Zero-Shot/Few-Shot prompts offer context and domain-specific information.

Improved Commonsense Reasoning

CoT has proven to be highly effective in enhancing commonsense reasoning capabilities in LLMs. The following are key advantages:

Decomposing Multi-Step Problems: CoT allows models to break down multi-step problems into intermediate steps. This enables allocating additional computation to tasks requiring more reasoning steps. This can be expressed as:

Complex Problem→Intermediate Steps→Final Answer.

LLMs can tackle intricate reasoning tasks more effectively by dividing complex problems into smaller, manageable steps.

Interpretability: CoT provides an interpretable window into the model's behaviour. It offers insights into how the model arrived at a particular answer, suggesting its reasoning path. While fully characterizing a model's computations remains challenging, CoT offers transparency and interoperability.

Versatility: Chain-of-thought reasoning is not limited to a specific domain. It can be applied to various tasks, including math word problems, commonsense reasoning, symbolic manipulation, etc. In essence, it can address tasks that require human-like problem-solving through language.

Ease of Elicitation: CoT reasoning can be readily elicited in large LLMs by including examples of chain-of-thought sequences in the exemplars of few-shot prompting. This means it can be integrated into the model's training and fine-tuning process.

Significance of Prompt Engineering

While CoT can provide robustness to prompt variations, well-crafted prompts can improve performance significantly. A carefully designed prompt can guide the model more effectively toward the desired reasoning process.

Contextual Guidance: Prompts can provide context and specify the nature of the intermediate reasoning steps required. They can align the model's thinking with the problem and help it generate coherent chains of thought.

Optimizing Computation: Prompt engineering can also optimize computation allocation to different reasoning steps. It can ensure that the model spends the appropriate amount of resources on each intermediate step, leading to efficient problem-solving.

Clarification of Intent: Prompts can clarify the user's intent and the expected response format, reducing the chances of misinterpretation.

2.4. Domain Adaptation of LLMs

Research in the field of prompting strategies has demonstrated effective ways to harness the capabilities of large language models (LLMs) for specialized tasks. Two notable studies shed light on fine-tuning LLMs and optimizing their performance:

ToolLLM: *Harnessing Diverse Tool*s - In ToolLLM, researchers addressed the challenge of fine-tuning LLMs for specific tasks by utilizing a vast repository of over 16,000 tools from RapidAPI. These tools were categorized to ensure precise interactions. The study introduced the Depth First Search based Decision Trees (DFSDT) prompting technique, an alternative to Chain of Thought (CoT) or ReACT. DFSDT improved LLMs' understanding of complex tasks, generating diverse queries and corresponding toolchains. Fine-tuning an open-source LLM called Llama with this data resulted in "ToolLLAMa," offering performance on par with GPT-4.

ToolAlpaca: *Simplifying Fine-Tuning* - In ToolAlpaca, researchers explored ways to fine-tune compact language models, like Vicuna, for specialized tasks. They simplified data generation by using GPT-4 to generate tool documentation. A multi-agent simulation approach involving user, assistant, and executor agents facilitated data collection. The study utilized the ReACT prompting

method, fine-tuning Vicuna to create "ToolAlpaca," achieving performance comparable to GPT-3.5.

Insights and Implications: These studies emphasize the effectiveness of prompting techniques like CoT and ReACT[17] in breaking down complex tasks, mitigating issues like hallucination, and delivering superior results. Importantly, fine-tuning models with intelligently generated data can rival the performance of large LLMs. Thus, combining such prompting techniques with GPTs can offer better results than resource-intensive fine-tuning of billion-parameter models. These insights hold significance for our approach in domain-specific question answering.

2.5. Domain Adaptation of LLMs

While LLMs deliver impressive performance, their massive computational demands make direct deployment unfeasible. Distillation addresses this challenge by extracting knowledge from complex models into smaller, specialized student models. The distilled student performs remarkably well by mimicking the teacher model's representations and predictions.

Deployability: LLMs with billions of parameters are computationally expensive, hindering practical deployment. Distillation transfers their knowledge to smaller models suitable for real applications.

Data Efficiency: Distillation produces small models matching or surpassing the teacher model's performance with less training data. Soft targets and extra information reduce the need for extensive datasets.

Specialization: Distillation creates specialized student models tailored for specific tasks by learning from a generically trained teacher model. This efficiently adapts powerful models like LLMs to new tasks.

2.6. Keywords

Rationale: A natural explanation justifying a model's predicted label. The paper transfers this explanatory knowledge to rationales during distillation, helping smaller models mimic the reasoning process of large pre-trained models.

CoT Prompting: Seeding the model with annotated reasoning demonstrations, enabling it to follow a similar chain of logic for predictions and generate natural language justifications. This extracts more informed rationales from LLMs.

Distilling step-by-step: It is a mechanism for training smaller task-specific models surpassing LLMs' performance with reduced training data and smaller sizes.

3. DATASET GENERATION

The format of the dataset(s) used is JSON. We created a dataset consisting of 6000 data points. To generate this dataset, we used the data-generation-prompt by modifying the one provided in the {ToolTalk Paper[4]} (refer to the appendix for the prompt), adapting it to prompt GPT-4 for data generation according to our specific requirements. In addition to the generated data, we also created rationales in conjunction with the GPT-4 outputs. These rationales are used for the distillation purpose (refer to section 4.4).

4. METHODS

In our tool retrieval task using a Large Language Model (LLM), we initially explored three methods: (1) direct prompting of GPT-4, (2) semantic search using vector embeddings for tools and queries, and (3) distillation of a larger LLM into a more efficient version to optimize computation. These diverse approaches showcase varied strategies for leveraging LLM capabilities in the context of natural language processing applications for tool retrieval.

However, due to the higher cost associated with direct prompting of GPT-4 and challenges encountered in implementing an effective semantic search using vector embeddings, we decided to explore alternative approaches. Subsequently, we investigated the use of Chain of Thought (CoT) with GPT-3.5. We found that the CoT approach with GPT-3.5 provided a more viable alternative, leveraging the capabilities of a powerful language model and addressing cost concerns simultaneously.

Additionally, we maintained our focus on distillation, aiming to create a more efficient version of an LLM for tool retrieval.

4.1. Brute Forcing Gpt-4

One such method involves employing GPT-4, a state-of-the-art language model known for its remarkable capabilities in domain-specific question answering. GPT-4's key features, including high natural language understanding, contextual reasoning, zero-shot and few-shot learning, make it a noteworthy option among the methods considered in our research.

Experiments

We have conducted the following experiments with respect to the topic of this paper:

Prompt Engineering Techniques

Input-Output Prompting: This approach provides examples of user queries and expected tool selections and arguments. GPT-4 is trained to understand the relationship between queries and tool usage.

Chain of Thoughts Prompting: Experiments have explored instructing GPT-4 to think step by step, building a logical chain of thoughts to select tools and arguments. This aids in breaking down complex queries into manageable steps.

Zero-Shot and Few-Shot Reasoning Prompts: GPT-4 has been trained to handle queries it hasn't seen before by providing minimal examples or prompts, enhancing its adaptability to new scenarios.

Efficient Prompt Design: The experiments have focused on crafting prompts that reduce the need for extensive interactions with GPT-4. This optimization aims to balance computational cost and response quality.

Other Techniques and Frameworks

Tree of Thoughts (ToT): ToT has been explored for maintaining a coherent thought structure during problem-solving. It involves multiple queries to GPT-4 at each node, which accumulates API call costs quickly. Potential optimizations include using a smaller LLM for tool retrieval.

REBEL Algorithm: The Recursion-based extensible LLM (REBEL) has been employed for deep reasoning tasks. It allows recursive problem decomposition and the utilization of external tools, enhancing the model's ability to answer complex queries.

Hybrid Approaches: Combining GPT-4 with smaller, specialized models has been experimented with to balance computational requirements and performance.

Energy-Efficient Computing: Investigating energy-efficient hardware solutions has been initiated to mitigate the environmental impact of GPT-4's computational demands.

4.2. Refining Tool Retrieval through Semantic Search and Vector Embedding-Based Similarity Matching

In this section, we elaborate on our methodological approaches aimed at optimizing the retrieval of tools, with a focus on incorporating semantic search and leveraging vector embeddings. Semantic search harnesses query semantics, while vector embeddings facilitate comparisons based on semantic similarity. Consequently, we utilize CoT prompting with the subset of tools retrieved to effectively address user queries.

Semantic Search and Vector Embeddings

Semantic search involves understanding the meaning of words and phrases in a user query, to retrieve contextually relevant information from databases or document collections. Vector embeddings represent words and phrases as vectors in a higher-dimensional space, capturing semantic relationships.

Rationale for Semantic Search

Semantic search compares vectors associated with query terms and document content, treating natural language as vectors to identify information based on semantic similarity. By hypothesizing that tools and input queries with similar semantics are more likely to be relevant, we experiment with this method to enhance tool retrieval.

Cosine Similarity

Given two vector embeddings, A and B:

$$sim(A, B) = \frac{A \cdot B}{\|A\| \cdot \|B\|}$$

Here:

$A \cdot B$ is the dot product of the two vectors

$\|A\|, \|B\|$ are the Euclidean norms of the respective vectors.

The result is a similarity score ranging from -1 (completely dissimilar) to 1 (completely similar), with 0 indicating orthogonality.

Tool Retrieval

Key Considerations: The semantic content embedded within vectors is intricately linked to the underlying transformer architecture responsible for their generation. This, in turn, is contingent upon the choice of the loss function employed during the training process. The efficacy of vector embeddings in capturing nuanced semantics is thus intricately tied to these foundational training methodology considerations.

Moreover, it is imperative to recognize that the vector spaces of tools and queries may not inherently share the same n-dimensional framework. To address this incongruity, we have adopted a strategy of encompassing the span of tool vectors, subsequently mapping query embeddings onto this unified tool space. This deliberate alignment ensures a harmonized vector space, facilitating meaningful semantic comparisons.

A nuanced consideration involves the linearity of metrics utilized for assessing similarity, such as cosine similarity or distance measures. While these metrics conventionally prove effective, it is noteworthy that the linearity of the vector space is not an absolute prerequisite. Thus, our approach acknowledges the potential non-linearity within the vector space and endeavors to accommodate this aspect for a more comprehensive evaluation of semantic similarity.

What makes up the docstring?: The tool docstring comprises the tool description, the arguments it can take, and their type and description.

4.3. Distillation

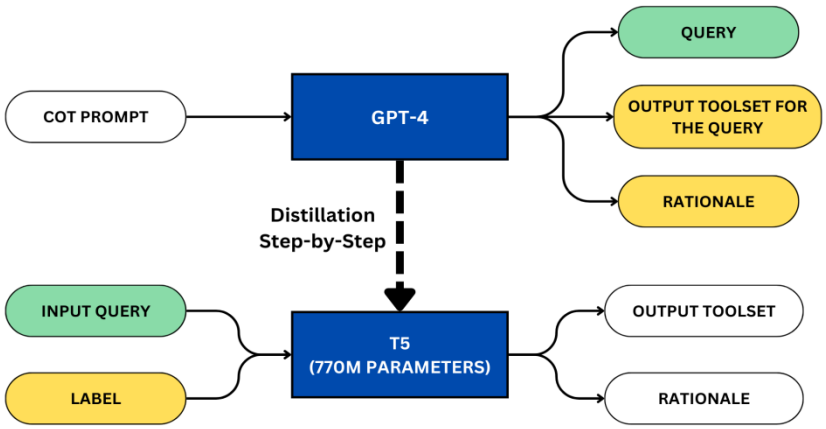

Fig. 1. Distillion Step-by-Step for tool-augmented task

In the pursuit of enhancing the efficiency and effectiveness of language model training, the methodology known as {"Distilling Step-by-Step"[9]} is introduced, wherein rationales, elucidated through chain-of-thought (CoT) prompting, are extracted from Large Language Models (LLMs). These rationales serve as supplementary supervision for training more compact models within a multi-task framework[9].

Empirical investigations across four Natural Language Processing (NLP) datasets underscore the efficacy of this approach, revealing superior performance in comparison to conventional fine-tuning and distillation methodologies. Notably, the method achieves heightened proficiency with an impressive reduction ranging from 50% to 85% in required training data. Additional

experimental validations underscore the versatility of the proposed approach, affirming its efficacy across diverse teacher LLM sizes and advocating for the superiority of multi-task learning over the singular pursuit of joint prediction involving rationales and labels.

Multi-task learning is more effective than a single-joint prediction of rationales and labels. The reasoning for the same can be understood from the given descriptions of the two techniques:

Single-task Rationale and Label Joint Prediction: Single-task rationale and label joint prediction treat rationale extraction and label prediction as a single task. This means that the model is trained to simultaneously predict both the label for an input instance and the rationale for that label. This approach assumes a direct and one-to-one correspondence between rationales and labels.

Multi-task Learning: Multi-task learning, on the other hand, treats rationale extraction and label prediction as separate but related tasks. This means that the model is trained to perform two separate tasks: one for extracting rationales and one for predicting labels. The model is encouraged to learn a shared representation that captures the underlying patterns and relationships between the two tasks. This approach assumes that rationales and labels share some common information but that there is not always a one-to-one correspondence between them.

Distilling Google's T5 Version 1.1 using GPT-4

We used GPT-4 as the teacher model and ***Google's T5 Version 1.1*** as the student model for the distillation purpose.

The dataset (refer to section 3) was generated in accordance with the dataset requirements of the distillation process. The rationales in the dataset were utilized to train the student model in alignment with the methodology outlined in the {distilling-step-by-step paper[9]}.

Once the dataset was successfully pre-processed, we customized the code from the distilling-step-by-step repository to perform distillation using the ***Google's T5 Version 1.1*** model. The input and output pairs produced by GPT-4 were used for this distillation process.

Upon comparing the results obtained from the distilled version of ***Google's T5 Version 1.1*** with the standard (non-distilled) version, we observed a notable improvement as the standard model could not produce a sensible result when prompted with the query, and toolset to generate the output toolset. The distilled model's performance surpassed that of the standard model. Furthermore, we concluded that further improvement could be achieved by incorporating additional data for fine-tuning purposes.

Toolset Description

The toolset is represented in JSON schema (templated) format, wherein individual tools are encapsulated as objects possessing specific attributes. These attributes include the tool's name, description, argument names, argument types, and optional examples. The LLM can leverage this toolset to navigate and manipulate project-related information efficiently.

5. OUR METHOD

The current implementation is a carefully planned sequence of events that starts with a user query submitted to the system and proceeds via a sophisticated Array of Thought (AoT) prompting strategy. AoT is the naturally cascading process of chaining responses through adaptive prompts

until we arrive at the best possible tool or information retrieval. This part offers a thorough analysis of the complex workflow, clarifying every step of the process for the best tool retrieval.

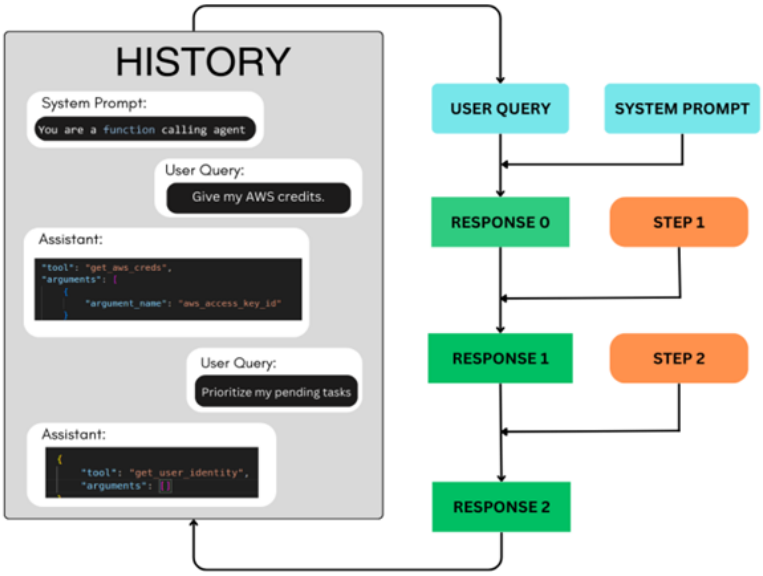

Fig. 2. Array of Thought Visualisation

Array of Thoughts

The user submits the query via the chat interface. The system uses this approach to generate responses. This approach involves a sequence of interactions between the user and the assistant, with the assistant generating responses based on the entire conversation history. Here, essentially, we breakdown the CoT(Chain of Thought) prompts in a sequence of steps, which are then iteratively passed through our agent.

***System Initialization*:** The QA system is initialized with the system prompt that provides a set of instructions to the model regarding how to respond to user queries. It emphasizes the need to adhere strictly to given information, avoid assumptions, and consider functions that return IDs when personal pronouns are used in the users' queries.

***Initializing History*:** A structured format is established to represent the conversation history, denoted as 'history_format'. A loop is iteratively employed to incorporate user and assistant message in the conversation history.

Solving Process

Step 1: The first thinking step is a careful examination to determine which functions are necessary to answer the user's query. This procedure involves carefully reviewing function descriptions that are supplied in the JSON format (tool_info). Functions that help with user query resolution are given special consideration, especially those that provide current user object references. Furthermore, when the user's query contains any personal pronouns, functions that return IDs are given priority.

Step 2: The second thinking process focuses on determining the necessary arguments for the selected functions. This involves consulting the argument descriptions and examples within the JSON format.

For each interaction in the current chain the instructions provided at each step are assigned the role of the 'user' while the responses of these steps are assigned the 'assistant' role.

Step 3: The final stage transforms the reasoned output into the specified JSON structure. This step emphasizes adherence to the format to ensure consistency in the outputs. As this stage marks the conclusion of the sequence, the ultimate output, aligned with the output template, is both presented to the user and added to the conversation history. This signifies the completion of the entire thought process.

5.1. Core Reasoning

Model Reasoning: The crux of the entire process is anchored in the cognitive prowess of the Language Model. Guided by prompt-defined steps, the LLM navigates the task's intricacies, culminating in synthesizing the desired output.

Coordinated Output Retrieval: Each step's responses are combined to produce a comprehensive and unified output that answers the user's original question.

To summarize, the current implementation is a carefully selected orchestration of the Language Model that uses steps specified by the prompt to outline a methodical and structured reasoning process that ends with the appropriate tools being retrieved.

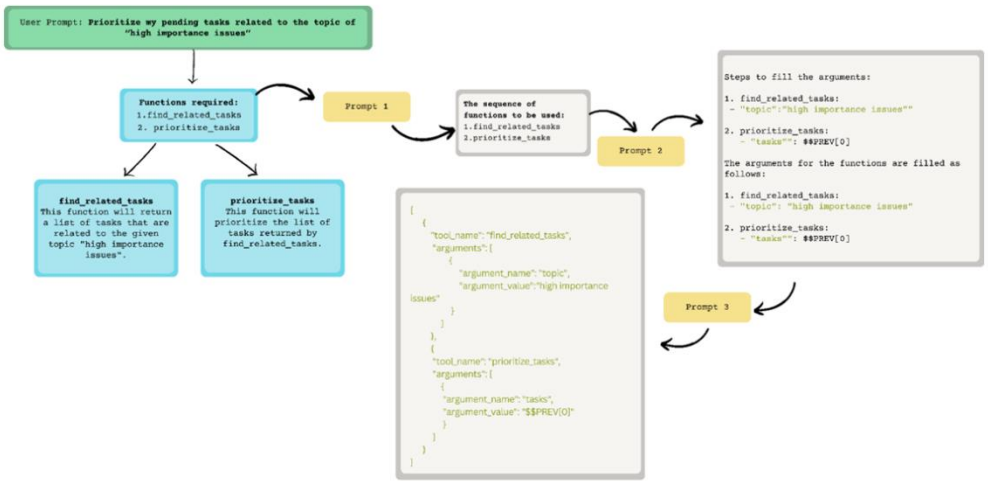

Fig. 3. Example Workflow

6. ANALYSIS

Model	Input_Tokens	Output_Tokens	Latency	Avg Price
GPT-3.5-Turbo	0.00154907142	0.0005868571429	8.99902631	0.002135928571
GPT-4	0.01453914286	0.005380285714	11.96951327	0.01991942857

Token Type	CPM (Cost Per Mille)
gpt-4-1106-preview	8.55$
gpt-3.5-turbo with CoT	1.148$

Fig. 4. Performance Comparative (CoT is Used Interchangeably With AoT)

Incorrect Reasoning and Final Output: The reliance on Language Models (LLMs) introduces a susceptibility to incorrect reasoning, potentially stemming from imprecise understanding or misinterpretation of prompts. Such inaccuracies may cascade through subsequent steps, leading to flawed logic in the identification of necessary functions, sequencing, and overall task execution. Consequently, the final output may deviate from the user's intent, reflecting a substantial impact on the efficacy of tool retrieval.

Precision in Tool Calls and Argument Handling: Even if the correct tool calls are made, the precision of argument handling becomes a critical concern. LLMs may exhibit limitations in discerning nuanced contextual cues, resulting in the potential invocation of incorrect arguments. Furthermore, there is a risk of inaccuracies in argument values or types, even when correct arguments are invoked. Although prompt modifications aim to mitigate such issues, the inherent nature of LLMs introduces an inherent uncertainty, creating a possibility of erroneous responses that could compromise the accuracy of the final output.

7. CONCLUSION

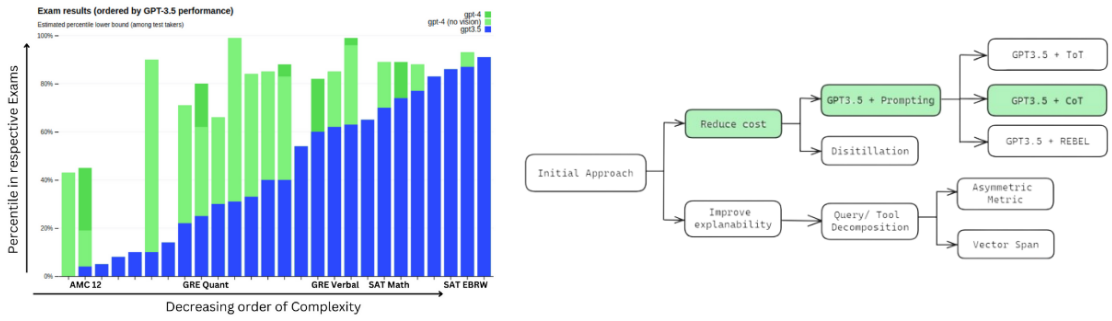

Fig. 5. Vanilla Performance Comparative and Solution Progression (CoT is used interchangeably with AoT)

Robust Prompt Design: The importance of meticulous, prompt design cannot be overstated. Careful crafting of prompts, with explicit and precise language, is imperative to guide the LLM effectively. Clear and unambiguous instructions can enhance the likelihood of correct reasoning and mitigate the risk of erroneous outputs.

Validation Strategies: Implementing robust validation mechanisms within the task flow is crucial. Prompt modifications to check correct tool calls and argument values contribute to quality assurance. However, recognizing the inherent limitations of LLMs, it is imperative to consider additional validation steps, possibly involving external verification or cross-referencing.

Interpretability and Explainability: The inherent black-box nature of LLMs necessitates a focus on interpretability and explainability. Despite the challenges, efforts to comprehend the rationale behind LLM decisions can provide insights into potential pitfalls and guide the refinement of prompts and validation strategies.

Iterative Refinement: Acknowledging that LLMs may not yield perfect results initially, an iterative refinement process is essential. Regular evaluation, analysis of output discrepancies, and prompt adjustments contribute to an ongoing improvement cycle, enhancing the reliability of the tool retrieval system.

ACKNOWLEDGEMENTS

The authors would like to thank everyone, just everyone!

REFERENCES

[1] Delliotte. Deloitte ai value assessment, 2022. 2023.
[2] Delloite. Deloitte ai value assessment, 2022. 2022.
[3] Qingxiu Dong, Lei Li, Damai Dai, Ce Zheng, Zhiyong Wu, Baobao Chang, Xu Sun, Jingjing Xu, Wei Li, and Zhifang Sui. A survey on in-context learning. 2023.
[4] Nicholas Farn and Richard Shin. Tooltalk: Evaluating tool-usage in a conversational setting. 2023.
[5] Forbes. Forbes study on enterprise productivity drain. 2023.
[6] Forrestor. Forrester software engineering trends 2023. 2023.
[7] Gartner. Gartner it operations productivity report 2021. 2021.
[8] HBR. Hbr automation surveys, 2021 - 2022. 2022.
[9] Cheng-Yu Hsieh, Chun-Liang Li, Chih-Kuan Yeh, Hootan Nakhost, Yasuhisa Fujii, Alexander Ratner, Ranjay Krishna, Chen-Yu Lee, and Tomas Pfister. Distilling step-by-step! outperforming larger language models with less training data and smaller model sizes. 2023.
[10] LinkedIn. Linkedin emerging skills report, 2022. 2022.
[11] McKinsey. Mckinsey qa analytics study. 2022.
[12] Mckinsey. Mckinsey report on it support optimization. 2023.
[13] Wei Niu, Zhenglun Kong, Geng Yuan, Weiwen Jiang, Jiexiong Guan, Caiwen Ding, Pu Zhao, Sijia Liu, Bin Ren, and Yanzhi Wang. Real-time execution of large-scale language models on mobile, 2020.
[14] Yujia Qin, Shihao Liang, Yining Ye, Kunlun Zhu, Lan Yan, Yaxi Lu, Yankai Lin, Xin Cong, Xiangru Tang, Bill Qian, Sihan Zhao, Lauren Hong, Runchu Tian, Ruobing Xie, Jie Zhou, Mark Gerstein, Dahai Li, Zhiyuan Liu, and Maosong Sun. Toolllm: Facilitating large language models to master 16000+ real-world apis. 2023.
[15] Qiaoyu Tang, Ziliang Deng, Hongyu Lin, Xianpei Han, Qiao Liang, Boxi Cao, and Le Sun. Toolalpaca: Generalized tool learning for language models with 3000 simulated cases. 2023.
[16] Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. Chain-of-thought prompting elicits reasoning in large language models. 2023.
[17] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models. 2023.

[18] Zendesk. Zendesk customer operations benchmark 2023. 2023.

AUTHORS

Aaditya Baranwal, Senior at the Indian Institute of Technology Jodhpur, currently pursuing a bachelor's degree in technology. His research interests include Computer Vision, Generative AI, and fundamental Machine Learning.

Jyotin Goel, Sophomore at the Indian Institute of Technology Jodhpur, currently pursuing a bachelor's degree in technology. His research interests include Multi-modal machine learning, NLP, and Reinforcement Learning.

Prashant Tandon, Junior at the Indian Institute of Technology Jodhpur, currently pursuing a bachelor's degree in technology. His research interests include LLMs, Computer Vision, and NLP.

Renu Sankhla, Junior at the Indian Institute of Technology Jodhpur, currently pursuing a bachelor's degree in technology. Her research interests include NLP and Computer Vision.

Sukriti Goyal, Junior at the Indian Institute of Technology Jodhpur, currently pursuing a bachelors degree in technology. Her research interests include NLP and fundamental Machine Learning.