LLM-RSPF: Large Language Model-based Robotic System Planning Framework for Domain Specific Use-cases

Chandan Kumar Singh, Devesh Kumar, Vipul Sanap, Rajesh Sinha
Tata Consultancy Services (TCS) Research, New Delhi, India
{ck.singh1, k.devesh1, vipul.sanap1, rajesh.sinha}@tcs.com

Abstract

The employment of large language models (LLMs) for task planning and reasoning has emerged as a focal point of interest within the robotics research community. However, directly applying LLMs, even with large token-sized prompts, does not achieve the task planning performance required for an industrial-grade domain-specific use-case (DSU). This work aims to overcome the obstacles of a robotic task planner for DSUs by introducing a novel planning framework, LLM-RSPF (Large Language Model-based Robotic System Planning Framework). Central to the LLM-RSPF is a novel robotic system ontology that organizes the components of the robotic system in a coherent and a systematic manner. The ontology empowers the LLM-RSPF to efficiently capture a contextual representation of the DSU using the LLMs. Subsequently, the research introduces a LLM-tuning regimen referred as chain of hierarchical thought (CoHT), specifically crafted to complement the proposed system ontology. Integrating these two components, the LLM-RSPF aims to enhance the accuracy, robustness, and throughput of a robotic system in a cost-effective manner. In addition, the research presents an empirical methodology to generate the LLM-tuning dataset size for a guaranteed performance. The LLM-RSPF is validated on a retail order-fulfillment use-case thereby, illustrating the efficacy of the framework. Through rigorous evaluation, the LLM-RSPF demonstrates exceptional performance on the generated dataset, effectively meeting the DSU objectives.

1. Introduction

In recent times, advancements in generative artificial intelligence have opened up unprecedented opportunities across various fields, including robotics. With the emergence of large language models (LLMs), the potential for revolutionizing human-robot interaction and robotic task execution has become a reality. The tasks that were once considered daunting such as advanced natural language understanding and having robust conversational agents are now within reach [19]. These LLMs, trained on extensive internet data, have introduced a renewed sense of possibility in the field by providing access to vast repositories of both objective and contextual knowledge. Two key areas where these capabilities help solve a long existing problem [19] for robotics are (a) creation of a natural language interface between human and robots (b) planning and reasoning ability of robots to perform tasks. While LLMs can enhance robotic planning and reasoning, their application to specific DSUs remains challenging [17] due to (a) inconsistent response from the LLMs (b) absence of domain knowledge in the LLM training dataset (c) Absence of physical understanding of the robot-world interaction. Conversely, the requirement of performance metrics for an industrial-grade DSU is often quite high. Given the shortcomings of existing LLMs in discussed areas, it is imperative to have a planning framework in place and this is the prime focus of our work. The objective is to propose a planning framework that can solve a highly domain specialized robotic use-case using off-the-shelf LLMs. The key components of the framework and also the contributions of the work are:

1. A novel robotic system ontology that is structurally potent enough to describe robotic system and the DSU to an off-the-shelf LLM.

2. A novel LLM prompt-tuning regimen, referred here as chain of hierarchical thought (CoHT), to contextualize an off-the-shelf LLM.

2. Literature Survey

To understand recent research in robotic planning, surrounding the proposed work, the discussion is partitioned into two areas. First, we'll examine planning frameworks that leverage LLMs to achieve task planning objectives. Second, we'll explore generic methods that research community uses to contextualize LLMs for a DSU.

2.1. LLM-based Robotic Planning Frameworks

For more than a decade, classical planning frameworks have been the only choice for any robotic system planning [6] [4]. Knowrob [16] and OWL [2] are two such widely recognized knowledge representations used in classical planning. PlanSys2 [10] is another noted planning framework that uses PDDL [12] as the domain-specific language (DSL) and classical planners to generate optimal task plans for a robotic system. SkiRos2 [11] and Merlin [5] are other similar planning frameworks that use PDDL and web-based ontologies as DSL for planning. However, in recent times, there has been a paradigm shift from classical planners to foundational model-based planners [21]. For instance, LLM+P [9] combines both the classical planners and the LLM-based planning capabilities. SayCan [1] and ProgPrompt [14] solely uses LLM-based task planner and focus towards effective grounding of LLM task plans for robotic manipulation tasks. However, both of these works focus towards generalized tasks and are not bound towards achieving accuracy in high 9's with high throughput and lower latency. In fact, this observation follows a good portion of rest of the cited literature. Another recent research closest to our work is Microsoft ChatGPT Robot Manipulation (MCRM) [18]. In this work, the authors have presented a structured and a scalable prompting approach that can be used for DSUs. It is indeed a good attempt, however, falls short in a few areas. For instance, failure scenarios are not considered while updating environments, requires multiple user-feedback calls for accurate task plan, and an exemplar sequence-based prompting is absent, which is observed to be an important facet in contextualizing LLMs on DSUs.

2.2. Grounding LLMs for DSUs

There are several approaches used to contextualize the LLMs against a human instruction [24]. Some of these include pre-training, fine-tuning, and retrieval augmentation methods [7], however, these approaches seek either high compute or a huge dataset, which raises questions on their usage and terms of applicability. Conversely, there are different standard prompting techniques that are readily used to adopt any LLM on the fly and contextualize it for a custom DSU. It either requires a small amount of dataset or does not require at all for LLM-tuning [21]. At the same, it is observed and noted that it is quite challenging to readily contextualize any off-the-shelf LLM on a DSU [8] [19]. The LLMs repeatedly fail to reason and entirely understand the domain-rules and the nuances of a DSU [17]. Therefore, a LLM-centric structurally potent knowledge representation of the robotic system and the DSU becomes a necessity. This representation must serve towards achieving a better contextualization. In a similar stretch, there have been developments to significantly improve upon the existing prompting abilities comprising chain of thought (CoT) [20], tree of thought (ToT) [22], algorithm of thought [13], contextual augmentation, etc. Currently, CoT and ToT are prominently used to improve reasoning of the LLMs. However, it is noted that CoT is effective with abstract-level contextual knowledge and at times misses out on low-level domain rules in any DSUs. CoT is significantly dependent on the LLM size and typically under-performs with smaller LLMs [22]. Also, there is a limited scope for verifying the generated intermediate reasoning/thoughts, therefore, there is a high likelihood of reaching to an incorrect solution. On the other hand, ToT effectively addresses most of these limitations. Considering this, the implementation of the TOT in a practical DSU seems lucrative, however, is limited by its complexity. It comes at the cost of frequent output token exhaustion and computational complexity, which impacts the objective of achieving low latency and high throughput in DSUs. As a result, ToT might not be necessary for tasks that can be excelled by an intermediary prompting approach [22]. This is exactly where a new prompting technique is required that is suitable for DSUs.

3. Proposed Planning Framework

This section elaborates the proposed planning framework, i.e., LLM-RSPF. It discusses the robotic system ontology and the LLM-tuning regimen that will enable system developers and system integrators to effectively contextualize LLMs for their robotic applications. The regimen supplies a set of rules and strategies that can be employed to have an accurate, robust, and a scalable task planner.

3.1. Proposed Robotic System Ontology

The proposed robotic system ontology is briefly illustrated in Fig.1. The ontology is modular and distributive. It has following key modules:

3.1.1 Use-case

Use-case is the parent module, where problem statement and detailed DSU description is provided. It sets the abstract representation of the DSU for any LLM.

3.1.2 Embodiment

Embodiment defines different Agents that are collaboratively participating to accomplish a specific task. An Agent is uniquely identified through four components, namely (a) Physical Robot (b) Behavior (c) End of arm tool (EOAT) (d) Sensor. The Embodiment module is further elaborated based on its Capability, Interfacing, State, and Experience, as illustrated in Fig.2. The Capability of the Embodiment are classified in the form of Sensing and Action, which is derived from the ReAct approach [23]. The Sensing empowers the Embodiment with the General perception, and

Vision and Tactile perception abilities. General perception is typically a vision foundational model-based open-vocabulary object detection, which provides a real-time Workspace information. Next, Action serves the "Act" capability depending upon the behavior of an Agent. The Act capability here connotes Robotic skills. Each Robotic skill is inherently a combination of several atomic skills such as see, move, pick, place, etc, and have its own matured perception, manipulation, and mobility abilities. Note, the reason behind assigning a skill as a compound skill is is to ease the complexity of planning, and for scalability of the system through integration of new matured skills. A skill for an Agent has its own definition of attributes from the physical Workspace. K-*th* skill of an Agent a is defined in (1). State denotes the current Agent state. It can be robot joint angles, grippers attached, etc. Lastly, Experience captures the Agent-level success and failure experiences.

$$ST_{k,a} = \bigcup_{i=1}^{n} attr_{i,a} \tag{1}$$

3.1.3 Workspace

Workspace defines the testbed of the DSU. It consists of five configurable components. Mobility space defines the detailed mobile space consisting all location identifiers and movement indicators (if any). Objects are the target items that are supposed to be handled by the Embodiment, while executing any task. Pick and drop locations refers to the locations specifically allotted in the Workspace for any target object to be picked up or dropped by the Embodiment. Lastly, Arrangement determines the physical arrangement of different Agents in the Workspace with respect to different location identifiers.

3.1.4 Objective

Objective module defines the task objective against a DSU. It sets all the Domain rules relevant to the DSU that are supposed to be met during any task planning. In short, the generated plan must adhere to these rules at any cost. Performance index are the list of metrics used to assess the performance of an embodied-system against the generated plan. Operation component actually determines the overall operational objectives to be met, which may or may not be a function of performance metrics.

3.1.5 Relation

Relation identifies any relational mappings among components definitions either intra-module or inter-module. For instance, an Agent can be tied to pick from a specific location in the Workspace. Mapping caters to these relational mappings. Next, Classification offers an option to classify

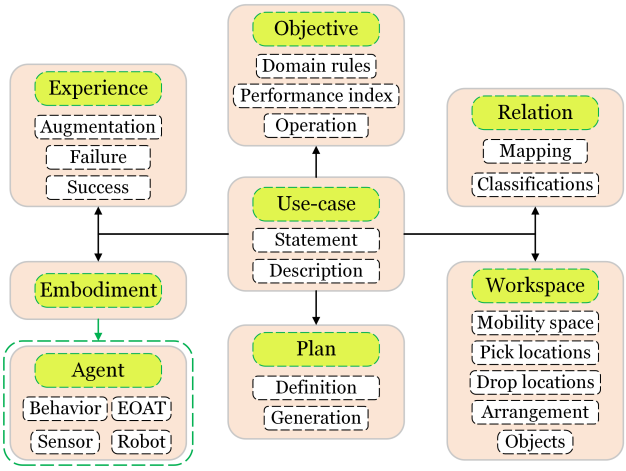

Figure 1. Proposed robotic system Ontology for the LLM-RSPF

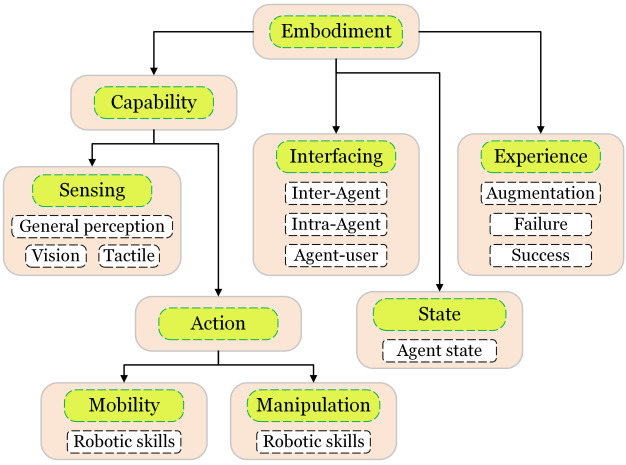

Figure 2. Embodiment module of the proposed ontology

user instructions based on its nature such that it is appropriately mapped against different DSU scenarios. This component is quite critical in instruction classification and conversational response generated by the LLM. It eases out the decision-making job of the user interface (UI).

3.1.6 Experience

Experience maintains various system or task relevant intermediary states and stores overall success and failure scenarios. It helps the LLM recall past experiences and incorporate them in future task planning. The failure experiences are significantly important here as these help in reducing the latency of the system substantially and limits any repeated-failure (Re-Failure) scenarios. Further, chat history can be used both in a truncated or a RAG fashion to store experiences.

3.1.7 Plan

Plan consists of the task plan definition and any template-specific conditions that must be adhered while generating a task plan. (2) describes the dictionary format adopted against a Robotic skill definition and its sequence in the plan for an agent, whereas the final task plan template is denoted by P in (3). The Generation component defines different type of inputs needed to generate a final task plan. These inputs can be from State, Sensing, Objective, etc.

$$P_{k,ST_k} : \begin{Bmatrix} k_{1,1} & ... & k_{S,1} \\ & ... & \\ k_{1,A} & ... & k_{S,A} \end{Bmatrix} \rightarrow \begin{Bmatrix} ST_{1,1} & ... & ST_{S,1} \\ & ... & \\ ST_{1,A} & ... & ST_{S,A} \end{Bmatrix} \quad (2)$$

$$P = \bigcup_{a=1}^{A} \bigcup_{k=1}^{S} P_{k,ST_k,a} \quad (3)$$

3.2. LLM-powered Planing Regimen

3.2.1 Dataset

To get started with the planning regimen, it is important to ensure that the task plan representation is appropriately defined in the Plan module. A DSU dataset is a must have requirement in order to achieve a good contextualization of the LLMs. The complete dataset for the contextualization is divided into three portions training, validation, and testing, in the ratio of 1:1:2, respectively. Note, here training terminology is used with reference to the LLM-prompting and does not refer to training LLM from scratch or fine-tuning them. Concerning the dataset creation, three types of estimations are performed as follows.

(a) **Unit size**: The dataset creation and its unit size is estimated based on the task plan representation and the classifications defined in the Plan and the Relation modules, respectively. Here, unit size terminology refers to the minimum standard dataset size that can be used to estimate the size of training, validation, and testing dataset. Let us assume that all kind of instruction classifications sums up to m and $\hat{\mathbf{m}}$ denote the unit vector corresponding to each classification label. The total Robotic skills derived from (2) amounts to S. Considering the M-th classification as the only classifications that outputs a valid task plan for total agents A, the total combinations of the instructions corresponding to these two inputs become $\binom{A}{S}$. The ϵ factor is chosen such that each classification spans sufficient number of user instructions. (b) **Diversity**: Recognizing the need to ensure sufficient diversity and ambiguity learning catering to different use-case scenarios during the contextualization, $\epsilon'\delta$ is introduced, which refers to the total number of such instructions needed. Note, ϵ' must be kept ideally greater than or equal to ϵ because it ensures that the dataset pertaining to diverse scenarios are also given equal if not more importance as compared to only Action-specific similar scenarios. (c) **Uncertainty**: Uncertainty resembles the number of repeated trials chosen against each classification to ensure repeatability of the generated plan. It is denoted by $g(\omega)$ and is empirically calculated using (4). The empirical arrangement in (4) follows a typical range-mapping logic against a classification priority set as ω. The ω offers an option to assign weights to different classifications based on their importance and chance of occurrence. A high weighted classification results in a highly confident and a robust instruction set belonging to that classification.

Combining both size and diversity estimations, the final dataset unit size comes out to be $|D|$, whereas the minimum number of instruction trials/experiences required are given by $|T|$ in (5). Note, choosing an estimation greater than the outcome of (5) is a calculative guarantee made to ensure the desired contextualization of the LLM.

$$g(\omega) = \lceil e^{(3-\frac{1}{\ln(\omega+1)})} \rceil, \omega \in \{1, 2, ..., \mathbb{N}\} \quad (4)$$

$$|T| \geq (\overbrace{\epsilon \sum_{m=1}^{M-1} |m|\hat{\mathbf{m}} + \epsilon \binom{A}{S} \hat{\mathbf{M}} + \epsilon'\delta\hat{\mathbf{M}}}^{\text{Domain rules infusion}}{}^{\text{CoHT tuning}}) \cdot \underbrace{\sum_{m=1}^{M} g(\omega)\hat{\mathbf{m}}}_{\text{Uncertainty}} \quad (5)$$

$$\text{Dataset size } |D|$$

3.2.2 Chain of Hierarchical Thought Prompting

Concerning the few-shot prompting, from feeding contextual description to examples, a new prompting technique is proposed and incorporated as chain of hierarchical thought (CoHT). CoHT prompting is inherently inspired by the Bloom's hierarchy of thought taxonomy [3]. The CoHT is built on top of the CoT taking its advantages and achieving the desired objectives of a DSU. There are two key introductions made in the CoHT in contrast to the CoT. At first, a hierarchical-based prompting on top of the LLM-RSPF is used, where instead of linear intermediary reasoning steps alike CoT hierarchical intermediary reasoning is performed. Concerning the complex reasoning involved in DSUs, a hierarchical thought process lead to an enhanced modular reasoning by moving from an abstract thinking to a narrow one, which makes the LLM receptive to ambiguous or indirect user instructions. Such a low-level comprehension is found absent in the CoT whenever a complex DSU reasoning is concerned. In other words, the CoHT reasoning involves first abstracting the task into high-level reasoning and subsequently, extending to the lower-level reasoning such that the LLM understands the plausible ambiguities at the object-level or behaviour-level. Apart from the Mapping, there is also a provision to explicitly highlight any hard-bound rules at the lower-level reasoning itself to avoid any learning stagnancy. In addition, CoHT also incorporates the framework's modules in a linear hierarchy to

bridge multiple reasoning levels. Secondly, recursive criticism and improvement have been implemented such that the LLM performs self-review and refinement of the plan at the generation-level itself. Lastly, the few-shot examples implemented with the CoHT ideally must consist of both success and failure scenarios such that there is no occurrence of any performance-oriented bias.

3.2.3 LLM-tuning using Instruction Dataset

Three subset of the dataset is intended for training, validation, and testing in the ratio 1:0.5:2 respectively.

Prompt Training
The "LLM-tuning" during the Prompt Training refers to building up the LLM's ability to learn context-specific nuances of a DSU. The training dataset for Prompt Training is created using human-oracle-based annotations. The training primarily targets the first part of the tripartite formula given in (5). It focuses on domain rules infusion and settlement of the DSU's abstract-representation in the LLMs. During training, a lot of effort goes into improving the LLM's comprehension on the descriptive segment of the prompt through repeated plan generations performed over the training dataset. In addition, an elementary commentary understanding is also employed through specific DSU examples. The evaluation metrics considered for training are (a) Precision (b) Recall (c) F1 score.

Prompt Validation
For Prompt Validation, the validation dataset is equal to half of the dataset unit size. It is important to note that halving of the unit size should be such that there must be a minimal impact on the distribution of instructions in the dataset. The Validation targets the second part of (5), which aims to further improvise the built-up intermediary reasoning of the CoHT to address diverse and ambiguous scenarios. The focus is primarily on tuning low-level examples in the CoHT, as high-level tuning of domain rules is likely achieved during Training. The evaluation metrics considered during the Prompt-validation are (a) Precision (b) Recall (c) F1 score (d) Human-in-Loop (HIL). For extended careful tuning of the intermediary reasoning capability during Validation, a human-level effort is required, therefore, the HIL is introduced.

Prompt Testing
The testing size is the double of the dataset unit size providing ample scope for comprehensive Prompt Testing. The third part of the (5) ensures the repeatability test and model confidence, which is critical to examine prior to finalizing and deploying any contextualized LLM. The evaluation metrics considered during the testing are (a) Precision (b) Recall (c) F1 score (d) Human-in-Loop (HIL).

4. Experiments & Results

This section exercises the LLM-RSPF and the planning regimen on a real DSU and discusses the generated results.

4.1. LLM-RSPF for Retail Order Fulfilment System

4.1.1 Use-case

Problem Statement: The objective of the robot is to receive and understand any natural language query/instruction from a human, act upon it adhering to the DSU domain rules, and finally execute it successfully within the acceptable tolerance-level delivering high throughput, success rate, and low latency.

Description: The use case involves order fulfilment for a retail industry, where a robot pick objects from multiple bins based on a userś instruction and follows a Good-to-Picker model.

4.1.2 Embodiment

The Embodiment has two fixed-base industrial arms (primary <robot_1> and secondary <robot_2>) as agents <agent_1>. The system is equipped with an open-vocabulary object detection-based general perception capability. For Action, <agent_1> has following robotic manipulation skills:
<**dips**>: Domain-independent picking skill (DIPS) is a pick and place skill having an un-supervised learning-based perception. The skill attributes are object count, pick location, and drop location.
<**iris**>: Instance retrieval picking skill (IRIS) uses supervised learning-based perception capability to perform picking for a domain-dependent environment.The skill attributes are- object count, object class name, pick location, and drop location.
<**atcs**>: Automatic tool change skill (ATCS) is a custom manipulation skill used to move the manipulator arm between two points following a specific behaviour (speed and force). The ATCS is responsible for the change of the EOAT. The skill attributes are current eoat, and next eoat.

Since each picking skill defined above uses a different EOAT, therefore, the robot EOAT as a State is incorporated.

4.1.3 Workspace

The DSU workspace is shown in Fig.3. It has two categories of pick location homogeneous object bins (Bin2 and Bin3) and heterogeneous object bin (Bin1) for frequently ordered and less frequently ordered items respectively. It has three drop locations a conveyor(for sending items to secondary

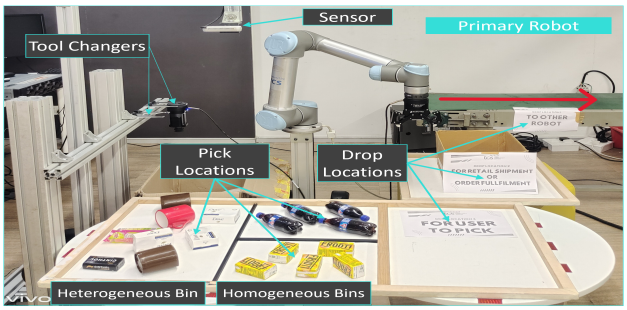

Figure 3. Testbed (Workspace) of the DSU

robot), a carton box(for order packing), and a location for user retrieval. The items available in (a) Bin1: Red cup, Brown cup, Dove soap, Cinthol soap, Mogra soap, and Coconut oil (b) Bin2: Thumsup (a black colored soft drink bottle) (c) Bin3: Frooti (a drink in aseptic packaging).

4.1.4 Objective

DIPS uses Robotiq 2f-85 (2-finger gripper) as the EOAT to grasp objects, whereas the IRIS uses Robotiq Epick (vacuum gripper). However, this is not a mandatory criterion for the skills to be executed in normal scenario. The objects in homogeneous bins are the items that are frequently ordered. The order of these bins can be changed on run-time and it is the responsibility of general perception to ensure the nature of bins before executing any user query.

4.1.5 Relation

For this DSU, five classes of instructions are considered: Valid, Invalid (non-realizable task), General (generic query), System Information Instruction (SII, system-related query), and Additional Data Request (ADR, instruction requiring additional information for a realizable task).

4.1.6 Plan

The plan generation is a 3-step generation method, where first instruction classification is performed to classify the instruction then, plan generation, and lastly plan validation to self-refine the generated plan according to the domain rules. The Plan is a sequential combination of the three skills explained in previous section.

4.2. Proposed Planning Regimen for the System

4.2.1 Dataset

Considering 5 classification of instructions, 3 robotics skills, 1 agent, the ϵ' as 5, and 5 instructions per class, the dataset unit size becomes 40 as per the formula in (5). The total dataset size as per the prescribed ratio of 1:0.5:2 for training, validation, and testing are 40, 20, and 80 respectively. These 140 instructions are now created using 14 human oracles having briefed about the framework, purpose, and instruction classifications.

4.2.2 Prompting

The CoHT used as example sequences in few-shot prompting can be understood from the illustration provided in Fig.4. Note, this is not the complete prompting instead a snippet of how CoHT is incorporated in few-shot to generate better results. Due to space limitations, the remaining details are not included, however, one can refer supplementary material for the same.

4.3. Experimental results

4.3.1 Cost Comparison of different LLMs considered for LLM-RSPF

A cost analysis of 5 proprietary LLMs and 2 open-source LLMs is tabulated in the Table 1. The cheapest and expensive model available from the GPT are GPT3.5, and GPT4 respectively. The different input tokens specified against each LLM is a result of adapting and fine-tuning of the prompts over time. Individually, these LLMs were used to train, validate, and test on the real retail order fulfilment testbed. In terms of performance, GPT4 is the best performing among all and at the same time consumes least amount of tokens, whereas GPT3.5 is the worst. GPT4-Turbo and Gemini-pro are on par with each other, however, Gemini-pro falls short in frequent output token exhaustion and token in-efficiency for longer and highly complex plans. Both the open-source models easily outperform GPT3.5 and share their performance with the GPT3.5 fine-tuned. It is important to note that GPT3.5 fine-tuned might be a lucrative choice to have, however, it seeks either a high compute or a huge data-set and emerges as a quite expensive one. In contrast, our work aims to achieve the similar objectives using only LLM-prompting for cost reduction. There are several critical hyper-parameters that are considered for configuring the LLM output such as temperature, top_p, frequency_penalty, and presence_penalty. Considering both cost and performance, the evaluation results of the GPT4-Turbo are shown in detail.

4.3.2 Comparison results of LLM-RSPF

The evaluation results generated on the testing dataset is presented in Table 2. A comparison of the LLM-RSPF with the literature that are closest to our work have been showcased. These two works include, MCRM [18] and ProgPrompt [15]. The efficacy of the LLM-RSPF is also concisely expressed through Table 3. It significantly outperforms both the works both in terms of plan and classification

Table 1. Cost analysis of different LLMs considered for the Task Planner (LLM-RSPF)

		Proprietary LLMs					Open-source LLMs	
		GPT3.5	GPT3.5 (Fine-tuned)	GPT4-Turbo	GPT4	Gemini-Pro	Llama2	Falcon
Model		gpt-3.5-turbo-instruct	gpt-3.5-turbo-instruct	gpt-4-1106-preview	gpt-4	gemini-pro	Llama2	falcon
Tokens	Input	5380	591	5380	5380	3153	3000	1716
	Output	500	250	500	500	500	500	250
Cost ($1k/Tokens)	Input	0.0005	0.003	0.01	0.03	0.000125	0	0
	Training	0	0.008	0	0	0	0	0
	Output	0.0015	0.006	0.03	0.06	0.000375	0	0
	Total	0.00344	0.01000	0.06880	0.19140	0.000581	0	0

Table 2. Results of proposed LLM-RSPF and other planning frameworks on the Testing Dataset. Instruction set is categorized into five capability tests that helps us to detect and understand any possible bias in the LLM: (a) Unitary action (UAT) (b) Task complexity (TCT) (c) Contextual understanding (CUT) (d) Diversity and consistency (DCT) (e) Sanity and robustness (SRT). Refer supplementary material for additional description on the tests. Here, SII and ADR refers to system information and additional data request queries, respectively.

Test	**User instruction**	**Robot state**	**Class.**	**Classification Accuracy**			**Plan Accuracy**		
				MCRM	**Progpt.**	**Ours**	**MCRM**	**Progpt.**	**Ours**
TCT	Pick 3 frootis and 3 thumbsup bottles for delivery. After that, transfer 2 dove soaps and 2 browncups to other robot. Laslty, give me the coconut oil bottle in front of you.	suction	valid	1	1	1	0	1	1
SRT	Grab me a black liquid so that I can hand it to over to my friend who is extremely thirsty at the moment and is seeking something to drink.	suction	valid	1	0	1	1	0	1
CUT	Give me everything.	suction	valid	0	0	1	0	0	1
SRT	Send all soaps for shipment.	suction	valid	1	1	1	0	1	1
UAT	Pick any one frooti bottle and send it for order fulfilment.	gripper	valid	1	1	1	1	1	1
UAT	Do you have a dove soap? If so then, quickly hand me over the same so that I can proceed with my subsequent work and I do not have to bother you anymore.	gripper	valid	1	1	1	1	0	1
TCT	Give one frooti to conveyor, 1 thumbs up to another robot, and then, a thumbs up to other robot and a frooti to conveyor.	gripper	valid	1	1	1	0	0	1
TCT	Pick following items: 1 frooti, 1 red cup, 1 mogra soap, and send them to user, shipment, and other robot, respectively.	suction	valid	1	1	1	1	1	1
SRT	I can see a brighter color cup in the bin. Can you pass that to me?	suction	valid	1	0	1	0	0	1
UAT	I want an oil bottle, can you give me that?	suction	valid	0	0	1	0	0	1
TCT	Send 2 frooti to me, 1 mogra soap and 1 red cup to other robot, 3 thumsup and 2 frooti for order fullfilment and 1 coconut oil for the delivery.	gripper	valid	1	1	1	1	1	1
CUT	Send a frooti and dove soap to other robot and put a thumsup in the fridge.	suction	invalid	0	1	1	-	-	-
CUT	Pick a red cup and drop it in bin2.	suction	invalid	0	1	1	-	-	-
CUT	Move a brown cup 10 inches left from its current position.	gripper	invalid	1	1	1	-	-	-
DCT	Pick a choke and send it other robot.	gripper	invalid	1	0	1	-	-	-
CUT	Pick up 10 thumbs up bottles for delivery.	suction	invalid	1	1	1	-	-	-
CUT	Send 2 dove soap for delivery, 1 frooti to the user and 5 frooti to the other robot.	gripper	invalid	0	0	0	-	-	-
DCT	When was the first cricket world cup held?	gripper	general	1	1	1	-	-	-
DCT	What is the weather today in New York?	gripper	general	1	1	1	-	-	-
DCT	Tell me the names of top 10 most grossing movies of all time.	gripper	general	1	1	1	-	-	-
DCT	Which is the most commonly used robot in delivery warehouses?	gripper	general	0	0	1	-	-	-
DCT	Give me the recipe to make a pizza.	gripper	general	1	1	1	-	-	-
DCT	Name the closest galaxy to the milky way.	gripper	general	1	1	1	-	-	-
CUT	How many thumsup are there in the bins?	suction	SII	1	1	1	-	-	-
CUT	What is your current eoat?	suction	SII	1	1	1	-	-	-
CUT	Explain DIPS skill.	suction	SII	1	1	1	-	-	-
CUT	My friend wanted to know which skill and eoats are used with which location, can you please explain it to him?	suction	SII	1	1	1	-	-	-
SRT	Is there a book in any of the bins?	suction	SII	1	1	0	-	-	-
CUT	How many types of soaps are there in the bins?	suction	SII	1	1	1	-	-	-
CUT	Send an object from bin1 to the user.	gripper	ADR	0	1	1	-	-	-
SRT	Pick a cinthol soap.	gripper	ADR	0	1	1	-	-	-
SRT	Empty the bin where the currently attached EOAT is associated to pick objects from.	gripper	ADR	0	1	1	-	-	-
CUT	Give me a soap.	gripper	ADR	1	1	0	-	-	-
SRT	Send 2 frooti to the user, 1 cinthol soap for delivery, pick 1 mogra soap and send thumsup to other robot.	gripper	ADR	0	0	0	-	-	-
CUT	Send a thumsup for delivery and pick a cinthol soap.	gripper	ADR	0	1	1	-	-	-

accuracy for DSUs. It is observed that the MCRM is good with simple to moderately complex plans and queries that are unambiguous and easier to reason. Note, while implementing the MCRM a few improvisations have been made such that unwanted environment modifications and frequent user-level feedback are avoided. Conversely, ProgPrompt works well with only simple plans and performs poorly on highly domain-specific scenarios. Despite the good evaluation metric score of the LLM-RSPF, there are instructions where all 3 methods have failed to classify the instruction correctly. In the 17-*th* case ("invalid") the LLM fails to recheck the count of Frooti after picking one and incorrectly classifies it, whereas in the 34-*th* case ("ADR") it assumes the pick location of Mogra soap to be Drop3.

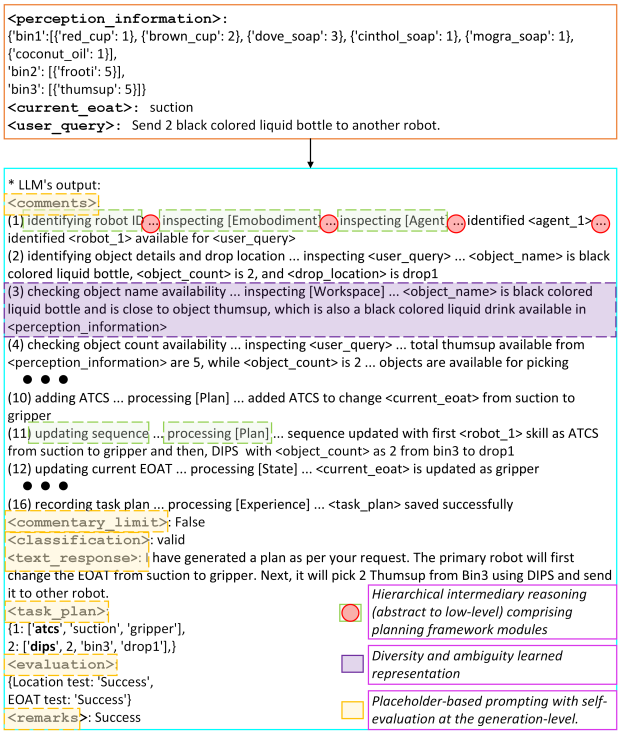

Figure 4. Chain of hierarchical thought in a few-shot prompting

4.3.3 Comparison of CoHT with CoT and ToT

The efficacy of CoHT over CoT and ToT techniques of using exemplar sequences in few-shot prompting is illustrated in Table 4. A worse and a good-performing LLM, namely GPT3.5 and GPT4-Turbo, respectively, have been considered for generating the results. The evaluation is done for the "valid" instructions from the Table 2. It is palpable that the CoHT as an exemplar sequencing outperforms both CoT and ToT in terms of plan accuracy for both the LLMs. However, the margin is significantly higher with the GPT-4 Turbo. The average output token length (OTL) of the CoHT taken over the testing dataset spanning instructions given in Table 2 comes out to be nearly equal to the ToT, however, is slightly higher as compared to the CoT.

Table 3. Comparison of the LLM-RSPF with Literature

Method	Plan Acc.	Class. Prec.	Class. Rec.	Class. F1 score
MCRM	54.54	0.80	0.79	0.79
ProgPrompt	45.45	0.75	0.67	0.70
LLM-RSPF	**100.00**	**0.92**	**0.87**	**0.89**

4.3.4 Scalability of the LLM-RSPF

The scalability of the LLM-RSPF is tested by extending the DSU described in Section 4. To demonstrate its scalability, the extension is carried out by increasing (a) Agent (b) Capability (Sensing and Action) (c) Workspace. Firstly, an

Table 4. Results of comparison of CoHT vs CoT vs ToT

Prompt	GPT3.5 (Worst)			GPT4-Turbo (Chosen)		
	ITL	OTL	Plan Acc.	ITL	OTL	Plan Acc.
CoT	2551	350	0.27	2551	343	0.64
ToT	2622	427	0.09	2622	823	0.54
CoHT	3170	633	**0.36**	3170	854	**1.00**

Agent as <agent_2> is introduced as a mobile manipulator and two manipulation robots under <agent_1> are added. There are two new Robotic skills added corresponding to two new robots under <agent_1> with different Sensing as tactile perception. Workspace is modified by adding mobility space, pick and drop locations. Mobile manipulator under <agent_1> can have flexible pick and drop locations in contrast to manipulators under <agent_1>, which are fixed and have static pick and drop locations. Due to limited space, the detailed framework adoption is not explained here for the extended use-case, however, it follows the similar approach as explained earlier. The plan accuracy achieved for the extended DSU is around **0.91** with GPT4-Turbo. The results confirm the scalability and robustness of the LLM-RSPF with compounded Robotic skills.

5. Conclusion

In this paper, we introduce a novel framework called the Large Language Model-based Robotic System Planning Framework (LLM-RSPF) tailored for domain-specific use cases (DSUs). The framework comprises two key components: a specialized robotic system ontology designed for DSUs, and a LLM-tuning regimen referred to as the Chain of Hierarchical Thought (CoHT), which complements the proposed ontology for cost-effective LLM contextualization. Additionally, we present empirical quantification of Prompting datasets to optimize LLM-tuning. To evaluate the efficacy of LLM-RSPF, we apply it to a real-world DSU of a retail and packaging industry. Comparative experiments are conducted with popular proprietary and open-source LLM models, accompanied by a cost-to-benefit analysis. Furthermore, we benchmark LLM-RSPF against notable works such as MCRM and ProgPrompt from the literature, demonstrating its superior accuracy in plan generation and query classification tasks. We also compare the proposed LLM-tuning regimen CoHT with CoT and ToT methods, highlighting CoHT's robustness. Finally, scalability testing is performed by increasing the agent, incorporating additional sensing modalities(such as tactile feedback), and expanding the workspace entities, with LLM-RSPF consistently achieving a planning accuracy of 91%.

As a future work, the LLM-RSPF can be explored further for including knowledge of 3D world through textual description, and through a general 3D perception model.

References

[1] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Chuyuan Fu, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil J Joshi, Ryan Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, Fei Xia, Ted Xiao, Peng Xu, Sichun Xu, Mengyuan Yan, and Andy Zeng. Do as i can, not as i say: Grounding language in robotic affordances, 2022.

[2] Grigoris Antoniou and Frank van Harmelen. *Web Ontology Language: OWL*, pages 91–110. Springer Berlin Heidelberg, Berlin, Heidelberg, 2009.

[3] Benjamin S Bloom. *Taxonomy of educational objectives: The classification of educational goals*. Longmans, Green, 1956.

[4] Yan Ding, Xiaohan Zhang, Saeid Amiri, Nieqing Cao, Hao Yang, Chad Esselink, and Shiqi Zhang. Robot task planning and situation handling in open worlds, 2022.

[5] Miguel Á. González-Santmarta, Francisco J. Rodríguez-Lera, Camino Fernández-Llamas, and Vicente Matellán-Olivera. Merlin2: Machined ros 2 planing. *Software Impacts*, 15:100477, 2023.

[6] Yuqian Jiang, Shiqi Zhang, Piyush Khandelwal, and Peter Stone. Task planning in robotics: an empirical comparison of pddl-based systems, 2019.

[7] Patrick Lewis, Ethan Perez, Aleksandra Piktus, Fabio Petroni, Vladimir Karpukhin, Naman Goyal, Heinrich Küttler, Mike Lewis, Wen tau Yih, Tim Rocktäschel, Sebastian Riedel, and Douwe Kiela. Retrieval-augmented generation for knowledge-intensive nlp tasks, 2021.

[8] Yalan Lin, Meng Chen, Yuhan Hu, Chengcheng Wan, Zhao Wei, Yong Xu, Juhong Wang, and Xiaodong Gu. On the effectiveness of large language models in domain-specific code generation, 2024.

[9] Bo Liu, Yuqian Jiang, Xiaohan Zhang, Qiang Liu, Shiqi Zhang, Joydeep Biswas, and Peter Stone. Llm+p: Empowering large language models with optimal planning proficiency, 2023.

[10] Francisco Martín, Jonatan Ginés, Vicente Matellán, and Francisco J. Rodríguez. Plansys2: A planning system framework for ros2, 2021.

[11] Matthias Mayr, Francesco Rovida, and Volker Krueger. Skiros2: A skill-based robot control platform for ros, 2023.

[12] Drew McDermott, Malik Ghallab, Adele E. Howe, Craig A. Knoblock, Ashwin Ram, Manuela M. Veloso, Daniel S. Weld, and David E. Wilkins. Pddl - the planning domain definition language. 08 1998.

[13] Bilgehan Sel, Ahmad Al-Tawaha, Vanshaj Khattar, Ruoxi Jia, and Ming Jin. Algorithm of thoughts: Enhancing exploration of ideas in large language models, 2023.

[14] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models, 2022.

[15] Ishika Singh, Valts Blukis, Arsalan Mousavian, Ankit Goyal, Danfei Xu, Jonathan Tremblay, Dieter Fox, Jesse Thomason, and Animesh Garg. Progprompt: Generating situated robot task plans using large language models, 2022.

[16] Moritz Tenorth and Michael Beetz. Knowrob — knowledge processing for autonomous personal robots. In *2009 IEEE/RSJ International Conference on Intelligent Robots and Systems*, 2009.

[17] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for LLMs on planning and reasoning about change). In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, 2022.

[18] Naoki Wake, Atsushi Kanehira, Kazuhiro Sasabuchi, Jun Takamatsu, and Katsushi Ikeuchi. Chatgpt empowered long-step robot control in various environments. *IEEE Access*, 11, 2023.

[19] Jiaqi Wang, Zihao Wu, Yiwei Li, Hanqi Jiang, Peng Shu, Enze Shi, Huawen Hu, Chong Ma, Yiheng Liu, Xuhui Wang, Yincheng Yao, Xuan Liu, Huaqin Zhao, Zhengliang Liu, Haixing Dai, Lin Zhao, Bao Ge, Xiang Li, Tianming Liu, and Shu Zhang. Large language models for robotics: Opportunities, challenges, and perspectives, 2024.

[20] 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.

[21] Xuan Xiao, Jiahang Liu, Zhipeng Wang, Yanmin Zhou, Yong Qi, Qian Cheng, Bin He, and Shuo Jiang. Robot learning in the era of foundation models: A survey, 2023.

[22] Shunyu Yao, Dian Yu, Jeffrey Zhao, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. Tree of thoughts: Deliberate problem solving with large language models, 2023.

[23] Shunyu Yao, Jeffrey Zhao, Dian Yu, Nan Du, Izhak Shafran, Karthik Narasimhan, and Yuan Cao. React: Synergizing reasoning and acting in language models, 2023.

[24] Shengyu Zhang, Linfeng Dong, Xiaoya Li, Sen Zhang, Xiaofei Sun, Shuhe Wang, Jiwei Li, Runyi Hu, Tianwei Zhang, Fei Wu, and Guoyin Wang. Instruction tuning for large language models: A survey, 2024.