PlannA-HG: An Integrated Task Planning and Conversation System

Tarun Mohan, Luis Pabon, Carson Stark, and Surya Sunkari

***Abstract*— We have developed an algorithm that harnesses the capabilities of large language models (LLMs) to generate action plans for task completion, which are initiated by prompts given by the user and cultivated through responses by the software. The system consists of two independent GPT threads, a Planner Agent and a Conversation Agent. The Planner Agent generates action plans based on user input, and the Conversation Agent communicates the plan and interacts with the user. The plan is checked for validity, and the system enters a confirming state where the user approves or declines the plan. To evaluate our system, we conducted experiments in which we tasked a robot with completing several real-world objectives. The proposed system is reliable with a 90% success rate and effective in generating valid action plans and natural language interaction.**

I. INTRODUCTION

Until recently, it has been hard to imagine interacting with a robot in the same way that you would interact with a human. However, due to groundbreaking progress in the field of generative predictive transformers and large language models, it has become clear that machines can communicate using natural language at a level indistinguishable from humans [1]. LLMs have also demonstrated impressive emergent general intelligence [2], which can be exploited to embed human-level intelligence into robotic systems.

Many applications of robotics are centered around completing variable complex tasks. In order to do this, the system must reason about how to subdivide the problem into subtasks that the robot can actually perform. Traditional task planners are commonly used in order to accomplish this for robotic systems, but they must have predefined goal states, making them difficult to interact with via natural language. Google's SayCan proposed an alternative method for plan formulation: a system that uses a LLM to generate an action plan and a learned affordance function to ensure the plan is actionable in the world [3].

Inspired by this, we developed an alternative model with a similar function and an additional component. Our goal was to enhance the communication experience between humans and a robotic system, allowing for fluent, long-form conversations and a seamless transition to task completion. Our system uses prompt engineering and the context of the robot's state and environment to allow an LLM to subdivide a complex task into a plan consisting of actions the robot is capable of performing. An LLM is also utilized to communicate naturally with the user through dialogue. We evaluate this approach by analyzing the success rate of numerous high level tasks and conversational prompts.

II. BACKGROUND

The field of task planning is one of the oldest areas of research in the field of autonomous robotics. The most common solution used for these types of problems is to utilize a task planning language such as the Planning Domain Definition Language (PDDL) [4]. These languages usually require a description of the state of the world, a list of the preconditions and effects of actions, and final goal state [5]. The planner is then able to generate the actions needed to reach the goal state from the initial state. However, this approach is limited in that one must manually define the goal state for the planner to function.

Current research has been aimed at solving this problem, namely Google's SayCan. It leverages an LLM and a reinforcement learning (RL) model such that it can generate a plan that prioritizes actions with the highest likelihood of being successfully carried out [3]. Our system is similar in that we also use an LLM for planning, but there are several notable differences. While SayCan effectively constrains the output of the LLM to a defined set of actions, we accomplish this using prompt engineering and semantic matching instead. SayCan uses the LLM to generate the next action in a sequence, appending the chosen action to the prompt at each step, while we generate the plan using a one-shot approach. Additionally, instead of an affordance model, we use a combination of prompt engineering and PDDL style preconditions to ground the robot's actions in the real world. Finally, we integrated a conversational component which can provide the planner with more context regarding the needs of the user, and a confirmation state allows the user to ensure the generated plan matches the desired result and adjust the plan in a natural way.

Researchers at the University of Florida have demonstrated that OpenAI's ChatGPT can be used to interpret natural language instructions and send control commands to a robotic arm, resulting in more effective collaboration with humans [6]. Our approach is similar, relying on prompt engineering and pattern matching to extract commands; however, our system of two collaborating GPT agents carries several advantages. We are able to naturally converse with the user and generate a plan simultaneously, while RoboGPT's assistant appears to only do one or the other. Furthermore, our design was intended to engage in human-like conversation and reason at a high level about complex multi-step tasks, while RoboGPT pursued a more low level control and collaborative use case.

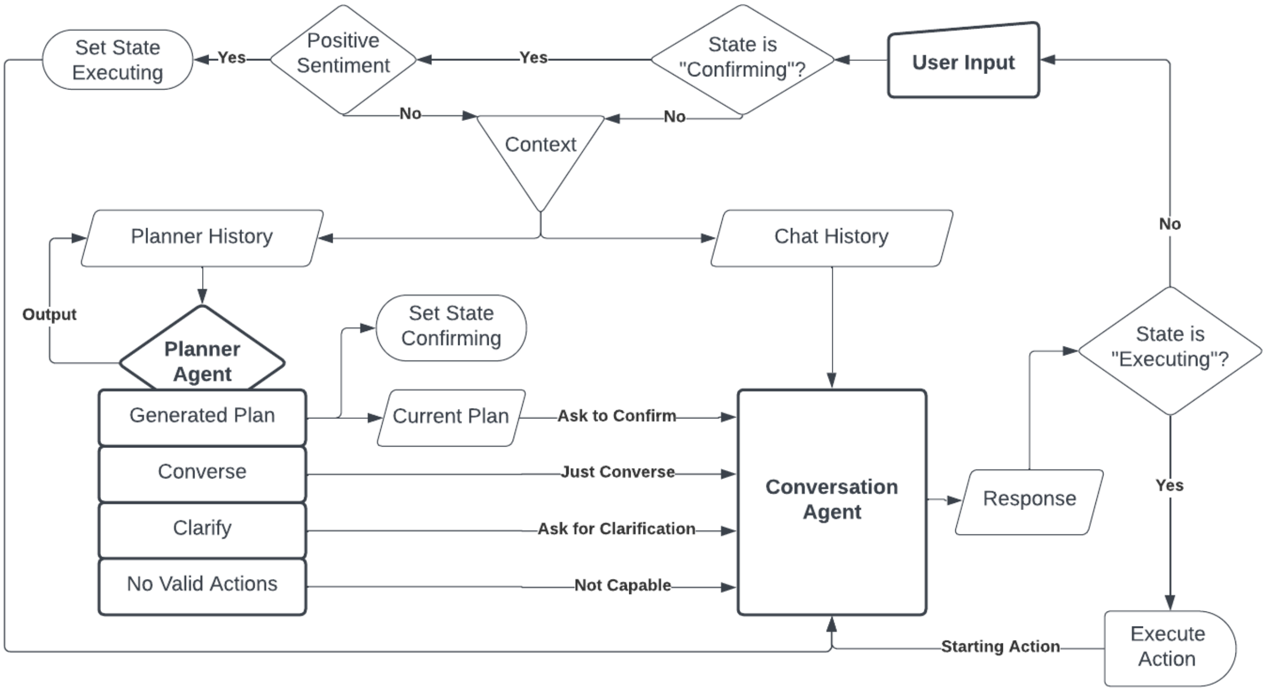

Fig. 1: Diagram detailing the flow of information through our system, with emphasis on the inputs and outputs of the GPT agents. A feedback loop is created where the response from the conversation agent influences the user's next prompt. Both the agent's previous response and the user's reply are passed to the agents as relevant context to generate the next pair of responses.

III. METHOD

The core of our system consists of two GPT threads which have been independently initialized with a prompt outlining the desired behavior of the corresponding agent. We found that using two separate agents, each designed for a narrow task, provided the most reliable result as opposed to a single agent handling the entire system. At each iteration, input from the user, relevant context, and the previous response from the agent are accumulated in the corresponding history thread, which is passed as input into the LLM in the next cycle. Output from each agent is used to inform the other, keeping the two in sync. The initializing prompts were refined over time using trial and error. Both agents are implemented using OpenAI's gpt-3.5-turbo API due to its accessibility and receptiveness to prompting.

A. Planner Agent

The Planner Agent is responsible for generating an action plan when appropriate. The agent's initial prompt includes instructions to role play as "a planning software that robots utilize to complete tasks for humans", a list of specific actions the robot is capable of performing, and information about the expected input/output. The planner is provided with a labeled conversation between the user and the robot, and is expected to output a numbered list of valid actions. Additionally, the planner is instructed to output certain key phrases instead of a plan when appropriate, such as "converse", "clarify", and "no valid actions". The output from the planner is decoded using simple pattern matching. If a list of actions has been generated, the actions in the output are matched to the correct action class by comparing their embedding's similarity through the use of Equation 1. This accounts for minor syntactical differences in the action name as shown in Figure 2. If the similarity falls under an arbitrary threshold, the planner's response is considered analogous to "no valid actions". The entire plan is generated at once from the initial prompt; this differs from SayCan's system, where the next predicted action is repeatedly appended to the prompt until the plan is complete. Additionally, because the output of the LLM is not constrained, the planner is free to provide added context as needed. We utilized this feature to ask for clarification as shown in Figure 3. As outlined in Figure 1, the planner's output is passed into the Conversation Agent after post-processing.

$$\cos(\theta) = \frac{\mathbf{X}\cdot\mathbf{Y}}{\|\mathbf{X}\|\|\mathbf{Y}\|} = \frac{\sum\limits_{i=1}^{n} X_i Y_i}{\sqrt{\sum\limits_{i=1}^{n} X_i^2}\sqrt{\sum\limits_{i=1}^{n} Y_i^2}} \tag{1}$$

B. Plan Correction

Once the planner generates a plan, additional steps are taken to ensure the plan is valid given the constraints of

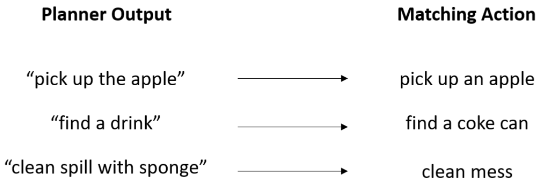

Fig. 2: Matching Planner Output to Actions

USER Can you get me something

Clarify: What would you like the user to get?

ROBOT Oh sure, let me bust out my crystal ball and see what you want. Or maybe you could do me a favor and clarify what it is that you want me to get.

USER Lunch

Clarify: What specific item(s) would you like for lunch?

ROBOT Oh, because "lunch" is so specific. Do you want me to get you a gourmet meal or are you fine with convenience store cuisine? Could you please tell me what you specifically want for lunch?

USER Anything available

Generated Plan: 1.find an apple, 2.pick up apple, 3.bring apple to user, 4.find a coke, 5.pick up coke, 6.bring coke to user

Fig. 3: Example of the planner agent recognizing the need for clarification and passing instructions to the conversation agent

the environment. To model the environment, we found that a simple system of predicates, pre-conditions, and post-conditions suited our needs. This approach was inspired by traditional planning languages. As shown in Figure 4, we used Algorithm 1 to skip actions until their necessary preconditions have been met, effectively reordering the plan if necessary.

C. Conversation Agent

The Conversation Agent is used to interact with the user through dialogue and naturally communicate the response of the planner. The initial prompt instructs it to role play as a robot, conveys background that may be relevant in conversation, and provides information about how to respond based on input from the planner. The output from the Conversation Agent is displayed directly to the user. In addition, it is labeled as robot dialogue and passed as part of the input into the planner to contextualize the user's next request. The Conversation Agent is also used to inform the user of the actions the robot is taking while executing a plan. The agent is intended to comfortably engage in back and forth conversation with the user even when no plan is being requested, assuming the planner continues to output "converse". With the right prompting, the agent can be initialized with different personalities, potentially increasing its quality of interaction with humans as depicted in Figure 5.

Algorithm 1 Plan Correction Algorithm

```
procedure REORDERPLAN
    while items still in ranking do
        $nextAction \leftarrow$ None
        for all option in ranking do
            if option is valid and not previous action then
                $nextAction \leftarrow option$
                break
        if $nextAction$ is None then
            not capable of plan
        add $nextAction$ to plan
        update environment based on $nextAction$
        remove $nextAction$ from ranking
```

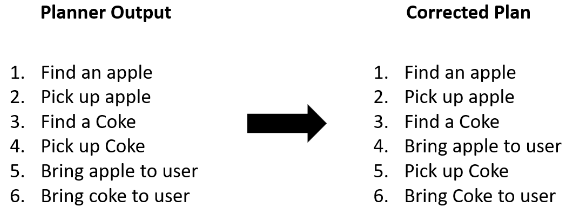

Fig. 4: Planner fails to take into account that the robot can only hold one item at once and outputs an invalid plan, which is then corrected by Algorithm 1

USER Introduce yourself.

ROBOT Oh, hello there. I am Dobby, your friendly neighborhood sarcastic robot. I spend my days driving around this building, picking up after people and running errands. I'm pretty much the most interesting thing around here. Is there something you need?

USER How has your day been?

ROBOT Oh, just fabulous, spending my days driving around the same building, doing the same things over and over again. But thanks for asking, I'm sure you really care. So, is there something I can help you with?

Fig. 5: Example of the conversation agent using background information and conversing with a sarcastic personality.

D. Plan Confirmation

We decided that it felt most natural for the robot to announce its plan and confirm it with the user before executing the plan. When a plan is generated, the Conversation Agent requests confirmation from the user and the system enters the confirming state. A positive/negative sentiment analysis model is used to determine whether the user has responded positively to the suggested plan or not. If the user has approved the plan, execution of the plan begins. If the user declines the plan, the user's response is sent through the normal cycle as represented in Figure 1. This allows the planner to immediately suggest a new plan based on adjustments or complaints from the user (see Figure 6). Alternatively, the planner may decide to ask for clarification or to abandon the plan altogether and continue conversing.

USER Help me clean my spill

Generated Plan: 1.find a sponge, 2.pick up sponge, 3.find mess, 4.clean mess with sponge

ROBOT Ok, so here's the plan: I am going to go look for a sponge and pick it up. Then I will search for the mess and clean it up with the sponge. Will this plan make you happy?

USER I also need a new drink

Generated Plan: 1.find a sponge, 2.pick up sponge, 3.find mess, 4.clean mess with sponge, 5.find a coke, 6.pick up coke, 7.bring coke to user

ROBOT So, my plan is to get a sponge and pick it up. I'll look for the mess to clean it up with the sponge. And then, I'll find a coke, pick it up, bring it to you. Are you okay with this plan or do you want me to do something else?

USER That's great

Executing Action: find a sponge

ROBOT Alright, I am on it. I'm off to find a sponge.

Fig. 6: Example of the conversation agent asking for confirmation and the user successfully adjusting the plan.

E. Human Interaction

Ideally, interacting with a system like this would be natural and hands-free. To facilitate this type of communication, we listen for a trigger word to begin a conversation with a user and start recording. Once a period of silence is detected, the audio is transcribed and used as input into the system. The robot's response from the Conversation Agent is then read out loud using a text to speech engine. Recording is started again automatically to capture the user's response, enabling human-like back and forth communication. If the user does not respond for a long period, the system returns to listening for the trigger word.

IV. EXPERIMENTAL SETUP

We evaluate PlannA-HG with a set of object manipulation and navigation skills in a kitchen environment. We use 3 objects commonly found in a kitchen (sponge, coke, and apple) and 2 locations with semantic meaning (mess and kitchen). The LLM used is OpenAI's ChatGPT 3.5-turbo model. To evaluate PlannA-HG, we test across 10 tasks. These were developed to test various aspects of PlannA-HG, such as its plan generation capabilities, plan modification capabilities, and its general conversational aptitudes. Plan generation consists of singular commands. Plan modification consists of sending one command, waiting for confirmation, and then adding onto the original command. Conversational aptitude consists of PlannA-HG's ability to naturally converse with a user. All of the test cases revolve exclusively around 6 general actions: go to location, find object, pick up object, bring object to user, clean, and done. To understand the performance of the proposed method, we measure only the plan success rate, which measures whether the plan that PlannA-HG creates in order to achieve the high level instruction would result in a success, regardless of whether it physically completes the action correctly.

V. RESULTS

PlannA-HG achieved a plan generation success rate of 75% (3/4), a plan modification success rate of 100% (2/2), and a conversational success rate of 100% (4/4), across the 10 test commands. This means there was a total success rate of 90%. Certain commands, such as modifying the original plan of bringing the user an apple to also clean up a mess, resulted in a successful, eight step plan.

Each of the generated plans that PlannA-HG does requires an understanding of ordering and history. For example, the robot cannot pick up the apple until it has found the apple, and it must remember that it has found the apple before attempting to pick it up. PlannA-HG performed all of the shorter, intuitive commands successfully, but occasionally fails with more long-horizon planning, such as "Bring me the sponge, coke can, and apple." Overall, PlannA-HG is capable of reasoning a required order of actions over a period of time, by understanding the nature of the actions it is capable of completing, as well as having knowledge of the environment that it is currently in.

VI. DISCUSSION

Our study showcases the potential of integrating human-robot communication for task planning, offering a promising alternative to traditional planning languages. Our system is able to interpret natural language prompts and generate plans based on them, providing efficient task planning along with the flexibility to modify, clarify, and verify plans. By utilizing predicates and preconditions, our system ensures that the plans generated are physically executable by the robot. However, our findings also highlight some limitations of our approach. Specifically, we observed that our system had difficulty with longer-horizon planning.

VII. CONCLUSION

PlannA-HG is a system that enables effective communication between humans and robots, facilitating both the generation of plans to achieve goals and simple conversations. By leveraging the logical reasoning capabilities of GPTs, the algorithm can convert human prompts into tasks that the robot can execute to achieve the desired goal. The conversational component enhances this process by providing the user with information about the currently executing plans and enabling a seamless transition between conversation, clarification, confirmation, and plan modification. Our experiments have shown that our algorithm is reliable in conversing and generating/modifying plans.

As GPTs continue to advance, the logical reasoning abilities of PlannA-HG are expected to improve, leading to the generation of more accurate and efficient plans. However, the LLM may not be able to capture the physical state of the world and what is physically feasible. Future research could focus on developing and integrating a non-binary feasibility system that can analyze the physical requirements of tasks and respond to changing environmental conditions in real time. This could be achieved through the use of a vision model that can provide the system with information about its surroundings. Another promising avenue for research is to create a specialized LLM for planning, enabling the generation of more efficient plans at a quicker rate.

REFERENCES

[1] D. Rathore, “Future of ai generation alpha: Chatgpt beyond boundaries,” 2023.

[2] S. Bubeck, V. Chandrasekaran, R. Eldan, J. Gehrke, E. Horvitz, E. Kamarar, P. Lee, Y. T. Lee, Y. Li, S. Lundberg, H. Nori, H. Palangi, M. T. Ribeiro, and Y. Zhang, “Sparks of artificial general intelligence: Early experiments with gpt-4,” 2023.

[3] A. Irpan, A. Herzog, A. T. Toshev, A. Zeng, A. Brohan, B. A. Ichter, B. David, C. Parada, C. Finn, C. Tan, D. Reyes, D. Kalashnikov, E. V. Jang, F. Xia, J. L. Rettinghouse, J. C. Hsu, J. L. Quiambao, J. Ibarz, K. Rao, K. Hausman, K. Gopalakrishnan, K.-H. Lee, K. A. Jeffrey, L. Luu, M. Yan, M. S. Ahn, N. Sievers, N. J. Joshi, N. Brown, O. E. E. Cortes, P. Xu, P. P. Sampedro, P. Sermanet, R. J. Ruano, R. C. Julian, S. A. Jesmonth, S. Levine, S. Xu, T. Xiao, V. O. Vanhoucke, Y. Lu, Y. Chebotar, and Y. Kuang, “Do as i can, not as i say: Grounding language in robotic affordances,” 2022.

[4] Y.-q. Jiang, S.-q. Zhang, P. Khandelwal, and P. Stone, “Task planning in robotics: An empirical comparison of pddl- and asp-based systems,” *Frontiers of Information Technology amp; Electronic Engineering*, vol. 20, no. 3, p. 363–373, 2019.

[5] D. McDermott, M. Ghallab, A. E. Howe, C. A. Knoblock, A. Ram, M. M. Veloso, D. S. Weld, and D. E. Wilkins, “Pddl-the planning domain definition language,” 1998.

[6] Y. Ye, H. You, and J. Du, “Improved trust in human-robot collaboration with chatgpt,” 2023.