Bootstrapping Human-Like Planning via LLMs

David Porfirio[1], Vincent Hsiao[2], Morgan Fine-Morris[2], Leslie Smith[1], and Laura M. Hiatt[1]

***Abstract*— Robot end users increasingly require accessible means of specifying tasks for robots to perform. Two common end-user programming paradigms include drag-and-drop interfaces and natural language programming. Although natural language interfaces harness an intuitive form of human communication, drag-and-drop interfaces enable users to meticulously and precisely dictate the key actions of the robot's task. In this paper, we investigate the degree to which both approaches can be combined. Specifically, we construct a large language model (LLM)-based pipeline that accepts natural language as input and produces human-like action sequences as output, specified at a level of granularity that a human would produce. We then compare these generated action sequences to another dataset of hand-specified action sequences. Although our results reveal that larger models tend to outperform smaller ones in the production of human-like action sequences, smaller models nonetheless achieve satisfactory performance.**

I. INTRODUCTION

Robot end-user programming (EUP) tools assist novice or otherwise non-expert users in intuitively and effectively specifying tasks for robots to perform. Many such approaches design programming graphical user interfaces (such as drag-and-drop interfaces) that users can employ to correctly and fluently specify the robot's desired behavior. Such approaches give users precise control over robot behavior, but can be difficult for novices [1] or have a learning curve [2]. Other approaches that encourage users to incompletely or abstractly specify the robot's task, such as by specifying only a subset of task checkpoints and relying on an automated planner to come up with an actionable plan for the robot to execute, require a shift in task comprehension [3].

This problem is not unique to EUP. People in general find it difficult to generate detailed, efficient plans, even if they are meant for themselves [4], [5]. People often create physical artifacts, elaborate plans, and routines around scheduled activities to make it easier to manage the mental load of scheduling complicated goals under multiple constraints [6].

Here, we are interested in developing an approach for *bootstrapping* the EUP workflow based on an initial, small set of human-provided natural language (NL) commands. We envision accomplishing this via a multi-step *command-to-action-sequences (CAS)* pipeline that serves as the initial source of input to the EUP tool, as depicted in Figure 1. Prior

*This work was supported by NRL

[1]Navy Center for Applied Research in AI, US Naval Research Laboratory, Washington, DC 20375, USA {david.j.porfirio2, leslie.n.smith20, laura.m.hiatt}.civ@us.navy.mil

[2]NRC Postdoctoral Researcher, Navy Center for Applied Research in AI, US Naval Research Laboratory, Washington, DC 20375, USA {vincent.hsiao, morgan.f.fine-morris}.ctr@us.navy.mil

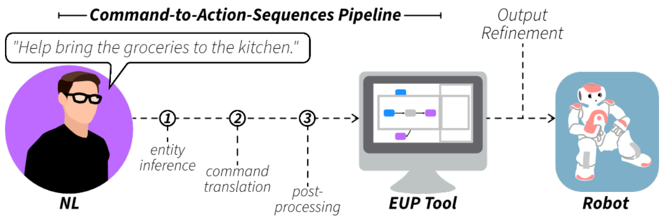

Fig. 1. We envision a multi-step natural language command-to-action-sequence (CAS) pipeline as an entry point to robot end-user programming.

to using the tool, users provide an NL command of what they want the robot to do. Then, the CAS pipeline uses an LLM to identify symbolic entities in the world that are relevant to the command. Next, a *translator Large Language Model (LLM)* translates the command and relevant entities into a symbolic representation of the key actions that the robot must perform to fulfill that command. Post-processing aligns the symbolic actions output by the LLM with the actual actions in the domain of interest. Once the CAS pipeline has run to completion, users can then refine the action sequence via a drag-and-drop interface.

As part of this pipeline, we strongly focus on generating *human-like* action sequences from the NL commands. Mismatches in how the human *expects* their NL to be interpreted by the robot and how their NL is *actually* interpreted require explanations, which can be costly if not applied appropriately [7], or techniques for reconciling the human's expectation with the robot's behaviors [8]. This can place a high burden on the interface and user during the refinement of the action sequence. The production of *human-like* action sequences is thus crucial to our objective, differentiating our research from prior successes in using LLMs within automated task planning [9], [10], [11].

To address this, we seek to understand how well LLMs translate an NL command to a sequence of key actions that are similar to those a human would produce. We consider several LLMs for this role, including LLMs of various sizes and LLMs that are fine-tuned on a dataset of hand-created task plans for common household tasks. Overall, we show that the CAS pipeline produces human-like action sequences and how smaller LLMs perform compared to larger ones in several metrics. Our contributions are as follows:

- *Technical*: a CAS pipeline that translates NL commands into *human-like* action sequences.
- *Empirical*: a characterization of how different LLMs affect the performance of the CAS pipeline.
- *Design*: implications for integrating the CAS pipeline into future robot EUP systems.

II. Related Work

Our research is informed by robot EUP, LLMs for task planning, and datasets of human-robot task communication.

A. End-User Robot Programming

Robot EUP has traditionally involved meticulous *block-based* [12], [1], *flow-based* [13], [14], or *trigger-action* based [15] visual programming. Recent research has focused on how novel interfaces can be designed to facilitate intuitive use of each of these paradigms [16], [17], [18].

Overwhelmingly, robot EUP interfaces are *action-oriented* [3], meaning that users express robot tasks in terms of hand-crafted *task plans*, or the sequences of actions that a robot must perform in order to achieve a goal. Hand-crafting, rather than automating, the task planning process is vital for enabling end-users to shape *how* robots perform tasks. However, building tasks from scratch can be difficult for novice users, as evidenced by prior work [3], [1].

Due to the intuitiveness of NL for human communication, *natural language programming* is a promising solution for overcoming the difficulty of hand-specifying action sequences. Historically, NL has a rich history in robot EUP, with prior work mapping NL tokens to programmatic representations [19], [20], [21], [22], [23]. More recently, EUP systems have successfully leveraged LLMs in order to generalize beyond specific domains and alleviate the need for explicit mapping between NL and symbols [24], [25], [18]. These works excel at translating NL to individual commands, though none explore the use of LLMs for replicating hand-specified action sequences for the robot to perform.

B. LLMs for Planning

There has been a recent surge of work that investigates how LLMs and Large Reasoning Models (LRMs) can effectively be used for planning [9], [11], [10]. Despite the power of these types of models, planning remains a challenge [26], with LLM-based planners unable to meet many established planning benchmarks. Unlike prior work that focuses on generating correct, complete, or executable plans, or plans grounded in a specific context [27], our work has a different purpose. Informed by prior work in explainable AI that suggests humans attribute human-like psychological mechanisms to AI agents [28], we instead aim to translate NL commands to action sequences in a manner similar to how a human would do it. This can then be used to pre-populate a visual EUP interface to assist the human with making further refinements to the plan. From an LLM planning perspective, this allows humans to correct the (messy/incorrect) plan output that LLMs produce.

C. Robot Task Specification Datasets

Numerous datasets exist for human-robot task communication. Of these datasets, only a few pair NL task descriptions with discretized sequences of a user's desired high-level robot actions. As part of this work, we utilized the *VirtualHome* Activity Dataset [29]. This dataset contains human-generated NL descriptions of common household activities. For example, a user may describe throwing away a newspaper as: *Take the newspaper on the living room table and toss it*. These descriptions are paired with human-generated "programs" that accomplish them, represented in a predicate-based form. There are 2821 pairs of descriptions/programs in the dataset that span numerous household tasks. Separate from *VirtualHome*, Porfirio et al. (2023)'s *Task Traces* dataset asked crowdworkers to enumerate sequences of actions that they would perform to achieve an objective and optionally pair individual actions with NL descriptions [30]. Another example is the *ALFRED* dataset, which is similarly comprised of action sequences paired with NL descriptions [31]. In contrast to its counterparts, however, *ALFRED*'s action sequences are not created by hand. Rather, they are produced by an automated task planner.

III. Technical Approach

The goal of this work is to use LLMs to translate between NL commands and the human-like action sequences that accomplish them. By NL command, we mean a spoken command that a person says to a robot in order to get the robot to perform desired behavior(s). It can be general (*i.e.*, "get the mail") or specific (*i.e.*, "get the mail from the mail slot, sort it, and leave it on the kitchen counter"), depending on the user's preferences. Given this command, the robot should automatically generate a symbolic representation of how it believes it can best operationalize that behavior. We envision the user then being able to adjust the behavior in a traditional EUP interface, if necessary.

The CAS pipeline has three main steps to perform this translation task, each depicted in Figure 2. First, *Entity Inference* narrows the known entities in the environment to those that are relative to the NL command being translated. Second, the *Command Translation* step uses an LLM to translate the NL command and relevant entities into an action sequence. Third, a *Post Processing* step maps the action sequence's commands to ones that are pertinent to the domain. We next describe each step in more detail.

A. Entity Inference

Given the set of known entities in the robot's environment, the first step in our pipeline is to narrow this set to the entities that are pertinent to the user's command. To do this, we use off-the-shelf Mistral Codestral 22B v0.1. For example, given an NL command, "Find your roommate and tell them they have a phone call," the prompt looks like:

> *The set of entities in the world is: master_bedroom_lamp, bedside_table, desk, master_bedroom, hallway, bathroom, car, garage, bedroom, bedroom_lamp, refrigerator, kitchen_cabinets, countertop, kitchen, back_door, table, living_room, living_room_lamp, entrance, coffee_table, front_door, living_room_cabinets, vacuum, clock.*
>
> *Given the command "Find your roommate and tell them they have a phone call", provide a short list of the entities only from this set that relate to this command.*

The list of known entities can change depending on the robot's domain. Given the prompt above, the output is `phone, roommate`.

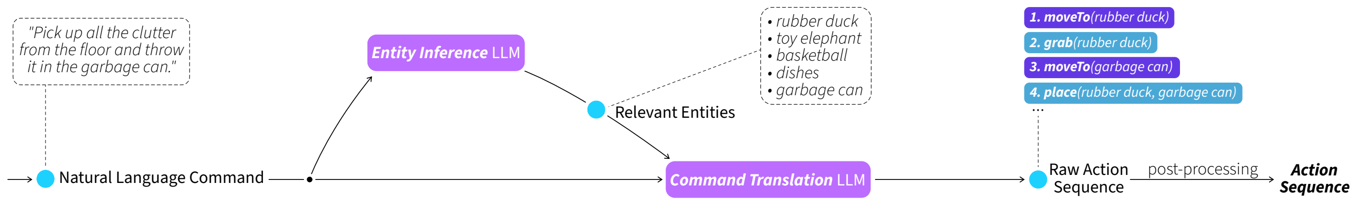

Fig. 2. The CAS pipeline for translating NL commands into action steps.

B. Command Translation

Next, the CAS pipeline uses another LLM to perform command translation, where the NL command is translated into an action sequence. To perform this translation, we provide the LLM with prompts of the following form, filling in the entities and the task description as appropriate:

You are a robot. Given a task expressed in NL, you need to produce the steps necessary to achieve the goals of the task.

The entities in the environment are as follows: phone, roommate

Your task in NL is as follows:
"Find your roommate and tell them they have a phone call."

If using a pretrained (not fine tuned) LLM for command translation, the prompt additionally requires a list of actions in predicate form that are available to the robot. If fine-tuned on a dataset of action sequences (*e.g.*, the *VirtualHome* dataset), a list of available actions is not necessary. An example output from the above prompt is `Walk(roommate)`, `Find(phone)`, `Grab(phone)`, `TurnTo(roommate)`, `LookAt(roommate)`, `PointAt(phone)`, `Talk(``I found my phone!'')`, `PutObjBack(phone)`.

C. Post-Processing

The command translation step produces a raw action sequence (Figure 2, right) that requires post-processing before being given to an EUP tool or a robot. If the command-transation LLM has been fine-tuned on a dataset like *VirtualHome*, the actions in the raw output may differ from the actions that are recognizable by the EUP tool or the robot. Keyword-based mapping between the dataset actions and the set of acceptable actions suffices here. Dataset actions that fail to map to acceptable actions, refer to nonexistent entities, duplicate the previous action, or are extraneous (such as `wait()`, which does nothing) are removed.

IV. EVALUATION

We evaluated the effectiveness of the CAS pipeline's ability to produce action sequences that match human-produced action sequences. Our evaluation considers five different LLMs for the *Command Translation* step, depicted in Table I. Our evaluation focuses on characterizing how lightweight, smaller LLMs perform this task against larger, heavyweight models, and how fine-tuned models perform against their pre-trained counterparts. Note that the pretrained version of Mistral 7B (**M**) did not produce consistently usable output; thus, **M** is excluded from further analyses.

ID	Model	Fine-Tuned	Parameters	Included in Further Analysis
$\mathbf{M}$	Mistral		7B	
$\mathbf{M}_t$	Mistral	✓	7B	✓
$\mathbf{P}$	Phi-4		14B	✓
$\mathbf{P}_t$	Phi-4	✓	14B	✓
$\mathbf{S}$	Sonnet v2		unk. (large)	✓

TABLE I
LLMS USED IN OUR EVALUATION.

A. Fine-Tuning $\mathbf{M}_t$ and $\mathbf{P}_t$

Using QLoRA [32], we fine-tuned $\mathbf{M}_t$ and $\mathbf{P}_t$ on the *VirtualHome* dataset described in §II-C, which contains 2821 hand-generated NL descriptions paired to hand-created action sequences. $\mathbf{M}_t$ was fine-tuned with 1000 steps. $\mathbf{P}_t$, a larger model, was fine-tuned with 2000 steps. Our training-validation split is 80% and 20%, respectively.

Due to being fine-tuned with the *VirtualHome* dataset, $\mathbf{M}_t$ and $\mathbf{P}_t$ produce actions defined via *VirtualHome* predicates. As described in §III-C, we use a keyword-based approach to map *VirtualHome* actions to those accepted by our pipeline.

B. Evaluation Dataset

The *VirtualHome* dataset, while large and comprehensive for capturing how humans describe tasks using natural language, is not strongly indicative of EUP paradigms. Specifically, participants were required to specify actions sequences that included *all* steps necessary for a robot to execute a task [29]. In contrast, many EUP paradigms allow users to omit implied steps and let a planner fill in the small details automatically. To this end, we looked to a different dataset to evaluate how well this pipeline would apply to EUP tools.

Evaluating the CAS pipeline requires a dataset that pairs action sequences with a single, overarching NL description. The dataset must also be comprehensive, that is, encompass multiple task categories. Few datasets exist that fully satisfy these requirements, though the *Task Traces* dataset [30] comes close. This dataset is comprehensive—it is comprised of 207 action sequences created by crowdworkers for 18 different task categories that someone may perform around the home, ranging from getting the mail to answering the front door. However, the dataset does *not* pair NL descriptions to *whole* action sequences. Rather, crowdworkers optionally annotated individual actions with individual NL descriptions. A simplified task, example action sequence, and its corresponding description sequence is shown in Figure 3; see [30] for full details of the dataset.

Task: *Imagine that you are home and the phone rings. The person on the other end of the line asks to speak to your roommate.*

User action sequence and descriptions:
Step: 0. move to<[{'argname': 'destination', 'argval': 'Bedroom'}]>
Individual NL Command: "look for him"
Step: 1. approach<[{'argname': 'person', 'argval': 'Roommate'}]>
Individual NL Command: "inform him about the call"
Step: 2. place<[{'argname': 'object', 'argval': 'phone '},
{'argname': 'destination', 'argval': 'Table'}]>
Individual NL Command: "place the phone on the table and ask him to talk"

Fig. 3. Simple data point from the *Task Traces* dataset [30].

Of the 207 action sequences, we first eliminated 150 that were purely social tasks or contained critical actions that lacked NL descriptions. Determining *critical* actions involved an author reviewing each action by hand to determine if its inclusion in the sequence is implied by a future action; if not, the action is critical. For example, if an action sequence is {**move_to**(broom), **grab**(broom)}, then **move_to**(broom) is not critical while **grab**(broom) is. We then eliminated another 17 sequences where the action sequence did not match the corresponding description sequence (such as if a user described putting down the groceries, but the actions specify only moving into the kitchen). This second stage of elimination was performed by two authors coding each example separately and then meeting to resolve differences. 40 action sequences remained.

We then augmented the *Task Traces* dataset to ensure that it meets the requirement of having a single overarching NL description per whole action sequence. Specifically, for each action sequence, we created overarching NL summaries of the individual NL commands paired to individual actions. Three summaries were created per action sequence. Two of these summaries were generated by the authors of the paper, who were instructed to consider the task, the NL descriptions paired to each action in sequence, and the actions themselves. Additionally, we used off-the-shelf Anthropic Sonnet 3.5 v2, with a temperature of 0, to generate a third overarching NL summary. The prompt began with a description of the task similar to that in the original dataset, followed by instructions for the LLM and the NL descriptions of the action sequence:

> *You are home and the phone rings. The person on the other end of the line asks to speak to your roommate. Please summarize the following steps as a short sentence. Skip or combine unimportant steps. Phrase it as a command to yourself in the second person.*
>
> *1. look for him*
> *2. inform him about the call*
> *3. place the phone on the table and ask him to talk*

An example human-generated summary of these steps is *Inform my roommate of the call and place the phone on the table for him.* Example LLM-generated output is *Find your roommate and tell them they have a phone call.*

C. Evaluation Measures

Our evaluation includes several measures intended to compare LLM-generated action sequences to their human-produced counterparts. We consider action sequences to be human-like if they (1) contain the same actions as their human-created counterparts (*action similarity*); (2) are similar to their human-created counterparts in the final state of the world resulting from executing the sequences (*final state similarity*); and (3) are substantially different in length to their human-created counterparts (*length discrepancy*).

First, we consider *action similarity* via two separate measures, the first being *plan difference*, which is a measure of distance between two action sequences that focuses on actions that are different between the two sequences, while de-emphasizing the order in which those actions appear [33]. The distance is calculated as $|A - B| + |B - A|$ for two action sequences A and B, or the number of actions that appear in one action sequence but not the other (and vice versa) without regard for action order. Lower *plan difference* indicates higher *action similarity*.

Our second *action similarity* measure considers the order in which actions appear, and is the *Levenshtein distance* between the two action sequences. This distance calculates the number of edits that must be made to an action sequence A in order to turn it into action sequence B, including insertion, deletion and replacement edits. Lower *Levenshtein distance* indicates higher *action similarity*.

For our next measure, *final state similarity*, we calculate the similarity between the final state of the world produced by LLM-generated action sequences and their human-created counterparts. We first translate each action sequence to the *Interaction Specification Language (ISL)* [34], which requires a planning problem definition. We based our problem definition on the *Task Traces* dataset. We then calculate an actionable plan for each sequence using Porfirio et al. (2024) [3]. Actions deemed impossible for the robot to perform, such as **grab**(car), are ignored.

Then, to perform the similarity calculation, let I be the initial state of the world, F_h be the human-produced final state, and F_l be the LLM-produced final state. The similarity calculation is the size of the difference between the LLM-produced and human-produced states normalized by the size of the difference between the human-produced state and the initial state. We then subtract this value by 1 to create an inverse relationship, where higher similarity values are better: $1 - |F_h - F_l| \div |F_h - I|$.

For our final measure, *length discrepancy*, we consider the difference in length between LLM-produced action sequences and their human-created counterparts. To calculate this measure, we count the actions in each action sequence and take the absolute value of the difference between its length and the length of its human-created counterpart. Note that our measures for *action similarity* implicitly account for length differences. However, explicitly calculating the difference in lengths helps determine how much *action similarity* can be accounted for by varying plan lengths.

D. Evaluation Procedure

First, we fed the NL summaries created in §IV-B to the CAS pipeline. Next, we compared each output from the CAS pipeline to its corresponding human-generated action

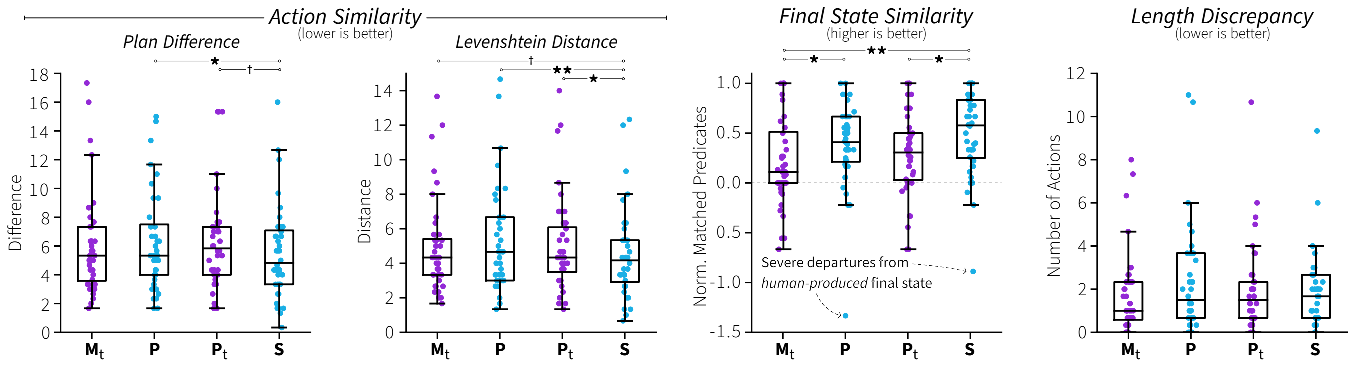

Fig. 4. Results for *Action Similarity* (left), *Final State Similarity* (center), and *Length Discrepancy* (right). †, *, and ** denote $p < 0.1$, $p < 0.05$, and $p < 0.01$, respectively.

sequence using the measures discussed in §IV-C. Recall that for each action sequence, there are three NL summaries—two originating from the authors and one originating from an LLM. For each of our measures, we averaged the output from each of the three summaries to produce a single value.

V. RESULTS

a) Action Similarity: A Friedman test detected a marginal effect of LLM model on *plan difference*, (Figure 4, left), $\chi^2(3) = 6.75$, $p = 0.080$. Post-hoc Wilcoxon tests with the Bonferroni correction[1] indicate a significant difference between $\mathbf{P}$ and $\mathbf{S}$ ($p = 0.024$) and a marginal difference between $\mathbf{P}_t$ and $\mathbf{S}$ ($p= 0.083$). Another Friedman test detected a significant effect of LLM model on *Levenshtein distance* (Figure 4, center left), $\chi^2(3) = 7.93$, $p = 0.048$, with pairwise differences between $\mathbf{P}$ and $\mathbf{S}$ ($p < 0.01$) and $\mathbf{P}_t$ and $\mathbf{S}$ ($p = 0.036$), and a marginal difference between $\mathbf{M}_t$ and $\mathbf{S}$ ($p = 0.071$). ***Takeaway:* The largest pretrained model (S) tends to produce better-matching action sequences than the smaller models.**

b) Final State Similarity: A Friedman test detected a significant effect of LLM model on *Final State Similarity* (Figure 4, center right), $\chi^2(3) = 21.8$, $p < 0.001$. Post hoc comparisons with the Bonferroni correction indicate significance between $\mathbf{M}_t$ and $\mathbf{P}$ ($p = 0.011$), $\mathbf{M}_t$ and $\mathbf{S}$ ($p < 0.01$), and $\mathbf{P}_t$ and $\mathbf{S}$ ($p = 0.020$). One-sample Wilcoxon tests indicate that the action sequences resulting from all four models produce final states that differ significantly from 0, indicating some progression from the initial state of the world ($p < 0.01$ for $\mathbf{M}_t$ and $p < 0.001$ for $\mathbf{P}$, $\mathbf{P}_t$, and $\mathbf{S}$). ***Takeaway:* All models produce action sequences that are useful, advancing the robot towards ground-truth final state. Although fine-tuning helps Mistral 7B ($\mathbf{M}_t$) produce useful action sequences compared to the incoherent output of its pretrained baseline (M), fine-tuning does not have a significant effect on Phi-4 (P vs. $\mathbf{P}_t$). Larger models tend to outperform smaller models.**

c) Length Discrepancy: We did not detect any significant effect of LLM model on *length discrepancy* (Figure 4, right). Post-hoc tests revealed no significance in any pairwise comparison. ***Takeaway:* Our models produce action sequences that consistently differ in length.**

[1]We applied the Bonferroni correction by multiplying the p-value of each pairwise comparison by the total number of comparisons, which is six.

VI. CONCLUSION

In this paper, we introduce a novel pipeline for translating natural language commands to human-like action sequences. Our aim is to produce human-like action sequences given only natural language as input, in which *human-likeness* of an LLM-produced action sequence is defined in terms of (1) similarity to its hand-crafted counterpart; (2) the degree to which the LLM-produced action sequence achieves the final state of the world achieved by its hand-crafted counterpart; and (3) the discrepancy in action sequence length to its hand-crafted counterpart. In what follows, we summarize our findings and produce concrete implications for applying the CAS pipeline in robot EUP.

A. Implications for Human-Like Action Sequences

First, as evidenced by our action-sequence similarity and final state similarity results, larger LLMs appear better than smaller ones at producing human-like action sequences. That being said, smaller models do remarkably well. $\mathbf{M}_t$ and $\mathbf{P}_t$, both fine-tuned small models, show no significant difference in length to their largest counterpart, $\mathbf{S}$, and $\mathbf{M}_t$ does not show a significant difference ($p < 0.05$) to $\mathbf{S}$ in action similarity. Furthermore, all models, both small and large, produce action sequences that significantly advance the robot towards its intended final state. ***Implication:*** *The CAS pipeline should use larger models when available and practical. If using larger models is infeasible, such as when tasking robots in the field, smaller models may produce similar performance to the large ones across some metrics.*

Second, in all of our measures, fine-tuned models exhibit no significant difference to their pretrained counterparts. However, the fine-tuned version of Mistral 7B, $\mathbf{M}_t$, was able to produce coherent, useful output, even when its pretrained counterpart ($\mathbf{M}$) could not. ***Implication:*** *Fine-tuning may or may not be necessary, depending on the model being used.*

Lastly, we note that the discrepancy of lengths between human-produced and LLM-produced action sequences is largely consistent across all four models, with no significant difference between models. ***Implication:*** *Length discrepancy is a weak differentiator of human-likeness for the models included in our study, though future work is needed to see if it is a better differentiator for other models.*

B. Limitations and Future Work

Our future work is informed by the current limitations of the CAS pipeline. First, although we have shown how different LLMs cause varying performance in the CAS pipeline, we have not yet determined what level of human-likeness is sufficient for actual human users. Future work must therefore evaluate the CAS pipeline with real human users in order to determine acceptable levels of action similarity, final state similarity, and length discrepancy.

Second, we have not yet embedded the CAS pipeline into an actual EUP pipeline. Future work must therefore connect the CAS pipeline with an actual EUP tool, such as *Polaris* [3], in order to fully realize the vision of our system. We must also test the effectiveness of the CAS pipeline within a full-fledged EUP workflow via additional user studies.

Third, the CAS pipeline itself still has room for improvement. Although we tested our pipeline with LLMs that have had a subset of their weights updated via QLoRA, we have yet to test it with LLMs that have had all of their weights updated via full fine tuning. Additionally, we have yet to determine whether fine-tuning on different datasets (*e.g.,* ALFRED [31]) affects model performance. Future work must thereby investigate how different datasets and fine-tuning paradigms affect the performance of the CAS pipeline.

Lastly, while investigated in the context of end-user programming, our pipeline may be useful to a variety of other applications, including scheduling assistance. Future work must investigate the CAS pipeline in these other contexts.

REFERENCES

[1] J. Huang and M. Cakmak, "Code3: A system for end-to-end programming of mobile manipulator robots for novices and experts," in *12th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2017.

[2] G. Huang, P. S. Rao, M.-H. Wu, X. Qian, S. Y. Nof, K. Ramani, and A. J. Quinn, "Vipo: Spatial-visual programming with functions for robot-iot workflows," in *Proc. CHI Conf. on Human Factors in Comput. Syst.*, 2020.

[3] D. Porfirio, M. Roberts, and L. M. Hiatt, "Goal-oriented end-user programming of robots," in *19th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2024.

[4] S. Rosenthal and L. M. Hiatt, "Human-centered decision support for agenda scheduling," in *Proc. 19th Int. Conf. on Autonomous Agents and MultiAgent Syst.*, 2020.

[5] P. J. Modi, M. Veloso, S. F. Smith, and J. Oh, "CMRadar: A personal assistant agent for calendar management," in *Int. Bi-Conf. Workshop on Agent-Oriented Information Syst.* Springer, 2004.

[6] J. A. Auld, *Agent-based dynamic activity planning and travel scheduling model: Data collection and model development.* University of Illinois at Chicago, 2011.

[7] T. Miller, P. Howe, and L. Sonenberg, "Explainable AI: Beware of inmates running the asylum or: How I learnt to stop worrying and love the social and behavioural sciences," in *IJCAI 2017 Workshop on Explainable Artificial Intelligence (XAI)*, 2017.

[8] T. Chakraborti, S. Sreedharan, S. Grover, and S. Kambhampati, "Plan explanations as model reconciliation – an empirical study," in *14th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2019.

[9] C. H. Song, B. M. Sadler, J. Wu, W.-L. Chao, C. Washington, and Y. Su, "LLM-Planner: Few-shot grounded planning for embodied agents with large language models," in *Proc. IEEE/CVF Int. Conf. on Computer Vision*, 2023.

[10] T. Silver, S. Dan, K. Srinivas, J. B. Tenenbaum, L. Kaelbling, and M. Katz, "Generalized planning in pddl domains with pretrained large language models," *Proc. AAAI Conf. on Artif. Intell.*, 2024.

[11] S. Kambhampati, K. Valmeekam, L. Guan, M. Verma, K. Stechly, S. Bhambri, L. P. Saldyt, and A B Murthy, "Position: LLMs can't plan, but can help planning in LLM-modulo frameworks," in *Proc. 41st Int. Conf. on Machine Learning*, 2024.

[12] M. J.-Y. Chung, J. Huang, L. Takayama, T. Lau, and M. Cakmak, "Iterative design of a system for programming socially interactive service robots," in *Social Robotics*, 2016.

[13] D. Glas, S. Satake, T. Kanda, and N. Hagita, "An interaction design framework for social robots," in *Proc. Robot.: Sci. and Syst.*, 2011.

[14] E. Pot, J. Monceaux, R. Gelin, and B. Maisonnier, "Choregraphe: a graphical tool for humanoid robot programming," in *18th IEEE Int. Symp. on Robot and Human Interactive Commun.*, 2009.

[15] N. Leonardi, M. Manca, F. Paternò, and C. Santoro, "Trigger-action programming for personalising humanoid robot behaviour," in *Proc. 2019 CHI Conf. on Human Factors in Comput. Syst.*, 2019.

[16] Y. Cao, Z. Xu, F. Li, W. Zhong, K. Huo, and K. Ramani, "V.Ra: An in-situ visual authoring system for robot-IoT task planning with augmented reality," in *Proc. Designing Interactive Syst. Conf.*, 2019.

[17] E. Senft, M. Hagenow, R. Radwin, M. Zinn, M. Gleicher, and B. Mutlu, "Situated live programming for human-robot collaboration," in *ACM Symp. User Interface Softw. Technol.*, 2021.

[18] B. Ikeda, M. Gramopadhye, L. Nekervis, and D. Szafir, "Marcer: Multimodal augmented reality for composing and executing robot tasks," in *20th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2025.

[19] Y. Ge, Y. Dai, R. Shan, K. Li, Y. Hu, and X. Sun, "Cocobo: Exploring large language models as the engine for end-user robot programming," in *IEEE Symp. Vis. Lang. Human-Centric Comput.*, 2024.

[20] V. Schlegel, B. Lang, S. Handschuh, and A. Freitas, "Vajra: step-by-step programming with natural language," in *Proc. 24th Int. Conf. on Intelligent User Interfaces*, 2019.

[21] J. F. Gorostiza and M. A. Salichs, "End-user programming of a social robot by dialog," *Robot. Auton. Syst.*, vol. 59, 2011.

[22] S. Beschi, D. Fogli, and F. Tampalini, "Capirci: A multi-modal system for collaborative robot programming," in *End-User Develop.*, 2019.

[23] N. G. Buchina, P. Sterkenburg, T. Lourens, and E. I. Barakova, "Natural language interface for programming sensory-enabled scenarios for human-robot interaction," in *28th IEEE International Conf. on Robot and Human Interactive Commun.*, 2019.

[24] U. B. Karli, J.-T. Chen, V. N. Antony, and C.-M. Huang, "Alchemist: LLM-aided end-user development of robot applications," in *19th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2024.

[25] K. Mahadevan, B. Lewis, J. Li, B. Mutlu, A. Tang, and T. Grossman, "Imageinthat: Manipulating images to convey user instructions to robots," in *20th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2025.

[26] K. Valmeekam, K. Stechly, A. Gundawar, and S. Kambhampati, "Planning in strawberry fields: Evaluating and improving the planning and scheduling capabilities of lrm o1," *arXiv:2410.02162*, 2024.

[27] M. Ahn, A. Brohan, N. Brown, Y. Chebotar, O. Cortes, B. David, C. Finn, C. Fu, K. Gopalakrishnan, K. Hausman, *et al.*, "Do as i can, not as i say: Grounding language in robotic affordances," *arXiv:2204.01691*, 2022.

[28] M. M. de Graaf and B. F. Malle, "How people explain action (and autonomous intelligent systems should too)." in *AAAI Fall Symp. on Artificial Intelligence for Human-Robot Interaction*, 2017.

[29] X. Puig *et al.*, "Virtualhome: Simulating household activities via programs," in *2018 IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2018.

[30] D. Porfirio, A. Sauppé, M. Cakmak, A. Albarghouthi, and B. Mutlu, "Crowdsourcing task traces for service robotics," in *ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2023.

[31] M. Shridhar *et al.*, "Alfred: A benchmark for interpreting grounded instructions for everyday tasks," in *IEEE/CVF Conf. on Computer Vision and Pattern Recognition*, 2020.

[32] T. Dettmers, A. Pagnoni, A. Holtzman, and L. Zettlemoyer, "Qlora: Efficient finetuning of quantized llms," in *Advances in Neural Information Processing Syst.*, 2023.

[33] M. Fox, A. Gerevini, D. Long, and I. Serina, "Plan stability: Replanning versus plan repair," in *Proc. Int. Conf. on Automated Planning and Scheduling*, 2006.

[34] D. Porfirio, M. Roberts, and L. M. Hiatt, "An interaction specification language for robot application development," in *20th ACM/IEEE Int. Conf. on Human-Robot Interact.*, 2025.