Tailoring with Targeted Precision: Edit-Based Agents for Open-Domain Procedure Customization

Yash Kumar Lal[13*], **Li Zhang**[23*], **Faeze Brahman**[3],
Bodhisattwa Prasad Majumder[3], **Peter Clark**[3], **Niket Tandon**[3]

[1] Stony Brook University, [2] University of Pennsylvania
[3] Allen Institute for Artificial Intelligence
[1]ylal@cs.stonybrook.edu, [2]zharry@upenn.edu,
[3]{faezeb,bodhisattwam,peterc,nikett}@allenai.org

Abstract

How-to procedures, such as how to plant a garden, are now used by millions of users, but sometimes need customizing to meet a user's specific needs, e.g., planting a garden without pesticides. Our goal is to measure and improve an LLM's ability to perform such customization. Our approach is to test several simple multi-LLM-agent architectures for customization, as well as an end-to-end LLM, using a new evaluation set, called CUSTOMPLANS, of over 200 WikiHow procedures each with a customization need. We find that a simple architecture with two LLM agents used sequentially performs best, one that edits a generic how-to procedure and one that verifies its executability, significantly outperforming (10.5% absolute) an end-to-end prompted LLM. This suggests that LLMs can be configured reasonably effectively for procedure customization. This also suggests that multi-agent editing architectures may be worth exploring further for other customization applications (e.g. coding, creative writing) in the future.

1 Introduction

AI is headed towards a future where human-machine interactions are seamlessly integrated to enrich our daily routines, offering personalized and tailored experiences (Chen et al., 2023). For instance, a software engineer's daily routine would involve a co-pilot that customizes the same underlying logic differently for two engineers (even though we observe the structure might broadly remain the same). Another application is smart assistant and planners that can customize how-to procedures, a popular query making up a large fraction of search engine queries (De Rijke et al., 2005; Zhang, 2022), based on a user's specifications (though the sequence of steps broadly stay the same). For example, a user looking for "How to plant a garden", may have space restrictions in their apartment, or not want to use pesticides. Despite the need for customization, it is challenging to author new customized how-to procedures for every users' nuanced needs.

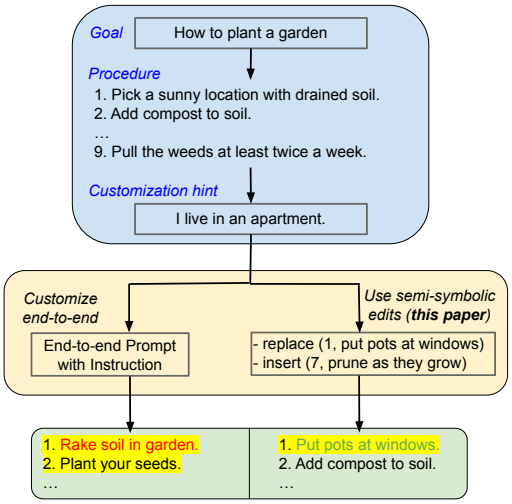

Figure 1: Procedures, e.g., how to plant a garden, need customization, e.g., this user lives in an apartment. Given a goal, uncustomized procedure and customization hint, employing semi-symbolic edits (right) produces more desirable outputs than an E2E LLM (left; which suggests a garden patch inside an apartment).

Automatically customizing procedures requires the interpretation of the nuanced user needs expressed in natural language (Du et al., 2006). These customization requests or hints can take up various forms e.g., constraints ("I live in an apartment"), personal preferences ("I prefer organic farming"), or execution method ("plant a hydroponic garden"). These implicitly encode multiple requirements and their interpretation is subjective — for example, living in an apartment can entail a lack of space, limited space, convincing roommates, and more. Contemporary approaches to customization focus on constraints in specific domains (Yuan et al., 2023; Welch et al., 2022). LLMs could be considered

*Work done as an intern at AI2 Aristo

a strong baseline for faithfully customizing procedures to different users' needs (Acher, 2024), and we did find that the generated customized procedures are fluent and coherent. However, in our experiments §2.2, we found that ~60% of the procedures generated by contained errors (missing steps, extra steps, wrong steps, underspecified steps), making the output inadequately customized or inexecutable, as shown in Figure 1. We observe that even though uncustomized and customized procedures share some inherent structure, end-to-end systems disregard that and produce entirely new structures which introduces unwanted changes.

Rather than using LLMs as end-to-end customizers, we distenagle the task into modifying a procedure based on a customization requirement, and verifying for executability. We propose a multi-agent framework comprising two LLM-based agents, `Modify` agent and `Verify` agent for customization and execution verification respectively. We create a new evaluation set called CUSTOMPLANS of over 200 WikiHow procedures each with a customization need, and show that these agents are most effective when operating based on semi-symbolic edits rather than free-form natural language edits. We also discuss the generalizability of our framework to support multiple `Verify` agents. Through extensive experiments with CUSTOMPLANS, we find that our multi-agent framework leads to 10.5% more customized and executable procedures over just using LLMs as end-to-end customizers.

In summary, our contributions are:

- Using a new evaluation set CUSTOMPLANS, we show that LLMs are yet unsuited to customize how-to procedures in an end-to-end fashion.
- We propose a multi-agent framework comprising `Modify` and `Verify` agents, and show that semi-symbolic edits is the most effective means of communicating results. This framework achieves an improvement of 10.5% over using LLMs as end-to-end customizers.
- We show the generalizability of our framework to support multiple agent configurations, and the limits of current methods when employed for broader applications.

2 Task Setup

In this section, we define a problem formulation and evaluation scheme for procedure customization and showcase the shortcomings of using LLMs as end-to-end customizers. LLMs are capable of generating fluent texts, including procedures (Sakaguchi et al., 2021; Lyu et al., 2021). However, because LLMs generate texts in an autoregressive manner based on the previous context, they cannot edit those texts like humans would. This means that they need to rely on re-generation, which leads to unsatisfactory performance on our proposed task.

2.1 Task Formulation

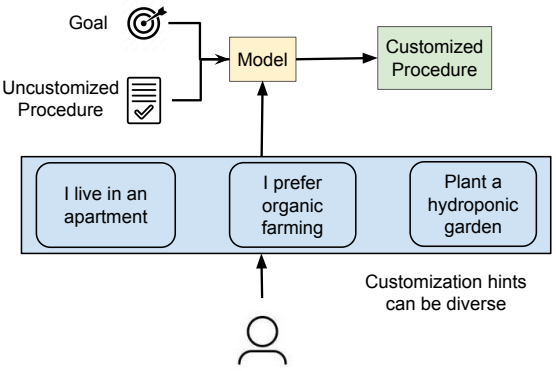

Figure 2: Given a goal G, uncustomized procedure P and a user's customization hint H, we generate P', a customized procedure that accounts for H.

The task, illustrated in Figure 2, involves taking as input the goal G of the procedure P as well as a user customization hint H. The output should be a procedure to fulfill the goal while also taking into account the user scenario described in the hint P'.

2.2 How good are vanilla LLMs on this task?

We conduct an initial study using vanilla LLMs (see Figure 1 (left)) on 40 procedures [1]. Given the goal G, uncustomized procedure P and customization hint H, the LLM (`gpt-3.5-turbo`) is expected to make changes as per the customization hint. While this approach is simple, the output is often undesirable (contains extra steps and yet results inadequate or erroneous customization). Table 1 presents two examples of these errors. We find that 32.5% of these errors are due to addition of extra steps which do not apply to G, and 15% of the errors are due to inadequate or erroneous customization. Given the shortcomings of end-to-end methods, we need a more structured approach to edit procedures for customization.

[1]These procedures are taken from the new evaluation set CUSTOMPLANS that we include in this paper (see §4.1).

Error Type	Goal G	Uncustomized Procedure P	Customization Hint H	Customized Procedure P'
Wrong Order	Bake Bread	1. Dissolve yeast in water. 2. Mix the ingredients. 3. Knead the dough. 4. Put dough into greased pan ...	I would like to make something sweet and I also like nuts.	... 3. Place into a baking pan. 4. Add chopped nuts. 5. Knead the dough. ...
Extra Steps	Customize Shoes	1. doodle on shoes. 2. add embellishments. 3. change out laces for ribbon. 4. glue rhinestones on straps. 5. wrap ribbon around straps.	I am a ballet dancer and would like to improve the comfort of my shoes.	1. Identify areas of discomfort. 2. Purchase gel pads. 3. Insert in areas of discomfort. ... 11. Break in shoes gradually.

Table 1: Example of two major types of errors (shown in red) produced by vanilla LLMs. In the first example (Wrong Order), placing the dough in a baking pan in Step 3 implies that the dough is ready to be baked, so Step 5 (knead the dough) is out of order. In the second example (Extra Steps), there are six extra steps being added (most of them are unnecessary) because the resulting procedure contains 11 steps while the uncustomized one only has five.

3 Models

In this section, we describe our multi-agent approach for customizing procedures for users' needs using semi-symbolic edits. We disentangle the task of generating customized procedures into two aspects, customizability and executability.

3.1 Agents for Procedure Customization

Recently, model-based agents (Xi et al., 2023) have been used to perform different types of reasoning in service of achieving a larger goal (Yoran et al., 2023). We use instances of LLMs to modify for customization and verify for execution and use them in conjunction. **`Modify agent`** suggests edits that address a user's customization needs, while **`Verify agent`** suggests edits to maintain the executability of procedures. Next, we describe how these agents can interact with each other to best generate customized procedures.

Each agent produces a bag of semi-structured edits E that indicate the operations to be performed, the step in the original procedure P to anchor the edit, and the updated text for the step. We only allow for two types of edits, insert and replace:

- *insert(2, XX)* - a new step with text XX should be added after step 2 of the input procedure.
- *replace(3, YY)* - the text of step 3 of the input procedure should be replaced by YY.

Note that the replace operation can also perform step deletion by specifying an empty string as its second argument. The semi-symbolic nature of the edits allows algorithmic application of the edits and decreases model hallucinations compared to end-to-end approaches which make unstructured edits. We then apply these edits deterministically on P to obtain P' i.e., we find the step number in P that corresponds to a suggested edit and insert/replace it with the edited text. Having a separate module to apply edits allows us to study the reasoning a model performs when trying to address user requirements.

3.2 How Do The Agents Interact?

We experiment with three ways of interaction for the defined agents: UNIFIED, SEQUENTIAL, and PARALLEL, demonstrated in Figure 3, Figure 4 and Figure 5 respectively.

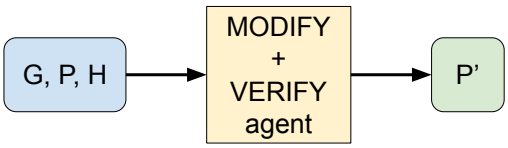

Figure 3: `Modify` agent and `Verify` agent in UNIFIED setting. Here, one LLM agent is asked to suggest edits for both customizability as well as executability.

UNIFIED - We first define a single agent that is prompted to suggest edits E to P that both customizes it and ensures its executability. Mechanically applying these edits results in the customized plan P'. This agent is required to understand how to perform both customization towards a hint H as well as execution to achieve the goal G. This is somewhat similar to the end-to-end method which is also required to understand both aspects of customization. However, rather than generating the customized procedure P' directly, it generates *edits* to P that result in P' when applied. This setting is shown in Figure 3.

SEQUENTIAL - In this setting, we first obtain a set of edits E_c from `Modify` agent , and apply E_c

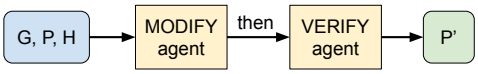

Figure 4: `Modify` agent and `Verify` agent in SEQUENTIAL setting. Here, `Modify` agent first generates edits to customize P. Then, `Verify` agent makes changes such that the edited procedure is executable, producing P'.

to obtain P_c. E_c represents the changes that need to be made in order to suit a user's customization needs. We obtain P_c, a customized procedure, by deterministically applying E_c in P. P_c denotes a customized procedure. Then, to ensure that this procedure can be executed, `Verify` agent takes P_c as input and suggests another set of edits E_e. Finally, we deterministically apply E_e on P_c to obtain the output customized procedure P'. Here, the agents are used in a sequential order, first suggesting edits to meet customization requirements and next to address any execution-related issues that may arise from those edits. Figure 4 shows the interaction of the agents in the SEQUENTIAL setting.

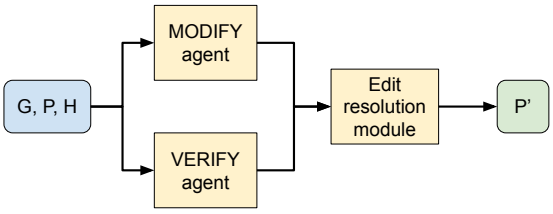

Figure 5: `Modify` agent and `Verify` agent in PARALLEL setting. Here, `Modify` agent suggests edits to customize the procedure P, while `Verify` agent suggests adding often-missing details so that P can be followed. Finally, the edit resolution module takes both sets of edits into account and produces a final bag of edits to be applied to P.

PARALLEL - In this setting, shown in Figure 5, both `Modify` agent and `Verify` agent propose a bag of edits for their respective aspect, E_c and E_e, on the uncustomized procedure P. Since changes for customization and execution are different, conflicts arise between those bags of edits. It is non-trivial to understand how to select the correct edit, since both conflicting edits are important to generate P' but serve different objectives. To address this, we use an edit resolution module, an LLM that is prompted to take as input two bags of edits, E_c and E_e, and produce a merged set of edits E. Finally, applying E on the uncustomized procedure P results in P'. Functionally, this module is intended to resolve conflicts, merge possible edits and remove redundant edits. It is also required to remove any edits which cannot be applied to the procedure deterministically. This is inspired by the meta-reasoner in Yoran et al. (2023).

4 Experiments

To tailor procedures according to customization hints, one needs to make implicit inferences out of the hints, identify the steps that require changes, and finally consistently apply changes to the different steps. Through our experiments, we aim to understand how well models can modify generic procedures to incorporate aspects captured in customization hints.

4.1 Our CUSTOMPLANS evaluation set

We use WikiHow as the source of diverse goals and corresponding procedures to accomplish them. Given a broad goal, users are required to write their situation in which they want to accomplish the goal, which acts as their customization hint. We collect 206 goals over 9 domains, their corresponding WikiHow procedures and customization hints collected from humans to build CUSTOMPLANS. Each record is associated with constraint, expertise and critical type (which subtype is more important to perform customization), shown in Figure 6. More details can be found in Appendix A.

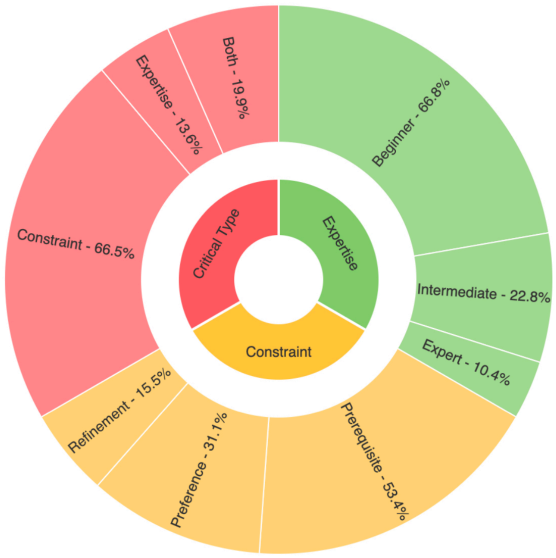

Figure 6: Different types of needs expressed in customization hints in CUSTOMPLANS.

Method	CUSTOMIZED	EXECUTABLE	FULLYCORRECT
E2E Customize	55.05%	48.45%	41.46%
SEQUENTIAL	**60.68%**	**72.33%**	**51.94%**
UNIFIED	54.85%	71.36%	47.09%
PARALLEL	53.88%	70.87%	45.63%
Reverse SEQUENTIAL	42.23%	63.59%	34.47%

Table 2: Customizability, executability and fully correct (strictest measure) of procedures generated by different approaches as judged by majority of human evaluators. We note that the SEQUENTIAL setting performs the best across all criteria. Note that all approaches built on edit-based agents lead to more executable procedures.

4.2 Evaluation

For open-ended text generation tasks (without gold references) like procedure customization, the absence of an automatic evaluation that correlates well with human judgments is a major challenge (Chen et al., 2019; Ma et al., 2019; Caglayan et al., 2020; Howcroft et al., 2020; Lal et al., 2021). As a result, we use human evaluation directly, rather than rely on proxies such as GPT-4.

We conduct a human evaluation with a standardized interface to compare different models. To this end, we pose questions about customizability, whether a procedure can be performed as is to accomplish the given goal (EXECUTABLE), and executability, whether it satisfies all the requirements in a presented customization hint (CUSTOMIZED). This serves as the human evaluation interface for our task. The study is illustrated in Figure 7. The task instructions and annotator details can be found in Appendix B.

For each goal, customization hint, and a model's generated procedure, we ask 3 distinct annotators to provide judgments about customizability and executability of that procedure. A procedure can be complete in an aspect (customizability or executability), missing steps, have extra steps, have underspecified steps, or have incorrect steps. An annotator can point out multiple errors about an aspect in the presented plan (negative), or judge the plan to be correct (positive) in that aspect. We take the majority vote of annotator judgments for the CUSTOMIZED and EXECUTABLE criteria. If a plan is judged to be both CUSTOMIZED and EXECUTABLE by a majority of annotators, we consider it to be FULLYCORRECT.

4.3 Results

We report majority statistics for customizability, executability, and correctness of different models in Table 2. Simply using LLMs in an end-to-end

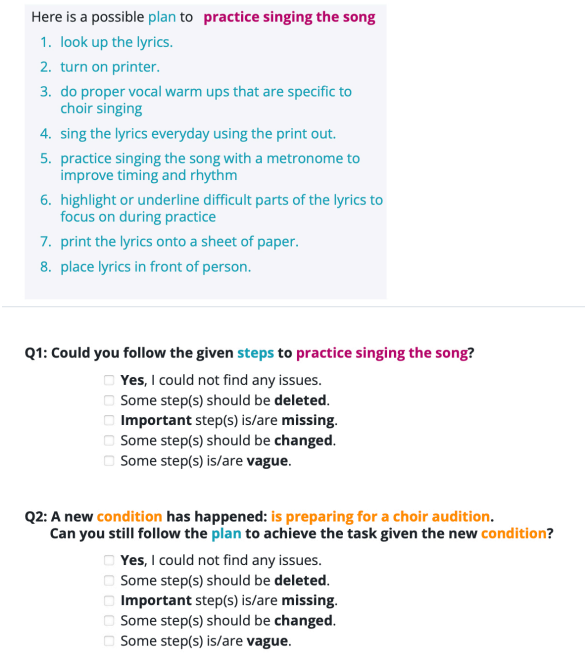

Figure 7: MTurk interface presented to crowdsource workers to judge model generated procedures.

fashion is not good enough to generate customized procedures, only generating 41.46% fully correct procedures. We make the following observations.

Customize first, fix later. Using `Modify` agent and `Verify` agent in SEQUENTIAL order is the best at producing customized procedures P', generating fully correct customized procedures 51.94% of the time. Customizing first allows for making changes to suit a user's customization requirements, before editing the modified procedure to make sure it is executable.

Modifying and verifying together is hard. Combining both agents into one to obtain a bag of edits in the UNIFIED setting requires suggesting edits that serve the purpose of both customization as well as execution. This is an inherently harder task and, as expected, does not perform as well as the

SEQUENTIAL setting.

Edit-based customization is interpretable. Using LLMs as end-to-end customizer is directly comparable to the UNIFIED setting. The former uses natural language while the latter relies on semi-symbolic edits. By construction, not only does the UNIFIED setting perform better, it is also more interpretable since the end-to-end approach sometimes tends to completely change the procedure.

Resolving conflicts is difficult. We find that the PARALLEL setting is the worst, even though its performance on CUSTOMIZED and EXECUTABLE are similar to other methods. However, the intersection of both (FULLYCORRECT) is significantly lower than others. We hypothesize that this problems arises because the edit resolution module is unable to correctly merge bags of edits from `Modify` agent and `Verify` agent . This agent can be improved by providing a more complete outlook of how to resolve conflicts and merge relevant edits.

Despite operating on gold procedures, `Verify` agent removes redundant steps or adds critical details to underspecified procedures (often the case with WikiHow). For example, the instructions to make microwave banana bread start with dry ingredients being mixed in one bowl, while wet ingredients are mixed in another. However, it does not specify what the dry or wet ingredients are. The `Verify` agent expands on these details so that the procedure can be followed. More generally, this framing allows fixing issues in the uncustomized procedure if it isn't from a gold source.

5 Analysis

We analyze different aspects of the SEQUENTIAL setting and its generated plans.

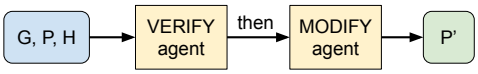

Figure 8: Reverse order of interaction of `Verify` agent and `Modify` agent in SEQUENTIAL setting. First, `Verify` agent adds missing details to improve executability, before `Modify` agent proposes changes to suit a user's customization needs.

Ordering of agents matter. In the SEQUENTIAL setting, we flip the order of the `Modify` agent and `Verify` agent , as illustrated in Figure 8. We find that this approach only generates usable customized procedures 34.47% of the time. While unintuitive, this setting addresses the lack of detail in some WikiHow procedures while also correctly making direct changes for customization.

Edits help executability. When comparing the end-to-end approach with any of the edit-based agents, we find that using the agentive framework is better for executability. Table 2 shows that all of our approaches produce executable procedures >70% while using LLMs naively generates <50% procedures that can be followed.

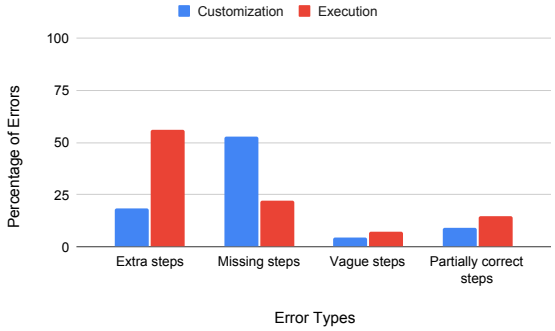

Figure 9: Error distribution in customized plans produced by SEQUENTIAL setting.

Procedures cannot always be executed. We note that execution accuracy for all models is similar. Figure 9 shows that each proposed interaction of the agents suffers the most from generating unnecessary steps, which often hinder achieving the goal of the procedure G.

Procedures are not sufficiently customized. All methods suffer from the problem of missing steps, indicating that none of them adequately address the requirements stated implicitly or explicitly in the customization hint for the procedure. We use the SEQUENTIAL setting as an illustration in Figure 9.

5.1 Analyzing subtypes of customization needs

CUSTOMPLANS is annotated with different metadata related to types of expertise and constraints. We use that to perform fine-grained analysis of the generated procedures.

It is the hardest to satisfy prerequisites. Figure 10 shows the performance of using the SEQUENTIAL setting to address different types of constraints expressed in the customization hint. Generating customized procedures that address prerequisites is the hardest. Performance on procedures incorporating preferences is similar. Upon closer

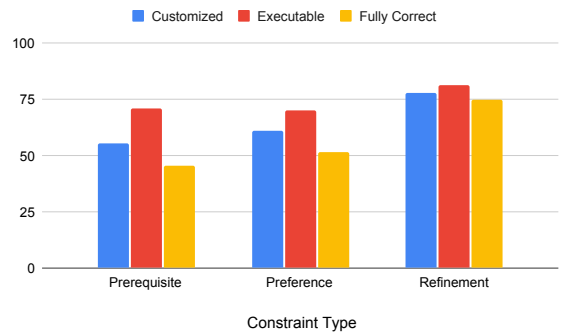

Figure 10: Performance in the SEQUENTIAL setting for subtypes of constraints in customization hints.

inspection, we see that while the generated procedures might be executable, they do not adhere to the hard constraints set by prerequisities such as an allergy to gluten.

Figure 11: Performance in the SEQUENTIAL setting for subtypes of expertise in customization hints.

Complexity of customization matters. To study the effect of depth of customization, we compare performance on procedures that require varying degrees of expertise. It is harder to generate procedures for either experts or for beginners, as presented in Figure 11. We hypothesize that domain experts rarely require detailed feedback and can work with small amounts of information to achieve their task. Conversely, beginners require careful instructions to be able to achieve a goal. It is difficult for one approach to generate both detailed as well as succinct, high-level plans.

Constraints are harder to account for than expertise. Figure 12 shows that it is the most difficult to customize according to constraints. Correctly satisfying requirements expressed in compositional hints, where it is important to consider both expertise and constraint, is also difficult even for the best agent setting.

Figure 12: Performance in the SEQUENTIAL setting when there are multiple aspects of customization expressed in a customization hint. Among these, constraints are hardest to fully satisfy in the resulting customized procedures.

5.2 Qualitative Analysis

On the subset of data points in §2, we make binary judgments about customizability and executability, akin to §4.2, after anonymizing the source of the plans to mitigate bias. The trends for FULLYCORRECT are similar to full-scale evaluation, except that the PARALLEL setting is slightly better than the UNIFIED one. Executability of procedures on this sample are higher than on the full evaluation set, but trends across methods remain the same, implying that our observations apply to the full CUSTOMPLANS evaluation set. We then perform qualitative analysis on plans generated by each approach to understand their characteristics.

Edit-based agents are conservative. Edit-based agents tend to suggest less changes than their end-to-end counterparts. We hypothesize that these agents only generate edits that have higher confidence, and more likely to maintain the coherency of the plan. On the flip side, it makes it harder to use these agents when the plan needs to be completely changed in order to fully satisfy the requisite customization needs.

End-to-end methods have greater creativity. While completely rewriting procedures to suit customization needs is not ideal, we observe that it is important to do so when the fundamental way of achieving the goal needs to be altered. We quantify procedures that require changes in >4 steps to fall into this category. For such procedures, we observe that using LLMs as end-to-end customizers is better since they produce more creative changes while

edit-based agents are more conservative.

5.3 Discussion

While our task formulation works on uncustomized gold procedures, our edit-based agents can also be used to fix incorrectly customized procedures, since it treats customization as a modification problem. Instead of using a gold plan as input, our proposed framework supports starting with any related procedure for customization.

More complex tasks would require collaboration between more agents, each dedicated to one aspect of the task. Despite the effectiveness of the SEQUENTIAL setting for procedure customization, it suffers from the problem of agent ordering. With a larger number of agents, it would be non-trivial to determine the correct order of using agents.

The PARALLEL setting is flexible and allows the opportunity to integrate edits from multiple agents, similar to prior work (Li et al., 2023; Yoran et al., 2023; Wang et al., 2023). For instance, we can integrate different agents to verify the executability of procedures. An agent that use entity state changes can be used as a proxy for checking the coherence of a procedure. Similarly, an agent that enforces the cause and effect relationship between subsequent steps can also be used as a way to verify executability of procedures. This setting can also be used to integrate different agents for broader applications such as collaborative tool use. Using edits from such diverse agents is only practically possible when using them in a parallel setting.

6 Related Work

Over the past few years, as information about users has been readily available, user-facing technology has become increasingly customized. Customization has been widely studied in the context of such technology, like search engines (Rashid et al., 2002), chatbots (Majumder et al., 2020) and game content (Shaker et al., 2010). There is some work on customized procedural content generation (Togelius et al., 2011; Yannakakis and Togelius, 2011). However, they focus on video game content rather than everyday procedures related to pragmatic goals. Previous research has shown the value of customization in various settings. Kapusta et al. (2019) present a method to customize collaborative plans for robot-assisted dressing. Building customized care plans is crucial for patients with severe illnesses (Lin et al., 2017; Anbari et al., 2020).

Procedural text understanding addresses the task of tracking entity states throughout the text (Bosselut et al., 2018; Henaff et al., 2017). Generating such texts (Aouladomar and Saint-Dizier, 2005) involves different types of understanding such as temporal reasoning and entity linking. Mori et al. (2014) generated procedures from a meaning representation taking intermediate states into account. ChattyChef (Le et al., 2023) uses the conversational setting to generate cooking instructions and iteratively refine its step ordering. To study decision branching in procedures, (Hou et al., 2023) generated user scenarios and choices for various steps in a procedure and presented CHOICE-75. For the task of counterfactual planning, COPLAN (Brahman et al., 2023) collects conditions and combines them with a revised list of steps. Majumder et al. (2019) use historical user preferences to generate customized recipes. But they only focus on one domain (cooking) and on one dimension to model customization. LLMs have been shown to generate procedural text well but it is unclear how they can be customized for diverse user preferences.

Customized models are designed to capture language patters specific to individual users. King and Cook (2020) examined methods for creating customized LMs using interpolating, fine-tuning and priming. Similarly, Welch et al. (2022) present another approach for fine-tuning and interpolation to customize LMs. LLMs have also been used to study constrained planning (Yuan et al., 2023) and new interfaces for personalization (Ma et al., 2023). Role-based prompting is a recent trend. Pseudo Dialog Prompting (Han et al., 2022) is a method to build prompts for LLMs so that chatbots mimic fictional characters, while Vincent et al. (2023) release a dataset of character annotations to induce personas in LLM output. However, it is still unclear how to encapsulate more fine-grained aspects of customization.

7 Conclusion

This paper studies the capabilities of current LLMs to customize open-domain procedures. First , we show that current LLMs cannot effectively customize open-domain procedures. Next, we propose several LLM-based agent architectures, and find that sequentially using semi-symbolic edits from the `Modify` agent and the `Verify` agent provides an improvement of 10.5% in generating fully correct procedures over naive prompting. Even though

we show that using edits helps executability, we find that generated procedures are often not sufficiently customized, and there is clear room for improvement. Finally, we discuss the generalizability of our framework for other diverse applications such as coding and creative writing that require customization.

Limitations

In this work, we focus on using LLMs in a zero-shot setting. It has been shown that model performance improves by providing in-context examples, performing chain-of-thought reasoning, and incorporating notions of self-consistency. However, we leave these explorations for future work.

We acknowledge that there can be multiple ways to interpret a customization hint and only the user providing that hint can truly know their needs from the procedure. This also means that only that person is the right person to evaluate a customized procedure. Even so, due to the open-domain nature of this task, it is difficult for any one person to evaluate the various aspects of generated procedures. This can only be done by domain experts. To alleviate these problems, we use 3 annotators (master turkers) to judge model generated plans and consider their majority judgment (Lal et al., 2022), but recognize that this might not be the perfect solution.

Due to the nature of LLMs, they are unable to encapsulate personal preferences (V Ganesan et al., 2023; Dey et al., 2024). Since the procedure customization inherently involves user preferences, LLMs cannot be reliably used as automatic judges yet.

Ethics Statement

Our setting assumes that users will not provide any adversarial customization hints. However, in real-world environments, this assumption is unlikely to hold. Users can generate malicious procedures by providing such hints to LLMs.

LLMs can generate text that might be biased and insensitive to a user's socio-cultural context (Bordia and Bowman, 2019; Sharma et al., 2021; Hovy and Prabhumoye, 2021). Since customization hints can have multiple interpretations, it is possible that LLMs can misinterpret the customization needs of users and generate biased and stereotypical procedures. Thus the system will need checks and balances to ensure that the generated customized procedure is not harmful.

References

Mathieu Acher. 2024. A demonstration of end-user code customization using generative ai. In 18th International Working Conference on Variability Modelling of Software-Intensive Systems, Bern, Switzerland.

Allison Brandt Anbari, Pamela L. Ostby, and Pamela Ginex. 2020. Breast cancer–related lymphedema: Personalized plans of care to guide survivorship. Current Breast Cancer Reports, 12:237 – 243.

Farida Aouladomar and Patrick Saint-Dizier. 2005. Towards generating procedural texts: An exploration of their rhetorical and argumentative structure. In Proceedings of the Tenth European Workshop on Natural Language Generation (ENLG-05), Aberdeen, Scotland. Association for Computational Linguistics.

Shikha Bordia and Samuel R. Bowman. 2019. Identifying and reducing gender bias in word-level language models. In Proceedings of the 2019 Conference of the North American Chapter of the Association for Computational Linguistics: Student Research Workshop, pages 7–15, Minneapolis, Minnesota. Association for Computational Linguistics.

Antoine Bosselut, Omer Levy, Ari Holtzman, Corin Ennis, Dieter Fox, and Yejin Choi. 2018. Simulating action dynamics with neural process networks. ICLR.

Faeze Brahman, Chandra Bhagavatula, Valentina Pyatkin, Jena D. Hwang, Xiang Lorraine Li, Hirona J. Arai, Soumya Sanyal, Keisuke Sakaguchi, Xiang Ren, and Yejin Choi. 2023. Plasma: Making small language models better procedural knowledge models for (counterfactual) planning.

Ozan Caglayan, Pranava Madhyastha, and Lucia Specia. 2020. Curious case of language generation evaluation metrics: A cautionary tale. In Proceedings of the 28th International Conference on Computational Linguistics, pages 2322–2328, Barcelona, Spain (Online). International Committee on Computational Linguistics.

Anthony Chen, Gabriel Stanovsky, Sameer Singh, and Matt Gardner. 2019. Evaluating question answering evaluation. In Proceedings of the 2nd Workshop on Machine Reading for Question Answering, pages 119–124, Hong Kong, China. Association for Computational Linguistics.

Jin Chen, Zheng Liu, Xu Huang, Chenwang Wu, Qi Liu, Gangwei Jiang, Yuanhao Pu, Yuxuan Lei, Xiaolong Chen, Xingmei Wang, Defu Lian, and Enhong Chen. 2023. When large language models meet personalization: Perspectives of challenges and opportunities.

Maarten De Rijke et al. 2005. Question answering: What's next?

Gourab Dey, Adithya V Ganesan, Yash Kumar Lal, Manal Shah, Shreyashee Sinha, Matthew Matero, Salvatore Giorgi, Vivek Kulkarni, and H. Schwartz. 2024. SOCIALITE-LLAMA: An instruction-tuned model for social scientific tasks. In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 2: Short Papers), pages 454–468, St. Julian's, Malta. Association for Computational Linguistics.

Xuehong Du, Jianxin Jiao, and Mitchell M Tseng. 2006. Understanding customer satisfaction in product customization. The International Journal of Advanced Manufacturing Technology, 31:396–406.

Seungju Han, Beomsu Kim, Jin Yong Yoo, Seokjun Seo, Sangbum Kim, Enkhbayar Erdenee, and Buru Chang. 2022. Meet your favorite character: Open-domain chatbot mimicking fictional characters with only a few utterances. In Proceedings of the 2022 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies, pages 5114–5132, Seattle, United States. Association for Computational Linguistics.

Mikael Henaff, Jason Weston, Arthur Szlam, Antoine Bordes, and Yann LeCun. 2017. Tracking the world state with recurrent entity networks. ICLR.

Zhaoyi Joey Hou, Li Zhang, and Chris Callison-Burch. 2023. Choice-75: A dataset on decision branching in script learning. arXiv preprint arXiv:2309.11737.

Dirk Hovy and Shrimai Prabhumoye. 2021. Five sources of bias in natural language processing. Language and Linguistics Compass, 15(8):e12432.

David M. Howcroft, Anya Belz, Miruna-Adriana Clinciu, Dimitra Gkatzia, Sadid A. Hasan, Saad Mahamood, Simon Mille, Emiel van Miltenburg, Sashank Santhanam, and Verena Rieser. 2020. Twenty years of confusion in human evaluation: NLG needs evaluation sheets and standardised definitions. In Proceedings of the 13th International Conference on Natural Language Generation, pages 169–182, Dublin, Ireland. Association for Computational Linguistics.

Ariel Kapusta, Zackory M. Erickson, Henry M. Clever, Wenhao Yu, C. Karen Liu, Greg Turk, and Charles C. Kemp. 2019. Personalized collaborative plans for robot-assisted dressing via optimization and simulation. Autonomous Robots, 43:2183 – 2207.

Milton King and Paul Cook. 2020. Evaluating approaches to personalizing language models. In Proceedings of the Twelfth Language Resources and Evaluation Conference, pages 2461–2469, Marseille, France. European Language Resources Association.

Yash Kumar Lal, Nathanael Chambers, Raymond Mooney, and Niranjan Balasubramanian. 2021. TellMeWhy: A dataset for answering why-questions in narratives. In Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021, pages 596–610, Online. Association for Computational Linguistics.

Yash Kumar Lal, Niket Tandon, Tanvi Aggarwal, Horace Liu, Nathanael Chambers, Raymond Mooney, and Niranjan Balasubramanian. 2022. Using commonsense knowledge to answer why-questions. In Proceedings of the 2022 Conference on Empirical Methods in Natural Language Processing, pages 1204–1219, Abu Dhabi, United Arab Emirates. Association for Computational Linguistics.

Duong Minh Le, Ruohao Guo, Wei Xu, and Alan Ritter. 2023. Improved instruction ordering in recipe-grounded conversation.

Yifei Li, Zeqi Lin, Shizhuo Zhang, Qiang Fu, Bei Chen, Jian-Guang Lou, and Weizhu Chen. 2023. Making language models better reasoners with step-aware verifier. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 5315–5333, Toronto, Canada. Association for Computational Linguistics.

Hung-Hsin Lin, Nien Wei, T-Y. Chou, Chun-Chi Lin, Yuan-Tsu Lan, Shin-Ching Chang, Huann-Sheng Wang, Shung-Haur Yang, Wei-Shone Chen, Tzu chen Lin, Jen-Kou Lin, and Jeng-Kai Jiang. 2017. Building personalized treatment plans for early-stage colorectal cancer patients. Oncotarget, 8:13805 – 13817.

Qing Lyu, Li Zhang, and Chris Callison-Burch. 2021. Goal-oriented script construction. In Proceedings of the 14th International Conference on Natural Language Generation, pages 184–200, Aberdeen, Scotland, UK. Association for Computational Linguistics.

Qingsong Ma, Johnny Wei, Ondřej Bojar, and Yvette Graham. 2019. Results of the WMT19 metrics shared task: Segment-level and strong MT systems pose big challenges. In Proceedings of the Fourth Conference on Machine Translation (Volume 2: Shared Task Papers, Day 1), pages 62–90, Florence, Italy. Association for Computational Linguistics.

Xiao Ma, Swaroop Mishra, Ariel Liu, Sophie Su, Jilin Chen, Chinmay Kulkarni, Heng-Tze Cheng, Quoc Le, and Ed Chi. 2023. Beyond chatbots: Explorellm for structured thoughts and personalized model responses.

Bodhisattwa Prasad Majumder, Harsh Jhamtani, Taylor Berg-Kirkpatrick, and Julian McAuley. 2020. Like hiking? you probably enjoy nature: Persona-grounded dialog with commonsense expansions. In Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP), pages 9194–9206, Online. Association for Computational Linguistics.

Bodhisattwa Prasad Majumder, Shuyang Li, Jianmo Ni, and Julian McAuley. 2019. Generating personalized recipes from historical user preferences. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 5976–5982, Hong Kong, China. Association for Computational Linguistics.

Shinsuke Mori, Hirokuni Maeta, Tetsuro Sasada, Koichiro Yoshino, Atsushi Hashimoto, Takuya Funatomi, and Yoko Yamakata. 2014. Flow-Graph2Text: Automatic sentence skeleton compilation for procedural text generation. In Proceedings of the 8th International Natural Language Generation Conference (INLG), pages 118–122, Philadelphia, Pennsylvania, U.S.A. Association for Computational Linguistics.

Al Mamunur Rashid, Istvan Albert, Dan Cosley, Shyong K. Lam, Sean M. McNee, Joseph A. Konstan, and John Riedl. 2002. Getting to know you: Learning new user preferences in recommender systems. In Proceedings of the 7th International Conference on Intelligent User Interfaces, IUI '02, page 127–134, New York, NY, USA. Association for Computing Machinery.

Keisuke Sakaguchi, Chandra Bhagavatula, Ronan Le Bras, Niket Tandon, Peter Clark, and Yejin Choi. 2021. proScript: Partially ordered scripts generation. In Findings of the Association for Computational Linguistics: EMNLP 2021, pages 2138–2149, Punta Cana, Dominican Republic. Association for Computational Linguistics.

Noor Shaker, Georgios Yannakakis, and Julian Togelius. 2010. Towards automatic personalized content generation for platform games. Proceedings of the AAAI Conference on Artificial Intelligence and Interactive Digital Entertainment, 6(1):63–68.

Shanya Sharma, Manan Dey, and Koustuv Sinha. 2021. Evaluating gender bias in natural language inference.

Julian Togelius, Georgios N. Yannakakis, Kenneth O. Stanley, and Cameron Browne. 2011. Search-based procedural content generation: A taxonomy and survey. IEEE Transactions on Computational Intelligence and AI in Games, 3(3):172–186.

Adithya V Ganesan, Yash Kumar Lal, August Nilsson, and H. Schwartz. 2023. Systematic evaluation of GPT-3 for zero-shot personality estimation. In Proceedings of the 13th Workshop on Computational Approaches to Subjectivity, Sentiment, & Social Media Analysis, pages 390–400, Toronto, Canada. Association for Computational Linguistics.

Sebastian Vincent, Rowanne Sumner, Alice Dowek, Charlotte Blundell, Emily Preston, Chris Bayliss, Chris Oakley, and Carolina Scarton. 2023. Personalised language modelling of screen characters using rich metadata annotations. arXiv preprint arXiv:2303.16618.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc V Le, Ed H. Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. In The Eleventh International Conference on Learning Representations.

Charles Welch, Chenxi Gu, Jonathan K. Kummerfeld, Veronica Perez-Rosas, and Rada Mihalcea. 2022. Leveraging similar users for personalized language modeling with limited data. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 1742–1752, Dublin, Ireland. Association for Computational Linguistics.

Zhiheng Xi, Wenxiang Chen, Xin Guo, Wei He, Yiwen Ding, Boyang Hong, Ming Zhang, Junzhe Wang, Senjie Jin, Enyu Zhou, Rui Zheng, Xiaoran Fan, Xiao Wang, Limao Xiong, Yuhao Zhou, Weiran Wang, Changhao Jiang, Yicheng Zou, Xiangyang Liu, Zhangyue Yin, Shihan Dou, Rongxiang Weng, Wensen Cheng, Qi Zhang, Wenjuan Qin, Yongyan Zheng, Xipeng Qiu, Xuanjing Huang, and Tao Gui. 2023. The rise and potential of large language model based agents: A survey.

Georgios N. Yannakakis and Julian Togelius. 2011. Experience-driven procedural content generation. IEEE Transactions on Affective Computing, 2(3):147–161.

Ori Yoran, Tomer Wolfson, Ben Bogin, Uri Katz, Daniel Deutch, and Jonathan Berant. 2023. Answering questions by meta-reasoning over multiple chains of thought. In Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, pages 5942–5966, Singapore. Association for Computational Linguistics.

Siyu Yuan, Jiangjie Chen, Ziquan Fu, Xuyang Ge, Soham Shah, Charles Jankowski, Yanghua Xiao, and Deqing Yang. 2023. Distilling script knowledge from large language models for constrained language planning. In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers), pages 4303–4325, Toronto, Canada. Association for Computational Linguistics.

Li Zhang. 2022. Reasoning about procedures with natural language processing: A tutorial. arXiv preprint arXiv:2205.07455.

A The CUSTOMPLANS Evaluation Set

Prior work on editing and customizing procedures has focused on specific domains. So, data for the task of procedure customization is limited. To alleviate this problem, we re-purpose two related datasets and collect more high-quality data from WikiHow to create the CUSTOMPLANS evaluation set.

A.1 CUSTOMPLANSREAL

Each WikiHow article consists of a goal (indicated in the title starting with "How to") and ordered, richly described steps that lead to that goal. It already contains articles that include multiple methods for a task. Additionally, search queries can be used to retrieve and select articles broadly related to the same goal. However, user interaction signals like fine-grained querying related to specific versions of procedures are not available publicly. Such signals can be used to improve user experiences. Therefore, we create a new evaluation set that contains a procedural goal and an individual user's relevant context.

To compile this data, we start by identifying 9 diverse domains. For each domain, we employ two methods to curate WikiHow articles. 1) Search Results: We devise relevant, popular and pragmatic goals which have relevant instructions present in WikiHow. We present each goal to users of relevant persona and ask them to provide their feedback according to their personal preferences. For example, for the broad goal of staying healthy, a collected user persona was that of a teenager looking to build long-term healthy habits. 2) Multiple Methods: For the same set of broad goals, we filter out WikiHow articles that contain multiple methods to achieve the same goal. Given the broad goal, users are required to provide situational feedback corresponding to each method. Using this approach, we collect 106 customized goals (corresponding to user feedback) over 9 domains and build CUSTOMPLANSREAL, which contains customization hints collected from humans.

A.2 CUSTOMPLANSSIMULATED

To study decision branching in scripts, Hou et al. (2023) created CHOICE-75, a benchmark of 565 data points which requires selecting the next step in a procedure given descriptive scenarios. These scenarios are valid only at a step level. However, they can also be incorporated into the full procedure if treated as constraints. We treat these scenarios as customization hints.

COPLAN is a dataset of machine-generated, human-verified scripts. We use the test set of 861 data points for the counterfactual plan revision task for our purposes. Brahman et al. (2023) collect goals from diverse topics and prompt `text-curie-001` to generate a set of ordered steps as a plan to achieve that goal. `text-curie-001` is also used, in a few-shot manner, to generate conditions that can alter these plans. Human verification of these components leads to the creation of a test set that contains a goal, an uncustomized procedure and a relevant condition. These conditions describe prerequisites to be fulfilled that require the original procedure to be customized in a specific manner. We treat these conditions as customization hints. The combination of data from these two sources is referred to as CUSTOMPLANSSIMULATED since it contains customization hints generated automatically.

A.3 Statistics: CUSTOMPLANS evaluation set

We randomly select 100 procedures and their corresponding customization hints from CUSTOMPLANSSIMULATED and add them to CUSTOMPLANSREAL to form the CUSTOMPLANS evaluation set. Overall, CUSTOMPLANS contains 206 data points, each of which is made up of a goal, a list of steps to achieve that goal and a customization hint from a user according to which the procedure should be modified. It contains 106 unique goals and 203 unique customization hints.

We also store relevant metadata for each user persona providing the annotation such as their level of expertise and their constraints like dietary preferences or availability of tools. 6.2% of these customized procedures can only be performed by domain experts, 11.6% need an intermediate level of expertise, and the rest are beginner-friendly. The user comments contain implicit or explicit constraints — 58% contain hard prerequisites, 30% reflect some user preference and the rest mention a target refinement to be achieved. The types of customization hints, as well as corresponding examples, in this dataset are presented in Figure 13.

B Mechanical Turk tasks

We present the instructions given to annotators for both the tasks in Figure 14. Annotators were given clear direction for the task, as well as provided

Decorate a room	Stay healthy	Bake bread
Constraints **Prerequisite** I only use oil-based paint **Preference** I like flowers and origami **Refinement** I have to beautify the furniture	*Expertise* **Beginner** I eat a lot of junk food **Intermediate** I want to build long-term habits **Expert** I used to do strength training	*Both* **Preference + Beginner** I like Hawaiian flavors **Prerequisite + Intermediate** I only have a stovetop **Refinement + Expert** I have been cooking for years and want to make some special bread

Figure 13: Types of customization hints present in CUSTOMPLANS, enabling analysis of diverse customization hints. Prerequisites are usually expressed explicitly, while preferences and refinements may be expressed implicitly in the hint. Expertise usually needs to be inferred from the hint, except in hints collected experts. Some hints also encode both user expertise as well as constraints in their scenarios.

examples for various cases of desirable and undesirable characteristics of a procedure. We restricted the task to master turkers. The master turkers are paid 0.30$ per HIT, which translates to 17$/hr according to average time needed for completion. We do not collect any demographic information, or apply any restrictions on who can provide judgments (except master turker qualification). The average lifetime approval rate of the turkers is 97.98% while their average approval rate over the last 30 days was 97.06%.

C Reproducibility

C.1 LLM Settings

We used a temperature of 0.0 for all the experiments to select the most likely token at each step, as this setting allow for reproducibility[2].

We use the following code snippet for any experiments performed with `gpt-3.5-turbo`:

```
import openai
openai.api_key = os.getenv("OPENAI_API_KEY")
response = openai.ChatCompletion.create(
            model="gpt-3.5-turbo-0301",
            messages=[{'role': 'user',
                       'content': prompt}],
            temperature=0.0, # reproducibility.
            max_tokens=500,
            top_p=1,
            frequency_penalty=0.1,
            presence_penalty=0
            )
```

[2]We note that some researchers have shown that even this setting might not make it completely reproducible: https://twitter.com/ofirpress/status/1542610741668093952?s=46&t=f9v5k9RzVKnTK1e0UyauOA and https://twitter.com/BorisMPower/status/1608522707372740609

C.2 Prompts Used

We list the prompts we use in different parts of our methods in Figure 15.

Thanks for participating in this HIT!

Please read carefully:

- As you are working through these hits, you may see repeats in goal/plan. In such cases, conditions are different.

In this HIT, you are shown a goal, and a condition to the goal. You will also be provided with a plan that is meant to fulfill the condition in order to make the goal achievable. Following are the definitions.

Goal	A goal/desire that can be achievable through a plan. (e.g., go to Hawaii, spend my Sundays at the beach, and so on)
Condition	A **constraint** or a **particular preference** that is applicable to the plan. (e.g., for the above goals, "not having a car", "visit during a Aloha Week", and so on.)
Plan	A **step-by-step proposed actions to achieve** the goal which consists of a list of typical subgoals or steps to achieve the main goal which also takes the condition into account. (e.g., for the condition "not having a car" when the goal is to "visit Hawaii": 1. buy ticket to Hawaii, 2. decide what you want to see, 3. book lodging, 4. look for shared cabs, 5. pack, 6. leave for the airport)

Think about how you would go about writing a procedure yourself to achieve this goal given the condition. **YOUR TASK** is to evaluate the **quality** of the plan by answering these questions.

Q1: Could you follow the given steps to achieve the goal?

- Yes, I can follow it exactly and achieve the goal.
- No, some step(s) should be **changed** - Details in one (or more) step(s) aren't right, and should be edited.
- No, important step(s) is/are **missing** - Additional steps are required in order for the procedure to be successful.
- No, some step(s) should be **deleted** - Some steps are simply wrong, irrelevant, or duplicates, and should be removed.
- No, some step(s) is/are **vague** - It is unclear what the step means, and/or how to perform it (e.g., "feel happy").

Q2: A new condition has happened. Can you still follow the plan to achieve the task given the new condition?

- Yes: The plan contains **all the necessary steps** to meet the requirements of the condition on the goal.
- No, some step(s) should be **changed** - Details in one (or more) step(s) aren't right, and should be edited.
- No, important step(s) is/are **missing** - Additional steps are required in order for the procedure to be successful.
- No, some step(s) should be **deleted** - Some steps are simply wrong, irrelevant, or duplicates, and should be removed.
- No, some step(s) is/are **vague** - It is unclear what the step means, and/or how to perform it (e.g., "feel happy").

NOTES:

- Steps are allowed to be general so long as the key information is there. Think: is the plan enough to give students solid grounding to start of asking relevant questions and taking relevant steps to achieve the goal?
- Please do not hover too much over fine-grained differences. When in doubt, choose go with your gut instincts.

Figure 14: Instructions for MTurk tasks

E2E customize	[{“role”: “user”, “content”: f“Write a list of steps for the following goal\nGoal: {goal}”}, {“role”: “assistant”, “content”: f“Steps:\n{procedure}”}, {“role”: f“My situation is that {customization_hint}. List the new set of steps.”}]
MODIFY	[{'role': 'user', 'content': f"Here is a student generated steps to {goal} given the condition that "{customization_hint}".\n\n{procedure}\n\nYour task is to go over every step and suggest a change in the step only if the condition is not met. Suggest changes only if really necessary. \n\nWhen changing a step M, you are only allowed to use these two operations: \nreplace(3, XX): Replace the full text of step 3 with new full text XX\ninsert(2, XX): Insert the full text XX as a new step after step 2\n\nGive the set of revisions."}]
VERIFY	[{'role': 'user', 'content': f"Here is a student generated steps to {goal} given the condition that "{customization_hint}".\n\n{procedure}\n\nYour task is to go over every step and suggest a change in the step only if the step won't work. Suggest changes only if really necessary. \n\nWhen changing a step M, you are only allowed to use these two operations: \nreplace(3, XX): Replace the full text of step 3 with new full text XX\ninsert(2, XX): Insert the full text XX as a new step after step 2\n\nGive the set of revisions."}]
RESOLVE	[{'role': 'user', 'content': f”Here is a student generated steps to {goal} given the condition that "{customization_hint}".\n\n{procedure}\n\nThe student realized their mistake and decided to edit the steps in the following way. \n\n{customization_edits}\n{execution_edits}\nHowever, these edits can be repetitive, conflicting or unnecessary. Now, imagine you are a teacher. Your task is to tell the student the correct set of edits. You must only give the essential edits.\n\nCorrect and minimal set of edits are:\n”}]

Figure 15: Prompts used in various parts of our experiments.