# NEURO-SYMBOLIC CAUSAL PROCEDURAL PLANNING WITH COMMONSENSE PROMPTING

**Anonymous authors**
Paper under double-blind review

## ABSTRACT

Procedural planning aims to implement complex high-level goals by decomposition into sequential simpler low-level steps. Such procedural reasoning ability is essential for applications such as household robots and virtual assistants. Although procedural planning is a basic skill set for humans in daily life, it remains a challenge for large language models (LLMs) that lack deep-level commonsense knowledge in the real world. Previous methods require either manual exemplars or annotated programs to acquire such ability from LLMs. In contrast, this paper proposes a neuro-symbolic **C**ausal **LA**nguage **P**lanner (CLAP) that elicits procedural knowledge from the LLMs with commonsense-infused prompting. Pre-trained knowledge in LLMs is essentially an unobserved confounder that causes spurious correlations between tasks and action plans. Through the lens of a Structural Causal Model (SCM), we propose an effective strategy in CLAP to construct prompts as a causal intervention toward our SCM. Using graph sampling techniques and symbolic program executors, our strategy formalizes the structured causal prompts from commonsense knowledge bases. CLAP obtains state-of-the-art performance on WikiHow and RobotHow, achieving a relative improvement of $5.28\%$ in human evaluations under the counterfactual setting. This indicates the superiority of CLAP in causal procedural planning semantically and sequentially.

## 1 INTRODUCTION

How to make a cup of coffee? As humans, we can easily identify the procedure with the commonsense reasoning ability we acquire. However, can we endow machines with the same ability to sketch out a sequential plan list? Procedural planning (Huang et al.; Lyu et al., 2021; Wu et al., 2022; Zhang et al., 2020a;b) is a task that aims to decompose a specific task into a sequence of temporally extended procedural steps. Such ability is a necessity for a wide range of downstream applications, such as household robots (Ahn et al., 2022; Blukis et al., 2021; Min et al., 2022; Padmakumar et al., 2022; Shridhar et al., 2020) and conversational assistants (Budzianowski et al., 2018; Ding et al., 2022; Eric et al., 2017; Ilievski et al., 2018; Yang et al., 2022).

Existing formalization of the procedural planning task can mainly be categorized as: 1) sequential choice making (Lyu et al., 2021; Wu et al., 2022; Zhang et al., 2020a;b), which reasons about the next step from the options given, the task and previous steps; 2) controllable planning, which prompts the large language models to generate procedural texts that satisfy domain-specific admissible steps set. We study the controllable planning problem since the sequential choice-making requires a skillfully annotated dataset and is distinct from real-world scenarios. Previous approaches require either holdout samples (Huang et al.) or manual exemplars (Wei et al., 2022) to acquire such procedural knowledge. However, it is impossible to cover the ever-changing task setups and the flexible dependency relations between internal steps. In fact, the biased data may cause the model to learn spurious correlations and hinder the model from generalizing to counterfactual tasks.

While existing approaches require annotated exemplars, we take a step back and rethink how humans conduct procedural planning. Studies in cognitive science state that "chunking" - the process of breaking down a complex problem into subcomponents - is important to humans' information-processing (de Groot, 1978; Gobet et al., 2001; Miller, 1956), and children learn such ability at an early age (Dietz et al., 2019). Humans also consider past experience during problem-solving (Duncker & Lees, 1945; Knoblich et al., 1999; Smith, 1995; Zhang et al., 2019). Inspired by humans' experience-

based problem-solving and chunking mechanisms, we propose a neuro-symbolic planning model using commonsense-infused prompting to tackle the causal procedural planning task without annotated exemplars or further training.

We first devise a Structural Causal Model (SCM) (Peters et al., 2017) that describes the causal effect between high-level goals and low-level steps in the procedural planning system. We show that pre-trained knowledge in LLMs confounds language models to make biased decisions toward an unreasonable action. Since the confounder is unobservable, intervention techniques such as backdoor adjustment (Hu & Li, 2021; Weber et al., 2020; Yue et al., 2020) are not applicable in our SCM. Instead, we build a mediator and implement it as a commonsense prompt. Through the mediator, we can identify causal effect from goals to planning steps by investigating the indirect effect from the goals, which is essentially the frontdoor adjustment in causality (Pearl, 2009).

Then we apply a symbolic program (Mao et al., 2019; Yi et al., 2018) to construct procedural commonsense knowledge from external resource that attends to the high-level tasks. The program that composes of the goal and step is considered a procedural structure. Thus directly incorporating the graph structure external commonsense knowledge into language models cannot maximally elicit reasoning ability for procedural planning. To mitigate the structure gap, we parse the goal into an entity to retrieve the subgraph that represents the common knowledge related to the entity, including the prerequisite event, subsequent event, location, usage, etc. Finally, we semantically translate the knowledge to the admissible procedure and feed them into the LLMs as commonsense-infused prompts.

We conduct experiments on two datasets that involve completing daily household goals, RobotHow (Puig et al., 2018) and WikiHow (Koupaee & Wang, 2018). Furthermore, we devise 3 families of automatic intervention methods to construct a counterfactual dataset based on the original dataset. CLAP surpasses powerful baselines (Chain of Thoughts (Wei et al., 2022) and Zero-shot Planner (Huang et al.)) by large margins on both the original and the counterfactual samples. Our major contributions in this work can be summarized as:

- We propose a neuro-symbolic causal planning model using commonsense-infused prompting to tackle the procedural planning task without annotated exemplars or further training.
- We define a temporally extended Structural Causal Model for procedural planning and resolve spurious correlation between high-level tasks and low-level steps via frontdoor adjustment with a prompt-based mediator.
- Our evaluation results show the state-of-the-art performance of CLAP on semantic and sequential metrics in standard and counterfactual settings.

## 2 METHODOLOGY

Our goal is to build a language planner to decompose high-level tasks into lower-level time-extended steps. We define a Structural Causal Model of Our CLAP in Figure 2, which describes the temporal dynamics and procedural cause-effect relationship. Our causal assumption in SCM indicates that there is a backdoor path from task to steps, which must be blocked with frontdoor adjustment. Therefore, we model the input prompt as a mediator which is created from external commonsense knowledge in a neuro-symbolic approach. This allows us to to prompt the procedural reasoning ability of the language model in a zero-shot setting. The overview of CLAP is shown in Figure 1 and the full procedure is shown in Algorithm 1.

### 2.1 PROBLEM DEFINITION

Given the high-level task $T$ (e.g. watch television in the living room) sampled from a task domain $D_T$ (e.g. RobotHow), the procedural planning task requires the agent to reason the admissible low-level step $\mathbf{S} = \{s_1, ..., s_t | s \in \bar{S}\}$ at each $t$ timestep to complete the task. Such task decomposition is achieved by eliciting the knowledge from the large pre-trained language model (LLM) auto-regressively as $P(S_t | D_T, T, A, O, s_{0:t-1})$. Each $s_t$ is composed of sampled action $a$ and object $o$ from the action set $A$ and the object set $O$ respectively, which satisfy certain affordances in the task domain.

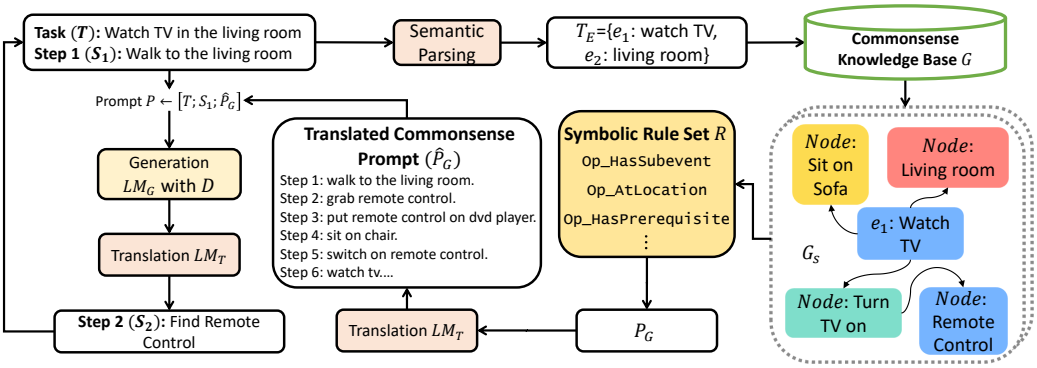



Figure 1: **The Overview of Causal Language Planner.** The architecture is composed of three main components: 1) parsing tasks into structural concept subgraph based on the external commonsense knowledge base. 2) constructing causal mediator with procedural prior. 3) generating and translating time-extended steps auto-regressively.

**Algorithm 1** Neuro-Symbolic Causal Procedural Planning using Commonsense-Infused Prompting

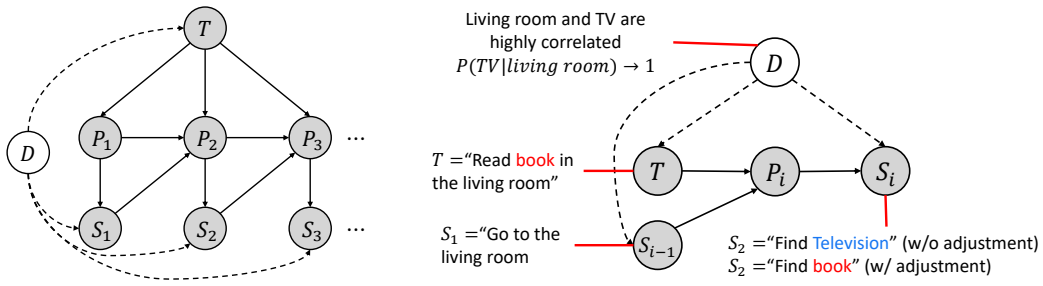
**Require:**
Task Dataset $\mathcal{D}$, Admissible Step Set $S$, External knowledge Graph $G$;
Language Model for Generation $LM_G$ and Translation $LM_T$, Symbolic Rule Set $R$;
**Ensure:**
Zero-shot Inference for Procedure $S_T = s_1, ..., s_t$;
**for** each Task Sample $T$ in $\mathcal{D}$ **do**
  Parse $T$ into entity set $T_E$;
  Retrieve commonsense subgraph $G_s$ for each $e \in T_E$;
  Construct procedural commonsense knowledge $P_G$ using Symbolic Rule Set $R$;
  Translate $P_G$ in Admissible Knowledge $\hat{P_G} = LM_T(P_G)$;
  Prompt $P \leftarrow$ Initial task name $T$ + External Knowledge $\hat{P_G}$;
  **for** each timestep $t$ **do**
    $s_t = LM_T(LM_G(P_t))$;
    Generated procedure $S_T \leftarrow s_t$, Prompt $P_{t+1} \leftarrow s_t$;
  **end for**
**end for**

## 2.2 External Knowledge Injection

We define our Full Temporal Causal Graph as in Figure 2a, which is an unrolled Structural Causal Model (SCM) for sequential decision making. Our goal is to identify the causal effect between attended task $T$ and plan procedures $\mathbf{S} = \{S_1, S_2, \ldots\}$ from LLMs. Initially, there are direct paths $T \rightarrow S_i$ and $S_k \rightarrow S_i, k < i$ because $S_i$ relies on the LLM attended task entities and previous accomplished steps. $D$ is an unobserved confounder from learned knowledge during pre-training. $D$ builds a backdoor path between $T$ and $S_i$ and misguides the LLM to attend on false entities to generate the next step (see Fig. 2b). Note that $D$ is unobservable as we directly adopt the LLM without knowing the pre-training data. To mitigate the spurious correlation, we introduce a mediator $P_i$ for each $S_i$ as shown in Figure 2a to achieve our frontdoor adjustment by injecting external commonsense knowledge into LLMs in a neuro-symbolic approach.

**Step 1: Task-Specific Knowledge Subgraph Sampling** In the first step, we investigate the causal effect $T \rightarrow P_i$ and $S_{i-1} \rightarrow P_i$. As a language prompt, $P_i$ inherit the content from $P_{i-1}$ and thus can be detached from steps before $S_{i-1}$ for simplicity. Note that in the path $T \rightarrow D \rightarrow S_i \leftarrow P_i$, $S_i$ is a collider that blocks the association between $D$ and $P_i$. Since there is no backdoor path for $T \rightarrow P_i$ and similarly for $S_{i-1} \rightarrow P_i$, we simply have the conditional probability after applying *do*-operators:

$$P(P_i = p|do(T)) = P(P_i = p|T), \quad P(P_i = p|do(S_{i-1})) = P(P_i = p|S_{i-1}) \tag{1}$$




(a) Full temporal causal graph

(b) SCM at step $i$

Figure 2: **The Structural Causal Model of our Language Planner.** (a) the full temporal causal graph for procedural planning. $T$ denotes the task name and $S_i$ denotes the step at timestep $i$. $D$ is the unobservable confounding variable introduced by the pre-training data. $P_i$ denotes the mediating variables we construct to mitigate the spurious correlation. (b) the structural causal model at timestamp $i$. $D$ opens up backdoor paths for $T \rightarrow S_i$ and $S_{i-1} \rightarrow S_i$ which can be blocked by introducing $P_i$.

To achieve the *do*-operation in a prompting way, we modify the input so that the model can attend on the correct entities. We extract the entities from the task name, and retrieve a one-hop graph from an external knowledge base. The graph can be verbalized and applied as a prompt so that the LLM attends on the correct entities to plan $S_1$.

Specifically, the high-level task name $T$ is parsed into the entity set $T_E$ with the named entity information to extract the representative nodes from the graph. We use NLTK to tokenize and `pos_tag` the task text. Then we use the noun (e.g. television), noun phrases (e.g. remote control) and verb phrases (e.g. watch television) as the entity node. Each entity $e \in T_E$ is used as a query for sampling the local subgraph $G_e \subseteq \mathcal{N}_e \times \mathcal{R}_s \times \mathcal{N}_e$ from the external knowledge base $G \subseteq \mathcal{N} \times \mathcal{R} \times \mathcal{N}$ (e.g., ConceptNet (Speer et al., 2017)), where $\mathcal{N}$ and $\mathcal{R}$ represent the number of concept nodes and commonsense relations respectively. When extracting subgraphs, we consider the triplets with relation type in household domain (e.g., `AtLocation`, `UsedFor`) instead of the linguistic domain (e.g., `DistinctFrom`, `DerivedFrom`) to the procedural planning task are kept, while the others are filtered out. $\mathcal{N}_e$ is maintained in a set of top-$k$ task-specific nodes using the weight of each $R_e$, which is adapted using the edge-wise invariance module illustrated in the next paragraph. The global subgraph $G_s$ is a composite of the $\{G_e, e \in E\}$, representing the task-specific external knowledge.

**Step 2: Edge-wise Adaption and Symbolic Commonsense Prompt Construction** In the second step, we need to find the causal effect for $P_i \rightarrow S_i$. Since the path $P_i \leftarrow T \leftarrow D \rightarrow S_i$ contains a backdoor from $P_i$ to $S_i$, we cannot rely on the conditional probability. Instead, we add intervention on $P_i$ by applying the *do*-operator to cut off $D \rightarrow T$:

$$\begin{aligned} P(S_i|do(P_i = p)) &= \sum_{t,s} P(S_i|p, T = t, S_{i-1} = s)P(T = t, S_{i-1} = s) \\ &= \sum_{t,s} P(S_i|p, T = t, S_{i-1} = s)P(S_{i-1} = s|T = t)P(T = t) \end{aligned} \tag{2}$$

The retrieved entity-centered graph has multiple edges representing various relationships with other actions/entities. Therefore, the summation over intervened $T$ can be achieved by incorporating these edges into the prompt. For instance, "living room" can be "walked to" and "used for reading" while "book" can locate in "living room" and "bedroom". Similarly, we extrapolate over the edges for $i-1$ hops to aggregate the intervened $S_i$, i.e. $P(S_{i-1} = s|T = t)$. We refer to this aggregation and conversion to prompt as **Edge-wise Adaption** and **Symbolic Commonsense Prompt Construction**.

Directly ranking the retrieved nodes $N_e$ with the pre-computed weight in the external knowledge base will result in spurious correlation. Such retrieved local subgraphs tend to capture the task-invariant concept nodes as the causal factors. To mitigate this, we propose to adapt the weight of each triplet (**Edge-wise Adaption**). The adaptive weight is the addition of the original edge weight and the cosine similarity between the tail node embedding $\boldsymbol{n}_{tail}$ and the task embedding $\boldsymbol{v}_{task}$. The embeddings are

projected from the node text and task name using the sentence-transformer (Reimers & Gurevych, 2019) based translation language model $LM_T$.

To better track the utilized external knowledge during inference, we construct the task-dependent commonsense prompt with a symbolic program executor (**Symbolic Commonsense Prompt Construction**) guided by the relation type of each triplet in $G_s$ with the adapted edge weight beyond threshold $\theta_e$. Specifically, the executor acquires the neural information of each natural language node and executes the sequential mapping program by sampling the operation $Op$ from the Symbolic Rule Set $R$ according to the edge relation type. For instance, the $AtLocation$ edge samples the operation `Op_AtLocation` from $R$, which takes the commonsense relation of the triplet from $G_s$ as the parameters to query the procedural concept output given the natural language meaning of the linked nodes (e.g., go to the location of `Start_Node_Of`($r_e$) in this case). Similarly, `Op_UsedFor` may refer to "go to find `End_Node_Of`($r_e$) and use it for `Start_Node_Of`($r_e$)". And operators `Op_HasSubevent` and `Op_HasPrerequisite` will recursively navigate the subgraph $G_e$. After navigating the subgraph, we linearize the transformed triplets as the final **Symbolic Commonsense Prompt** $P_G$, which is then translated to **Admissible Knowledge** $\hat{P_G}$ by the translation language model $LM_T$.

Combining the two steps, we can obtain a prompt with aggregation of external knowledge in a procedural script format. The underlying causal mechanism is a combination of Eq. 1 and Eq. 2:

$$\begin{aligned} P(S_i|do(T), do(S_{i-1})) &= \sum_p P(S_i|do(P_i = p))P(p|do(T), do(S_i)) \\ &= \sum_p P(p|T) \sum_{t,s} P(S_i|p, T = t, S_{i-1} = s)P(T = t, S_{i-1} = s) \end{aligned} \tag{3}$$

The adjustment and marginalization in Eq. 3 is achieved in the input space by forming neuro-symbolic prompts that allows the LLM to attend on the causal entities instead of the highly correlated ones for next step prediction. We relates the task entities with abundant concepts from an external knowledge base originally in a graph format. The LLM can reason over the most relevant edges to link the concepts with the task entities as a context. The prompts from knowledge bases are independent of the pre-training data distribution so that $P_i$ is independent of $D$ and satisfies the frontdoor criterion. Please refer to Appendix A for more details on the simplification of our causal temporal graph.

### 2.3 Procedural Planning

**Semantic Generation** The external knowledge is further concatenated with the high-level task name as the initial prompt. The initial prompt contains limited semantics and executability, but still contains external knowledge and structure that help leverage the reasoning ability of the large language models. Given the prompt, the language model $LM_G \in \{P_{AR}, P_{AE}\}$ generates the next sentence, and the most confident prediction is then appended to previous prompts. The language model terminates when reaching the max step $t$ or has confidence below threshold $\theta$. The joint probabilities of auto-regressive ($P_{AR}$) and auto-encoder ($P_{AE}$) model is factorized as:

$$P_{AR}(x) = \prod_{i=1}^{n} p(s_n|\hat{P_G}, s_{1:n-1}, T), \quad P_{AE}(x) = \prod_{i=1}^{n} p(s_n|\hat{P_G}, \{s_{1:n-1}, [MASK]\}, T) \tag{4}$$

where $\hat{P_G}$ represent the commonsense knowledge and $T$ represent the task name.

**Admissible Step Translation** Procedural planning requires grounding the step into an admissible composite of action and object. Following (Huang et al.), we utilize a sentence-transformer based translation language model $LM_T$ for semantic matching with cosine similarity:

$$P(s_i|x) = LM_T(LM_G(x)), \text{where } s_i = \{k|k \in [a, o], a \in A, o \in O\} \tag{5}$$

which translates $LM_G(x)$ into the admissible step $s_i \in S$ that is the closest in the embedding space measured by the cosine similarity. $A$ and $O$ represent the action set and object set respectively.

### 2.4 Counterfactual Procedural Data Construction

To investigate the counterfactual reasoning ability of the procedural planning models, we design three families of intervention methods: 1) **Initial Configuration**: intervene the initial configuration,

such as the location for implementing the task. 2) **Intermediate Step**, randomly selecting the step from the program as an additional constraint the task name for implementing the task. 3) **Final Goal**, intervene the task goal as the composite of another randomly sampled task. The counterfactual dataset construction details and post-intervention examples are provided in Appendix B.2.

# 3 EXPERIMENTS

## 3.1 PROCEDURAL PLANNING SETUP

**Datasets** We conduct zero-shot experiments on two datasets with procedural information, WikiHow[1] (collected following (Koupaee & Wang, 2018)) and RobotHow (Puig et al., 2018) without training. **WikiHow** is a large-scale text summarization dataset that is constructed from a human-written knowledge base, involving procedural tasks that spans various topics. We utilize "how to" title as the task names and the summarized headlines as the steps. **RobotHow** is a large knowledge base of common household tasks collected in the VirtualHome (Puig et al., 2018) simulator. The dataset contains the programs with high-level task names and low-level steps. $D_T$ is composed of 292 and 2000 distinct tasks from RobotHow and WikiHow respectively. In RobotHow, $A$ is of size 42 and $O$ is of size 195. In WikiHow, $A$ is of size 1161 and $O$ is of size 13873. Human evaluations use randomly sampled 50 task examples for each dataset. Automatic evaluations use 150 and 1000 task examples randomly sampled from RobotHow and WikiHow respectively. Table 1 summarizes the category, description, and example of our designed intervention methods. Please refer to Appendix B.1 and Appendix B.2 for dataset details.

| Category | Description | Example |
|---|---|---|
| Initial Configuration | Constrain the environment configuration, e.g. location | Watch TV in bedroom |
| Intermediate Step | Constrain the way to finish the task | Work (Find Computer) |
| Final Goal | Change the final effect of the task by composition | Watch youtube and Put away jackets |

Table 1: **Counterfactual Procedural Planning.** Three types of methods, including initial configuration, intermediate step, and final goal are applied to intervene the original procedural data.

For the external knowledge base, we utilize ConceptNet to leverage commonsense reasoning ability to help ground language generation in goal-guided procedural text generation. ConceptNet (Speer et al., 2017) captures commonsense knowledge explicitly with triplets of (*head node*, *relation*, *end node*). It contains $799,273$ nodes and $2,487,810$ edges that represent both symmetric and asymmetric relations. Specifically, the core relations we utilized are *Synonym*, *AtLocation*, *CapableOf*, *Causes*, *CausesDesire*, *HasPrerequisite*, *HasSubevent*, and *UsedFor*. Since we are looking at the commonsense knowledge in house-holding tasks, so we filter out the relations (/r/DistinctFrom, /r/DerivedFrom, /r/SymbolOf, /r/EtymologicallyRelatedTo, /r/EtymologicallyDerivedFrom) that are related to the linguistic.

**Baselines** We compare our approach with two vanilla generative pre-trained language models (BART and GPT2) and two powerful generation baselines (Zero-shot Planner (Huang et al.) noted as "LLMaP" and Chain of Thought (Wei et al., 2022) noted as "Chain"). LLMaP propose a procedure to extract temporally extended plans from large pre-trained language models. Chain explores manually creating exemplars that mimic the reasoning process and uses them to prompt large language models for reasoning tasks. To compare with Chain on the procedural planning task, we manually generate exemplars that contain the chain of thought for $1\%$ of the inference task programs. In addition, we conduct the experiments with GPT3 (davinci) using OpenAI API (May, 2022). We perform a hyperparameter search for all evaluated methods. More implementation and configuration details can be found in Appendix B.

**Metrics** We ask human annotators on the Amazon Mechanical Turk platform to rate model performance on two aspects: 1) Coverage: captures semantic completeness. 2) Order: captures sequential order correctness. We evaluate each individual model setting with the 5-point Likert scale. We also conduct the head-to-head comparison between models. Each example is rated by 3 annotators. The human evaluation in the `Order` category depicts which set of steps can better complete the target

[1] `https://www.wikihow.com`

| Dataset | $\text{Model}_{base}$ | Original-Coverage | | | Original-Order | | | Counterfactual-Coverage | | | Counterfactual-Order | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | | Win(↑) | Tie | Lose(↓) | Win(↑) | Tie | Lose(↓) | Win(↑) | Tie | Lose(↓) | Win(↑) | Tie | Lose(↓) |
| **RobotHow** | BART (Lewis et al., 2020) | **46.67** | 31.33 | 22.00 | **50.00** | 22.67 | 27.33 | **42.00** | 22.67 | 35.33 | **50.00** | 18.67 | 31.33 |
| | GPT2 (Radford et al.) | **42.67** | 22.00 | 35.33 | **44.00** | 18.67 | 37.33 | **56.67** | 11.33 | 32.00 | **45.33** | 16.00 | 38.67 |
| | GPT3 | **50.00** | 23.33 | 26.67 | **53.33** | 23.33 | 23.33 | **54.67** | 16.67 | 28.67 | **56.00** | 15.33 | 28.67 |
| **WikiHow** | BART (Lewis et al., 2020) | **56.67** | 12.67 | 30.67 | **69.33** | 10.00 | 20.67 | **50.00** | 26.67 | 23.33 | **46.00** | 21.33 | 32.67 |
| | GPT2 (Radford et al.) | **48.00** | 16.00 | 36.00 | **49.33** | 11.33 | 39.33 | **46.67** | 16.67 | 36.67 | **44.67** | 19.33 | 36.00 |
| | GPT3 | **75.17** | 10.74 | 14.09 | **72.67** | 8.67 | 18.67 | **44.00** | 22.67 | 33.33 | **48.67** | 25.33 | 26.00 |

Table 2: Percentages of procedural planning results of CLAP that are better than, tied with, or worse than Planner (Huang et al.), in coverage and order metrics under original and counterfactual setting.

| Architecture | Model | RobotHow | | | | WikiHow | | | |
|---|---|---|---|---|---|---|---|---|---|
| | | Original | | Counterfactual | | Original | | Counterfactual | |
| | | Coverage | Order | Coverage | Order | Coverage | Order | Coverage | Order |
| | Human (Koupaee & Wang, 2018; Puig et al., 2018) | 3.76 | 3.70 | 3.60 | 3.96 | 3.98 | 3.92 | 3.94 | 3.86 |
| **BART (Lewis et al., 2020)** | Chain (Wei et al., 2022) | 2.99 | 2.80 | 2.71 | 2.76 | 2.88 | 3.42 | 3.34 | 2.97 |
| | LLMaP (Huang et al.) | 3.06 | 2.84 | 2.96 | 2.82 | 2.78 | 3.35 | 3.46 | 3.02 |
| | CLAP (Ours) | **3.16** | **3.10** | **3.07** | **2.98** | **3.05** | **3.47** | **3.62** | **3.18** |
| **GPT2 (Radford et al.)** | Chain (Wei et al., 2022) | 2.43 | 2.28 | 3.12 | 2.88 | 2.97 | 3.44 | 3.60 | 3.01 |
| | LLMaP (Huang et al.) | 3.09 | 2.94 | 2.93 | **2.90** | 3.20 | 3.53 | 3.63 | 3.24 |
| | CLAP (Ours) | **3.12** | **2.99** | **3.43** | 2.88 | **3.67** | **3.69** | **3.81** | **3.31** |

Table 3: Averaged 5-point Likert scale human evaluations on "coverage" and "order" aspects.

task, while the `Coverage` category depicts which sequence covers more steps that are necessary to complete the target task. In addition, we use Sentence-BLEU (S-BLEU) (Papineni et al., 2002), BERTScore (Zhang* et al., 2020), ROUGE-1 (Lin, 2004) and Word Mover's Distance (WMD) (Kusner et al., 2015) as automatic evaluation metrics. These metrics are used to compute the semantic scores between the annotated programs and the predictions. Details of the crowdsourcing human evaluation can be found in Appendix C.1.

### 3.2 Human Evaluation Results with Coverage and Order Metric

For the **Win-Lose Comparison**, we ask the human rater to choose between ours and the baseline LLMaP (Huang et al.). Results reported in Table 2 show that our CLAP is more frequently rated as better for both coverage and order metrics, outperforming baselines over the winning ratio by $21\%$ in coverage and $26\%$ in ordering, across two datasets. We report the average of 3 crowdsourcing **Human Ratings** in Table 3. The consistent performance boost of CLAP indicates the superiority of injecting external commonsense knowledge into the procedural planning task. The performance drop of LLMaP and Chain in the counterfactual setting indicates the vulnerability of fixed holdout knowledge and the pre-defined manual exemplars in causal procedural planning. Please refer to Appendix C.1 for the crowdsourcing human evaluation interface details. Table 6 shows two examples for **Qualitative Comparison**. More examples can be found in Appendix D.

### 3.3 Automatically Measuring the Procedural Planning

Table 4 summarizes The automatic evaluation results. CLAP achieves the best results regardless of the architecture of the language model architecture, either autoregressive or autoencoder based. The performance gain of "LLMaP" over "Chain" may probably be due to direct exposure to the holdout task from the dataset. While the "Chain" baseline still outperforms the vanilla baseline that only takes the high-level task name as the prompt. Note that the annotated program is not the only solution, thus these automatic metrics provide limited absolute performance information. Details for the correlation between automatic metrics and human evaluation can be found in Section 3.5.

**Effects of Edge-wise Adaption and Symbolic Program Execution** We build the variant "$w/o$ Adaption", which maintains the top-$k$ task-specific nodes ranked by the original weight in the external knowledge graph $G$ without adaption. And we build a variant "$w/o$ Symbolic" that directly takes the extracted concept nodes from external knowledge base as prompt. As shown in Table 4, these two variants experience consistent performance drop, demonstrating the importance of adaption and symbolic components. We conduct the paired-t test (p¡$0.05$) statistics results over the variant "$w/o$ Symbolic" and "$w/o$ Adaption". Compared with the full model CLAP, the variants have experienced

| Model | RobotHow | | | | WikiHow | | | |
|---|---|---|---|---|---|---|---|---|
| | S-BLEU | WMD | BERT-f1 | ROUGE-f1 | S-BLEU | WMD | BERT-f1 | ROUGE-f1 |
| BART Lewis et al. (2020) | 0.069 | 0.923 | 0.870 | 0.442 | 0.083 | 0.937 | 0.836 | 0.379 |
| BART + Chain (Wei et al., 2022) | 0.079 | 0.913 | 0.862 | 0.448 | 0.095 | 0.939 | 0.782 | 0.377 |
| BART + LLMaP (Huang et al.) | 0.094 | 0.940 | 0.870 | 0.467 | 0.131 | 0.950 | 0.816 | 0.371 |
| BART + CLAP | **0.110** | **0.951** | **0.890** | **0.528** | **0.142** | **0.958** | **0.833** | **0.400** |
| $w/o$ Adaption | 0.104 | 0.929 | 0.886 | 0.492 | 0.132 | 0.952 | 0.824 | 0.398 |
| $w/o$ Symbolic | 0.062 | 0.858 | 0.835 | 0.392 | 0.087 | 0.939 | 0.828 | 0.386 |
| GPT2 (Radford et al.) | 0.056 | 0.891 | 0.846 | 0.356 | 0.051 | 0.925 | 0.826 | 0.345 |
| GPT2 + Chain (Wei et al., 2022) | 0.079 | 0.906 | 0.861 | 0.405 | 0.124 | 0.937 | 0.817 | 0.352 |
| GPT2 + LLMaP (Huang et al.) | 0.115 | 0.931 | 0.885 | 0.481 | 0.115 | 0.957 | 0.833 | 0.363 |
| GPT2 + CLAP | **0.148** | **0.945** | **0.898** | **0.547** | **0.133** | **0.971** | **0.835** | **0.373** |
| $w/o$ Adaption | 0.142 | 0.944 | 0.896 | 0.542 | 0.123 | 0.965 | 0.830 | 0.360 |
| $w/o$ Symbolic | 0.143 | 0.942 | 0.895 | 0.538 | 0.121 | 0.967 | 0.829 | 0.357 |

Table 4: Automatic evaluation results on the Original RobotHow and WikiHow. Metrics are computed between the annotated programs and the predictions.

| Model | Initial Configuration | | | | Intermediate Step | | | | Final Goal | | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | S-BLEU | WMD | BERT-f1 | ROUGE-f1 | S-BLEU | WMD | BERT-f1 | ROUGE-f1 | S-BLEU | WMD | BERT-f1 | ROUGE-f1 |
| Chain (Wei et al., 2022) | 0.125 | 0.906 | 0.875 | 0.518 | 0.136 | 0.926 | 0.892 | 0.550 | 0.063 | 0.918 | 0.857 | 0.467 |
| LLMaP (Huang et al.) | 0.148 | 0.929 | 0.887 | 0.566 | 0.141 | 0.886 | 0.902 | 0.547 | 0.070 | 0.928 | 0.868 | 0.490 |
| CLAP (Ours) | **0.169** | **0.934** | **0.897** | **0.570** | **0.183** | **0.953** | **0.913** | **0.590** | **0.082** | **0.934** | **0.873** | **0.493** |

Table 5: Automatic evaluation results on the Counterfactual RobotHow with language model GPT2.

statistical significant performance drop. Especially on BERTScore-f1, the p-value is $8.884e-13$ and $1.4e-8$ respectively. This further confirms the importance of the modules.

**Effects of the Pre-trained Language Model Architecture** To investigate the effects of the pre-trained language model architecture, we use GPT2 as our autoregressive architecture and BART (Lewis et al., 2020) as our autoencoder architecture. The autoregressive architecture achieves better results than the autoencoder one. Since the pre-training objective of autoregressive-based GPT2 is to predict the next token given the previous input tokens. We assume the performance gain of GPT2 is due to a smaller gap between the objective of their pre-training and the procedural planning task.

**Level of Complexity** Under standard setting, we also show report results that use the test set which are separated into several buckets according to the the number of steps in the procedural planning task. The step number reflects the difficulty of the task. In Table 8 and Table 9 in Appendix **??**how that the performance gain of CLAP over the baselines are more significant in more complicated procedural planning settings. This indicates the superiority of our proposed CLAP in solving long-horizon tasks.

### 3.4 Results on Counterfactual Task Samples

We apply *Initial Configuration*, *Intermediate Step*, *Final Goal* intervention methods on RobotHow and *Intermediate Step* on WikiHow. Counterfactual setting results for human evaluation are summarized in Table 2 and Table 3. CLAP consistently outperforms baselines by a large margin and experiences a much smaller performance drop compared with the powerful baselines when switching to the counterfactual setting. We assume it's due to the biased knowledge of the holdout examples and manual exemplars utilized in the baselines, which are vulnerable to counterfactual samples. Automatic evaluation results on counterfactual RobotHow are summarized in Table 5. Aligned with human evaluations, CLAP achieves the best performance. The overall poor performance in *the Final Goal* category indicates the challenge for long-horizon and composite procedural planning tasks. While the overall better performance in the *Intermediate Step* category benefits from the intermediate guidance.

### 3.5 Correlation between Automatic and Human Evaluation

We evaluate segment-level **Pearson Correlation** between human and automatic metrics. We observe that BERTScore has a moderate correlation to the human coverage score and WMD has a moderate correlation to the human order score, with $23.3\%$ and $32.3\%$ respectively. Similar to the prior

| Dataset | Task | Model | Steps |
|---|---|---|---|
| **RobotHow** | Toast Bread | Chain | Step 1: Walk to kitchen. Step 2: Put bowl on oven. Step 3: Switch off oven. Step 4: Open oven. Step 5: Grab bowl. Step 6: Put bowl in oven. |
| | | LLMaP | Step 1: Put bread in oven. Step 2: Put food bread in oven. Step 3: Put food bread on table. |
| | | CLAP | Step 1: Walk to kitchen counter. Step 2: Pull bread slice. Step 3: Put bread slice on toaster. Step 4: Switch on toaster. Step 5: Look at toaster. Step 6: Turn to toaster. |
| **WikiHow** | How to Take Care of a Pet | Chain | Step 1: Take care of your pets. Step 2: Scroll to the end of the text file. |
| | | LLMaP | Step 1: Make sure your pet always has enough food and water. Step 2: Be sure to feed your dog at least three times a day. . . . Step 4: Give your dog fruits and vegetables as snacks. . . . |
| | | CLAP | Step 1: Research the type of pet, so you can make sure the pet fits into your environment and lifestyle. . . . Step 4: Watch your pet's diet carefully. Step 5: Monitor your pet for illness and injury. . . . |

Table 6: Showcases of procedural steps predicted by different models with GPT2 as the base LLM.

findings (Xu et al., 2021), n-gram based metrics (Sentence-BLEU and ROUGE) have a relatively weaker correlation to the human coverage score, with Pearson correlation of $16.4\%$ and $21.1\%$. Overall, our automatic and human evaluation scores are consistent with the main claim of this paper. However, human evaluation is still irreplaceable for procedural planning at the current stage.

## 4 Related Work

**Procedural Planning** Learning to generate goal-guided sequential language actions is an important task for many applications, including goal-step inference (Lyu et al., 2021; Yang et al., 2021; Zhang et al., 2020b), embodied agentTellex et al. (2011); Jansen (2020); Ahn et al. (2022), and language-aided task adaptation Goyal et al. (2021). Previous work views procedural script learning as a structured form of commonsense knowledge Gupta et al. (2004); Regneri et al. (2010); Wanzare et al. (2016), while more recent work strengthens its association with the changing environments for executable action planning Puig et al. (2018); Shridhar et al. (2020). Lyu et al. (2021) proposes a generative approach using multilingual T5 and a retrieval approach with a step-inference-ordering pipeline. Huang et al. uses prompt engineering to extract language plans from language models. Our method tackles the problem by utilizing commonsense-infused prompts via a neuro-symbolic approach (Mao et al., 2019; Nye et al., 2021; Yi et al., 2018) for zero-shot causal planning.

**Causality for language generation** The integration of causality and machine learning has been an intriguing topic for many problems Pearl (2009); Schölkopf (2022). Previous studies focus on causal inference for natural language understanding Chen et al. (2020); Keith et al. (2020); Wood-Doughty et al. (2018) and generating counterfactual text representations Feder et al. (2021). Weber et al. (2020) proposes an intervention method for script learning. However, their method cannot be directly applied to sequence generation. There are also studies that create counterfactual training samples for dialogue Zhu et al. (2020) and court's view generation Wu et al. (2020). More recently, Hu & Li (2021) shapes the rungs of the causal ladder Pearl (2009) as different objects to optimize. The generated text in our case has a formal structure and our method is based on mediation analysis VanderWeele (2015) and causal intervention Pearl (2009); Peters et al. (2017).

**Prompt for LLM** There is an emerging interest in using prompts to extract knowledge from large language models Chen et al. (2022); Le Scao & Rush (2021); Su et al. (2022); Ye et al. (2022); Zhou et al. (2022). AutoPrompt automatically learns to generate prompts as an alternative to handcrafted probing methods Shin et al. (2020). Cao et al. (2022) treats the prompt as a cause of the task-specific predictor and investigates biases in prompt-based probing evaluations. Chain of thought Wei et al. (2022) discovers that LLM can perform better on reasoning tasks when the prompt is designed as a series of short sentences that mimic the reasoning process of humans.

## 5 Conclusion and Future Work

Procedural planning is a newly emerged research area of great importance to various applications, such as household robots and virtual assistants. We propose a neuro-symbolic Causal Language Planner (CLAP) with commonsense prompts elicited from the external knowledge base to solve the procedural planning problem in a zero-shot manner. Experiments show the effectiveness of our proposed CLAP on both automatic and human evaluation results. Extending causal procedural

planning to handle the long-horizon composite tasks and provide effective automatic evaluation metrics are important directions for future work.

## REFERENCES

Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alex Herzog, et al. Do as i can, not as i say: Grounding language in robotic affordances. *arXiv preprint arXiv:2204.01691*, 2022.

Valts Blukis, Chris Paxton, Dieter Fox, Animesh Garg, and Yoav Artzi. A persistent spatial semantic representation for high-level natural language instruction execution. In *CoRL*, 2021.

Paweł Budzianowski, Tsung-Hsien Wen, Bo-Hsiang Tseng, Iñigo Casanueva, Stefan Ultes, Osman Ramadan, and Milica Gasic. Multiwoz - a large-scale multi-domain wizard-of-oz dataset for task-oriented dialogue modelling. In *EMNLP*, 2018.

Boxi Cao, Hongyu Lin, Xianpei Han, Fangchao Liu, and Le Sun. Can prompt probe pretrained language models? understanding the invisible risks from a causal view. *ACL*, 2022.

Wenqing Chen, Jidong Tian, Liqiang Xiao, Hao He, and Yaohui Jin. Exploring logically dependent multi-task learning with causal inference. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 2213–2225, 2020.

Xiang Chen, Ningyu Zhang, Xin Xie, Shumin Deng, Yunzhi Yao, Chuanqi Tan, Fei Huang, Luo Si, and Huajun Chen. Knowprompt: Knowledge-aware prompt-tuning with synergistic optimization for relation extraction. In *Proceedings of the ACM Web Conference 2022*, pp. 2778–2788, 2022.

Adrianus Dingeman de Groot. Thought and choice in chess. 1978.

Griffin Dietz, James A Landay, and Hyowon Gweon. Building blocks of computational thinking: Young children's developing capacities for problem decomposition. In *CogSci*, pp. 1647–1653, 2019.

Bosheng Ding, Junjie Hu, Lidong Bing, Sharifah Aljunied Mahani, Shafiq R. Joty, Luo Si, and Chunyan Miao. Globalwoz: Globalizing multiwoz to develop multilingual task-oriented dialogue systems. 2022.

Karl Duncker and Lynne S Lees. On problem-solving. *Psychological monographs*, 58(5):i, 1945.

Mihail Eric, Lakshmi Krishnan, Francois Charette, and Christopher D. Manning. Key-value retrieval networks for task-oriented dialogue. In *Proceedings of the 18th Annual SIGdial Meeting on Discourse and Dialogue*, pp. 37–49, Saarbrücken, Germany, August 2017. Association for Computational Linguistics. doi: 10.18653/v1/W17-5506. URL `https://aclanthology.org/W17-5506`.

Amir Feder, Nadav Oved, Uri Shalit, and Roi Reichart. Causalm: Causal model explanation through counterfactual language models. *Computational Linguistics*, 47(2):333–386, 2021.

Fernand Gobet, Peter C.R. Lane, Steve Croker, Peter C-H. Cheng, Gary Jones, Iain Oliver, and Julian M. Pine. Chunking mechanisms in human learning. *Trends in Cognitive Sciences*, 5(6):236–243, 2001. ISSN 1364-6613. doi: https://doi.org/10.1016/S1364-6613(00)01662-4. URL `https://www.sciencedirect.com/science/article/pii/S1364661300016624`.

Prasoon Goyal, Raymond J Mooney, and Scott Niekum. Zero-shot task adaptation using natural language. *arXiv preprint arXiv:2106.02972*, 2021.

Rakesh Gupta, Mykel J Kochenderfer, Deborah Mcguinness, and George Ferguson. Common sense data acquisition for indoor mobile robots. In *AAAI*, pp. 605–610, 2004.

Zhiting Hu and Li Erran Li. A causal lens for controllable text generation. *Advances in Neural Information Processing Systems*, 34, 2021.

Wenlong Huang, Pieter Abbeel, Deepak Pathak, and Igor Mordatch. Language models as zero-shot planners: Extracting actionable knowledge for embodied agents gpt-2 1.5b. URL https://huangwl18.github.io/language-planner.

Vladimir Ilievski, Claudiu Cristian Musat, Andreea Hossmann, and Michael Baeriswyl. Goal-oriented chatbot dialog management bootstrapping with transfer learning. In *IJCAI*, 2018.

Peter Jansen. Visually-grounded planning without vision: Language models infer detailed plans from high-level instructions. In *Findings of the Association for Computational Linguistics: EMNLP 2020*, pp. 4412–4417, Online, November 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.findings-emnlp.395. URL https://aclanthology.org/2020.findings-emnlp.395.

Katherine Keith, David Jensen, and Brendan O'Connor. Text and causal inference: A review of using text to remove confounding from causal estimates. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 5332–5344, 2020.

Giinther Knoblich, Stellan Ohlsson, Hilde Haider, and Detlef Rhenius. Constraint relaxation and chunk decomposition in insight problem solving. *Journal of Experimental Psychology: Learning, Memory and Cognition*, 25:1534–1555, 1999.

Mahnaz Koupaee and William Yang Wang. Wikihow: A large scale text summarization dataset. *ArXiv*, abs/1810.09305, 2018.

Matt Kusner, Yu Sun, Nicholas Kolkin, and Kilian Weinberger. From word embeddings to document distances. In Francis Bach and David Blei (eds.), *Proceedings of the 32nd International Conference on Machine Learning*, volume 37 of *Proceedings of Machine Learning Research*, pp. 957–966, Lille, France, 07–09 Jul 2015. PMLR. URL https://proceedings.mlr.press/v37/kusnerb15.html.

Teven Le Scao and Alexander M Rush. How many data points is a prompt worth? In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pp. 2627–2636, 2021.

Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pp. 7871–7880, Online, July 2020. Association for Computational Linguistics. doi: 10.18653/v1/2020.acl-main.703. URL https://aclanthology.org/2020.acl-main.703.

Chin-Yew Lin. ROUGE: A package for automatic evaluation of summaries. In *Text Summarization Branches Out*, pp. 74–81, Barcelona, Spain, July 2004. Association for Computational Linguistics. URL https://aclanthology.org/W04-1013.

Qing Lyu, Li Zhang, and Chris Callison-Burch. Goal-oriented script construction. In *Proceedings of the 14th International Conference on Natural Language Generation*, pp. 184–200, 2021.

Jiayuan Mao, Chuang Gan, Pushmeet Kohli, Joshua B. Tenenbaum, and Jiajun Wu. The Neuro-Symbolic Concept Learner: Interpreting Scenes, Words, and Sentences From Natural Supervision. In *International Conference on Learning Representations*, 2019. URL https://openreview.net/forum?id=rJgMlhRctm.

George A. Miller. The magical number seven plus or minus two: some limits on our capacity for processing information. *Psychological review*, 63 2:81–97, 1956.

So Yeon Min, Devendra Singh Chaplot, Pradeep Ravikumar, Yonatan Bisk, and Ruslan Salakhutdinov. Film: Following instructions in language with modular methods. 2022.

Maxwell Nye, Michael Tessler, Josh Tenenbaum, and Brenden M Lake. Improving coherence and consistency in neural sequence models with dual-system, neuro-symbolic reasoning. *Advances in Neural Information Processing Systems*, 34, 2021.

Aishwarya Padmakumar, Jesse Thomason, Ayush Shrivastava, Patrick Lange, Anjali Narayan-Chen, Spandana Gella, Robinson Piramuthu, Gokhan Tur, and Dilek Hakkani-Tür. Teach: Task-driven embodied agents that chat. In *AAAI 2022*, 2022. URL https://www.amazon.science/publications/teach-task-driven-embodied-agents-that-chat.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics*, pp. 311–318, Philadelphia, Pennsylvania, USA, July 2002. Association for Computational Linguistics. doi: 10.3115/1073083.1073135. URL https://aclanthology.org/P02-1040.

Judea Pearl. Causal diagrams for empirical research. *Biometrika*, 82(4):669–688, 1995.

Judea Pearl. *Causality*. Cambridge university press, 2009.

Jonas Peters, Dominik Janzing, and Bernhard Schölkopf. *Elements of causal inference: foundations and learning algorithms*. The MIT Press, 2017.

Xavier Puig, Kevin Ra, Marko Boben, Jiaman Li, Tingwu Wang, Sanja Fidler, and Antonio Torralba. Virtualhome: Simulating household activities via programs. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pp. 8494–8502, 2018.

Alec Radford, Jeffrey Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. URL https://github.com/codelucas/newspaper.

Michaela Regneri, Alexander Koller, and Manfred Pinkal. Learning script knowledge with web experiments. In *Proceedings of the 48th Annual Meeting of the Association for Computational Linguistics*, pp. 979–988, 2010.

Nils Reimers and Iryna Gurevych. Sentence-BERT: Sentence embeddings using Siamese BERT-networks. 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)*, pp. 3982–3992, Hong Kong, China, November 2019. Association for Computational Linguistics. doi: 10.18653/v1/D19-1410. URL https://aclanthology.org/D19-1410.

Bernhard Schölkopf. Causality for machine learning. In *Probabilistic and Causal Inference: The Works of Judea Pearl*, pp. 765–804. 2022.

Taylor Shin, Yasaman Razeghi, Robert L Logan IV, Eric Wallace, and Sameer Singh. Autoprompt: Eliciting knowledge from language models with automatically generated prompts. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4222–4235, 2020.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. *2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pp. 10737–10746, 2020.

Steven M. Smith. Getting into and out of mental ruts: A theory of fixation, incubation, and insight. 1995.

Robyn Speer, Joshua Chin, and Catherine Havasi. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the Thirty-First AAAI Conference on Artificial Intelligence*, AAAI'17, pp. 4444–4451. AAAI Press, 2017.

Yusheng Su, Xiaozhi Wang, Yujia Qin, Chi-Min Chan, Yankai Lin, Huadong Wang, Kaiyue Wen, Zhiyuan Liu, Peng Li, Juanzi Li, Lei Hou, Maosong Sun, and Jie Zhou. On transferability of prompt tuning for natural language processing. In *Annual Conference of the North American Chapter of the Association for Computational Linguistics (NAACL)*, 2022.

Stefanie Tellex, Thomas Kollar, Steven Dickerson, Matthew Walter, Ashis Banerjee, Seth Teller, and Nicholas Roy. Understanding natural language commands for robotic navigation and mobile manipulation. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 25, pp. 1507–1514, 2011.

Tyler VanderWeele. *Explanation in causal inference: methods for mediation and interaction*. Oxford University Press, 2015.

Lilian DA Wanzare, Alessandra Zarcone, Stefan Thater, and Manfred Pinkal. A crowdsourced database of event sequence descriptions for the acquisition of high-quality script knowledge. In *Proceedings of the Tenth International Conference on Language Resources and Evaluation (LREC'16)*, pp. 3494–3501, 2016.

Noah Weber, Rachel Rudinger, and Benjamin Van Durme. Causal inference of script knowledge. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 7583–7596, 2020.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Ed Chi, Quoc Le, and Denny Zhou. Chain of thought prompting elicits reasoning in large language models. 1 2022. URL `http://arxiv.org/abs/2201.11903`.

Zach Wood-Doughty, Ilya Shpitser, and Mark Dredze. Challenges of using text classifiers for causal inference. In *Proceedings of the Conference on Empirical Methods in Natural Language Processing. Conference on Empirical Methods in Natural Language Processing*, volume 2018, pp. 4586. NIH Public Access, 2018.

Te-Lin Wu, Alexander Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph M. Weischedel, and Nanyun Peng. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals. In *ACL*, 2022.

Yiquan Wu, Kun Kuang, Yating Zhang, Xiaozhong Liu, Changlong Sun, Jun Xiao, Yueting Zhuang, Luo Si, and Fei Wu. De-biased court's view generation with causality. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 763–780, 2020.

Wenda Xu, Michael Saxon, Misha Sra, and William Yang Wang. Self-supervised knowledge assimilation for expert-layman text style transfer. In *AAAI*, 2021.

Shiquan Yang, Rui Zhang, Sarah Monazam Erfani, and Jey Han Lau. An interpretable neuro-symbolic reasoning framework for task-oriented dialogue generation. 2022.

Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. Visual goal-step inference using wikihow. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pp. 2167–2179, 2021.

Hongbin Ye, Ningyu Zhang, Shumin Deng, Xiang Chen, Hui Chen, Feiyu Xiong, Xi Chen, and Huajun Chen. Ontology-enhanced prompt-tuning for few-shot learning. *WWW*, 2022.

Kexin Yi, Jiajun Wu, Chuang Gan, Antonio Torralba, Pushmeet Kohli, and Joshua B. Tenenbaum. Neural-symbolic vqa: Disentangling reasoning from vision and language understanding. In *NeurIPS*, 2018.

Zhongqi Yue, Hanwang Zhang, Qianru Sun, and Xian-Sheng Hua. Interventional few-shot learning. *Advances in neural information processing systems*, 33:2734–2746, 2020.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Intent detection with wikihow. In *Proceedings of the 1st Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics and the 10th International Joint Conference on Natural Language Processing*, pp. 328–333, 2020a.

Li Zhang, Qing Lyu, and Chris Callison-Burch. Reasoning about goals, steps, and temporal ordering with wikihow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 4630–4639, 2020b.

Tianyi Zhang*, Varsha Kishore*, Felix Wu*, Kilian Q. Weinberger, and Yoav Artzi. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*, 2020. URL https://openreview.net/forum?id=SkeHuCVFDr.

Zhonglu Zhang, Yu Luo, Chaolun Wang, Christopher M. Warren, Qianfeng Xia, Qiang Xing, Bihua Cao, Yi Lei, and Hong Li. Identification and transformation difficulty in problem solving: Electrophysiological evidence from chunk decomposition. *Biological Psychology*, 143:10–21, 2019.

Kaiyang Zhou, Jingkang Yang, Chen Change Loy, and Ziwei Liu. Conditional prompt learning for vision-language models. In *CVPR*, 2022.

Qingfu Zhu, Weinan Zhang, Ting Liu, and William Yang Wang. Counterfactual off-policy training for neural dialogue generation. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pp. 3438–3448, 2020.

# A SCM THEORETICAL DETAILS

## A.1 CAUSAL PRELIMINARIES

The Structural Causal Model (SCM) is a directed acyclic graph (DAG) to describe the causal relationships within a system Pearl (2009). In this paper, we refer to the unrolled SCM along the time dimension as the full temporal causal graph, while the rolled-up version is also called the causal summary graph Peters et al. (2017). In an SCM, if the variable $D$ is a cause of both $T$ and $S_i$, then it is called a **confounder**. A confounder opens up a backdoor path and causes a spurious correlation between $T$ and $S_i$. The **backdoor path** is defined as the remaining path between $T$ and $S_i$ when all the arrows pointing out of $T$ are removed. Therefore, $T \leftarrow D \rightarrow S_i$ is a backdoor path. For our SCM with mediator $P_i$ shown in Figure 3c (same as Figure 2b) from the main paper, there is no backdoor path between $T$ and $\{P_i, S_{i-1}\}$ because only $D \rightarrow T$ is left after removing outgoing arrows of $T$. On the other hand, there is a backdoor path between $P_i$ and $S_i$, i.e. $P_i \leftarrow T \leftarrow D \rightarrow S_i$ so that $P_i$ indirectly affect the observation of $S_i$ through $\{T, S_{i-1}\}$ and $D$.

To identify the true causal effect between $X$ and $Y$, we aim to estimate the conditional $P(Y|do(X))$ after intervention with the *do*-operator. The *do*-operator is to break the backdoor path by setting $X$ to a fixed value independent of $Z$. Then the path $Z \rightarrow X$ can be removed to eliminate the backdoor paths. In practice, the backdoor adjustment and frontdoor adjustment are two fundamental methods to implement interventions and obtain the conditional $P(Y|do(X))$.

**Causal Intervention** There are two types of operation to control the confounding bias: the *backdoor adjustment* and the *frontdoor adjustment* (Pearl, 2009). The backdoor adjustment is intractable in our case because it requires the prior distribution of the confounding variables. On the other hand, we can construct an input prompt as a mediator $P_i$ for $T \rightarrow S_i$ and $S_{i-1} \rightarrow S_i$. Then the frontdoor adjustment applies a two-step *do*-operation to mitigate bias by investigating $P \rightarrow S_i$ (Pearl, 2009). Specifically, we construct the prompt mediator $P_i$ using techniques illustrated in Section 2.2.

## A.2 THE BACKDOOR ADJUSTMENT

The backdoor adjustment is one way to realize the intervention $do(T = t)$ by considering the conditional probability over the existing data distribution with observed confounder $D$. It calculates the average causal effects by considering all stratums of the dataset:

$$P(S_i|do(T)) = \sum_d P(S_i|T, D = d)P(D = d) \tag{6}$$

However, for LLMs, the pretraining data is usually unobservable and has been transformed as knowledge incorporated into the hidden space. Therefore, we are not able to directly apply the backdoor adjustment.

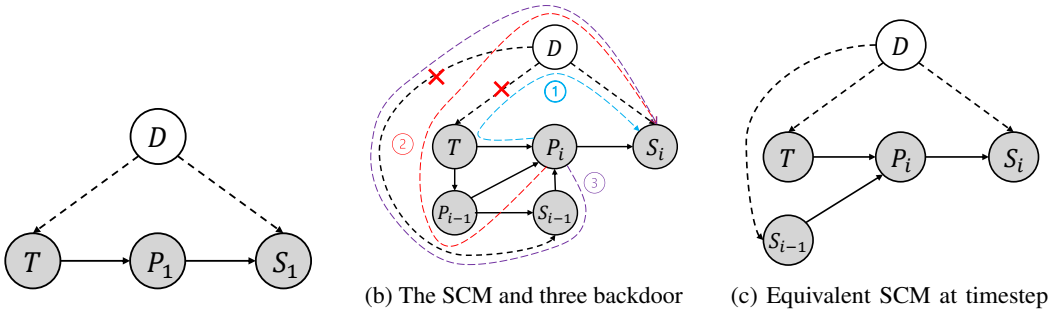
## A.3 THE FRONTDOOR ADJUSTMENT

The frontdoor adjustment is another technique to apply intervention by introducing a mediator $P_i$ when the confounder is unobservable. As is explained in Section 2.2 from the main paper, the frontdoor adjustment is equivalent to two consecutive *do*-operation on task $T$ and prompt $P_i$. We first investigate the generation of $S_1$ and then expand it to $S_t$.

**Timestep** $i = 1$ As is shown in Figure 3a, since there is no preceding steps, the first step generation involves $D$, $T$ and $P_1$ only. Similar to the proof in Section 2.2 from the main paper, we have:

$$\begin{aligned} P(S_1|do(T)) &= \sum_p P(S_1|do(P_1 = p))P(p|do(T)) \\ &= \sum_p P(p|T) \sum_t P(S_i|p, T = t)P(T = t) \end{aligned} \tag{7}$$

By adding intervention to $T$, we make the value of $do(T = t)$ independent of the confounder $D$ at the beginning. The backdoor path through $D \rightarrow T$ is eliminated as a result.




(a) SCM at timestep $i = 1$

(b) The SCM and three backdoor paths at timestep $i > 1$

(c) Equivalent SCM at timestep $i > 1$ after eliminating $P_{i-1}$

Figure 3: **The Frontdoor Adjustment for Causal Language Planner.** (a) the structural causal model at timestamp $i = 1$. $T$ denotes the task name and $S_1$ denotes the step at timestep 1. $D$ is the unobservable confounding variable introduced by the pre-training data. $P_1$ denotes the mediating variables we construct to mitigate the spurious correlation at timestep 1. (b) $D$ opens up backdoor paths for $T \to S_i$, $P_{i-1} \to S_i$ and $S_{i-1} \to S_i$ which can be blocked by introducing $P_i$. path 1 and path 2 share the same path $D \to T$. Intervention on $T$ blocks $D \to T$ and the backdoor path 2. Intervention on $S_{i-1}$ blocks $D \to S_{i-1}$ and the backdoor path 3. (c) the structural causal model at timestamp $i > 1$ after simplification based on Equation 13-17.

**Timestep** $i > 1$ As is shown in Figure 2a from the main paper, we model the mediator $P_1$ as an effect of three variables, $T$, $P_{i-1}$ and $S_{i-1}$. The first step of our frontdoor adjustment is to apply the *do*-operator on the three variables and observe the change in $P_i$ as explained in Section 2.2 from the main paper. Since there are no backdoor paths between $P_i$ and these variables, we have the probability after intervention equal to the conditional probability without intervention:

$$P(P_i = p|do(T)) = P(P_i = p|T) \tag{8}$$
$$P(P_i = p|do(P_{i-1})) = P(P_i = p|P_{i-1}) \tag{9}$$
$$P(P_i = p|do(S_{i-1})) = P(P_i = p|S_{i-1}) \tag{10}$$

The second step is to apply *do*-operator on $P_i$ and then identify the causal effect as:

$$P(S_i|do(P_i)) = \sum_{t,p',s} \Big( P(S_i|P_i, T = t, P_{i-1} = p', S_{i-1} = s) \\ P(T = t, P_{i-1} = p', S_{i-1} = s) \Big) \tag{11}$$

Combining Equation8-10 and Equation 11, we have the frontdoor adjustment. Note that there are three backdoor paths from each of the variables $T$, $P_{i-1}$, and $S_{i-1}$, as is shown in Figure 3b (drawn in blue, red and purple). More importantly, the one through $T$, i.e. $P_i \leftarrow T \leftarrow D \to S_i$ (the blue path in Figure 3b) and the one through $P_{i-1}$, i.e. $P_i \leftarrow P_{i-1} \leftarrow T \leftarrow D \to S_i$ (the red path in Figure 3b) shares the same subpath. The intervention on the task $T$ breaks the backdoor paths for both $T$ and $P_{i-1}$. Therefore, we have our frontdoor adjustment as

$$P(S_i|do(S_{i-1}),do(P_{i-1}), do(T)) \tag{12}$$
$$= \sum_p P(S_i|do(P_i = p))P(p|do(S_{i-1}), do(P_{i-1}), do(T)) \tag{13}$$
$$= \sum_p P(S_i|do(P_i = p))P(p|do(S_{i-1}), P_{i-1}, do(T)) \tag{14}$$
$$= \sum_p P(S_i|do(P_i = p))P(p|do(S_{i-1}), do(T)) \tag{15}$$
$$= \sum_p P(p|S_{i-1}, T) \sum_{s,t} P(S_i|p, S_{i-1} = s, T = t)P(S_{i-1} = s, T = t) \tag{16}$$
$$= P(S_i|do(S_{i-1}), do(T)) \tag{17}$$

We have Equation 14 because of the intervention on $T$ and **Rule 2** (Pearl, 1995), Equation 15 because of **Rule 1** (Pearl, 1995). We end up with Equation 15-17 which is shown as Equation 3 in Section 2.2 from the main paper.

# B IMPLEMENTATION DETAILS

## B.1 ORIGINAL DATASET DETAILS

**RobotHow** This dataset is Attribution-NonCommercial-ShareAlike 4.0 International Creative Commons License. We evaluate the inference of 150 tasks by random selection from the dataset. Each program contains the task name, task description and steps. We use the task name and sequence of steps as our input and output references.Each step is a composition of `[Action]`, `[Object]` and `[Number]`. For example, the sequence of steps of the task "Watch TV" are: 1. [Walk] ¡TELEVISION¿ (1) 2. [SwitchOn] ¡TELEVISION¿ (1) 3. [Walk] ¡SOFA¿ (1) 4. [Sit] ¡SOFA¿ (1) 5. [Watch] ¡TELEVISION¿ (1).

**WikiHow** This dataset is under an Attribution-Noncommercial-Share Alike 3.0 Creative Commons License. And the text content is free to modify, republish and share. We evaluate the inference of 1000 tasks by random selection from the dataset. The admissible action space and interaction object space are more complex than the programs in RobotHow. And there is no fixed "[Action] ¡Object¿ (Number)" form of each step. For each article, it contains the title, the bold headlines and text. We utilize the title and headlines as our task name and steps respectively.

## B.2 COUNTERFACTUAL DATASET AND EXPERIMENT DETAILS

| Category | Original Program | Counterfactual Program |
|---|---|---|
| Initial Configuration | Task: Watch TV<br>Step 1: Find remote control.Step 2: Grab remote control.<br>Step 3: Find television.Step 4: Switch on television.<br>Step 5: Turn to television.Step 6: Watch television.<br>Step 7: Switch off television.<br>Step 8: Put back remote control | Task: Watch TV <u>in bedroom</u><br>Step 1: Walk to bedroom Step 2: Find remote control.<br>Step 3: Grab remote control.Step 4: Find television.<br>Step 5: Switch on television.Step 6: Turn to television.<br>Step 7: Watch television.Step 8: Switch off television.<br>Step 9: Put back remote control |
| Intermediate Step | Task: Work<br>Step 1: Walk to home office.Step 2: Walk to chair.<br>Step 3: Find chair.Step 4: Sit on chair.<br>Step 5: Find computer.Step 6: Switch on computer.<br>Step 7: Turn to computer.Step 8: Look at computer | Task: Work <u>(Find Computer)</u><br>Step 1: Walk to home office.Step 2: Walk to chair.<br>Step 3: Find chair.Step 4: Sit on chair.<br>Step 5: Find computer.Step 6: Switch on computer.<br>Step 7: Turn to computer.Step 8: Look at computer |
| Final Goal | Task1: Turn light off<br>Step 1: Walk to bedroom Step 2: Walk to light<br>Step 3: Switch off light<br>Task2: Clean<br>Step 1: Walk to home office Step 2: Walk to rag<br>Step 3: Find rag Step 4: Grab rag<br>Step 5: Walk to desk Step 6: Find computer<br>Step 7: Wipe computer Step 8: Wipe desk<br>Step 9: Put back rag | Task: Turn light off <u>and Clean</u><br>Step 1: Walk to bedroom Step 2: Walk to light<br>Step 3: Switch off light<br>Step 4: Walk to home office<br>Step 5: Walk to rag Step 6: Find rag<br>Step 7: Grab rag Step 8: Walk to desk<br>Step 9: Find computer Step 10: Wipe computer<br>Step 11: Wipe desk Step 12: Put back rag |

Table 7: **Counterfactual Language Planning.** Three types of methods, including initial configuration, intermediate step, and final goal are applied to intervene the original procedural data.

Table 7 show the examples that compare the original program and the counterfactual program of each intervention method are also provided. Specifically, for **Initial Configuration**, we randomly append the location to a given task name to constrain the location of completing the task. The steps are prepended with the initial step "walk to ¡Location¿". For **Intermediate Step**, we randomly sampled a step from the task-specific program and append it to the task name to constrain the way to implement a given task. For **Final Goal**, we randomly combine two tasks by combining both the task names and the programs to construct a set of long-horizon composite tasks.

We conduct counterfactual experiments by applying randomly selected intervention methods over RobotHow. And we only apply the Intermediate Step intervention method over WikiHow due to the loose configuration requirement and the long text of the WikiHow contents. Note that the performance gain of CLAP under the counterfactual setting mainly comes from the additional guidance of the task introduced from the Intermediate Step intervention method. However, the baselines mostly experience

performance drops due to the limited annotated exemplars. CLAP consistently outperforms baselines by a large margin, indicating its superiority under the counterfactual setting.

### B.3 METHOD DETAILS

Conceptnet5 define a set of 34 relations ([2]). Within the relations we consider in the language planning task, the averaged sampling time of subgraph sampling is 0.03576 milliseconds per task program.

To ensure that the generated language plans could ground to the specific task domain (e.g. RobotHow, WikiHow), we should avoid to produce the steps that are inadmissible (e.g. Toast the table). In other words, the generated steps should be fully constrained to admissible ones. Thus previous works ( (Huang et al.; Ahn et al., 2022)) have explored to use the model (which is $LM_T$ in our case) to score a step selected from a fixed set of available options, instead of directly sampling from the output distributions of the language model (which is $LM_G$ in our case). Specifically, the Admissible Step Translation is conducted by mapping the generated step by $LM_G$ into admissible step by matching based on the similarity in the Sentence-Transformer (Reimers & Gurevych, 2019) ($LM_T$) embedding space, following the baseline LLMaP( (Huang et al.)).

### B.4 HYPERPARAMETER SEARCH

The confidence threshold $\theta$, which terminate the generation when below it, is searched in $\{0, 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8\}$. The steps horizon, which constrains the maximal number of language planning steps, is searched in $\{10, 20, 40\}$. The number of hops for retrieving the subgraph from the external knowledge base is searched in $\{1, 2, 3\}$. The ratio of maximal concepts to the length of the task name is searched in $\{1, 2, 3\}$. The cosine similarity threshold for keeping the task-specific concept is searched in $\{0.4, 0.6, 0.8\}$. The edge weight threshold $\theta_e$ is searched in $\{0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8\}$. The top-$k$ task-specific nodes value is searched in $\{1, 5, 10, 15, 20, 25, 50, 100\}$.

The configurations used in the experiments are: $\theta$=0.7, 20 step horizon, 3 hops, 3 ratio of concepts to task length, cosine similarity threshold $0.4$, $\theta_e$=0.6 and $k$=10.

### B.5 COMPUTATION AND RESOURCES

We use one single NVIDIA A100 GPU Server for all the experiments. Since there is no training in our zero-shot settings, the computation is only used for the inference stage of the experiments. Note that for the BART language model, we use BART-large version. And we use the 1.5 billion parameter GPT-2 (aka gpt2-xl). For the translation model $LM_T$, we use sentence-transformers (RoBERTa-large). All these models are released by HuggingFace.

## C EVALUATION DETAILS

### C.1 CROWDSOURCING HUMAN EVALUATION

We conduct all the human evaluations (rating and win-lose comparison) on Amazon Mechanical Turk platform. We ask Amazon Mechanical Turk workers, for every assignment, to evaluate the quality of the provided low-level steps given the high-level task description. For the **Win-Lose Comparison**, they were asked to choose one from the two provided model generated results by *1:the first one is better*, *2:equal* and *3:the second one is better*. For the **Human Ratings**, they were asked to score each sample with 5-point Likert scale. This process does not involve collecting any personal information. And we manually check no offensive content is produced by the models.

The assignment layout templates for workers are shown in Figure 5 and Figure 4. Specifically, we evaluate randomly selected 50 task examples from each dataset (RobotHow and WikiHow) under all the settings (standard and counterfactual). We only collect the examples that the workers read the instructions carefully by checking whether they give 1 score for the empty program as a sanity check. The hourly wage paid to participants is estimated $9. And the total amount spent on participant

[2] https://github.com/commonsense/conceptnet5/wiki/Relations

compensation is $1296. The details of the Human Intelligence Tasks process are described in the following sections.

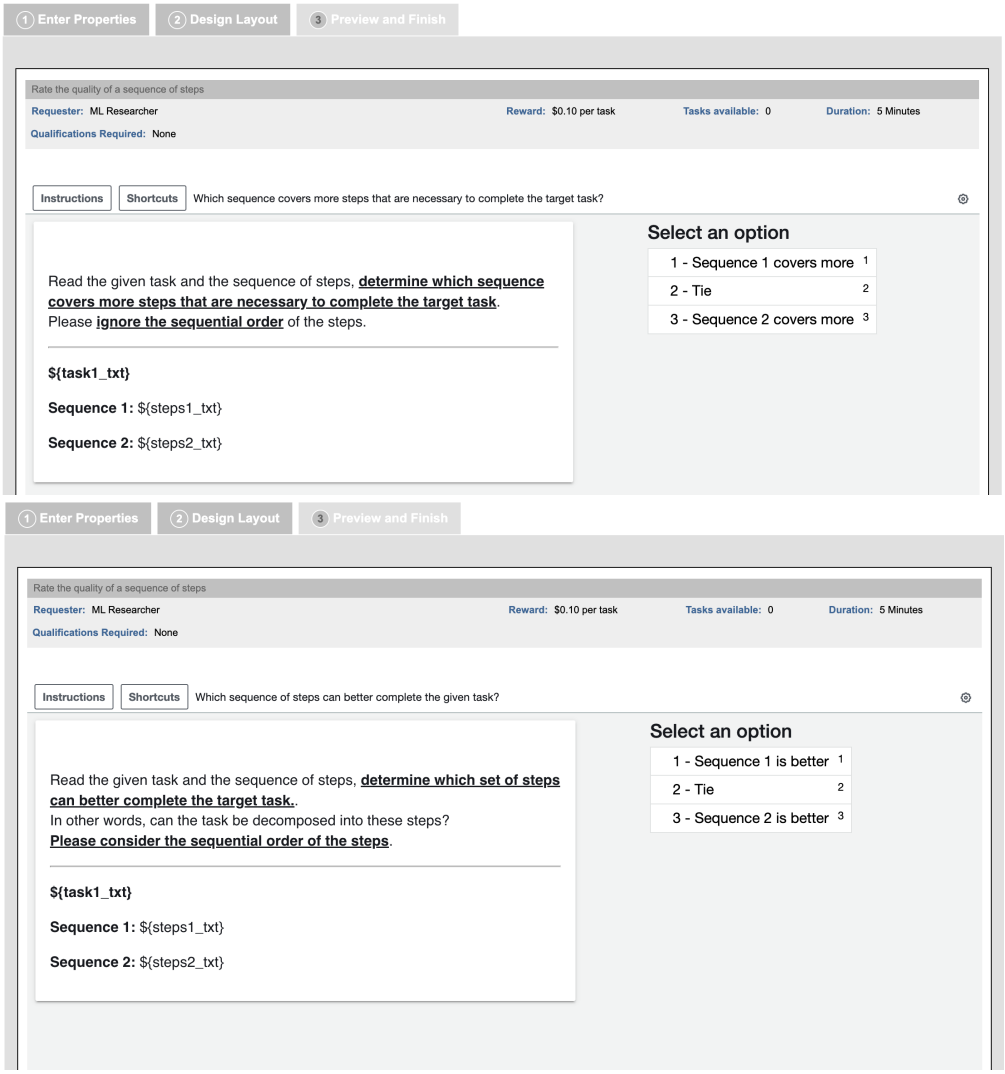



Figure 4: Amazon Mechanical Turk Platform. Questions Layout for Human Raters for Win-Tie-Lose Comparison.

### C.1.1 WIN-LOSE COMPARISON

During the process of Human Intelligence Tasks, the workers are shown the following instructions: *Read the given task and the sequence of steps, determine which set of steps can better complete the target task. In other words, can the task be decomposed into these steps? Please consider the sequential order of the steps.*

Then the program to be evaluated is provided as:

**Question** Task: Study

**Sequence 1:**: Step 1: Walk to textbook Step 2: Read book Step 3: Walk to book

**Sequence 2:**: Step 1: Walk to home office Step 2: Find desk

Finally, the workers are asked to score the program by following the instructions below: *Select an option: 1 - Sequence 1 is better; 2 - Tie; 3 - Sequence 2 is better*

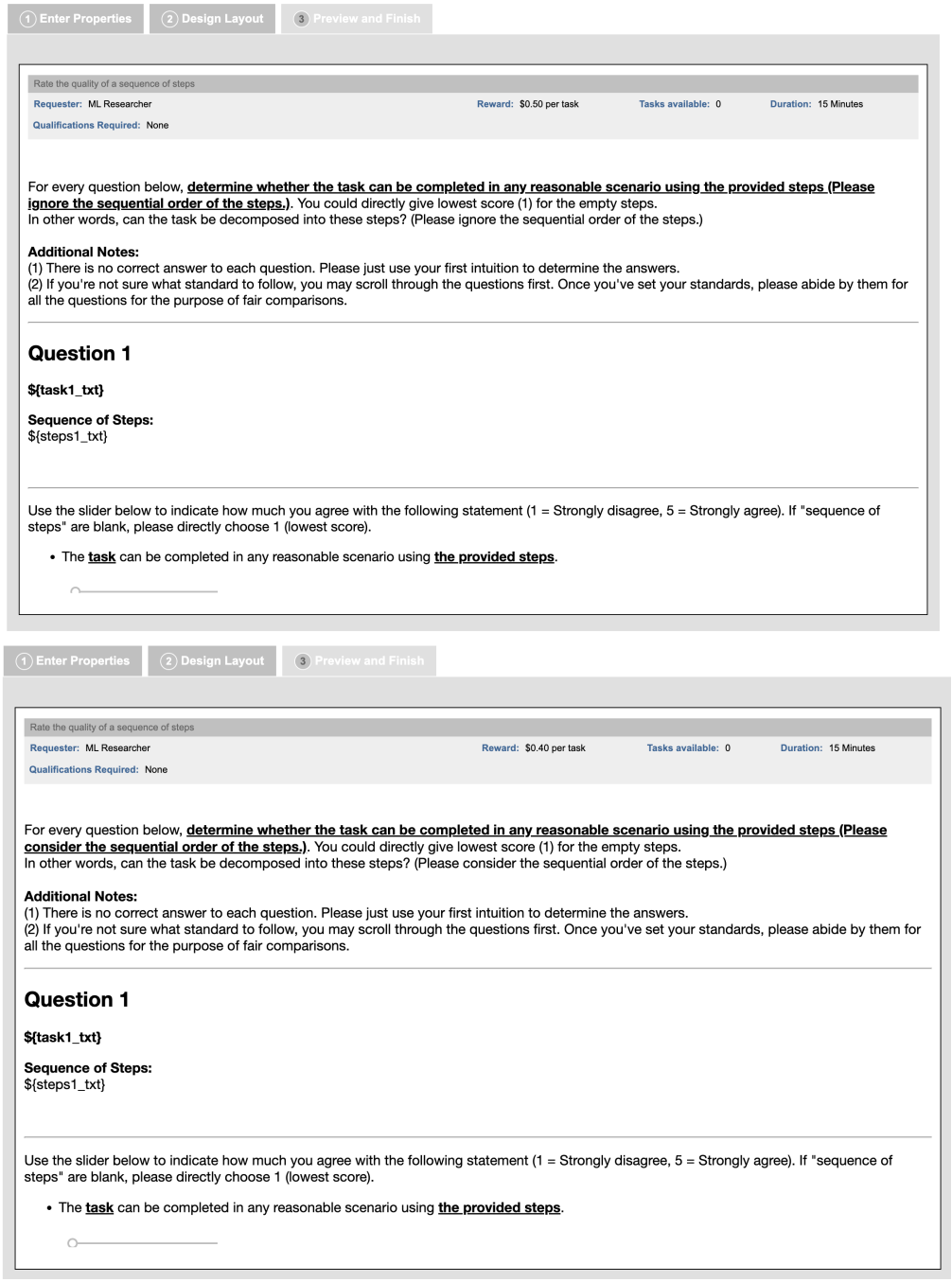



Figure 5: Amazon Mechanical Turk Platform. Questions Layout for Human Raters for 5 Point Likert Scale.

The above example is to evaluate the order metric, for the coverage metric, the same process are conducted, except for the instructions are: *Read the given task and the sequence of steps, and determine which sequence covers more steps that are necessary to complete the target task. Please ignore the sequential order of the steps.*

#### C.1.2 HUMAN RATINGS

Similar as the Win-Lose Comparison Human Intelligence Tasks, the workers are shown the following instructions: *For every question below, determine whether the task can be completed in any reasonable scenario using the provided steps (Please consider the sequential order of the steps.). You could directly give the lowest score (1) for the empty steps. In other words, can the task be decomposed into these steps? (Please consider the sequential order of the steps.)*

Then the program to be evaluated is provided as:

**Question** Task: Write an email

**Sequence of Steps**: Step 1: Walk to home office Step 2: Walk to computer Step 3: Find computer Step 4: Turn to computer Step 5: Look at computer Step 6: Walk to computer Step 7: Find chair Step 8: Sit on chair Step 9: Find keyboard Step 10: Grab keyboard Step 11: Find mouse Step 12: Grab mouse Step 13: Type on keyboard

Finally, the workers are asked to score the program by following the instructions below: *Use the slider below to indicate how much you agree with the following statement (1 = Strongly disagree, 5 = Strongly agree). If "sequence of steps" are blank, please directly choose 1 (lowest score). The task can be completed in any reasonable scenario using the provided steps.* `[SLIDER PROVIDED HERE]`

The above example is to evaluate the order metric, for the coverage metric, the same process is conducted, except for the instructions are: *For every question below, determine whether the task can be completed in any reasonable scenario using the provided steps (Please ignore the sequential order of the steps.). You could directly give the lowest score (1) for the empty steps. In other words, can the task be decomposed into these steps? (Please ignore the sequential order of the steps.)*

### C.2 MORE RESULTS

| **Model** | **RobotHow** | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Step Bucket | S-BLEU | WMD | BERT-f1 | ROUGE-f1 | Coverage | Order | Step Avg. | Time Cost (ms) |
| BART + Chain (Wei et al., 2022) | (0,10] | 0.073 | 0.915 | 0.863 | 0.432 | 2.947 | 2.760 | 6.600 | 3.330 |
| | (10,20] | 0.049 | 0.909 | 0.857 | 0.442 | 2.921 | 2.825 | 13.714 | 3.820 |
| BART + LLMaP (Huang et al.) | (0,10] | 0.076 | 0.941 | 0.867 | 0.450 | 2.973 | 2.760 | 6.600 | 3.298 |
| | (10,20] | 0.028 | 0.931 | 0.853 | 0.393 | 3.095 | 2.889 | 13.714 | 3.866 |
| BART + CLAP | (0,10] | 0.099 | 0.955 | 0.894 | 0.532 | 3.187 | 3.013 | 6.600 | 3.272 |
| | (10,20] | 0.041 | 0.935 | 0.869 | 0.437 | 3.079 | 3.206 | 13.714 | 4.022 |
| GPT2 + Chain (Wei et al., 2022) | (0,10] | 0.076 | 0.891 | 0.856 | 0.395 | 2.453 | 2.147 | 6.600 | 3.370 |
| | (10,20] | 0.057 | 0.877 | 0.845 | 0.359 | 2.365 | 2.476 | 13.714 | 3.804 |
| GPT2 + LLMaP (Huang et al.) | (0,10] | 0.112 | 0.942 | 0.894 | 0.486 | 3.147 | 2.987 | 6.600 | 3.212 |
| | (10,20] | 0.064 | 0.906 | 0.859 | 0.394 | 2.921 | 2.73 | 13.714 | 3.875 |
| GPT2 + CLAP | (0,10] | 0.167 | 0.94 | 0.901 | 0.554 | 3.173 | 2.813 | 6.600 | 3.344 |
| | (10,20] | 0.101 | 0.924 | 0.882 | 0.480 | 2.984 | 3.063 | 13.714 | 3.954 |

Table 8: Evaluation results on the Original RobotHow by separating test set into several Step Bucket.

**Significance Test** We provide paired-t test (p¡$0.05$) statistics results for Table 3. On RobotHow, our CLAP significantly outperforms all baselines on Original-Order(BART) and Counterfactual-Coverage(GPT2). On WikiHow, our CLAP significantly outperforms all baselines on Original-Coverage(BART, GPT2), Counterfactual-Coverage(BART, GPT2), and Counterfactual-Order(BART). For the coverage metric under the counterfactual setting, the human-provided program is not significantly better than our CLAP.

**Results on GPT-3** In addition, we conduct experiments with GPT-3 (davinci version) using OpenAI API. We showcase the comparison in Table 10.

| Model | WikiHow | | | | | | | | |
|---|---|---|---|---|---|---|---|---|---|
| | Step Bucket | S-BLEU | WMD | BERT-f1 | ROUGE-f1 | Coverage | Order | Step Avg. | Time Cost (ms) |
| BART + Chain (Wei et al., 2022) | (0,10] | 0.053 | 0.919 | 0.789 | 0.356 | 2.969 | 3.521 | 6.156 | 8.233 |
| | (10,20] | 0.032 | 0.921 | 0.784 | 0.294 | 2.644 | 3.311 | 14.467 | 7.349 |
| | (20,30] | 0.025 | 0.902 | 0.769 | 0.206 | 3.333 | 3.667 | 25.000 | 4.579 |
| BART + LLMaP (Huang et al.) | (0,10] | 0.068 | 0.934 | 0.814 | 0.353 | 2.802 | 3.438 | 6.156 | 8.289 |
| | (10,20] | 0.032 | 0.924 | 0.794 | 0.293 | 2.600 | 3.178 | 14.467 | 7.487 |
| | (20,30] | 0.017 | 0.897 | 0.778 | 0.238 | 4.000 | 4.333 | 25.000 | 4.919 |
| BART + CLAP | (0,10] | 0.108 | 0.939 | 0.834 | 0.431 | 3.083 | 3.594 | 6.156 | 8.341 |
| | (10,20] | 0.059 | 0.927 | 0.812 | 0.372 | 2.978 | 3.244 | 14.467 | 7.829 |
| | (20,30] | 0.011 | 0.923 | 0.8 | 0.228 | 3.333 | 3.667 | 25.000 | 4.961 |
| GPT2 + Chain (Wei et al., 2022) | (0,10] | 0.107 | 0.928 | 0.817 | 0.353 | 3.031 | 3.438 | 6.156 | 8.367 |
| | (10,20] | 0.077 | 0.933 | 0.812 | 0.328 | 2.733 | 3.422 | 14.467 | 7.585 |
| | (20,30] | 0.067 | 0.933 | 0.831 | 0.371 | 3.667 | 4.0 | 25.000 | 5.368 |
| GPT2 + LLMaP (Huang et al.) | (0,10] | 0.111 | 0.946 | 0.831 | 0.360 | 3.292 | 3.625 | 6.156 | 8.218 |
| | (10,20] | 0.066 | 0.955 | 0.829 | 0.342 | 2.978 | 3.378 | 14.467 | 7.583 |
| | (20,30] | 0.055 | 0.952 | 0.846 | 0.342 | 3.333 | 3.333 | 25.000 | 5.388 |
| GPT2 + CLAP | (0,10] | 0.382 | 0.975 | 0.904 | 0.661 | 3.688 | 3.833 | 6.156 | 8.333 |
| | (10,20] | 0.259 | 0.97 | 0.892 | 0.592 | 3.689 | 3.444 | 14.467 | 7.983 |
| | (20,30] | 0.164 | 0.928 | 0.861 | 0.526 | 3.667 | 4.000 | 25.000 | 4.956 |

Table 9: Evaluation results on the Original WikiHow by separating test set into several Step Bucket.

| Model | Program |
|---|---|
| | **RobotHow** Task: Write an Email |
| Human | Step 1: Walk to home office. Step 2: Walk to computer. Step 3: Find computer. Step 4: Turn to computer. Step 5: Look at computer. Step 6: Walk to computer. Step 7: Find chair. Step 8: Sit on chair. Step 9: Find keyboard. Step 10: Grab keyboard. Step 11: Find mouse. Step 12: Grab mouse. Step 13: Type on keyboard |
| LLMaP | Step 1: Switch on computer. |
| CLAP | Step 1: point at mail. Step 2: put envelope on printer.Step 3: put notes on printer. |
| | **WikiHow** Task: How to Become an Art Investor |
| Human | Step 1: Start with some experience or interest in art. Step 2: Understand the difference between art collectors, art investors and art speculators. Step 3: Figure out what you are willing to pay for art, before going to an auction house. Step 4: Pay attention to what schools of art are selling well, and which are down. Step 5: Focus art investments on fine art paintings, rather than decorative art. Step 6: Reach out to trusted auction houses and dealers when you are looking to buy art. Step 7: Buy your investment art when you feel confident of its worth, its price and its ability to grow in value. Step 8: Study how art is properly stored. Step 9: Have your art investments appraised occasionally.Step 10: Consider renting out your art investments. Step 11: Understand that selling an art investment can take time. |
| LLMaP | Step 1: Reach out to trusted auction houses and dealers when you are looking to buy art. |
| CLAP | Step 1: figure out what you are willing to pay for art, before going to an auction house. Step 2: consider renting out your art investments. Step 3: buy your investment art when you feel confident of its worth, its price and its ability to grow in value. |

Table 10: Showcases of procedural steps predicted by different models with GPT3 as the base LLM.

# D QUALITATIVE EXAMPLES

More qualitative examples that are randomly selected are provided. Table 11 show random samples on the original dataset. Table 12 show random samples on the counterfactual datasets with the Intermediate Step intervention method. And Table 13 shows random samples on the counterfactual RobotHow with the Initial Configuration and Final Goal intervention methods.

| Model | Program |
|---|---|
| | **RobotHow** Task: Play Games |
| Human | Step 1: Walk to bedroom.Step 2: Walk to video game console.<br>Step 3: Find video game console.Step 4: Walk to video game console.<br>Step 5: Switch on video game console.Step 6: Find television.<br>Step 7: Switch on television.Step 8: Find video game controller.<br>Step 9: Grab video game controller.Step 10: Find chair.<br>Step 11: Walk to chair.Step 12: Sit on chair |
| Chain | Step 1: Put chef knife on water glass.Step 2: Find sink. |
| LLMaP | Step 1: Walk to bedroom.Step 2: Walk to video game console.<br>Step 3: Find video game console |
| CLAP | Step 1: Walk to video game controller.Step 2: Find video game controller.<br>Step 3: Switch on video game controller.Step 4: Find tv.Step 5: Switch on tv. |
| | **WikiHow** Task: How to Be an Organized Artist |
| Human | Step 1: Keep related supplies in the same area.,.<br>Step 2: Make an effort to clean a dedicated workspace after every session.,.<br>Step 3: Place loose supplies in large, clearly visible containers.,.<br>Step 4: Use clotheslines and clips to hang sketches, photos, and reference material.,.<br>Step 5: Use every inch of the room for storage, especially vertical space.,.<br>Step 6: Use chalkboard paint to make space for drafting ideas right on the walls.,.<br>Step 7: Purchase a label maker to make your organization strategy semi-permanent.,.<br>Step 8: Make a habit of throwing out old, excess, or useless stuff each month. |
| Chain | Step 1: Feel for a dense, kinky coat.<br>Step 2: Look for rounded ears with crimped inner hairs. |
| LLMaP | Step 1: Make sure that all money has been covered on both end.<br>Step 2: Check your bill and banking statements regularly.to ensure that your bills are<br>getting paid as they should be, you should check all statements that you receive.<br>Step 3: Keep track of your expenses.Step 4: Keep track of your income. |
| CLAP | Step 1: Prioritize your goals for the workshop.Step 2: Prepare to create your neopoprealist mural.<br>Step 3: Organize your computer-based materials.Step 4: Have a clear plan.<br>Step 5: Work on being the best you.<br>Step 6: Paint patterns with brushes of suitable size for the particular portion of work you are painting.<br>Step 7: Put together a schedule and chart.<br>Step 8: Buy your investment art when you feel confident of its worth, its price and its ability to grow<br>in value.<br>Step 9: Make a clean brea.<br>Step 10: First, you will need to make sure you have all of the materials listed below. |
| | **WikiHow** Task: How to Use Saddle Soap |
| Human | Step 1: Read the manufacturer's directions., Step 2: Avoid lye heavy products.,<br>Step 3: Make your own saddle soap. |
| Chain | Step 1: Give her a safe space.Step 2: Let him exercise.<br>Step 3: Be kind to him.Step 4: Offer him a treat. |
| LLMaP | Step 1: Dip a damp cloth in the saddle soap.<br>Step 2: Dampen several cloths and work them into a lather with the saddle soap.<br>Step 3: Dip a cloth into the mixture.Step 4: Rub the product on with a cloth.<br>Step 5: Clean the grime off with the cloth.Step 6: Dampen a cloth with water.<br>Step 7: Rub a damp microfiber cloth over the surface.Step 8: Use a damp cloth to wipe once again.<br>Step 9: Dampen a towel with lukewarm water.Step 10: Wipe the area clean with a damp towel.<br>Step 11: Soak a towel in warm water. |
| CLAP | Step 1: Brush your horse.Step 2: Clean the saddle.<br>Step 3: Start by using the first cloth to rub down the leather, making sure you remove any built<br>up grass and dirt.<br>Step 4: Rub the leather down so you remove most of the oil and wax.<br>Step 5: Wipe off excess oil.Step 6: Rinse the leather.<br>Step 7: Clean any other leather tack items (martingale, breastplate, draw reins, cruppers, etc. |

Table 11: Showcases of procedural steps predicted by different models with GPT2 as the base LLM.

| Model | Program |
|---|---|
| | **RobotHow** Intermediate Step<br>Turking (Find computer.). |
| Human | Step 1: Walk to home office.<br>Step 2: Walk to desk.<br>Step 3: Find chair.<br>Step 4: Sit on chair.<br>Step 5: Find computer.<br>Step 6: Switch on computer |
| Chain | Step 1: Switch on computer.<br>Step 2: Switch off computer. |
| LLMaP | Step 1: find computer. |
| CLAP | Step 1: Walk to chair.<br>Step 2: Turn to computer.<br>Step 3: Look at computer.<br>Step 4: Switch off computer.<br>Step 5: Switch on computer. |
| | **WikiHow** Intermediate Step<br>Task: How to Become an Art Investor (Figure out what you are willing to pay for art, before going to an auction house.). |
| Human | Step 1: Start with some experience or interest in art.,.<br>Step 2: Understand the difference between art collectors, art investors and art speculators.,.<br>Step 3: Figure out what you are willing to pay for art, before going to an auction house.,.<br>Step 4: Pay attention to what schools of art are selling well, and which are down.,.<br>Step 5: Focus art investments on fine art paintings, rather than decorative art.,.<br>Step 6: Reach out to trusted auction houses and dealers when you are looking to buy art.,.<br>Step 7: Buy your investment art when you feel confident of its worth, its price and its ability to grow in value.,.<br>Step 8: Study how art is properly stored.,.<br>Step 9: Have your art investments appraised occasionally.,.<br>Step 10: Consider renting out your art investments.,.<br>Step 11: Understand that selling an art investment can take time. |
| Chain | Step 1: See art being sold directly by artists by clicking on the ”direct from the artist” link.<br>Step 2: Look at the price.<br>Step 3: Look at the writer’s website.<br>Step 4: Create a portfolio of your work.<br>Step 5: Look at price again. |
| LLMaP | Step 1: Create a sketch in the neopoprealist manner of the future mural on a small piece of paper 8”x10” using<br>the black ink pen.<br>Step 2: Prepare to create your neopoprealist mural.<br>Step 3: Prepare your paint.<br>Step 4: Begin your project with a design.<br>Step 5: Produce a scaled down version of your finished mural.<br>Step 6: Prepare the wall to be painted.<br>Step 7: Allow the background and base coats to dry.<br>Step 8: Draw the lines, then fill the appeared section with different repetitive patterns (examine the images above).<br>Step 9: Paint patterns with brushes of suitable size for the particular portion of work you are painting.<br>Step 10: Clean up the lines and shapes as needed.<br>Step 11: Seal the mural if needed.<br>Step 12: Be inspired and it will help you succee.<br>Step 13: This is where you get to be the most creative. |
| CLAP | Step 1: Figure out what you are willing to pay for art, before going to an auction house.<br>Step 2: Sell your art online.<br>Step 3: Go to the auction house.<br>Step 4: Buy it.<br>Step 5: Install the first floor.,repeat<br>steps 1-4 to add more floor.<br>Step 6: Enter in the second round values.<br>Step 7: Get the 3-carat diamond.<br>Step 8: Know the four cs of diamonds.<br>Step 9: Find a reputable diamond appraiser in your area. |

Table 12: Showcases of procedural steps predicted by different models with GPT2 as the base LLM under counterfactual setting (Intermediate Step).

| Model | Program |
|---|---|
| | **RobotHow** Initial Configuration<br>Task: Play games in dining room. |
| Human | Step 1: Walk to dining room.Step 2: Walk to video game console.<br>Step 3: Find video game console.Step 4: Walk to video game console.<br>Step 5: Switch on video game console.Step 6: Find television.<br>Step 7: Switch on television.Step 8: Find video game controller.<br>Step 9: Grab video game controller.Step 10: Find chair.<br>Step 11: Walk to chair.Step 12: Sit on chairTask: Play games in dining room.<br>Step 1: Walk to video game controller.Step 2: Put video game controller on diningtable.<br>Step 3: Put boardgame on kitchen table.Step 4: Put boardgame on diningtable. |
| Chain | Step 1: Walk to dining room.Step 2: Walk to sauce pan.<br>Step 3: Pour bottle water into dish bowl.Step 4: Walk to water.<br>Step 5: Walk to carrot.Step 6: Walk to food salt. |
| LLMaP | Step 1: Walk to kitchen.Step 2: Walk to food food.Step 3: Walk to dining room. |
| CLAP | Step 1: walk to dining room.Step 2: find diningtable.Step 3: walk to board game.<br>Step 4: put video game controller on diningtable.Step 5: put food snack on diningtable.<br>Step 6: put board game on diningtable.<br>Step 7: put boardgame on diningtable.Step 8: put board game on kitchen table.<br>Step 9: find video game console.Step 10: put glass on board game.<br>Step 11: grab video game controller.Step 12: put glass on boardgame. |
| | **RobotHow** Final Goal<br>Task: Turn light off and clean |
| Human | Step 1: Walk to bedroom.Step 2: Walk to light.Step 3: Switch off light.<br>Step 1: Walk to home office.Step 2: Walk to rag.Step 3: Find rag.<br>Step 4: Grab rag.Step 5: Walk to desk.Step 6: Find computer.<br>Step 7: Wipe computer.Step 8: Wipe desk.Step 9: Put back rag. |
| Chain | Step 1: Walk to kitchen.Step 2: Walk to cooking pot.Step 3: Walk to water.<br>Step 4: Walk to dishwasher. |
| LLMaP | Step 1: Put light bulb on bowl.Step 2: Switch off light bulb.Step 3: Switch on light. |
| CLAP | Step 1: plug out lighting.Step 2: put cleaning solution on desk.<br>Step 3: find dish soap.Step 4: scrub light switch.<br>Step 5: wipe lighting. |

Table 13: Showcases of procedural steps predicted by different models with GPT2 as the base LLM under counterfactual setting (Initial Configuration, Final Goal).

# E DISCUSSION

## E.1 LIMITATIONS

Though pointing out a direction to prompt out actionable knowledge in large-scale pre-trained language models with external commonsense knowledge, the limitations of reasoning long-horizon language plan still exist. Existing datasets for language planning like WikiHow and RobotHow are all monolingual supporting only English goals and plans. In the future, it is important to expand these datasets or having novel datasets that support multiple languages used across the world. The inherent difference between these languages may also result in different planning strategies in granularity or abstraction levels, which is potentially challenging. In addition, the long-horizon and complex composite tasks still remain challenging for the existing language planners.

## E.2 ETHICAL CONSIDERATIONS

We hope to debias the language planning to avoid misleading either humans or robots with daily life instructions, which may result in unsafe situations. The cultural bias behind these datasets can be a critical issue for future work. As the ground truth planning steps usually reflect the culture shared by the English-speaking group, other cultures may have a completely different practical consideration that leads to different orders of these steps or even novel steps that are failed to be proposed by the LLMs we utilized in this paper. In the future, we will consider the cultural bias as a proxy variable so that we could adjust the implicit knowledge from LLM or commonsense from external sources according to different needs of cultural backgrounds.