Swap or Skip? Challenging Step Type Identification in Instructional Manuals

Anonymous ACL submission

Abstract

Large language models (LLMs) have been widely used as procedural planners, providing step-by-step guidance across applications. However, in a human-assistive scenario where the environment and users' knowledge constantly change, their ability to detect various step types for alternative plan generation remains under-explored. To fill this gap, we assess whether models can identify steps that are: (i) sequential, (ii) interchangeable, and (iii) optional in textual instructions. We compare LLMs to two vision-aware models relevant for procedural understanding: a large vision-language model and a heuristic approach that uses video-mined knowledge graphs. Our results indicate that LLMs struggle to capture the notion of mutual exclusivity between sequential and interchangeable steps. Furthermore, we report comprehensive analyses highlighting the advantages and limitations of using LLMs as procedural task guides. While the largest LLM shows expert-level task knowledge, our findings reveal its limitations in several key areas: broad task coverage, robustness towards diverse user phrasings, and physical reasoning.[1]

1 Introduction

Large language models (LLMs) have demonstrated impressive performance on abstract planning in various scenarios, ranging from classical planning problems (Valmeekam et al., 2023b,a; Guan et al., 2023), embodied household tasks (Song et al., 2022; Lin et al., 2023a), to a real-world human-assistive setup (Patel et al., 2023). In a situated assistive setting where users' environment and knowledge constantly evolve, an AI agent's ability to provide alternative step orders for a procedural task becomes crucial (Bao et al., 2023). This requires the AI agent to identify different transition types between steps within a manual as illustrated in Figure 1, such as (i) **sequential** ordering, (ii) **interchangeable** execution, and (iii) **optional** step skipping. By understanding these step types, an AI agent can perform what-if reasoning (Wu et al., 2023a) to generate alternative plans that account for users' current situation. Take "growing geraniums" for instance (Fig. 1): if the user lives in a climate where planting outdoors is possible year-round, Step 2 wouldn't be necessary to successfully achieve the goal.

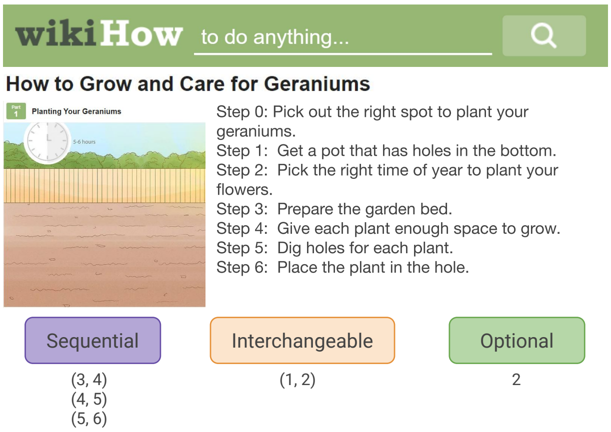

Figure 1: We evaluate whether models can identify various step types in instructional manuals, including (i) **sequential** ordering, (ii) **interchangeable** execution, and (iii) **optional** step skipping.

Despite its importance, current research has not systematically evaluated LLMs' ability to identify step transition types within procedural manuals. Evaluating this ability can further reveal whether models possess physical knowledge related to action pre/postconditions (Wu et al., 2023b; Brahman et al., 2023). It requires reasoning about object state changes (Tandon et al., 2020) and resolving co-references (Anthonio and Roth, 2021; Rim et al., 2023). In response to this, we propose a procedural **step type identification task** in a text-based environment. To this end, we annotate pro-

[1]The code and the dataset will be published at github.com/anonymous/sio upon acceptance.

cedural tasks across three domains featuring physical activities from WikiHow (Koupaee and Wang, 2018), an online instructional resource for everyday tasks, and formulate each step type identification task, i.e., sequential, interchangeable, and optional, as a binary classification problem. We evaluate LLMs of varied size (7B-70B parameters) against each step type identification task and compare them with two classes of vision-aware models, relevant for procedural knowledge acquisition (Wu et al., 2022): (a) a vision-enhanced counterpart, i.e., Large Vision-Language Model (LVLM), exposed to visual signals during pretraining; (b) a heuristic-based pipeline that offers more control over how these step types can be inferred from a video-mined probabilistic procedural knowledge graph (§5). In addition, to assess LLMs' potentials and limitations as procedural task guides, we conduct a comprehensive analysis that considers domain coverage, knowledge types (expert or commonsense knowledge), user phrasing variations, and the ability to explain the impact of step reordering/skipping for a given goal. Our empirical findings indicate that even advanced black-box systems like LLMs exhibit greater difficulty compared to a heuristic-based baseline in capturing the mutual exclusivity between sequential and interchangeable types. While even the largest LLM shows promise for providing expert-level tips, LLMs still require further development to address proficiency gaps across domains. In addition, their handling of negated expressions and reasoning about object state changes needs to be improved to support humans in procedural tasks. Our contributions are as follows:

- we investigate the capabilities of L(V)LMs and a heuristic pipeline in identifying various step types in instructional manuals.
- we demonstrate that while LLMs excel at understanding sequential order, they struggle with its mutually exclusive concept, i.e., interchangeable execution.
- we report extensive analyses focusing on aspects relevant to assisting humans in procedural tasks.

2 Related Work

Procedural Knowledge Modeling and Evaluation. Procedural knowledge entails comprehending a goal along with its feasible steps for achieving it, encompassing various levels of granularity and forms of reasoning (Zhang, 2022). We focus our discussion on instructional text as it is a common form for studying procedural information. On the hierarchical dimension, some works study goal–subgoal relationships (Zhou et al., 2022), goal–step inference (Yang et al., 2021) and cross-task generalization (Zhou et al., 2023a). The horizontal axis, i.e., step–step relations, has received extensive research attention, especially in understanding temporal step ordering (Zhang et al., 2020; Wu et al., 2022) and causal dependencies (Jang et al., 2023). Various aspects of reasoning about individual steps have been widely studied, including action condition inference (Wu et al., 2023b), co-reference resolution (Anthonio and Roth, 2021; Rim et al., 2023), and entity/state tracking (Tandon et al., 2020; Wu et al., 2023c; Kim and Schuster, 2023; Zhang et al., 2024). Our contribution is to advance research on understanding step–step relations, focusing on **sequential order, interchangeable execution, and optional steps** (Zhou et al., 2023b). These aspects are crucial for gauging the underlying physical reasoning of systems and have practical applications in generating alternative step sequences.

Procedural Knowledge Graph Construction. A procedural knowledge graph (PKG) effectively captures metadata, entity information and goal–step hierarchies, making it useful for representing complex procedural processes (Zhang, 2022). Various techniques have been employed to create PKGs, each focusing on a distinct aspect. Jang et al. (2023) annotated step conditions (e.g., "completed") to train a graph generation model for capturing causal dependencies. Zhang et al. (2022) modeled the temporal and cross-modal evolution of entities for machine reading comprehension. Numerous works have explored multimodal grounding between instructional texts and videos to create a visually-aware PKG that represents diverse task demonstrations. Based on the information of such a type of PKG, Ashutosh et al. (2023) leveraged a probabilistic prior for key step recognition while Zhou et al. (2023a) generated pseudo-labels to enable cross-task step recognition. Zhou et al. (2023b) extracted multiple step sequences for nonsequentiality acquisition in a path generation model. Our work differs from prior works by inducing step types through a heuristic algorithm applied to a visually-aware probablisitic PKG.

3 Definitions & Task Formulations

We focus on three step types: sequential, interchangeable and optional. Given a procedural task t consisting of ordered steps $s_0 \rightarrow s_1 \rightarrow ... \rightarrow s_{|t|}$, each step type identification is formulated as a **binary classification task** on an adjacent step pair or an individual step. For tractability, we aim for local reasoning by contextualizing the type identification within a context window, similar to the step ordering task (Zhang et al., 2020) that focuses on adjacent steps only.

Sequential. We define an adjacent step pair (s_i, s_{i+1}) as sequential if their order cannot be swapped to ensure the task completion. The context window is limited to these two steps. The potential sequential step pairs form a sequence $[(s_0, s_1), ..., (s_{|t|-1}, s_{|t|})]$, totaling at most $|t| - 1$ pairs.

Interchangeable. At any given current step s_i, an interchangeable step pair (s_{i+1}, s_{i+2}) is defined as one where s_{i+1} and s_{i+2} can be executed in any order without impacting task completion. In this case, s_i acts as the anchor step, and steps beyond its succeeding two-step context are not considered. Thus, starting from the first step s_0 as the anchor, a sequence of $[(s_1, s_2), ..., (s_{|t|-1}, s_{|t|})]$, totaling at most $|t| - 2$ pairs, might be interchangeable.

Optional. The optionality of a step is also determined within a two-step context. This implies that at any current step s_i, if one can skip the immediately following step s_{i+1} and proceed directly to s_{i+2} without disrupting the task completion, s_{i+1} is deemed optional. Beginning with the first step as the anchor, optional steps might form a sequence $[s_1, s_2, ..., s_{|t|-1}]$ with at most $|t| - 2$ optional steps.

4 Human Annotated SIO Dataset

We create our *Sequential, Interchangeable, Optional* (SIO) dataset by annotating data from WikiHow, a large online instructional resource for everyday tasks[2], based on the definitions in §3. We focus on three domains—*Food and Entertaining (F&E)*, *Home and Garden (H&G)*, and *Hobbies and Crafts (H&C)*, which contain step transitions with strong logical dependencies, e.g., "clear the soil of weeds" is followed by "add compost to the soil". Conversely, other domains like *Pets and Animals* involve mostly independent activities (e.g., "kill fleas" and "remove visible ticks"; Zhang et al. 2020). Specifically, we use the dataset introduced by Zhang et al. (2020), referred to as WikiHow-Clean. This dataset is a curated subset of WikiHow articles where the title is merged with each section name, forming what we call *task-parts*, e.g., *How to Grow and Care for Geraniums - Planting Your Geraniums*. Additionally, the steps within each task-part are automatically identified as ordered (see §5.1 for details).[3] The annotation guidelines are detailed in Appendix A.1.

Annotation Process. We randomly hold out 45 task-parts from WikiHowClean, 15 for the dev set and 30 for the test set, equally distributed across domains. For each procedural task, we ask 3 annotators[4] to select the step tuples for each step category that fulfill the definitions described in §3. Moreover, for both the "selected" step tuples and the overall procedural task, annotators are instructed to indicate whether expert knowledge is required based on public perception rather than personal experience. If expert knowledge is not involved, we specify commonsense knowledge is applied even for unfamiliar tasks. For example, consider "Fixing a Laundry Machine": one could intuitively grasp that "opening the door" must precede "unloading the laundry" without specific expertise. We report the averaged pairwise inter-annotator-agreements (see App. A.2 for more details) for each annotation field and found they range from slight to fair $(0.15 - 0.37)$ on the dev set and increase to moderate $(0.43 - 0.5)$ on the test set. Upon examining, we found that the low agreement might be associated with a high percentage of reported expert knowledge requirement (see Fig. 4, App. A.2).

SIO Dataset. For each step type, we assign a positive label for the majority-voted (2 out of 3) step tuples and a negative label for the rest of the cases. In addition, we aggregate the corresponding majority-assigned expert-knowledge labels for the positive cases.[5] Table 1 shows an example of the SIO dataset, and Table 2 gives the dataset size (refer

[2]Available at www.wikihow.com

[3]Despite WikiHowClean's emphasis on ordered step sequences, a qualitative examination of a subset indicates that interchangeable step pairs and optional steps can be identified.

[4]We recruit 9 volunteers familiar with NLP annotation studies, consisting of 6 doctoral students and 3 postgraduate students. Each participant is asked to annotate 15 tasks, with 5 tasks assigned per domain.

[5]If only two annotators select a step tuple to be true for a specific type and there's a conflict in expert-level knowledge, we assign the expert knowledge requirement as `Unsure`.

Task Title	Steps	Annotation
(A) How to Make Huevos Rancheros - Assembling the Dish (*Food and Entertaining*)	Step 0: Spread about 3 oz (85 g) of refried beans onto each of the tortillas Step 1: Place 1 cooked egg onto each of the tortillas Step 2: Pour warm salsa over the eggs Step 3: Top the dish with avocado, lime juice, cilantro, cheese, or sour cream	sequential: [(1, 2)] sequential_expert: [No] interchangeable: [(2, 3)] interchangeable_expert: [No] optional: [2] optional_expert: [No] task_expert: No

Table 1: An example of the SIO test set (see Tab. 6, App. A.1 for more examples).

	Dev (15 tasks)			Test (28 tasks)		
Step Type	positive	negative	all	positive	negative	all
Sequential	44	20	64	77	51	128
Interchangeable	11	38	49	23	77	100
Optional	7	42	49	19	81	100

Table 2: Instances of step types across the dataset splits.

to Tab. 9, App. A.3 for the label distribution across task domains).[6] Our analysis confirms that step tuples labeled as sequential are never interchangeable, supporting the concept of mutual exclusivity between these types.

5 Visually Inferred Step Types

Online video demonstrations offer a valuable resource to explore alternative ways to realize a task. We built upon several works (Zhou et al., 2023a,b; Ashutosh et al., 2023) on constructing visually-aware procedural knowledge graphs (PKGs) by linking the procedural repository WikiHow (Koupaee and Wang, 2018) to the instructional video resource HowTo100M (Miech et al., 2019b). In our case, we build topic-specific PKGs that represents how likely a step transition is for a given task (§5.1). We then leverage this information to automatically infer step types for sequential order, interchangeable execution, and optional skipping. Figure 2 shows the overall pipeline for inferring step types from the PKG.

5.1 Video-mined PKG Construction

Resource Control. To ensure the usefulness of the PKG, we use specific derivatives of WikiHow and HowTo100M: WikiHowClean (Zhang et al., 2020) and CAE (Yang and Silberer, 2023), respectively. WikiHowClean has undergone quality control to ensure the logical order of steps, predicted by a fine-tuned RoBERTa.[7] CAE (Yang and Silberer, 2023) is a condensed set of HowTo100M containing text-video clip pairs targeted at visually perceivable effect-causing actions.

Topic Clustering. To find sensible alterations in step orders, we first group similar topics, e.g., *Grow Geraniums Indoors* and *Grow Geraniums in Pots*. Concretely, we extract sentence representation of task titles in both WikiHowClean and CAE, and perform agglomerative clustering to identify topic clusters.

Probabilistic PKG Construction. To construct a topic-wise PKG, we consider two inputs on an instructional topic X: (1) a key step library K^X sourced from a single WikiHowClean task-parts t containing a set of step headlines $(s_0^t, ..., s_j^t, ..., s_{|t|}^t)$; (2) a video set V^X collected from CAE, where each $V_i = (v_1, ..., v_i, ..., v_{|V_i|}) \in V^X$ is a sequence of video clips. A topic-wise task graph $\mathcal{T}^X = (\mathcal{V}, \mathcal{E}, w)$ has the vertex set $\mathcal{V} = K^X$, the edge set $\mathcal{E}$ that represents the one-hop step transitions grounded in the videos, and $w_{(j,k)}$ represents edge probabilities of $s_j^t \rightarrow s_k^t$. We will use "step" and "node" interchangeably in the following. A valid node transition $s_j^t \rightarrow s_k^t$ can be determined by their respective cross-modal grounding scores to any consecutive video clips $(v_i,\ v_{i+1})$. We employ VideoCLIP (Xu et al., 2021), a language–video alignment model pretrained to be robust toward temporal misalignment, to compute the cross-modal scoring function, $f(v_i, s_j^t)$. The output of $f(v_i, s_j^t)$ is the dot product between the pooled visual and the pooled textual representations. We keep the `top-k` grounding scores as a quality threshold. In other words, if both $f(v_i, s_j^t)$ and $f(v_{i+1}, s_k^t)$ exceed a certain threshold, it is more likely that (s_j^t, s_k^t) is a valid step transition. A transition score $s(s_j^t, s_k^t)$ is obtained by multiplying the respective grounding scores, i.e., $s(s_j^t, s_k^t) = f(v_i, s_j^t) \cdot f(v_{i+1}, s_k^t)$. The final tran-

[6] We discarded 2 ambiguous tasks reported by annotators from the *Craft* domain in the test set.

[7] The prediction file can be found here.

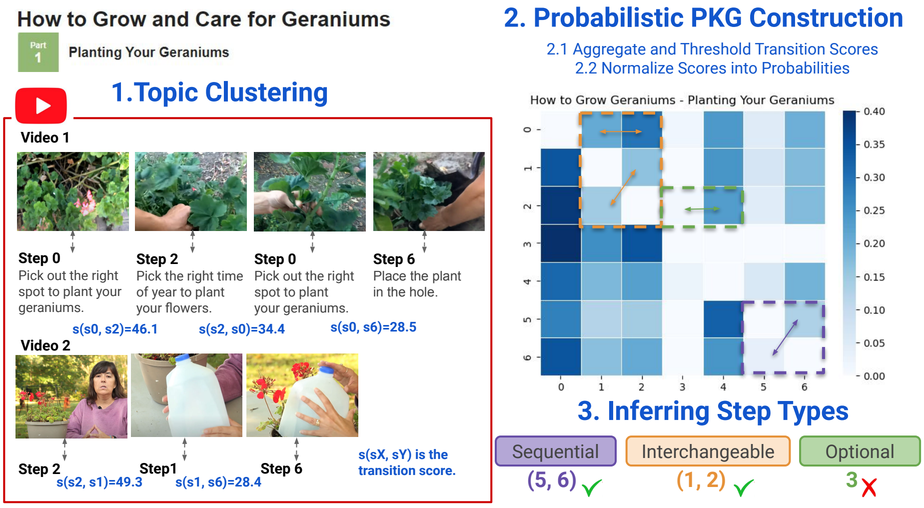

Figure 2: An overview based on the example in Figure 1 shows the PKG construction and step type inference. For simplicity, we only show the top aligned steps for each video segment and provide the step descriptions once.

sition scores of a pair $s(s^t_j, s^t_k)$ are aggregated and we prune the edge connections by keeping the ones whose scores lie above the q-th percentile among all. To calculate $w_{(j,k)}$, we divide $s(s^t_j, s^t_k)$ by the sum of transition scores of all outgoing edges from s^t_j. The final topic-wise PKG represents probabilistically of how likely $s^t_j \rightarrow s^t_k$ is compared to $s^t_j \rightarrow s^t_l$ according to the aligned video clips. See Appendix B for details on preprocessing and hyperparameters.

5.2 Inferring Step Types

The probability difference between pairs of step transitions on PKG can be leveraged to infer step types as defined in §3. To reduce the complexity, we restrict the step transitions to occur within a context window that spans a maximum of 2 subsequent steps. To extract a *sequential* pair (s^t_i, s^t_{i+1}), we compare the transition probability of $s^t_i \rightarrow s^t_{i+1}$ and $s^t_{i+1} \rightarrow s^t_i$ against a threshold SEQ:

$$(s^t_i, s^t_{i+1}) = \begin{cases} 1 & \text{if } w_{(i,i+1)} - w_{(i+1,i)} > \texttt{SEQ} \\ 0 & \text{otherwise} \end{cases}$$

The larger the threshold SEQ, the stricter the sequential order that must be held. For an *interchangeable* step pair (s^t_{i+1}, s^t_{i+2}) to be valid, we require at any given step s^t_i, the absolute difference between the transition likelihood $s^t_i \rightarrow s^t_{i+1}$ and $s^t_i \rightarrow s^t_{i+2}$ to be smaller than a threshold INT, and $s^t_{i+1} \rightarrow s^t_{i+2}$ must not follow a strict sequential order:

$$(s^t_{i+1}, s^t_{i+2}) = \begin{cases} 1 & \text{if} \quad |w_{(i,i+1)} - w_{(i,i+2)}| < \texttt{INT} \\ & \quad \text{and } w_{(i+1,i+2)} - w_{(i+2,i+1)} < \texttt{SEQ} \\ 0 & \text{otherwise} \end{cases}$$

This indicates that there is no strict order of executing s^t_{i+1} or s^t_{i+2} after completing s^t_i. However, if $s^t_i \rightarrow s^t_{i+2}$ exceeds $s^t_i \rightarrow s^t_{i+1}$ by a threshold OPT, we can infer that s^t_{i+1} is *optional*:

$$s^t_{i+1} = \begin{cases} 1 & \text{if } w_{(i,i+2)} - w_{(i,i+1)} > \texttt{OPT} \\ 0 & \text{otherwise} \end{cases}$$

For the employed algorithms, refer to Appendix B.2.

6 Experimental Setup

6.1 Pipeline Baseline

As the baseline to compare to black-box LLMs we use the modular method described in §5.2, termed as PKG BASELINE, to examine the feasibility of incorporating procedural knowledge in a more intuitive way. We tune the threshold set (SEQ, INT and OPT) based on the annotated SIO dev set (see App. B.2 for more details).

6.2 Large (Vision-)Language Models

Model Selection. To systematically understand if procedural knowledge emerges through model scale, we opt for open-source instruction-tuned models of variable sizes and choose the Llama-2 series (Touvron et al., 2023) for evaluation: LLAMA-7B, LLAMA-13B, and LLAMA-70B. Moreover, we examine if VIDEO-LLAVA-7B (Lin et al., 2023b), a vision-enhanced version of Llama that underwent multimodal pretraining (images and videos), better encodes procedural knowledge due to its exposure to visual signals.[8] Refer to Ap-

[8]We did not choose Llava 1.5 (Liu et al., 2023) since its language backbone is frozen during multimodal pretraining.

	Sequential			Interchangeable			Optional			All		
Model	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
PKG Baseline	**71.9**	29.9	42.2	**26.9**	**78.3**	**40.0**	20.7	31.6	25.0	**39.8**	46.6	35.7
Llama-7B	58.3	81.8	68.1	23.5	34.8	28.1	23.1	47.4	31.0	35.0	54.7	42.4
Video-Llava-7B	58.9	85.7	69.8	13.3	8.7	10.5	16.4	63.2	26.1	29.5	52.5	35.5
Llama-13B	71.8	72.7	72.3	25.0	73.9	37.4	22.0	52.6	31.2	39.7	66.4	47.0
Llama-70B	63.3	**89.6**	**74.2**	20.8	43.5	28.2	**33.3**	**68.4**	**44.8**	39.1	**67.2**	**49.1**

Table 3: Results (%): step type identification task. Boldface indicates the best result across models.

pendix C for implementation details.

Prompt Design. We leverage the instruction-following capabilities of L(V)LMs for step type identification by employing a two-shot prompt containing one positive and one negative example. To prompt Video-Llava-7B on a linguistic task, we feed in a white blank image for the visual input and adapt the template with a textual prefix: "`This is a blank image. Please only attend to the following text.`". To evaluate the usefulness of the model's explanations for the user, we further prompt the model zero-shot to provide a rationale for its **correct** predictions.[9] All the templates, example prompts as well as the decoding strategies can be found in Appendix D.

Answer Parser. For sequential and interchangeable tasks, we parse the output as `{"step pairs": [(x, x+1), (y, y+1), ...]}`, while for optional we use `{"steps": [x, y, ...]}`. To extract valid outputs from a model's generated raw texts for automatic evaluation, we first remove all quotation marks and redundant white spaces and add quotation marks for the dictionary keys, e.g., `{"steps":}`. Then, we take the first occurrence of the desired output format as the model's prediction. If there is no desired output format, we assign an empty dictionary `{"step pairs": []}` as the model's prediction for sequential and interchangeable types, and `{"steps": []}` for the optional.

6.3 Evaluation Metrics

For each binary classification task of step types, i.e., sequential, interchangeable and optional, we report precision (P), recall (R) and F_1 score averaged across step/step pair instances in the test set. As an additional evaluation of L(V)LMs' instruction-following ability in the step type identification task, we propose two error-quantifying metrics: constraint violation (CV) and step range exceedance (SE). CV occurs when the output does not adhere to the desired output constraint. For example, a non-adjacent pair is not a desired answer, e.g., `{"step pairs": [(0, 2)]}`. SE captures cases where the step index exceeds the length of the input step sequence.[10]

7 Results

As shown in Table 3, Llama-70B achieves the best performance averaged across all types (avg. F_1: 49.1). The effectiveness on the step type identification task seems to correlate with the model scale (see Fig. 5 in App. E), except for the interchangeable type (F_1 = 37.4 for Llama-13B). L(V)LMs' strong performance on identifying sequential step pairs should help to detect interchangeable pairs, as the two concepts potentially entail mutual exclusivity. However, the interchangeable type unexpectedly proves to be the most challenging for L(V)LMs. Except for Llama-13B, L(V)LMs even perform significantly worse than PKG Baseline (min. −2.6pp in F_1). This highlights a key advantage of PKG: its flexibility enables the manipulation of step transition probabilities, thereby integrating mutual exclusivity between sequential and interchangeable steps into the extraction heuristics (cf. §5.2). To identify optional steps, we hypothesize that L(V)LMs can leverage certain linguistic markers in step headlines. While this phenomenon is more evident in Llama-70B, the limited data (only two steps contain such information[11]) prevents conclusive findings. Overall, in the interchangeable and optional tasks where negative step pairs outnumber positive ones (cf. Tab. 2), all examined models tend to overpredict, i.e., generate

[9] We found few-shot prompts led the models to syntactically generate rationales in a slot-filling manner following the example style, e.g., "the precondition of step x is to perform step y".

[10] SE is a subcase of CV; thus, a case of SE is also a case of CV. E.g., in a task with a five-step sequence, an output of `(0,4)` is a CV error, while `(5,6)` is both, a CV and SE error.

[11] "wet the runway (optional)" and "wipe away excess ink as necessary"

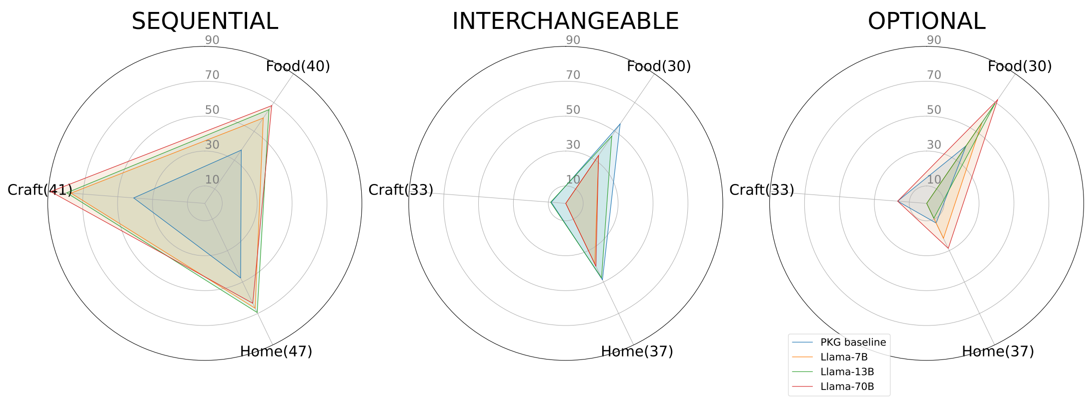

Figure 3: Procedural task performance per domain in F_1 score (%); `domain (#)` represents instances per domain.

false positives. Upon examination, all L(V)LMs examined here exhibit a 100% type 1 error rate under all negative cases (i.e., generate false positives), despite encountering negative scenarios in the crafted few-shot examples (cf. App. D).

Compared to its vision-enhanced counterpart, LLAMA-7B consistently outperforms VIDEO-LLAVA-7B with the exception of a slight performance drop for the sequential type (-1.7pp in F_1). However, LLAMA-7B displays more type 2 errors (i.e., false negative) than VIDEO-LLAVA-7B. Upon closer inspection, the high recall observed in VIDEO-LLAVA-7B can be largely attributed to chance, as it generates every possible step pair and step range. On the interchangeable task, VIDEO-LLAVA-7B shows lower recall (-26.1pp) because it fails to generate any desired output, leading our answer parser to interpret it as "empty" (cf. §6.2). This undesired response is also reflected in its highest instruction-following error rates (see Tab. 18 in App. E). The results suggest that VIDEO-LLAVA-7B's ability to adapt to purely linguistic (unimodal) tasks is limited, likely because it had little exposure to the unimodal input scenario during the instruction-tuning stage. This is an important aspect to address when using one foundation model across all modalities. In conclusion, our results show that accounting for instruction-following error rates is crucial to assess L(V)LMs' capabilities in a structured answer prediction task.

8 Analysis

Our analyses investigate key aspects of effective procedural task systems: broad task coverage, expert level knowledge, adaptability to user phrasings, and the ability to explain the impact of step changes. Note that since VIDEO-LLAVA-7B's quantitative performance might be overestimated (§7), we only provide its qualitative analysis.

Procedural Task Domains. As depicted in Figure 3, while all models perform well on sequential steps across domains (especially *Craft*), the effectiveness on interchangeable and optional steps varies across domains substantially. Notably, *Craft*'s scarcity of positive labels for these steps (cf. Tab. 9) likely explains the severe drop to 0% F_1. From the model perspective, the Llama2-family performs well in the *Food* domain regardless of the step types considered, while also showing consistent weaknesses in the *Craft* domain.

Expert vs. Commonsense Knowledge. We explore how the underlying knowledge types affect the effectiveness and hypothesize that LLMs may possess more expert knowledge than PKG BASELINE due to its pretraining on web-scale data. Indeed, we observe that with larger model size, F_1 improves on the sequential type[12] for expert-knowledge instances and the performance gap between two knowledge types narrows (see Fig. 6 in App. E). The same pattern for LLMs cannot be observed for the commonsense knowledge type.

Robustness towards User Phrasings. We prompt LLMs with semantically equivalent paraphrases through the usage of negation words, referred to as **negated** template. Take the interchangeable type for instance: the original phrasing "can be executed interchangeably" becomes "do not necessarily follow a sequential order" in its corresponding negated template (see App. D.2). A

[12]The only step type annotated for expert knowledge involvement, which suggests that it may be challenging to identify other step types without domain-specific knowledge.

Examples	LLAMA-70B's Suboptimal Rationales	Required Reasoning
Example (D) How to Make an Avocado Shake; Sequential (1, 2). Step 1: Blend the ingredients until the **mixture** is no longer chunky. Step 2: Line the sides of your glasses with chocolate syrup, and pour the **puree** into the glasses.	Blending the ingredients until they are no longer chunky creates a uniform texture for the shake.	Entity Co-reference: "puree" in Step 2 refers to the "mixture" in Step 1; thus, the sequential order should hold.
Example (E) How to Iron on a Patch; Interchangeable (1, 2). Step 1: Place the **patch** in the position you chose. Step 2: Heat up an **iron**	Both steps involve preparing the patch and base item for ironing	State Tracking: heating an iron first (Step 2) before placing the patch (Step 1) does not cause the state change of the patch and still achieve the desired final state of the patch (being ironed).

Table 4: Error analysis on the explanation quality. Refer to Table 6 for full step sequences.

highly effective model should exhibit little performance differences across prompt phrasings (Aroca-Ouellette et al., 2021). Contrary to our intuition that larger models could handle diverse phrasings better, the smaller model, LLAMA-7B, shows the lowest performance variability (shown in Tab. 19, App. E). The Llama-family models generally remain limited in understanding negative sentences (García-Ferrero et al., 2023) as they show **worse recall on the negated prompt**. An exception can be seen for the interchangeable type (LLAMA-70B's R: +17.4 pp), suggesting a transferable understanding on the semantics of "sequential order" (present in both sequential-original & interchangeable-negated prompts). Yet, the mutual exclusivity between the concept of "sequential" and "interchangeable" is far from being captured (diff. in F_1: 37.4%[13]). Interestingly, LLAMA-13B shows the opposite pattern, i.e., better at handling interchangeable-original phrasings than their negations. Therefore, **model size seems to play a role in biasing over one type of expression than the other**.

Explanation Quality. Effective individualized user guidance in a procedural task requires models to explain consequences of step reordering/skipping. This requires reasoning about involved entities and the impact of actions on those entities (action-effect changes). In the procedural context, key reasoning aspects for high-quality explanations are entity co-reference (Anthonio and Roth, 2021; Rim et al., 2023) and state tracking (Tandon et al., 2020; Kim and Schuster, 2023; Zhang et al., 2024). Entity co-reference links different expressions in a text that refer to the same entity. State tracking monitors state changes that an entity undergoes throughout a procedural task. This underlying reasoning allows the model to explain if altering step orders or skipping a step would disrupt the postcondition of the current step and the precondition for subsequent steps. While LLAMA-70B emerged as the best-performing model, its explanations remain high-level as shown in Table 4 (refer to App. F for more analysis across L(V)LMs). Importantly, LLAMA-70B's explanations lack information that is necessary for non-expert users to understand the consequences of alternative task executions.

[13] LLAMA-70B's F_1 when prompted with sequential-original and interchangeable-negated template are 74.2% and 36.8%, respectively.

9 Conclusions

We investigate models' capabilities in identifying various step types in instructional manuals, namely sequential order, interchangeable execution and optional step skipping. Despite their vast knowledge storage, LLMs struggle to grasp the mutually exclusive concept of sequential and interchangeable steps. Furthermore, they still face challenges in covering various task domains, handling diverse phrasings, and explaining relevant impacts about pre/postconditions between step transitions.

10 Limitations

Procedural knowledge for some tasks can be quite complex and requires specialized expertise. Our evaluation scope and fine-grained annotation are limited by the lack of domain-specific experts among our recruited participants. Therefore, large-scale domain-expert annotation on procedural knowledge is crucial for future work, enabling creation of a well-annotated dataset for evaluating L(V)LMs on real-world knowledge tasks. Our analysis focuses on models' local reasoning within a specific context and does not yet explore models' ability to reason globally about non-adjacent steps and their relationships. Furthermore, we identify several challenges in constructing the procedural knowledge graph (PKG). First, despite the initial topic clustering stage, cross-modal alignment often falls short of true goal awareness. In other words, PKG doesn't consider the impact of changing step order on task success, nor does it effectively account for longer contextual dependencies. Second, grounding to demonstration videos for alternative step orders can be error-prone for tasks with short and concise steps. In these cases, videos introduce noise by suggesting potentially unintended re-ordering. Nevertheless, PKG BASELINE offer several advantages in which LLMs examined here are limited. First, their task coverage is comprehensive as long as corresponding video demonstrations can be retrieved. Second, their modular nature allows for easier intervention to ensure their usefulness. This can be achieved through, for example, more rigorous resource control and a stronger cross-modal alignment backbone. Finally, they are not susceptible to variations in user phrasing. With these complementary strengths to LLMs, a hybrid setup that augment LLMs' knowledge with external resource (Gao et al., 2023) is worth investigating in procedural task guidance.

References

Talita Anthonio and Michael Roth. 2021. Resolving implicit references in instructional texts. In *Proceedings of the 2nd Workshop on Computational Approaches to Discourse*, pages 58–71, Punta Cana, Dominican Republic and Online. Association for Computational Linguistics.

Stéphane Aroca-Ouellette, Cory Paik, Alessandro Roncone, and Katharina Kann. 2021. PROST: Physical reasoning about objects through space and time. In *Findings of the Association for Computational Linguistics: ACL-IJCNLP 2021*, pages 4597–4608, Online. Association for Computational Linguistics.

Kumar Ashutosh, Santhosh Kumar Ramakrishnan, Triantafyllos Afouras, and Kristen Grauman. 2023. Video-Mined task graphs for keystep recognition in instructional videos. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Yuwei Bao, Keunwoo Yu, Yichi Zhang, Shane Storks, Itamar Bar-Yossef, Alex de la Iglesia, Megan Su, Xiao Zheng, and Joyce Chai. 2023. Can foundation models watch, talk and guide you step by step to make a cake? In *Findings of the Association for Computational Linguistics: EMNLP 2023*, pages 12325–12341, Singapore. Association for Computational Linguistics.

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.

Yunfan Gao, Yun Xiong, Xinyu Gao, Kangxiang Jia, Jinliu Pan, Yuxi Bi, Yi Dai, Jiawei Sun, Meng Wang, and Haofen Wang. 2023. Retrieval-Augmented generation for large language models: A survey.

Iker García-Ferrero, Begoña Altuna, Javier Alvez, Itziar Gonzalez-Dios, and German Rigau. 2023. This is not a dataset: A large negation benchmark to challenge large language models. In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 8596–8615, Singapore. Association for Computational Linguistics.

Lin Guan, Karthik Valmeekam, Sarath Sreedharan, and Subbarao Kambhampati. 2023. Leveraging pre-trained large language models to construct and utilize world models for model-based task planning. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*, volume 36, pages 79081–79094.

Ari Holtzman, Jan Buys, Li Du, Maxwell Forbes, and Yejin Choi. 2019. The curious case of neural text degeneration. In *8th International Conference on Learning Representations, ICLR 2020, Addis Ababa, Ethiopia, April 26-30, 2020*.

Yunseok Jang, Sungryull Sohn, Lajanugen Logeswaran, Tiange Luo, Moontae Lee, and Honglak Lee. 2023. Multimodal subtask graph generation from instructional videos.

Najoung Kim and Sebastian Schuster. 2023. Entity tracking in language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3835–3855, Toronto, Canada. Association for Computational Linguistics.

Mahnaz Koupaee and William Yang Wang. 2018. WikiHow: A large scale text summarization dataset.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. 2023a. On grounded planning for embodied tasks with language models. In *Thirty-Seventh AAAI Conference on Artificial Intelligence, AAAI 2023, Thirty-Fifth Conference on Innovative Applications of Artificial Intelligence, IAAI 2023, Thirteenth Symposium on Educational Advances in Artificial Intelligence, EAAI 2023, Washington, DC, USA, February 7-14, 2023*, volume 37, pages 13192–13200.

Bin Lin, Yang Ye, Bin Zhu, Jiaxi Cui, Munan Ning, Peng Jin, and Li Yuan. 2023b. Video-LLaVA: Learning united visual representation by alignment before projection.

Haotian Liu, Chunyuan Li, Yuheng Li, and Yong Jae Lee. 2023. Improved baselines with visual instruction tuning.

Antoine Miech, Jean-Baptiste Alayrac, Lucas Smaira, Ivan Laptev, Josef Sivic, and Andrew Zisserman. 2019a. End-to-end learning of visual representations from uncurated instructional videos. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9879–9889.

Antoine Miech, Dimitri Zhukov, Jean-Baptiste Alayrac, Makarand Tapaswi, Ivan Laptev, and Josef Sivic. 2019b. Howto100m: Learning a text-video embedding by watching hundred million narrated video clips. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 2630–2640.

Dhruvesh Patel, Hamid Eghbalzadeh, Nitin Kamra, Michael Louis Iuzzolino, Unnat Jain, and Ruta Desai. 2023. Pretrained language models as visual planners for human assistance. In *IEEE/CVF International Conference on Computer Vision, ICCV 2023, Paris, France, October 1-6, 2023*.

Nils Reimers and Iryna Gurevych. 2019. 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)*, pages 3982–3992, Hong Kong, China. Association for Computational Linguistics.

Kyeongmin Rim, Jingxuan Tu, Bingyang Ye, Marc Verhagen, Eben Holderness, and James Pustejovsky. 2023. The coreference under transformation labeling dataset: Entity tracking in procedural texts using event models. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 12448–12460, Toronto, Canada. Association for Computational Linguistics.

Chan Hee Song, Jiaman Wu, Clayton Washington, Brian M Sadler, Wei-Lun Chao, and Yu Su. 2022. LLM-planner: Few-shot grounded planning for embodied agents with large language models. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV), 2023, pp. 2998-3009*, pages 2998–3009.

Niket Tandon, Keisuke Sakaguchi, Bhavana Dalvi, Dheeraj Rajagopal, Peter Clark, Michal Guerquin, Kyle Richardson, and Eduard Hovy. 2020. A dataset for tracking entities in open domain procedural text. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 6408–6417, Online. Association for Computational Linguistics.

Hugo Touvron, Louis Martin, Kevin Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, Dan Bikel, Lukas Blecher, Cristian Canton Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, Anthony Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel Kloumann, Artem Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, Ranjan Subramanian, Xiaoqing Ellen Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zheng Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. 2023. Llama 2: Open foundation and Fine-Tuned chat models.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2023a. PlanBench: An extensible benchmark for evaluating large language models on planning and reasoning about change. In *Thirty-seventh Conference on Neural Information Processing Systems Datasets and Benchmarks Track*.

Karthik Valmeekam, Matthew Marquez, Sarath Sreedharan, and Subbarao Kambhampati. 2023b. On the planning abilities of large language models : A critical investigation. In *Advances in Neural Information Processing Systems 36: Annual Conference on Neural Information Processing Systems 2023, NeurIPS 2023, New Orleans, LA, USA, December 10 - 16, 2023*.

Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Remi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander Rush. 2020. Transformers: State-of-the-art natural language processing. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: System Demonstrations*, pages 38–45, Online. Association for Computational Linguistics.

Te-Lin Wu, Zi-Yi Dou, Qingyuan Hu, Yu Hou, Nischal Chandra, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2023a. ACQUIRED: A dataset for answering counterfactual questions in real-life videos.

In *Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing*, pages 11753–11770, Singapore. Association for Computational Linguistics.

Te-Lin Wu, Alex Spangher, Pegah Alipoormolabashi, Marjorie Freedman, Ralph Weischedel, and Nanyun Peng. 2022. Understanding multimodal procedural knowledge by sequencing multimodal instructional manuals. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 4525–4542, Dublin, Ireland. Association for Computational Linguistics.

Te-Lin Wu, Caiqi Zhang, Qingyuan Hu, Alexander Spangher, and Nanyun Peng. 2023b. Learning action conditions from instructional manuals for instruction understanding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 3023–3043, Toronto, Canada. Association for Computational Linguistics.

Xueqing Wu, Sha Li, and Heng Ji. 2023c. OpenPI-C: A better benchmark and stronger baseline for open-vocabulary state tracking. In *Findings of the Association for Computational Linguistics: ACL 2023*, pages 7213–7222, Toronto, Canada. Association for Computational Linguistics.

Hu Xu, Gargi Ghosh, Po-Yao Huang, Dmytro Okhonko, Armen Aghajanyan, Florian Metze, Luke Zettlemoyer, and Christoph Feichtenhofer. 2021. VideoCLIP: Contrastive pre-training for zero-shot video-text understanding. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 6787–6800, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Hsiu-Yu Yang and Carina Silberer. 2023. Implicit affordance acquisition via causal Action–Effect modeling in the video domain. In *Proceedings of the 13th International Joint Conference on Natural Language Processing and the 3rd Conference of the Asia-Pacific Chapter of the Association for Computational Linguistics*, pages 846–871, Nusa Dua, Bali. Association for Computational Linguistics.

Yue Yang, Artemis Panagopoulou, Qing Lyu, Li Zhang, Mark Yatskar, and Chris Callison-Burch. 2021. Visual goal-step inference using wikiHow. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 2167–2179, Online and Punta Cana, Dominican Republic. Association for Computational Linguistics.

Huibin Zhang, Zhengkun Zhang, Yao Zhang, Jun Wang, Yufan Li, Ning Jiang, Xin Wei, and Zhenglu Yang. 2022. Modeling Temporal-Modal entity graph for procedural multimodal machine comprehension.

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

Li Zhang, Qing Lyu, and Chris Callison-Burch. 2020. Reasoning about goals, steps, and temporal ordering with WikiHow. In *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 4630–4639, Online. Association for Computational Linguistics.

Li Zhang, Hainiu Xu, Abhinav Kommula, Chris Callison-Burch, and Niket Tandon. 2024. OpenPI2.0: An improved dataset for entity tracking in texts. In *Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 166–178, St. Julian's, Malta. Association for Computational Linguistics.

Honglu Zhou, Roberto Martín-Martín, Mubbasir Kapadia, Silvio Savarese, and Juan Carlos Niebles. 2023a. Procedure-Aware pretraining for instructional video understanding. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition, CVPR 2023, Vancouver, BC, Canada, June 17-24, 2023*.

Shuyan Zhou, Li Zhang, Yue Yang, Qing Lyu, Pengcheng Yin, Chris Callison-Burch, and Graham Neubig. 2022. Show me more details: Discovering hierarchies of procedures from semi-structured web data. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2998–3012, Dublin, Ireland. Association for Computational Linguistics.

Yu Zhou, Sha Li, Manling Li, Xudong Lin, Shih-Fu Chang, Mohit Bansal, and Heng Ji. 2023b. Non-sequential graph script induction via multimedia grounding. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 5529–5545, Toronto, Canada. Association for Computational Linguistics.

A Details of the SIO Dataset

A.1 Annotation Task Description

Annotation Fields Explanation. The annotation fields and the corresponding explanations are shown in Table 5. For both interchangeable and optional types, we remind annotators to base their judgment ONLY on the overall task goal and the future two steps `(i+1, i+2)` at the current step `i` without looking into far distant steps. More examples for the annotated SIO dataset are displayed in Table 6.

A.2 Inter-Annotator Agreements (IAAs)

Table 7 shows the IAAs across annotation fields in the dev and the test set. The breakdown by procedural domains can be further seen in Table 8. Note that when calculating IAA for expert-knowledge

Annotation Field	Explanation
Sequential `(i, i+1)`	If you find step `i` & step `i+1` follow a strict sequential order; in other words, the task won't succeed if you swap the order, please note down `(i, i+1)` in the column "Sequential `(i, i+1)`".
Interchangeable `(i, (i+1, i+2))`	At any given step i, judge if step i+1 & step i+2 can be interchangeably executed without affecting the task completion. If yes, please note down `(i, (i+1, i+2))` in the column "Interchangeable `(i, (i+1, i+2))`". In this case, `(i+1, i+2)` can be executed interchangeably when one is at step `i`.
Optional `(i, (i+1), i+2)`	At any given step i, judge if one can omit step `i+1` and proceed to step `i+2` without affecting the task completion. If yes, please note down `(i, (i+1), i+2)` in the column "Optional `(i, (i+1), i+2)`". In this case, `(i+1)` can be omitted and one can go from step `i` to step `(i+2)` and still complete the task.
Step Level Expert Knowledge Required?	For each step annotation, please additionally answer whether it requires expert/domain knowledge. If a step annotation does not require domain/expert knowledge, it could be solved by commonsense knowledge. For example, anyone who does not have prior experience in cutting a "durian" would know a step transition **Step 0:** use a knife to cut a "durian" → **Step 1:** take out the seed of a "durian") can not be swapped based on the commonsense knowledge that "Before taking out seeds from a fruit with a spiky hard surface, one has to cut it open".
Task Level Expert Knowledge Required?	After annotating all the step relations, please also answer whether this task generally requires expert/domain knowledge in the column "Overall Task: Domain Knowledge Required?". Requiring expert/domain knowledge means one has to follow a step-by-step instruction to ensure the task can be completed even with similar prior experiences.

Table 5: Annotation fields and the corresponding descriptions.

requirement, we only consider selected step tuples under full agreement. The low IAA could be attributed to expert knowledge requirement as can be seen in Figure 4.

A.3 Label Distributions

The label distributions across procedural task domains is shown in Table 9.

B Details on PKG Construction and Inferred Step Types

B.1 PKG Construction

Preprocessing Details. For topic clustering (Step 1), we lower-cased and removed stop words from the task title before extracting the sentence features ($\mathbb{R}^1 \times 768$) using `all-mpnet-base-v2` model from the `sentence-transformer` library (Reimers and Gurevych, 2019). When performing cross-modal grounding w.r.t topic-wise PKG construction , we follow VideoClip's recipe to decode each video clip v_i at 30 fps and extracted features with S3D (Miech et al., 2019a) per second, resulting in $v_i = \{\mathbf{v}_x\}_{x=1}^{|v_i|}$ video tokens ($\mathbb{R}^{|v_i|} \times 512$). To calculate the cross-modal alignment, we mean pooled across the video tokens to represent a whole video clip representation ($\mathbb{R}^1 \times 512$). Since VideoCLIP expects an input of text–video clip pair, we duplicated a v_i to pair with each of the $s_j^t \in K^X$ and calculated the respective cross-modal grounding score $f(v_i, s_j^t)$ accordingly.

Hyperparameters. For the clustering, the following parameters are used: `task clustering linkage=Average`, `task clustering distance threshold=0.5`, `task clustering affinity=cosine`. Sanity check shows that the same task title (exact string match) across WikiHow and HowTo100M are clustered together. For the cross-modal grounding, `topk=3` and `q-th=25`.

B.2 Details on Inferring Step Types

Algorithm 1 show the functions on inferring step type from a topic-wise PKG. In Table 10, we provide the results on the SIO dev set when experimenting different combinations of step type identification hyperparameters. The final hyperparameter set we use is: `SEQ`$= 0.07$, `INT`$= 0.25$, `OPT`$= 0.08$.

C Implementation Details

All L(V)LMs we evaluated here can be accessed through the HuggingFace Library (version: `4.39.2`) (Wolf et al., 2020): `meta-llama/Llama-2-7b-chat-hf`, `meta-llama/Llama-2-13b-chat-hf`, `meta-llama/Llama-2-70b-chat-hf`, `LanguageBind/Video-LLaVA-7B-hf`. The

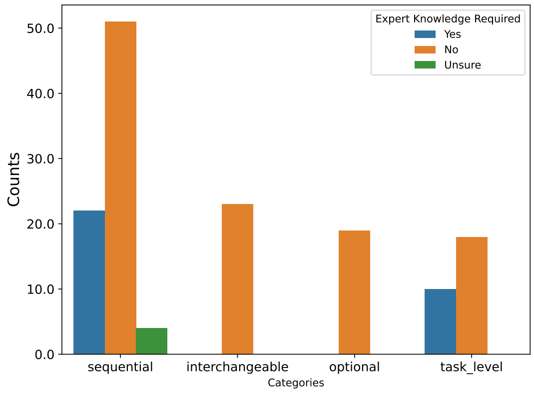

Figure 4: Test set: distribution of expert knowledge requirement across step types.

inference is run on a single NVIDIA RTX A6000 GPUs within 2 hours through 4-bit quantization techniques.

D Details on Prompt Design & Decoding Strategies

D.1 Original Template and Prompt

The template of Llama-2 family and VIDEO-LLAVA-7B are shown in Table 11 and Table 12 respectively. As for the prompts across step types, we demonstrate some examples by initializing the template for Llama-2 family: Table 13, 14, 15.

D.2 Negated Template

For brevity, Table 16 only highlights the difference between the original and the negated template on the template variables: `{Step Type Task Definition}` and `{Step Type Question}`.

D.3 Rationale Template and Prompt

To exemplify how we obtain models' rationales, the template and an example prompt of Llama-2 family can be found in Table 17. For VIDEO-LLAVA-7B, the special tokens, e.g., `USER` and the textual prefix, are additionally added.

D.4 Decoding Strategy

We opt for greedy-search (`temperature`: 0.1) to ensure step type prediction to be more deterministic and apply nucleus sampling (Holtzman et al., 2019) (`temperature`: 0.3, `top-p`: 0.9) for rationale generation to encourage more diverse and coherent text. For nucleaus sampling, we explore two sets of hyperparamers: [`temperature`: 0.1, `top-p`: 0.9] and [`temperature`: 0.3, `top-p`: 0.9], and found little change in the rationale quality (as a reference, the overall F_1 difference on the step type identification tasks is $< 1\%$ across models).

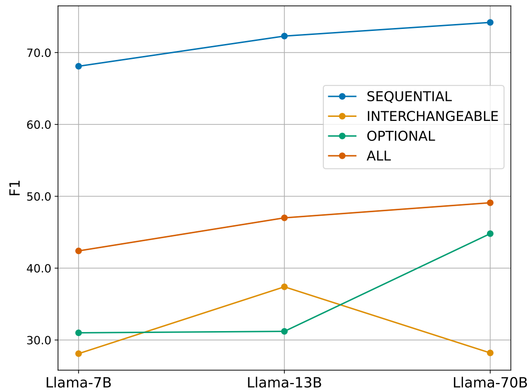

Figure 5: Performance of different model size

E Results: Figures & Tables

Figure 5 examines the role of parameter size on the step type identification task performance.
Table 18 details the instruction-following error rate across L(V)LMs examined here.
Figure 6 shows the performance by knowledge type; note that only sequential type has the expert-knowledge annotation.
Table 19 shows score changes of L(V)LMs when prompted with negated template.

F Model Rationales

For more examples on the generated rationales based on the test inputs in Table 1 and Table 6 across models, refer to Table 20, 21, and 22, respectively. LLAMA-7B tends to generate generic rationales without making distinction between step types. In contrast, the largest model examined here,

Algorithm 1 Inferring Step Types

```
function FINDSEQUENTIAL(PKG, SEQ_threshold)
    result ← empty list
    num_nodes, _ ← shape of PKG
    for i in range(num_nodes) do
        if i is the last node then
            continue
        end if
        SEQ_diff ← PKG[i, i+1] - PKG[i+1, i]
        if SEQ_diff > SEQ_threshold then
            item ← ((i, i+1))
            append item to result
        end if
    end for
    return result
end function
function FINDINTERCHANGEABLE(PKG, INT_threshold, SEQ_threshold)
    result ← empty list
    num_nodes, _ ← shape of PKG
    for i in range(num_nodes) do
        if i is the last or second-to-last node then
            continue
        end if
        INT_diff ← | PKG[i, i+1] - PKG[i, i+2] |
        SEQ_diff ← PKG[i+1, i+2] - PKG[i+2, i+1]
        if INT_diff < INT_threshold & SEQ_diff < SEQ_threshold then
            item ← ((i+1, i+2))
            append item to result
        end if
    end for
    return result
end function
function FINDOPTIONAL(PKG, OPT_threshold)
    result ← empty list
    num_nodes, _ ← shape of PKG
    for i in range(num_nodes) do
        if i is the last or second-to-last node then
            continue
        end if
        OPT_diff ← PKG[i, i+2] - PKG[i, i+1]
        if OPT_diff > OPT_threshold then
            item ← i+1
            append item to result
        end if
    end for
    return result
end function
```

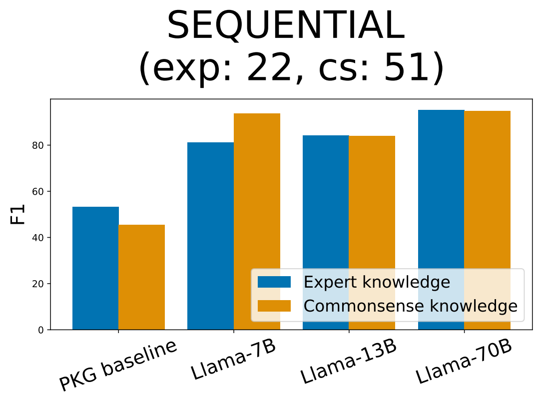

Figure 6: Knowledge type performance in F_1 score (%). (exp:#, cs:#) represents instances per knowledge type; note that only positive labels have such annotation.

LLAMA-70B, even includes informative tips (see row (D) of Tab. 22), suggesting its potential usefulness in an human-assistive scenario. Furthermore, we validate our previous findings that VIDEO-LLAVA-7B's effectiveness is overestimated, as it merely replicates the input step description for some cases (see Tab. 23)

G Usage of Writing Assistance

We use public available writing assistance tools including chatGPT, HuggingChat and Gemini for refining language and suggesting alternative phrasings for readability.

Task Title	Steps	Annotation
(B) How to Operate a Roomba - Maintaining Your Roomba After Use (*Home and Garden*)	Step 0: Empty the Roomba's bin Step 1: Use a dry cloth to clean the Roomba Step 2: Charge it after every use Step 3: Store your Roomba on the charger Step 4: Replace the filter every two months	sequential: [] sequential_expert: [] interchangeable: [(1, 2), (2, 3), (3, 4)] interchangeable_expert: [No, No, No] optional: [1, 2, 3] optional_expert: [No, No, No] task_expert: No
(C) How to Collect Stamps - Removing Paper from Used Stamps (*Hobbies and Crafts*)	Step 0: Handle stamps with stamp tongs Step 1: Cut off most of the envelope Step 2: Soak most stamps in lukewarm water Step 3: Rinse and dry the stamps Step 4: Remove self-adhesive stamps with air freshener	sequential: [(1, 2), (2, 3), (3, 4)] sequential_expert: [Yes, No, Yes] interchangeable: [] interchangeable_expert: [] optional: [] optional_expert: [] task_expert: Yes
(D) How to Make an Avocado Shake - Making the Shake (*Food and Entertaining*)	Step 0: Mix the avocado, milk, ice, and sugar in a blender Step 1: Blend the ingredients until the mixture is no longer chunky Step 2: Line the sides of your glasses with chocolate syrup, and pour the puree into the glasses Step 3: Garnish the shake	sequential: [(0, 1), (1, 2), (2, 3)] sequential_expert: [No, No, No] interchangeable: [] interchangeable_expert: [] optional: [] optional_expert: [] task_expert: No
(E) How to Iron on a Patch - Ironing on the Patch (*Home and Garden*)	Step 0: Lay the base item on a flat, heat-resistant surface Step 1: Place the patch in the position you chose Step 2: Heat up an iron Step 3: Place a thin towel over the patch Step 4: Position the heated iron over the patch and press down Step 5: Remove the iron and allow the patch to to cool	sequential: [(0, 1), (3, 4), (4, 5)] sequential_expert: [No, No, No] interchangeable: [(1, 2), (2, 3)] interchangeable_expert: [No, No] optional: [] optional_expert: [] task_expert: No
(F) How to Tattoo Leather - Applying the Design to Leather *(Hobbies and Crafts)*	Step 0: Test an out of sight portion of the leather Step 1: Ink the main outline of your design Step 2: Wipe away excess ink as necessary Step 3: Add accents and details after the main body of the design Step 4: Fill in solid features of the design Step 5: Clean off any excess ink and show off your tattooed leather	sequential: [(0, 1), (1, 2), (2, 3), (3, 4), (4, 5)] sequential_expert: [No, No, Unsure, No, No] interchangeable: [] interchangeable_expert: [] optional: [2] optional_expert: [No] task_expert: Yes

Table 6: More examples of SIO test set.

	Dev	Test
sequential	0.15	0.43
sequential_expert	0.30	0.21
interchangeable	0.37	0.53
interchangeable_expert	0.33	0.33
optional	0.25	0.50
optional_expert	0.33	1.00
task_level_expert	0.34	0.25

Table 7: Averaged pair-wise IAAs.

	Dev			Test		
	Food	Home	Craft	Food	Home	Craft
sequential	0.24	*-0.01*	0.03	0.41	0.55	0.07
sequential_expert	0.33	0.03	0.33	1.00	0.29	0.11
interchangeable	0.41	0.22	0.32	0.52	0.54	0.18
interchangeable_expert	1.00	N/A	0.33	1.00	0.33	N/A
optional	0.44	*-0.02*	0.11	0.62	0.29	0.10
optional_expert	0.33	N/A	N/A	1.00	1.00	N/A
task_level_expert	0.33	0.18	0.44	*-0.05*	0.28	0.05

Table 8: Averaged pair-wise IAA by procedural task domains, where "N/A" denotes no instances available for calculating IAA.

	Dev						Test					
Step Type	Food		Home		Craft		Food		Home		Craft	
	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.	pos.	neg.
sequential	13	12	19	3	12	5	18	22	24	23	35	6
interchangeable	6	14	2	15	3	9	9	21	13	24	1	32
optional	4	16	1	16	2	10	12	18	6	31	1	32

Table 9: Label distribution across procedural task domains.

	Sequential			Interchangeable			Optional			All		
Hyperparameters	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
SEQ= 0.1, INT= 0.05, OPT= 0.08	45.0	11.0	18.0	17.0	9.0	12.0	9.0	14.0	11.0	24.0	11.0	14.0
SEQ= 0.07, INT= 0.25, OPT= 0.45	64.0	32.0	42.0	21.0	55.0	31.0	0.0	0.0	0.0	28.0	29.0	24.0
SEQ= 0.07, INT= 0.25, OPT= 0.1	64.0	32.0	42.0	21.0	55.0	31.0	0.0	0.0	0.0	28.0	29.0	24.0
SEQ= 0.07, INT= 0.25, OPT= 0.08	64.0	32.0	42.0	21.0	55.0	31.0	7.0	14.0	10.0	**31.0**	**34.0**	**28.0**

Table 10: Dev Results (%): step type binary classification task. Boldface indicates best result.

Template for Llama-2 family
<s>[INST] We analyze step relations in a procedural task.
{Step Type Task Definition}
You will be given a task along with its corresponding step sequence wrapped in «».
You will be prompted with a question wrapped in <>.
{(Applicable for interchangeable & optional) Constraint} [/INST] Got it! Give me the first example. </s>
<s>[INST] Great! Here is your first example:
«{Few-shot Example 1 (positive case)}»
<{Step Type Question}>
{Output Format Specification}[/INST] **{Few-shot Example 1 Answer}** </s>
<s>[INST] Great! Here is another example:
«{Few-shot Example 2 (negative case)}»
<{Step Type Question}>
{Output Format Specification} [/INST] **{Few-shot Example 2 Answer}** </s>
<s>[INST] Perfect! Here is another example:
«{Inference Time Input}»
<{Step Type Question}>
{Output Format Specification} [/INST]

Table 11: Few-shot (2 shots) template for Llama-2 family, where one conversational turn (<s>...</s>) contains one example.

Template for VIDEO-LLAVA-7B
A chat between a curious human and an artificial intelligence assistant.
The assistant gives helpful, detailed, and polite answers to the human's questions.
USER: <image>
This is a blank image. Please only attend to the following text.
We analyze step relations in a procedural task.
{Step Type Task Definition}
You will be given a task along with its corresponding step sequence wrapped in «».
You will be prompted with a question wrapped in <>.
{(Applicable for interchangeable & optional) Constraint} ASSISTANT: Got it! Give me the first example. </s>
USER: Great! Here is your first example:
«{Few-shot Example 1 (positive case)}»
<{Step Type Question}>
{Output Format Specification} ASSISTANT: **{Few-shot Example 1 Answer}** </s>
USER: Great! Here is another example:
«{Few-shot Example 2 (negative case)}»
<{Step Type Question}>
{Output Format Specification} ASSISTANT: **{Few-shot Example 2 Answer}** </s>
USER: Perfect! Here is another example:
«{Inference Time Input}»
<{Step Type Question}>
{Output Format Specification}
ASSISTANT:

Table 12: Few-shot (2 shots) template for VIDEO-LLAVA-7B, where one conversational turn (USER ...</s>) contains one example.

Sequential Original Prompt
<s>[INST] We analyze step relations in a procedural task. Your task is to select adjacent step pairs that have to be executed sequentially to ensure the completion of the task. You will be given a task along with its corresponding step sequence wrapped in «». You will be prompted with a question wrapped in <>. [/INST] Got it! Give me the first example. </s> <s>[INST] Great! Here is your first example: «Task: How to Grow Geraniums - Planting Your Geraniums, Step sequence: Step 0: Pick out the right spot to plant your geraniums -> Step 1: Get a pot that has holes in the bottom -> Step 2: Pick the right time of year to plant your flowers -> Step 3: Prepare the garden bed -> Step 4: Give each plant enough space to grow -> Step 5: Dig holes for each plant -> Step 6: Place the plant in the hole» <Which adjacent step pairs have to be executed sequentially to ensure the completion of the task?> Return selected step pairs in a json format with ’step pairs’ as the key and no more details. [/INST] ’step pairs’: [(4, 5), (5, 6)] </s> <s>[INST] Great! Here is another example: «Task: How to Stargaze Comfortably - Getting Ready to Stargaze, Step sequence: Step 0: Stargaze on a dry, clear night -> Step 1: Stargaze in the summertime -> Step 2: Get out of the city for the most relaxation -> Step 3: Watch out for wildlife» <Which adjacent step pairs have to be executed sequentially to ensure the completion of the task?> Return selected step pairs in a json format with ’step pairs’ as the key and no more details. [/INST] ’step pairs’: [] </s> <s>[INST] Perfect! Here is another example: «Task: How to Season a Grill - Cleaning the Grates, Step sequence: Step 0: Remove the grates from the grill -> Step 1: Brush the grates with a wire grill brush -> Step 2: Wash and dry the grates» <Which adjacent step pairs have to be executed sequentially to ensure the completion of the task?> Return selected step pairs in a json format with ’step pairs’ as the key and no more details. [/INST]

Table 13: Sequential few-shot (2 shots) prompts initialized from the Llama-2 family’s template. Note that the few-shot examples are chosen from the dev set and specific to the step type.

Interchangeable Original Prompt
<s>[INST] We analyze step relations in a procedural task.
Your task is to select adjacent step pairs that can be executed interchangeably without affecting task completion.
You will be given a task along with its corresponding step sequence wrapped in «». You will be prompted with a question wrapped in <>.
Constraint: At any given step i, look only into step i+1 and step i+2 and judge if one can execute step i+1 and step i+2 interchangeably without affecting task completion.
Do not consider the step pairs (0, 1) and (last step index, non-existing step index) as interchangeable.
For example, if one consider step 3 and step 4 as ’interchangeable’,
it means that when one is at step 2, one can execute step 3 or step 4
in any order without failing the task.[/INST] Got it! Give me the first example. </s>
<s>[INST] Great! Here is your first example: «Task: How to Grow Geraniums - Planting Your Geraniums,
Step sequence: Step 0: Pick out the right spot to plant your geraniums ->
Step 1: Get a pot that has holes in the bottom ->
Step 2: Pick the right time of year to plant your flowers ->
Step 3: Prepare the garden bed ->
Step 4: Give each plant enough space to grow ->
Step 5: Dig holes for each plant ->
Step 6: Place the plant in the hole»
<Which adjacent step pairs can be executed interchangeably without affecting task completion?>
Return selected step pairs in a json format with ’step pairs’ as the key and no more details. [/INST] ’step pairs’: [(1, 2)] </s>
<s>[INST] Great! Here is another example: «Task: How to Grow Portobello Mushrooms - Harvesting the Portobellos,
Step sequence: Step 0: Remove the newspaper in 2 weeks if the mushrooms are growing ->
Step 1: Continue misting the mushrooms as they grow ->
Step 2: Dig out the portobellos when the caps have fully unfurled ->
Step 3: Repeat moistening the compost until new mushrooms form»
<Which adjacent step pairs can be executed interchangeably without affecting task completion?>
Return selected step pairs in a json format with ’step pairs’ as the key and no more details. [/INST] ’step pairs’: [] </s>
<s>[INST] Perfect! Here is another example:
«Task: How to Season a Grill - Cleaning the Grates,
Step sequence: Step 0: Remove the grates from the grill ->
Step 1: Brush the grates with a wire grill brush ->
Step 2: Wash and dry the grates»
<Which adjacent step pairs can be executed interchangeably without affecting task completion?>
Return selected step pairs in a json format with ’step pairs’ as the key and no more details. [/INST]

Table 14: Interchangeable few-shot (2 shots) prompts initialized from the Llama-2 family’s template.

Optional Original Prompt
<s>[INST] We analyze step relations in a procedural task. Your task is to select steps that can be made optional without affecting task completion. You will be given a task along with its corresponding step sequence wrapped in «». You will be prompted with a question wrapped in <>. Constraint: At any current step i, look only into step i+1 and step i+2 and judge if one can omit step i+1 and proceed to the i+2 without affecting task completion. Do not consider the first and the last step as optional. For example, if one consider step 3 as 'optional', it means that when one is at step 2, one can skip step 3 and directly proceed to step 4 without failing the task. [/INST] Got it! Give me the first example. </s> <s>[INST] Great! Here is your first example: «Task: How to Grow Geraniums - Planting Your Geraniums, Step sequence: Step 0: Pick out the right spot to plant your geraniums -> Step 1: Get a pot that has holes in the bottom -> Step 2: Pick the right time of year to plant your flowers -> Step 3: Prepare the garden bed -> Step 4: Give each plant enough space to grow -> Step 5: Dig holes for each plant -> Step 6: Place the plant in the hole» <Which steps can be made optional without affecting task completion?> Return selected step pairs in a json format with 'step pairs' as the key and no more details. [/INST] 'step pairs': [(1, 2)] </s> <s>[INST] Great! Here is another example: «Task: How to Grow Portobello Mushrooms - Harvesting the Portobellos, Step sequence: Step 0: Remove the newspaper in 2 weeks if the mushrooms are growing -> Step 1: Continue misting the mushrooms as they grow -> Step 2: Dig out the portobellos when the caps have fully unfurled -> Step 3: Repeat moistening the compost until new mushrooms form» <Which steps can be made optional without affecting task completion?> Return selected step pairs in a json format with 'step pairs' as the key and no more details. [/INST] 'step pairs': [] </s> <s>[INST] Perfect! Here is another example: «Task: How to Season a Grill - Cleaning the Grates, Step sequence: Step 0: Remove the grates from the grill -> Step 1: Brush the grates with a wire grill brush -> Step 2: Wash and dry the grates» <Which steps can be made optional without affecting task completion?> Return selected step pairs in a json format with 'step pairs' as the key and no more details. [/INST]

Table 15: Optional few-shot (2 shots) prompts initialized from the Llama-2 family's template.

Step Type	Original $\rightarrow$ Negated
Sequential	(A) Your task is to select adjacent step pairs that *have to be executed sequentially* $\rightarrow$ *cannot be swapped* to ensure the completion of the task. (B) Which adjacent step pairs *have to be executed sequentially* $\rightarrow$ *cannot be swapped* to ensure the completion of the task?
Interchangeable	(A) Your task is to select adjacent step pairs that *can be executed interchangeably* $\rightarrow$ *do not necessarily follow a sequential order* without affecting task completion. (B) Which adjacent step pairs *can be executed interchangeably* $\rightarrow$ *do not necessarily follow a sequential order* without affecting task completion.
Original	Your task is to select steps that *can be made optional* $\rightarrow$ *are not necessary* without affecting task completion. (B) Which adjacent steps *can be made optional* $\rightarrow$ *are not necessary* without affecting task completion.

Table 16: Original $\rightarrow$ Negated template: changes in (A) `{Step Type Task Definition}` and (B) `{Step Type Question}`.

Template for Llama-2 family

<s>[INST] We analyze step relations in a procedural task.
Your task is to provide a rationale for each of your previous selected **{sequential step pairs | interchangeable step pairs | optional steps}**.
The rationale should be formulated based on the action preconditions and postconditions.
You will be given a task along with its corresponding step sequence and your previous answers wrapped in «».
You will be prompted with a question wrapped in <>. [/INST] Got it! Give me the first example. </s>
<s>[INST] Great! Here is your first example:
`«{Inference Time Input}»`
<What is the rationale behind each of your selected **{sequential step pairs | interchangeable step pairs | optional steps}**>
Provide the rationale for each step pair in a list of dictionary.
Each dictionary contains keys **{step pair | step}** and 'reason'. No more details. [/INST]

Example Prompt - Sequential

<s>[INST] We analyze step relations in a procedural task.
Your task is to provide a rationale for each of your previous selected sequential step pairs.
The rationale should be formulated based on the action preconditions and postconditions.
You will be given a task along with its corresponding step sequence and your previous answers wrapped in «».
You will be prompted with a question wrapped in <>. [/INST] Got it! Give me the first example. </s>
<s>[INST] Great! Here is your first example:
«Task: How to Season a Grill - Cleaning the Grates,
Step sequence: Step 0: Remove the grates from the grill ->
Step 1: Brush the grates with a wire grill brush ->
Step 2: Wash and dry the grates,
Answers: 'step pairs': [(4, 5), (5, 6)]»
<What is the rationale behind each of your selected sequential step pair?>
Provide the rationale for each step pair in a list of dictionary.
Each dictionary contains keys 'step pair' and 'reason'. No more details. [/INST]

Table 17: Zero-shot rationale generation template and an example prompt for Llama-2 family. The content is the same for Video-Llava-7B.

		Sequential ↓		Interchangeable ↓		Optional ↓	
Template Type	Model	CV	SE	CV	SE	CV	SE
Original	Llama-7B	10.7	**0.0**	64.3	**0.0**	**28.6**	**0.0**
	Video-Llava-7B	46.4	46.4	96.4	21.4	82.1	3.6
	Llama-13B	17.9	14.3	**3.6**	**0.0**	39.3	**0.0**
	Llama-70B	**3.6**	3.6	50	3.6	46.4	**0.0**
Negated	Llama-7B	**3.6**	**0.0**	57.1	**0.0**	**32.1**	**0.0**
	Video-Llava-7B	71.4	60.7	92.9	17.9	85.7	7.1
	Llama-13B	17.9	10.7	**3.6**	**0.0**	39.3	**0.0**
	Llama-70B	**3.6**	3.6	50.0	3.6	50.0	**0.0**

Table 18: Instruction-following error rate (%): CV stands for "constraint violation" whereas SE for "step range exceedance". We calculate both error rates across all procedural tasks (28 in SIO test set).

	Sequential			Interchangeable			Optional			All		
Model	P	R	F_1	P	R	F_1	P	R	F_1	P	R	F_1
Llama-7B	**+2.6**	-27.3	-10.6	**0.0**	**0.0**	**0.0**	**-0.4**	**+5.2**	**+0.7**	**+0.7**	-7.4	-3.3
Llama-13B	-3.4	**-5.2**	-4.3	-4.7	-17.4	-7.5	**-0.4**	-10.5	-2.6	-2.9	-11.0	-4.8
Llama-70B	+6.2	-15.6	**-2.5**	+5.6	+17.4	+8.6	-4.7	-15.8	-7.8	+2.4	**-4.7**	-0.6

Table 19: Robustness towards user phrasing (%): score change from the original template to the negated one. The least difference is boldfaced.

On Example (A)	Sequential (1, 2)	Interchangeable (2, 3)	Optional (2)
(a) LLAMA-7B	step 1: To provide a source of protein and flavor to the dish, step 2: To add flavor and spice to the dish	The third step adds the avocado, lime juice, cilantro, cheese, or sour cream, which enhance the flavors of the dish and make it more enjoyable to eat.	N/A
(b) VIDEO-LLAVA-7B	A cooked egg is placed on top of the refried beans to add protein and flavor to the dish.	N/A	Warm salsa adds flavor and moisture to the eggs.
(c) LLAMA-13B	N/A	Both steps involve adding toppings to the dish, and the order of these steps does not affect the final product.	To add moisture and flavor to the dish
(d) LLAMA-70B	Eggs are placed on top of the beans to create a base for the dish.	Because pouring warm salsa over the eggs will help to add flavor and moisture to the dish, which complements the refried beans and eggs.	Step 2 (Pour warm salsa over the eggs) can be made optional without affecting task completion, as it's a matter of personal preference and not essential to the dish.

Table 20: Rationales across models on the correct prediction per step type for the test input provided in Table 1. N/A indicates no correct prediction.

On Example (B)	Sequential (None)	Interchangeable [(1, 2), (2, 3), (3, 4)]	Optional [1, 2, 3]
(a) LLAMA-7B	None	{'step pair': (1, 2), 'reason': To ensure the Roomba is properly cleaned and ready for the next use}	{'step': 3, 'reason': To ensure the Roomba continues to function properly and efficiently, it is important to store it on the charger after every use. This helps to maintain the battery life and prevent any damage to the internal components.}
(b) VIDEO-LLAVA-7B	None	N/A	{'step': 3, 'reason': 'To ensure the cleaning efficiency of the Roomba, it is important to clean the bin after every use. This helps to remove any dirt or debris that may have accumulated during the cleaning process.' }
(c) LLAMA-13B	None	{'step pair': (2, 3), 'reason': 'Cleaning the Roomba with a dry cloth after each use helps to remove any dirt or debris that may have accumulated during its last cleaning cycle, ensuring that the Roomba remains in good condition and continues to perform effectively.' }, {'step pair': (3, 4), 'reason': 'Storing the Roomba on the charger after each use helps to keep the battery charged and ready for the next cleaning cycle, ensuring that the Roomba is always fully charged and ready to use when needed.'}	{'steps': ['3': {'reason': 'To keep the Roomba clean and functional'}]
(d) LLAMA-70B	None	['step pair': (1, 2), 'reason': 'both steps involve cleaning the Roomba'] ['step pair': (3, 4), 'reason': 'both steps involve maintaining the Roomba's battery and filter']	Step 3: Storing the Roomba on the charger is optional and can be skipped if the user prefers to store it elsewhere.

Table 21: Rationales across models on the correct prediction per step type for the test input provided in Table 6. N/A indicates no correct prediction.

On Example (C)	Sequential [(1, 2), (2, 3), (3, 4)]	Interchangeable	Optional
(a) LLAMA-7B	'step 2': 'Soak most stamps in lukewarm water', 'reason': 'To remove any remaining adhesive residue and make the stamps easier to handle', 'step 3': 'Rinse and dry the stamps', 'reason': 'To remove any remaining moisture and prevent mold or mildew from growing on the stamps'	None	None
(b) VIDEO-LLAVA-7B	Step 3: Soak most stamps in lukewarm water is necessary to remove any ink or glue residue from the stamps. Step 4: Rinse and dry the stamps is important to remove any remaining moisture and to prevent mold or mildew growth.	None	None
(c) LLAMA-13B	'step pairs': [(2, 3), (3, 4)], 'reasons': ['To prepare the stamps for soaking and remove any remaining envelope glue', 'To remove any remaining moisture and prepare the stamps for further processing']	None	None
(d) LLAMA-70B	'step pair': (2, 3), 'reason': 'soaking most stamps in lukewarm water **helps loosen the adhesive**', 'step pair': (3, 4), 'reason': 'rinsing and drying the stamps **helps remove any remaining adhesive and prevents damage to the stamps**'	None	None

Table 22: Rationales across models on the correct prediction per step type for the test input provided in Table 6. N/A indicates no correct prediction.

On Example (D)	Sequential [(0, 1), (1, 2), (2, 3)]	Interchangeable	Optional
	'The first step, mixing the avocado, milk, ice, and sugar in a blender, is necessary to create a smooth and consistent mixture.', 'The second step, blending the ingredients until the mixture is no longer chunky, ensures that the shake is smooth and well-blended.', 'The third step, lining the sides of the glasses with chocolate syrup, adds a decorative touch and enhances the overall presentation of the shake.', 'The fourth step, pouring the puree into the glasses, is necessary to transfer the shake from the blender to the glasses. This step also ensures that the shake is evenly distributed in the glasses.	None	None
On Example (E)	Sequential [(0, 1), (3, 4), (4, 5)]	Interchangeable [(1, 2), (2, 3)]	Optional
	{ 'step pairs': [(0, 1)], 'reasons': [['Lay the base item on a flat, heat-resistant surface is a precondition for ironing on the patch.']] } { 'step pairs': [(3, 4)], 'reasons': [['Place a thin towel over the patch is a precondition for ironing on the patch.']] } { 'step pairs': [(4, 5)], 'reasons': [['Position the heated iron over the patch and press down is a postcondition for ironing on the patch.']] }	'step pairs': [(1, 2)], 'reasons': ['Step 0 and Step 1 are the initial steps in the task, where the base item is laid out and the patch is positioned.', 'Step 2 and Step 3 are the next steps, where the iron is heated and the thin towel is placed over the patch.', 'Step 4 and Step 5 are the final steps, where the iron is positioned over the patch and pressed down, and the iron is removed to allow the patch to cool.',]	

Table 23: VIDEO-LLAVA-7B's erroneous rationales for the test input provided in Table 6.