LLM-based Open Domain Planning by Leveraging Entity-Attribute-Level Domain Models

Dongning Rao[1], **Songlin He**[1], **Zhihua Jiang**[2], **Ruishi Liang**[3]*
[1] School of Computer, Guangdong University of Technology, Guangzhou 510006, China
[2] Department of Computer Science, Jinan University, Guangzhou 510632, China
[3] School of Computer Engineering, University of Electronic Science and Technology of China, Zhongshan Institute, Zhongshan 528402, China
raodn@gdut.edu.cn, 2112305297@mail2.gdut.edu.cn,
tjiangzhh@jnu.edu.cn, liangruishi@foxmail.com

Abstract

Currently, large language models (**L**LMs) based **O**pen domain **N**atural language plannin**G** (LONG) has considerable room for improvement. E.g., non-reusable plans with incomplete intermediate states and missing steps hinder real-world applications. To remedy these flaws, this paper establishes a dataset with a baseline for LONG. The GOLD dataset provides the largest dataset for textual procedures, along with corresponding reusable formal planning domain definitions, to date. The baseline, DIGGER, leverages entity-attribute-level action models, which reveal relevant implicit physical properties (aka attributes) of salient entities in actions. DIGGER first extracts action models and builds typed entity lists from textual procedures. Then, it builds goal states for new tasks and instantiates grounded actions using domain prediction. At last, plans are generalized and translated into textual procedures by LLM. Reference-based metrics, LLM-as-a-Judge, and human evaluation are employed to comprehensively evaluate LONG. Experiments on GOLD validate that DIGGER is stronger and more generalizable than recently proposed approaches and LLMs. I.e., DIGGER is the best in seen domains and applicable to unseen domains without adaptation. Specifically, the BLEU-1 score increased from 0.385 to 0.408 on seen domains and rose to 0.310 on unseen domains. [1]

1 Introduction

Large language models (**L**LMs) based **O**pen domain **N**atural language plannin**G** (LONG[2]) is ideal for real-world open domain planning (Zhang et al., 2024c), which uses natural language (NL). However, the text procedures drafted by humans or

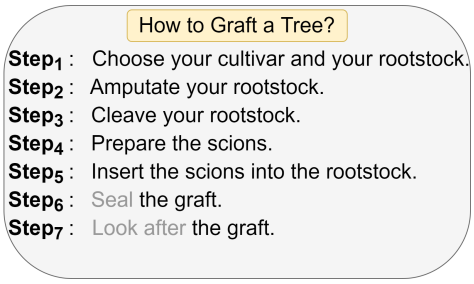

Figure 1: The reference textual procedure for "How to Graft a Tree". Unspecified verbs are in gray.

LONG contain flaws, such as incomplete states and missing steps (Wang et al., 2023). E.g., Fig. 1 is a textual procedure on WikiHow[3] for `"How to Graft a Tree"`. There are at least three flaws in this plan. First, it ignores objects' positions (e.g., `"hole"`). Second, details of `"look after"` are missing (e.g., `"cut away any excess rootstock"`). Third, steps for preparing or waiting are absent (e.g., `"wait for the graft to heal"`).

An appealing approach to boost the performance of LONG (Zhang et al., 2023) is to define domains in the Planning Domain Description Language (PDDL) (Ghallab et al., 1998). However, there are three challenges in facilitating PDDL in LONG: the scarcity of annotated data, the difficulty of building a complete domain model, and the challenges in NL-PDDL translation.

Thus, to benchmark LONG, we propose DIGGER (llm-base**D** open doma**I**n plannin**G** by levera**G**ing **E**ntity-att**R**ibute-level domain models) and a dataset GOLD (natural lan**G**uage pr**O**cedural texts with pdd**L** **D**efinitions).

GOLD includes 103 domains, 1100 tasks, and 1086 actions from four source datasets: WikiHow (Koupaee and Wang, 2018), ALFRED-L (Shridhar et al., 2020), Proc2PDDL (Zhang et al., 2024c), and OPENPI2.0 (Zhang et al., 2024b).

* Corresponding author: Ruishi Liang.

[1] Our source code can be visited via GitHub: https://github.com/fip-lab/DIGGER.

[2] Appx. B Tab. 5 is the list of abbreviations and mathematical symbols.

[3] https://www.wikihow.com/Graft-a-Tree

Findings of the Association for Computational Linguistics: EMNLP 2025, pages 2567–2588
November 4-9, 2025

OPENPI, with human-drafted and planner-verified PDDL definitions, is referred to as OPENPI+ in this paper.

DIGGER has two stages: training and planning. Training with GOLD, DIGGER builds domain clusters, entity-attribute-level action models, and typed entity (Scala and Vallati, 2021) lists. Domain clustering prepares for domain prediction. Then, with NL sentences and PDDL definition as labels, a fine-tuned COMET (Bosselut et al., 2019) extracts action models. We find entity attributes (i.e., physical properties of entities) using ConceptNet (Speer et al., 2017), and action instantiating relies on typed entity lists. Planning via DIGGER has six steps: 1) predicting the domain; 2) extracting goal states from NL using COMET; 3) instantiating grounded actions (Liu and Claßen, 2025) with typed entity lists; 4) providing a few-shot prompt for LLMs; 5) planning via LLMs in PDDL format; 6) generating text procedures.

While evaluating LONG is challenging, we employ a comprehensive evaluation using multiple metrics. As previous studies, we uses reference-based metrics (Schmidtová et al., 2024) including BLEU (Papineni et al., 2002), ROUGE (Lin, 2004), METEOR (Banerjee and Lavie, 2005), and BERTScore (Zhang et al., 2020). We further use LLM-as-a-Judge (LaJ) and human evaluation in this paper. Experiments on GOLD show that DIGGER outperforms three LLMs and two recently proposed approaches. The three LLMs are DeepSeek-V3 (Bi et al., 2024), GPT-4, and GPT-4o, and the two approaches are Open-Ground-Planning (Guo et al., 2024) (or OGP for short) and PLAN (Lu et al., 2023). Specifically, in seen domains where OGP and PLAN can be applied without adaptation, the best BLEU-1 score increased from 0.385 to 0.408. On unseen domains, while the best LLM's BLEU-1 score is 0.089 (GPT-4), DIGGER's score is 0.310.

Our contribution can be summarized as follows:

1) DIGGER uses entity-attribute to identify hidden states and intermediate steps.

2) GOLD is the largest dataset for textual procedures with corresponding PDDL definitions;

3) Using most metrics, DIGGER is the best model among all compared models on GOLD.

2 Backgrounds

2.1 Planning with Large Language Models

We are particularly interested in three common hybrid LLM-symbolic planning approaches (Huang et al., 2025). First, reasoning with LLM on simulation domains (Zhao et al., 2024). E.g., iteratively constructs and refines plans as the response to feedback (Zhang et al., 2024a). However, LLMs' probabilistic nature might induce errors like snowballing (Sukai Huang and Cohn, 2025) on long problems (Valmeekam et al., 2024b). Second, assisting International Planning Competitions (IPC) domain planning via interfacing with planners (Li et al., 2025; Singh et al., 2025; Silver et al., 2024). However, LLMs are likely to be trained on many leaked IPC domains (Huang et al., 2025), and complete PDDL definitions are necessitated. Third, constructing PDDL domain models via NL-PDDL translations to enable classical planners (Huang et al., 2025). However, NL-PDDL translation is challenging (Stein et al., 2025; Oswald et al., 2024) and errors (Smirnov et al., 2024) are unavoidable.

However, GPT-4 (Achiam et al., 2023) only achieves 35% accuracy in generating plans for simple problems (Valmeekam et al., 2024a). Thus, trade-offs between efficiency, consistency, and scalability might be unavoidable (Shlomi et al., 2025). I.e., elevating incomplete domain models with entity-attribute and leveraging the robustness of LLMs to handle errors could be promising.

Listing 1: A DIGGER-generated (entity-attribute enhanced) action model for the exmaple in Fig. 1.

```
(:action transplant
:parameters
 (?entity - object ?location - location)
:precondition
 (and (ready_for_transplant ?entity)
 (not (at ?entity ?location)))
:effect
 (and (at ?entity ?location)
      (transplanted ?entity ?location)))
```

2.2 AI Planning and PDDL

List. 1 shows a sample PDDL action model. Formally, a lifted AI planning problem $\Pi =< \mathcal{P}, \mathcal{O}, \mathcal{A}, \mathcal{T}, \mathcal{I}, \mathcal{G} >$, where $\mathcal{S} \leftarrow \mathcal{P}_g^{\mathcal{O}}, \mathcal{T} : \mathcal{S} \times \mathcal{A} \rightarrow \mathcal{S}$ (Ghallab et al., 2016). Let $\mathcal{P}$ be the finite set of predicates that describe the world, and $\mathcal{O}$ be the finite set of objects in the world. While grounding the first-order form predicates results

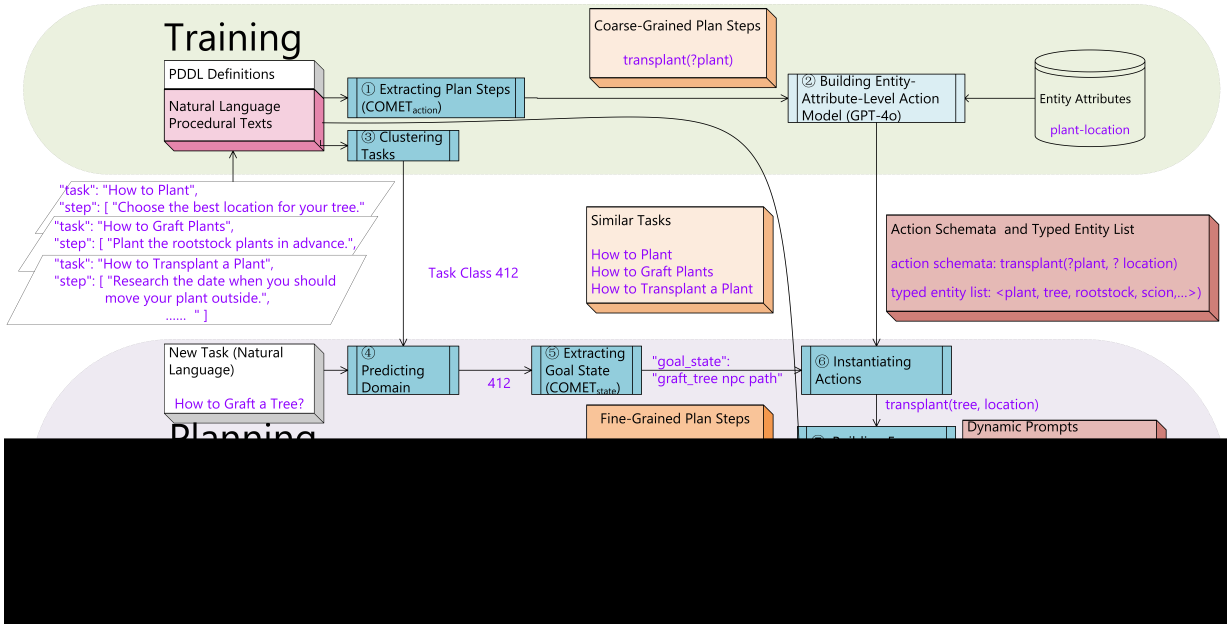

Figure 2: The overall architecture of DIGGER. Blue process blocks are numbered subprocesses (from ① to ⑨). The green oval represents the training stage, and the purple oval represents the planning stage. Red hyper-rectangles are output messages between subprocesses. White hyper-rectangles and the database icon represent external information. The input/output examples of processes are highlighted in purple.

in prepositions, we call the first-order form action definition as schemata (aka action model), and all results of grounding are noted with the subscript $_g$. Then $\mathcal{P}_g^{\mathcal{O}}$ is the result of grounding $\mathcal{P}$ with $\mathcal{O}$, i.e., the set of all states ($\mathcal{S}$). Moreover, $\mathcal{A}$ is the set of all actions, $\mathcal{I}$ is the initial state, and $\mathcal{G}$ is the goal state. Thus, a planning domain $\mathcal{D} =< \mathcal{P}, \mathcal{O}, \mathcal{A}, \mathcal{T} >$, and the grounded action space is $\mathcal{A}_g^{\mathcal{O}}$. The result plan is $\pi =< a_1, ..., a_k >$, where $a_i \in \mathcal{A}_g^{\mathcal{O}}, i \in \mathbb{N}$. In grounding actions and predicates, the parameters are objects and their attributes. E.g., names of entities are objects whose position is an attribute.

E.g., in List. 1 the action is $transplant$, which is an $a_i \in \mathcal{A}_g$. It has two parameters, which are $?entity$ and $?location$ ($?entity$, $?location \in \mathcal{O}$). The precondition and effects of this action are grounded predicates in $\mathcal{P}_g^{\mathcal{O}}$ like $(not(at\ ?entity\ ?location))$. However, due to expertise requirements, complete PDDL definitions exist only for IPC domains.

3 Large Language Models based Open Domain Natural Language Planning

LONG aims to generate a textual procedure TP for a given NL task title tsk (e.g. `"How to Graft a Tree?"`). tsk is an NL goal $\mathcal{G}$ in terms of a desired state that involves objects (e.g., `"a grafted tree"`). TP should be an executive series of (maybe unseen) grounded actions that lead to $\mathcal{G}$. I.e., TP consists of a series of NL steps $NL(a) \in NL(\mathcal{A})$, where $NL(a)$ is an NL action (a short sentence) and $NL(\mathcal{A})$ is the set of all possible actions.

Suppose $LLM(\cdot)$ is the function of LLMs, a probabilistic predictor of language tokens. I.e., given a sequence of tokens $T = (t_1, ..., t_n), n \in \mathbb{N}$ in a corpus C, LLMs (as a function $LLM(\cdot)$) will output a new (reusable) t_{n+1} and associated probabilities $P \subseteq C \times \mathbb{R}$. See Eq. 1.

$$P(T) = \prod_{i=1}^{n} P\left(t_i \mid t_{<i}\right) \tag{1}$$

The goal of LONG is to acquire TP for tsk. We first generate a PDDL plan π for tsk with a partial domain $\mathcal{D}$. Then, we prompt LLMs to generate TP. I.e., $TP \leftarrow LLM(\pi) \leftarrow LLM(\mathcal{D}, tsk)$.

4 A Two-Stage Solution for LONG

4.1 The Overall Architect of DIGGER

The architecture of DIGGER in Fig. 2 has two stages: the training stage (top, green-yellow ellipse) and the planning stage (bottom, thistle ellipse).

4.2 The Training Stage

There are three steps in the training stage: 1) training the $COMET_{action}$ model with GOLD(upper

left icon) which includes models Π and text procedures TP; 2) building entity-attribute-level action models a_i' (a skyblue sub-process) using ConceptNet (a dataset icon); 3) clustering tasks into classes.

Step ① extracts coarse-grained plan steps a_i (an apricot hyperrectangle) from NL sentences $NL(a)$. Following previous studies (Li et al., 2024; Chen et al., 2024), we fine-tune COMET (Bosselut et al., 2019). I.e., adjusting the weights in Eq. 2, with human-drafted actions in GOLD as labels. E.g., for the task "`How to Transplan a Plant`"($NL(a)$) in GOLD, "`transplan(?plant)`"(a_i) should be extracted. Because this COME translates $NL(a)$ to a_i, it is called $COMET_{action}$.

$$\theta_{ft} = \theta_{pt} - \eta \cdot \nabla_\theta L(\theta, GOLD) \quad (2)$$

We now explain symbols in Eq. 2. θ_{ft} : model parameters after fine-tuning. θ_{pt} : parameters obtained from pre-training. η: the learning rate. L: the loss function. $\nabla_\theta L(\theta, GOLD)$: the gradient of L with respect to the parameters θ on GOLD.

Step ② builds entity-attribute-level action models and typed entity lists. All entities' attributes (e.g., `plant-location`) in OPENPI and ConcetpNet (Speer et al., 2017) are embedded with sentence-BERT (Reimers and Gurevych, 2019) as Emb. E.g., suppose the function of sentence-BERT is $SBERT(\cdot)$, then $Emb = \{e|e \leftarrow SBERT(entity\_attribute)\}$. Emb are prestored in a vector database, FAISS (Johnson et al., 2019). Then, entity e's attributes e' are found using Eq. 3.

$$CosineSim(e, e') = \frac{f(e)f(e')}{\|f(e)f(e')\|} \quad (3)$$

Then, with related attributes, LLMs generate entity-attribute-level action models a_i'. E.g., `transplant(?plant, ?location)`. I.e., $a_i' \leftarrow LLM(a_i, e')$, and $\Pi \leftarrow LLM(\mathcal{A})$, where $\mathcal{A}$ is the set of all a_i'. Appx. E Tab. 8 illustrates the prompt. We merge all task entities and attributes into a typed list. E.g., `<plant, tree, rootstock, scion,...>`.

Step ③ clusters WikiHow tasks. We first perform the principal component analysis on embedded sentences (Raunak et al., 2020). Then, based on the silhouette score, we select 2,800 classes by k-means ($k = 2,800$) clustering (Ikotun et al., 2023). Further, we manually choose 20 classes from the top-100 results of the DBSCAN algorithm (Schubert et al., 2017). E.g., tasks "`How to Plant`", "`How to Grapft Plants`", and "`How to Transplant a Plant`" (in the left of Figure 1, in the middle of the planning and training stages) are clustered as the task class "`412`".

To prepare for the instantiation, we merge typed entity lists of tasks to form the domain's typed entity list. For unseen domains, LLMs build an extra global typed entity list.

4.3 The Planning Stage

There are six steps (blue sub-processes) in the planning stage (④∼⑨). Step ① instantiates parameters, and the $Sim(\cdot)$ function implements Eq. 3. For a new task tsk (a white hyperrectangle), step ④ predicts its domain $\hat{D}$. Step ⑤ extracts the goal state $\mathcal{G}$. With $\hat{D}$ and similar tasks, step ⑥ instantiates actions (Alg. 1). Step ⑦ prepares the few-shot prompts for step ⑧, which generates fine-grained plan steps (in PDDL) via LLM (light blue sub-processes). At last, step ⑨ outputs the textual procedure TP via LLM.

Algorithm 1 Action Instantiating Algorithm

Input: $\mathcal{G}$, the goal; $\mathcal{G}_i$: existed goals in domain D_i, where $1 \le i \le n$; TE_i : the typed entity list for D_i; the action schemata list sch_i for D_i; TE_{global}: the global typed entity list; θ^*: the threshold for unseen domain determination; k: a boundary for the number of grounded actions.

Output: $\mathcal{A}'^{\mathcal{O}}_g = \{a_1, ..., a_k\}$, grounded actions;

$\theta, sch \leftarrow \arg\max_{1 \le i \le n} \{Sim(\mathcal{G}, \mathcal{G}_i)\}, sch_i$
if $\theta > \theta^*$ **then**
  $TE \leftarrow TE_i$ // Seen domain.
else
  $TE \leftarrow TE_{global}$ // Unseen domain.
$\mathcal{A}'^{\mathcal{O}}_g \leftarrow$ instantiate all $sch^j \in sch$ with TE
if $\|\mathcal{A}'^{\mathcal{O}}_g\| > k$ **then**
  $\mathcal{A}'^{\mathcal{O}}_g \leftarrow$ top-k relevant actions
return $\mathcal{A}'^{\mathcal{O}}_g = \{a_1, ..., a_k\}$

Step ④ predicts the domain $\hat{D}$ of tsk. The criterion of this prediction is the cosine similarity of the embeddings of tsk and goals in GOLD[4]. E.g., for task "`How to Graft a Tree`", DIGGER predicts it belongs to the task class 412.

Step ⑤ extracts $\mathcal{G}$ from tsk. We fine-tuned another COMET for this task, leveraging the weight of $COMET_{action}$. As this model output goal states, it is called $COMET_{state}$. I.e., $\mathcal{G} \leftarrow$

[4]Appx. C.1 explains the selection of the threshold θ.

$COMET_{state}(tsk, \hat{D})$. E.g., for the task in step ④, we get "`graft_tree npc path`".

Step ⑥ instantiates grounded actions and provides examples in a few-shot prompt. Alg. 1 is the action instantiating algorithm which grounds first-order action schemata from step ②.

In Alg. 1, $\mathcal{G}$ is the goal of the new task, and $\mathcal{G}_i$ is the most similar goal for $\mathcal{G}$ in the training set. We use Eq. 3 to find $\mathcal{G}_i$ (and i). Line 1 records $\mathcal{G}$ and $\mathcal{G}_i$'s similarity (θ), and uses sch_i as the target sch. In line 2, θ is compared with a pre-defined threshold (as step ④) to determine whether the new task is from unseen domains. Line 3 uses the predicted domain's typed entity list as the typed entity list TE for the new task. Otherwise, we use the global typed entity list as TE (line 5). Then, in line 6, we instantiate sch as in previous studies (Höller et al., 2020; Savaş et al., 2016) with TE. However, to control the size of the action space, we set k as the upper bound of the number of instantiated actions. If the number of instantiated actions exceeds k, we ask LLMs to select the top-k actions for us (line 8).

E.g., with the output of step ② and step ⑤, Alg. 1 generates fine-grained grounded actions like "`transplant(tree, location)`". Further, the prompt for top-k relevant grounded actions selection is shown in Appx. E as Tab. 9.

Step ⑦ builds a few-shot prompts for LLMs. First, it locates (three) similar tasks (e.g., the tasks in step ③) of tsk as examples of the few-shot prompts. Second, step ⑥-generated instantiated actions (i.e., the output of Alg. 1) are used for the action set part in the prompts.

Step ⑧ generates a plan π in PDDL for tsk by LLM. I.e., $\pi \leftarrow LLM(\Pi_{\hat{D}}, \mathcal{G}, \pi')$. Tab. 10 in Appx. 10 exemplifies the prompt. In Tab. 10, there are three sections: instruction, input, and output. While the instructions after leading hints are fixed, the task, goal state, domain file, steps so far, and examples for reference are dynamically built. I.e., in Tab. 10, texts in curly brackets are variables that change from case to case. E.g., the output of Alg. 1 is a plan step in this example.

Step ⑨ translates a plan π in PDDL to a textual procedure via LLMs. I.e., $TP \leftarrow LLM(\pi)$. E.g., it translates PDDL actions to NL sentences like "`Cut the scion`". Inspired by recent advances (Wang et al., 2024), we choose o1-mini[5] as our component. Appx. G contains more details about the example in Fig. 2.

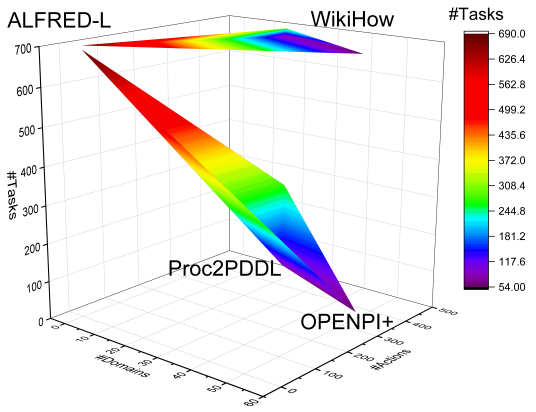

Figure 3: Statistics of the tasks from different source sub-datasets of GOLD. #: number of. X-axis: # domains, Y-axis: # tasks, Z-axis:# actions.

5 Experiments

5.1 Experiment Settings

The experiment settings are in Appx. A.1 Tab. 4, and the computation costs are in Appx. D.

5.2 Datasets

The GOLD domain-task-action relationship is quantified in Fig. 3. In Fig. 3, the X-axis displays the number of domains (total: 103), the Y-axis represents the number of tasks (total: 1100), and the Z-axis shows the number of actions (total: 1086). Our experiment used an 8:1:1 split of GOLD for training, validation, and testing.

5.2.1 WikiHow

WikiHow (Koupaee and Wang, 2018) is a dataset of textual procedures. It contains tutorials that describe the steps to achieve tasks related to diversity topics. We adhere to previous studies (Lu et al., 2023; Guo et al., 2024) and select 271 tasks in 20 domains with 434 actions from WikiHow.

5.2.2 ALFRED-L

ALFRED (Action Learning From Realistic Environments and Directives) is a human-annotated household planning domain (Shridhar et al., 2020). It is a benchmark for robot action planning, whose subset is used in previous studies and is named ALFRED-L (Lin et al., 2023). There is only one domain and 11 actions in ALFRED-L, but the number of tasks is 690 (i.e., all tasks in the test set from ALFRED-L are from the same seen domain).

[5] https://openai.com/index/openai-o1-mini-advancing-cost-efficient-reasoning/

5.2.3 OPENPI+

OPENPI (OPEN Procedural Inference) (Zhang et al., 2024b) tracks entity state changes in procedural texts selected from WikiHow. OPENPI+ has 55 domains, 286 actions, and 55 tasks. We draft PDDL domain definitions for all tasks, and the results are verified by the Fast Downward planner (Helmert, 2006). As in previous studies (Zhang et al., 2024c), three graduate students with prior knowledge of PDDL are recruited as annotators. Because there is only one task for a domain from OPENPI+, it cannot be included in the training set if it is also in the test set. I.e., OPENPI+ domains are unseen test domains in our experiments. Previous approaches cannot solve these unseen test domains without adaptation, as there are no plan traces[6].

5.2.4 Proc2PDDL

Proc2PDDL (Zhang et al., 2024c) contains open-domain procedural texts and their corresponding human-vetted PDDL representations, which have 27 domain files, 355 actions, and 84 problem files. However, to control the plan length, Proc2PDDL divided tasks into subtasks, using the original task names as the names of all subtasks.

5.3 Compared LLM-based Planning Methods

Besides DeepSeek (Bi et al., 2024), GPT-4[7] and GPT-4o[8], the compared models are as follows. Prior IPC or simulation-based studies are excluded because our targets are real-world open domains.

5.3.1 Open Grounded Planning

OGP (Guo et al., 2024) aims to generate an executable plan based on a variable action set. It requires LLMs to retrieve and rewrite plans within pre-collected executable action sets to plan for various task fields. Thus, training OGP with GOLD is difficult because the textual procedure and domain pairs in GOLD are often one-on-one.

5.3.2 PLAN

PLAN (neuro-symbolic procedural PLANner) (Lu et al., 2023) uses symbolic program executors on the latent procedural representations for planning. It only utilizes available steps in a specific domain (e.g., selected tasks from WikiHow) and is therefore inapplicable to unseen domains, relying on external knowledge graphs. As most existing LONG methods that fine-tune LLMs with planning traces[9], PLAN cannot be trained on GOLD.

Sub-Dataset	Model			
	LLM	OGP[1]	PLAN	Ours
ALFRED-L	✓	×	×	✓
OPENPI+	✓	✓	✓	✓
Proc2PDDL	✓	×	×	✓
WikiHow	✓	✓[2]	✓	✓

[1] OGP: Open-Grounded-Planning.
[2] OGP/PLAN can only solve selected tasks.

Table 1: Solving ability of models on datasets.

5.3.3 Solving Ability of Models on Datasets

We summarize the solving ability of models in Tab. 1. It indicates that OGP and PLAN are domain-specific, as they cannot handle unseen domains.

5.4 Metrics

Although LLMs might check soundness symbolically (Katz et al., 2024), evaluating LONG is difficult. Previous assessments focused on plan validity in IPC domains (Stechly et al., 2024; Kambhampati et al., 2024). However, the assessments of validity require complete formal definitions and simulation software. E.g., the success rate (SR) depends on the simulator. In this paper, we argue that multi-dimensional evaluation is beneficial. We used reference-based metrics, LaJ, and human evaluation. LaJ relies on biased LLMs, and human judgments are subjective. However, they improve upon reference-based metrics. These metrics provide a comprehensive evaluation of plan quality, regardless of unmet goals, without requiring formal definitions or simulations.

5.4.1 Reference-based Automatic Evaluation

This paper utilizes four popular automated reference-based metrics. First, BLEU (BiLingual Evaluation Understudy) (Papineni et al., 2002). BLEU measures textual similarity according to N-gram overlaps. Second, ROUGE-L (Recall-Oriented Understudy for Gisting Evaluation) (Lin, 2004). ROUGE-L considers the length of the longest common sequences between sentences. Third, METEOR (Metric for Evaluation of Translation with Explicit ORdering) (Banerjee and Lavie, 2005). METEOR aims to align human

[6] We adapt the OGP without the plans in the test set and the results are in Appx. C.3.
[7] Version: gpt-4-turbo-2024-04-09.
[8] Version: gpt-4o-2024-08-06.
[9] The planning traces are state-action-state sequences only generated in simulated environment-oriented or IPC domains.

Model		Reference-based Automatic Evaluations									LaJ	Human	
		B-1[3]	B-2	B-3	B-4	M	R-1	R-2	R-L	BS	WTLR[4]	SR↑	NS↑
LLM	DeepSeek[1]	0.072	0.029	0.013	0.006	0.177	0.129	0.018	0.124	-0.082	72.2	88.6	0.638
	GPT-4	0.089	0.037	0.015	0.007	0.188	0.141	0.018	0.135	0.001	71.4	77.1	0.599
	GPT-4o	0.068	0.027	0.011	0.005	0.173	0.115	0.015	0.109	-0.136	58.7	82.9	0.614
Other	OGP	0.385	0.293	**0.252**	**0.230**	**0.395**	0.427	0.274	0.418	0.401	47.6	**90.0**	0.650
	PLAN	0.020	0.005	0.003	0.002	0.043	0.035	0.002	0.033	0.005	**100**	0.0	0.070
Ours	DI[2]	**0.408**	**0.303**	0.245	0.207	0.359	**0.534**	**0.336**	**0.528**	**0.407**	N/A	88.6	**0.656**
	w/o C	0.344	0.235	0.173	0.128	0.287	0.460	0.213	0.453	0.377	44.4	88.6	0.626
	w/o I	0.387	0.289	0.231	0.189	0.343	0.519	0.315	0.512	0.383	40.5	82.9	0.603
	w/o E	0.357	0.250	0.187	0.146	0.322	0.475	0.249	0.468	0.371	50.8	77.1	0.600

[1] D: Deepseek (https://www.deepseek.com/), G4: GPT-4, Go: GPT-4o, OGP: Open-Grounded-Planning (Guo et al., 2024), PLAN: see Lu et al. (2023). As most tasks can not be solved by OGP and PLAN, we only report the results on the WikiHow sub-dataset. Further, many PLAN generated plans are empty.
[2] DI: DIGGER, w/o C: without $COMET_{action}$, w/o I: without instantiation, w/o E: without Entity-Attributes.
[3] B: BLEU, M: METEOR, BS: BERTScore, R: ROUGE.
[4] WTLR: (%)↑.

Table 2: Comparisons of models, and ablation studies of DIGGER on GOLD. N/A: not applicable; LaJ: LLM-as-a-judge; WTLR: win-to-loss ratio defined by Eq. 4; SR: Success Rate; NS: Normalized human evaluation Likert scale score (Eq. 5)); w/o: without. The best and second-best results are boldfaced and underlined, respectively.

evaluation on the closeness of words. Fourth, BERTScore (Zhang et al., 2020). BERTScore matches words in candidate and reference sentences with the cosine similarity of embeddings.

5.4.2 LLM-as-a-Judge (LaJ)

LaJ is a new paradigm of automated evaluation in the era of LLMs. Compared with human consensus, it can achieve over 80% agreement (Zheng et al., 2023). Following recent studies (Zhou et al., 2024), we use o1-mini to judge the WTLR (win-to-loss ratio). WTLR aims to compare plans generated by two different models (Lu et al., 2023). However, unlike previous finance studies (Huang et al., 2019), we define WTLR as eqref eq:WTLR. In Eq. 4 the number of comparisons our model wins is $nWin$, and the number of total comparisons is $nTotal$.

$$WTLR = \frac{nWin}{nTotal} \tag{4}$$

Example prompts are in Appx. E.4 Tab. 11. While fixed instructions are next to the leading hints, we dynamically fill the task and compare plans. As a result, Appx. C.2 reports the correlation between human judgments and LaJ.

5.4.3 Human Evaluation

While LaJ is attractive, human validators remain indispensable for ensuring the correctness of planning applications (Vishal Pallagani and Srivastava, 2025). Thus, in light of prior research (Guo et al., 2024; Lu et al., 2023), we recruit human annotators to evaluate plans' SR, which denotes the likelihood of agents completing tasks. Furthermore, we ask the annotators to assign Likert scale scores to the plans. The score s is normalized to [0,1] as Eq. 5, where $s_{max} = 5$ and $s_{min} = 1$. We abbreviate this normalized Likert scale score as NS.

$$NS = \frac{s - s_{min}}{s_{max} - s_{min}} \tag{5}$$

Following previous studies (Zhang et al., 2024c), three graduate students are recruited as human evaluators to evaluate samples (ten from each sub-dataset). The Krippendorff's coefficient (Krippendorff, 2018) (inter-annotator agreement) among three annotators is required to satisfy $\alpha \geq 0.80$. We document details of the human evaluation in Appx. E.5 Tab. 12.

5.5 Comparison Between Models

The experiment results comparing different models are presented in the upper part of Table ref table:modelCompare. We divide the compared models into two classes: three LLMs and two recently proposed models. We use the metrics and their category in §5.4 in Tab. 2. While the second category only used the WTLR, we performed SR and NS in human evaluations as compensation.

We draw two conclusions from Tab. 2. First, in terms of reference-based metrics, DIGGER is as effective as non-generalizable retrieve and rewrite methods. Second, on both LaJ and human evaluations, LLMs exhibit advantages over existing methods, but DIGGER stably improved the performance. I.e., it doubles the WTLR of OGP and is much better than PLAN on SR. E.g., as the result of PLAN is the worst according to LLMs, the LaJ score for PLAN is the largest. However, the LLM-favor effect might stem from a shared human/LLM preference for interestingness over accuracy.

Sub-Dataset	Reference-based Automatic Evaluations				LaJ	Human	
	BLEU-1	METEOR	ROUGE-L	BERTScore	WTLR[1]↑	SR↑	NS↑
ALFRED-L	**0.541**	**0.477**	**0.789**	**0.578**	**100**[2]	**100**	**0.733**
OPENPI+	0.310	0.240	0.247	0.223	0	83.3	0.628
Proc2PDDL	0.073	0.077	0.122	0.025	33.3	66.7	0.544
WikiHow	0.277	0.242	0.226	0.233	4.76	100	0.697

[1] %, vs. GPT-4o.
[2] The prime consideration of our WTLR definition (Eq. 4) is that there is no loss case in the ALFRED-L subdataset.

Table 3: Performance of DIGGER on four sub-datasets of GOLD. See Tab. 2 for the abbreviations.

5.6 Ablation Study

The lower part of Tab. 2 reports the ablation study of DIGGER on GOLD. We can rank the efficiency of components of DIGGER as: $COMET_{action}$ > Entity-Attributes > Instantiation, based on Tab. 2. There are three observations. First, the effectiveness of $COMET_{action}$ witnesses the power of PDDL planning. Second, for reference-based metrics, the Entity-Attributes is a double-edged sword. I.e., it introduces extra names (entity attributes that are ignored in references), which might decrease the score. Third, Instantiation is crucial for LaJ and human evaluation.

5.7 Results on Sub-Datasets

Tab. 3 reports the performance of DIGGER on different source sub-datasets. The results lead to three observations. First, for tasks from seen domains (e.g., ALFRED-L), we can observe that the more examples are provided, the better. Second, entity-attributes make a difference in tasks from unseen domains. Lacking domain models (e.g., OPENPI+ tasks), similar cross-domain tasks are helpful due to the presence of similar cross-domain entities. Third, we believe the LaJ in Tab. 3 is untrustworthy for the self-preference bias of LLMs (Gulcehre, 2024; Shimabucoro et al., 2024; Panickssery et al., 2024; Dai et al., 2024). It is also possible that the procedural knowledge of some open-world tasks (e.g., from WikiHow) might already be (partially) learned by LLMs. See Appx. C.2 for more analysis between human judgments and LaJ. As previous studies (Zheng et al., 2023), Appx. C.2 Tab. 7 shows that LaJ achieve an 80% agreement with human.

5.8 Qualitative Examples

Fig. 4 shows the result plan with entity-attribute for Fig. 1. As a comparison, plans for `" How to Graft a Tree"` that are generated by OGP and GPT-4o are shown in Appx. F Fig. 5∼Fig. 6. Comparing Fig. 1 and 4, we can see

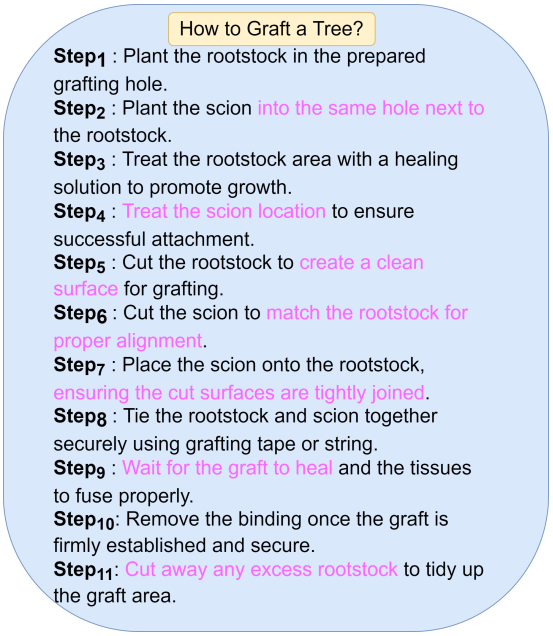

Figure 4: The plan for "How to Graft a Tree" that is generated with entity-attribute-level information by DIGGER. Details added by DIGGER are in magenta.

the advantages of DIGGER. First, there are more intermediate steps in Fig. 4, e.g., $Step_4$ and $Step_6$. Second, the plan clarifies details about the positions of objects (e.g., `"same hole next to"`), and constraints of actions (e.g., `"tightly joined"`). As a comparison, no other models pay attention to the positions of objects. Third, our model fills in the missing details of obscure verbs. E.g., the gray verb `"look after "`is extended as `"cut away any excess rootstock"` .

6 Conclusion

While there are many ways to utilize AI planning, this paper will hopefully spark further developments in LONG. First, considering that a high-quality PDDL definition is costly and unavailable under most circumstances, DIGGER leverages reusable, incomplete domain models with entity-attribute-level information to identify the hidden, fine-grained state information in intermediate

states. Second, benefit from the fact that LLMs can tolerate NL-PDDL translation-induced errors; extra validation mechanisms via (incomplete) PDDL definitions boost the performance of LONG. Third, as the most extensive dataset for textual procedures with corresponding PDDL definitions, GOLD is indispensable for LONG.

7 Limitations

Despite our best efforts, our study may still have at least eight limitations.

- Due to our limited resources, we have been able to perform our experiments on open source LLMs no larger than BART.
- One foreseeable limitation of our work is the dependency of the fine-tuning process.
- LLMs-generated augmentations are only silver-standard. The human evaluation is necessary as a benchmarking tool.
- Because the stability of LLMs is out of scope (of this paper), all LLMs involved in experiments are just a single run.
- We only consider the English language, thus limiting the study of LLM-based planning to these cultures and languages/dialects.
- The annotated procedures are still insufficient.
- Existing entity-attribute information in OPENPI+ and ConceptNet is not enough.
- Better automated evaluation metrics for LONG are absent.

Addressing these limitations will be the focus of our future work.

8 Ethical Considerations

First, licenses. All four sub-datasets of GOLD except WikiHow have annotated PDDL definitions and use the MIT License code. WikiHow uses the Attribution-Noncommercial-Share Alike 3.0 Creative Commons license. Second, safety prompts. The proposed prompts do not involve collecting or using personal information to train other individuals. Furthermore, as illustrated in Appx. E, our prompts would not compromise the safety of others. Third, reinforcing LM biases. Aligning AI systems with humans requires utmost sensitivity. If our study is applied in the real world, it might strengthen LM biases by encouraging inappropriate actions.

References

Josh Achiam, Steven Adler, Sandhini Agarwal, Lama Ahmad, Ilge Akkaya, Florencia Leoni Aleman, Diogo Almeida, Janko Altenschmidt, Sam Altman, Shyamal Anadkat, et al. 2023. Gpt-4 technical report. *arXiv preprint arXiv:2303.08774*.

Satanjeev Banerjee and Alon Lavie. 2005. METEOR: An automatic metric for MT evaluation with improved correlation with human judgments. In *Proceedings of the ACL Workshop on Intrinsic and Extrinsic Evaluation Measures for Machine Translation and/or Summarization*, pages 65–72, Ann Arbor, Michigan. Association for Computational Linguistics.

Xiao Bi, Deli Chen, Guanting Chen, Shanhuang Chen, Damai Dai, Chengqi Deng, Honghui Ding, Kai Dong, Qiushi Du, Zhe Fu, et al. 2024. Deepseek llm: Scaling open-source language models with longtermism. *arXiv preprint arXiv:2401.02954*.

Antoine Bosselut, Hannah Rashkin, Maarten Sap, Chaitanya Malaviya, Asli Celikyilmaz, and Yejin Choi. 2019. Comet: Commonsense transformers for automatic knowledge graph construction. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics*, pages 4762–4779.

Jiali Chen, Yi Cai, Ruohang Xu, Jiexin Wang, Jiayuan Xie, and Qing Li. 2024. Deconfounded emotion guidance sticker selection with causal inference. In *Proceedings of the 32nd ACM International Conference on Multimedia*, pages 3084–3093.

Sunhao Dai, Chen Xu, Shicheng Xu, Liang Pang, Zhenhua Dong, and Jun Xu. 2024. Bias and unfairness in information retrieval systems: New challenges in the llm era. In *Proceedings of the 30th ACM SIGKDD Conference on Knowledge Discovery and Data Mining*, pages 6437–6447.

Malik Ghallab, Adele Howe, Craig Knoblock, Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, and David Wilkins. 1998. Pddl-the planning domain definition language.

Malik Ghallab, Dana Nau, and Paolo Traverso. 2016. *Automated planning and acting*. Cambridge University Press.

Caglar Gulcehre. 2024. Self-recognition in language models. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 12032–12059. Association for Computational Linguistics.

Shiguang Guo, Ziliang Deng, Hongyu Lin, Yaojie Lu, Xianpei Han, and Le Sun. 2024. Open grounded

planning: Challenges and benchmark construction. *Preprint*, arXiv:2406.02903.

Malte Helmert. 2006. The fast downward planning system. *Journal of Artificial Intelligence Research*, 26:191–246.

Daniel Höller, Gregor Behnke, Pascal Bercher, Susanne Biundo, Humbert Fiorino, Damien Pellier, and Ron Alford. 2020. Hddl: An extension to pddl for expressing hierarchical planning problems. In *Proceedings of the AAAI conference on artificial intelligence*, volume 34, pages 9883–9891.

Jing-Zhi Huang, William Huang, and Jun Ni. 2019. Predicting bitcoin returns using high-dimensional technical indicators. *The Journal of Finance and Data Science*, 5(3):140–155.

Sukai Huang, Nir Lipovetzky, and Trevor Cohn. 2025. Planning in the dark: Llm-symbolic planning pipeline without experts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 26542–26550.

Abiodun M Ikotun, Absalom E Ezugwu, Laith Abualigah, Belal Abuhaija, and Jia Heming. 2023. K-means clustering algorithms: A comprehensive review, variants analysis, and advances in the era of big data. *Information Sciences*, 622:178–210.

Jeff Johnson, Matthijs Douze, and Hervé Jégou. 2019. Billion-scale similarity search with gpus. *IEEE Transactions on Big Data*, 7(3):535–547.

Subbarao Kambhampati, Karthik Valmeekam, Lin Guan, Mudit Verma, Kaya Stechly, Siddhant Bhambri, Lucas Paul Saldyt, and Anil B Murthy. 2024. Position: Llms can't plan, but can help planning in llm-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Michael Katz, Harsha Kokel, Kavitha Srinivas, and Shirin Sohrabi Araghi. 2024. Thought of search: Planning with language models through the lens of efficiency. *Advances in Neural Information Processing Systems*, 37:138491–138568.

Mahnaz Koupaee and William Yang Wang. 2018. Wikihow: A large scale text summarization dataset. *Preprint*, arXiv:1810.09305.

Klaus Krippendorff. 2018. *Content analysis: An introduction to its methodology*. Sage publications.

Ruiqi Li, Leyang Cui, Songtuan Lin, and Patrik Haslum. 2024. Naruto: Automatically acquiring planning models from narrative texts. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20194–20202.

Siyuan Li, Feifan Liu, Lingfei Cui, Jiani Lu, Qinqin Xiao, Xirui Yang, Peng Liu, Kewu Sun, Zhe Ma, and Xun Wang. 2025. Safe planner: Empowering safety awareness in large pre-trained models for robot task planning. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 39, pages 14619–14627.

Bill Yuchen Lin, Chengsong Huang, Qian Liu, Wenda Gu, Sam Sommerer, and Xiang Ren. 2023. On grounded planning for embodied tasks with language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 37, pages 13192–13200.

Chin-Yew Lin. 2004. Rouge: A package for automatic evaluation of summaries. In *Annual Meeting of the Association for Computational Linguistics*.

Daxin Liu and Jens Claßen. 2025. On action theories with iterable first-order progression. *Proceedings of the AAAI Conference on Artificial Intelligence*, 39(14):15041–15048.

Yujie Lu, Weixi Feng, Wanrong Zhu, Wenda Xu, Xin Eric Wang, Miguel Eckstein, and William Yang Wang. 2023. Neuro-symbolic procedural planning with commonsense prompting. In *The Eleventh International Conference on Learning Representations*.

James Oswald, Kavitha Srinivas, Harsha Kokel, Junkyu Lee, Michael Katz, and Shirin Sohrabi. 2024. Large language models as planning domain generators. *Proceedings of the International Conference on Automated Planning and Scheduling*, 34(1):423–431.

Arjun Panickssery, Samuel Bowman, and Shi Feng. 2024. Llm evaluators recognize and favor their own generations. *Advances in Neural Information Processing Systems*, 37:68772–68802.

Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. 2002. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th Annual Meeting of the Association for Computational Linguistics, July 6-12, 2002, Philadelphia, PA, USA*, pages 311–318. ACL.

Vikas Raunak, Vaibhav Kumar, Vivek Gupta, and Florian Metze. 2020. On dimensional linguistic properties of the word embedding space. In *Proceedings of the 5th Workshop on Representation Learning for NLP*, pages 156–165.

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.

Emre Savaş, Maria Fox, Derek Long, and Daniele Magazzeni. 2016. Planning using actions with control parameters. In *ECAI 2016*, pages 1185–1193. IOS Press.

Enrico Scala and Mauro Vallati. 2021. Effective grounding for hybrid planning problems represented in pddl+. *The Knowledge Engineering Review*, 36:e9.

Patrícia Schmidtová, Saad Mahamood, Simone Balloccu, Ondřej Dušek, Albert Gatt, Dimitra Gkatzia, David M Howcroft, Ondřej Plátek, and Adarsa Sivaprasad. 2024. Automatic metrics in natural language generation: A survey of current evaluation practices. In *Proceedings of the 17th International Natural Language Generation Conference*, pages 557–583.

Erich Schubert, Jörg Sander, Martin Ester, Hans Peter Kriegel, and Xiaowei Xu. 2017. Dbscan revisited, revisited: why and how you should (still) use dbscan. *ACM Transactions on Database Systems (TODS)*, 42(3):1–21.

Luísa Shimabucoro, Sebastian Ruder, Julia Kreutzer, Marzieh Fadaee, and Sara Hooker. 2024. Llm see, llm do: Leveraging active inheritance to target non-differentiable objectives. In *Proceedings of the 2024 Conference on Empirical Methods in Natural Language Processing*, pages 9243–9267.

Eliezer Shlomi, Guy Azran, Eilam Shapira, Omer Nahum, Guy Uziel, Michael Katz, Ateret Anaby Tavor, Roi Reichart, and Sarah Keren. 2025. Transition function prediction in ai planning using llms. In *AAAI 2025 Workshop LM4Plan*.

Mohit Shridhar, Jesse Thomason, Daniel Gordon, Yonatan Bisk, Winson Han, Roozbeh Mottaghi, Luke Zettlemoyer, and Dieter Fox. 2020. Alfred: A benchmark for interpreting grounded instructions for everyday tasks. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 10740–10749.

Tom Silver, Soham Dan, Kavitha Srinivas, Joshua B Tenenbaum, Leslie Kaelbling, and Michael Katz. 2024. Generalized planning in pddl domains with pretrained large language models. In *Proceedings of the AAAI Conference on Artificial Intelligence*, volume 38, pages 20256–20264.

Chandan Kumar Singh, Devesh Kumar, Vipul Sanap, and Rajesh Sinha. 2025. Llm-rspf: Large language model-based robotic system planning framework for domain specific use-cases. In *2025 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV)*, pages 7277–7286. IEEE.

Pavel Smirnov, Frank Joublin, Antonello Ceravola, and Michael Gienger. 2024. Generating consistent pddl domains with large language models. *arXiv preprint arXiv:2404.07751*.

Robyn Speer, Joshua Chin, and Catherine Havasi. 2017. Conceptnet 5.5: An open multilingual graph of general knowledge. In *Proceedings of the AAAI conference on artificial intelligence*, volume 31.

Kaya Stechly, Karthik Valmeekam, and Subbarao Kambhampati. 2024. Chain of thoughtlessness? an analysis of cot in planning. In *The Thirty-eighth Annual Conference on Neural Information Processing Systems*.

Katharina Stein, Daniel Fišer, Jörg Hoffmann, and Alexander Koller. 2025. Automating the generation of prompts for llm-based action choice in pddl planning. In *Proceedings of the 2025 International Conference on Automated Planning and Scheduling*.

Nir Lipovetzky Sukai Huang and Trevor Cohn. 2025. Chasing progress, not perfection: Revisiting strategies for end-to-end llm plan generation. In *Proceedings of the 2025 International Conference on Automated Planning and Scheduling*.

Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. 2024a. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change. *Advances in Neural Information Processing Systems*, 36.

Karthik Valmeekam, Kaya Stechly, and Subbarao Kambhampati. 2024b. Llms still can't plan; can lrms? a preliminary evaluation of openai's o1 on planbench. *arXiv preprint arXiv:2409.13373*.

Bharath Chandra Muppasani Vishal Pallagani, Nitin Gupta and Biplav Srivastava. 2025. Revisiting llms in planning from literature review: a semi-automated analysis approach and evolving categories representing shifting perspectives. In *Proceedings of the 2025 International Conference on Automated Planning and Scheduling*, United States. AAAI press. 35th International Conference on Automated Planning and Scheduling, ICAPS 2025 ; Conference date: 09-11-2025 Through 14-11-2025.

Kevin Wang, Junbo Li, Neel P Bhatt, Yihan Xi, Qiang Liu, Ufuk Topcu, and Zhangyang Wang. 2024. On the planning abilities of openai's o1 models: Feasibility, optimality, and generalizability. *arXiv preprint arXiv:2409.19924*.

Lei Wang, Wanyu Xu, Yihuai Lan, Zhiqiang Hu, Yunshi Lan, Roy Ka-Wei Lee, and Ee-Peng Lim. 2023. Plan-and-solve prompting: Improving zero-shot chain-of-thought reasoning by large language models. In *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 2609–2634.

Li Zhang, Peter Jansen, Tianyi Zhang, Peter Clark, Chris Callison-Burch, and Niket Tandon. 2024a. Pddlego: Iterative planning in textual environments. *arXiv preprint arXiv:2405.19793*.

Li Zhang, Hainiu Xu, Abhinav Kommula, Chris Callison-Burch, and Niket Tandon. 2024b. 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.

Li Zhang, Hainiu Xu, Yue Yang, Shuyan Zhou, Weiqiu You, Manni Arora, and Chris Callison-Burch. 2023. Causal reasoning of entities and events in procedural texts. *Findings of the Association for Computational Linguistics: EACL 2023*.

Hardware/Software/Model	Setting
OS	Ubuntu 20.04.1 LTS
CPU	Intel Core i9-10900K
GPU	RTX 6000*4
Python	3.10.13
PyTorch	2.4.0
Sentence Embedding Tool	SBERT [1]
Vector Database	Faiss-gpu 1.5.3[2]
Action/State Extraction Model	COMET[3]
Loss function	Cross-entropy
Train batch size	8
Epoch	3
Learning rate	1e-4
Learning rate schedule (warmup)	Linear decay
Fp-16	False
Early stopping	False

[1] https://www.sbert.net/,all-MiniLM-L6-v2.
[2] https://pypi.org/project/faiss-gpu/1.5.3/.
[3] https://github.com/atcbosselut/comet-commonsense.

Table 4: Experiment settings.

Tianyi Zhang, Varsha Kishore, Felix Wu, Kilian Q Weinberger, and Yoav Artzi. 2020. Bertscore: Evaluating text generation with bert. In *International Conference on Learning Representations*.

Tianyi Zhang, Li Zhang, Zhaoyi Hou, Ziyu Wang, Yuling Gu, Peter Clark, Chris Callison-Burch, and Niket Tandon. 2024c. Proc2pddl: Open-domain planning representations from texts. *Preprint*, arXiv:2403.00092.

Zirui Zhao, Wee Sun Lee, and David Hsu. 2024. Large language models as commonsense knowledge for large-scale task planning. *Advances in Neural Information Processing Systems*, 36.

Lianmin Zheng, Wei-Lin Chiang, Ying Sheng, Siyuan Zhuang, Zhanghao Wu, Yonghao Zhuang, Zi Lin, Zhuohan Li, Dacheng Li, Eric Xing, et al. 2023. Judging llm-as-a-judge with mt-bench and chatbot arena. *Advances in Neural Information Processing Systems*, 36:46595–46623.

Zhehua Zhou, Jiayang Song, Kunpeng Yao, Zhan Shu, and Lei Ma. 2024. Isr-llm: Iterative self-refined large language model for long-horizon sequential task planning. In *2024 IEEE International Conference on Robotics and Automation (ICRA)*, pages 2081–2088. IEEE.

A Experiment Settings and Datasets

A.1 Experiment Settings

Tab. 4 lists the settings and implement details of our experiments.

B Abbreviations

Tab. 5 lists the abbreviations and mathematical symbols used in our paper.

C More Experiments

C.1 Theta for Unseen Domains

We sample 56 plans from WikiHow to determine the threshold for determining unseen domains. Of these 56 domains, 42 plans are from seen domains, and 14 are from unseen domains. We summarize the results in Tab. 6. The average similarity of plans from seen domains is 0.4110, with a standard deviation of 0.0937, and a 95% confidence interval of [0.3619, 0.4601]. The average similarity of plans from unseen domains is 0.5957, the standard deviation is 0.1005, and the 95% confidence interval is [0.5653, 0.6261]. Thus, we use 0.4601 in our experiments.

C.2 Correlation between Human Judgments and LaJ

We compare human judgments and LaJ by asking humans the same question for LLMs, and the result is in Tab. 7. In this experiment, we sample five tasks from each subdataset.

Suppose there are two plans π_1 and π_2 that are generated by different models for a task. Let J_1 be "π_1 is better than π_2", J_2 be "π_1 is as good as π_2", and J_3 be "π_2 is better than π_3". Then, if the LaJ human both believe J_1, we say the judgments are consistent. In another case, we say the judgments are inconsistent. Divide the number by the total number of sampled tasks, and we obtain the ratio of consistency.

Tab. 7 shows that LaJ achieve an 80% agreement with human. By contrast, LaJ in Tab. 3 is untrustworthy. E.g., for the example in Fig. 4, our model's plan in Fig. 1 is superior to GPT-4o's in Fig. 5. However, according to GPT-4o, the GPT-4o-generated plan is better than ours.

C.3 Open-Ground-Planning in Unseen Domains

We adapt the OGP approach without the plans in the test set, and the results are as follows: the BLEU-1 score is 0.2513, the METEOR score is 0.2285, the ROUGE-L accuracy is 0.2210, and the BERTScore score is 0.2157. Comparing the values in Tab. 2, we can see that although OGP can be adapted to serve unseen domains, the performance is not as good as ours.

D Computational Cost

Regarding computational cost, $\widetilde{95}\%$ of the cost is spent on LLMs. The process involves four API

	Abbreviation or Symbol	Meaning
Abbreviation	LLM	large language models
	LONG	LLMs based open domain natural language planning
	PDDL	Planning Domain Description Language
	GOLD	natural language procedural text with PDDL definitions
	DIGGER	LLMs based open domain natural language planning by leveraging entity-attribute-level domain models
	BLEU	bilingual evaluation understudy
	NL	natural language
	ALFRED	Action Learning From Realistic Environments and Directives
	OPENPI	OPEN Procedural Inference
	OPENPI+	OPENPI with additional PDDL domain definitions
	COMET	commonsense transformers
	ROUGE	recall-oriented understudy for gisting evaluation
	METEOR	metric for evaluation of translation with explicit ordering
	LaJ	LLM-as-judge
	PLAN	the neuro-symbolic procedural planner
	SBERT	the sentence-BERT model
	FAISS	the facebook AI similarity search vector database
	DBSCAN	density-based spatial clustering of applications with noise
	#	the number of
	w/o	without
	N/A	not applicable
	OGP	Open-Grounded-Planning
	IPC	the international planning competitions
	WTLR	the win-to-loss ratio
	SR	the success rate
	NS	the Likert scale score
	CI	the confidence interval
Symbol	α	the Krippendorff's coefficient
	TE	the typed entity list
	sch	the action schemata list
	Π	the lifted AI planning problem
	P	the probability function
	$\mathcal{P}$	the finite set of predicates
	$\mathcal{O}$	the finite set of objects
	$_g$	the result of grounding
	$\mathcal{S}$	the set of states
	$\mathcal{A}$	the set of actions
	$\mathcal{T}$	the transition function
	$\mathcal{I}$	the initial state
	$\mathcal{G}$	the goal state
	$\mathcal{D}$	the planning domain
	π	the plan
	a	the action
	TP	the textual procedure
	tsk	the task
	C	the corpus
	$\mathbb{N}$	the set of all natural numbers
	$\mathbb{R}$	the set of real numbers
	θ	the parameter
	η	the learning rate
	∇	the gradient
	L	the loss function
	σ	the standard deviation

Table 5: Abbreviations and mathematical symbols used in this paper.

Domain	Avg.sim[1]	σ[2]	95% CI[3]
Seen	0.4110	0.0937	[0.3619, 0.4601]
UnSeen	0.5957	0.1005	[0.5653, 0.6261]

[1] Avg.sim : average similarity of plans.
[2] σ : standard deviation.
[3] CI : confidence interval.

Table 6: Similarity analysis of seen/unseen domains.

Sub-Dataset	DeepSeek-v3	GPT-4	GPT-4o
ALFRED-L	100 [1]	100	100
OPENPI+	60	60	80
Proc2PDDL	80	80	80
WikiHow	80	100	100

[1] %. The ratio of consistent between LaJ and humans.

Table 7: Comparison between human judgments and LaJ.

calls to LLMs per task, while the training time of the $COMET_{action}$, $COMET_{state}$, and the domain predictor is accomplished in 30 minutes. We conducted experiments with gpt-4-turbo-2024-04-09 and gpt-4o-2024-08-06 API, at a cost of $0.4 per 1 million tokens input and $1 per 1 million tokens output. We spent $300 in total. Additionally, we spent 150 minutes on training the $COMET_{action}$ and $COMET_{state}$ model.

E Prompt Examples

E.1 An Example of the Prompt for the Entity-Attribute-Level Action Model Generation

Tab. 8 is an example of the prompt for LLMs to generate entity-attribute-level action models.

E.2 An Example of the Prompt for the Top-k Relevant Grounded Actions Selection

Tab. 9 is an example of the prompt for LLMs to select the top-k relevant grounded actions.

E.3 An Example of the Prompt for the LLM-based Planning

Tab. 10 is an example of the prompt for LLM-based planning.

E.4 An Example of the Prompt for the LLM-as-a-Judge

Tab. 11 is an example of the prompt for LLM-based judgment.

E.5 An Example of the Prompt for the Human Evaluation

Tab. 12 is an example of the prompt for human evaluation.

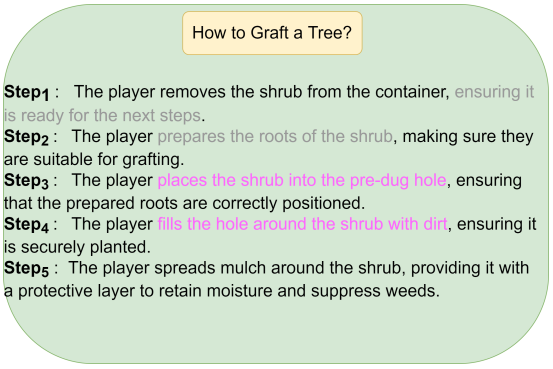

Figure 5: The plan for "How to Graft a Tree" that is generated by GPT-4o. Unspecified verbs are in gray, and additional steps and supplementary details are in magenta.

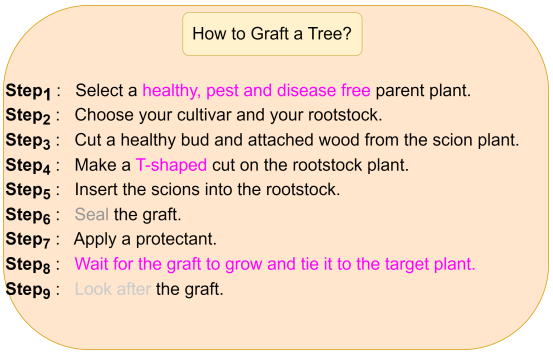

Figure 6: The plan for "How to Graft a Tree" that is generated by OGP. Unspecified verbs are in gray, additional steps and supplementary details are in magenta.

F Qualitative Examples for Comparison

As a comparison, we put the plan for "How to Graft a Tree" that is generated by OGP in Fig. 6, and Fig. 5 is the result of GPT-4o. We found that intermediate steps, such as $Step_4$ and $Step_8$, are inserted in Fig. 6. However, the positions of objects are still ignored, and details of actions are still missing (e.g., "seal" and "look after" in Fig. 6). The output of PLAN for task "How to Graft a Tree" is empty. On the other hand, While addition details like "place the shrub into the pre-dug hole" are attached to the procedure in Fig. 5, some steps are still missing, and many of the instructions are unspecified (e.g., "ready for" and "prepares").

G More Details of the Example in Fig. 2

While the steps in natural language for our new task are in Fig. 1, the generated PDDL steps for our new task are in List. 2.

INSTRUCTION:
You are a brilliant AI planning assistant, responsible for generating PDDL (Planning Domain Definition Language) actions for a given task. Your objective is to translate the task's steps into corresponding PDDL actions that capture the intent and logic of each step. Follow the example provided below and generate the appropriate PDDL actions based on the steps and their descriptions. The point of action is to consider descriptions, steps, and other states as auxiliary information.
In this section, the entity states before and after each action will be provided to help define the PDDL effects and conditions. Each state includes attributes like location, condition, or readiness of objects involved.
Instructions for Generating PDDL Actions:
For each step, generate a corresponding PDDL action that follows these guidelines:

- **Action name:** Provide a concise and descriptive name for the action.
- **Parameters:** Parameters: List the objects, agents, or items involved in this action.
- **Preconditions:** Define the conditions that must be met before the action can be performed.
- **Effect:** Specify the resulting changes or outcomes once the action is completed, based on the entity states provided.

```
Task: {{task}}
File Name: {{file_name}}
Steps:
    Step 1: {{step_1}}
    Step 2: {{step_2}}
    Step 3: {{step_3}}
    Step 4: {{step_4}}
Descriptions: [Descriptions]
Entity State:[Entity State]
```

Here is an example:
Suppose the task is:
Task:{task description}
Steps:{step description}
Descriptions:{Descriptions}
PDDL Actions:{PDDL Actions}
Here is Our Task:
Now, generate PDDL actions based on the following new task and steps:
Task:{task description}
Steps:{step description}
Entity state:{Entities in states}
Entity attribute knowledge:{entity attribute knowledge}
If a step is irrelevant to the task or description, or appears to be wrong, modify it to be relevant to the step and generate task-specific PDDL actions.Generate the PDDL actions using the format and structure shown in the example.
OUTPUT:
.....

Table 8: An example of the prompt for entity-attribute-level action model generation. Section names are in brown and text variables are in curly brackets.

Further, List. 2 is translated to natural language steps in Fig. 4.

The instantiated actions for our new task are in List. 3∼4. Similar tasks for our new task are in List. 5∼7.

Further, more action models for this domain are in List. 8∼9.

Listing 2: The PDDL steps (plan) for the new task in Fig. 2.

```
choose(tree target)
dig(agent resource hole)
place(agent item destination)
cut(agent target)
transplant(tree location)
tie(tree target)
wait(tree target)
remove(tree target)
```

INSTRUCTION:
You are an expert in AI planning and PDDL (Planning Domain Definition Language). I'll give you the names of similar objects in the domain and tasks to solve. You need to determine how these objects relate to the task name
INPUT:
Available object:{a dictionary of objects}
Target task:{name of the task}
Processing Rules:

1. Three-level correlation judgment:
    - Strong association: The following conditions must be met
        (a) The object name contains the task keyword.
        (b) The object type matches core requirements of the task.
    - Medium association: One of the following conditions is met
        (a) Objects that may be used in solving tasks.
2. Special Treatment:
    - Compound word objects are divided into separate lexical judgments.

** **Output requirements:** ** - Strictly in order of correlation strength: Strong -> Medium
Here is an example:
Available object:
'book', 'schedule', 'school', 'policy', 'opinion', 'reading', 'return', 'list', 'family', 'deal', 'loan', 'library', 'member', 'question', 'forum', 'friend', 'rule', 'acquisition', 'discussion', 'community', 'club'
Target task:
"How to Start a Book Club at School"
Output:

- Strong correlation: book club school
- Related in: member community reading library list discussion

Please strictly follow the preceding format to output the result. Do a good job of checking when you output, and do not output any objects that do not exist in [Available object], otherwise the PDDL field specification will be incorrect.
OUTPUT:
Strong correlation:
Related in:
.....

Table 9: An example of the prompt for top-k relevant grounded actions selection. Section names are in brown and text variables are in curly brackets.

INSTRUCTION:
You are a brilliant AI specialist in PDDL (Planning Domain Definition Language), tasked with generating strictly formatted PDDL planning steps for a new task.
Task Description
Define the task objectives. Select appropriate actions from the provided instantiated action set and combine them logically to generate a series of steps that achieve the goal.
Action Selection and Combination Guidance

1. **Select actions that support mission completion:** Choose only those actions that directly contribute to achieving the goal.
2. **Ensure alignment with task requirements:** Each action must be consistent with the mission requirements and drive the task toward its objective.
3. **Arrange actions in logical sequence:** Order the actions according to the complexity of the task and the dependencies between them.
4. **Consider dependencies:** Some actions may require the result or condition of a previous action. Ensure these dependencies are respected.
5. **Dynamically adjust based on progress:** Modify the sequence as necessary based on the current status of the task.
6. **Deeply consider various planning forms:** Use your self-judgment to choose the most appropriate steps that truly satisfy the task requirements.
7. **Ensure appropriateness:** When choosing an action, always consider current conditions to guarantee that each step is appropriate.
8. **Operability and specificity of the steps:** Emphasize that each step should be as specific and actionable as possible. For example, require each step to clearly identify a tool or material. This ensures that the generated steps are more realistic and avoids actions that are too abstract or vague.

Note: The provided Goal_state is for reference only and does not necessarily represent the correct or unique target state. The generated planning steps must ensure logical coherence and valid state changes, but they do not have to force the final state to exactly match the reference.
Output Requirements

1. **Strict PDDL Format:** Output only in PDDL, with no natural language descriptions or action definitions.
2. **Step Structure:** Each step must include an action name and the corresponding parameters.

Step Constraint The plan should contain a maximum of {max_steplen} steps ($\pm$1-2 steps). If the goal has been partially accomplished, the generated plan can be appropriately reduced in the number of steps. Each step should effectively change the state of the entity, be concise and actionable, and move the task toward the final goal.
INPUT:
Task:{task description}
Goal State:{goal state}
Action set:{instantiating action}
Example Workflow

1. By learning examples of similar tasks, understand how to generate target steps and refer to their generated steps.
2. Logically combine operations to build a complete PDDL plan, ensuring that the sequence of steps drives tasks toward the end goal.

Similar task:{Similar task}
Now, generate the PDDL planning step for task:{task description}
Output results in strict PDDL format. The output should contain only PDDL steps, each including the action name, corresponding parameters, and state changes. .....
OUTPUT:
.....

Table 10: An example of the prompt for LLM planning. Section names are in brown and text variables are in curly brackets.

INSTRUCTION:
You are an expert evaluator tasked with comparing two texts: Text 1 and Text 2. Your primary goal is to evaluate them based on how well they align with the task goal. The evaluation should focus on how effectively each text helps in achieving the core objective of the task, emphasizing comprehensiveness, practicality, and relevance.
Please evaluate both texts based on the following criteria, prioritizing the alignment with the task goal:
Task Alignment & Goal Focus:
Does the text clearly understand and align with the task goal?
Does the text directly address the core objective of the task without deviating or introducing irrelevant content?
Information Richness & Task Completeness:
Does the text provide sufficient details and cover all necessary steps to complete the task as per the task goal?
Does it guide the user in understanding and completing the task successfully?
Clarity of Expression & Practical Usability:
Is the text easy to understand and clearly expressed?
Does it provide actionable, practical guidance for the task goal?
Linguistic Fluency & Relevance:
Does the text flow naturally and logically without awkward phrasing?
Is the information entirely relevant to the task goal, avoiding redundant or unrelated details?
After evaluating both Text 1 and Text 2 based on these criteria, provide an overall judgment on which text performs better in terms of helping the user achieve the task goal.
INPUT:
Task: How to Graft a Tree
Gold Step: ['Choose your cultivar and your rootstock.', 'Amputate your rootstock.', 'Cleave your rootstock.', 'Prepare the scions.', 'Insert the scions into the rootstock.', 'Seal the graft.', 'Look after the graft.']
Text 1:
*Task**: How to Graft a Tree
Steps in Natural Language:
1. **Choose** the appropriate rootstock for your tree grafting project.
2. **Select** a compatible scion to join with the rootstock.
3. **Mix** the necessary soil and dig a hole in the chosen location.
4. **Place** a stake into the hole to support the grafted area.
5. **Cut** the scion to prepare it for grafting.
6. **Insert** the cut scion into the prepared hole in the rootstock.
7. **Tie** the graft securely around the insertion point to hold it in place.
8. **Wait** for the binding to secure the graft properly.
9. **Remove** the binding once the graft has stabilized.
10. **Cut** any excess branches from the rootstock to promote healthy growth.
11. **Wait** for the scion to grow and establish itself on the rootstock.
12. **Remove** any additional branches from the rootstock as needed to ensure proper development.
Text 2:
'Select a healthy, pest and disease free parent plant.'
, 'Choose your cultivar and your rootstock.'
, 'Cut a healthy bud and attached wood from the scion plant.'
, 'Make a T-shaped cut on the rootstock plant.'
, 'Insert the scions into the rootstock.'
, 'Seal the graft.'
, 'Apply a protectant.'
, 'Wait for the graft to grow and tie it to the target plant.'
, 'Look after the graft.'
OUTPUT:
Only output the result without explanatory texts.
Output format example: Final Judgment: Text 1: Win

Table 11: An example of the prompt for LaJ. The problem is "How to Graft a Tree." Text1 is the planning step generated by DIGGER, and Text2 is the planning step generated by the OGP.

Manual Evaluation Guide:

1. Evaluator Requirements:

- Three graduate students specializing in AI planning conducted independently of the evaluation. Independent scoring is mandatory during the evaluation; collaboration with reviewers and information sharing is forbidden.

2. Normalization of Scoring Criteria (0-5 points:

Reviewers must assess each of the three dimensions and round their scores to two decimal places.

Fluency: The assessment criteria consider both the fluency of expression and the logical soundness of the conclusion.

Goal Alignment: Evaluation of the degree to which generated results align with predefined task goals, including identification of any deviations.

Rationality: Determining if the solution generated is practical and aligns with the logical principles of AI planning.

3. Correctness Judgment:

- **Correct**: The generated content successfully meets all specified requirements; its method is sound, and its outcomes are demonstrably effective.
- **Error**: The generated content contains critical flaws, cannot meet objectives, and deviates from specified requirements.

4. Evaluation process:

(1). **Independent review stage** - Each reviewer must score the generated results and provide a **correctness judgment (right/wrong)**.

(2). **Summary stage**

- **Aggregating Scores**: Compute the average of the three reviewers' scores (range: 0-1).
- **Determining Correctness**: The final correctness will be determined by a majority vote (affirmation from a minimum of two reviewers).

Table 12: An example of the prompt for human evaluation. Section names are in brown and text variables are in curly brackets.

Listing 3: The instantiated actions for the new task in Fig. 2 (part 1).

```
remove(entity hole agent),
dig(agent tree hole),
dig(agent rootstock hole),
dig(agent scion hole),
dig(agent branch hole),
dig(agent root hole),
dig(agent plant hole),
dig(agent soil hole),
place(agent item destination),
choose(entity target),
drink(entity liquid pot),
test(entity tree hole),
test(entity rootstock hole),
test(entity scion hole),
test(entity branch hole),
test(entity root hole),
test(entity plant hole),
test(entity soil hole),
modify_state(actor target state1 state2)
    ,
add(agent tree hole),
add(agent rootstock hole),
add(agent scion hole),
add(agent branch hole),
add(agent root hole),
add(agent plant hole),
add(agent soil hole),
move(entity from to),
plan(agent tree),
plan(agent rootstock),
plan(agent scion),
plan(agent branch),
plan(agent root),
plan(agent plant),
plan(agent soil)
```

Listing 4: The instantiated actions for the new task in Fig. 2 (part 2).

```
slide(entity pot),
cut(agent target),
use(agent tool),
keep(entity target),
treat(tree loc),
treat(rootstock loc),
treat(scion loc),
treat(branch loc),
treat(root loc),
treat(plant loc),
treat(soil loc),
stop(agent target),
order(tree hole),
order(rootstock hole),
order(scion hole),
order(branch hole),
order(root hole),
order(plant hole),
order(soil hole),
press(entity hole item),
plant(agent item hole),
determine(agent subject context),
unpack(agent pot item),
stake(agent tree target hole),
stake(agent rootstock target hole),
stake(agent scion target hole),
stake(agent branch target hole),
stake(agent root target hole),
stake(agent plant target hole),
stake(agent soil target hole),
replant(item hole),
transplant(entity hole),
gotolocation(entity from to),
harvest(agent tree),
harvest(agent rootstock),
harvest(agent scion),
harvest(agent branch),
harvest(agent root),
harvest(agent plant),
harvest(agent soil)
```

Listing 5: The 1^{st} similar task for the new task in Fig. 2.

```
cluster_label: 412,
task: How to Plant,
step: [
Choose the best location for your tree.
Till the soil lightly with a pick or a
    shovel.
Dig a hole in the area that is twice as
    wide as the root ball of your tree.
Loosen the roots in preparation for the
    transplant.
Pull the container away from the root
    ball until it is completely exposed.
Look for roots that are circling.
Place the root ball of the tree in your
    hole.
Mix one part compost with three parts
    soil before you fill the hole around
     the tree.
Fill in the area around the root ball
    with your mixture of compost and
    soil.
Create a circular berm or small mounded
    hill.
Cover the area with a layer of mulch.
Use stakes to support the tree.
Water the area.
Enjoy your gift to the earth!.
]
```

Listing 6: The $2^{n}d$ similar task for the new task in Fig. 2.

```
cluster_label: 412,
task: How to Graft Plants,
step: [
Plant the rootstock plants in advance.
Prepare to graft the plants in cool
    weather when the scion plant is
    budding.
Make a T-shaped cut on the rootstock
    plant.
Cut a healthy bud and attached wood from
     the scion plant.
Insert the bud wood into the T cut.
Tie the plants together.
Wait for it to heal before removing the
    binding.
Cut the rootstock branch some distance
    above the new bud.
Once the bud wood has grown a few new
    leaves, remove the rest of the
    rootstock branch.
]
```

Listing 7: The $3^{r}d$ similar task for the new task in Fig. 2.

```
cluster_label: 412,
task: How to Transplant a Plant,
step: [
Research the date when you should move
    your plant outside.
Begin to harden the plant off 2 weeks
    before the transplanting date.
Plan to transplant during the cool part
    of the day.
Fill the planting bed with gardening
    soil.
Dig a hole big enough to hold the plant'
    s pot.
Turn the pot upside down and slide the
    root ball out.
Leave the plant in the pot if it's made
    from peat or paper.
Loosen the root ball with your fingers,
    if needed.
Place the root ball into the hole.
Fill the space around the root ball with
     more soil and pat it down.
Water the plant thoroughly.
]
```

Listing 8: Action models generated by our model in PDDL for the exmaple in Fig. 1.

```
(:action remove
:parameters (?entity - object ?location
    - location ?agent - agent)
:precondition (and
(at ?entity ?location)
(at ?agent ?location)
(not (removed ?entity))
)
:effect (and
(removed ?entity)
(not (at ?entity ?location))
)
)
(:action dig
:parameters (?agent - agent ?resource -
    resource ?location - location)
:precondition (and (at ?agent ?location)
(or (and (moisture ?location dry) (type
    ?resource root_center))
(and (in_bag ?resource) (type ?location
    garden))
(and (soil_removed ?resource) (type ?
    resource fir))
(and (unselected ?resource) (type ?
    location area))
(and (covering_tracks ?resource) (type ?
    resource track))
(and (empty ?resource) (type ?resource
    berm))
(and (at ?resource ?location) (type ?
    resource plant))
(or (and (not (cut ?resource)) (type ?
    resource plant))
(and (not (roots_dug ?resource)) (type ?
    resource fir))
(and (not (built_water_basin ?location))
     (type ?resource location))
(and (not (contacted_extension_office ?
    agent)) (type ?resource track))
(and (not (fuller ?resource)) (type ?
    resource berm))))
)
:effect (and (moisture ?location wet))
)
```

Listing 9: More action models generated by our model in PDDL for the exmaple in Fig. 1.

```
(:action place
:parameters (?agent - agent ?item - item
     ?destination - destination)
:precondition (and
(available ?agent)
(not (placed ?item ?destination))
(or
(and (at ?item ?destination) (not (
    in_hole ?item)))
(and (inventory ?agent ?item) (loosened
    ?item))
(and (planned ?item) (empty ?destination
    ))
(and (in_pot ?item ?destination) (not (
    in_soil ?item)))
(and (at ?item ?destination) (selected ?
    destination))
(and (at ?agent ?destination) (not (
    created ?destination)))
(and (selected ?destination) (fuller ?
    destination))
(and (at ?agent ?destination))
(and (in_hole ?item) (full_of_soil ?
    destination))
(and (at ?item ?destination) (inventory
    ?agent ?item))
)
)
:effect (and
(placed ?item ?destination)
(not (in_hole ?item))
)
)
(:action choose
:parameters (?entity - entity ?target -
    target)
:precondition (and (at ?entity ?target)
    (not (state_changed ?target)))
:effect (and (state_changed ?target))
)
(:action cut
:parameters (?agent - agent ?target -
    target)
:precondition (and (at ?agent ?target) (
    or (not (cut ?target)) (reforested ?
    target)))
:effect (and (cut ?target) (when (
    reforested ?target) (and (mowed ?
    target) (not (reforested ?target))))
    )
)
(:action gotolocation
:parameters (?entity - object ?from -
    location ?to - location)
:precondition (and (at ?entity ?from) (
    not (at ?entity ?to)))
:effect (and (at ?entity ?to) (not (at ?
    entity ?from)))
)
```