Large Language Models as Planning Domain Generators (Student Abstract)

James Oswald[1], **Kavitha Srinivas**[2], **Harsha Kokel**[2], **Junkyu Lee**[2], **Michael Katz**[2], **Shirin Sohrabi**[2]

[1]Rensselaer Polytechnic Institute
[2]IBM Research AI
oswalj@rpi.edu, {kavitha.srinivas,harsha.kokel,junkyu.lee,michael.katz1}@ibm.com, ssohrab@us.ibm.com

Abstract

The creation of planning models, and in particular domain models, is among the last bastions of tasks that require extensive manual labor in AI planning; it is desirable to simplify this process for the sake of making planning more accessible. To this end, we investigate whether large language models (LLMs) can be used to generate planning domain models from textual descriptions. We propose a novel task for this as well as a means of automated evaluation for generated domains by comparing the sets of plans for domain instances. Finally, we perform an empirical analysis of 7 large language models, including coding and chat models across 9 different planning domains. Our results show that LLMs, particularly larger ones, exhibit some level of proficiency in generating correct planning domains from natural language descriptions.

Introduction

Large language models (LLMs) have demonstrated robust emergent abilities for open-ended tasks. If LLMs can bridge the gap between natural language description of the problem and symbolic representation, it would enable large-scale adoption of symbolic methods and reduce the dependency on technical experts. Motivated by this, we investigate LLMs for generating problem representations for automated planning (Ghallab, Nau, and Traverso 2004). We explore whether the commonsense knowledge, natural language capabilities, and emergent structured code generation ability of LLMs help constructing declarative planning domains. Specifically, we leverage LLMs to automatically translate natural language description of a domain to Planning Domain Description Language (PDDL) (Fox and Long 2003).

Approach

We are interested in generating and evaluating domains on an action-by-action basis, where each prompt to the LLM is a request to generate one action in a domain using context examples from other domains. This action-by-action prompting was inspired by Guan et al. (2023) and is primarily a concern due to the size of the LLM's context window. In order to evaluate generated domains automatically, a ground truth domain is needed to compare the generated domains against. For this we use existing PDDL domains as a starting point in our approach. Given a starting domain $\mathbf{D} = \langle \mathcal{F}, \mathcal{A} \rangle$, composed of a set of predicates describing the world $\mathcal{F}$ and set of action schema $\mathcal{A}$, we begin by converting all action schema in $\mathcal{A}$ to natural language descriptions of action schema, $N(\mathcal{A})$. We assume that a list of the predicates in the domain $\mathcal{F}$ and natural language descriptions of these predicates $N(\mathcal{F})$ are given to us as context for the domain. The natural language action $N(a) \in N(\mathcal{A})$, along with a specification of domain predicates $\langle \mathcal{F}, N(\mathcal{F}) \rangle$, is used as the query for the in-context learning prompt. For the prompt's context examples, other actions are randomly sampled from action schema outside of the domain $\mathbf{D}$ of the current action. A model then takes these prompts and transforms them into a sequence of tokens $T(a)$ representing a as a PDDL action. An attempt is made to parse $T(a)$ as a PDDL action a'. This is the first location at which automated evaluation is possible, as there are numerous reasons why $T(a)$ may fail to be a valid PDDL action, many of which can be extracted by just attempting to parse $T(a)$. For all $T(a)$ that were successfully parsed into a reconstructed PDDL action a', we add them to the set of successfully reconstructed actions $\mathcal{A}'$. Next, for each $a' \in \mathcal{A}'$ we create a reconstructed domain $\mathbf{D}'$ from $\mathbf{D}$ by replacing $\mathcal{A}$ with $(\mathcal{A}/a) \cup a'$ where a is the original action that generated a'. Our task then, is to evaluate the quality of each $\mathbf{D}'$ with respect to $\mathbf{D}$.

Evaluation

The primary reason a planning domain is created is so that it can be used as the underlying representation for a set of problems in the domain. The problems implicitly define a set of plans, and when reconstructing domains, we can measure domain equivalence in terms of equivalence of the sets of plans for a collection of problems. While it is not practical to check if the full set of plans is equivalent, it is possible to check for a number of plans on some problems we care about in the domain. The domain equivalence heuristic is computed as follows: given an original planning domain $\mathbf{D}$, a reconstructed planning domain $\mathbf{D}'$, and a set of solvable planning problems for D, $\mathbf{P}_D$, each problem $\Pi \in \mathbf{P}_D$ can be transformed into a problem $\Pi' \in \mathbf{P}_{D'}$ that uses $\mathbf{D}'$ as its underlying domain. For each such pair of problems Π and Π' and some corresponding subsets of their plans $P \subseteq \mathcal{P}_\Pi$ and $P' \subseteq \mathcal{P}_{\Pi'}$, we can cross check whether $P \subseteq \mathcal{P}_{\Pi'}$ and

$P' \subseteq \mathcal{P}_{\Pi}$. For each individual plan, the test can be efficiently performed using a plan validator. This heuristic, plan equivalence on **P** for a subset of plans, is a necessary condition for true domain equivalence, and its negation is a sufficient condition to show true domain inequality.

Result Classes

We propose four result classes with their own subclasses for classifying the action from an LLMs output: **Syntax Error**: The model produced syntactically invalid PDDL. This PDDL cannot be parsed to evaluate an action reconstruction error with. Subclasses are (in precedence order): (1) No PDDL (NoPDDL): Model did not output any PDDL, (2) Parenthesis Mismatch (PError): issues regarding the matching parenthesis in the PDDL (3) Unexpected Token (UToken): The PDDL parser failed after finding an unexpected token. **Semantic Error**: The model produced syntactically valid PDDL, but the PDDL doesn't integrate with the intended problems. Subclasses are (1) Type Error (TError): The model produced an unexpected type (2) Predicate Argument Error (PAError): the wrong number of variables were passed to a predicate (3) Wrong Action Name (NError), The name of the action is wrong (4) Bad Precondition (BPError): PDDL STRIPS does not allow negated preconditions, but one is present. **Different Domain**: The model produced syntactically valid PDDL that integrates with the original domain, but the underlying domains are different by way of the domain equivalence heuristic. The behavior of the actions is not as intended, plans from the original domain cannot be applied in the new domain and vice versa. Subclasses are (1) No Plans Found (NoPlan): No plans were able to be found on problems in the new domain (2) New Plan Application Error (NPApp): Could not apply a new plan to the original domain (3) Original Plan Application Error (OPApp): The original plan could not be applied to the new domain. **(Heuristically) Equivalent Domain**: The model produced syntactically valid PDDL that integrates with the desired domain under the domain equivalence heuristic, plans from the original domain can be applied in the new domain and vice versa.

Experiments and Results

For evaluation, we evaluate over the LLaMA family of LLMs (Touvron, Lavril, and Izacard 2023), as well as the 15 billion parameter StarCoder (SC) model (Li, Allal, and Zi 2023). For LLaMA we evaluate over both the base pretrained models at 7b, 13b, 70b parameters (7b, 13b, 70b). We also evaluate the 7b, 13b, 70b LLaMA models that have been fined tuned for chat using reinforcement learning with human feedback (7bC, 13bC, 70bC). For our domains, we use 9 total, 5 from the international planning competition domains, and 4 from Silver et al. (2023). For heuristic domain equivalence we used KStar planner (Lee, Katz, and Sohrabi 2023) to generate the plans, and VAL (Howey, Long, and Fox 2004) for testing them.

Table 1 displays the result class breakdown. For syntax errors, there were no instances of the No PDDL subclass, all models evaluated output something minimally interpretative as PDDL. For semantic errors, the primary breakdown was dominated by issues related to predicate argument counts where the model added or removed arguments to predicates in the action schema. The results on the different-domain subclasses show that across the board, the majority of valid generated domains in the different-domain result class are not able to be used for planning, with the planner failing to produce any valid plan using the reconstructed problems in domain $P_{D'}$. Overall, we observe that the trend that the larger LLaMA modes perform better, with our best performance being on the largest LLaMA 70b base model.

Result	SC	7b	7bC	13b	13bC	70b	70bC
Syntax	3.70	15.31	22.03	1.30	25.73	**0.36**	8.49
NoPDDL	0.00	0.00	0.00	0.00	0.00	0.00	0.00
PError	0.31	0.21	0.16	**0.00**	0.10	**0.00**	0.05
UToken	3.39	15.10	21.82	1.30	25.62	**0.36**	8.07
Semantics	18.02	22.29	36.15	14.64	22.97	**7.08**	11.72
PAError	15.57	17.55	25.10	8.59	15.26	**4.58**	9.17
NError	0.16	0.10	**0.00**	0.05	0.16	0.10	0.05
TError	**2.29**	4.64	10.57	5.78	7.45	2.40	2.50
BPError	**0.00**	0.21	0.47	0.21	0.10	**0.00**	**0.00**
Diff	67.55	56.46	**36.25**	75.21	43.13	63.75	58.07
NoPlan	51.72	44.64	**23.07**	59.43	26.72	42.50	41.77
NPApp	**7.76**	8.85	8.80	8.96	11.67	12.34	11.20
OPApp	8.07	**2.97**	21.72	6.82	4.74	8.91	5.10
Equiv	10.73	5.94	5.57	8.85	8.18	**28.80**	21.72

Table 1: Distribution of Result Classes and Subclasses. Lower is better for all classes and subclasses except Equiv.

References

Fox, M.; and Long, D. 2003. PDDL2.1: An Extension to PDDL for Expressing Temporal Planning Domains. *J. Artif. Intell. Res.*, 20: 61–124.

Ghallab, M.; Nau, D. S.; and Traverso, P. 2004. *Automated planning - theory and practice*. Elsevier.

Guan, L.; Valmeekam, K.; Sreedharan, S.; and Kambhampati, S. 2023. Leveraging Pre-trained Large Language Models to Construct and Utilize World Models for Model-based Task Planning. arXiv:2305.14909.

Howey, R.; Long, D.; and Fox, M. 2004. VAL: automatic plan validation, continuous effects and mixed initiative planning using PDDL. In *16th IEEE International Conference on Tools with Artificial Intelligence*, 294–301.

Lee, J.; Katz, M.; and Sohrabi, S. 2023. On K* Search for Top-k Planning. In *Proceedings of the 16th Annual Symposium on Combinatorial Search (SoCS 2023)*. AAAI Press.

Li, R.; Allal, L. B.; and Zi, Y. 2023. StarCoder: may the source be with you! arXiv:2305.06161.

Silver, T.; Dan, S.; Srinivas, K.; Tenenbaum, J. B.; Kaelbling, L. P.; and Katz, M. 2023. Generalized Planning in PDDL Domains with Pretrained Large Language Models. arXiv:2305.11014.

Touvron, H.; Lavril, T.; and Izacard, G. 2023. LLaMA: Open and Efficient Foundation Language Models. arXiv:2302.13971.