Proceedings of the Thirty-Second International Joint Conference on Artificial Intelligence (IJCAI-23)
Demonstrations Track

# Plansformer Tool: Demonstrating Generation of Symbolic Plans Using Transformers

**Vishal Pallagani**[1] , **Bharath Muppasani**[1] , **Biplav Srivastava**[1] , **Francesca Rossi**[2] , **Lior Horesh**[2] , **Keerthiram Murugesan**[2] , **Andrea Loreggia**[3] , **Francesco Fabiano**[4] , **Rony Joseph**[5] , **Yathin Kethepalli**[5]
[1]University of South Carolina - USA
[2]IBM Research - USA
[3]University of Brescia - Italy
[4]University of Udine - Italy
[5]IIIT Naya Raipur - India

## Abstract

Plansformer is a novel tool that utilizes a fine-tuned language model based on transformer architecture to generate symbolic plans. Transformers are a type of neural network architecture that have been shown to be highly effective in a range of natural language processing tasks. Unlike traditional planning systems that use heuristic-based search strategies, Plansformer is fine-tuned on specific classical planning domains to generate high-quality plans that are both fluent and feasible. Plansformer takes the domain and problem files as input (*in PDDL*) and outputs a sequence of actions that can be executed to solve the problem. We demonstrate the effectiveness of Plansformer on a variety of benchmark problems and provide both qualitative and quantitative results obtained during our evaluation, including its limitations. Plansformer has the potential to significantly improve the efficiency and effectiveness of planning in various domains, from logistics and scheduling to natural language processing and human-computer interaction. In addition, we provide public access to Plansformer via a website as well as an API endpoint; this enables other researchers to utilize our tool for planning and execution. The demo video is available at https://youtu.be/_1rlctCGsrk.

## 1 Introduction

Large Language Models (LLMs) have revolutionized the field of Natural Language Processing (NLP), outperforming humans in various natural language tasks [Vaswani *et al.*, 2017; Devlin *et al.*, 2018; Brown *et al.*, 2020; Scao *et al.*, 2022; Chowdhery *et al.*, 2022; Li, 2022]. However, their use in domains involving symbols, such as mathematics [Hendrycks *et al.*, 2021b; Cobbe *et al.*, 2021], coding [Hendrycks *et al.*, 2021a; Chen *et al.*, 2021], and automated planning [Lamanna *et al.*, 2023; Jiménez *et al.*, 2012], has been limited due to their inability to reason with symbolic data. In this paper, we propose using LLM trained for code generation to generate valid plans for automated planning domains.

To accomplish this, we create a training and test set for four classical planning domains and use CodeT5 (base) [Wang *et al.*, 2021], a pre-trained code generation model, as the LLM. We then present Plansformer, which is obtained by fine-tuning CodeT5 on planning problems, making it capable of generating symbolic plans of high quality. Our experimental results indicate that the syntactic and symbolic knowledge learned from different programming languages in the CodeT5 model can be useful for the PDDL-based automated planning task, achieving promising results in generating valid and optimal plans.

Plansformer is not intended to replace traditional automated planners, which are capable of generating valid or optimal plans, but rather complement them. A Plansformer can play to its benefit as a fast solver and has relaxation in terms of correctness, while the traditional planner can be used as a sound and complete solver, which is deliberative and always generates a correct output. This work also explores LLMs' capabilities in dealing with symbolic language, revealing a promising direction to harness LLMs for symbolic tasks such as planning. This is a significant contribution as prior work [Valmeekam *et al.*, 2022; Silver *et al.*, 2022] has shown that even state-of-the-art LLMs, such as GPT-3 [Brown *et al.*, 2020], cannot reason with symbolic data.

## 2 Background and Methodology

Automated planning is a field of AI concerned with generating plans to achieve goals [Ghallab *et al.*, 2004a; Ghallab *et al.*, 2004b]. Traditional approaches use search algorithms but have scalability and uncertainty limitations [Ghallab *et al.*, 2014]. Learning-based approaches, leveraging machine learning, can overcome these limitations, learn from data, generalize to new domains, and improve performance [Veloso *et al.*, 1995; Zimmerman and Kambhampati, 2003]. We present a comparison of traditional and learning-based planning approaches in Table 2.

Plansformer, a learning-based planner is generated and tested in two phases: modeling and evaluation. Fine-tuning

| **Models** | **Valid Plans** (%) | **Invalid Plans** | | **Optimal Plans (%)** | **Avg. Time (sec)** |
|---|---|---|---|---|---|
| | | Failed (%) | Incomplete/Wrong (%) | | |
| FastDownward (Ground Truth) | 100% | - | - | 100% | 10.28s |
| GPT-2 | 0% | 0% | 100% | 0% | 0.05s |
| T5-base | 0.25% | 17.3% | 82.7% | 0.25% | 0.47s |
| Codex | 0.15% | 99.85% | 0% | 0.15% | 1s |
| CodeT5-base | 0.6% | 0% | 99.4% | 0.6% | 0.68s |
| Plansformer | **83.64%** | 16.18% | 0.19% | **73.27%** | **0.06s** |
| Plansformer-bw | 90.04% | 9.94% | 0.02% | 88.44% | 0.05s |
| Plansformer-hn | 84.97% | 14.72% | 0.31% | 82.58% | 0.05s |
| Plansformer-gr | 82.97% | 16.61% | 0.42% | 69.47% | 0.06s |
| Plansformer-dl | 76.56% | 23.44% | 0% | 52.61% | 0.09s |

Table 1: Results of plan validation.

| **Criteria** | **Traditional Planners** | **Learning-based Planning** |
|---|---|---|
| Representation | Symbolic representation, logical reasoning | Neural network-based, data-driven |
| Scalability | Limited scalability, exponential growth of state space | Scalable to large state spaces, can learn from large data sets |
| Accuracy | Can guarantee correctness, optimal solutions | Prone to errors, suboptimal solutions |
| Generalization | Limited ability to generalize to unseen domains | Can generalize to unseen domains with sufficient training data |
| Interpretability | Human-understandable, explainable | Lack of interpretability, black-box models |
| Efficiency | Inefficient for large state spaces, computationally expensive | Can be more efficient than traditional planners, especially for large state spaces |

Table 2: Comparison of traditional and learning-based planning

the CodeT5 to address planning syntax and semantics for the first phase, and evaluating the competency of Plansformer as a language model and planner for the second phase. The sequence of actions generated by Plansformer are validates using both language based (*e.g. ROUGE, BLEU*) and planning based metrics (*e.g. validity, optimality*).

### 2.1 Modeling Phase

In the modeling phase, we fine-tune CodeT5 by creating a planning-based dataset. We focus on four classical planning benchmark domains from International Planning Competitions [ICAPS, 2022; Younes *et al.*, 2005; Long and Fox, 2003]: *Blocksworld* [Gupta and Nau, 1991], *Towers of Hanoi* [Gerety and Cull, 1986], *Grippers* [Seipp *et al.*, 2016], and *Driverlog* [Roberts *et al.*, 2014], each with multiple problem instances. We generate optimal plans [Helmert and Domshlak, 2011] for each problem instance using the FastDownward planner [Helmert, 2006]. The generated dataset for each domain contains 18,000 plans with different problem configurations, and we use 5-fold cross-validation for training. We use a Byte-level BPE tokenizer with a vocabulary size of 32,005 and add PDDL-specific tokens ([GOAL], [INIT], [ACTION], [PRE], [EFFECT]) to simplify the input to Plansformer, which represents the goal state, initial state, possible actions with their associated preconditions and effects caused by the actions in the environment. CodeT5 is well-suited for planning tasks as it can generate goal-directed, structured code with semantic meaning. We fine-tune it with 80% of the 18,000 generated samples for each of the four domains in the planning dataset.

### 2.2 Evaluation Phase

In the evaluation phase of Plansformer, the model is tested for both plan validation and language model competency. For plan validation, the sequence of actions generated by Plansformer must guide an agent from the initial state to the goal state for a given problem instance, and we evaluate for optimality and validity using a plan validation tool called VAL [Howey *et al.*, 2004]. For language model competency, we use metrics such as BLEU [Papineni *et al.*, 2002] and ROUGE-L [Lin, 2004] to measure precision and recall, respectively. Although these metrics have no direct intuition in automated planning, we use them to evaluate the task of plan generation from the perspective of LLMs. The results of these evaluations provide conclusive evidence on how well Plansformer generates plans and its performance as a language model.

## 3 System Demonstration

The website [Pallagani, 2023b] provides an easy-to-use interface to exploit all the functionalities of Plansformer. It also provides all the information to guide the user through the different features made available, for instance how to use and leverage the various features and tools for editing and generating plans. On the landing page, the tool is introduced to the user, along with instructions on how to use it. By clicking on the editor button (in the menu bar), the user is directed to the tool usage page. The website provides an intuitive PDDL [Aeronautiques *et al.*, 1998] editor, where the user can add or edit code files as per their requirements. Figure 1 provides an

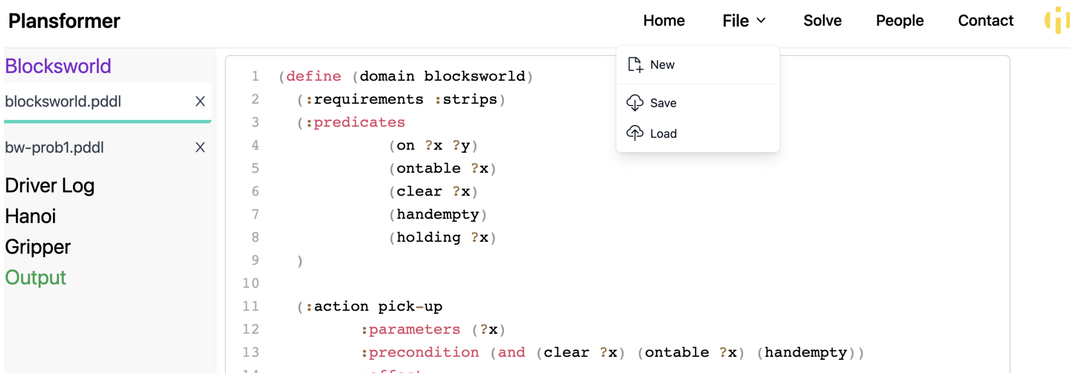


Figure 1: Screenshot of the editor page.

example of the editor page with the Blocksworld domain in PDDL.

For the user's convenience, the website offers a reference to some problem instances from the four domains under study. Furthermore, the user can create new files on the website or upload files from their local system. To facilitate the user's experience, the website also offers the functionality of saving files from the Plansformer tool to their local file storage.

The solve functionality is the next step in the user journey. To initiate the plan generation process, the user needs to choose a domain and its corresponding problem file. The website provides an optional plan validity checker, which enables the user to know the validity and optimality characteristics of the generated plan. Once the solution is obtained, the output is displayed, and the user is shown the generated plan, a summary of the selected domain and problem, along with the validation results if requested. Researchers can also leverage the API to use Plansformer for their work [Pallagani, 2023a], in order to use the API endpoint domain and problem files must be specified as input, as reported in Listing 1

Listing 1: Example API Request

```
curl -X POST \
  -H 'Content-Type: multipart/form-data'\
  -F 'domain=@/path/to/domain.pddl' \
  -F 'problem=@/path/to/problem.pddl' \
  http://129.252.131.13/plansformer/
```

Sometimes users may experience some delays in the computation of a solution. The website is implemented for demonstration purposes and it is based on a basic server. This is the reason for the increased latency of plan generation in the demo website which can leverage only on CPUs for the computation of a solution. The website is inspired by the well-known online planning tool [Muise, 2015], which should help users in getting familiar with our solution.

## 4 System Evaluation

Plansformer is evaluated on multiple planning domains of varying complexities using both quantitative and qualitative measures. For model evaluation, Plansformer is compared

| Models | ROUGE-$L_{recall}$ | ROUGE-$L_{precision}$ | ROUGE-$L_{fmeasure}$ | BLEU |
|---|---|---|---|---|
| GPT-2 | 0.04 | 0.14 | 0.06 | 0.07 |
| T5-base | 0.16 | 0.70 | 0.26 | 0.02 |
| Codex | 0.72 | 0.52 | 0.60 | 0.36 |
| CodeT5-base | 0.41 | 0.28 | 0.33 | 0.02 |
| Plansformer | **0.93** | **0.93** | **0.93** | **0.89** |
| Plansformer-bw | 0.97 | 0.99 | 0.98 | 0.90 |
| Plansformer-hn | 0.99 | 0.96 | 0.97 | 0.95 |
| Plansformer-gr | 0.94 | 0.94 | 0.94 | 0.92 |
| Plansformer-dl | 0.82 | 0.83 | 0.82 | 0.79 |

Table 3: Results of model testing (best performance in bold).

with other language models using the model evaluation metrics (ROUGE and BLEU) and the results as seen in Table 3, show that Plansformer outperforms all other models, including Codex [Chen *et al.*, 2021]. However, for plan validation, FastDownward, a traditional classical planning system is also added to the test-bed. Plans generated by Plansformer are evaluated for validity and optimality, and the results as seen in Table 1, show that Plansformer performs best in simple planning domains (such as blocksworld) but generates fewer optimal plans in complex domains (such as driverlog). The paper reports that the average time taken by Plansformer to solve the test-bed of problems is approximately 200 times faster than the FastDownward planner [1]. The study by [Pallagani *et al.*, 2022] provides a comprehensive account of the experimental methodology and outcomes achieved through the application of Plansformer, offering an in-depth analysis of the results obtained.

## 5 Conclusion

In this demonstration, we have showcased a novel tool that leverages an LLM for generating and validating plans for classical problems in four selected domains. As a next step, we are actively expanding the tool's applicability to various other planning domains, aiming to enhance its generalizability and versatility. In future iterations, we plan to extend the tool's capabilities to address more complex planning scenarios, including epistemic and hierarchical planning. Additionally, we aim to enable the tool to automatically repair invalid plans, and to enhance its visual output to facilitate easier plan interpretation and analysis.

[1] when used with GPU capabilities

## References

[Aeronautiques *et al.*, 1998] Constructions Aeronautiques, Adele Howe, Craig Knoblock, ISI Drew McDermott, Ashwin Ram, Manuela Veloso, Daniel Weld, David Wilkins SRI, Anthony Barrett, Dave Christianson, et al. Pddl— the planning domain definition language. *Technical Report, Tech. Rep.*, 1998.

[Brown *et al.*, 2020] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, et al. Language models are few-shot learners. *Advances in neural information processing systems*, 33:1877–1901, 2020.

[Chen *et al.*, 2021] Mark Chen, Jerry Tworek, Heewoo Jun, Qiming Yuan, Henrique Ponde de Oliveira Pinto, Jared Kaplan, Harri Edwards, Yuri Burda, Nicholas Joseph, Greg Brockman, et al. Evaluating large language models trained on code. *arXiv preprint arXiv:2107.03374*, 2021.

[Chowdhery *et al.*, 2022] Aakanksha Chowdhery, Sharan Narang, Jacob Devlin, Maarten Bosma, Gaurav Mishra, Adam Roberts, Paul Barham, Hyung Won Chung, Charles Sutton, Sebastian Gehrmann, et al. Palm: Scaling language modeling with pathways. *arXiv preprint arXiv:2204.02311*, 2022.

[Cobbe *et al.*, 2021] Karl Cobbe, Vineet Kosaraju, Mohammad Bavarian, Jacob Hilton, Reiichiro Nakano, Christopher Hesse, and John Schulman. Training verifiers to solve math word problems. *arXiv preprint arXiv:2110.14168*, 2021.

[Devlin *et al.*, 2018] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Bert: Pre-training of deep bidirectional transformers for language understanding. *arXiv preprint arXiv:1810.04805*, 2018.

[Gerety and Cull, 1986] C Gerety and P Cull. Time complexity of the towers of hanoi problem. *SIGACT News*, 18(1):80–87, mar 1986.

[Ghallab *et al.*, 2004a] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: theory and practice*. Elsevier, 2004.

[Ghallab *et al.*, 2004b] Malik Ghallab, Dana Nau, and Paolo Traverso. *Automated Planning: Theory and Practice*. The Morgan Kaufmann Series in Artificial Intelligence. Morgan Kaufmann, Amsterdam, 2004.

[Ghallab *et al.*, 2014] Malik Ghallab, Dana Nau, and Paolo Traverso. The actors view of automated planning and acting: A position paper. *Artificial Intelligence*, 208:1–17, 2014.

[Gupta and Nau, 1991] Naresh Gupta and Dana S. Nau. Complexity results for blocks-world planning. In *In Proceedings of AAAI-91*, pages 629–633, 1991.

[Helmert and Domshlak, 2011] Malte Helmert and Carmel Domshlak. Lm-cut: Optimal planning with the landmark-cut heuristic. *Seventh international planning competition (IPC 2011), deterministic part*, pages 103–105, 2011.

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

[Hendrycks *et al.*, 2021a] Dan Hendrycks, Steven Basart, Saurav Kadavath, Mantas Mazeika, Akul Arora, Ethan Guo, Collin Burns, Samir Puranik, Horace He, Dawn Song, et al. Measuring coding challenge competence with apps. *arXiv preprint arXiv:2105.09938*, 2021.

[Hendrycks *et al.*, 2021b] Dan Hendrycks, Collin Burns, Saurav Kadavath, Akul Arora, Steven Basart, Eric Tang, Dawn Song, and Jacob Steinhardt. Measuring mathematical problem solving with the math dataset. *arXiv preprint arXiv:2103.03874*, 2021.

[Howey *et al.*, 2004] Richard Howey, Derek Long, and Maria Fox. Val: Automatic plan validation, continuous effects and mixed initiative planning using pddl. In *16th IEEE International Conference on Tools with Artificial Intelligence*, pages 294–301. IEEE, 2004.

[ICAPS, 2022] ICAPS. International planning competitions at international conference on automated planning and scheduling (icaps). In *https://www.icaps-conference.org/competitions/*, 2022.

[Jiménez *et al.*, 2012] Sergio Jiménez, Tomás De La Rosa, Susana Fernández, Fernando Fernández, and Daniel Borrajo. A review of machine learning for automated planning. *The Knowledge Engineering Review*, 27(4):433–467, 2012.

[Lamanna *et al.*, 2023] Leonardo Lamanna, Luciano Serafini, Mohamadreza Faridghasemnia, Alessandro Saffiotti, Alessandro Saetti, Alfonso Gerevini, and Paolo Traverso. Planning for learning object properties. *arXiv preprint arXiv:2301.06054*, 2023.

[Li, 2022] Hang Li. Language models: Past, present, and future. *Commun. ACM*, 65(7):56–63, jun 2022.

[Lin, 2004] Chin-Yew Lin. Rouge: A package for automatic evaluation of summaries. In *Text summarization branches out*, pages 74–81, 2004.

[Long and Fox, 2003] Derek Long and Maria Fox. The 3rd international planning competition: Results and analysis. *Journal of Artificial Intelligence Research*, 20:1–59, 2003.

[Muise, 2015] Christian Muise. Planning domains. http://planning.domains/, 2015. Accessed: 2023-02-20.

[Pallagani *et al.*, 2022] Vishal Pallagani, Bharath Muppasani, Keerthiram Murugesan, Francesca Rossi, Lior Horesh, Biplav Srivastava, Francesco Fabiano, and Andrea Loreggia. Plansformer: Generating symbolic plans using transformers. *arXiv preprint arXiv:2212.08681*, 2022.

[Pallagani, 2023a] Vishal Pallagani. Plansformer api. http://129.252.131.13/plansformer/, 2023. Accessed: 2023-02-20.

[Pallagani, 2023b] Vishal Pallagani. Plansformer website. https://uo-sc-plansformer.vercel.app/, 2023. Accessed: 2023-02-20.

[Papineni *et al.*, 2002] Kishore Papineni, Salim Roukos, Todd Ward, and Wei-Jing Zhu. Bleu: a method for automatic evaluation of machine translation. In *Proceedings of the 40th annual meeting of the Association for Computational Linguistics*, pages 311–318, 2002.

[Roberts *et al.*, 2014] Mark Roberts, Adele Howe, and Indrajit Ray. Evaluating diversity in classical planning. In *Proceedings of the International Conference on Automated Planning and Scheduling*, volume 24, pages 253–261, 2014.

[Scao *et al.*, 2022] Teven Le Scao, Angela Fan, Christopher Akiki, Ellie Pavlick, Suzana Ilić, Daniel Hesslow, Roman Castagné, Alexandra Sasha Luccioni, François Yvon, Matthias Gallé, et al. Bloom: A 176b-parameter open-access multilingual language model. *arXiv preprint arXiv:2211.05100*, 2022.

[Seipp *et al.*, 2016] Jendrik Seipp, Florian Pommerening, Gabriele Röger, and Malte Helmert. Correlation complexity of classical planning domains. 2016.

[Silver *et al.*, 2022] Tom Silver, Varun Hariprasad, Reece S Shuttleworth, Nishanth Kumar, Tomás Lozano-Pérez, and Leslie Pack Kaelbling. Pddl planning with pretrained large language models. In *NeurIPS 2022 Foundation Models for Decision Making Workshop*, Oct 2022.

[Valmeekam *et al.*, 2022] Karthik Valmeekam, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Large language models still can't plan (a benchmark for llms on planning and reasoning about change). *arXiv preprint arXiv:2206.10498*, 2022.

[Vaswani *et al.*, 2017] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017.

[Veloso *et al.*, 1995] Manuela Veloso, Jaime Carbonell, Alicia Perez, Daniel Borrajo, Eugene Fink, and Jim Blythe. Integrating planning and learning: The prodigy architecture. *Journal of Experimental & Theoretical Artificial Intelligence*, 7(1):81–120, 1995.

[Wang *et al.*, 2021] Yue Wang, Weishi Wang, Shafiq Joty, and Steven CH Hoi. Codet5: Identifier-aware unified pre-trained encoder-decoder models for code understanding and generation. In *Proceedings of the 2021 Conference on Empirical Methods in Natural Language Processing*, pages 8696–8708, 2021.

[Younes *et al.*, 2005] Håkan LS Younes, Michael L Littman, David Weissman, and John Asmuth. The first probabilistic track of the international planning competition. *Journal of Artificial Intelligence Research*, 24:851–887, 2005.

[Zimmerman and Kambhampati, 2003] Terry Zimmerman and Subbarao Kambhampati. Learning-assisted automated planning: Looking back, taking stock, going forward. *AI Magazine*, 24(2):73–73, 2003.