Plansformer Tool: Demonstrating Generation of Symbolic Plans Using Transformers

Vishal Pallagani\textsuperscript{1}, Bharath Muppasani\textsuperscript{1}, Biplav Srivastava\textsuperscript{1}, Francesca Rossi\textsuperscript{2}, Lior Horesh\textsuperscript{2}, Keerthiram Murugesan\textsuperscript{2}, Andrea Loreggia\textsuperscript{3}, Francesco Fabiano\textsuperscript{4}, Rony Joseph\textsuperscript{5}, Yathin Ketepalli\textsuperscript{5}

\textsuperscript{1}University of South Carolina - USA \hfill \textsuperscript{2}IBM Research - USA \hfill \textsuperscript{3}University of Brescia - Italy \hfill \textsuperscript{4}University of Udine - Italy \hfill \textsuperscript{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
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 Cul1, 1986], Grippers [Seipp et al., 2016], and Driverlog [Roberts et al., 2014], each with multiple problem instances. We generate optimal plans [Helmert and Domsh- lak, 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 for optimality and validity, they provide a measure of how well Plansformer generates plans and its performance as a language model.

Table 1: Results of plan validation.

<table>
<thead>
<tr>
<th>Models</th>
<th>Valid Plans (%)</th>
<th>Invalid Plans Failed (%)</th>
<th>Optimal Plans (%)</th>
<th>Avg. Time (sec)</th>
</tr>
</thead>
<tbody>
<tr>
<td>FastDownward Truth</td>
<td>100%</td>
<td>-</td>
<td>100%</td>
<td>10.28s</td>
</tr>
<tr>
<td>GPT-2</td>
<td>0%</td>
<td>0%</td>
<td>100%</td>
<td>0.05s</td>
</tr>
<tr>
<td>T5-base</td>
<td>0.25%</td>
<td>17.3%</td>
<td>82.7%</td>
<td>0.47s</td>
</tr>
<tr>
<td>Codex</td>
<td>0.15%</td>
<td>99.85%</td>
<td>0%</td>
<td>1s</td>
</tr>
<tr>
<td>CodeT5-base</td>
<td>0.6%</td>
<td>0%</td>
<td>99.4%</td>
<td>0.6%</td>
</tr>
<tr>
<td>Plansformer</td>
<td>83.64%</td>
<td>16.18%</td>
<td>0.19%</td>
<td>0.06s</td>
</tr>
<tr>
<td>Plansformer-bw</td>
<td>90.04%</td>
<td>9.94%</td>
<td>0.02%</td>
<td>0.05s</td>
</tr>
<tr>
<td>Plansformer-hn</td>
<td>84.97%</td>
<td>14.72%</td>
<td>0.31%</td>
<td>0.05s</td>
</tr>
<tr>
<td>Plansformer-gr</td>
<td>82.97%</td>
<td>16.61%</td>
<td>0.42%</td>
<td>0.06s</td>
</tr>
<tr>
<td>Plansformer-dl</td>
<td>76.56%</td>
<td>23.44%</td>
<td>0%</td>
<td>0.09s</td>
</tr>
</tbody>
</table>

Table 2: Comparison of traditional and learning-based planning

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

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 validated using both language based (e.g. ROUGE, BLEU) and planning based metrics (e.g. validity, optimality).
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.

```bash
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 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. 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


