# Transition Function Prediction in AI Planning Using LLM

**Eliezer Shlomi**[1], **Guy Azran**[1], **Eilam Shapira**[1], **Omer Nahum**[1], **Guy Uziel**[2], **Michael Katz**[2], **Ateret Anaby Tavor**[2], **Roi Reichart**[1], **Sarah Keren**[1]

[1]Technion - Israel Institute of Technology
[2]IBM Research

## Abstract

Large Language Models (LLMs) excel in natural language processing tasks but face significant challenges in classical planning, where accurate and feasible transitions between states are required. This work presents a novel approach that embeds states and actions into a structured feature space, using a shallow neural network as a transition function. By performing planning in the latent space, the method significantly reduces the computational cost associated with frequent LLM calls while retaining logical consistency. We evaluate this framework as a classifier and demonstrate promising results in state transition prediction and planning tasks across natural language-described domains. The approach offers insights into efficient and scalable LLM-based planning, bridging the gap between natural language understanding and practical planning systems.

## Introduction

The integration of Large Language Models (LLMs) into various computing tasks has led to transformative advances in natural language processing, allowing models to excel in tasks such as text generation, translation, and complex reasoning (Devlin et al. 2019; Radford et al. 2019). These capabilities suggest potential for automated planning, yet leveraging LLMs in classical planning poses distinct challenges due to the need for executable action sequences that accurately reflect environmental dynamics (Kambhampati et al. 2024).

Recent efforts have explored various strategies for integrating LLMs into planning. Inspired by System 1 and System 2 thinking (Kahneman 2011), some approaches directly employ LLMs for plan generation, relying on their language understanding capabilities to produce plans in a single step (Brown et al. 2020). This is often referred to as System 1 thinking. In contrast, System 2 inspired approaches, such as chain-of-thought prompting, generate plans incrementally, reasoning step by step through intermediate states to enhance logical consistency and feasibility (Wei et al. 2022). More recent advances such as the Tree of Thoughts framework expand on this concept by exploring multiple reasoning paths simultaneously, further improving the logical coherence of plans (Yao et al. 2023). These methods, while promising, often result in frequent LLM calls during inference, increasing computational costs (Katz et al. 2025).

In the realm of automated planning, *LatPlan* has emerged as a significant precursor to our current research. Introduced by Asai and Fukunaga (2018), *LatPlan* pioneers the idea of modeling state transitions in latent spaces for image-based domains, leveraging autoencoders to learn symbolic representations from visual inputs, which are then utilized in conjunction with classical planning frameworks (Asai and Fukunaga 2018). Inspired by their innovative approach, our work aims to extend these concepts into the domain of natural language processing. By adapting the foundational principles set forth by *LatPlan*, we develop *EmbedPlan*, which utilizes Large Language Models (LLMs) to embed textual descriptions of states and actions into a structured latent space. This adaptation not only preserves the contextual relationships inherent in textual data but also enhances the scalability and applicability of automated planning across more diverse and complex scenarios.

Other methods transform natural language inputs into structured planning languages such as the Planning Domain Definition Language (PDDL) (McDermott et al. 1998), subsequently using conventional solvers (Liu et al. 2023; Guan et al. 2023). An innovative approach, the Reasoning via Planning framework, attempts to build a world model by repurposing LLMs as both world models and reasoning agents (Hao et al. 2023). However, this approach requires frequent LLM calls during inference, resulting in high computational costs (Katz et al. 2025).

This paper introduces *EmbedPlan*, a method designed to address these limitations by reducing the dependency on LLM calls. EmbedPlan embeds states and actions into a structured feature space and uses a lightweight neural network as a transition function, leveraging the semantic richness of LLM embeddings while significantly reducing computational overhead. By minimizing LLM queries, EmbedPlan has the potential to improve the efficiency of planning in domains described in natural language without compromising logical consistency. Results demonstrate that EmbedPlan has the potential to bridge the gap between natural language understanding and classical planning by leveraging state-action embeddings.

*For example, consider a ferry transporting vehicles between locations.* Initially, the ferry is docked at the northern port, and there is a car waiting to be transported. The first action is to "load the car onto the ferry," after which the intermediate state becomes "the car is on the ferry at the northern port." Next, the ferry performs the action "sail to the southern port," resulting in the intermediate state "the ferry and car are at the southern port." Finally, the action "unload the car" leads to the goal state: "the car is at the southern port, and the ferry is empty."

This example illustrates the complexity of planning in natural language-described domains, where state-action sequences must capture both semantic and logical relationships.

## Background and Related Work

Classical planning involves finding a sequence of actions that transforms an initial state into a goal state while respecting the constraints of a transition function and associated costs. Formally, a planning problem is defined as a tuple $X = (S, s_0, S_g, A, f, c)$, where $S$ represents the set of possible states, $s_0$ is the initial state where the execution of actions begins, $S_g$ denotes the set of goal states specifying the desired outcomes, and $A$ is the set of actions available in the domain. The transition function $f : S \times A \rightarrow S$ defines the effects of actions on states, while the cost function $c : S \times A \rightarrow R$ assigns a cost to each action taken in a state. The objective is to find a sequence of actions $\langle a_1, a_2, \ldots, a_n \rangle$, $a_i \in A$, that transforms $s_0$ into a state within $S_g$, adhering to the constraints of $f$ and minimizing the total cost defined by $c$ (Russell and Norvig 2003).

Planning in natural language domains introduces additional complexity, as states and transitions are expressed in textual representations. These representations require both semantic understanding and reasoning capabilities. Large Language Models (LLMs) have demonstrated strong performance in handling such tasks due to their ability to generate contextualized embeddings that capture semantic and syntactic relationships (Lee et al. 2024). Pre-trained LLMs, such as BERT (Devlin et al. 2019) and GPT (Brown et al. 2020), map natural language descriptions into dense vector spaces, allowing downstream tasks to operate on structured representations. These embeddings provide a natural fit for representing states and actions in planning problems, as they preserve contextual relationships inherent in textual descriptions.

Existing LLM-based approaches to planning often face trade-offs between computational efficiency, logical consistency, and scalability. For instance, the Tree of Thoughts extends reasoning by exploring multiple reasoning paths, which enhances logical consistency but requires frequent LLM invocations (Katz et al. 2025). This makes the approach computationally expensive and less practical for large domains (Yao et al. 2023). LatPlan (Asai and Fukunaga 2018) models state transitions in latent spaces for image-based domains but does not address the challenges of natural language representations. It relies on learning representations of states and actions from visual inputs and uses a trained transition model for planning with classical Planning Domain Definition Language (PDDL) models (McDermott et al. 1998). While LatPlan focuses on learning symbolic representations from raw images for planning, EmbedPlan uses natural language descriptions of states and actions as inputs, leveraging LLMs to embed them into a continuous latent space. This enables planning by directly predicting the next state embedding, bypassing the need for explicit symbolic representations and bridging natural language with modern embedding-based planning approaches.

## Problem Formulation

This work addresses the challenge of efficiently predicting state transitions in natural language-described planning problems. Given an initial state $s_0$ and a goal state $s_g$, the objective is to compute a sequence of actions $\langle a_1, a_2, \ldots, a_n \rangle$, $a_i \in A$, that transitions $s_0$ to $s_g$. The transitions are defined by a function $T : S \times A \rightarrow S$, where $S$ represents the set of states, and $A$ represents the set of actions, both expressed in natural language. The key challenge lies in accurately predicting $T(s_i, a_i)$, ensuring logical consistency while minimizing computational overhead.

## Method

This section details the proposed method for integrating Large Language Models (LLMs) in classical planning tasks. Our approach efficiently combines LLM embeddings with a lightweight transition function, minimizing LLM queries during inference. Refer to Figure 1 for an overview of the method.

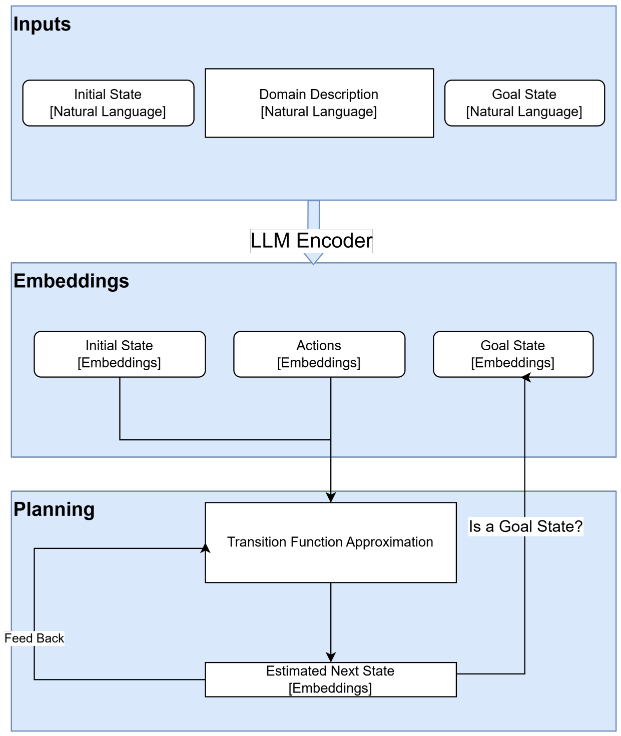
Inputs
Initial State [Natural Language]
Domain Description [Natural Language]
Goal State [Natural Language]
LLM Encoder
Embeddings
Initial State [Embeddings]
Actions [Embeddings]
Goal State [Embeddings]
Planning
Transition Function Approximation
Is a Goal State?
Feed Back
Estimated Next State [Embeddings]

Figure 1: Overview of the proposed method.

Our approach utilizes LLM encodings to approximate the transition function, enhancing planning efficiency by minimizing direct model queries and significantly reducing resource consumption typically associated with LLM-based planning. By leveraging the interpretative capabilities of LLMs in natural language domains, we train a domain-specific model to map state-action pairs to subsequent states, addressing execution limitations and overhead of repeated LLM calls during inference. Results demonstrate the feasibility of this method in planning across diverse domains, suggesting areas for future work to improve the method's robustness and applicability to more complex planning tasks.

Our methodology consists of three primary components. First, embeddings for states and actions are generated using pre-trained LLMs, specifically leveraging their capacity to encode natural language into dense vector representations. The embeddings for a state $s$ and an action $a$ are obtained as follows:

$$\mathbf{s}_{\text{embed}} = \text{Enc}(s), \quad \mathbf{a}_{\text{embed}} = \text{Enc}(a).$$

Second, a shallow neural network approximates the transition function $T : S \times A \to S$, predicting the next state's embedding from the current state and action. This transition function, denoted as $T_\theta$, is trained on state-action pairs $(s_i, a_i, s_{i+1})$ to minimize the cosine similarity loss:

$$\mathcal{L} = 1 - \frac{1}{N}\sum_{i=1}^{N} \frac{T_\theta(\mathbf{s}_{i,\text{embed}}, \mathbf{a}_{i,\text{embed}}) \cdot \mathbf{s}_{i+1,\text{embed}}}{\|T_\theta(\mathbf{s}_{i,\text{embed}}, \mathbf{a}_{i,\text{embed}})\| \|\mathbf{s}_{i+1,\text{embed}}\|}.$$

The cosine similarity loss is particularly relevant for this task because it measures the angular similarity between the predicted and actual embeddings, which is well-suited for text-based representations. Unlike Euclidean distance, cosine similarity focuses on the direction of the embeddings rather than their magnitude, ensuring that semantically similar states and actions are correctly aligned in the latent space. This property makes it an effective choice for optimizing the transition function in natural language-described planning domains where $\mathbf{s}_{\text{next}}$ is the embedding of the state $s_{i+1}$ and serves as the ground truth for the subsequent state.

Finally, the planning process is formalized in Algorithm 1, which outlines the steps to utilize the transition function to efficiently achieve the planning objectives. The algorithm begins by encoding the initial state $s_0$ in its corresponding embedding $\mathbf{s}_{0,\text{embed}}$, which serves as the starting point for the planning process. After that, it iteratively explores potential actions to determine the sequence that leads to the goal state $s_g$. For each action $a$ in the set of possible actions $A$, the algorithm computes the embedding of the action $\mathbf{a}_{\text{embed}}$ using the encoder Enc, uses the transition function $T_\theta$ to predict the embedding of the next state $\mathbf{s}_{\text{next}}$, and calculates a heuristic $h(a)$ for each action, which estimates the cost or distance to the goal state from the predicted next state. The action $a^*$ that minimizes the heuristic $h(a)$ is selected, and the current state embedding is updated to the newly predicted state embedding $\mathbf{s}_{\text{next}}$. This process repeats until the goal state is reached or a set limit of steps is exceeded. The resulting sequence of actions constitutes the plan that transitions the initial state to the goal state.

Algorithm 1: Planning with Transition Function

```
Require: Initial state $s_0$, Goal state $s_g$, Set of actions $A$
 1: Encode the initial state $s_0$ into its embedding $\mathbf{s}_{0,\text{embed}}$
 2: Set $\mathbf{s}_{\text{current}} = \mathbf{s}_{0,\text{embed}}$
 3: for each action $a$ in $A$ do
 4:     Compute action embedding $\mathbf{a}_{\text{embed}}$ using Enc$(a)$
 5: end for
 6: while goal state not reached and within step limits do
 7:     for each action $a$ in $A$ do
 8:         Compute the next state embedding $\mathbf{s}_{\text{next}} = T_\theta(\mathbf{s}_{\text{current}}, \mathbf{a}_{\text{embed}})$
 9:         Calculate the heuristic $h(a)$ for reaching the goal state:
                $h(a) = \|\mathbf{s}_{\text{next}} - \mathbf{s}_{g,\text{embed}}\|^2$
10:     end for
11:     Select the action $a^*$ that minimizes $h(a)$
12:     Update $\mathbf{s}_{\text{current}}$ to $\mathbf{s}_{\text{next}}$ for the action $a^*$
13: end while
Ensure: Sequence of actions leading to the goal state
```

## Experiments

This section describes the experimental setup, the evaluation metric, and the results used to assess the effectiveness of the proposed planning method, referred to as *EmbedPlan*.

**Evaluation Metric Transition Prediction Accuracy:** This metric quantifies the cosine similarity between the predicted and actual embeddings of the next state during validation. A higher score indicates better accuracy in state transition prediction. To provide a detailed view of performance, we use top@k accuracy, which measures the proportion of cases where the correct next state is among the top $k$ predictions based on cosine similarity. For example, for $k = 5$, the model correctly predicts the next state if the ground truth embedding is within the top 5 closest predictions. This evaluation method reflects how well the model captures relevant transitions as more candidate predictions are considered. Graphs depict accuracy for $k$ values of 1, 5, 10, and 100 across different domains.

**Experimental Setup** Experiments were conducted to evaluate the performance of the transition function in planning tasks across various domains. Each domain involved problems written in PDDL and translated into natural language descriptions, allowing the use of conventional planning datasets with LLM-based state and action embeddings(Kokel et al. 2024). The transition function was implemented using a shallow neural network optimized for quick inference. Models were trained on increasing subsets of data (10%, 20%, 30%, etc.) and tested on the remaining data out of the total 10,000 samples per domain to evaluate scalability and learning efficiency.

**Results and Insights** The results indicate competitive transition prediction accuracy across seven domains: Ferry, FloorTile, Grid, Gripper, Logistics, Rovers, and VisitAll. Each domain features unique tasks, such as vehicle transportation in Ferry, tile laying in FloorTile, and navigation tasks in Grid and Rovers. Figures 2 to 8 depict the accuracy across these scenarios. *EmbedPlan* shows substantial top@k accuracy improvements in all tested domains, demonstrating its capability to effectively approximate transition functions in complex planning environments. This effectiveness confirms the practicality of embedding-based methods for planning tasks, offering a scalable and computationally efficient solution for integrating Large Language Models into classical planning frameworks.

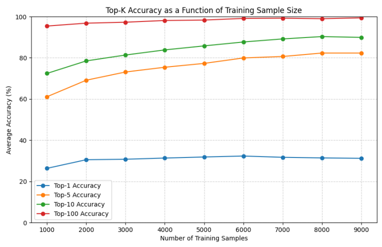

Figure 2: Transition accuracy in the Ferry domain.

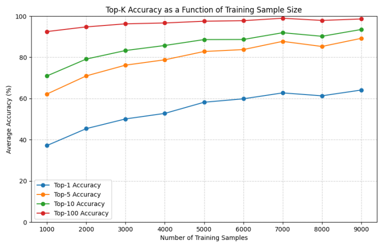

Figure 3: Transition accuracy in the FloorTile domain.

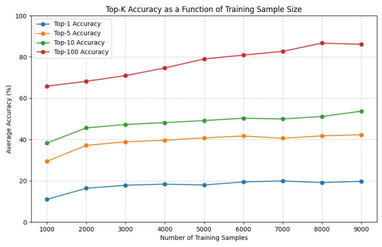

Figure 4: Transition accuracy in the Grid domain.

## Conclusion

This study introduced *EmbedPlan*, a method that leverages transition functions to enhance the integration of Large Language Models (LLMs) into classical planning tasks. The approach has demonstrated substantial improvements in transition prediction accuracy across multiple domains, confirming its potential for real-time planning applications. *EmbedPlan* reduces computational demands and effectively classifies the correct next states, showcasing adaptability to com-

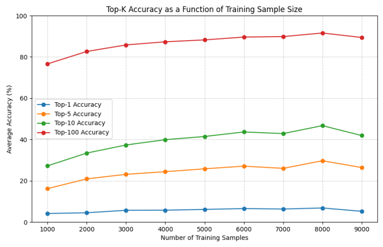

Figure 5: Transition accuracy in the Gripper domain.

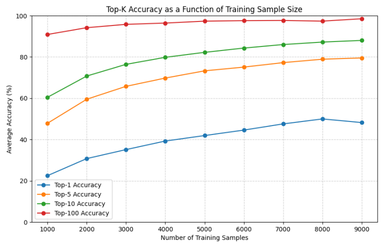

Figure 6: Transition accuracy in the Logistics domain.

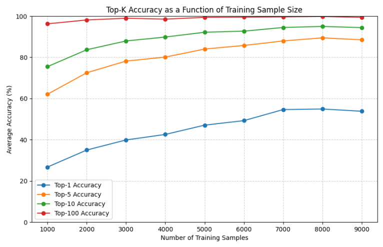

Figure 7: Transition accuracy in the Rovers domain.

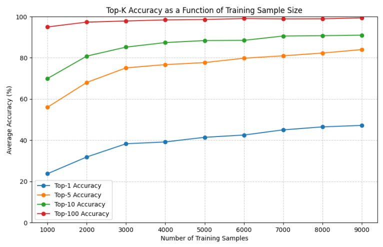

Figure 8: Transition accuracy in the VisitAll domain.

plex scenarios. These promising results open the gate for further research, which will focus on enhancing robustness and exploring the method's applicability to more complex tasks, such as multi-agent systems or dynamic environments. Future work will also investigate methods to improve generalization across diverse domains and optimize transition function accuracy for long-horizon planning tasks, further bridging the gap between natural language understanding and practical planning systems.

## References

Asai, M.; and Fukunaga, A. 2018. Classical Planning in Deep Latent Space: Bridging the Subsymbolic-Symbolic Boundary. *Proceedings of the AAAI Conference on Artificial Intelligence*, 32(1).

Brown, T.; Mann, B.; Ryder, N.; Subbiah, M.; Kaplan, J. D.; Dhariwal, P.; Neelakantan, A.; Shyam, P.; Sastry, G.; Askell, A.; Agarwal, S.; Herbert-Voss, A.; Krueger, G.; Henighan, T.; Child, R.; Ramesh, A.; Ziegler, D.; Wu, J.; Winter, C.; Hesse, C.; Chen, M.; Sigler, E.; Litwin, M.; Gray, S.; Chess, B.; Clark, J.; Berner, C.; McCandlish, S.; Radford, A.; Sutskever, I.; and Amodei, D. 2020. Language Models are Few-Shot Learners. In Larochelle, H.; Ranzato, M.; Hadsell, R.; Balcan, M.; and Lin, H., eds., *Advances in Neural Information Processing Systems*, volume 33, 1877–1901. Curran Associates, Inc.

Devlin, J.; Chang, M.-W.; Lee, K.; and Toutanova, K. 2019. Bert: Pre-training of deep bidirectional transformers for language understanding. In *Proceedings of the 2019 conference of the North American chapter of the association for computational linguistics: human language technologies, volume 1 (long and short papers)*, 4171–4186.

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. *Advances in Neural Information Processing Systems*, 36: 79081–79094.

Hao, S.; Gu, Y.; Ma, H.; Hong, J. J.; Wang, Z.; Wang, D. Z.; and Hu, Z. 2023. Reasoning with language model is planning with world model. *arXiv preprint arXiv:2305.14992*.

Kahneman, D. 2011. *Thinking, Fast and Slow*. Macmillan.

Kambhampati, S.; Valmeekam, K.; Guan, L.; Verma, M.; Stechly, K.; Bhambri, S.; Saldyt, L. P.; and Murthy, A. B. 2024. Position: LLMs can't plan, but can help planning in LLM-modulo frameworks. In *Forty-first International Conference on Machine Learning*.

Katz, M.; Kokel, H.; Srinivas, K.; and Sohrabi Araghi, S. 2025. Thought of Search: Planning with Language Models Through The Lens of Efficiency. *Advances in Neural Information Processing Systems*, 37: 138491–138568.

Kokel, H.; Katz, M.; Srinivas, K.; and Sohrabi, S. 2024. ACPBench: Reasoning about Action, Change, and Planning. *arXiv preprint arXiv:2410.05669*.

Lee, C.; Roy, R.; Xu, M.; Raiman, J.; Shoeybi, M.; Catanzaro, B.; and Ping, W. 2024. Nv-embed: Improved techniques for training llms as generalist embedding models. *arXiv preprint arXiv:2405.17428*.

Liu, B.; Jiang, Y.; Zhang, X.; Liu, Q.; Zhang, S.; Biswas, J.; and Stone, P. 2023. Llm+ p: Empowering large language models with optimal planning proficiency. *arXiv preprint arXiv:2304.11477*.

McDermott, D.; Ghallab, M.; Howe, A. E.; Knoblock, C. A.; Ram, A.; Veloso, M. M.; Weld, D. S.; and Wilkins, D. E. 1998. PDDL-the planning domain definition language.

Radford, A.; Wu, J.; Child, R.; Luan, D.; Amodei, D.; and Sutskever, I. 2019. Language Models are Unsupervised Multitask Learners. *OpenAI Blog*, 1(8).

Russell, S. J.; and Norvig, P. 2003. *Artificial Intelligence: A Modern Approach*. Prentice Hall.

Wei, J.; Wang, X.; Schuurmans, D.; Bosma, M.; Xia, F.; Chi, E.; Le, Q. V.; Zhou, D.; et al. 2022. Chain-of-thought prompting elicits reasoning in large language models. *Advances in neural information processing systems*, 35: 24824–24837.

Yao, S.; Yu, D.; Zhao, J.; Shafran, I.; Griffiths, T.; Cao, Y.; and Narasimhan, K. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Advances in neural information processing systems*, 36: 11809–11822.