Benchmarking Large Language Models for Path Planning on Discrete Grids

Wayne Chen[1], **Tyler King**[1]
[1]Cornell University, Department of Computer Science
{zc272,ttk22}@cornell.edu

Abstract

There has been tremendous interest in large language models (LLMs) recently due to their ability to tackle novel tasks. For example, many GPT variants since GPT-3 have been able to generate code, write essays, and even create meal recipes. Another recent trend has been using LLMs for planning. In this paper, we explore how well LLMs can solve simple path planning games and to what extent they can generalize to games not seen during training. We create a synthetic benchmark of several path planning games and fine-tune T5 to generate paths from a natural language description of the game. In addition, we benchmark the potential of GPT-4 with Chain of Thought reasoning on all synthetic datasets. We find that the fine-tuned model is capable of solving such games, but performance on unseen games is poor.

1 Introduction

The NLP landscape has transformed drastically with the advent of large language models (LLMs) [6, 11, 16, 19], which have shown surprising capabilities in linguistics and reasoning [7, 14]. Although trained only on next token predictions, these models have exemplified seemingly emergent abilities such as spatial-temporal reasoning [1, 4]. At the same time, challenges in generalizing traditional RL methods have sparked interest in using language to better ground models [15, 24, 9]. This has led to efforts such as [2] which embed a planning task's information into an LLM and uses the enhanced model to solve the task.

Recently, there has been work proposing benchmarks to standardize evaluating LLMs on planning problems. [1] created the Path Planning from Natural Language (PPNL) benchmark, which tasks the model with navigating a grid from a start to target location while avoiding obstacles. This was tested both via fine-tuned T5 [20] model and through few-shot prompting of GPT-4 [16]. Meanwhile, PlanBench [22] is a benchmark featuring block stacking and package shipping games. Here, all evaluation was done via few-shot prompting GPT-4.

While in both papers GPT-4 and the fine-tuned T5 models were performant on the task, we are interested in performance when the planning task changes. For example, given a fine-tuned T5 model trained to solve the obstacle environment in PPNL, could it solve a variation of the environment where it were only permitted to move in hops? Could a model trained on a variety of planning problems act as a foundational model that can solve new planning problems? Note that these inquiries are mostly relevant to "smaller" LLMs that could be fine-tuned, since the largest ones can perform well with just in-context learning [6].

This is particularly relevant since more prior work [1, 8, 22] have focused on a simple obstacle environment (i.e. moving from a start to end position while avoiding obstacles), but there exists a much richer class of discrete path planning problems. Furthermore, there exists literature exploring reinforcement learning approaches for both generalized graphs [5], grid-like pathfinding problems

CS 6756: Learning for Robot Decision Making (Fall 2023).

[10, 17], and multi-agent games [26] but to the best of our knowledge, we have not seen anyone tackle general classes of path-planning problems with LLMs.

We create a synthetic dataset of games similar to the one in PPNL and evaluate a T5-model on learning all the games as well as learning multiple games and testing on an unseen game. We also compare these results against in-context learning with GPT-4.

2 Proposed Datasets

Task Formulation: Formally, we construct our problems as 5×5 grids with $k < 23$ obstacles defined by the set $\mathbb{O} = \{O_1, O_2, \ldots, O_k\}$, special cells $\mathbb{S} = \{S_1, S_2, \ldots, S_i\}$ and constraint(s) C. The locations for all these points are randomly sampled. For such a problem, we define each path-planning task with an initial position P_0 and a goal location P_1. The task for the LLM agent is then to perform a list of actions $\mathbb{A} = (A_1, A_2, \ldots, A_t)$ that navigate from P_0 to the goal location(s) while avoiding obstacles in $\mathbb{O}$ and adhering to constraints C.

For each potential list of actions $\mathbb{A}$, we describe a path as "optimal" if the length of the list is no longer than by applying Dijkstra's algorithm to the same setting. To compute these minimum paths, we convert the underlying grids into a weighted graph where edge (u, v) cost is determined based on C, and then computing the Dijkstra's algorithm on the graph from node P_0 to P_1.

These additional constraints C and special cells $\mathbb{S} = \{S_1, S_2, \ldots, S_i\}$ are contingent on the underlying problem being solved. In our work, we introduce **8** new environments that build upon PPNL. Each environment contains obstacle cells. They are as follows:

1. **Teleport**: There exist special cells S_1 and S_2, and reaching either on action A_t allows movement to the other on action A_{t+1} (or also to just move to a neighbor cell). For all environments with special cells, the agent is permitted to move away from the cell in a normal manner.
2. **Beamer**: There exists on special cell S_1, which after reaching it on action A_t allows movement to any other non-obstacle cell on action A_{t+1}. This means after reaching S_1, one can move to goal state P_1 in the step afterwards.
3. **Activated Beamer**: There exist multiple teleportation cells $S_1, S_2, \ldots, S_i$, where landing on any teleportation cell S_j on action A_t and staying on S_j for action A_{t+1} allows for teleportation to any other S_k, for $j \neq k$ and $1 \leq k \leq i$ on step A_{t+2}. This can be considered a k-generalization of teleport, where there exist i teleport cells instead of 2 teleport cells.
4. **Beamer Group**: Similar to activated beamer, except without requiring to stay on the first teleportation cell for an extra step (i.e. can immediately teleport from one teleport cell to another teleport cell as an action A_{t+1}).
5. **Diagonal**: Movements can only be made along the diagonal. That is at coordinate position (i, j), the valid moves are to positions $(i-1, j-1)$, $(i-1, j+1)$, $(i+1, j-1)$, and $(i+1, j+1)$.
6. **Hop**: There exist no additional special cells $\mathbb{S}$, and every action hops 2 cells over (as opposed to moving to a neighboring cell). Note that this allows hopping "over" obstacle cells. In the case where an action moves outside the grid, the action is mended to place the agent on a border cell.
7. **Activated Hop**: Similar to activated beamer, except the special ability is a hop of 3 cells in any direction. Note that permitting only hops of 2 steps will not make sense in most scenarios since the agent can just move the 2 steps (assuming no obstacle cell is blocking). Similar to hop, the agent can move less than 3 steps in the case where it would place it outside the grid boundary.
8. **Prereq**: There exist special cells $S_1, S_2, \ldots, S_i$, where each represents a prerequisite (i.e. S_i must be reached before P_1). This is similar to the multi-goal environment in [1], except 1 target is treated differently.

For each game variant, we construct 1000 valid samples (i.e. the instance is solvable), and then randomly split it into 80/10/10 for train, validation, and test respectively.

To ensure there were enough training examples that made use of special cells and the validation and test splits contained some solution paths using special cells, we report the percentage of paths with special cells (for the environments that do have special cells)2.

Table 1: Percentage of each split's paths that uses special cells

Dataset/Split	TRAIN	VALIDATION	TEST
TELEPORT	13.75	21.00	12.00
BEAMER	31.63	31.00	26.00
ACTIVATED BEAMER	11.13	8.00	14.00
BEAMER GROUP	12.00	15.00	15.00
ACTIVATED HOP	16.60	16.00	19.00

3 Related Work

In terms of benchmarks and experiments conducted, PPNL [1] is most similar to our work. The authors created 5×5, 6×6, and 7×7 grid environments with between 1 and 5 obstacles and the model was tasked with generating a route from a start to end location. A multi-goal variant was also created which requires the model to visit multiple target cells. T5 and BART [12] models were trained on 668 instances of the single goal 6×6 grids and evaluated on the 5×5 and 7×7 grids (including multi goal). Evaluating GPT-4 was done with the following few-shot prompting schemes:

1. Naive: the output sequence is something like "right, up, up, left, up, ..."
2. Action-and-Effect: the model tracks the effects of its actions by outputting something like "Go right. You are at (0, 2). Go right. You are now at (0, 3) ..."
3. Chain of Thought: the model is encouraged to reason about the task and outputs something like "(3,4) is 3 steps down and 3 steps to the right of (0,1). To avoid the obstacle at (2,1), which is 2 steps down from (0,1), I should start by moving right..."
4. ReAct (Reason and Act): This is similar to chain of thought, except following every an action (or series of actions) of the model's choice, an oracle informs the model of its state

Additionally, [1] contained metrics for distance to goal, which measures for valid paths (those that obey the constraints of the game) the number of steps required to reach the target after traversing the path, and for unreachable accuracy, which measures how often the model was able to identify the target is not reachable from the goal (unlike our examples, some of the game instances generated did not have a solution).

The authors found that on 6×6 single goal grids, the fine-tuned T5 model generally outperforms all the GPT-4 prompting variants (and performs best overall), and all the fine-tuned models had some success in unreachable accuracy while the GPT-4 variants had no success. The T5 model also tended to generalize well to smaller grids, but performance suffered somewhat for multi-goal or when using 6-11 obstacles, and performance dropped significantly when the grid was 7×7. ReAct was also the most successful prompting method and performance remained the same on 7×7 grids and dropped somewhat with more obstacles or a multi-goal setup.

Notably, our work differs in that we investigate simultaneously learning multiple games via fine-tuning and we explore if this helps generalization to out-of-distribution settings. Although it is likely that prompting GPT-4 through in-context learning will yield better performance, this approach is not lightweight nor easily examinable: the best performing prompt variation is ReAct which requires multiple queries, and, unless GPT-4's implementation is open sourced, there is no easy method of inspecting its operation to examine failure cases or develop improvements.

4 Models

We fine-tuned a pretrained T5 model from Huggingface [23] for 25 epochs, selecting the best performance on the validation split to report results on the test split. An example of the inputs and outputs is as follows:

Input: activated_beamer: tags: obstacle teleport activation; Navigate a 5x5 grid with blocker cells and activated beamer cells, which allow you to teleport only between beamer cells once activated, from (0, 0) to (3, 4). You can only move up, down, left, or right. A blocker cell is [a blocker cell: you cannot travel to these cells]. The blocker cells are located at (0, 1), (1, 3), (1, 4), (2, 2), (3, 0), (4, 0), (4, 4). A beamer cell is [a beamer cell: after landing on this cell and staying for 1 step, you can teleport to any other beamer cell 1 step later]. The beamer cells are located at (1, 1), (2, 4), (3, 2), (3, 3).
Output: (0, 0)(1, 0)(1, 1)(1, 1)(2, 4)(3, 4)

For GPT-4, we used Chain of Thought prompting and 4 in-context examples. For games that have special cells, the first 2 examples featured solution paths that only used regular cells, while the last 2 examples had paths making use of the special cells. We did not test ReAct, despite the results in [1] due to needing to prompt multiple times for a single path, which we believed to require excessively more compute than inference on T5 (and hence is not a good comparison).

Our chain of thought inputs and outputs are formatted similar to the following:

Input: Navigate a 5x5 grid with blocker cells and activated beamer cells, which allow you to teleport only between beamer cells once activated, from (0, 1) to (1, 0). You can only move up, down, left, or right. A blocker cell is [a blocker cell: you cannot travel to these cells]. The blocker cells are located at (4, 1). A beamer cell is [a beamer cell: after landing on this cell and staying for 1 step, you can teleport to any other beamer cell 1 step later]. The beamer cells are located at (1, 1), (3, 0).
Output: (1, 0) is 1 left and 1 down from (0, 1). We can directly follow such a path. This corresponds with the sequence: (0, 1)(0, 0)(1, 0)

5 Results

For each of our datasets, we tracked the following statistics for all models' generated paths:

1. **Invalid Syntax**: Percentage of output strings that cannot be parsed
2. **Valid**: Percent of paths that do not violate any properties (i.e. traveling on a cell that contains an obstacle from the set $\mathbb{O}$) or underlying constraints C of the problem. The path does not have to reach the target.
3. **Special Path**: Percent of paths that performed a special move from one of the special cells. The path does not have to reach the target, but does have to be valid.
4. **Reach Goal**: Percent of paths that reach the goal from the starting location while satisfying all of C of the problem.
5. **Optimal**: Percent of paths that are no longer than that generated via Dijkstra's algorithm.
6. **Exact**: Percent of paths exactly the same as the optimal path observed in Dijkstra's algorithm.

5.1 Multitask Finetuned T5 Outperforms GPT-4

Across most datasets, training the T5 model on all the datasets resulted in more paths reaching the goal, more paths reaching the goal optimally, and more paths exactly matching a path from Dijkstra's algorithm than prompting GPT-4. This shows that "smaller" LLMs can learn multiple tasks. Perhaps this is unsurprising given results in information extraction tasks in NLP that use the same set of model weights for different roles [18].

Outside of prereq, all problem formulations had agents who generated valid paths with probabilities between 0.68 and 0.79 (and similarly reached the goal state within the same bounds). Perhaps most surprisingly, the "easy" obstacles case was not the one with highest percentage of valid paths, potentially due to the additional special cells leading to easier-to-optimize problems. For example, the beamer environment, while on paper more complicated than the obstacle environment due to the larger search space of potential paths, is significantly easier from a human perspective since one can focus on optimizing a minimum path (P_0, P_1), or optimize the minimum path (P_0, S_1) since the beamer cell allows for movement to any other cell, including the target cell P_1.

Table 2: **Benchmarks** for T5 fine-tuned on all datasets across each environment on the test split. We do not report "valid syntax" since the model performed perfectly here. Bolded results are the highest percentage in their respective category. INVALID results are not included since output was always parsable.

Dataset	VALID	SPECIAL PATH	REACH GOAL	OPTIMAL	EXACT
OBSTACLES	70.00	0.00	70.00	65.00	56.00
TELEPORT	77.00	8.00	77.00	62.00	51.00
BEAMER	78.00	**20.00**	77.00	69.00	61.00
ACTIVATED BEAMER	68.00	7.00	68.00	63.00	58.00
BEAMER GROUP	**79.00**	7.00	**79.00**	67.00	58.00
DIAGONAL	78.00	0.00	76.00	72.00	**64.00**
HOP	**79.00**	0.00	**79.00**	**76.00**	63.00
ACTIVATED HOP	74.00	10.00	74.00	66.00	58.00
PREREQ	36.00	0.00	20.00	10.00	5.00

Table 3: **Benchmarks** for Chain of Thought GPT-4 prompting on the test split. Bolded results are the highest percentage in their respective category, except for INVALID where lower percentages are better.

Dataset	INVALID	VALID	SPECIAL PATH	REACH GOAL	OPTIMAL	EXACT
OBSTACLES	4.00	71.00	0.00	71.00	69.00	17.00
TELEPORT	2.00	70.00	70.00	62.00	49.00	8.00
BEAMER	2.00	70.00	32.00	70.00	58.00	46.00
ACTIVATED BEAMER	**1.00**	68.00	**55.00**	68.00	29.00	22.00
BEAMER GROUP	7.00	63.00	25.00	63.00	46.00	32.00
DIAGONAL	10.00	**75.00**	0.00	**75.00**	**72.00**	**66.00**
HOP	3.00	58.00	0.00	58.00	53.00	17.00
ACTIVATED HOP	5.00	61.00	37.00	61.00	42.00	34.00
PREREQ	3.00	35.00	0.00	33.00	17.00	2.00

As for prereq, it seems to be a difficult version of path-planning given GPT-4's performance here. Certainly, solving it optimally seems to require time exponential to the number of prerequisite cells. We conjecture the main challenge lies in this problem requiring longer horizon planning. For example, the average path length was 14.4, compared to 4.7 for all other datasets. An empirical result that can be observed from the table is that this was the only setting with more than 1% between the percentage of valid paths and percentage of paths that reached the goal. Looking closer at the generated outputs, we often observed the model's path getting "stuck" in a loop.

5.1.1 GPT-4 has Some Probability of Producing Malformed Outputs

Although not significant, we observed that GPT-4's outputs were sometimes malformed. The percentage of malformed outputs was highest for the diagonal environment. Checking the malformed outputs, we observed a lot of them consisting of paths that moved "regularly" (so directly up, down, left, or right). We hypothesize that such movements represent an overwhelming majority of grid navigation scenarios, and this results in biased training data for the model.

5.1.2 GPT-4 is Much More Prone to Using Special Cells

Perhaps the most drastic difference between T5 and GPT-4 was the latter consistently produced more paths that used special cells across all datasets. Notably, the percentage of special paths was even higher than what was in the test split. We initially hypothesized that this might be due to the last 2

in-context examples being examples that used special paths, which some experiments suggest biases the model [25]. However, when we switched to the first 2 in-context examples being the ones with special paths, the results changed an insignificant amount.

5.2 Generalizing to Unseen Games

We additionally tested the ability of the T5 model to solve games not seen during training. In particular, we investigated if it could solve beamer group, prereq, and diagonal, while utilizing all remaining datasets to train the model. Specifically, we train on all but 1 game and test on the 1 held out game. Beamer group was chosen given that it was the best performing game (based on percentages of valid paths and paths that reached the goal) and perhaps was easiest. Prereq was chosen due to its difficulty. Diagonal was chosen to really test if the model was processing the input prompt (as opposed to pattern matching the prompt and "memorizing" games and deciding what to output).

Table 4: **Results for Evaluating on Unseen Games** for T5. We do not report "valid syntax" since the model performed perfectly here.

Dataset	VALID	SPECIAL PATH	REACH GOAL	OPTIMAL	EXACT
BEAMER GROUP	68.00	19.00	68.00	64.00	57.00
PREREQ	86.00	0.00	3.00	3.00	3.00
DIAGONAL	0.00	0.00	0.00	0.00	0.00

Unsurprisingly, the beamer group results are somewhat worse than when the training set included the environment 3. Interestingly, many more generated paths used teleporting. We think this indicates one of 2 possibilities:

1. This is similar to how GPT-4 aggressively uses special cells when exposed to a new prompt (a prompt it never saw during training)
2. The model is able to pattern match beamer group to beamer (which it does see during training) from the input instructions and pretends it's actually solving an instance of a beamer game

We believe the latter to be more likely given the similar proportion of paths generated using special cells in beamer (when trained on all datasets) and in beamer group (in this setting). The fact that the model fails to actually understand the prompt is shown in the results for diagonal. This setting is trivial generalization beyond left, right, up, down grid traversals for humans and the previous results in 3 indicate it is likely easy for the model too. However, the model fails to generate a valid path. Upon manual inspection of some outputs, we saw the paths corresponded to an obstacle setting. This strongly suggests that the model memorizes the format of games to decide how to operate as opposed to processing the prompts each time.

Finally, although significantly fewer paths for prereq successfully reach the target, many more of these paths are valid. Looking at the generated paths, we observe better avoidance of obstacles and most paths reaching the target location but not visiting all the prerequisite cells. We hypothesize that, because the model is encouraged to generate long paths when training prereq, generated paths are more likely to run into obstacles. In this setting, the model might believe it is navigating an obstacle environment, and hence is biased to avoid obstacles.

6 Conclusions and Future Work

In this paper, we introduce 8 new synthetic datasets that expand upon existing grid-style path planning problems. Most of them feature non-trivial movement or constraints to challenge models beyond path-finding in an obstacle-course-like setting.

We also extend previous work and show that models can be fine-tuned to learn all these variations at once and can even outperform much larger LLMs such as GPT-4. Additionally, we showed that

fine-tuned models likely pattern-match inputs, as opposed to understanding them like in human reading, and therefore fail to generalize to environments not seen during training.

Something to investigate more thoroughly might be analyzing how scaling models affect performance. Given that GPT-4 shows impressive performance with only in-context learning, perhaps fine-tuning models larger than T5 (for example Llama-2 [21]) could significantly improve performance. This would also require significantly more data, but this is easily obtainable given the synthetic nature of most planning datasets.

Methods for solving reasoning problems might be another area worth investigating in order to improve path planning. Works such as [13, 3] propose breaking down complex tasks into smaller components that are more easily solvable by existing methods. While prompting methods such as Chain of Thought and ReAct emulate this, perhaps there are better ways to simplify planning problems than just splitting by timesteps.

References

[1] Mohamed Aghzal, Erion Plaku, and Ziyu Yao. Can large language models be good path planners? a benchmark and investigation on spatial-temporal reasoning, 2023.

[2] Michael Ahn, Anthony Brohan, Noah Brown, Yevgen Chebotar, Omar Cortes, Byron David, Chelsea Finn, Keerthana Gopalakrishnan, Karol Hausman, Alexander Herzog, Daniel Ho, Jasmine Hsu, Julian Ibarz, Brian Ichter, Alex Irpan, Eric Jang, Rosario Jauregui Ruano, Kyle Jeffrey, Sally Jesmonth, Nikhil Jayant Joshi, Ryan C. Julian, Dmitry Kalashnikov, Yuheng Kuang, Kuang-Huei Lee, Sergey Levine, Yao Lu, Linda Luu, Carolina Parada, Peter Pastor, Jornell Quiambao, Kanishka Rao, Jarek Rettinghouse, Diego M Reyes, Pierre Sermanet, Nicolas Sievers, Clayton Tan, Alexander Toshev, Vincent Vanhoucke, F. Xia, Ted Xiao, Peng Xu, Sichun Xu, and Mengyuan Yan. Do as i can, not as i say: Grounding language in robotic affordances. In *Conference on Robot Learning*, 2022.

[3] Anonymous. Autogen: Enabling next-gen LLM applications via multi-agent conversation. In *Submitted to The Twelfth International Conference on Learning Representations*, 2023. under review.

[4] Yejin Bang, Samuel Cahyawijaya, Nayeon Lee, Wenliang Dai, Dan Su, Bryan Wilie, Holy Lovenia, Ziwei Ji, Tiezheng Yu, Willy Chung, Quyet V. Do, Yan Xu, and Pascale Fung. A multitask, multilingual, multimodal evaluation of chatgpt on reasoning, hallucination, and interactivity, 2023.

[5] Britney Brown. An application of reinforcement learning techniques in traditional pathfinding, 2022.

[6] Tom Brown, Benjamin Mann, Nick Ryder, Melanie Subbiah, Jared D Kaplan, Prafulla Dhariwal, Arvind Neelakantan, Pranav Shyam, Girish Sastry, Amanda Askell, Sandhini Agarwal, Ariel Herbert-Voss, Gretchen Krueger, Tom Henighan, Rewon Child, Aditya Ramesh, Daniel Ziegler, Jeffrey Wu, Clemens Winter, Chris Hesse, Mark Chen, Eric Sigler, Mateusz Litwin, Scott Gray, Benjamin Chess, Jack Clark, Christopher Berner, Sam McCandlish, Alec Radford, Ilya Sutskever, and Dario Amodei. Language models are few-shot learners. In H. Larochelle, M. Ranzato, R. Hadsell, M.F. Balcan, and H. Lin, editors, *Advances in Neural Information Processing Systems*, volume 33, pages 1877–1901. Curran Associates, Inc., 2020.

[7] Sébastien Bubeck, Varun Chandrasekaran, Ronen Eldan, Johannes Gehrke, Eric Horvitz, Ece Kamar, Peter Lee, Yin Tat Lee, Yuanzhi Li, Scott Lundberg, Harsha Nori, Hamid Palangi, Marco Tulio Ribeiro, and Yi Zhang. Sparks of artificial general intelligence: Early experiments with gpt-4, 2023.

[8] Yongchao Chen, Jacob Arkin, Charles Dawson, Yang Zhang, Nicholas Roy, and Chuchu Fan. Autotamp: Autoregressive task and motion planning with llms as translators and checkers, 2023.

[9] Yuqing Du, Olivia Watkins, Zihan Wang, Cédric Colas, Trevor Darrell, Pieter Abbeel, Abhishek Gupta, and Jacob Andreas. Guiding pretraining in reinforcement learning with large language models. In Andreas Krause, Emma Brunskill, Kyunghyun Cho, Barbara Engelhardt, Sivan

Sabato, and Jonathan Scarlett, editors, *Proceedings of the 40th International Conference on Machine Learning*, volume 202 of *Proceedings of Machine Learning Research*, pages 8657–8677. PMLR, 23–29 Jul 2023.

[10] Geesara Kulathunga. A reinforcement learning based path planning approach in 3d environment. *Procedia Computer Science*, 212:152–160, 2022.

[11] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[12] Mike Lewis, Yinhan Liu, Naman Goyal, Marjan Ghazvininejad, Abdelrahman Mohamed, Omer Levy, Veselin Stoyanov, and Luke Zettlemoyer. BART: Denoising sequence-to-sequence pre-training for natural language generation, translation, and comprehension. In *Proceedings of the 58th Annual Meeting of the Association for Computational Linguistics*, pages 7871–7880, Online, July 2020. Association for Computational Linguistics.

[13] Zhengzhong Liang, Steven Bethard, and Mihai Surdeanu. Explainable multi-hop verbal reasoning through internal monologue. In *Proceedings of the 2021 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 1225–1250, Online, June 2021. Association for Computational Linguistics.

[14] Gary Marcus, Evelina Leivada, and Elliot Murphy. A sentence is worth a thousand pictures: Can large language models understand human language?, 2023.

[15] Karthik Narasimhan, Regina Barzilay, and Tommi Jaakkola. Grounding language for transfer in deep reinforcement learning. *J. Artif. Int. Res.*, 63(1):849–874, sep 2018.

[16] OpenAI. Gpt-4 technical report, 2023.

[17] Aleksandr I. Panov, Konstantin S. Yakovlev, and Roman Suvorov. Grid path planning with deep reinforcement learning: Preliminary results. *Procedia Computer Science*, 123:347–353, 2018.

[18] Giovanni Paolini, Ben Athiwaratkun, Jason Krone, Jie Ma, Alessandro Achille, Rishita Anubhai, Cicero Nogueira dos Santos, Bing Xiang, and Stefano Soatto. Structured prediction as translation between augmented natural languages. In *9th International Conference on Learning Representations, ICLR 2021*, 2021.

[19] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer. *Journal of Machine Learning Research*, 21(140):1–67, 2020.

[20] Colin Raffel, Noam Shazeer, Adam Roberts, Katherine Lee, Sharan Narang, Michael Matena, Yanqi Zhou, Wei Li, and Peter J. Liu. Exploring the limits of transfer learning with a unified text-to-text transformer, 2023.

[21] Hugo Touvron, Louis Martin, Kevin R. Stone, Peter Albert, Amjad Almahairi, Yasmine Babaei, Nikolay Bashlykov, Soumya Batra, Prajjwal Bhargava, Shruti Bhosale, D. Bikel, Lukas Blecher, Cristian Cantón Ferrer, Moya Chen, Guillem Cucurull, David Esiobu, Jude Fernandes, Jeremy Fu, Wenyin Fu, Brian Fuller, Cynthia Gao, Vedanuj Goswami, Naman Goyal, A. Hartshorn, Saghar Hosseini, Rui Hou, Hakan Inan, Marcin Kardas, Viktor Kerkez, Madian Khabsa, Isabel M. Kloumann, A. Korenev, Punit Singh Koura, Marie-Anne Lachaux, Thibaut Lavril, Jenya Lee, Diana Liskovich, Yinghai Lu, Yuning Mao, Xavier Martinet, Todor Mihaylov, Pushkar Mishra, Igor Molybog, Yixin Nie, Andrew Poulton, Jeremy Reizenstein, Rashi Rungta, Kalyan Saladi, Alan Schelten, Ruan Silva, Eric Michael Smith, R. Subramanian, Xia Tan, Binh Tang, Ross Taylor, Adina Williams, Jian Xiang Kuan, Puxin Xu, Zhengxu Yan, Iliyan Zarov, Yuchen Zhang, Angela Fan, Melanie Kambadur, Sharan Narang, Aurelien Rodriguez, Robert Stojnic, Sergey Edunov, and Thomas Scialom. Llama 2: Open foundation and fine-tuned chat models, 7 2023.

[22] Karthik Valmeekam, Matthew Marquez, Alberto Olmo, Sarath Sreedharan, and Subbarao Kambhampati. Planbench: An extensible benchmark for evaluating large language models on planning and reasoning about change, 2023.

[23] Thomas Wolf, Lysandre Debut, Victor Sanh, Julien Chaumond, Clement Delangue, Anthony Moi, Pierric Cistac, Tim Rault, Rémi Louf, Morgan Funtowicz, Joe Davison, Sam Shleifer, Patrick von Platen, Clara Ma, Yacine Jernite, Julien Plu, Canwen Xu, Teven Le Scao, Sylvain Gugger, Mariama Drame, Quentin Lhoest, and Alexander M. Rush. Huggingface's transformers: State-of-the-art natural language processing, 2020.

[24] Danyang Zhang, Lu Chen, Situo Zhang, Hongshen Xu, Zihan Zhao, and Kai Yu. Large language models are semi-parametric reinforcement learning agents, 2023.

[25] Tony Zhao, Eric Wallace, Shi Feng, Dan Klein, and Sameer Singh. Calibrate before use: Improving few-shot performance of language models. In *International Conference on Machine Learning*, 2021.

[26] Mert Çetinkaya. Multi-agent path planning using deep reinforcement learning, 2021.