TeamName at SemEval-2025 Task 8: Question-Answering over Tabular Data

Anonymous ACL submission

Abstract

SemEval-2025 Task 8 focuses on querying tabular data using code generated by large language models(LLMs). The dataset for this task consists of natural language questions, relevant column information and corresponding answers. The challenge primarily lies in the multiple-step reasoning process of transforming natural language queries into executable code. This challenge is enlarged by the limitations of current approaches, such as Chain-of-Thought reasoning, which struggle with complex multi-step reasoning path because of the difficulty of evaluation of intermediate steps. To address these limitations, we propose a Monte Carlo Tree Search (MCTS) algorithm to simulate the reasoning process during code generation. It includes rewarding, regret and planning mechanism, which enables LLMs to explore multiple reasoning paths, evaluate outcomes and refine the code.

1 Introduction

The task aims to implement a code-generating large language model(LLM) to query tabular data. The goal is to implement the generated code Python to answer questions with regard to a given data frame. The training data provided by the organizers contains 960 data points. Each includes a natural language question, the target answer and its data type, relevant columns with their data types, and associated data frame. The aim is to predict the answer to each question based on the generated Python code, which extracting the required information from the data frame(Osés Grijalba et al., 2024). [1]

We propose treating LLM as a step-by-step reasoning agent which decomposes the task into subtasks, making look-ahead planning and iteratively debugs to reach the goal. In practice a programmer usually write several lines of code to narrow down the data to a single column or several columns, and might also include debugging along the way. This differs from the reasoning path of current code LLMs trained based on auto regressive reasoning manner like Chain-of-Thought, which follows linear, greedy reasoning paths[slots for code llm]. This discrepancy introduces challenges in terms of evaluation of intermediate states, lack of planning mechanism based on back propagated rewarding of intermediate reasoning states, and also lack of mechanism to explore the promising reasoning paths(Zhang et al., 2023).

To address these challenges, we propose a Monte Carlo Tree Search(MCTS) algorithm to simulate the planning and debugging process. MCTS iteratively build a branching reasoning tree by selecting the most promising reasoning steps based on the potential rewarding and the balance between exploration and exploitation. This approach enables LLM to refine its reasoning path and improve its abilities to reach accurate solutions(Gao et al., 2024).

2 Related Work

LLMs have remarkable reasoning abilities in tackling complex mathematical, logical and programming tasks(OpenAI, 2024). Approaches have been focused on simulating the step-by-step decomposition of reasoning process, which aim to mimic human cognitive processes by generating coherent reasoning path that leads to the correct outcomes(Wei et al., 2023). Planning has emerged as a crucial aspect of LLM reasoning, especially for tasks which require multi-step problem solving and long-term coherence. Unlike traditional autoregressive approaches, in which each steps is solely based on the previous tokens or sequences, planning introduces a forward-looking strategy which evaluate multiple possible approaches and select the most promising ones(Hao et al., 2024). In this sec-

[1] Our code is available at: https://github.com/HuixinYang/SemEval25-Task8-Adrianna-Aakarsh-Yang

tion, we formalize and compare several planning approaches to illustrate their impact on improving the reasoning capabilities of LLMs.

2.1 CoT and its variant

Chain-of-Thought approach follows an implicit greedy planning policy. Each intermediate state is sampled by $a_i \sim p_{LLM}(a_0 \dots a_{i-1} \mid x)$, and the solution is sampled by $y \sim p_{LLM}(y \mid x, a_0 \dots a_n)$(Wei et al., 2023). To increase the robustness of this greedy approach, Chain-of-Thought with self consistency(CoT-SC) introduced the self-consistent planning by selecting the most frequent path among set of reasoning chains. It is based on the intuition that the correct answer of a complex reasoning problem might from multiple reasoning paths(Wang et al., 2023). CoT-SC iteratively samples i CoT independent reasoning chains by $[a_0 \dots a_n, y]^i \sim p_{\text{CoT},LLM}(x)$, and selects the most frequent path $\arg\max_{a_0 \dots a_n, y} \sum_{i=0}^{n} \mathbb{1}\{y_i = y\}$. The underlying autoregressive mechanism of these approaches introduces the difficulties of evaluation the intermediate reasoning steps and exploration of more promising path from the less likely steps(Sprague et al., 2024).

Its application in code generating tasks, the challenge of intermediate steps is more pronounced due to the non-deterministic property of code generation(Ouyang et al., 2024), which introduces difficulties of evaluating the intermediate steps to reduce false positive. Approaches has been done in breaking down the task and interact between code and natural language multiple times until it passes the unit tests(Ding et al., 2024).Besides this back-and-forth process, models are also encouraged to perform deeper reasoning by introducing increasing difficulty in the tasks which aims at ultimately improving their ability to generate robust code(Luo et al., 2023).

2.2 Tree reasoning

Inspired by deliberate reasoning, Tree-of-Thought(ToT) reasoning structures the reasoning paths as a branching tree, which allows parallel reasoning across paths(Yao et al., 2023). However, it locally selects the next state greedily based on votes or score from the model, which is $\arg\max_s p_{value,LLM}(v \mid s)$ or $\arg\max_s p^{\text{vote}}_{LLM}(s)$. Monte Carlo Tree Reasoning with Planning improves this by allowing LLM to plan a reasoning path based on the world model and back propagated rewards which iteratively build a tree policy, which enables a human-like deliberate reasoning strategies(Hao et al., 2023). MCTS can handle the intermediate steps challenges by balancing exploration and exploitation, offering a more robust approach compared to greedy methods.

3 Our Approach

We adopt the Reasoning via Planning(RAP) framework to build a Monte Carlo Tree Search algorithm for our reasoning process(Hao et al., 2023). In this framework, each node represent a state, and each edge denotes an action from current state to the next. The aim is to guide the LLM to search for a path which maximizes accumulated rewarding, $\sum_{t=0}^{n} r(s_t, at)$. Each iteration of look-ahead planning contains two phases. First, the algorithm selects the next node roughly proportion to its back-propagated rewarding along the visited path, and also adjusted with an exploration term(see session 3.1) to balance the exploration and exploitation. Second, once reaching the leaf node, the next node is selected either randomly or greedily. The first phase expanding the tree step-by-step which enables the simulation through look-ahead planning that is not solely greedy as in autoregressive approach but instead based on estimating the expected future reward of taking action a in state s, $Q(s, a)$, which is updated in the backpropagation phase of each iteration. The following is an detailed illustration about its steps.

3.1 Selection

This process starts at the root node and recursively select its child node until a leaf node is reached. It is based on Upper Confidence Bound for Trees (UCT) to balance exploration and exploitation(Kocsis and Szepesvári, 2006).

$$UCT(s) = \hat{Q}(s, a) + C\sqrt{\frac{\ln N(s)}{N(c(s, a))}} \quad (1)$$

3.2 Expansion

New child node is selected by sampling d possible reasoning path.

3.3 Simulation

The model samples randomly to simulate the rollout process until a terminal node is reached with default policy.

3.4 Back Propagation

Once a terminal node is reached, the rewarding is back propagated up to the tree to update the values of the visited nodes to refine the tree policy.

References

Yangruibo Ding, Jinjun Peng, Marcus J. Min, Gail Kaiser, Junfeng Yang, and Baishakhi Ray. 2024. Semcoder: Training code language models with comprehensive semantics reasoning. *Preprint*, arXiv:2406.01006.

Zitian Gao, Boye Niu, Xuzheng He, Haotian Xu, Hongzhang Liu, Aiwei Liu, Xuming Hu, and Lijie Wen. 2024. Interpretable contrastive monte carlo tree search reasoning. *Preprint*, arXiv:2410.01707.

Shibo Hao, Yi Gu, Haotian Luo, Tianyang Liu, Xiyan Shao, Xinyuan Wang, Shuhua Xie, Haodi Ma, Adithya Samavedhi, Qiyue Gao, Zhen Wang, and Zhiting Hu. 2024. Llm reasoners: New evaluation, library, and analysis of step-by-step reasoning with large language models. *Preprint*, arXiv:2404.05221.

Shibo Hao, Yi Gu, Haodi Ma, Joshua Jiahua Hong, Zhen Wang, Daisy Zhe Wang, and Zhiting Hu. 2023. Reasoning with language model is planning with world model. *Preprint*, arXiv:2305.14992.

Levente Kocsis and Csaba Szepesvári. 2006. Bandit based monte-carlo planning. In *Machine Learning: ECML 2006*, pages 282–293, Berlin, Heidelberg. Springer Berlin Heidelberg.

Ziyang Luo, Can Xu, Pu Zhao, Qingfeng Sun, Xiubo Geng, Wenxiang Hu, Chongyang Tao, Jing Ma, Qingwei Lin, and Daxin Jiang. 2023. Wizardcoder: Empowering code large language models with evol-instruct. *Preprint*, arXiv:2306.08568.

OpenAI. 2024. Gpt-4 technical report. *Preprint*, arXiv:2303.08774.

Jorge Osés Grijalba, L. Alfonso Ureña-López, Eugenio Martínez Cámara, and Jose Camacho-Collados. 2024. Question answering over tabular data with DataBench: A large-scale empirical evaluation of LLMs. In *Proceedings of the 2024 Joint International Conference on Computational Linguistics, Language Resources and Evaluation (LREC-COLING 2024)*, pages 13471–13488, Torino, Italia. ELRA and ICCL.

Shuyin Ouyang, Jie M. Zhang, Mark Harman, and Meng Wang. 2024. An empirical study of the non-determinism of chatgpt in code generation. *ACM Transactions on Software Engineering and Methodology*.

Zayne Sprague, Fangcong Yin, Juan Diego Rodriguez, Dongwei Jiang, Manya Wadhwa, Prasann Singhal, Xinyu Zhao, Xi Ye, Kyle Mahowald, and Greg Durrett. 2024. To cot or not to cot? chain-of-thought helps mainly on math and symbolic reasoning. *Preprint*, arXiv:2409.12183.

Xuezhi Wang, Jason Wei, Dale Schuurmans, Quoc Le, Ed Chi, Sharan Narang, Aakanksha Chowdhery, and Denny Zhou. 2023. Self-consistency improves chain of thought reasoning in language models. *Preprint*, arXiv:2203.11171.

Jason Wei, Xuezhi Wang, Dale Schuurmans, Maarten Bosma, Brian Ichter, Fei Xia, Ed Chi, Quoc Le, and Denny Zhou. 2023. Chain-of-thought prompting elicits reasoning in large language models. *Preprint*, arXiv:2201.11903.

Shunyu Yao, Dian Yu, Jeffrey Zhao, Izhak Shafran, Thomas L. Griffiths, Yuan Cao, and Karthik Narasimhan. 2023. Tree of thoughts: Deliberate problem solving with large language models. *Preprint*, arXiv:2305.10601.

Shun Zhang, Zhenfang Chen, Yikang Shen, Mingyu Ding, Joshua B. Tenenbaum, and Chuang Gan. 2023. Planning with large language models for code generation. *Preprint*, arXiv:2303.05510.

A Example Appendix

This is an appendix.