Generative Problem Solving

Bridging Classic and Modern AI

Hsiang-Shun Shih, Chengheng Lyu, Goli Vaisi, Phillip C-Y Sheu
Department of EECS, University of California, Irvine, USA

Abstract*—While *Large Language Models (LLMs) have strong natural language understanding capabilities, they continue to face significant challenges in solving complex computational problems. In this position paper, we argue that a core limitation lies in the underdeveloped ability of LLMs to generate and explore diverse, meaningful problems while solving them with deeper reasoning. To address this gap, we introduce the concept of Generative Problem Solving (GPS) that emphasizes the capability of LLM for solving problems that require algorithmic reasoning.

Keywords— Generative Problem Solving, Large Language Model, Modern AI, Classic AI

I. Introduction

This paper introduces *Generative Problem Solving* (*GPS*) that has the capability of solving computational problems and generating new problems. For problems that GPS cannot solve, it can learn from solutions provided by the user. While GPS and generative AI deal with the act of "generation," GPS narrows the notion of "generative" to focus on computational problems and problem solving. In contrast, generative AI more broadly targets the creation of novel outputs that mimic or extend existing data patterns extracted from texts and multimodal data [1]. GPS solves new computational problems by systematically creating them and solving them step by step, as well as increasing transparency so that users can follow the reasoning path. With user-in-the-loop, GPS may significantly boost machine's learning curve and solution quality.

Despite the impressive natural language understanding capabilities of Large Language Models (LLMs), significant challenges remain in their development, particularly their limitations in solving complex problems that require logical reasoning or computational algorithms. LLMs currently lack the capability to explore the full range of possible problems and fail to ensure the correctness of solutions by providing concrete, step-by-step reasoning processes. GPS aims to bridge this gap by empowering LLMs with advanced problem-solving capabilities.

II. Large Language Models

The current large language models (LLMs) are built on transformer architectures [2]. Transformers function as next-word predictors: given a user prompt as the context, the LLM iteratively predicts the next word and appends it to the existing context. Our research uses the state-of-the-art Llama3.3 model [3] for experiments. Models like Llama3.3 have demonstrated strong natural language understanding and general question-answering capabilities, as evidenced by their performance on established benchmarks such as GLUE (which evaluates linguistic comprehension across diverse tasks [4]), SquAD (a reading comprehension dataset for contextual question answering [5]), and TriviaQA (which assesses open-domain question answering [6]). Our objective is to further enhance the problem solving and reasoning abilities of LLMs.

III. Generative Problem Solving

GPS, as a problem solver, comprises three core components: problem-solving, new problem generation, and learning from user solutions. We argue that the ability of systematically generating new problems and learning from user-provided solutions enables the expansion of GPS's knowledge base to serve as a solid foundation for addressing similar problems.

In this position paper, we highlight some key limitations of current LLMs, including their underdeveloped reasoning abilities, their tendency to overlook unsolvable problems, and their limitations on performing planning tasks that were a center piece of classic AI. We then underscore the importance of generating new problems—for example, through processes like question extraction, decomposition, and synthesis—as a catalyst for more advanced problem solving. By framing LLM around structured problem generation, future research can enable LLMs to handle more complex problems with greater transparency and reliability.

A-1 Limitations of LLMs on Reasoning

Llama3.3 is an exceptional model with diverse capabilities, including accurate program generation, article writing, and demonstrating professional knowledge. However, based on our experiments, we believe that true reasoning ability is difficult to achieve solely through next-token prediction. Llama3.3 lacks the basic resolution capability, for example. To address this, we are working to develop a logic-guided question generative method to enhance the reasoning ability of an LLM.

A-2 Limitations of LLMs on Planning

We experimented how Llama3.3 approaches a classic blocks world problem. Given the initial state, goal state, and allowed moves, the LLM successfully provides a solution that achieves the goal state. Our observations suggest that if a problem is similar to a well-known example seen during training, LLMs like Llama3.3 can make minor adjustments to produce acceptable but not optimal solutions. However, in more complex scenarios, the model struggles with true planning and fails to generate valid solutions.

A-3 Limitations of LLMs on Combinatorial Problem Solving

A combinatorial problem finds solutions for an optimal arrangement or ordering of distinct elements based on specific rules. Sorting is a classic example of a combinatorial problem, requiring the rearrangement of elements into a defined order. The Graph of Thoughts (GoT) framework [7] aims to enhance the reasoning capabilities of LLMs by representing information as a graph, where its nodes correspond to individual thoughts and edges denote their dependencies. This structure allows

LLMs to process complex tasks, such as sorting, by decomposing them into interconnected reasoning steps. The framework has demonstrated success in validating its sorting ability in experiments. To further strengthen GoT's effectiveness, we propose integrating a more structured logical system to ensure consistent rule-based reasoning. By embedding logical principles directly, LLMs can achieve greater accuracy and reliability, particularly in solving combinatorial problems which require logical reasoning.

A-4 Limitations of LLMs on Relational Problem Solving

While LLMs have been employed for translating NLQs (Natural Language Queries) into SQL, they face notable limitations, particularly when facing complex queries and unseen schemas. For instance, RAT-SQL [8] utilizes LLMs for query conversion. It is trained on the Spider dataset that includes a variety of pre-defined schemas, queries and the corresponding SQL queries across different databases. The ability of RAT-SQL heavily relies on the previously seen data in the training dataset. The model still struggles to adapt to real-world scenarios including situations where schemas may be new or evolve over time. To address this, we plan to leverage GPS to explore semantic resources, enabling LLMs to interact with specific problem solvers.

B. Problem Generation

As far as we know, few papers have explored research idea generation [9][10][11], but none explicitly discuss the process of problem generation. Paper [11] is somewhat similar to our approach, as it decomposes research problems from existing papers and generates research ideas based on the decomposed results. It uses LLMs to analyze research papers, extracting semi-structured questions by identifying key concepts and mapping them across varying levels of abstraction. We tried to reproduce their extraction process using Llama3.3 to extract the research problems from the original transformer paper, Attention Is All You Need [2]. While the original Transformer model was designed to address long-distance dependency challenges in sequence transduction through self-attention, our experiments show that Llama3.3's output overlooked the key mechanism for problem generation, focusing instead on parallelization and training times, emphasizing the need for more precise extraction processes. To advance LLMs' ability to generate new problems, we plan to leverage semantic graph representations integrated with logic-based systems. This would enable the extracted problems to be more structured in order to be expressed with greater precision, facilitating deeper semantic understanding and ensuring alignment with complex research contexts.

C. Solution Synthesis

Reasoning was interpreted as "a dynamic process to integrate multiple knowledge to get new conclusions, rather than direct recourse to memorized or provided first-hand information" [12]. Key reasoning strategies include end-to-end reasoning, forward reasoning, and backward reasoning. Forward reasoning involves iteratively applying existing knowledge to derive new insights until the desired answers are obtained. In contrast, backward reasoning breaks problems into smaller sub-problems, solving each step progressively until the final answers are reached. While it remains uncertain whether LLMs adhere to a desired reasoning chain to arrive at the correct final answer, numerous studies have demonstrated that NLP models can be guided toward the correct answer by decomposing questions into a set of sub-questions [7][13][14][15]. These approaches heavily rely on structured inputs and predefined reasoning chains. In our approach, utilizing LLMs integrated knowledge base to generate questions iteratively eliminates the need for explicitly defined reasoning paths.

IV. Conclusions

In this position paper, we have highlighted several critical limitations of employing current LLMs as problem solvers, including their underdeveloped ability to perform complex reasoning, the tendency to overlook unsolvable problems, and shortages in generating meaningful problems.

We introduced the concept of Generative Problem Solving (GPS) to address these challenges by focusing on systematic problem generation and problem solving.

Our goal is to inspire discussions and collaborative work on bridging classic and modern AI that may be started with the transformation of LLMs into more robust, reliable, and versatile problem solvers.

References

[1] Yin, Shukang, et al. "A survey on multimodal large language models." arXiv preprint arXiv:2306.13549 (2023).

[2] Vaswani, A. "Attention is all you need." Advances in Neural Information Processing Systems (2017).

[3] Dubey, Abhimanyu, et al. "The llama 3 herd of models." arXiv preprint arXiv:2407.21783 (2024).

[4] Wang, Alex, et al. "GLUE: A multi-task benchmark and analysis platform for natural language understanding." arXiv preprint arXiv:1804.07461, 2018.

[5] Rajpurkar, Pranav, et al. "SQuAD: 100,000+ Questions for Machine Comprehension of Text." arXiv preprint arXiv:1606.05250, 2016.

[6] Joshi, Mandar, et al. "TriviaQA: A large scale dataset for reading comprehension and question answering." arXiv preprint arXiv:1705.03551, 2017.

[7] Besta, Maciej, et al. "Graph of thoughts: Solving elaborate problems with large language models." *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 38. No. 16. 2024.

[8] Wang, Bailin, et al. "Rat-sql: Relation-aware schema encoding and linking for text-to-sql parsers." arXiv preprint arXiv:1911.04942 (2019).

[9] Si, Chenglei, Diyi Yang, and Tatsunori Hashimoto. "Can llms generate novel research ideas? a large-scale human study with 100+ nlp researchers." arXiv preprint arXiv:2409.04109 (2024).

[10] Wang, Qingyun, et al. "Scimon: Scientific inspiration machines optimized for novelty." *arXiv preprint arXiv:2305.14259* (2023).

[11] Gu, Tianyang, et al. "LLMs can realize combinatorial creativity: generating creative ideas via LLMs for scientific research." arXiv preprint arXiv:2412.14141 (2024).

[12] Yu, Fei, et al. "Natural language reasoning, a survey." ACM Computing Surveys 56.12 (2024): 1-39.

[13] Wu, Jian, et al. "GenDec: A robust generative Question-decomposition method for Multi-hop reasoning." arXiv preprint arXiv:2402.11166 (2024).

[14] Wei, Jason, et al. "Chain-of-thought prompting elicits reasoning in large language models." Advances in neural information processing systems 35 (2022): 24824-24837.

[15] Yu, Jianxing, et al. "Multi-hop reasoning question generation and its application." *IEEE Transactions on Knowledge and Data Engineering* 35.1 (2021): 725-740.