How can large language models and prompt engineering be leveraged in Computer Science education?

Authors
Author: Alexandra ioana Neagu
Email: A.I.Neagu@student.tudelft.nl

Supervisors: Fenia Aivaloglou, Xiaoling Zhang

Affiliations
Technische Universiteit Delft

This systematic literature review investigates:

- What prompt engineering techniques are used to support problem solvers to modify the problem description successfully?
- What is the potential use of natural language processing (NLP) techniques in teaching and learning practices that leverage large language models (LLMs)?

01 Introduction

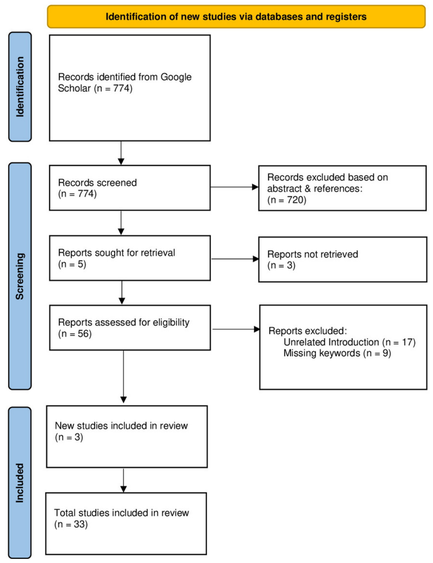
LLMs have advanced NLP and demonstrated remarkable capabilities in tasks like language generation and program repair. Effective utilization of LLMs requires prompt engineering and an understanding of their limitations. LLMs also offer potential in education for enriching learning experiences and developing computational thinking skills. This paper investigates leveraging NLP and prompt engineering to generate successful programming solutions and integrate them into the educational environment.

02 Background

LLMs, like BERT and ChatGPT, are based on the Transformer architecture, which uses attention mechanisms to capture long-range dependencies. Transformers have outperformed traditional neural networks in NLP tasks due to their attention mechanisms. LLMs employ self-attention and tokenization techniques, such as subword tokenization. BERT is trained using a Masked Language Model and is bi-directional, while ChatGPT is based on an autoregressive left-to-right Transformer. BERT is encoder-only, generating fixed-length representations, while ChatGPT is decoder-only, making it more suitable for text-generation tasks.

03 Methodology

The selection process for this systematic literature review involved using Google Scholar to search for papers relevant to the 2 sub-topics. The search queries and filters were applied to retrieve papers related to natural language processing (NLP), large language models (LLMs), program synthesis, and education. The PRISMA flow chart was used to visualize the record selection process. Inclusion criteria were established based on topic relevance, language (English), and publication date (2020 to present). Papers that met the inclusion criteria were analyzed by examining their titles, abstracts, and references, and if necessary, the full paper. Data extraction, quality assessment, synthesis of findings, and interpretation were conducted to analyze the selected papers and draw meaningful conclusions.



PRISMA Flowchart

04 Results

RQ1:

- Prompt engineering is crucial for LLMs' accuracy, coherence, and context appropriateness.
- Stylistic and structural constraints greatly impact LLMs' output quality [1].
- GPT-3 confuses style with subject matter and faces challenges with certain words and numerical constraints [1].
- Prompting techniques: zero-shot capability (without explicit training) and few-shot prompting (using demonstrations).
- Zero-shot prompting is less effective for tasks deviating from pre-training data format.
- LLMs' universal knowledge differs from specific behavioral patterns in private domain data.
- Instruction tuning combines pretrain-finetuning and prompting to enhance zero-shot performance of LLMs.
- Chain-of-thought prompting improves reasoning by generating intermediate steps for multi-step problems.
- Heuristic strategies for code generation involve rewording, expanding/reducing scope, retrying, and recalibrating targets.
- Conversational approaches and specific words like "obviously" enhance code output repair and performance in certain contexts.

RQ2:

- Agents enhance pair programming by assisting with expertise-related challenges.
- NLP techniques improve code generation.
- GPT-3 generates code explanations and aids in learning programming languages.
- Fine-tuning LLMs on domain-specific data supports learners in that domain.
- ChatGPT excels in programming tasks and learning new languages, leveraging recognized packages and online documentation in prompts [2].

05 Limitations

The study has validity threats due to time limitations, limited sources, language bias, omission of Google's Bard, and potential biases in paper retrieval. The author's lack of expertise may also impact paper selection

06 Conclusion

The study compares the architectures of BERT and ChatGPT, highlighting their respective strengths and suitability for different tasks. Prompt engineering techniques are explored to improve LLMs' responses, considering factors such as prompt length, complexity, context, and constraints. Zero-shot and few-shot prompting methods are discussed, with the latter shown to enhance model performance. NLP techniques using LLMs show promise in pair programming, code generation, explanations, and programming language learning. Familiarity with LLM syntax and providing comprehensive information are crucial for desired outputs. LLMs are reliable in detecting errors but may struggle with generating error-free code. The ease of obtaining information through LLMs can hinder critical thinking, emphasizing the need to use them as learning tools rather than substitutes for human expertise. Limitations include interpretability, biases, brittleness, and hallucinations. The training process relies on code from open-source projects, which may contain security vulnerabilities.

Related literature

[1] Albert Lu, Hongxin Zhang, Yanzhe Zhang, Xuezhi Wang, and Diyi Yang. Bounding the capabilities of large language models in open text generation with prompt constraints. arXiv preprint arXiv:2302.09185, 2023.
[2] adel M Megahed, Ying-Ju Chen, Joshua A Ferris, Sven Knoth, and L Allison Jones-Farmer. How generative ai models such as chatgpt can be (mis) used in spc practice, education, and research? an exploratory study. arXiv preprint arXiv:2302.10916, 2023.