The ability of chatGPT to generate code

1- Is ChatGPT the Ultimate Programming Assistant?

Idea:

By contrasting it with Codex and CodeGen, the study assesses ChatGPT's performance in code generation, program repair, and code summarizing. It examines ChatGPT's advantages, disadvantages, and capacity for generalization on invisible issues.

Methodology:

utilizing two programming benchmarks: Refactory (for program repair and summarizing) and LeetCode (for code generation).

To reduce randomness, numerous test attempts are made, five times for each problem.

evaluating performance utilizing both qualitative (explanation accuracy) and quantitative (success rates, time complexity, correctness) indicators.

Datasets:

The LeetCode Benchmark (2016–2020 & 2022) is used to assess code generation for a variety of problem types, including hash tables, arrays, sorting, and strings.

Benchmark for Refactory (2019): includes 2,442 correct programs for code summarizing and 1,783 incorrect Python programs submitted by students for program repair.

Evaluation metrics:

Code Generation: Correctness (AVG-5 & TOP-5): Indicates if at least one of the five solutions produced is accurate.

Ranking of Time Complexity: Evaluates effectiveness against handwritten solutions.

Program Repair: Repair Rate: The proportion of malfunctioning programs that were successfully fixed (TOP-5 & AVG-5).

Code Summarization: The Semantic Similarity (BERT-based Analysis) metric in code summarization assesses how well ChatGPT's explanations align with problem descriptions.

Results:

Code Generation: ChatGPT performs better than Codex and CodeGen, handling 95% of simple problems but having trouble with more complex, invisible ones.

Prompt engineering is essential; shorter prompts increase accuracy.

The success rate for program repair is 84%, which is comparable to that of the (%90) specialized Refectory tool.

Focused prompts are more effective than problem descriptions.

Reflection:

Contributions: compares ChatGPT's ability to generate, repair, and summarize code to more specialized models such as Codex. Limitations: ignores the demands of real-world developers: ChatGPT's usefulness in professional development contexts isn't examined, even though benchmarks evaluate it in controlled situations.
future research: Examine how developers utilize ChatGPT in their coding workflows to learn more about its application in real-world development**.**

2- Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues

Idea:

This paper investigates the quality and dependability of code created by ChatGPT by methodically examining 4,066 Python and Java code snippets from 2,033 programming tasks.

Evaluate the code generation's accuracy.
Determine typical quality problems.
Use feedback to test ChatGPT's capacity for self-healing.

Methodology:
2,033 programming LeetCode tasks (easy, medium, and hard) make up the **dataset**.

Correctness (generated code pass rates).

Problems with code quality (static analysis with Pylint, PMD, and Checkstyle tools).

Self-Repair Effectiveness (fix rate with prompts based on input).

Evaluation metrics:

Compilation and Runtime Errors, Performance and Efficiency.

Results:
Code generation accuracy ranges from 66% to 69%, however it falls by 5% on assignments after 2022.
Code that is longer (more than 50 lines) is more likely to fail.

Code issues include 4% runtime errors, 47% maintainability issues, and 27% incorrect outputs.

Self-Repair: 20–60% of problems are resolved by ChatGPT, and repairs are improved by 20% with thorough feedback.
Although iterative feedback increases accuracy, it can also result in new mistakes.

Reflection:
Contributions: gives a thorough assessment of ChatGPT's code quality, emphasizing its accuracy, maintainability, and capacity for self-healing. Limitations: The evaluation is restricted to Python and Java; further insights may be obtained by extending to additional languages (such as C++ and JavaScript).
future research: Examine ChatGPT's debugging capabilities when working on collaborative coding projects (such as integrating with GitHub Copilot).

3-Is Your Code Generated by ChatGPT Really Correct?

Idea:

By increasing test coverage, the study presents EvalPlus, a framework for thoroughly evaluating LLM-generated code. It reveals faults in LLMs that were previously unknown by extending the HUMANEVAL benchmark (used for evaluating LLM code synthesis) by 80× to create HUMANEVAL+.

Methodology:
Test-Case Generation: Generates a variety of test cases using mutation-based fuzzing and seed inputs supplied by ChatGPT.
Metrics include LLM ranking shifts, performance drop analysis, and mistake detection rate.

Datasets: Assesses 26 LLMs on HUMANEVAL+, an improved HUMANEVAL version with automated test creation (e.g., GPT-4, ChatGPT, CodeLlama, etc.).

Evaluation metrics:

Error Detection Rate, Performance Drop Analysis.

Results:
The accuracy of LLM decreases by as much as 28.9% when examined more thoroughly.
ChatGPT was mistakenly graded higher than WizardCoder & Phind-CodeLlama, indicating that HUMANEVAL overvalued certain LLMs.
HUMANEVAL's ground-truth code contained 18 errors (11%) demonstrating the flaws in the current standards.

Reflection:

Contributions: presents EvalPlus, a system for assessing LLM-generated code that significantly increases testing coverage. Limitations: Although EvalPlus improves HUMANEVAL, it must be extended to additional code synthesis benchmarks because it is benchmark-specific. Future research: Add multi-language benchmarks (such Java, C++, and JavaScript) to EvalPlus.
Optimize test-suite reduction to strike a balance between assessment effectiveness and accuracy.

4-A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages.

Idea: This study discusses the effectiveness of ChatGPT 3.5 code generating in 10 different programming languages. The study highlights ChatGPT strength and weakness in different tasks and language complexity in a pool pf 40 coding tasks. Evaluating proficiency based on time performance, exaction success rate and code quality.

Methodology: Languages tested: C, C++, Go, JavaScript, Julia, Perl, Python, R, Ruby, and Smalltalk. Tasks used for evaluation: data science, games, security and simple algorithms.

Testing: The model was asked to generate code without importing libraries, each task was tested 10 times per language resulting in total of 4000 tests, results are marked as: no code(ethical reasons), no code (other reasons), compilation failure, exaction failure, execution undetermined, execution success.

Datasets: A set of 40 predefined coding challenges of university level programming exercises were used across 4 fields: algorithms, games, data science and security.

Evaluation metrics: Execution success rate, Time performance, Code length, Consistency.

Results: ChatGPT exhibits nondeterministic behavior in code generating. Programming language highly chosen to solve tasks highly impact the understanding of the tasks by ChatGPT. ChatGPT struggles with low level languages compared to high level. ChatGPT is ethically inconsistent refusing to generate security related code in some languages while allowing it on others.

Reflection: Contributions: Sets an example of evaluating ChatGPT coding across 10 different coding languages, Highlights the strengths and weaknesses of LLM and raises concerns regarding security related code generating. Limitations: The study doesn't mention method naming and commenting, focuses only on ChatGPT3.5, a limited types of coding challenges were presented. Potential for future research:

More diverse LLM can be experimented on, more types of coding challenges can be experimented.

5-Extending the Frontier of ChatGPT: Code Generation and Debugging

Idea: This research examines ChatGPT 4 ability to solve programming challenges, focusing on both code generation and debugging skills. It evaluates its effectiveness in solving Leetcode problems by considering factors such as problem difficulty, execution efficiency, and memory consumption.

Methodology: 128 problems were chosen by the authors from Leetcode covering: Array & String, DP, Divide & Conquer, Graph, Greedy, Hash Table, Math, Sorting, Tree, Two Pointers. Each problem is given to ChatGPT, if the solution failed, feedback is given to ChatGPT asking for debugging and improving code, success rate is recorded.

Datasets: 128 Problems from Leetcode were used to balance problem types, difficulty and acceptance rate across all categories in different difficulties (hard, medium, easy) 15 problem for each category with 2 problems having acceptance rate less than 30% 1 problem between 40% and 70% and 2 problems more than 70% .

Evaluation metrics: Overall performance of success rate. feedback and debugging of how code is fixed after giving feedback to ChatGPT solution is incorrect. Analyzing success across different problem types. Analyzing success across different difficulty levels and against acceptance rates. Runtime and memory efficiency.

Results: Over 128 problems, 71.875% were solved correctly, performing best in tree problems and worst in greedy. Out of the 128 problems 8 were fixed and 36 failed to solve. 90% success rate on easy problems and 55% on hard problems hence easy problems have well-structured problem-solving methods.

Reflection: Contributions: This study highlights ChatGPT weakness in debugging and dealing with more difficult coding challenges and evaluates runtime and memory efficiency of the code provided. Limitations: Sample size is relatively limited, only LLM used is ChatGPT 4. Potential for future research: More problems with bigger sample size can be further tested on. More LLM can be tested on.

1- **TIAN, CHEUNG, KLEIN**, **WEIQI LU, TSZ-ON LI, TANG, BISSYANDÉ**. arXiv:2304.11938v2 [cs.SE] 31 Aug 2023 , Is ChatGPT the Ultimate Programming Assistant- How far is it?

2- **LIU, LE-CONG, WIDYASARI, TANTITHAMTHAVORN, LI ,D. LE,LO.** arXiv:2307.12596v2 [cs.SE] 15 Dec 2023, Refining ChatGPT-Generated Code: Characterizing and Mitigating Code Quality Issues.

3- **Jiawei Liu, Chunqiu StevenXia , Yuyao Wang, Lingming Zhang.** arXiv:2305.01210v3 [cs.SE] 30 Oct 2023, Is Your Code Generated by ChatGPT Really Correct? Rigorous Evaluation of Large Language Models for Code Generation.

4- **Buscemi, A.** (2023). A comparative study of code generation using chatgpt 3.5 across 10 programming languages. arXiv preprint arXiv:2308.04477.

5- **Sakib, F. A., Khan, S. H., & Karim,** A. R. (2024, July). Extending the frontier of chatgpt: Code generation and debugging. In 2024 International Conference on Electrical, Computer and Energy Technologies (ICECET (pp. 1-6). IEEE.