

An Evaluation of the Effectiveness of OpenAI's ChatGPT-3.5 for Automated Python Program Bug Fixing using QuixBugs

Marchel Christoper Wuisang
Computer Science Department
School of Computer Science
Bina Nusantara University,
Jakarta, Indonesia 11480
marchel.wuisang@binus.ac.id

Marcel Kurniawan
Computer Science Department
School of Computer Science
Bina Nusantara University,
Jakarta, Indonesia 11480
marcel.kurniawan001@binus.ac.id

Komang Andika Wira Santosa
Computer Science Department
School of Computer Science
Bina Nusantara University,
Jakarta, Indonesia 11480
komang.santosa@binus.ac.id

Abstract—In recent years, the use of Artificial Intelligence (AI) has become increasingly common in various fields, including in software development. One such field is where AI can automatically detect and fix bugs in code. GPT-3.5 is a state-of-the-art language model developed by OpenAI that has been trained on a massive amount of text data to generate natural language responses to a wide range of prompts. One of the main challenges in software development is bug fixing, which can be a time-consuming and complicated process. QuixBugs is a framework for evaluating automatic program repair techniques, which can be used to test the effectiveness of GPT-3.5 and similar bug-fixing tools. This paper evaluates the effectiveness of GPT-3.5 in automatically fixing bugs in Python code using QuixBugs. Through testing with 40 different Python bugs, We wanted to evaluate how well GPT-3.5 was able to accurately fix bug cases.

Keywords—ARP, Automatic program repair, Bug fixing, Python, GPT-3.5, ChatGPT, QuixBug

I. INTRODUCTION

Software development is a complex and constantly evolving field, with developers facing various challenges, including identifying and fixing bugs in code. Bugs in software that are not fixed can result in the collapse of important systems, which may have a high financial cost impact. To make it easier and support programmers in finding and fixing software errors, many automatic program improvement (APR) systems have been launched that automatically suggest software upgrades to fix detected errors [3].

Various approaches to automated program repair have been proposed, including generate-and-validate approaches that mutate software guided by a search strategy, and semantics-driven (or synthesis-based) approaches that use a constraint solver to synthesize repairs [5]. However, one of the key disadvantages of standard approaches to APR is their running cost. These approaches typically rely on test suites to verify program correctness or calls to a constraint solver, which can be time-consuming, making it difficult for programmers to efficiently detect and fix bugs in software.

Recently, program repair tools based on deep learning (DL) approaches have been introduced. These tools learn bug fixing patterns from existing databases and treat the automated program repair problem as a neural machine translation task, producing a ranking of patches. Unlike standard approaches, generated patches from DL-based program repair tools are not usually evaluated against a test suite or other automated verification strategy, so they may not

even compile. Nevertheless, DL-based program repair has shown competitive results to standard approaches [3].

Several large-scale language models based on the Transformer architecture have been introduced in recent years, such as CodeBERT, PyMT5, and GPT, which can process and extend source code and achieve comparable results to standard approaches on various coding tasks [2]. However, the quality of these suggestions is still unclear. In this work, we aim to evaluate and analyze the automatic bug fixing performance of OpenAI's GPT-3.5 for Python programs. We chose the QuixBugs benchmark set for our study, as it contains small yet challenging programs for current APR approaches. We will compare GPT-3.5's performance with that of ChatGPT and dedicated APR approaches. For the standard APR approaches, we will take the results from a recent paper that examines the performance of several methods on the QuixBugs benchmark set [3].

In this research paper, we evaluate the effectiveness of OpenAI's GPT-3.5 for automated Python program bug fixing using QuixBugs, a benchmark suite for automated program repair. We examine the accuracy and efficiency of GPT-3.5 in identifying and repairing bugs in Python programs, and compare its performance with another automated program repair tools.

The results of this study will contribute to the evaluation and analysis of automated program repair tools and their potential to improve software quality and reduce the economic costs associated with software bugs. Moreover, the study will provide insights into the strengths and limitations of OpenAI's GPT-3.5 as an automated program repair tool and its potential for future development.

II. LITERATURE REVIEW

There has been a recent surge of interest in using large language models for automated program repair, as demonstrated by OpenAI's Codex and ChatGPT models. Codex has been found to outperform most students on real questions taken from introductory programming exams, and has also demonstrated its capable to generate code from English prompts and do some programming tasks [1, 30]. Meanwhile, ChatGPT, although not specifically designed for programming tasks, has been found to be competitive with other state-of-the-art approaches in bug-fixing performance on the QuixBugs benchmark set [5]. Other experiments have also been conducted on the use of large language models in automated program repair [9, 10].

Large language models have had an enormous impact on various problems of natural language processing. One of the most well-known models is ChatGPT, which generates text in response to cues [22],[29]. ChatGPT is an NLP model launched in November 2022 that is capable of generating text

in a human-like conversational style [13, 15]. It has shown potential in various fields, including programming support, education, healthcare, finance, mathematics, and scientific research [8, 15, 16]. In the realm of programming, ChatGPT has been evaluated for program repair and found to outperform other models in terms of accuracy rate and repair time [13]. Automated program repair remains a challenging task for developers, who must understand the problem and localize its root cause in the source code [4]. The goal of automated program repair is to localize or discover problems, detect or rectify errors, and automatically apply patches to software bugs so that software faults can be fixed to increase software reliability [11, 12, 19, 20].

Automatic program repair introduced using different algorithm, genetic programming [25],[26],[28] and machine learning [21],[23],[24],[27] are the most general algorithm that used in ARP. As part of the effort to improve automated program repair, researchers have investigated different approaches to localizing bugs. One such approach is the use of fault localization techniques, which aim to identify the source of the bug by analyzing the program's execution behavior [2]. Other approaches include the use of static analysis, dynamic analysis, and program slicing [3, 7, 18]. These approaches can help developers identify the parts of the code that are likely to contain the bug, and can thus speed up the repair process.

In addition to localization, researchers have also explored different methods for finding bugs. One common approach is to use automated testing, which involves running tests on the code to detect errors [6]. Other methods include code review, debugging, and monitoring [14, 19, 21]. Automated program repair can benefit from the use of these techniques in identifying bugs and generating fixes.

Despite the promising results of automated program repair using large language models, there are still challenges that need to be addressed. One such challenge is the need to ensure that the generated fixes are correct and do not introduce new bugs [17]. Another challenge is the need to improve the performance of automated program repair systems, particularly in terms of their speed and scalability [22]. Addressing these challenges will be crucial in realizing the full potential of automated program repair using large language models.

- [1] Prather, J., Luxton-Reilly, A., Becker, B. A., Denny, P., & Finnie-Ansley, J. (2022). The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. *ACM Transactions on Computing Education (TOCE)*, 22(1), 1-28, doi: 10.1145/3511861.3511863.
- [2] Prenner, N., Luhana, H., & Kapfhammer, G. M. (2021). Automatic program repair with OpenAI's Codex: Evaluating QuixBugs. *arXiv preprint*, doi: 10.48550/arXiv.2111.03922
- [3] Chen, M., Zho, Y., Liu, X., Liu, D., & Song, L. (2021). Evaluating large language models trained on code. *arXiv preprint*, doi: 10.48550/arXiv.2107.03374
- [4] Surameery, R., & Shakor, A. H. (2023). Use chat GPT to solve programming bugs. In *Proceedings of the 12th International Conference on Computer and Automation Engineering (ICCAE '23)*, doi: 10.55529/ijitc.31.17.22.
- [5] Sobania, N., Moser, T., & Fraser, G. (2023). An analysis of the automatic bug fixing performance of ChatGPT. In *Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications (SPLASH '23)*, doi: 10.48550/arXiv.2301.08653.
- [6] Derrick Lin, James Koppel, Angela Chen, Armando Solar-Lezama. "QuixBugs: A Multi-Lingual Program Repair Benchmark Set Based on the Quixey Challenge." *Proceedings of the 38th International Conference on Software Engineering*, 2017, pp. 60-63, doi: g/10.1145/3505247.
- [7] He Ye, Matias Martinez, Thomas Durieux, Martin Monperrus. "A comprehensive study of automatic program repair on the QuixBugs benchmark." *Empirical Software Engineering*, vol. 26, no. 3, 2021, pp. 1-47, doi: 10.1016/j.jss.2019.01.069.
- [8] Fan, Z., Gao, X., Mirchev, M., Roychoudhury, A., & Tan, S. H. (2023). Automated repair of programs from large language models. In *Proceedings of the 2023 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications (SPLASH '23)*, doi: 10.48550/arXiv.2205.10583.
- [9] Z. Feng, D. Guo, D. Tang, N. Duan, X. Feng, M. Gong, L. Shou, B. Qin, T. Liu, D. Jiang et al., "CodeBERT: A pre-trained model for programming and natural languages," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing: Findings*, 2020, pp. 1536–1547, doi: 10.18653/v1/2020.findings-emnlp.139
- [10] C. Clement, D. Drain, J. Timcheck, A. Svyatkovskiy, and N. Sundaresan, "PyMT5: Multi-mode translation of natural language and Python code with transformers," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 2020, pp. 9052– 9065.
- [11] J. A. Prenner, H. Babii, and R. Robbes, "Can openai's codex fix bugs?," *Proceedings of the Third International Workshop on Automated Program Repair*, 2022, doi: 10.1145/3524459.3527351.
- [12] H. Ye, M. Martinez, T. Durieux, and M. Monperrus, "A comprehensive study of automatic program repair on the QuixBugs benchmark," 2019 *IEEE 1st International Workshop on Intelligent Bug Fixing (IBF)*, 2019, doi: 10.1016/j.jss.2020.110825.
- [13] Tian, Haoye, dkk. Is ChatGPT the Ultimate Programming Assistant -- How far is it? 2023. doi.org (Datacite), <https://doi.org/10.48550/ARXIV.2304.11938>.
- [14] Le Goues, Claire, dkk. "Automatic Program Repair." *IEEE Software*, vol. 38, no. 4, Juli 2021, hlm. 22–27. doi.org (Crossref), <https://doi.org/10.1109/MS.2021.3072577>.
- [15] M. M. Rahman and Y. Watanobe, "Chatgpt for Education and research: Opportunities, threats, and strategies," 2023, doi: 10.20944/preprints202303.0473.v1.
- [16] N. Oh, G.-S. Choi, and W. Y. Lee, "Chatgpt goes to operating room: Evaluating GPT-4 performance and its potential in surgical education and training in the era of large language models," 2023, doi: 10.1101/2023.03.16.23287340.
- [17] H. Cao, Y. X. Meng, J. Shi, L. Li, T. Liao, and C. Zhao, "A survey on automatic bug fixing," 2020 *6th International Symposium on System and Software Reliability (ISSSR)*, 2020, doi: 10.1109/ISSSR51244.2020.00029.
- [18] L. Gazzola, D. Micucci, and L. Mariani, "Automatic Software Repair: A survey," *IEEE Transactions on Software Engineering*, vol. 45, no. 1, pp. 34–67, 2019, doi: 10.1109/TSE.2017.2755013.
- [19] Zhang, Quanjun, dkk. A Survey of Learning-based Automated Program Repair. 2023. doi.org (Datacite), <https://doi.org/10.48550/ARXIV.2301.03270>.
- [20] N. Jiang, T. Lutellier, and L. Tan, "Cure: Code-aware neural machine translation for automatic program repair," 2021 *IEEE/ACM 43rd International Conference on Software Engineering (ICSE)*, 2021, doi: 10.1109/ICSE43902.2021.00107
- [21] Zhang, Jialu, dkk. Repairing Bugs in Python Assignments Using Large Language Models. 2022. doi.org (Datacite), doi: 10.48550/ARXIV.2209.14876
- [22] C. K. Lo, "What Is the Impact of ChatGPT on Education? A Rapid Review of the Literature," *Education Sciences*, vol. 13, no. 4, p. 410, Apr. 2023, doi: <https://doi.org/10.3390/educsci13040410>.
- [23] D. Drain, C. I. Clement, G. Serrato, and N. Sundaresan, "DeepDebug: Fixing Python Bugs Using Stack Traces, Backtranslation, and Code Skeletons," *arXiv (Cornell University)*, May 2021, doi: <https://doi.org/10.48550/arxiv.2105.09352>.
- [24] N. Nguyen and S. Nadi, "An empirical evaluation of GitHub copilot's code suggestions," *Proceedings of the 19th International Conference on Mining Software Repositories*, May 2022, doi: <https://doi.org/10.1145/3524842.3528470>.