International Journal of Science and Research Archive

eISSN: 2582-8185

Cross Ref DOI: 10.30574/ijsra

Journal homepage: https://ijsra.net/

(Review Article)

A comparative study of AI code bots: Efficiency, features, and use cases

Mihir Mehta *

Software Development Manager, Chewy.

International Journal of Science and Research Archive, 2024, 13(01), 595–602

Publication history: Received on 05 August 2024; revised on 12 September 2024; accepted on 14 September 2024

Article DOI: https://doi.org/10.30574/ijsra.2024.13.1.1718

Abstract

This article thus provides a comparative study of AI code bots, in terms of their performances, characteristics and applications. It looks at how via application of artificial learning and NLP for software development, these bots help in coding, debugging, and optimizing. The work also analyses AI code bots including GitHub Copilot, Tabnine, Replit Ghostwriter, Amazon CodeWhisperer, and others as well as strengths, weaknesses, and versatility of such software depending on the programming language and the development environment used. Although these tools work well in improving productivity, reducing on-task redundancies and forming a more accurate code, they do not contain self-authorization, especially for intricate occasions. The study shows that the employment of AI code bots has advantages and drawbacks that should be investigated further in the future emphasizing on the possibilities of enhancing the filters on errors, the degree of personalization and considering the ethical aspect of employing AI code generation.

Keywords: AI Code; Bots; Debugging; GitHub Copilot; Tabnine; Amazon CodeWhisperer; Chatgpt; Programming Languages

1. Introduction

Artificial Intelligence (AI) code bots are one of the most complex application types that are available in the market. Code bots are designed to assist engineers in writing, debugging and or optimizing line of codes. These bots use machine learning approaches, preferably NLP or natural language processing and analysis to get programming languages and hasten coding solutions. Research currently ongoing with regard to AI code bots relates to the extent they help to shorten the time taken in their development. It discusses how the completions happen on an automatic basis, errors are identified, they interface with many IDEs, and their application in diverse disciplines. AI-powered code generators are becoming important in the design of software. These generators simplify code development for many tasks. These tools, which are backed by advanced language representations like GPT-4 as well as Google Bard, facilitate development by interpreting natural language stimulates and generating executable source code in many programming languages. Their usefulness lies in their capacity to quickly build working code fragments, reduce development time, and repair bugs [10]. It can integrate with development environments, autocomplete code, and provide security-focused outputs. Production of boilerplate code, maintenance of complex algorithms, testing, and explanation are some conceivable use cases. AI-powered code generators may speed up prototype development, boost worker efficiency, and eliminate repetitious coding.

This comparative research focuses on how these bots improves efficiency, minimize errors and advance multiple programming languages. Besides, they assist developers in almost every single situation of development.

∗ Corresponding author: Mihir Mehta.

2. AI Code Generators Overview

AI code generators simplify the development of software using powerful machine learning as well as deep learning algorithms. The ultimate goal in this type of development is to eliminate the human factor. These technologies put a cut on human coding as it involves converting written or spoken words or graphics or high level description into working software [1]. AI-driven code makes frontend elements, mobile applications, and industrial automation applications easy to build. Among them are transform designs, recurrent neural networks, in addition to natural language processing to help in the completion of this mission. Others are Amazon CodeWhisperer and GitHub Copilot that enhance coding assistance with aid of Artificial Intelligence. These bots remain smart and give suggestions on the basis of contexts; they complete tasks smoothly. Promising tools such as "GPT-3" and BERT produce human-inspired, evenly formatted, and structured, and standard-compliant code [2]. However, there are issues that still need to be solved to ensure that the AI generated code is correct, efficient, and as good code craftsmanship as possible. By incorporating the use of AI in coding, the work has been made faster and the quality of code has also been enhanced, and also the gap between a normal non-technical person and a software engineer has been reduced.

Bots, or AI-powered code generators, may produce code according to user inputs or natural language processing. This automates many programming tasks that could have been difficult. These technologies can read queries written in natural languages and convert them into executable code chunks that perform certain operations [11]. Developers may use them to generate redundant code, recommend changes, and ensure best practices.

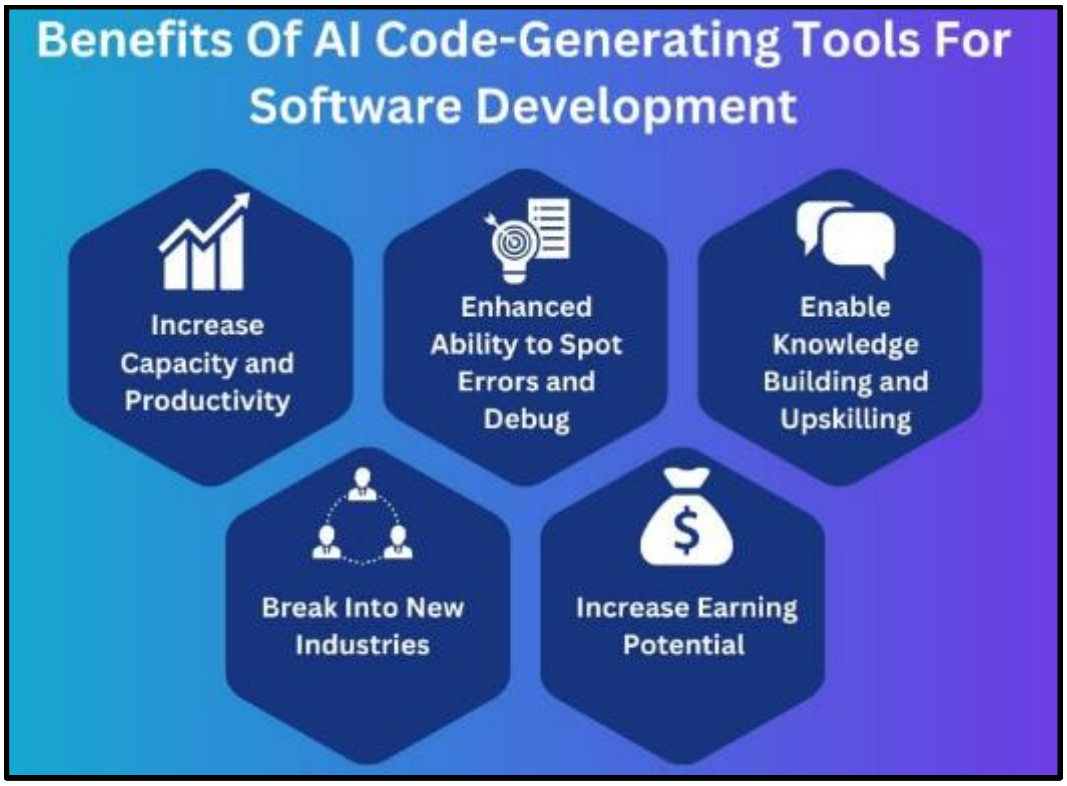
Benefits Of AI Code-Generating Tools For Software Development
Increase Capacity and Productivity
Enhanced Ability to Spot Errors and Debug
Enable Knowledge Building and Upskilling
Break Into New Industries
Increase Earning Potential

Figure 1 Advantages of AI Code Generators

The above Figure depicts the major benefits of applying AI-developed code bots in the creation of different kinds of software applications. AI code bots' ability to boost effectiveness and productivity is a major strength of these developers. Designers may focus on more complex and innovative projects since they minimise creating and fixing bug's time. Cognitive bots make coding easier for non-coders. This increases public involvement in programming and encourages innovation in many different industries. These bots may also eliminate human errors, resulting in more pleasant, less wasteful code [12].

AI code bots simplify prototype construction, language translation, software testing technology, and template code production. Their presence requires them to maintain large codebases, advice regarding performance improvements, and provide documentation. AI-powered bots improve coding learning more easily by offering immediate criticism and suggestions. However, AI-powered code bots might produce incorrect or wasteful code because they cannot understand confusing requests. They may perform poorly in high-risk situations or jobs which require specialised expertise due to their reliance on training data. Ethical and security considerations related to code production are also a concern. Bots may create prejudicial or vulnerable solutions without supervision.

3. Evaluation Criteria

The assessment criteria for AI code bot creation emphasise many important elements. Making sure the bot writes accurate and functioning code is crucial. As faster bots enhance user experience, processing speed and response time optimisation are crucial. As applied to the concept of flexibility, the bot can support many programming languages and fulfil many user requirements efficiently. Flexibility means a bot's ability to handle complex tasks or larger loads with no impact on the delivery of services. Durability is bringing assurance that no matter what the bot can take it. By reinforcement learning, the bot evolves thus making it efficient in the long run [3]. To ensure that the operating safety is at its best, code has to be protected from such chances. When it is a question of developing a bot's interface, usability should be considered as the top priority. This is because it facilitates the usage of the bot by everyone including programmers of different skills. The logic of choosing between two close options dictates that the process be kept simple so that the users can trust the bot code. These traits dictates the effectiveness of automated code bots for use in these real life situations.

In order to properly assess AI code bots, their code-writing, analysis, and security abilities must be assessed. When using the technique, various types of models are usually evaluated against established standards. This assessment will juxtapose the code with conventional code and look for security problems to determine its validity and efficacy. LLMs, or Large Language Models, are used to create smart code bots because they can understand and write human-like language. Large language models (LLMs) are trained by utilising massive datasets with a variety of code illustrations and programming concepts [13]. This training enables LLMs in identifying patterns as well as in the ability of providing the right piece of code. These models are used to fill and predict the code as per the input hints which are a part of the training method. Participants improve their capacity regarding the solving of different imperative coding problems.

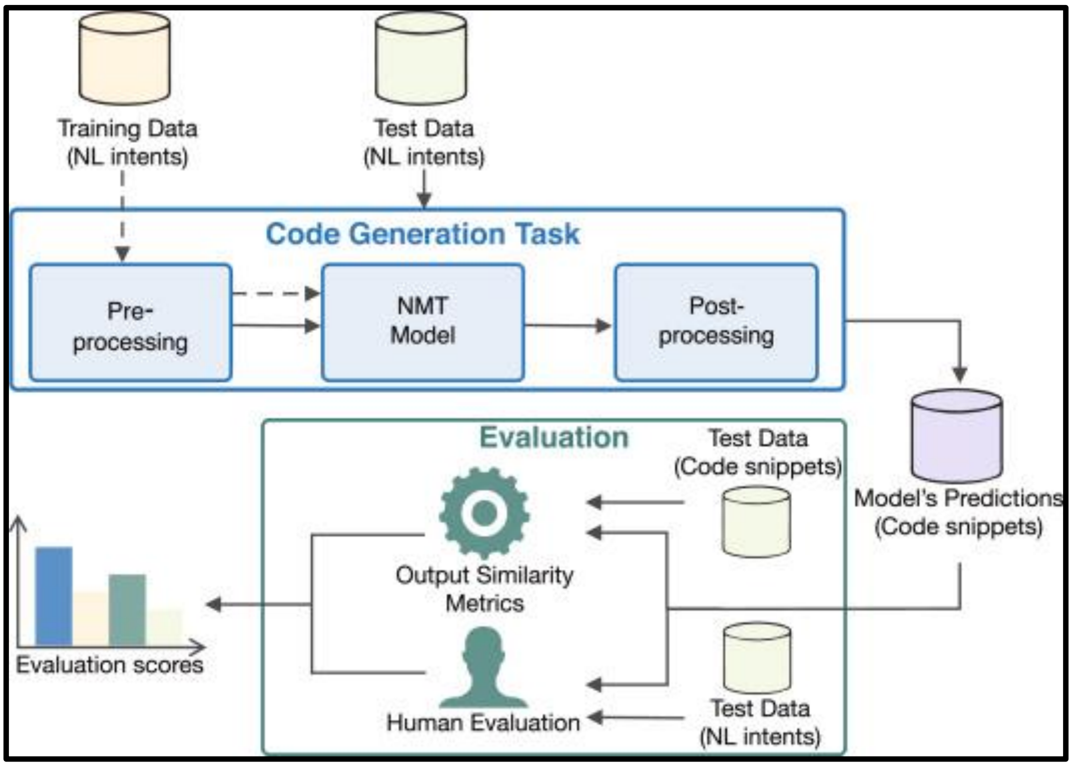

Figure 2 AI-powered Code Generators with NMT Model

When it comes to the AI code compilers which employ NMT models, the performance comparison is needed between the automated metrics and the human evaluations in order to check the accuracy of the translation and the optimisation of the code. Evaluations with the help of computer metrics can be applied to the estimate translation and code production and more often they do not coincide with people’s estimations. The framework's capacity to understand and evaluate difficult coding difficulties, write accurate and secure code, and adapt to a number of approaches to programming and platforms are all crucial [14]. Furthermore, the model's code creation performance in real-world scenarios must be examined to validate its practicality and durability.

4. Comparison and Analysis

4.1. GitHub Copilot

GitHub Copilot streamlines basic unit testing and saves time on repetitive tasks. It achieves this by rapidly writing boilerplate code. This program generates beneficial concepts from coding principles. However, it fails with complex company reasoning or customised circumstances. This could contribute to faulty or inappropriate code generation. Surveillance is vital to ensure that the individual is operating properly in their context. This is especially true in challenging or expert-required jobs. However, it has several drawbacks, such as only producing proper code 28.7% of the time while primarily generating accurate code 51.2% of the time [4]. System functionality plummets without essential docstrings or function names that reduce system reliability as well as precision.

GitHub Copilot helps developers work faster by automating processing code and executing construction. The capacity for flexibility helps programmers work more effectively. Additionally, it is interoperable with several programming languages, which comprises Python, JavaScript, and TypeScript, demonstrating its widespread usage in projects worldwide. It is the most used code generation tool in the business due to its 40-50% usage share. Users find GitHub Copilot more accessible since it integrates with popular IDEs comparable to Visual Studio Code along with JetBrains. GitHub Copilot lacks code localisation and explication, which other apps include. GitHub Copilot has this limitation. Individuals pay $10 each month and companies $100 per year [15]. This pricing arrangement may cause problems for certain consumers. This application is ideal for beginners due to its ease of use and minimal setup. Despite its lack of customising and handling of mistakes, its intuitive design and interoperability with a comprehensive integrated development system are its main benefits.

4.2. Tabnine

Tabnine is skilled in being appropriate with a broad spectrum of integrated development environments (IDEs), decreasing typing to speed up coding, and using an adaptive learning procedure to improve options depending on user preferences. However, the free version has fewer features than the premium version [5]. Erroneous advice can be annoying and disturb the programming cycle.

Tabnine offers code completion and context-sensitive suggestions, among other benefits. Its extensive language-specific frameworks enhance these benefits. Because it supports Python, JavaScript, Java, Go, and others, it has a 15-20% usage share. The application may be connected to over twenty editors, involving JetBrains IDEs and VS Code boosting product diversity. Tabnine's precise and quick execution of codes is a major benefit. All code suggestions, function creation, and operational fulfilment may be done concurrently. Basic and Pro plans are free and $49 per year, respectively [16]. Emerging computational code help professionals are drawn to the tool's ease of navigation and installation. Tabnine's minimal customisation possibilities and insufficient code structuring, unit testing and bug identification may turn certain individuals off. The free version has several limited features that could fail to satisfy the needs of more competent clients. Ideas may differ in intelligibility and relevance with respect to the programming language and circumstance.

4.3. Replit Ghostwriter

Replit GhostWriter, which produces code quickly, and simplifies feature presentation without extensive documentation. The chat function helps with restructuring and provides relevant thoughts. However, it struggles to produce pertinent code for particular applications, requiring constant human adjustments [6]. The lack of a distinguishing feature for assessing changes, the inability to argue coding choices and the lack of traditional testing may reduce development efficiency.

Replit Ghostwriter provides substantial advantages by expediting coding with functionalities such as autocomplete, code creation, and elucidations. It has the capacity to handle sixteen programming languages and interface effortlessly with Replit's IDE augments workflow. The program proficiently transforms and generates code, proving beneficial for both new and seasoned engineers. Nonetheless, it encounters obstacles, like sporadic repeating recommendations and a requirement for substantial programming expertise to implement some autocomplete recommendations [17]. The service's need for Cycles for accessibility may dissuade people who desire not to handle virtual tokens. Although Ghostwriter is a formidable tool, it does not yet serve as a comprehensive substitute for extensive coding proficiency.

4.4. Amazon CodeWhisperer

The highly skilled Amazon CodeWhisperer can provide code suggestions in several programming languages. It also seamlessly integrates with IDEs like IntelliJ. Amazon CodeWhisperer integrates with seamless IDEs and provides comprehensive security testing. Transitioning between activities takes less time, improving output. Additionally, it screens for security issues [7]. It could involve manual adjustments for importation as well as customisation file creation.

Amazon CodeWhisperer boosts developer productivity by delivering immediate time code recommendations and creating environment-specific code blocks. Enhanced Amazon Web Services code represents one of its perks. Other features incorporate code improvement, interoperability with many programming languages, and interaction with other IDEs [18]. It also offers security evaluations, solution suggestions, as well as open-source code reference tracking. Its drawbacks include making errors in code recommendations and requiring engineers to examine and validate ideas in detail. However, the free tier has certain perks, and the paid Professional version unlocks more features and customisation options.

4.5. Sourcegraph Cody

Sourcegraph Cody's expertise in context-aware support, code explanation, and software fault detection helps engineers handle complicated codebases. Delivering a complete code analysis and the latest changes boosts efficiency. The tool may require continual fine-tuning to properly support several coding environments and situations. The tool's effectiveness depends on the codebase it communicates with.

Sourcegraph Cody directly integrates AI support along with code searching into JetBrains IDEs to boost development productivity [21]. This application enables to employ of autocomplete, code elucidations, and situational enquiries across local and remote code sources. Additionally, contextual searches are present. Several parameters may speed up development, enhance code accuracy, and recover context faster. However, the dependency on LLMs, the likelihood of coding mistakes, and the constraints on handling complex queries or large codebases may cause complications.

4.6. OpenAI Codex

Several related materials lack the OpenAI Codex's unique attributes. This consists of powerful code creation skills, assistance with several languages, and sophisticated code translation and explanation capabilities. It seamlessly integrates many platforms, displaying its language conversion expertise. However, it has a difficult setup that needs technical skill, the prospect of rising accessibility to API prices, and inadequate data science help compared to rivals. Setting up is difficult and requires technical expertise [19]. As Codex contains a large feature set, beginners find it harder to learn. Its flexibility and key functionalities contribute to it being a desirable tool for professional software engineers.

4.7. Ponicode

Ponicode's focus on Python unit testing makes it stand out. This guarantees code consistency and dependability. Since it works with common Python-focused software development platforms (IDEs), it may help developers build strong, well-tested apps. Python programmers will appreciate finding the application simpler to operate since it has a basic graphic user interface and needs minimal settings. However, Ponicode only supports Python as it does not support any other programming languages. Due to its focus on unit testing, it does not give code suggestions or localisation [20]. Given that this is primarily utilised for unit testing. For individuals seeking more feature-rich code-generation tools, their value may be limited.

Concerning the discussion, “GitHub Copilot” proves to be the better AI tool as it is more widespread, supports multiple programming languages, and is perfectly compatible with famous IDEs during code generation.

5. Case Studies

Research on autonomous code compilers found that their performance varies by platform. GPT-3, GitHub Copilot, as well as Amazon CodeWhisperer, were examined while assessing 164 code problems. GPT-3 had the highest success rate, 93.3%. Following that, GitHub Copilot had a 91.5% performance along with Amazon CodeWhisperer at 90.2% [9]. Operation with inconsistent data types, syntax mistakes, and erroneous list indexing were common issues. Further tests showed that GPT-3's efficiency dropped as tasks and code got more difficult. Like GitHub Copilot and CodeWhisperer, CodeWhisperer has trouble solving code bugs. Further studies showed that GPT-3 had trouble debugging and that human participation was needed to improve the performance of the code [8]. Considering their many features, these applications have major limitations in the integrity of code and error handling.

Existing applications highlight considerable differences in usability, efficiency, and use between automated code bots like ChatGPT as well as Bard. ChatGPT can provide accurate replies, detailed explanations, and code testing, making it a good alternative for applications with requirements for thorough instruction and administration. In educational contexts, its ability to produce and examine code snippets helps learners who face programming challenges. However, Bard often produces erroneous results, making it unreliable for crucial coding tasks. Fairness of Recommendation through the Large Language Model (FaiRLLM) demonstrated that ChatGPT is more consistent than Bard [21]. Before deployment, bots must be tested, especially in accurate and transparent applications.

Software development is impacted by AI-based code generation approaches since they increase efficiency and allow for customised suggestions. By generating code blocks for services offered by AWS, which simplifies the creation of its setting, AWS CodeWhisperer may not offer enough diversity for basic coding tasks. GitHub Copilot is capable of connecting to GitHub despite any issues and supports several programming languages [22]. Despite its many applications, it must be rigorously inspected for security issues. Despite its natural language processing capabilities, Google AI's CodeAssist continues to be under development and is not yet available to individuals. SAP Build Code, a software delivery solution for ABAP code exclusively accessible in SAP, focuses on the SAP BTP platform. AWS CodeWhisperer can be beneficial for AWS initiatives, GitHub Copilot for several programming languages, and SAP Build Code for SAP tools [23]. Any tool's effectiveness depends on its growth context. Designers must assess whether these advancements are acceptable for their projects to maximise profitability and the integrity of the code.

6. Conclusion

Artificial intelligence-powered code bots automate code creation and bug fixes, thereby boosting efficiency. In this regard, GitHub Copilot, Tabnine, as well as GPT-3 are top performers. Monitoring mistakes, assuring accuracy, and adapting to difficult responsibilities remain challenges. Despite advances in the industry, human participation is still needed to verify the reliability of code and correct programming restrictions. Comparative examination of AI code bots shows significant gains in code development, operation, and applicability. These are among the development enhancing tools for which GitHub Copilot, Tabnine and Amazon CodeWhisperer are well known. These methods are also known to have constraints sometimes including the fact that they might not function well in some instances especially if the organization's environment is quite complicated. Despite the multiple capabilities they have, Replit Ghostwriter and Sourcegraph Cody need the angles of humans to give out the best outcome. Hence, it can be seen that LLMs or models such as GPT-3 and other LLMs have a higher code finalisation rate given the fact that it is not very easy debugging intricate problems at times. The study shows that with the help of AI code bots, the work becomes more effective and less error-prone. To address the boundary and ethical challenges in the case of AI-generated code, strict oversight, error-management designs should be financed. The findings and opinions presented in this article are solely those of author and do not represent the views of Author's employer.

Future Work

Future research should therefore aim to increase the accuracy and reliability of the technology-driven code bots through the eradicating of error mitigation and customisations. Developers need to develop better models to enhance code suggestions and to debug the error. There is also a problem of poor integration with other development frameworks and integration for more languages. The subsequent investigation should also look into minimising the humans' interaction and at the same time increasing the efficiency of the AI based code generation. Greater research effort should be dedicated towards developing principles for honesty and accountability of the artificial intelligence systems within the understanding of the ethical ramifications of the artificial intelligence programming. Doing this helps to preclude the risk of developing unsecure and coded discriminations and also considers in equalities of code deployments. In the opportunities and threats of applying artificial intelligence techniques in software engineering, it is essential to find out how these advancements help to complement rather than offset employment of people [24]. This study should also focus on how the relations between the robotics and developers could be fostered to enhance creativity as well as productivity. Furthermore, training domain-specific data to help AI systems might significantly enhance the systems' performance and applicability. With specialised data, AI models can be trained in such a way that they will be able to suggest accurate code and be able to solve company's problems in a very efficient manner. This target will create a focused approach towards enhancing the performance of AI solutions and guarantee that they meet the requirements of various fields in software development. More research and change are required to integrate novelty with ethical issues and benefits concerning the workforce.

Compliance with ethical standards

Disclosure of conflict of interest

No conflict of interest to be disclosed.

References

[1] A. Odeh, N. Odeh, and A. S. Mohammed, "A Comparative Review of AI Techniques for Automated Code Generation in Software Development: Advancements, Challenges, and Future Directions," *TEM Journal*, pp. 726–739, 2024, doi: https://doi.org/10.18421/tem131-76.

[2] L. Hamilton, D. S. Elliott, A. J. Quick, S. Smith, and V. Choplin, "Exploring the Use of AI in Qualitative Analysis: A Comparative Study of Guaranteed Income Data," *International journal of qualitative methods*, vol. 22, pp. 1609–4069, 2023, doi: https://doi.org/10.1177/16094069231201504.

[3] A. Iorliam and J. A. Ingio, "A Comparative Analysis of Generative Artificial Intelligence Tools for Natural Language Processing," *Journal of Computing Theories and Applications*, vol. 2, no. 1, pp. 91–105, 2024, doi: https://doi.org/10.62411/jcta.9447.

[4] B. Yetistiren, I. Ozsoy, and E. Tuzun, "Assessing the quality of GitHub copilot's code generation," *Proceedings of the 18th International Conference on Predictive Models and Data Analytics in Software Engineering*, pp. 62–71, 2022, doi: https://doi.org/10.1145/3558489.3559072.

[5] R. Rajkumar, "Tabnine and Atlassian reveal new generative AI tools for developers," *ZDNET*, 2024. https://www.zdnet.com/article/tabnine-and-atlassian-reveal-new-generative-ai-tools-for-developers/ (accessed 2024).

[6] J. Schomay, "Observations on using Replit's AI coding tool from an experienced coder," *Medium*, 2023. https://medium.com/@jschomay/observations-on-using-replits-ai-coding-tool-from-an-experienced-coder-3dd7d0146401 (accessed 2024).

[7] Mani, "Amazon CodeWhisperer —stepping on the accelerator ;-)," *Medium*, 2023. https://cmani.medium.com/amazon-codewhisperer-stepping-on-the-accelerator-1d0883ed649e (accessed 2024).

[8] S. Charles, "Impact of AI Assistants: Sourcegraph's Cody and the Future of Coding," *Medium*, 2023. https://medium.com/@ShawnBasquiat/impact-of-ai-assistants-sourcegraphs-cody-and-the-future-of-coding-450a99185c22 (accessed 2024).

[9] D. Tosi, "Studying the Quality of Source Code Generated by Different AI Generative Engines: An Empirical Evaluation," *Future Internet*, vol. 16, no. 6, p. 188, 2024, doi: https://doi.org/10.3390/fi16060188.

[10] Buscemi, A. (2023). *A Comparative Study of Code Generation using ChatGPT 3.5 across 10 Programming Languages*. [online] *Arxiv*, ArXiv.org, pp.1–12. Available at: https://arxiv.org/pdf/2308.04477 [Accessed 2024].

[11] Fucci, D., Romano, S., Baldassarre, M., Caivano, D., Scanniello, G., Thuran, B. and Juristo, N. (2022). A Longitudinal Cohort Study on the Retainment of Test-Driven Development. *Ocassionally Secure: A Comparative Analysis of Code Generation Assistants*, [online] pp.1–16. doi:https://doi.org/10.1145/nnnnnnn.nnnnnnn.

[12] Hultberg, P.T., Calonge, D.S., Kamalov, F. and Smail, L. (2024). Comparing and assessing four AI chatbots' competence in economics. *PloS one*, [online] 19(5), p.e0297804. doi:https://doi.org/10.1371/journal.pone.0297804.

[13] Krishnan, P. (2024). *Gen AI Code Generation Tools: A Comparative Analysis with Quality Evaluation*. [online] Medium. Available at: https://medium.com/@praveenct/gen-ai-code-generation-tools-a-comparative-analysis-with-quality-evaluation-4324ef756def [Accessed 2024].

[14] Le, K.T. and Andrzejak, A. (2024). Rethinking AI code generation: a one-shot correction approach based on user feedback. *Automated software engineering*, [online] 31(2), pp.1–42. doi:https://doi.org/10.1007/s10515-024-00451-y.

[15] Nguyen, N. and Nadi, S. (2022). An empirical evaluation of GitHub copilot's code suggestions. *Proceedings of the 19th International Conference on Mining Software Repositories*, [online] pp.1–10. doi:https://doi.org/10.1145/3524842.3528470.

[16] Glen, S. (2022). *Tabnine code completion platform adds more powerful AI*. [online] Techtarget. Available at: https://www.techtarget.com/searchsoftwarequality/news/252521540/Tabnine-code-completion-platform-adds-more-powerful-AI [Accessed 2024].

[17] Das, S. and Batterywala, H. (2024). Can AI Coding Assistants Enhance Secure Programming Education? A Comparative Analysis of AI-Based Programming Assistants for Code Optimization. *ACM Conference on International Computing Education Research V.2*, [online] 24(2), pp.527–528. doi:https://doi.org/10.1145/3632621.3671421.

[18] Yetiştiren, B., Özsoy, I., Ayerdem, M. and Tüzün, E. (2023). Evaluating the Code Quality of AI-Assisted Code Generation Tools: An Empirical Study on GitHub Copilot, Amazon CodeWhisperer, and ChatGPT. *arXiv:2304.10778 [cs]*, [online] pp.1–10. doi:https://doi.org/10.48550/arXiv.2304.10778.

[19] Finnie-Ansley, J., Denny, P., Becker, B.A., Luxton-Reilly, A. and Prather, J. (2022). The Robots Are Coming: Exploring the Implications of OpenAI Codex on Introductory Programming. *Australasian Computing Education Conference*, [online] pp.1–10. doi:https://doi.org/10.1145/3511861.3511863.

[20] Silvan (2020). *PoniCode : My feedback and a mixed overall feeling about the tool.* [online] Medium. Available at: https://sylvainleroy.medium.com/ponicode-my-feedback-and-a-mixed-overall-feeling-about-the-tool-13a04a5470a0 [Accessed 2024].

[21] Liguori, P., Improta, C., Natella, R., Cukic, B. and Cotroneo, D. (2023). Who evaluates the evaluators? On automatic metrics for assessing AI-based offensive code generators. *Expert Systems with Applications*, [online] 225, p.120073. doi:https://doi.org/10.1016/j.eswa.2023.120073.

[22] Ahmed, I., Kajol, M., Hasan, U., Datta, P.P., Roy, A. and Reza, Md.R. (2024). ChatGPT versus Bard: A comparative study. *Engineering reports*, [online] p.e12890. doi:https://doi.org/10.1002/eng2.12890.

[23] Negri-Ribalta, C., Stewart, R.G., Sergeeva, A. and Lenzini, G. (2024). A systematic literature review on the impact of AI models on the security of code generation. *Frontiers in Big Data*, [online] 7, pp.1–10. doi:https://doi.org/10.3389/fdata.2024.1386720.

[24] Coello, C.E.A., Alimam, M.N. and Kouatly, R. (2024). Effectiveness of ChatGPT in Coding: A Comparative Analysis of Popular Large Language Models. *Digital*, [online] 4(1), pp.114–125. doi:https://doi.org/10.3390/digital4010005.