SN Computer Science (2025) 6:673
https://doi.org/10.1007/s42979-025-04241-5

REVIEW ARTICLE

Usage of Large Language Model for Code Generation Tasks: A Review

Stefano Bistarelli[1] · Marco Fiore[2] · Ivan Mercanti[1] · Marina Mongiello[2]

Received: 1 February 2024 / Accepted: 17 July 2025

Abstract
Large Language Models have received a lot of attention in recent years due to their outstanding performance on various Natural Language Processing tasks. They can be used for lots of applications, including assistance in code generation tasks. Actual literature lacks an exhaustive analysis of the benefits and drawbacks of using a Large Language Model for the generation of simple and complex code. This paper aims to overcome the issue: we perform a Literature Review to explore the state-of-the-art of the proposed topic, answering 4 Research Questions. Using the PRISMA methodology, we reviewed 66 papers published between 2021 and 2023. Our analysis reveals Python's dominance as the preferred language and identifies a significant research gap in addressing ethical constraints. Additionally, we provide insights into the performance of models such as GPT-4 and CodeLlama, and their comparative utility in tasks ranging from debugging to multi-turn program synthesis. The findings offer a foundation for future research aimed at optimizing LLMs for code generation.

Keywords Large language models · Generative AI · Code generation · Review

Introduction

Generative AI is a type of artificial intelligence that includes various techniques and models used to generate data or content. These models can create new content from scratch, often without the need for direct human input or based on specific prompts. LLMs, on the other hand, are a type of Generative AI that uses Neural Networks trained on vast amounts of textual data from different sources, such as online blogs, code repositories, open data, and media content. Their training aims to enable them to understand and produce human language. In November of 2022, a sophisticated chatbot called ChatGPT[1] was launched. Developed by OpenAI,[2] the chatbot utilizes advanced deep learning algorithms to generate human-like responses to natural language inputs. The model has been trained on a vast and diverse dataset of text and is capable of comprehending and generating text on various topics. ChatGPT's versatile capabilities make it a suitable solution for several applications, including customer service, content creation, and language translation. Since its release, ChatGPT's popularity has grown exponentially, surpassing even the likes of TikTok, the current record-holder.[3] Within two months of its launch, ChatGPT boasts an impressive user base of 100 million. ChatGPT's success moves the discussion on the LLM and their ability.

In this paper we would like to review all existing work about LLMs used to generate code and programs. We will review using the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) methodology [55]. It involves several steps: defining research questions, filtering through relevant information using inclusion and exclusion criteria, and presenting our findings visually and through written discussion.

The rest of the paper is structured as follows. Section "Background" overviews the existing LLMs. Section "Related Work" discusses the most important works on

✉ Marco Fiore
marco.fiore@poliba.it

Stefano Bistarelli
stefano.bistarelli@unipg.it

Ivan Mercanti
ivan.mercanti@unipg.it

Marina Mongiello
marina.mongiello@poliba.it

[1] Department of Mathematics and Computer Science, University of Perugia, Perugia, Italy

[2] Department of Electrical and Information Engineering, Polytechnic University of Bari, Bari, Italy

[1] https://chat.openai.com.

[2] https://openai.com.

[3] https://www.reuters.com/technology/chatgpt-sets-record-fastest-growing-user-base-analyst-note-2023-02-01/.

Published online: 23 July 2025

LLMs. Section "Research Methodology - Literature Review" discusses our research methodology to perform the review. Then Sect. "Results and Discussion" provides a discussion of our review divided by research questions and analyzes some threats to the validity of the work. Finally, Sect. "Conclusion and Future Directions" draws the conclusions.

Background

Generative AI is a broad category of artificial intelligence that encompasses various techniques and models designed to generate data or content [32]. These models are capable of creating new content from scratch, often without direct human input or based on specific prompts. LLMs are a subset of Generative AI: they are Neural Networks trained on vast amounts of textual data, gathered from different sources, from online blogs to code repositories, from open data to media content. Their training aims to let them understand and generate human language. These models have demonstrated remarkable capabilities in a wide range of natural language processing tasks, including text generation [1], translation [23], summarization [38], and more. They can be considered as an evolution of chatbots [2], introduced in 1966 with ELIZA [69], a computer-based psychotherapist answering user's questions.

A survey conducted by [24] shows the evolution of LLMs, starting from GPT-1 in 2018. The diffusion of different LLMs starts at the end of 2022, with different producers showing their products. Figure 1 shows a timeline of the distribution of LLMs from 2021 to 2023, together with a division between open source and closed source models, as shown in [30].

Their interest grew exponentially after the born of ChatGPT on November 2022, the first application of LLMs open to public domain. A research on Google Trends confirms this trend, as shown in Fig. 2.

ChatGPT is based on the Generative Pre-trained Transformer (GPT) architecture, where a model is trained with high amounts of data and can be fine-tuned for different application. Different kinds of Large Language Models exist, such as Bidirectional Encoder Representations from Transformers (BERT), developed by Google. This approach is suitable for tasks requiring an understanding of context. XLNet is another LLM that improves the Transformer-XL architecture, incorporating a permutation-based training approach [65].

Interaction with LLMs

The interaction with a LLM can happen in different ways, from a chatbot-based style to the usage of Application Programming Interfaces (APIs). They can also be used in online platforms or run locally in the user computer. ChatGPT Plus [33] is a subscription-based chatbot that supports OpenAI's GPT-4 model. It offers both a chatbot to talk to and APIs to develop a custom platform. It also allows the user to upload and work on multimedia files instead of simple text. Local implementations such as GPT4All [5], LocalAI [16], and Ollama [31] require no GPU or active internet connection. They can be used both as a chatbot to interact with or an endpoint for API calls. They support a variety of models,

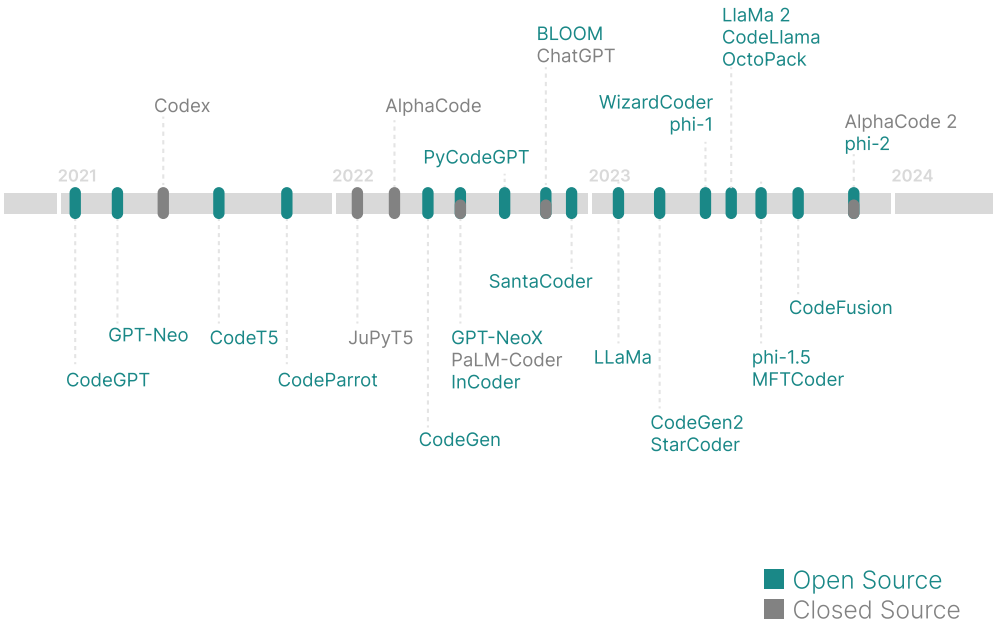

Fig. 1 LLMs evolution until 2023

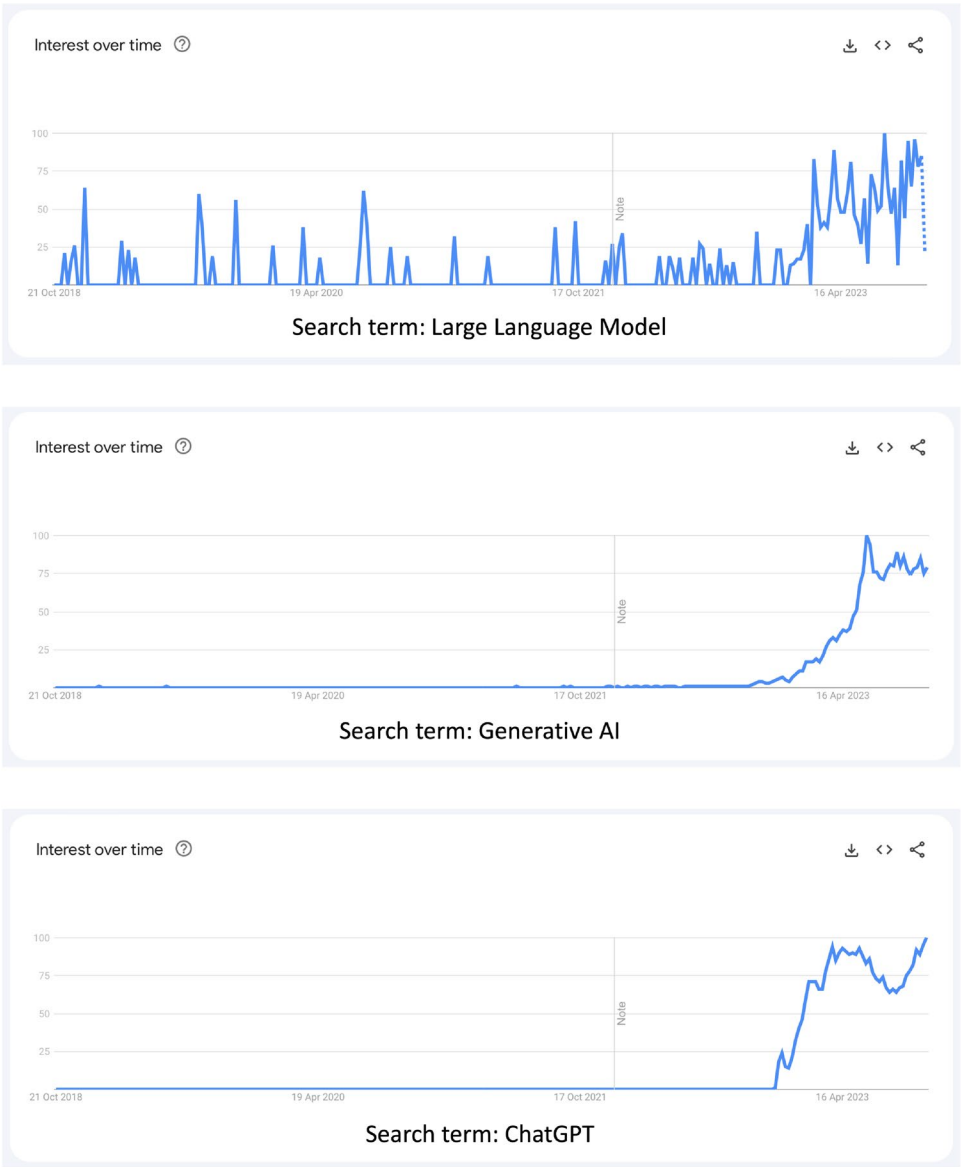

Fig. 2 Evolution of interest in generative AI related to the appearance of ChatGPT

including LLaMa 2 and its variations, gpt−3.5-turbo, BERT and more.

Prompting Techniques

Prompting techniques play a pivotal role in bridging reasoning and code generation capabilities within LLMs. Approaches like Chain-of-Thought (CoT) prompting, introduced by [68], significantly impact code generation by guiding LLMs to break down complex problems into sequential, intermediate reasoning steps. This explicit articulation of the thought process before generating the final code helps the model manage intricate logic, leading to more accurate and logically consistent code outputs, particularly beneficial for tasks demanding detailed algorithmic problem-solving. Similarly, Program-of-Thought (PoT) prompting, proposed by [11], further enhances code generation by structuring the task as executable program steps. It utilizes modular depictions similar to functions or modules found in Software Engineering, encouraging the LLM to draft the program's logic or steps prior to producing the actual code fragments. This approach not only improves the structural quality and adherence to best practices but also yields stepwise, more interpretable code that mirrors human software development workflows. An empirical categorization of prompting techniques can be found in [20]. Authors identify seven main categories:

1. Logical and Sequential Processing: break down reasoning tasks into simple steps.
2. Contextual Understanding and Memory: reference past interactions to ensure coherence in dialogues.

3. Specificity and Targeting: distinguish between information types and align with specific targets.
4. Meta-Cognition and Self-Reflection: improve the answer through introspective prompting.
5. Directional and Feedback: direct the model towards specific tasks.
6. Multimodal and Cross-Disciplinary: integrate diverse input modes and knowledge domains.
7. Creative and Generative: produce creative, and exploratory content.

Incorporating these prompting strategies can improve the accuracy, diversity, and practicality of LLM-generated code, underscoring the importance of reasoning in advancing code generation methodologies.

Related Work

A state-of-the-art analysis revealed that literature presents few reviews related to LLMs. Authors of paper [41] show the potentialities of ChatGPT in reflective writing within university courses. They evaluate the ability of ChatGPT to produce diverse reflective responses using nine prompting strategies. The findings suggest that ChatGPT can generate high-quality reflective responses, outperforming student-written reflections across various assessment criteria. The work presented in [19] explores the reasoning capacity of LLMs like GPT−3.5, GPT-4, and BARD. The paper highlights the limited proficiency of all three models in tasks involving Inductive, Mathematical, and Multi-hop Reasoning. To enhance zero-shot performance, the study proposes a set of engineered prompts for all three models, providing a detailed and comprehensive analysis of the results. The opinion paper presented in [14] provides a multi-disciplinary exploration of the applications, opportunities, challenges, and impacts of transformative AI tools like ChatGPT. Drawing insights from 43 contributors across diverse fields, including computer science, marketing, education, and nursing, the article acknowledges the potential for ChatGPT to enhance productivity in various industries. It highlights both positive aspects, such as gains in banking, hospitality, and IT, and concerns, including disruptions, privacy threats, biases, and misuse. The main thematic areas for further research are identified in knowledge, transparency and scholarly research. The study proposed in [21] employs a scalable crowdsourcing data-driven framework to investigate ChatGPT's code generation performance from diverse perspectives across social media platforms. The analysis reveals that ChatGPT is utilized in over 10 programming languages, with Python and JavaScript being the most popular, for tasks ranging from code debugging to interview preparation. The dominant emotion associated with ChatGPT's code generation is fear, overshadowing other emotions.

As far as we know, there are no comprehensive reviews investigating how LLMs are effectively used in the code generation domain while simultaneously providing detailed insights into the preferred programming languages and a structured analysis of the limitations versus benefits of these models. Existing surveys often focus on isolated aspects, such as performance evaluation or the application of LLMs in specific programming tasks, without addressing the broader implications of their adoption.

In particular, we analyze not only the technical performance of LLMs but also their ethical, security, and practical challenges, including biases in generated code, intellectual property concerns, and potential vulnerabilities.

Research Methodology - Literature Review

We follow the Preferred Reporting Items for Systematic Reviews and Meta-Analyses (PRISMA) methodology [55] to perform the proposed review. It is composed of different steps: definition of Research Questions, gathering and filtering process through inclusion and exclusion criteria, and graphical ad textual findings and discussion. The steps performed to reach the final number of papers are shown in Fig. 3. We analyze four main Research Questions (RQs), ranging from independent ones such as temporal distribution of published papers to specific ones like performance analysis and preferred programming languages. We then apply different criteria to filter gathered papers, from date of publication to language, and from main topic to novelty in the paper.

Definition of Research Questions

This review aims to identify the state-of-the-art on performing code generation tasks using a LLM, as well as to highlight the trend of this research field and research gaps to work on. To accomplish this, we develop the following RQs:

- **RQ1: What is the trend topic of LLM used for code generation, starting from 2021?** The goal of this question is to understand the evolution during years of the analyzed topic.
- **RQ2: What is the overall performance in accomplishing the code generation task?** This question aims to understand how LLMs are used in the topic of code generation, if they perform well in different operations (i.e.: identifying bugs, understand programming languages syntax, and more), and what model can perform better between the ones available online.

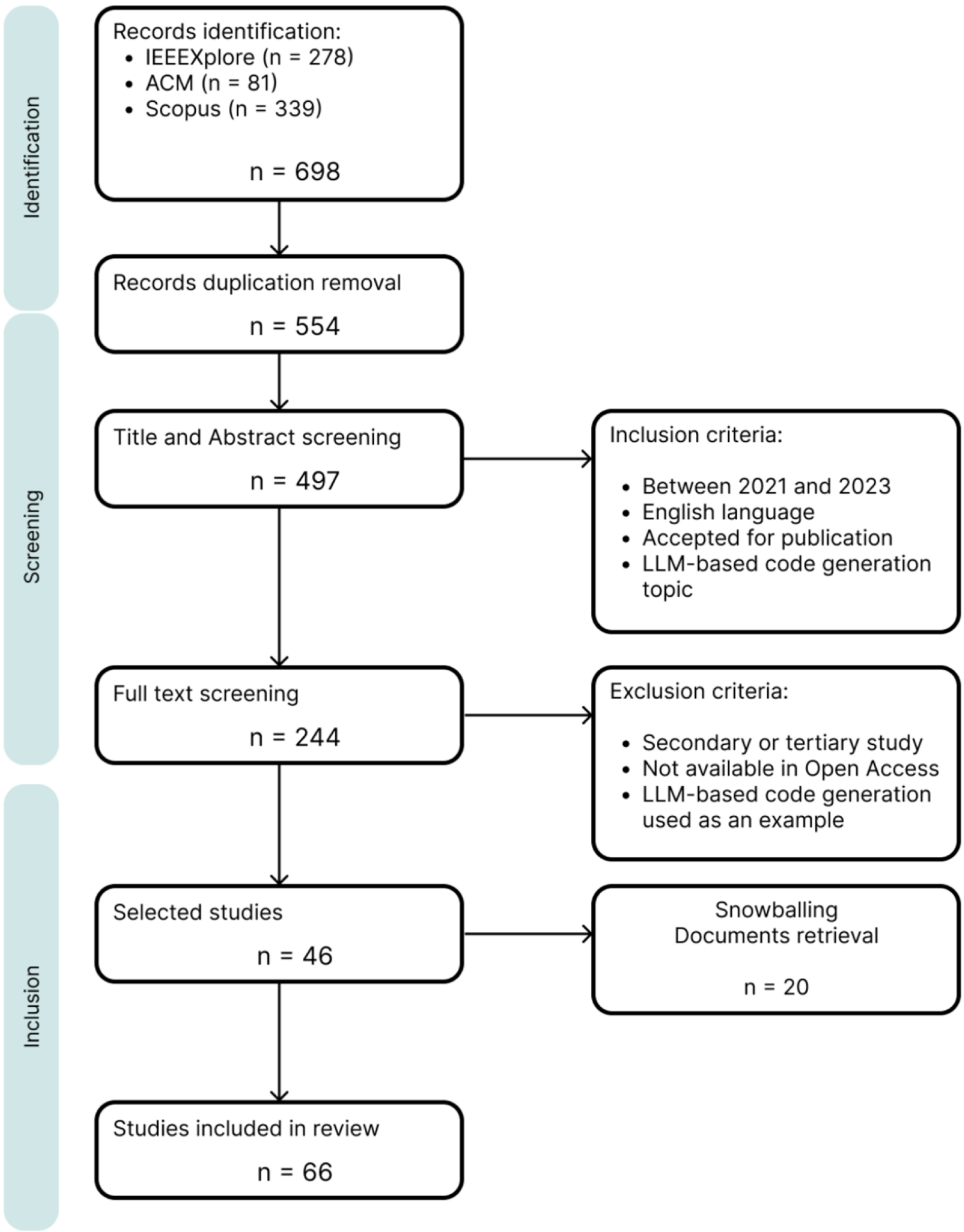

Fig. 3 PRISMA methodology

- **RQ3: What programming languages are used in current research?** This RQ analyzes the topic from the programming languages point of view. It aims to understand which languages are preferred and why.
- **RQ4: What is the gap between benefits and constraints of using an LLM to perform automatic code generation?** The purpose of this RQ is to investigate current open problems and identify research gaps and trends.

Papers gathering process

To ensure reliability and completeness of the review, we screened relevant research databases. A PICO (Population, Intervention, Comparison, Outcomes) technique is pursued to find the research string [18].

((Generative AND ai) OR llm OR (Large AND Language AND (Model OR Models))) AND code generation

The papers gathered from different databases, such as IEEEXplore, ACM, Google Scholar, Scopus, is 698.

Inclusion and Exclusion Criteria

The definition of inclusion and exclusion criteria helps in defining which studies will be analyzed to answer the defined RQs. In this work, inclusion criteria include:

Table 1 Value to be extracted for each Research Question

RQ	Data	Value
RQ1	Publication date	Between 2021 and 2023
RQ1	Research type	Conference, journal, book
RQ2	Research category	Proposal, implementation, etc
RQ2	Performance	Performance indicators
RQ3	Performance	Analyzed programming languages
RQ4	Ethics	Usage of generated code in critical tasks
RQ4	Validity	Gap between benefits and limitations

- Publication date between 2021 and 2023, following the results gathered from Fig. 1.
- Publication written in English.
- Publication focused on the code generation topic though LLMs.

Exclusion criteria are presented below:

- Secondary or tertiary study, such as other literature reviews or mapping studies.
- Publication not accepted in a conference, journal or book.
- Publication where the analyzed topic is used only as an example.

The obtained number of papers after the application of the explained criteria is 66. This is considered an acceptable number for our purposes.

The given Research Questions are addressed by the data extraction and categorization. Table 1 displays the form used for extracting data.

Results and Discussion

This section highlights the main findings obtained from the papers analysis and divided by the RQs defined in Sect. "Research Methodology - Literature Review".

RQ1: What is the Trend Topic of LLM Used for Code Generation, Starting from 2021?

Fig. 4 shows the papers distribution on a month basis, from 2021 to 2023. A concentration of these papers starts from April 2023: this is an expected result, because of the born of ChatGPT as mass-use tool in November 2022. The highest number of occurrences happens between August and October 2023, then there is a slight reduction of occurrences. The waning focus on research dedicated solely to code generation tasks within the domain of LLMs may be attributed, in part, to the broader versatility and applicability of these models across a spectrum of computational tasks. The advent of LLMs heralded a paradigm shift in Artificial Intelligence research, captivating scholars due to their potential to revolutionize numerous domains beyond code generation. Moreover, there are some already optimized tools for code generation tasks, starting from ChatGPT itself. In particular, the version powered by gpt-4 model performs very well in the Python language domain [13]. There exist also other industrial ready tools to generate code, like fine-tuned LLMs such as WizardCoder or CodeLlama.[4]

The graph proposed in Fig. 5 distinguishes between conference papers, journal articles or book sections. Journal articles cover about 85% of the considered papers: this is attributable to the choice not to include papers not available in open access among the exclusion criteria. It is important

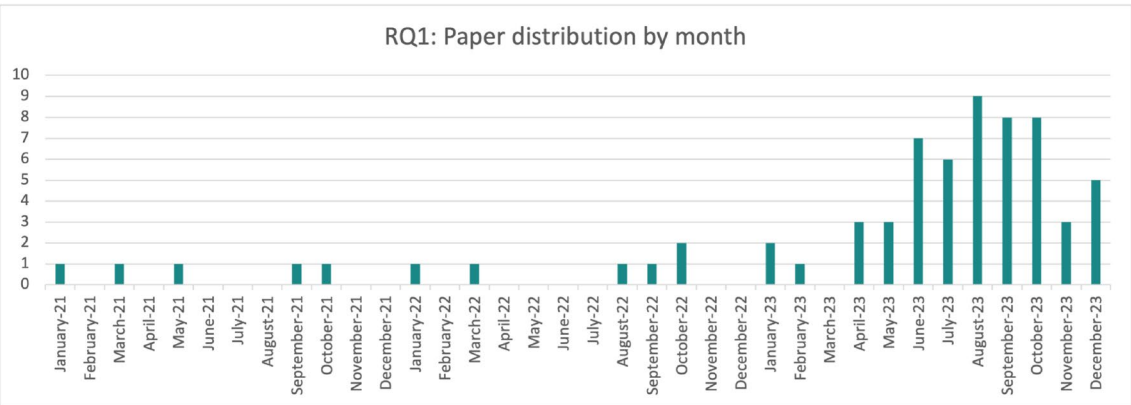

Fig. 4 Papers distribution by month between 2021 and 2023

[4] https://huggingface.co/spaces/mike-ravkine/can-ai-code-results.

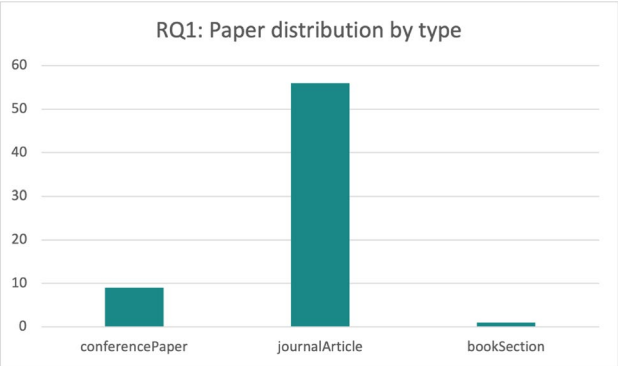

Fig. 5 Papers distribution by type

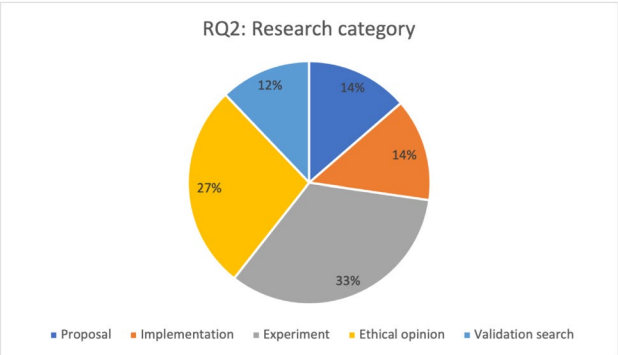

Fig. 6 Papers divided by research category

to underline that among the papers considered in the first collection, most of these were only in preprint format, therefore not yet published. This large number of preprints is a symptom of a research context that is still young and constantly growing.

The journal with the highest number of occurrences is Elsevier Computers and Education: Artificial Intelligence with 6 papers, followed by IEEE Software. Other international journals appear with one paper each.

RQ2: What is the Overall Performance in Accomplishing the Code Generation Task?

The second RQ is focused on understanding if papers evaluate the code generation task and the performances they achieve. Main findings are summarized in Fig. 6. 33% of the analyzed papers experiment with LLMs on different tasks, exploring how the intervention of Generative AI helps humans and their output. Some examples are given below:

- Paper [22] presents a crowdsourcing data-driven framework to investigate ChatGPT's code generation performance by analyzing social media data, revealing insights into its usage across programming languages, tasks, and emotional associations, and releasing a dataset for evaluating the quality of generated code.
- Paper [67] presents an innovative approach using a generative adversarial network (GAN) to enhance both code generation and code search techniques in software engineering. By treating code generation and code search as the generator and discriminator in the GAN framework, the study showcases consistent performance improvements across eight different settings.
- Paper [57] assesses the impact of ChatGPT, specifically GPT-4, on digital forensics through a series of experiments across various use cases such as artefact understanding, evidence searching, code generation, anomaly detection, incident response, and education. It evaluates the strengths, risks, and potential applications of ChatGPT in digital forensics. While acknowledging potential low-risk applications, the paper emphasizes limitations due to the need to upload evidence and the requirement for substantial topic knowledge to identify errors. It suggests that, with appropriate expertise, ChatGPT could serve as a supporting tool in certain circumstances.
- Paper [44] explores the utilization of GPT-4 for generating Governance, Risk, and Compliance (GRC) policies aimed at deterring and mitigating ransomware attacks involving data exfiltration. The research compares the efficacy, efficiency, comprehensiveness, and ethical adherence of GPT-generated policies against those formulated by established security vendors and government cybersecurity agencies. Results showcase instances where GPT-generated policies outperform human-generated ones, particularly when tailored input prompts are provided. Recommendations for corporate adoption of GPT in GRC policy-making are offered based on these findings.

Ethical and Security Issues in Code Generation Using LLMs

The application of LLMs in code generation introduces complex ethical and security challenges that require careful consideration. A critical concern is the inherent bias in generated code, which reflects the limitations of the datasets used for training. Models trained on public repositories often perpetuate biases, producing outputs that may be discriminatory or non-inclusive. Large generated code snippets can contain biased constructs, such as gender-specific variable names or assumptions about user behavior. Addressing these biases necessitates improved dataset curation, increased transparency in model training, and the integration of bias detection mechanisms in post-generation workflows.

Another pressing issue is the unintentional generation of insecure code. LLMs may produce code containing vulnerabilities, such as SQL injection or cross-site scripting (XSS),

which could be exploited if deployed without rigorous review. Furthermore, the reliance on proprietary datasets for training raises significant intellectual property (IP) concerns. Generated outputs occasionally replicate copyrighted material: robust attribution mechanisms and clear legal frameworks are necessary to ensure ethical use of generative AI in software development.

Beyond technical considerations, the ethical implications of LLMs extend to their use in education and industry. In academic settings, overreliance on LLMs risks diminishing students' foundational programming skills, while in industry, such tools could disrupt traditional developer roles. Addressing these challenges involves designing educational curricula that balance traditional and AI-augmented programming practices, as well as offering re-skilling programs for professionals.

A relevant number of papers talk about ethical implications of using LLMs in generating code or text.

- Paper [54] explores the future relevance of traditional programming languages in end-user programming among the rise of generative AI models, proposing the 'generative shift hypothesis' that anticipates qualitative and quantitative expansions in end-user programming facilitated by generative AI.
- Paper [51] reviews the evolution of software development processes concerning the appropriate targeting of automation to overcome obstacles. It recommends adopting a similar approach in integrating generative AI tools into the software development community.
- Paper [50] explores the transformative potential and risks of large language models (LLMs) in software engineering tasks, pondering the implications of AI-generated content and its impact on professionals in various domains.

A less relevant number of papers focus on implementing new platforms other than experimenting with an existing or a new fine-tuned model, on validating an existing search or on proposing new approaches. We expect this last category to be higher in other fields and not in code generation, because of the focus, by researchers, on better understanding the syntax and best practices for different programming languages and not on proposals on new programming languages or on new approaches on code understanding and documentation. The inclination towards established languages could also be practical in terms of applicability. Enhancing existing languages and their documentation aligns with real-world needs, facilitating better developer tools, code readability, and comprehension.

Figure 7 shows the value that each paper carries to the community in the analyzed field: software development if the paper develops new tools (3%), bibliometric if the paper just makes an overview of already available platforms (3%), reliability if the paper considers the reliability of the generated code (7%), technology if the paper focuses on analyzing the topic and how LLMs can help in code generation tasks (50%), quality if the paper is based on the quality of the developed code (27%), performance if the paper focuses on how the LLM performs in accomplishing the required task (2%), business analysis if there are economic considerations (6%), architecture if the paper relies on modeling a new software architecture (2%). The distribution of topics within papers reflects varying priorities and interests within the research community. The focus on technology follows the foundational drive to innovate and refine the technical capabilities of these models. Given the rapid evolution of LLMs, researchers are often driven to explore new techniques, optimization strategies, and model architectures to enhance their efficiency, accuracy, and versatility across different tasks such as code generation. The emphasis on quality in a significant portion of papers aligns with the goal of ensuring robustness and efficacy of code generated by these models. Quality considerations encompass factors such as code correctness, readability, maintainability, and adherence to coding standards. This focus on quality reflects the practical need to ensure that the code produced by these models meets industry standards and is usable in real-world scenarios.

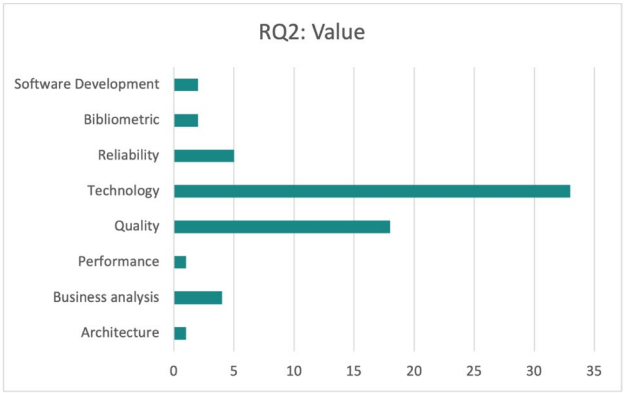

Fig. 7 Papers divided by value added to the community

Figure 8 shows the application area a paper falls into. 26% of papers belong to the education area, suggesting an interest in using LLMs for code generation tasks in classes for multiple reasons: creation of online learning environments and quizzes [12], analysis of how academy will react to LLMs [17, 59], classification between student-generated and ChatGPT-generated text and comments [42, 52], and analysis on the impact of using ChatGPT in programming education on students' computational thinking skills, programming self-efficacy, and motivation [70].

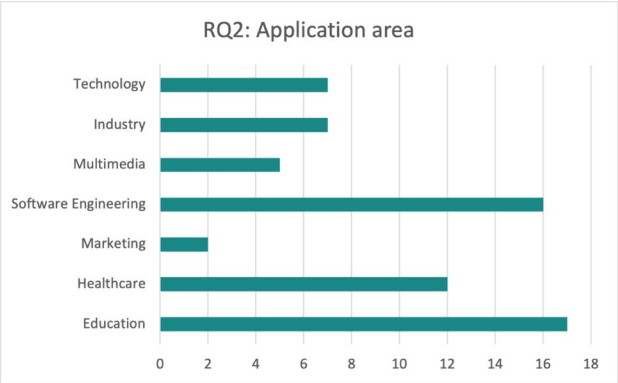

Fig. 8 Papers divided by main application area

Software Engineering comes at second place; this is an expected result, due to the fact that research on LLMs is constrained on code generation tasks, that strictly belong to the Software Engineering field: transforming ChatGPT into a low-code developer [43], generating commit messages through contextual information in repositories [48], analyzing LLMs software modeling capabilities and consistency issues [8], and exploring the potential impact of generative AI technologies like ChatGPT on the Software Development Lifecycle (SDLC) [53]. Healthcare is also popular in this kind of research: LLMs are used for making diagnoses of low- and medium-risk diseases based solely on symptoms [9], for offering reliable suggestions and treatment options to patients [27] and in general for supporting healthcare communication research [46].

RQ3: What Programming Languages are Used in Current Research?

An analysis on the preferred programming language used to conduct the research is also conducted. Only a small number of papers explicitly mentions the programming languages used within the research for code generation tasks. Python's prevalence among the discussed languages in these papers might reflect its widespread adoption in the field of machine learning and natural language processing, making it a natural choice for implementing and experimenting with LLMs. The absence of explicit mention of programming languages in the majority of the papers might indicate that researchers focus more on the methodologies, techniques, or algorithms applied to code generation tasks rather than the specific code generation. Results of this analysis are shown in Table 2.

LLMs Performance

The performance of LLMs in code generation tasks varies significantly based on the context and the specific models employed. Benchmarking tools like DevQualityEval [62] provide standardized frameworks to assess and compare these models' effectiveness in real-world software development scenarios. A comprehensive evaluation of over 80 LLMs was conducted for Java, Go, and Ruby programming languages. OpenAI's o1-preview and o1-mini models led the leaderboard in functional performance, narrowly outperforming Anthropic's Claude 3.5 Sonnet. However, they exhibited significantly higher latency and verbosity, highlighting trade-offs between speed and communication clarity. This evaluation was based on five full benchmarking runs, with different metrics and logs [63]. The extended task setup and reasoning provide a thorough comparison, showcasing the nuanced strengths of each model in context-specific code generation scenarios. Figure 9 shows the results of the conducted analysis.

Despite their demonstrated capabilities, LLMs exhibit several intrinsic limitations in code generation. A significant challenge resides in their propensity to generate code containing subtle errors or inaccuracies, potentially leading to downstream issues if not rigorously verified. Current models may struggle with context maintenance over complex, multi-turn code generation tasks, resulting in incomplete or erroneous outputs.

RQ4: What is the Gap Between Benefits and Constraints of Using an LLM to Perform Automatic Code Generation?

The last RQ is focused on understanding the advantages and the problems of using LLM for code generation. In Table 3, we discuss it according to the analyzed papers.

Figure 10 illustrates the taxonomy of programming languages used in the reviewed papers.

To summarise, LLMs are highly effective instruments for generating code. They are simple to train, and the outcomes are both speedy and highly encouraging. However, it is essential to acknowledge that there are some limitations associated with them. One significant challenge is identifying incorrect information or code that may be generated, which could lead to more severe issues when used. Therefore, it is essential to exercise caution when using LLMs and ensure that the generated code passes rigorous quality control checks to avoid any potential issues.

Threats to Validity

We highlight some threats to the review's validity, identified during the writing phase.

- Fast evolving landscape: The field of LLMs applied to code generation is characterized by its dynamic and constantly evolving nature. It is challenging to

Table 2 Programming languages analyzed in relevant papers

Programming Language	Paper	Description
Python	[47]	Introduces LEVER, a method enhancing language-to-code generation by training verifiers to assess generated programs based on natural language input, the code, and execution results. It consistently improves code LLMs across table QA, math QA, and basic Python programming datasets, achieving state-of-the-art results
	[67]	Presents an innovative approach utilizing a generative adversarial network (GAN) to enhance both code generation and code search techniques in software engineering. By treating code generation and code search as the generator and discriminator in the GAN framework, the study showcases consistent performance improvements across eight different settings
	[49]	Introduces CODEGEN, an open large language model for code synthesis, trained on natural language and programming language data, facilitating program generation from problem specifications. Additionally, it introduces the JAXFORMER training library and demonstrates CODEGEN's competitiveness in Python code generation through HumanEval and investigates multi-turn prompts for improved program synthesis using the Multi-Turn Programming Benchmark (MTPB)
	[61]	Introduces IntelliCode Compose, a multilingual code completion tool leveraging a generative transformer model trained on extensive source code data. It predicts sequences of code tokens for various languages like Python, C#, JavaScript, and TypeScript, achieving high edit similarity and low perplexity for Python programming
JavaScript	[39]	Explores the application of the GPT-2 model for Automated Program Repair (APR) in JavaScript. It fine-tunes the model with JS code snippets to automatically fix JavaScript bugs, achieving an overall accuracy of up to 17.25% but facing challenges in generating effective bug-fixes in certain cases
Multiple	[4]	The paper introduces DCServCG, a Data-Centric Service Code Generation model aimed at enhancing web service-based systems by capturing and utilizing service usage information from public repositories that share Open Source Software (OSS). DCServCG improves code generators by focusing on service-based code characteristics and addressing sequence overlap and bias issues through a data-centric concept employing conditional text generation. The evaluation demonstrates improved language modeling metrics, indicating reduced perplexity and better generalization performance compared to traditional models like ServCG
	[56]	Explores the capabilities of LLMs, using OpenAI Codex, in generating programming exercises and code explanations for programming courses, evaluating the quality and novelty of the generated content both qualitatively and quantitatively
	[22]	Presents a crowdsourcing data-driven framework to investigate ChatGPT's code generation performance by analyzing social media data, revealing insights into its usage across programming languages, tasks, and emotional associations, and releasing a dataset for evaluating the quality of generated code

encompass all the relevant literature, given the novelty of the subject, the limited time frame for the review, and the constant influx of new research. Moreover, the dynamic growth of the field introduces the possibility that some crucial contributions may have emerged after the review period.

- Pre-prints exclusion: A significant threat to the completeness of this review comes from the decision to exclude non-published papers, such as pre-prints. The exclusion of pre-prints, which are prevalent in emerging and fast-paced research domains, may lead to an oversight of valuable insights and advancements that have yet to undergo formal peer review.
- Open Access papers: The introduction of the inclusion criteria to include only open access papers introduces a potential bias in the selection process. This review may only encompass some of the available literature by excluding papers that are not freely available, thereby missing valuable contributions from subscription-based or proprietary sources. This limitation might result in a partial representation of the diverse research landscape surrounding the analyzed topic.
- Search string: The choice of the search string employed in online scientific databases may introduce a level of subjectivity and potential bias. While the chosen string is designed to capture a broad spectrum of relevant literature, alternative search strings may yield different results.
- Performance in the topic: The proficiency of LLMs in code generation tasks may result in a lack of research that explicitly addresses challenges or advancements, given that these models may already perform well in this domain [72].

Despite these potential limitations, this review presents a comprehensive overview of the current literature on LLMs applied to code generation tasks. We aim to improve the review further and update it in the future to include new publications and new approaches to the topic.

Conclusion and Future Directions

To conclude, our review provides a general picture of the capability of LLMs in code generation. We analyzed research papers published between 2021 and 2023. Our

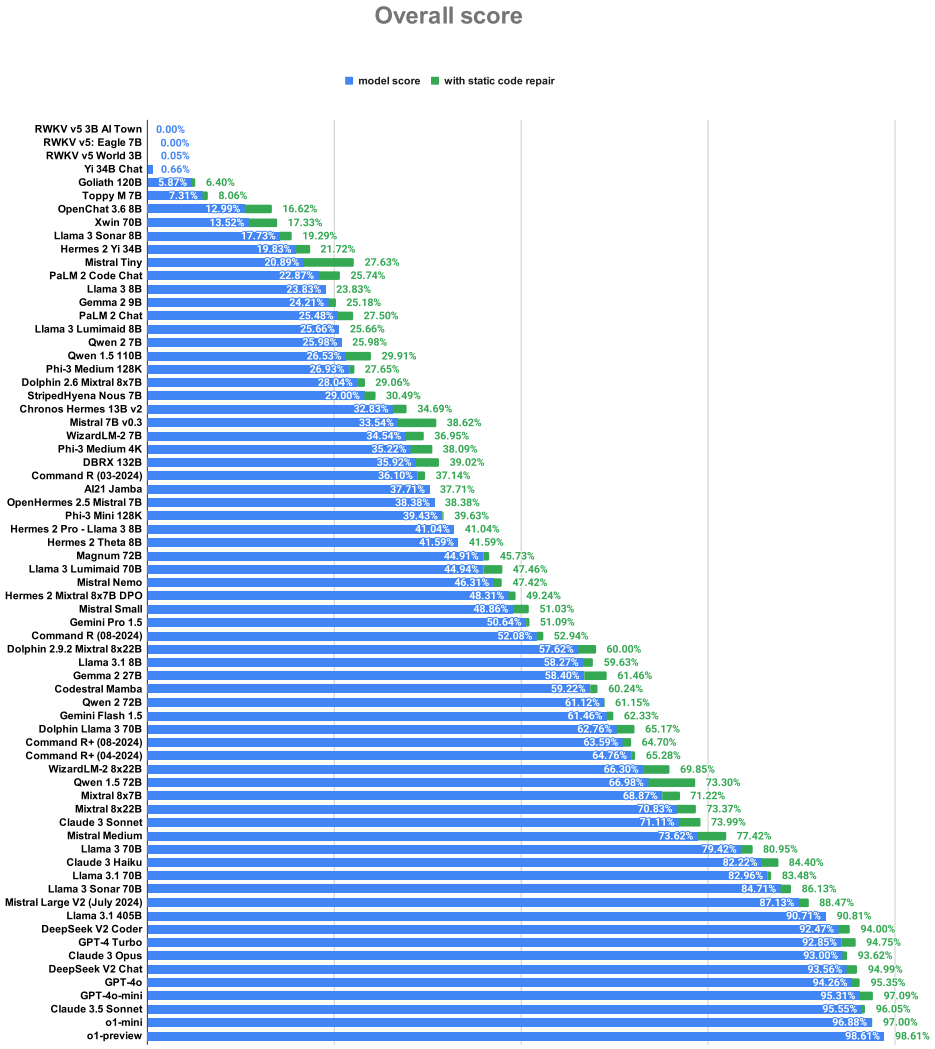

Fig. 9 DevQualityEval results as shown in [63]

findings reveal a significant increase in papers published from April 2023 onwards, as highlighted in Fig. 4. This can be attributed to the widespread adoption of ChatGPT as a tool for mass communication, which was introduced in November 2022. Most papers were published between August and October 2023, after which there was a slight decrease in published papers. This reduction in research focus dedicated solely to code generation tasks within the domain of LLMs could be because of these models' broader versatility and applicability across a spectrum of computational tasks.

We saw that Python is a popular choice for building and testing large language models, likely due to its widespread use in machine learning and natural language processing. Although most papers do not mention specific programming languages, this could suggest that researchers focus more on the methods, techniques, and algorithms used for generating code rather than the language used for implementation.

Finally, it is widely recognized that LLMs (Language Model Models) are invaluable in generating code. These models are characterized by their ease of training and the rapidity and reliability of their results. Nevertheless, it is crucial to acknowledge that these models have their limitations. One of the most significant challenges is identifying incorrect information or code generated by these models, which can have serious consequences when deployed. Therefore, it is imperative to exercise caution when using LLMs and ensure that the output code passes through rigorous quality control checks to mitigate potential risks.

Practical Implications for Developers

This review highlights the practical implications for software developers using LLMs in their workflows. While LLMs can accelerate development tasks like code generation, debugging, and task automation, developers must approach their use cautiously. Developers are advised to treat LLM-generated code as suggestions requiring rigorous human review, testing, and validation to avoid inaccuracies and ensure alignment with project standards. Security considerations

Table 3 Pros and cons according to the analyzed papers

Paper	Discussion
[4]	It improves the accuracy of code generation for web service-based systems by using service usage data from open-source software repositories. It tackles issues related to sequence overlap and bias by generating data-based text. The resulting model performs better generalization than traditional models like ServCG, as demonstrated by improved perplexity metrics
[7, 29, 37, 56]	The study emphasizes the role of ChatGPT as an educational tool in design processes, highlighting its benefits in creating interactive learning environments, including automated grading, personalized learning experiences, language lessons, and customized content creation. However, it also emphasizes the limitations of solely relying on ChatGPT and stresses the importance of combining it with manual calculations and established simulation software to validate engineering results. It is crucial to be cautious of potential biases in content generation and to curate educational materials carefully. The challenge is balancing the advantages of ChatGPT with responsible curation to ensure that the educational content is free of bias
[3, 10, 27, 34, 36, 40, 46]	They acknowledge ChatGPT's effectiveness in various NLP tasks and its potential applications in healthcare services. These systems can be beneficial in automating healthcare processes, improving patient care, and simplifying report generation. However, they emphasize ethical concerns, data interpretation challenges, accountability issues, and privacy concerns related to sensitive medical information when utilizing ChatGPT in healthcare. The articles also emphasize the importance of ensuring originality, accuracy, and academic integrity before these systems are extensively integrated into clinical practices. Additionally, they underline the lack of studies exploring LLM applications in nucleic acid research
[25, 45]	Authors recognize ChatGPT's ability to generate diverse content and its growing importance in various domains. However, it also points out certain restrictions or limitations of the tool, such as sometimes the generation of incorrect information, may rise with biased content, etc. They emphasize the need for validation and combined methods for accuracy
[48]	It highlights the significant improvements that CoreGen demonstrates over baseline models. Additionally, it discusses the potential benefits of training contextualized representations on larger code corpora and adapting this framework to other code-to-text generation tasks
[44]	The study shows that in certain cases, cybersecurity policies generated by GPT (a type of artificial intelligence) are better than policies created by humans, especially when tailored prompts are used. However, the study also recognizes that there are limitations and stresses the importance of having specialized input, human moderation, and expert input to achieve the best results. The study also provides recommendations for companies that are considering integrating GPT into their GRC (governance, risk, and compliance) policy formulation
[28]	For the authors, ChatGPT's advantages lie in its versatility for human–machine interface creation, text generation, and knowledge provision, offering potential applications like chatbots, virtual assistants, and automation within Industry 4.0. However, limitations may include biases in generated content and challenges in ensuring accuracy and reliability in automated processes. The gap involves balancing the advantages with mitigating biases and ensuring the reliability of automated operations
[6, 35]	According to them, AI-generated poetry's benefits showcase the difficulties people face distinguishing it from human-written poetry. This helps us understand how humans perceive AI-generated content. However, there are limitations as some people negatively respond to algorithm-generated poetry, indicating a preference for human-written work
[71]	It emphasizes ChatGPT's potential in overcoming challenges faced by existing deep learning approaches in agricultural text classification, demonstrating promising performance without fine-tuning on agricultural data. However, it acknowledges this as a preliminary study and highlights the need for further exploration to establish ChatGPT's applicability and performance in diverse agricultural contexts
[58]	The paper provides valuable empirical insights from a case study, identifying critical AI capabilities and emphasizing the need for business model innovation to scale AI effectively in manufacturing firms. However, it's essential to further explore the practical implementation and long-term impact of the proposed co-evolutionary framework and its mechanisms for scaling AI through business model innovation
[66]	The study emphasizes the effectiveness of schema congruence in improving retrieval accuracy and highlights gamified learning's potential to evoke positive emotions despite an underlying predominance of negative emotions
[8, 15, 22, 64]	The authors reveal patterns of ChatGPT usage in code generation across various programming languages and tasks, highlighting prevalent emotions like fear associated with its usage, and provide a dataset for evaluating the quality of generated code. They identified ChatGPT's limited performance in software modelling tasks compared to code generation, offering insights for short-term potential and suggestions for improvement in the modelling community
[39, 54, 60]	They demonstrate the GPT model's ability to generate syntactically correct code and highlight its limitations in producing effective bug fixes in specific scenarios. However, the overall accuracy suggests room for improvement in its performance for program repair. They explore the explainability needs of generative AI models in software engineering contexts and propose XAI features to address these needs, highlighting the importance of human-centred design in driving technical advancements

Table 3 (continued)

Paper	Discussion
[26]	It demonstrates advancements in video generation by addressing temporal aspects and auxiliary tasks, yet may require further refinement for real-world application

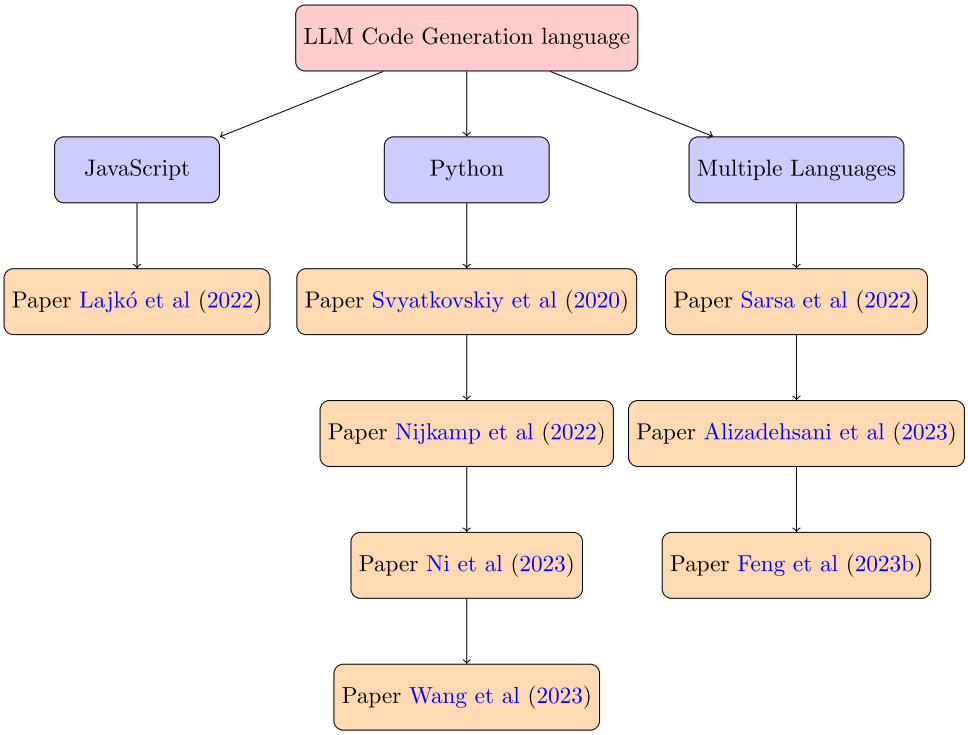

Fig. 10 Taxonomy of programming languages used in the reviewed papers

are crucial, as AI-generated code can introduce vulnerabilities like SQL injection and XSS, necessitating the integration of security scanning tools and manual reviews. Developers should understand LLMs' limitations, including difficulties with complex logic and specific domains. Effective prompt engineering and a hybrid approach that combines LLM assistance with traditional development practices are recommended.

Research Gaps and Future Directions

Despite the significant advancements in LLMs for code generation, several research gaps remain unaddressed, offering opportunities for further exploration. One major gap lies in improving the contextual understanding of LLMs during multi-turn code generation tasks. Current models often struggle with maintaining context over long sequences, resulting in incomplete or erroneous outputs. Enhancing the contextual capabilities of LLMs through advanced architectures or fine-tuning strategies is a critical area for future work.

Another gap involves addressing the ethical and security risks associated with LLM-generated code. While existing studies highlight vulnerabilities and biases, comprehensive frameworks for mitigating these risks remain underdeveloped. Future research should focus on integrating real-time code validation techniques, developing bias detection algorithms, and creating transparent auditing tools to enhance trust in LLM outputs. Furthermore, the lack of robust attribution mechanisms to trace generated code back to its training data poses challenges in intellectual property compliance and accountability. Efforts to establish such mechanisms will be vital for ensuring the responsible use of LLMs.

There is also a need for models tailored to specific domains or programming languages. While general-purpose LLMs demonstrate strong performance, domain-specific models could significantly improve accuracy and utility in specialized applications and real-world use cases, such as cybersecurity, embedded systems, or healthcare software development. Techniques such as Retrieval-Augmented Generation (RAG) or Fine Tuning can be explored to obtain such domain-specific models.

Addressing the urgent need for real-time validation and verification tools for LLM-generated code is critical to ensure its safety and reliability in practical applications. Furthermore, developing comprehensive frameworks to

mitigate security vulnerabilities and ethical biases remains an imperative area for future research. Collaborative research between academia and industry could drive the development of such models, leveraging domain expertise to address real-world challenges.

Funding Open access funding provided by Politecnico di Bari within the CRUI-CARE Agreement.

Data availability No new data were created during this study.

Declarations

Conflict of Interest On behalf of all authors, the corresponding author states that there is no conflict of interest.

Ethical Approval This article does not contain any studies with human participants or animals performed by any of the authors.

References

1. Abburi H, Suesserman M, Pudota N, et al. Generative ai text classification using ensemble llm approaches. 2023. arXiv:2309.07755
2. Adamopoulou E, Moussiades L. An overview of chatbot technology. In: IFIP international conference on artificial intelligence applications and innovations, Springer, 2020; pp 373–383
3. Ali H, Patel P, Obaitan I, et al. Evaluating ChatGPT's Performance in Responding to Questions About Endoscopic Procedures for Patients. iGIE 2023 https://doi.org/10.1016/j.igie.2023.10.001
4. Alizadehsani Z, Ghaemi H, Shahraki A, et al. DCServCG: A data-centric service code generation using deep learning. Eng Appl Artif Intell. 2023;123: 106304. https://doi.org/10.1016/j.engappai.2023.106304.
5. Anand Y, Nussbaum Z, Duderstadt B, et al. Gpt4all: Training an assistant-style chatbot with large scale data distillation from gpt-3.5-turbo. 2023 https://github.com/nomic-ai/gpt4all
6. Bezirhan U, von Davier M. Automated reading passage generation with OpenAI's large language model. Computers and Education Artificial Intelligence. 2023;5: 100161. https://doi.org/10.1016/j.caeai.2023.100161.
7. Bull C, Kharrufa A. Generative AI Assistants in Software Development Education: A vision for integrating Generative AI into educational practice, not instinctively defending against it. IEEE Software 2023 pp 1–9. https://doi.org/10.1109/MS.2023.3300574
8. Cámara J, Troya J, Burgueño L, et al. On the assessment of generative AI in modeling tasks: An experience report with ChatGPT and UML. Softw Syst Model. 2023;22(3):781–93. https://doi.org/10.1007/s10270-023-01105-5.
9. Caruccio L, Cirillo S, Polese G, et al. Can ChatGPT provide intelligent diagnoses? A comparative study between predictive models and ChatGPT to define a new medical diagnostic bot. Expert Syst Appl. 2024;235: 121186. https://doi.org/10.1016/j.eswa.2023.121186.
10. Chatterjee S, Bhattacharya M, Lee SS, et al. Can artificial intelligence-strengthened ChatGPT or other large language models transform nucleic acid research? Molecular Therapy - Nucleic Acids. 2023;33:205–7. https://doi.org/10.1016/j.omtn.2023.06.019.
11. Chen W, Ma X, Wang X, et al. Program of thoughts prompting: Disentangling computation from reasoning for numerical reasoning tasks. Transactions on Machine Learning Research 2023 https://openreview.net/forum?id=YfZ4ZPt8zd
12. Diwan C, Srinivasa S, Suri G, et al. AI-based learning content generation and learning pathway augmentation to increase learner engagement. Computers and Education Artificial Intelligence. 2023;4: 100110. https://doi.org/10.1016/j.caeai.2022.100110.
13. Doughty J, Wan Z, Bompelli A, et al. A comparative study of ai-generated (gpt-4) and human-crafted mcqs in programming education. arXiv preprint 2023. arXiv:2312.03173
14. Dwivedi YK, Kshetri N, Hughes L, et al. "so what if chatgpt wrote it?'' multidisciplinary perspectives on opportunities, challenges and implications of generative conversational ai for research, practice and policy. Int J Inf Manage. 2023;71: 102642.
15. Ebert C, Louridas P. Generative AI for Software Practitioners. IEEE Softw. 2023;40(4):30–8. https://doi.org/10.1109/MS.2023.3265877.
16. EDiGiacinto Localai repository. 2023 https://github.com/go-skynet/LocalAI
17. Eke DO. ChatGPT and the rise of generative AI: Threat to academic integrity? Journal of Responsible Technology. 2023;13: 100060. https://doi.org/10.1016/j.jrt.2023.100060.
18. Eriksen MB, Frandsen TF. The impact of patient, intervention, comparison, outcome (pico) as a search strategy tool on literature search quality: a systematic review. Journal of the Medical Library Association JMLA. 2018;106(4):420.
19. Espejel JL, Ettifouri EH, Alassan MSY, et al. Gpt-3.5, gpt-4, or bard? evaluating llms reasoning ability in zero-shot setting and performance boosting through prompts. Natural Language Processing Journal 2023;5:100032
20. Fagbohun O, Harrison RM, Dereventsov A. An empirical categorization of prompting techniques for large language models: A practitioner's guide. arXiv preprint 2024 arXiv:2402.14837
21. Feng Y, Vanam S, Cherukupally M, et al. Investigating code generation performance of chat-gpt with crowdsourcing social data. In: Proceedings of the 47th IEEE Computer Software and Applications Conference, 2023a;pp 1–10
22. Feng Y, Vanam S, Cherukupally M, et al. Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data. In: 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC). IEEE, Torino, Italy, 2023b; pp 876–885, https://doi.org/10.1109/COMPSAC57700.2023.00117
23. Guerreiro NM, Alves DM, Waldendorf J, et al. Hallucinations in large multilingual translation models. Transactions of the Association for Computational Linguistics. 2023;11:1500–17.
24. Hadi MU, Qureshi R, Shah A, et al. A survey on large language models: Applications, challenges, limitations, and practical usage. Authorea Preprints 2023
25. Haleem A, Javaid M, Singh RP. An era of ChatGPT as a significant futuristic support tool: A study on features, abilities, and challenges. BenchCouncil Transactions on Benchmarks Standards

and Evaluations. 2022;2(4): 100089. https://doi.org/10.1016/j.tbench.2023.100089.
26. Hong K, Uh Y, Byun H. ArrowGAN: Learning to generate videos by learning Arrow of Time. Neurocomputing. 2021;438:223–34. https://doi.org/10.1016/j.neucom.2021.01.043.
27. Javaid M, Haleem A, Singh RP. ChatGPT for healthcare services: An emerging stage for an innovative perspective. BenchCouncil Transactions on Benchmarks Standards and Evaluations. 2023;3(1): 100105. https://doi.org/10.1016/j.tbench.2023.100105.
28. Javaid M, Haleem A, Singh RP. A study on ChatGPT for Industry 4.0: Background, Potentials, Challenges, and Eventualities. Journal of Economy and Technology 2023b https://doi.org/10.1016/j.ject.2023.08.001
29. Javaid M, Haleem A, Singh RP, et al. Unlocking the opportunities through ChatGPT Tool towards ameliorating the education system. BenchCouncil Transactions on Benchmarks Standards and Evaluations. 2023;3(2): 100115. https://doi.org/10.1016/j.tbench.2023.100115.
30. Jiang J, Wang F, Shen J, et al. A survey on large language models for code generation. arXiv preprint 2024 arXiv:2406.00515
31. jmorganca Ollama repository. 2023 https://github.com/jmorganca/ollama
32. Jo A. The promise and peril of generative ai. Nature. 2023;614(1):214–6.
33. Katz DM, Bommarito MJ, Gao S, et al. Gpt-4 passes the bar exam. Available at SSRN 4389233 2023
34. Kim TH, Kang JW, Lee MS. AI Chat bot - ChatGPT-4: A new opportunity and challenges in complementary and alternative medicine. Integrative Medicine Research. 2023;12(3): 100977. https://doi.org/10.1016/j.imr.2023.100977.
35. Köbis N, Mossink LD. Artificial intelligence versus Maya Angelou: Experimental evidence that people cannot differentiate AI-generated from human-written poetry. Comput Hum Behav. 2021;114: 106553. https://doi.org/10.1016/j.chb.2020.106553.
36. Kocoń J, Cichecki I, Kaszyca O, et al. ChatGPT: Jack of all trades, master of none. Information Fusion. 2023;99: 101861. https://doi.org/10.1016/j.inffus.2023.101861.
37. Kong ZY, Adi V, Segovia-Hernández JG, et al. Complementary role of large language models in educating undergraduate design of distillation column: Methodology development. Digital Chemical Engineering. 2023;9: 100126. https://doi.org/10.1016/j.dche.2023.100126.
38. Laban P, Kryściński W, Agarwal D, et al. Summedits: Measuring llm ability at factual reasoning through the lens of summarization. In: Proceedings of the 2023 Conference on Empirical Methods in Natural Language Processing, 2023;pp 9662–9676
39. Lajkó M, Csuvik V, Vidács L. Towards JavaScript program repair with generative pre-trained transformer (GPT-2). In: Proceedings of the Third International Workshop on Automated Program Repair. Association for Computing Machinery, New York, NY, USA, APR '22, 2022;pp 61–68, https://doi.org/10.1145/3524459.3527350
40. Li SW, Kemp MW, Logan S, et al. ChatGPT outscored human candidates in a virtual objective structured clinical examination in obstetrics and gynecology. Am J Obstet Gynecol. 2023;229(2):172.e1-172.e12. https://doi.org/10.1016/j.ajog.2023.04.020.
41. Li Y, Sha L, Yan L, et al. Can large language models write reflectively. Computers and Education Artificial Intelligence. 2023;4: 100140.
42. Li Y, Sha L, Yan L, et al. Can large language models write reflectively. Computers and Education Artificial Intelligence. 2023;4: 100140. https://doi.org/10.1016/j.caeai.2023.100140.
43. Martins J, Branco F, Mamede H. Combining Low-Code Development with ChatGPT to Novel No-Code Approaches: A Focus-group Study. Intelligent Systems with Applications 2023;p 200289. https://doi.org/10.1016/j.iswa.2023.200289
44. McIntosh T, Liu T, Susnjak T, et al. Harnessing GPT-4 for generation of cybersecurity GRC policies: A focus on ransomware attack mitigation. Computers & Security. 2023;134: 103424. https://doi.org/10.1016/j.cose.2023.103424.
45. Megahed FM, Chen YJ, Ferris JA, et al. How generative AI models such as ChatGPT can be (mis)used in SPC practice, education, and research? An exploratory study. Quality Engineering 2023; pp 1–29. https://doi.org/10.1080/08982112.2023.2206479
46. Menichetti J, Hillen MA, Papageorgiou A, et al. How can ChatGPT be used to support healthcare communication research? Patient Educ Couns. 2023;115: 107947. https://doi.org/10.1016/j.pec.2023.107947.
47. Ni A, Iyer S, Radev D, et al. LEVER: Learning to Verify Language-to-Code Generation with Execution. In: Proceedings of the 40th International Conference on Machine Learning. PMLR, 2023; pp 26106–26128
48. Nie LY, Gao C, Zhong Z, et al. CoreGen: Contextualized Code Representation Learning for Commit Message Generation. Neurocomputing. 2021;459:97–107. https://doi.org/10.1016/j.neucom.2021.05.039.
49. Nijkamp E, Pang B, Hayashi H, et al. CodeGen: An Open Large Language Model for Code with Multi-Turn Program Synthesis. In: The Eleventh International Conference on Learning Representations 2022
50. Ozkaya I. Application of Large Language Models to Software Engineering Tasks: Opportunities, Risks, and Implications. IEEE Softw. 2023;40(03):4–8. https://doi.org/10.1109/MS.2023.3248401.
51. Ozkaya I. The Next Frontier in Software Development: AI-Augmented Software Development Processes. IEEE Softw. 2023;40(04):4–9. https://doi.org/10.1109/MS.2023.3278056.
52. Park YJ, Kaplan D, Ren Z, et al. Can ChatGPT be used to generate scientific hypotheses? Journal of Materiomics. 2023. https://doi.org/10.1016/j.jmat.2023.08.007.
53. Pothukuchi AS, Kota LV, Mallikarjunaradhya V. Impact of Generative AI on the Software Development Lifecycle (SDLC) 2023
54. Sarkar A. Will Code Remain a Relevant User Interface for End-User Programming with Generative AI Models? In: Proceedings of the 2023 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software. Association for Computing Machinery, New York, NY, USA, Onward! 2023, 2023;pp 153–167, https://doi.org/10.1145/3622758.3622882
55. Sarkis-Onofre R, Catalá-López F, Aromataris E, et al. How to properly use the prisma statement. Syst Rev. 2021;10(1):1–3.
56. Sarsa S, Denny P, Hellas A, et al. Automatic Generation of Programming Exercises and Code Explanations Using Large Language Models. In: Proceedings of the 2022 ACM Conference on International Computing Education Research - Volume 1, ICER '22, vol 1. Association for Computing Machinery, New York, NY, USA, 2022;pp 27–43, https://doi.org/10.1145/3501385.3543957
57. Scanlon M, Breitinger F, Hargreaves C, et al. ChatGPT for digital forensic investigation: The good, the bad, and the unknown. Forensic Science International Digital Investigation. 2023;46: 301609. https://doi.org/10.1016/j.fsidi.2023.301609.
58. Sjödin D, Parida V, Palmié M, et al. How AI capabilities enable business model innovation: Scaling AI through co-evolutionary processes and feedback loops. J Bus Res. 2021;134:574–87. https://doi.org/10.1016/j.jbusres.2021.05.009.
59. Steele JL. To GPT or not GPT? Empowering our students to learn with AI. Computers and Education Artificial Intelligence. 2023;5: 100160. https://doi.org/10.1016/j.caeai.2023.100160.

60. Sun J, Liao QV, Muller M, et al. Investigating Explainability of Generative AI for Code through Scenario-based Design. In: 27th International Conference on Intelligent User Interfaces. Association for Computing Machinery, New York, NY, USA, IUI '22, 2022;pp 212–228, https://doi.org/10.1145/3490099.3511119
61. Svyatkovskiy A, Deng SK, Fu S, et al. IntelliCode compose: Code generation using transformer. In: Proceedings of the 28th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering. Association for Computing Machinery, New York, NY, USA, ESEC/FSE 2020, 2020;pp 1433–1443, https://doi.org/10.1145/3368089.3417058
62. symflower. Devqualityeval. 2024a https://github.com/symflower/eval-dev-quality
63. symflower. Devqualityeval. 2024b. https://symflower.com/en/company/blog/2024/dev-quality-eval-v0.6-o1-preview-is-the-king-of-code-generation-but-is-super-slow-and-expensive/
64. Taulli T. Auto Code Generation. In: Taulli T (ed) Generative AI: How ChatGPT and Other AI Tools Will Revolutionize Business. Apress, Berkeley, CA, 2023;p 127–143, https://doi.org/10.1007/978-1-4842-9367-6_6
65. Topal MO, Bas A, van Heerden I. Exploring transformers in natural language generation: Gpt, bert, and xlnet. arXiv preprint 2021; arXiv:2102.08036
66. Vidanaralage AJ, Dharmaratne AT, Haque S. AI-based multidisciplinary framework to assess the impact of gamified video-based learning through schema and emotion analysis. Computers and Education Artificial Intelligence. 2022;3: 100109. https://doi.org/10.1016/j.caeai.2022.100109.
67. Wang S, Lin B, Sun Z, et al. Two Birds with One Stone: Boosting Code Generation and Code Search via a Generative Adversarial Network. Proceedings of the ACM on Programming Languages 7(OOPSLA2):239:486–239:515. 2023 https://doi.org/10.1145/3622815
68. Wei J, Wang X, Schuurmans D, et al. Chain-of-thought prompting elicits reasoning in large language models. Adv Neural Inf Process Syst. 2022;35:24824–37.
69. Weizenbaum J. Eliza-a computer program for the study of natural language communication between man and machine. Commun ACM. 1966;9(1):36–45.
70. Yilmaz R, Karaoglan Yilmaz FG. The effect of generative artificial intelligence (AI)-based tool use on students' computational thinking skills, programming self-efficacy and motivation. Computers and Education Artificial Intelligence. 2023;4: 100147. https://doi.org/10.1016/j.caeai.2023.100147.
71. Zhao B, Jin W, Del Ser J, et al. ChatAgri: Exploring potentials of ChatGPT on cross-linguistic agricultural text classification. Neurocomputing. 2023;557: 126708. https://doi.org/10.1016/j.neucom.2023.126708.
72. Zhuo TY. Large language models are state-of-the-art evaluators of code generation. arXiv preprint 2023. arXiv:2304.14317