Bibliometric and Content Analysis of Large Language Models Research in Software Engineering: The Potential and Limitation in Software Engineering

Annisa Dwi Damayanti[1], Hamdan Gani[2]*, Feng Zhipeng [3], Helmy Gani[4],
Sitti Zuhriyah[5], Nurani[6], St. Nurhayati Djabir[7], Nur Ilmiyanti Wardani[8]
Department of Environmental Engineering, Faculty of Engineering, Hasanuddin University, Makassar, Indonesia[1]
Department of Machinery Automation System, ATI Makassar Polytechnic, Makassar, Indonesia[2, 7]
School of Culture Creativity and Media, Hangzhou Normal University, Hangzhou, Zhejiang, China [3]
Department of Industrial Hygiene, Faculty of Public Health, Occupational Health and Safety,
Makassar College of Health Sciences, Indonesia[4]
Department of Computer System, Universitas Handayani Makassar, Makassar, Indonesia[5]
Department of Information Systems and Technology, Institut Teknologi dan Bisnis Nobel Indonesia,
Jl. Sultan Alauddin No.212, Makassar and 90221, Indonesia[6]
Department of Informatics Engineering, Universitas Handayani Makassar, Makassar, Indonesia[8]

***Abstract*—Large Language Models (LLM) is a type of artificial neural network that excels at language-related tasks. The advantages and disadvantages of using LLM in software engineering are still being debated, but it is a tool that can be utilized in software engineering. This study aimed to analyze LLM studies in software engineering using bibliometric and content analysis. The study data were retrieved from Web of Science and Scopus. The data were analyzed using two popular bibliometric approaches: bibliometric and content analysis. VOS Viewer and Bibliometrix software were used to conduct the bibliometric analysis. The bibliometric analysis was performed using science mapping and performance analysis approaches. Various bibliometric data, including the most frequently referenced publications, journals, and nations, were evaluated and presented. Then, the synthetic knowledge method was utilized for content analysis. This study examined 235 papers, with 836 authors contributing. The publications were published in 123 different journals. The average number of citations per publication is 1.44. Most publications were published in Proceedings International Conference on Software Engineering and ACM International Conference Proceeding Series, with China and the United States emerging as the leading countries. It was discovered that international collaboration on the issue was inadequate. The most often used keywords in the publications were "software design," "code (symbols)," and "code generation." Following the content analysis, three themes emerged: 1) Integration of LLM into software engineering education, 2) application of LLM in software engineering, and 3) potential and limitation of LLM in software engineering. The results of this study are expected to provide researchers and academics with insights into the current state of LLM in software engineering research, allowing them to develop future conclusions.**

***Keywords*—Large Language Models; LLM; software engineering; bibliometric; content analysis**

I. Introduction

Coupled with Generative Pre-trained Transformers, Large Language Models substantially advance natural language processing. ChatGPT, a cutting-edge conversational language model noted for its user-friendly interface, has attracted significant interest due to its advanced capacity to deliver human-like responses in various conversational scenarios. OpenAI has created an impressive conversational artificial intelligence (AI)-based language model known as the Chat Generative Pre-Trained Transformer (ChatGPT). On November 30, 2022, OpenAI released the ChatGPT GPT 3.5 series for free, followed by the premium version, GPT-4, on March 14, 2023 [1]. Additionally, other well-known LLMs include Google's Gemini, Microsoft Copilot, Meta's LLaMA, Anthropic's Claude, and Mistral AI's models [2].

The integration of this sophisticated technology in software engineering remains a subject of debate among stakeholders. Nevertheless, it holds potential for incorporation into the software engineering workflow [3]. Rahmaniar [4], suggests that more research should be conducted on LLM's effects and potential contributions to software engineering tasks such as code generation, bug fixing, and design decisions. LLMs have the potential to transform the software engineering sector by impacting various software development tasks, including code evolution and software testing [5].

Bibliometric research in software engineering is becoming increasingly popular [6]. According to their definition, bibliometrics is the study of a research issue using mathematical and statistical techniques based on bibliographic sources. The method of gathering and analyzing bibliometric data on a large scale allows bibliometric analysis to provide comprehensive details about the evolving patterns and intellectual structure of a research topic or discipline [7].

Bibliometric analysis comprises two techniques: scientific mapping and performance analysis [7]. Performance analysis examines how well individuals, organizations, and nations perform regarding research and publications [8]. Scientific mapping helps study scientific domains by revealing their structure and dynamics [9]. This study chose bibliometric analysis for our investigation because it can efficiently detect patterns and trends in the literature, highlight necessary studies and writers, and identify trends and topics for future research.

While previous bibliometric studies have examined the use of Large Language Models (LLMs) in fields such as public health, education, social sciences, and medicine, no such analysis has been conducted specifically in software engineering [10] to [16]. Existing research indicates that the application of LLMs in software engineering is still in its early stages. A detailed bibliometric analysis in this field is needed to identify key contributors, trends, and geographic activity, as well as to better understand the evolving potential and limitations of LLMs in software engineering.

The goal of this paper is to provide a bibliometric analysis of LLM studies in software engineering. The findings of our analysis provide information on the following issues:

1) What is the monthly distribution of publications on LLM?

2) What are the top five most cited publications in the LLM field?

3) Who are the top five most cited authors in the LLM field?

4) What are the dynamics of publications on LLM in the literature (journals and countries)?

5) What are the keywords most commonly used in publications on LLM?

6) Which themes emerged after the content analysis of LLM research?

II. Literature Review

Previous research has explored bibliometric analyses on the usage of LLM in public health [10], education [11], social science [12], and medicine [13]. However, no bibliometric analysis of LLM has been performed in software engineering. A scoping study on applying an LLM in software engineering revealed that LLM research was in its early phases [14], [15]. Marques et al. [16] investigated the role of LLM in software requirements engineering, an essential aspect of software engineering. They discovered that the possibilities and limitations of LLM in software engineering are still developing and in their early phases. A detailed bibliometric analysis of LLM in software engineering can provide academics and stakeholders with an overview of the study. This research can assist in identifying the field's most prolific authors, countries, and scientific trends. In this case, bibliometric analysis could lead us to explore the potential and limitations of LLM in the software engineering field.

III. Method

A. Search Strategy

The flowchart of the research methodology is shown in Fig. 1. Web of Science and Scopus are the two most commonly utilized databases in bibliometric studies [17]. Data sources for this study included Scopus (Elsevier; https://www.scopus.com/) and Web of Science (Clarivate; https://www.webofscience.com/wos). The search strategy was designed as follows: "(TITLE-ABS-KEY ("Large Language Model" OR ChatGPT OR LLM OR Chatbot OR OpenAI OR Gemini OR Copilot)) AND (TITLE-ABS-KEY ("Software Engineering" OR "Software Development"))." The search was carried out on July 17, 2024. The search criteria were "article title, abstract, keywords" and publications on LLM in software engineering. There were no exclusion criteria. The search retrieved 529 studies from the two databases. After deleting 80 duplicate publications, 214 were reviewed using the inclusion criteria, resulting in 235 papers for bibliometric analysis.

B. Bibliometric Methodology

Bibliometric analysis examines the performance of research elements, such as articles, authors, journals, keywords, and nations, and visualizes their intellectual, conceptual, and social structures through mapping methodologies [18]. In our work, the bibliometric methodology included performance analysis and scientific mapping techniques. The parameters for performance analysis were the number of articles and citations. Co-occurrence analysis was employed for scientific mapping [7].

Before data analysis, Scopus and Web of Science data were automatically combined using the RStudio IDE. The file was downloaded in BibTex format from Scopus and Web of Science, then updated in RStudio using R script code to produce a "database.csv" file (Source Code: Appendix). The data for this investigation was analyzed using VOS viewer version 1.6.20 (Leiden University, Netherlands, https://www.vosviewer.com) and Bibliometrix version 4.3.0 (University of Naples Federico II, https://www.bibliometrix.org). The VOS viewer allows users to observe things as well as their connections. "Items" are noteworthy objects, such as publications or researchers, while "links" represent the connections between these items. The VOS viewer establishes a correlation or relationship between two things. The stronger the link between the components, the greater the Total link strength (TLS), represented by a positive numerical value. On a map, things can be arranged into groups that form clusters. The weights applied to each item in network visualization represents its relative importance. The size of the circles or labels representing the object is strongly related to its weight; the more significant the circle or label representing an item, the heavier the item. This strategy simplifies understanding of the picture's components' relevance and relationships [8].

Bibliometrix is an open-source bibliometric software developed in R (The Comprehensive R Archive Network,

https://cran.r-project.org) [19]. After installing the Bibliometrix R package, the Bibliometrix web interface was accessible with the executed code "bibliometrix::biblioshiny ()". Bibliometrix was used to investigate publishing data (total citations, average citations, etc.) and international collaboration between countries.

C. Content Analysis

Bibliometric analysis is an accurate method for recognizing the many information clusters that may appear in the literature. Klarin [18] developed the guideline for the knowledge synthesis approach used in content analysis. The approach consists of the following steps:

1) Research publications on LLM and Software Engineering are compiled.

2) Publications are categorized into themes depending on author keywords. Co-occurrence keyword analysis contains the results of this stage. Then, this study analyzes and characterizes the relationships between codes within each cluster in Section Co-occurrence keyword analysis."

3) Identify categories and assign theme names to clusters. Content Analysis shows the results of this stage.

4) The qualitative analysis generates a list of topics as output. Table III shows the results of all processes.

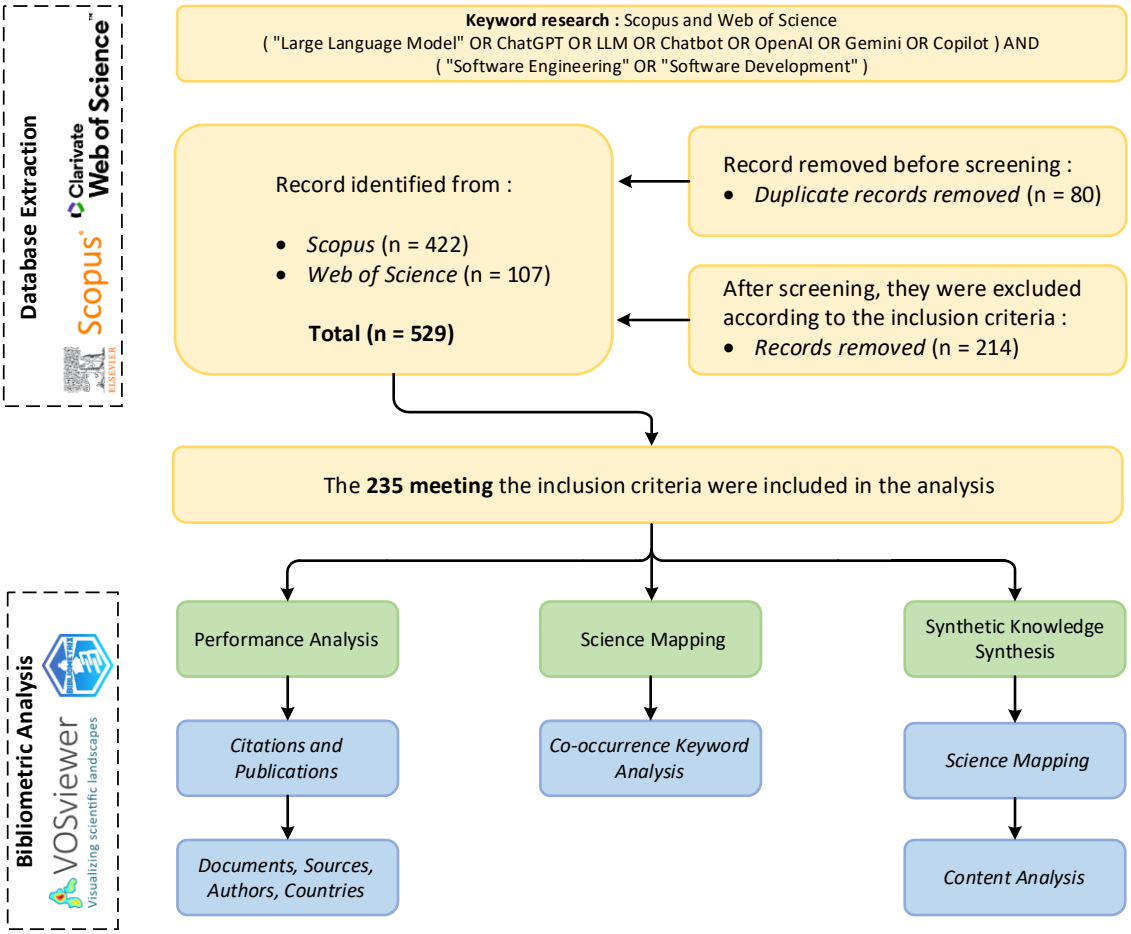

Fig. 1. Research methodology.

IV. Results

A. Publication Characteristics

Since ChatGPT was released on November 30, 2022, the publication distributions by month shown in Fig. 2, include publications published between January 2023 and June 2024. The analysis included 235 publications. The majority of the publications were published in April 2024. The publications comprised 51 articles, 138 conference papers, 39 proceeding papers, 3 reviews, 3 book chapters, and 1 lecture note. These papers, which had 836 authors, appeared in 123 different journals. The average number of citations was 1.44.

This analysis discovered that most articles were not research articles but conference and proceeding papers. In particular, scoping review research on LLM conducted on software engineering found that nearly one-quarter of the publications were "article" studies. This indicates that the research is still in its early stages. LLM, such as ChatGPT, is a new AI technology in software engineering [16].

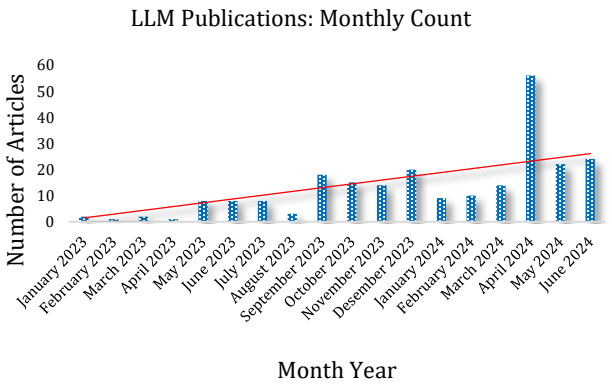

Fig. 2. Distribution of publications by months.

Since publications are published almost monthly, it is reasonable to expect future growth in LLM and software engineering studies.

B. Top 10 Most Cited Publications and Authors

The most cited publication (Table I) discusses how ChatGPT can be used in software engineering to translate, create, and autocomplete code [20]. The second most cited work investigates the use of few-shot training with the GPT Codex model, demonstrating that it outperforms state-of-the-art models in code summarization while exploiting limited, project-specific data, emphasizing its importance in software engineering (Sobania et al., 2022). The research discovered that Copilot boosts software development productivity as measured by code-generated lines. The third most referenced work investigates the application of LLM in software engineering education and discusses improving software engineering education by personalizing learning experiences. They also emphasize the importance of modifying software engineering programs to match evolving software engineer profiles [21].

TABLE I. MOST CITED PUBLICATIONS AND AUTHORS

No	Title	Publication Type	Authors	Journal	Number of Citation (Scopus)	Google Scholar Citation Count
1.	The Programmer's Assistant: Conversational Interaction with A Large Language Model for Software Development	Conference Paper	(Ross et al., 2023)	International Conference on Intelligent User Interfaces, Proceedings IUI	72	176
2.	Few-Shot Training LLMs for Project-Specific Code-Summarization	Conference Paper	[35]	ACM International Conference Proceeding Series	27	121
3.	How ChatGPT Will Change Software Engineering Education	Proceedings Paper	[21]	Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education, ITICSE 2023, Vol 1	25	68
4.	GitHub Copilot AI Pair Programmer: Asset or Liability?	Article	[36]	Journal of Systems and Software	24	241
5.	Generative AI for Software Practitioners	Article	[37]	IEEE Software	20	97
6.	Towards Human-Bot Collaborative Software Architecting with ChatGPT	Proceedings Paper	[22]	27th International Conference on Evaluation and Assessment in Software Engineering, EASE 2023	12	110
7.	Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data	Proceedings Paper	[38]	2023 IEEE 47th Annual Computers, Software, and Applications Conference, COMPSAC	10	84
8.	Exploring The Implications of OpenAI Codex on Education for Industry 4.0	Conference Paper	[39]	Studies in Computational Intelligence	9	16
9.	Natural Language Generation and Understanding of Big Code For AI-Assisted Programming: A Review	Review	[40]	Entropy	9	40
10.	Large Language Model Assisted Software Engineering: Prospects, Challenges, and A Case Study	Conference Paper	[41]	Lecture Notes in Computer Science	6	48

The sixth most cited publication investigates how Software Development Bots, specifically ChatGPT, might help architecture-centric software engineering (ACSE) processes. It explores the problems of ACSE and proposes using ChatGPT to combine human experience with AI-powered decision support. The paper also describes a case study in which a novice architect collaborated with ChatGPT to design a service-based software system. The authors suggest future research to collect more empirical evidence on the productivity and socio-technical aspects of using ChatGPT in software architecture [22].

The most cited publications generally discuss ChatGPT's use in software engineering for tasks such as code translation, generation, and autocomplete. Another study shows that GPT Codex outperforms state-of-the-art models in code summarization using project-specific data via few-shot training. Generative AI, such as ChatGPT, is also being considered to improve software engineering education and change curriculum. A study further emphasizes ChatGPT's importance in architecture-centric software engineering (ACSE) by supporting new architects and recommends additional research on productivity and socio-technical issues.

C. Most Productive Journals

The most productive journals in terms of the number of publications in the field of LLM are Proceedings - International Conference on Software Engineering (n = 27), ACM International Conference Proceeding Series (n = 20), Lecture Notes in Computer Science (n = 10), Proceedings - 2023 38TH IEEE/ACM International Conference on Automated Software Engineering, ASE 2023 (n = 9) and IEEE Transactions on Software Engineering (n = 6), respectively (Table II).

In the remaining journals, one or two articles were published. The most productive journals in terms of the number of citations are Proceedings - International Conference on Software Engineering (n = 54), ACM International Conference Proceeding Series (n = 33), and Lecture Notes in Computer Science (n = 15), respectively.

TABLE II. THE MOST PRODUCTIVE SOURCES

No	Journal	N*)	Total Citation	H-Index	G-Index
1	Proceedings - International Conference on Software Engineering	27	9	1	2
2	ACM International Conference Proceeding Series	20	29	1	5
3	Lecture Notes in Computer Science	10	12	2	3
4	Proceedings - 2023 38TH IEEE/ACM International Conference on Automated Software Engineering, ASE 2023	9	8	2	2
5	IEEE Transactions on Software Engineering	6	8	2	2
6	Proceedings - 2023 IEEE International Conference on Software Maintenance and Evolution, ICSME 2023	5	3	1	1
7	IEEE Software	4	22	2	4
8	Journal of Systems and Software	4	24	1	4
9	Lecture Notes in Business Information Processing	3	5	2	2
10	Automated Software Engineering	3	1	1	1

Note. (*) Sources that published at least two publications were listed.

The fact that the Proceedings - International Conference on Software Engineering has the most citations implies that research in this discipline is well-received in academic circles. The article "The Programmer's Assistant: Conversational Interaction with A Large Language Model for Software Development," published in the International Conference on Intelligent User Interfaces, Proceedings IUI [20], drew attention due to the number of citations it received in the fields of LLM and software engineering.

D. The Most Productive Countries and International Cooperation

The research articles were created by authors from 40 different countries (Fig. 3). The top five most productive countries in terms of number of publications were China (n = 29), the United States (n = 28), Germany (n = 16), Canada (n = 11), and Brazil (n = 6). The top five nations by number of citations were Canada (n = 74), the United States (n = 70), Germany (n = 57), China (n = 14), and Finland (n = 14). Consistent with our findings, other studies have identified China and the United States as significant countries in LLM research in different disciplines [23]–[25]. These countries' position as pioneers in LLM and software engineering research reflects their significant investments in these fields.

The review of international collaborations demonstrated that the country that collaborated most was China (Fig. 4). China collaborated with Australia (n = 3), Finland (n = 2), Germany (n = 2), and the United Kingdom (n = 2). Furthermore, there were collaborations between Germany and Finland (n = 2), Germany and the United Kingdom (n = 2), the United States and China (n = 2), the United States and the United Kingdom (n = 2), Australia and Finland (n = 1), and Australia and Singapore (n = 1). In the future, increased international collaboration and generative AI applications like ChatGPT may allow for the development of creative and comprehensive methodologies in software engineering. In addition, guidelines and policies for artificial intelligence and software engineering can be produced in collaboration with other countries.

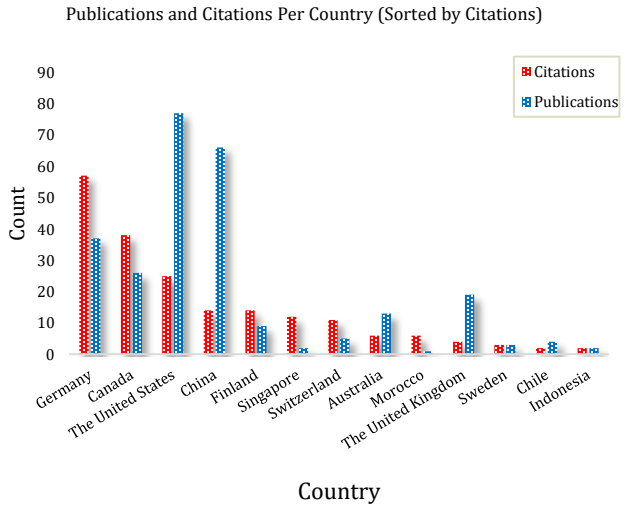

Fig. 3. The Number of publications and citations by country. Note. Countries with at least two publications are presented.

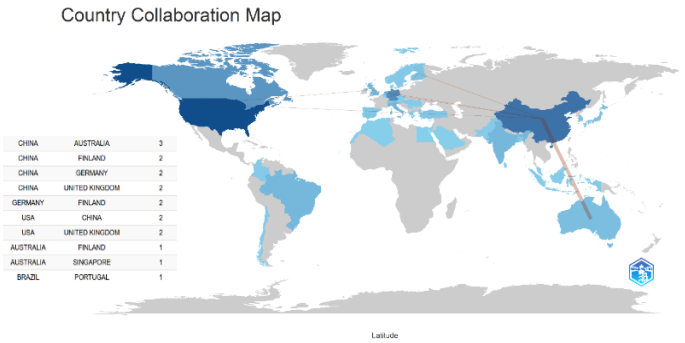

Fig. 4. Country collaboration map. Note. Dark blue indicates more frequent international cooperation. The thickness of the red line between the two countries shows the extent of cooperation.

E. Co-Occurrence Keyword Analysis

This study identified 1804 keywords (Table III). The following keywords appear at least ten times: software design (n = 158), code (symbols) (n = 68), code generation (n = 40), software testing (n = 34), learning system (n = 32), and artificial intelligence (n = 25). According to Ozkan et al. (Ozkan et al., 2024) and Mesa Fernández et al. (2022), the most commonly used words in software engineering are "software design," "code (symbols)," and "code generation." The frequent use of these words indicates that LLM-based technologies are becoming more popular in the software engineering, especially in code development.

Fig. 5 and Table III shows a visualization of the co-occurrence analysis using the VOS viewer. In the green cluster, the keywords "software design" (TLS = 277) and "code generation" (TLS = 105) are prominent. The emphasis is on the impact and prospective applications of LLM in software engineering. The red cluster contains the keywords "artificial intelligence" (TLS = 62), "generative AI" (TLS = 52), "engineering education" (TLS = 72), "students" (TLS = 82), and "software engineering education" (TLS = 54). In this cluster, the focus is on using LLM in software engineering education. In the yellow cluster, the keywords are "engineering task" (TLS = 45), "prompt engineering" (TLS = 35), "life cycle" (TLS = 39), "modeling languages" (TLS = 31), and "requirement engineering" (TLS = 19) and it is emphasized that the practical application of LLM on software life-cycle development are an essential research area. In the blue cluster, the focus is on the impact of LLM on automation software development tasks, whereas in the purple cluster, the focus is on using LLM in "code (symbols)" (TLS = 153), "quality control" (TLS = 46), "task analysis" (TLS = 37), A"code review" (TLS = 24), and "code quality" (TLS = 20). This analysis visualizes how the interaction between the software engineering field and AI intersects with different aspects and how these terms are positioned together in academic literature.

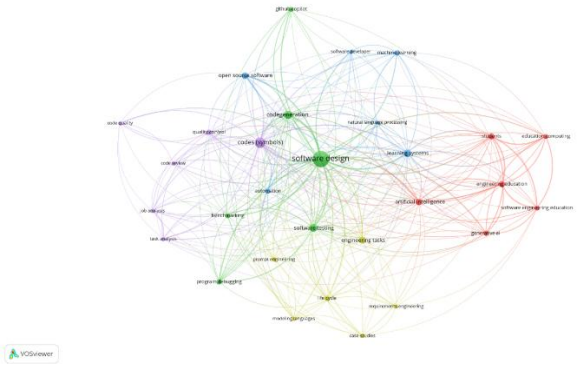

Fig. 5. Visualization of keywords. Note. The analysis is set to the minimum number of keyword occurrences (minimum 10). The network consists of 30 items, 5 clusters, 269 links, and 875 total link strengths.

TABLE III. CO-OCCURRENCE ANALYSIS

Cluster no. and color	Cluster theme	The number of items	Code (Keywords)	Explanation
1 Red	Software Engineering Education	6	Artificial Intelligence, Education Computing, Engineering Education, Generative AI, Software Engineering Education, Students	Researchers delve into educational aspects related to LLM and software engineering.
2 Green	Software Development Tools and Practices	6	Benchmarking, Code Generation, GitHub Copilot, Program Debugging, Software Design, Software Testing	Researchers explore topics like the application of LLM in practical software development scenarios.
3 Blue	Automation Software Development Task	6	Automation, Learning System Machine Learning, Natural Language Processing, Open-Source Software, Software Developer	The emphasis of ChatGPT's impact on automation, software development task
4 Yellow	Practical Application of LLM in Software Engineering	6	Case Studies, Engineering Tasks, Life Cycle, Modeling Languages, Prompt Engineering, Requirements Engineering	The cluster deals with various aspects of software engineering, from requirements engineering and software modeling to security checks.
5 Purple	Quality Control	6	Code (symbols), Code Quality, Code Review, Coding Standards, Job Analysis, Quality Control, Task Analysis	The cluster revolves around the various aspects of quality control and evaluation tasks within the software engineering domain.

Note. The network has 30 items, 5 clusters, 269 links, and a TLS value 875. *Index keywords used by the paper.

F. Content Analysis

Content analysis can assist prospective academics in identifying essential knowledge gaps in the literature [18]. During the content analysis, five key themes emerged.

Theme 1: Integration of LLM into Software Engineering Education

The first theme focuses on the discussions regarding the use of LLM in educational aspect of software engineering [15], [21], [34], [26]–[33].

Subtheme 1: Potential and threats of LLM in software engineering education.

Using LLMs like ChatGPT in software engineering education has generated debate, highlighting threats and opportunities. The integration of ChatGPT into software engineering education can provide various benefits. ChatGPT can give personalized feedback and enhance individualized student learning experiences [21].

It has advantages, such as how ChatGPT can change how software engineering is taught, making it more practical and relevant to real-life scenarios. Students use ChatGPT to do practical software engineering tasks like collecting user stories, creating use case and class diagrams, and formulating sequence diagrams, which helps them understand real-world applications of their learning [30]. It also has advantages such as Automated Programming Assessment Systems (APASs) as AI-tutors [27] and can aid educators in developing curriculum and preparing course material [29]. Additionally, it can be used in real-time problem solving, collaboration and peer learning, virtual mentoring, generating creative and novel ideas, and research and exploring [15].

Despite the potential benefits, integrating LLMs, such as ChatGPT, into software engineering education raises several concerns and possible limitations. An empirical study with 182 participants in a first-year programming course found no significant difference in performance between students using ChatGPT and those not using it, suggesting that ChatGPT can be safely integrated into education with proper measures [34]. Recent studies in LLMs, including ChatGPT and Copilot, have led to their integration into software development education. An experiment with 32 participants examined LLM use and its correlation with student performance, revealing a negative impact on grades when overused for essential tasks, emphasizing the need for balanced integration of LLM tools in education [31]. Petrovska et al. [32] focused on creatively developing assessments that encourage learners to critically evaluate ChatGPT's output, helping them understand the subject material without the risk of the AI tools "doing the homework." Additionally, Brennan and Lesage [33] evaluated the OpenAI Codex code completion in industry 4.0-oriented engineering programs. They reported that while Codex assisted with simple code completions, students still needed a solid understanding of software development principles, underscoring the importance of foundational knowledge even when using these advanced AI tools.

Overall, integrating LLMs like ChatGPT in software engineering education offers significant advantages, including personalized learning, practical application of concepts, and support for curriculum development. However, it also presents challenges, such as potential negative impacts on student performance if overused and the necessity for students to have solid foundational knowledge despite AI assistance. Students ' critical evaluation of AI-generated content is essential to ensure they truly understand the material. Balancing the use of these tools with traditional learning methods is crucial for maximizing their benefits in educational settings.

Subtheme 2: Future Directions and Adaptation of LLM in Education.

The roadmap for integrating LLMs into software engineering education includes adapting curricula to provide AI literacy, ensuring academic integrity, and reducing academic misconduct. This highlights the necessity for ongoing policy adaptation to technological advancements, marking a critical step toward responsibly integrating ChatGPT in education. Future directions for integrating ChatGPT and similar LLM tools in software engineering education include creating interactive and immersive learning platforms, adopting holistic educational approaches, and continuously evaluating the impact of these tools to optimize their integration and maximize educational benefits [15], [28]–[30].

In summary, integrating ChatGPT into software engineering education requires adapting curricula to include AI literacy, maintaining academic integrity, and preventing misconduct through evolving policies. Future directions involve creating interactive learning platforms, adopting holistic educational approaches, and continuously evaluating the impact of these tools to ensure they provide maximum educational benefits. These steps are essential for responsibly incorporating ChatGPT and similar technologies into education.

Theme 2: Software Development Tools and Practices

This theme delves into the utilization of LLMs for developing and maintaining software quality through advanced code generation and software testing practices. It highlights the potential and limitations of LLMs application in software engineering.

Subtheme 1: Code generation performance and evaluation

Researchers from multiple countries have examined the performance of code generated by LLM. Wang and Chen (Wang & Chen, 2023) noted that LLM-powered code generation has sparked considerable academic interest. An empirical study by Z. Liu et al. [42] assessed ChatGPT's code generation capabilities across five programming languages, focusing on correctness, complexity, and security. Their findings revealed that ChatGPT effectively generates accurate code for issues predating 2021 but encounters difficulties with more recent problems. Similarly, M. Liu et al. [43] conducted a case study using GPT-4 to generate safety critical software code. They explored various methods, including overall requirements, specific requirements, and augmented prompts, concluding that GPT-4 can autonomously generate safety-critical software code suitable for practical engineering applications.

Despite the promising results of LLM's performance in code generation, there is still a debate about their reliability, necessitating more study on evaluation studies [44]. Rodriguez-Cardenas et al. [45], examined LLM-generated code in different scenarios, emphasizing the need for comprehensive evaluation metrics to accurately measure the effectiveness of LLMs in producing reliable and functional code. Preliminary studies have been conducted to this end. Yeo et al. [46], introduced a framework for evaluating LLM-generated code using a metric based on test case pass rates. Similarly, Aillon et al. [47], suggested several metrics for assessing LLM-generated code, including code quality, solution quality, response time, and comparisons with human-generated code.

Other studies have identified significant issues in evaluating LLM-generated code. Mastropaolo et al. [48], assessed the robustness of code generated by GitHub Copilot, highlighting inconsistencies in Copilot's performance. They found that different but similar prompts often resulted in varied outputs, undermining code quality. Zhong and Wang [49] also evaluated the robustness and reliability of LLM-generated code, reporting several limitations. For example, 62% of the code generated by GPT-4 misused APIs, potentially leading to resource leaks, crashes, or unpredictable behavior.

Furthermore, while the generated code can run, it is not always reliable or robust enough for real-world applications. Tests often focus on small or straightforward tasks, which do not reflect the complexity of real-world software development challenges [50]. In addition, Mbaka [51] investigated ChatGPT's effectiveness in validating security threats. The study found that ChatGPT is unreliable in distinguishing real threats from fake ones.

In summary, while LLMs like ChatGPT show potential in code generation, significant issues related to reliability and robustness remain. Comprehensive evaluation metrics and further studies are needed to improve the effectiveness of LLMs in practical software development scenarios.

Subtheme 2: Software testing

This cluster examines the application of LLM in software testing, exploring their potential to innovate and enhance traditional testing methodologies. The research encompasses various aspects of software testing, from unit test generation to exploratory testing, highlighting the transformative potential of LLMs in improving these processes.

Tsigkanos et al. [52] explored the use of LLMs for metamorphic testing in addressing oracle problems within scientific software testing. Scientific software typically handles vast data sets, making manual extraction of essential variables for testing challenging. They developed a method using LLMs to extract these variables from user manuals automatically and compared the LLM-extracted variables with those identified by human experts, finding the LLM method effective. However, despite automating metamorphic testing and reducing human intervention, the approach may still face challenges in managing the vast and varied input-output spaces characteristic of scientific software. Schafer et al. [53] reported the effectiveness of using LLMs to create unit tests, introducing a tool called Test Pilot that generates diverse tests without needing extra training, outperforming existing methods. However, while the tool works well with certain LLMs and specific prompt information, it may not handle more complex or unusual cases in software testing effectively. Thus, further work is needed to enhance the tool's reliability and versatility across different testing scenarios. Tang et al. (Tang et al., 2024) systematically compared unit test suites generated by ChatGPT and EvoSuite, focusing on key factors such as correctness, readability, code coverage, and bug detection capability. Their findings indicate that ChatGPT is a promising tool for generating unit tests in software engineering. Yet, they also identify significant limitations, including reliance on a single LLM model, issues with generalization, and the necessity for ongoing research to improve the reliability and effectiveness of LLM-generated test cases.

Additionally, El Haji et al. [54], assessed GitHub Copilot's ability to generate unit tests automatically. Their experimental study revealed that while GitHub Copilot shows potential, a significant portion of its generated tests fail or are not helpful, particularly without an existing test framework. They concluded that including comments in the code could improve the tool's performance, suggesting that clear documentation might enhance results. This study highlights limitations related to high failure rates, usability issues, dependence on existing test suites, and the proprietary nature of the tool. Similarly, Mehmood et al. [55], compared GitHub Copilot-generated test cases with test cases created by humans, finding that while Copilot shows promise, it has limitations, such as restrictions on the range of scenarios for testing and potential prompt biases. Further research is necessary to understand its capabilities and best uses fully. Despite promising to generate test cases comparable to those created manually, Copilot has limitations in scope, reliance on the prompt quality, range of generated test cases, and the need for more extensive research across a broader range of software development tasks.

Moreover, Copche et al. [56] developed a chatbot called BotExpTest to assist human testers during exploratory testing. This study suggests that integrating chatbots into testing can improve bug detection efficiency and effectiveness. However, further research with larger and more varied sample sizes, extended testing durations, and comparisons across different testing environments and types is needed to understand its capabilities and limitations fully. Finally, LLMs have been utilized for bug detection and fixing [57]–[59]. A significant challenge with LLMs is the need for additional steps to make their output more helpful. After generating results, extra time is required to fix mistakes and provide more detailed instructions, which can be time-consuming. LLMs also struggle to fully understand the context of the code they are testing without clear explanations, leading to missed details and bugs.

In summary, LLMs have shown great potential in various aspects of software testing, including automatic test generation, exploratory testing, and bug detection. While these tools can significantly enhance testing processes, they often require additional steps to fix errors and provide detailed prompts, which can be time-consuming. LLMs struggle with understanding code context without clear descriptions, leading to missed details and bugs. LLM-generated tests have high failure rates and usability issues, especially without existing frameworks or clear documentation. Additionally, LLMs may not handle complex or unusual cases well and are limited by prompt biases and testing scenario restrictions. Despite their potential, LLMs need further refinement and research to improve their reliability and effectiveness in practical software engineering tasks.

Theme 3: Automation Software Development Task

This theme focuses on the integration of LLM in automating software engineering tasks. The key findings highlight significant advancements in this area, emphasizing the transformative impact of LLM automation on traditional software engineering processes.

Rathnayake et al. [60], explores automated technical interviews using advanced chatbot technologies. In the realm of software engineering, automating technical interviews facilitates the assessment of candidates' technical competencies during recruitment. However, challenges remain, such as accurately evaluating technical skills, potential biases in assessing candidate psychology, and ensuring all system components function cohesively. Therefore, further work is needed to enhance the reliability and effectiveness of these automated interview systems. Chen et al. [61] propose an LLM approach for converting written problem descriptions in natural language into domain models, which is typically time-consuming and requires significant expertise. Their findings indicate that while LLMs promise to automate domain modeling, they often miss essential details and do not always follow best practices. Consequently, additional research is necessary to make these tools practical and reliable for everyday use in software engineering. Martins et al. [62], apply LLMs to automatically analyze code, demonstrating their practical application in maintaining high coding standards and improving overall software maintainability. However, the effectiveness of LLMs heavily depends on the quality of input data and the implementation process. Asare et al. [63] examine the security implications of code generated by GitHub Copilot. They conclude that although Copilot performs differently across various vulnerability types, it is not worse than human developers at introducing vulnerabilities. Nonetheless, the study identifies limitations in creating secure code, with Copilot sometimes repeating old coding mistakes, thereby making software vulnerable to attacks. Specifically, Copilot suggests code with the same security flaws about 33% of the time. Its performance varies depending on the type of vulnerability, and it tends to struggle more with older issues. Thus, while Copilot can be helpful, it is not always reliable for generating safe code, necessitating careful instructions from developers to avoid security issues.

Additionally, Wuisang [64] evaluate the effectiveness of ChatGPT for automated bug fixing in Python. Their study highlights ChatGPT's potential as an effective tool for improving code quality and reducing the need for human intervention in bug fixes. They tested 40 different bugs, finding that ChatGPT could correctly fix 30 of them. Although this demonstrates ChatGPT's capability, the failure to fix 10 bugs indicates room for improvement. Despite outperforming other tools like standard bug-fixing methods and Codex, further enhancements are required to ensure reliability in fixing all bugs.

In summary, the research underscores the transformative potential of LLMs in automating various software engineering tasks, including technical interviews, domain modeling, code analysis, and bug fixing. However, challenges remain regarding the reliability and effectiveness of LLMs in these applications. Further refinement and research are essential to fully realizing their potential in practical software development tasks.

Theme 4: Practical Application of LLM in Software Engineering.

Subtheme 1: LLM in Requirements Engineering.

Research in this cluster focuses on integrating LLM in requirements engineering. Jain et al. [65] present a novel approach for summarizing requirements from obligations in software engineering contracts using LLM. This method leverages prompt engineering principles to guide GPT-3 in generating training and ground truth summaries, which are then used to train Natural Language Generation (NLG) models for contract text summarization. Despite its promise, the method heavily relies on the effectiveness of prompt engineering and the performance of NLG models. Consequently, there is a risk that essential details might be missed or not accurately captured, making LLM-generated summaries potentially unreliable. Therefore, analysts must review the original contracts carefully to ensure accuracy.

Spoletini and Ferrari [66] explore integrating automatic formal requirements engineering techniques with LLMs to enhance code generation reliability. These techniques are typically employed in developing complex systems to ensure adherence to specific standards. By combining formal methods with LLM models, the researchers aim to improve the accuracy and reliability of LLM-generated code. However, a significant challenge lies in ensuring that the code generated by LLMs consistently meets these high standards.

Subtheme 2: LLM in Software Modeling and Design.

This theme explores the application of LLMs in software modeling and Unified Modeling Language (UML). Ren et al. [67] investigate the role of chatbots in UML modeling, concluding that while chatbots are valuable for building class diagrams and lay a foundation for further research on their applicability in software engineering diagramming, they are not yet fully capable of capturing all necessary details. This indicates a need for further development and research to enhance their completeness and effectiveness in diagramming tasks. Camara et al. [68] highlight LLMs' practical applications and educational benefits, noting their use in enterprise and software modeling processes. However, they emphasize that educators must rethink how they design and administer assessments, as integrating LLMs requires significant adjustments. Chen and Zacharias (Chen & Zacharias, 2024) propose using generative AI to develop software design principles that assist software developers. Their research identifies fundamental issues with generative AI in software development, including usability problems, data privacy concerns, hallucinations, and a lack of transparency. De Vito et al. [69] introduce ECHO, a novel approach to enhancing the quality of UML use cases using LLMs. ECHO employs a co-prompt engineering technique and an interactive process with the LLM to improve use cases based on practitioner feedback. Despite its potential, ECHO faces challenges such as the need for substantial effort to develop effective prompts and ensure iterative improvements. Additionally, their experiment showed that while ECHO could improve use case quality, further refinement, and validation are necessary to ensure consistent and reliable outcomes across diverse scenarios.

Melo [70] proposes the design of context-based adaptive interactions between software developers and chatbots to foster solutions and knowledge support. Although the proposed method shows potential, it remains difficult for chatbots to

understand and adapt to the specific contexts of software development. The study highlights the need for more research to understand developers' expectations and improve the interaction between developers and chatbots. Finally, Petrovic [71] examines the integration of ChatGPT into software development practices, focusing on automated security checks and real-time feedback to enhance software security and reliability during the design phase. However, the study identifies challenges, such as the need to refine and process automated results from ChatGPT to be useful for developers and system administrators. Additionally, the evaluation was limited to specific tools, which may affect the generalizability of the findings to other tools or environments.

In summary, the research highlights that LLMs can significantly enhance various aspects of software engineering, from requirements engineering and software modeling to security checks. They provide practical tools for improving quality in software development processes. However, specialized models and further research are necessary to fully realize the potential of LLMs in practical applications and address existing limitations.

Theme 5: Quality Control

This theme delves into utilizing LLMs for quality control and evaluation tasks within the software engineering domain. Lu et al. [72] introduce Llama-reviewer, a model designed to automate code reviews in software development. The model is noted for its efficiency, using fewer resources than traditional models, and the findings indicate promising results even with less training. However, the study acknowledges that the limited training epochs might restrict the model's capability to manage more complex or diverse code review tasks. Ronanki et al. (Ronanki et al., 2024) investigate the application of ChatGPT for evaluating the quality of user stories in Agile software development. The results show that ChatGPT's assessments align well with human evaluations, but the study highlights the challenge of ensuring the trustworthiness of ChatGPT's outputs. Tufano et al. [73] compare a deep learning model and ChatGPT in mimicking developers' tasks during code reviews, such as adding comments on code changes or fixing code based on comments. They found that ChatGPT struggled to comment on code as effectively as human reviewers. This research underscores the need for more specialized studies to enhance code review automation, as general models like ChatGPT cannot fully replicate human reviewers' capabilities, especially in tasks like code review.

Furthermore, Pantelimon and Posedaru [74] explore how ChatGPT can generate code snippets, templates, and functions from natural language input, aiding in bridging the gap between technical and non-technical team members in software development. While this tool helps developers quickly find and fix bugs, enhancing the accuracy of automated code review and testing, concerns about potential over-reliance on ChatGPT and the limitations of its ability to comprehend intricate technical concepts are noted. In addition, Martins et al. [62] present an automated GitHub bot using LLMs to enforce SOLID principles during code reviews. This bot provides immediate feedback, improving code quality, particularly for new programmers, and integrates seamlessly into GitHub. However, they also state that the tool faces many challenges in handling various code review scenarios.

In summary, while these studies illustrate the potential of LLMs in software engineering, they also highlight the need for further research to address limitations related to model robustness, handling complex tasks, reducing biases, and improving integration and usability in real-world software development environments. Future work should focus on developing more specialized LLM models tailored to specific tasks like code reviews, enhancing the robustness and adaptability of these models through more extensive and varied datasets, and refining approaches like co-prompt engineering for better accuracy. Additionally, efforts should be made to mitigate over-reliance on automation, reduce biases in training data, and improve the interaction between developers and LLM tools. Expanding testing in diverse environments, integrating advanced LLM models, and developing comprehensive training for non-experts to use these AI tools effectively are crucial steps for future research.

V. Discussion

The current study acknowledges several limitations inherent in the bibliometric analysis approach. First, while Scopus and Web of Science are extensive databases, they may not encompass all relevant publications on the subject. Consequently, future studies should incorporate additional databases such as Google Scholar and PubMed to ensure a more comprehensive analysis. Second, although VOS Viewer and Bibliometrix software are reliable tools for bibliometric analysis, other software options such as SciMat, Sci2, Bibexcel, Gephi, Cite Space, Pajek, and UCINET should also be utilized in future research to enhance robustness and validity.

Furthermore, the number of studies comparing different versions of ChatGPT, specifically versions 3.5, 4.0, and 4o in the context of software engineering, remains limited. Future research should explore the impact of these versions on outcomes and consider comparisons with other LLM tools such as Google Gemini, LLAMA, Microsoft Copilot, and Claude. Additionally, it is essential to examine the effects of various AI technologies, including DALL-E, DeepL, Typecast AI, and Resemble AI, on software engineering processes and outcomes.

VI. Conclusion

To our knowledge, this is the first bibliometric analysis study on using LLMs in software engineering. This study identifies the nations, authors, and publications that have contributed significantly to the field. The content analysis results show that the publications are organized around three key themes: 1) integration of LLMs into software engineering education, 2) application of LLMs in software engineering, and 3) potential and limitations of LLMs in software engineering. Our investigation reveals that China and the United States have the most publications, but international collaboration is limited. Consequently, future studies should encourage scholars to interact with researchers from other nations.

There is a gap in the literature concerning studies that explore LLMs and specific software engineering topics, which

future studies should address. Firstly, integrating LLMs like ChatGPT into software engineering education offers significant benefits, including personalized learning, practical application of concepts, and support for curriculum development. However, challenges such as potential negative impacts on student performance if overused and the necessity for students to have solid foundational knowledge despite AI assistance must be addressed. Students must evaluate AI-generated content critically to ensure genuine understanding. Therefore, balancing these tools with traditional learning methods is essential for maximizing their benefits in educational settings. To effectively integrate LLMs, curricula must be adapted to include AI literacy, maintain academic integrity, and prevent misconduct through evolving policies. Future directions involve creating interactive learning platforms, adopting holistic educational approaches, and continuously evaluating the impact of these tools to ensure they provide maximum educational benefits. Secondly, although LLMs like ChatGPT show promise in code generation, significant reliability and robustness issues remain. Comprehensive evaluation metrics and further studies are needed to assess and improve the effectiveness of LLMs in practical software development scenarios. ChatGPT has demonstrated great potential in software testing in areas such as automatic test generation, exploratory testing, and bug detection. However, these tools also have limitations and require further research to optimize their application in software engineering. Thirdly, research under the theme of automation in software engineering highlights the transformative potential of LLMs in automating various tasks, from technical interviews to domain modeling, code analysis, and bug fixing. Despite these advancements, challenges remain regarding the reliability of LLMs in these automated tasks.

VII. Future Work

Furthermore, LLMs can significantly enhance various aspects of software engineering, including requirements engineering, software modeling, and security checks. They provide practical tools for improving quality in software development processes. Nevertheless, there is a need for specialized models and further research to fully realize the potential of LLMs in practical applications and address existing limitations. Finally, studies demonstrate the potential of LLMs in software engineering but also underscore the need for further research to address limitations related to model robustness, handling complex tasks, reducing biases, and improving integration and usability in real-world software development environments. Future work should focus on developing more specialized LLM models tailored to specific tasks like code reviews, enhancing the robustness and adaptability of these models through extensive and varied datasets, and refining approaches like co-prompt engineering for better accuracy. Additionally, efforts should be made to mitigate over-reliance on automation, reduce biases in training data, and improve interactions between developers and LLM tools. Expanding testing in diverse environments, integrating advanced LLM models, and developing comprehensive training for non-experts to use these tools effectively are crucial steps for future research.

Acknowledgment

The authors declare that there are no conflicts of interest.

References

[1] K. I. Roumeliotis and N. D. Tselikas, “ChatGPT and Open-AI Models: A Preliminary Review,” Future Internet, vol. 15, no. 6, p. 192, May 2023, doi: 10.3390/fi15060192.

[2] Y. Chang et al., “A Survey on Evaluation of Large Language Models,” ACM Transactions on Intelligent Systems and Technology, vol. 15, no. 3, pp. 1–45, Jun. 2024, doi: 10.1145/3641289.

[3] M. A. Akbar, A. A. Khan, and P. Liang, “Ethical Aspects of ChatGPT in Software Engineering Research,” IEEE Transactions on Artificial Intelligence, pp. 1–14, 2023, doi: 10.1109/TAI.2023.3318183.

[4] W. Rahmaniar, “ChatGPT for Software Development: Opportunities and Challenges,” TechRxiv, vol. 26, no. 3, pp. 1–8, May 2023, doi: 10.1109/MITP.2024.3379831.

[5] D. K. Kim, J. Chen, H. Ming, and L. Lu, “Assessment of ChatGPT’s Proficiency in Software Development,” in Proceedings - 2023 Congress in Computer Science, Computer Engineering, and Applied Computing, CSCE 2023, Jul. 2023, pp. 2637–2644, doi: 10.1109/CSCE60160.2023.00421.

[6] J. Michael, D. Bork, M. Wimmer, and H. C. Mayr, “Quo Vadis modeling?: Findings of a community survey, an ad-hoc bibliometric analysis, and expert interviews on data, process, and software modeling,” Software and Systems Modeling, vol. 23, no. 1, pp. 7–28, Feb. 2024, doi: 10.1007/s10270-023-01128-y.

[7] N. Donthu, S. Kumar, D. Mukherjee, N. Pandey, and W. M. Lim, “How to conduct a bibliometric analysis: An overview and guidelines,” Journal of Business Research, vol. 133, pp. 285–296, Sep. 2021, doi: 10.1016/j.jbusres.2021.04.070.

[8] W. M. Lim and S. Kumar, “Guidelines for interpreting the results of bibliometric analysis: A sensemaking approach,” Global Business and Organizational Excellence, vol. 43, no. 2, pp. 17–26, Jan. 2024, doi: 10.1002/joe.22229.

[9] O. Öztürk, R. Kocaman, and D. K. Kanbach, “How to design bibliometric research: an overview and a framework proposal,” Review of Managerial Science, pp. 1–29, 2024, doi: 10.1007/s11846-024-00738-0.

[10] G. Favara, M. Barchitta, A. Maugeri, R. Magnano San Lio, and A. Agodi, “The Research Interest in ChatGPT and Other Natural Language Processing Tools from a Public Health Perspective: A Bibliometric Analysis,” Informatics, vol. 11, no. 2, p. 13, Mar. 2024, doi: 10.3390/informatics11020013.

[11] A. D. Samala, E. V. Sokolova, S. Grassini, and S. Rawas, “ChatGPT: a bibliometric analysis and visualization of emerging educational trends, challenges, and applications,” International Journal of Evaluation and Research in Education (IJERE), vol. 13, no. 4, p. 2374, 2024, doi: 10.11591/ijere.v13i4.28119.

[12] M. Oliński, K. Krukowski, and K. Sieciński, “Bibliometric Overview of ChatGPT: New Perspectives in Social Sciences,” Publications, vol. 12, no. 1, p. 9, Mar. 2024, doi: 10.3390/publications12010009.

[13] S. Gande, M. Gould, and L. Ganti, “Bibliometric analysis of ChatGPT in medicine,” International Journal of Emergency Medicine, vol. 17, no. 1, p. 50, Apr. 2024, doi: 10.1186/s12245-024-00624-2.

[14] A. S. Bale et al., “ChatGPT in Software Development: Methods and Cross-Domain Applications,” International Journal of Intelligent Systems and Applications in Engineering, vol. 11, no. 9s, pp. 636–643, 2023, [Online]. Available: https://www.scopus.com/inward/record.uri?eid=2-s2.0-85171329846&partnerID=40&md5=4283848fd2a7f64e2f54575369e84b6a.

[15] Y. Li, J. Xu, Y. Zhu, H. Liu, and P. Liu, “The Impact of ChatGPT on Software Engineering Education: A Quick Peek,” in 2023 10th International Conference on Dependable Systems and Their Applications (DSA), Aug. 2023, pp. 595–596, doi: 10.1109/DSA59317.2023.00087.

[16] N. Marques, R. R. Silva, and J. Bernardino, "Using ChatGPT in Software Requirements Engineering: A Comprehensive Review," Future Internet, vol. 16, no. 6, p. 180, May 2024, doi: 10.3390/fi16060180.

[17] H. J. Kasaraneni and S. Rosaline, "Automatic Merging of Scopus and Web of Science Data for Simplified and Effective Bibliometric Analysis," Annals of Data Science, vol. 11, no. 3, pp. 785–802, Jun. 2024, doi: 10.1007/s40745-022-00438-0.

[18] A. Klarin, "How to conduct a bibliometric content analysis: Guidelines and contributions of content co-occurrence or co-word literature reviews," International Journal of Consumer Studies, vol. 48, no. 2, p. e13031, Mar. 2024, doi: 10.1111/ijcs.13031.

[19] M. Aria and C. Cuccurullo, "bibliometrix: An R-tool for comprehensive science mapping analysis," Journal of Informetrics, vol. 11, no. 4, pp. 959–975, Nov. 2017, doi: 10.1016/j.joi.2017.08.007.

[20] S. I. Ross, F. Martinez, S. Houde, M. Muller, and J. D. Weisz, "The Programmer's Assistant: Conversational Interaction with a Large Language Model for Software Development," in Proceedings of the 28th International Conference on Intelligent User Interfaces, Mar. 2023, pp. 491–514, doi: 10.1145/3581641.3584037.

[21] M. Daun and J. Brings, "How ChatGPT Will Change Software Engineering Education," in Proceedings of the 2023 Conference on Innovation and Technology in Computer Science Education V. 1, Jun. 2023, vol. 1, pp. 110–116, doi: 10.1145/3587102.3588815.

[22] A. Ahmad, M. Waseem, P. Liang, M. Fahmideh, M. S. Aktar, and T. Mikkonen, "Towards Human-Bot Collaborative Software Architecting with ChatGPT," in Proceedings of the 27th International Conference on Evaluation and Assessment in Software Engineering, Jun. 2023, pp. 279–285, doi: 10.1145/3593434.3593468.

[23] N. M. Barrington et al., "A Bibliometric Analysis of the Rise of ChatGPT in Medical Research," Medical Sciences, vol. 11, no. 3, p. 61, Sep. 2023, doi: 10.3390/medsci11030061.

[24] H. Baber, K. Nair, R. Gupta, and K. Gurjar, "The beginning of ChatGPT – a systematic and bibliometric review of the literature," Information and Learning Sciences, vol. 125, no. 7/8, pp. 587–614, Jan. 2024, doi: 10.1108/ILS-04-2023-0035.

[25] T. Yalcinkaya and S. Cinar Yucel, "Bibliometric and content analysis of ChatGPT research in nursing education: The rabbit hole in nursing education," Nurse Education in Practice, vol. 77, 2024, doi: 10.1016/j.nepr.2024.103956.

[26] O. Petrovska, L. Clift, and F. Moller, "Generative AI in Software Development Education: Insights from a Degree Apprenticeship Programme," in The United Kingdom and Ireland Computing Education Research (UKICER) conference, Sep. 2023, pp. 1–1, doi: 10.1145/3610969.3611132.

[27] E. Frankford, C. Sauerwein, P. Bassner, S. Krusche, and R. Breu, "AI-Tutoring in Software Engineering Education," in Proceedings of the 46th International Conference on Software Engineering: Software Engineering Education and Training, Apr. 2024, pp. 309–319, doi: 10.1145/3639474.3640061.

[28] P. Rajabi, "Experience Report: Adopting AI-Usage Policy in Software Engineering Education," in The 26th Western Canadian Conference on Computing Education, May 2024, pp. 1–2, doi: 10.1145/3660650.3660668.

[29] V. D. Kirova, C. S. Ku, J. R. Laracy, and T. J. Marlowe, "Software Engineering Education Must Adapt and Evolve for an LLM Environment," in Proceedings of the 55th ACM Technical Symposium on Computer Science Education V. 1, Mar. 2024, vol. 1, pp. 666–672, doi: 10.1145/3626252.3630927.

[30] A. M. Abdelfattah, N. A. Ali, M. A. Elaziz, and H. H. Ammar, "Roadmap for Software Engineering Education using ChatGPT," in 2023 International Conference on Artificial Intelligence Science and Applications in Industry and Society (CAISAIS), Sep. 2023, pp. 1–6, doi: 10.1109/CAISAIS59399.2023.10270477.

[31] G. Jošt, V. Taneski, and S. Karakatič, "The Impact of Large Language Models on Programming Education and Student Learning Outcomes," Applied Sciences, vol. 14, no. 10, p. 4115, May 2024, doi: 10.3390/app14104115.

[32] O. Petrovska, L. Clift, F. Moller, and R. Pearsall, "Incorporating Generative AI into Software Development Education," in Proceedings of the 8th Conference on Computing Education Practice, Jan. 2024, pp. 37–40, doi: 10.1145/3633053.3633057.

[33] R. W. Brennan and J. Lesage, "Exploring the Implications of OpenAI Codex on Education for Industry 4.0," in Studies in Computational Intelligence, vol. 1083 SCI, Cham: Springer International Publishing, 2023, pp. 254–266.

[34] T. Kosar, D. Ostojić, Y. D. Liu, and M. Mernik, "Computer Science Education in ChatGPT Era: Experiences from an Experiment in a Programming Course for Novice Programmers," Mathematics, vol. 12, no. 5, p. 629, Feb. 2024, doi: 10.3390/math12050629.

[35] D. Sobania, M. Briesch, and F. Rothlauf, "Choose your programming copilot," in Proceedings of the Genetic and Evolutionary Computation Conference, Jul. 2022, pp. 1019–1027, doi: 10.1145/3512290.3528700.

[36] A. Moradi Dakhel, V. Majdinasab, A. Nikanjam, F. Khomh, M. C. Desmarais, and Z. M. (Jack) Jiang, "GitHub Copilot AI pair programmer: Asset or Liability?," Journal of Systems and Software, vol. 203, no. 111734, p. 111734, Sep. 2023, doi: 10.1016/j.jss.2023.111734.

[37] C. Ebert and P. Louridas, "Generative AI for Software Practitioners," IEEE Software, vol. 40, no. 4, pp. 30–38, Jul. 2023, doi: 10.1109/MS.2023.3265877.

[38] Y. Feng, S. Vanam, M. Cherukupally, W. Zheng, M. Qiu, and H. Chen, "Investigating Code Generation Performance of ChatGPT with Crowdsourcing Social Data," in 2023 IEEE 47th Annual Computers, Software, and Applications Conference (COMPSAC), Jun. 2023, vol. 2023-June, pp. 876–885, doi: 10.1109/COMPSAC57700.2023.00117.

[39] R. W. Brennan and J. Lesage, "Exploring the Implications of OpenAI Codex on Education for Industry 4.0," in Studies in Computational Intelligence, vol. 1083 SCI, Springer, 2023, pp. 254–266.

[40] M.-F. Wong, S. Guo, C.-N. Hang, S.-W. Ho, and C.-W. Tan, "Natural Language Generation and Understanding of Big Code for AI-Assisted Programming: A Review," Entropy, vol. 25, no. 6, p. 888, Jun. 2023, doi: 10.3390/e25060888.

[41] L. Belzner, T. Gabor, and M. Wirsing, "Large Language Model Assisted Software Engineering: Prospects, Challenges, and a Case Study," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 14380 LNCS, Springer, 2024, pp. 355–374.

[42] Z. Liu, Y. Tang, X. Luo, Y. Zhou, and L. F. Zhang, "No Need to Lift a Finger Anymore? Assessing the Quality of Code Generation by ChatGPT," IEEE Transactions on Software Engineering, vol. 50, no. 6, pp. 1548–1584, Jun. 2024, doi: 10.1109/TSE.2024.3392499.

[43] M. Liu, J. Wang, T. Lin, Q. Ma, Z. Fang, and Y. Wu, "An Empirical Study of the Code Generation of Safety-Critical Software Using LLMs," Applied Sciences, vol. 14, no. 3, p. 1046, Jan. 2024, doi: 10.3390/app14031046.

[44] G. L. Scoccia, "Exploring Early Adopters' Perceptions of ChatGPT as a Code Generation Tool," in Proceedings - 2023 38th IEEE/ACM International Conference on Automated Software Engineering Workshops, ASEW 2023, Sep. 2023, pp. 88–93, doi: 10.1109/ASEW60602.2023.00016.

[45] D. Rodriguez-Cardenas, D. N. Palacio, D. Khati, H. Burke, and D. Poshyvanyk, "Benchmarking Causal Study to Interpret Large Language Models for Source Code," in Proceedings - 2023 IEEE International Conference on Software Maintenance and Evolution, ICSME 2023, Oct. 2023, pp. 329–334, doi: 10.1109/ICSME58846.2023.00040.

[46] S. Yeo, Y. S. Ma, S. C. Kim, H. Jun, and T. Kim, "Framework for evaluating code generation ability of large language models," ETRI Journal, vol. 46, no. 1, pp. 106–117, 2024, doi: 10.4218/etrij.2023-0357.

[47] S. Aillon, A. Garcia, N. Velandia, D. Zarate, and P. Wightman, "Empirical evaluation of automated code generation for mobile applications by AI tools," in 2023 IEEE Colombian Caribbean Conference (C3), Nov. 2023, pp. 1–6, doi: 10.1109/C358072.2023.10436306.

[48] A. Mastropaolo et al., "On the Robustness of Code Generation Techniques: An Empirical Study on GitHub Copilot," in 2023 IEEE/ACM 45th International Conference on Software Engineering (ICSE), May 2023, pp. 2149–2160, doi: 10.1109/ICSE48619.2023.00181.

[49] L. Zhong and Z. Wang, "Can LLM Replace Stack Overflow? A Study on Robustness and Reliability of Large Language Model Code Generation," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 19, pp. 21841–21849, Mar. 2024, doi: 10.1609/aaai.v38i19.30185.

[50] L. Zhong and Z. Wang, "Can LLM Replace Stack Overflow? A Study on Robustness and Reliability of Large Language Model Code Generation," Proceedings of the AAAI Conference on Artificial Intelligence, vol. 38, no. 19, pp. 21841–21849, Mar. 2024, doi: 10.1609/aaai.v38i19.30185.

[51] W. B. Mbaka, "New experimental design to capture bias using LLM to validate security threats," in Proceedings of the 28th International Conference on Evaluation and Assessment in Software Engineering, Jun. 2024, pp. 458–459, doi: 10.1145/3661167.3661222.

[52] C. Tsigkanos, P. Rani, S. Müller, and T. Kehrer, "Variable Discovery with Large Language Models for Metamorphic Testing of Scientific Software," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 14073 LNCS, Cham: Springer Nature Switzerland, 2023, pp. 321–335.

[53] M. Schafer, S. Nadi, A. Eghbali, and F. Tip, "An Empirical Evaluation of Using Large Language Models for Automated Unit Test Generation," IEEE Transactions on Software Engineering, vol. 50, no. 1, pp. 85–105, Jan. 2024, doi: 10.1109/TSE.2023.3334955.

[54] K. El Haji, C. Brandt, and A. Zaidman, "Using GitHub Copilot for Test Generation in Python: An Empirical Study," in Proceedings of the 5th ACM/IEEE International Conference on Automation of Software Test (AST 2024), Apr. 2024, pp. 45–55, doi: 10.1145/3644032.3644443.

[55] S. Mehmood, U. I. Janjua, and A. Ahmed, "From Manual to Automatic: The Evolution of Test Case Generation Methods and the Role of GitHub Copilot," in 2023 International Conference on Frontiers of Information Technology (FIT), Dec. 2023, pp. 13–18, doi: 10.1109/FIT60620.2023.00013.

[56] R. Copche, Y. Duarte, V. Durelli, M. Eler, and A. Endo, "Can a Chatbot Support Exploratory Software Testing? Preliminary Results," in Proceedings of the 26th International Conference on Enterprise Information Systems, 2024, vol. 2, pp. 159–166, doi: 10.5220/0012572400003690.

[57] Q. Han, Z. Shi, and Z. Zhao, "Research on trustworthy Software Testing Techniques Based on Large Models," in 2024 10th International Symposium on System Security, Safety, and Reliability (ISSSR), Mar. 2024, pp. 524–525, doi: 10.1109/ISSSR61934.2024.00075.

[58] A. M. Dakhel, A. Nikanjam, V. Majdinasab, F. Khomh, and M. C. Desmarais, "Effective test generation using pre-trained Large Language Models and mutation testing," Information and Software Technology, vol. 171, no. 107468, p. 107468, Jul. 2024, doi: 10.1016/j.infsof.2024.107468.

[59] L. Plein, W. C. Ouédraogo, J. Klein, and T. F. Bissyandé, "Automatic Generation of Test Cases based on Bug Reports: A Feasibility Study with Large Language Models," Proceedings - International Conference on Software Engineering. ACM, University of Luxembourg, Luxembourg, Luxembourg, pp. 360–361, 2024, doi: 10.1145/3639478.3643119.

[60] D. I. Rathnayake, D. N. Mahendra, B. C. Amarasinghe, S. C. Premaratne, and M. M. Buhari, "Next Generation Technical Interview Process Automation with Multi-level Interactive Chatbot Based on Intelligent Techniques," in Lecture Notes in Networks and Systems, vol. 834, Singapore: Springer Nature Singapore, 2024, pp. 93–103.

[61] K. Chen, Y. Yang, B. Chen, J. A. Hernández López, G. Mussbacher, and D. Varró, "Automated Domain Modeling with Large Language Models: A Comparative Study," in 2023 ACM/IEEE 26th International Conference on Model Driven Engineering Languages and Systems (MODELS), Oct. 2023, pp. 162–172, doi: 10.1109/MODELS58315.2023.00037.

[62] G. F. Martins, E. C. M. Firmino, and V. P. De Mello, "The Use of Large Language Model in Code Review Automation: An Examination of Enforcing SOLID Principles," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 14736 LNAI, Cham: Springer Nature Switzerland, 2024, pp. 86–97.

[63] O. Asare, M. Nagappan, and N. Asokan, "Is GitHub's Copilot as bad as humans at introducing vulnerabilities in code?," Empirical Software Engineering, vol. 28, no. 6, p. 129, Nov. 2023, doi: 10.1007/s10664-023-10380-1.

[64] M. C. Wuisang, M. Kurniawan, K. A. Wira Santosa, A. Agung Santoso Gunawan, and K. E. Saputra, "An Evaluation of the Effectiveness of OpenAI's ChatGPT for Automated Python Program Bug Fixing using QuixBugs," in 2023 International Seminar on Application for Technology of Information and Communication (iSemantic), Sep. 2023, pp. 295–300, doi: 10.1109/iSemantic59612.2023.10295323.

[65] C. Jain, P. R. Anish, A. Singh, and S. Ghaisas, "A Transformer-based Approach for Abstractive Summarization of Requirements from Obligations in Software Engineering Contracts," in 2023 IEEE 31st International Requirements Engineering Conference (RE), Sep. 2023, vol. 2023-Septe, pp. 169–179, doi: 10.1109/RE57278.2023.00025.

[66] P. Spoletini and A. Ferrari, "The Return of Formal Requirements Engineering in the Era of Large Language Models," in Lecture Notes in Computer Science (including subseries Lecture Notes in Artificial Intelligence and Lecture Notes in Bioinformatics), vol. 14588 LNCS, IEEE, 2024, pp. 344–353.

[67] R. Ren, J. W. Castro, A. Santos, O. Dieste, and S. T. Acuna, "Using the SOCIO Chatbot for UML Modelling: A Family of Experiments," IEEE Transactions on Software Engineering, vol. 49, no. 1, pp. 364–383, Jan. 2023, doi: 10.1109/TSE.2022.3150720.

[68] J. Cámara, J. Troya, J. Montes-Torres, and F. J. Jaime, "Generative AI in the Software Modeling Classroom: An Experience Report with ChatGPT and UML," IEEE Software, pp. 1–10, 2024, doi: 10.1109/MS.2024.3385309.

[69] G. De Vito, F. Palomba, C. Gravino, S. Di Martino, and F. Ferrucci, "ECHO: An Approach to Enhance Use Case Quality Exploiting Large Language Models," in 2023 49th Euromicro Conference on Software Engineering and Advanced Applications (SEAA), Sep. 2023, pp. 53–60, doi: 10.1109/SEAA60479.2023.00017.

[70] G. Melo, "Designing Adaptive Developer-Chatbot Interactions: Context Integration, Experimental Studies, and Levels of Automation," in 2023 IEEE/ACM 45th International Conference on Software Engineering: Companion Proceedings (ICSE-Companion), May 2023, pp. 235–239, doi: 10.1109/ICSE-Companion58688.2023.00064.

[71] N. Petrović, "Chat GPT-Based Design-Time DevSecOps," in 2023 58th International Scientific Conference on Information, Communication and Energy Systems and Technologies (ICEST), Jun. 2023, pp. 143–146, doi: 10.1109/ICEST58410.2023.10187247.

[72] J. Lu, L. Yu, X. Li, L. Yang, and C. Zuo, "LLaMA-Reviewer: Advancing Code Review Automation with Large Language Models through Parameter-Efficient Fine-Tuning," in 2023 IEEE 34th International Symposium on Software Reliability Engineering (ISSRE), Oct. 2023, pp. 647–658, doi: 10.1109/ISSRE59848.2023.00026.

[73] R. Tufano, O. Dabić, A. Mastropaolo, M. Ciniselli, and G. Bavota, "Code Review Automation: Strengths and Weaknesses of the State of the Art," IEEE Transactions on Software Engineering, vol. 50, no. 2, pp. 1–16, Feb. 2024, doi: 10.1109/TSE.2023.3348172.

[74] F. V. Pantelimon and B. Ștefan Posedaru, "Improving Programming Activities Using ChatGPT: A Practical Approach," in Smart Innovation, Systems and Technologies, vol. 367, Singapore: Springer Nature Singapore, 2024, pp. 307–316.

APPENDIX

https://bit.ly/3Ad9Qtf