DOI: 10.34185/1991-7848.itmm.2025.01.039

СУЧАСНІ ТЕНДЕНЦІЇ ТА ПРОБЛЕМИ НАЛАГОДЖЕННЯ ПРОГРАМНОГО ЗАБЕЗПЕЧЕННЯ НА БАЗІ ВЕЛИКИХ МОВНИХ МОДЕЛЕЙ

Завгородній А.Д.[1], Іванов О.П.[2]

[1]*Український державний університет науки і технологій, аспірант, Україна*

[2]*Український державний університет науки і технологій, к.т.н., доцент, Україна*

Анотація. *У доповіді представлено результати досліджень застосування великих мовних моделей (англ. LLM) у сфері налагодження програмного забезпечення. Розглянуто сучасний стан досліджень у цій галузі, включаючи як перспективні напрямки, так і існуючі проблеми, які обмежують широке практичне застосування такого застосування моделей. Проаналізовано різні підходи використання ВММ у процесі налагодження. Обґрунтовано, що ефективне використання ВММ для налагодження вимагає комплексного підходу, який враховує як розвиток самих моделей, так і підвищення кваліфікації розробників для забезпечення продуктивної взаємодії між людиною та штучним інтелектом. Робота спрямована на визначення оптимальних шляхів застосування ВММ у сфері налагодження програмного забезпечення, з урахуванням поточних технологічних можливостей та потреб розробників.*

Ключові слова: *великі мовні моделі (скороч. LLM), налагодження програмного забезпечення, автоматизоване налагодження, інтерактивні помічники, середовища розробки, взаємодія людина-ШІ, обмеження великих мовних моделей, розробники програмного забезпечення.*

На сьогоднішній день великі мовні моделі (далі скорочено ВММ) набули значної популярності в різноманітних сферах застосування, демонструючи підвищення ефективності при їх використанні [1, 15]. Зокрема, потужний поштовх отримала індустрія програмної інженерії. Моделі від таких компаній, як OpenAI (ChatGPT, та особливо Codex у співпраці з GitHub), Google (Gemini, PaLM 2), Anthropic (Claude), демонструють значний потенціал у генерації коду, процесах налагодження та рефакторингу, спрощуючи та полегшуючи трудомісткі процеси [1, 5, 15].

Основною темою даної доповіді є результати досліджень саме у сфері застосування ВММ у процесах налагодження програм: основні напрямки та підходи, а також проблеми та обмеження застосування ВММ.

Застосування ВММ для вирішення задач генерації коду продемонстрували певні можливості та потенціал у вирішенні синтаксичних помилок, що в свою чергу спровокувало ряд досліджень і експериментів щодо ширшого застосування ВММ у контексті налагодження [5]. Однак невдовзі було засвідчено певні проблеми такого застосування, зумовлені відсутністю глибокого розуміння контекстів (зокрема серед перших версій моделей) та обмеженими знаннями використаними для навчання моделей [2, 4, 5]. Ранні ВММ часто демонстрували обмежену здатність до складного логічного розмірковування та розуміння специфічних бібліотек або фреймворків, що використовувалися в реальних проєктах [2, 6]. Крім того, моделі могли генерувати правдоподібні, але невірні виправлення через галюцинації або використання застарілих знань, що становило значний ризик для процесу налагодження. Ці обмеження в контексті та знаннях безпосередньо впливали на ефективність ВММ при роботі зі складним, реальним кодом, що часто залежить від специфічних технологій та включає багатошарову логіку [5].

Для того аби подолати та покращити ефективність застосування ВММ для задач налагодження виникли певні підходи [7, 8, 9, 10]. Однак, дані напрацювання, які хоч і демонструють певні покращення ефективності процесу, все ж досі мають низку критичних проблем зумовлених обмеженнями цих моделей, які потребують вирішення та покращення.

Однією з критичних проблем є обмеження розміру вхідного параметру моделі, відомого як контекстне вікно, яке з легкістю порушується розмірами репозиторіїв, якими є більшість сучасних реальних проєктів, з якими працюють розробники [11, 17]. Хоча деякі новіші моделі, такі як ChatGPT4, Gemini 1.5 та 2, мають значно більші контекстні вікна, ефективність обробки інформації в межах дуже великого контексту активно досліджується [3, 11, 17]. Для того аби подолати проблему обмеженого контекстного вікна застосовуються розбиття проєкту на менші логічні одиниці для спрощення роботи [12], проте таке застосування потребує обережного проектування нових індексованих структур щоб знизити ризик втрати повноти розуміння як контексту програми так і зав’язків модулів та файлів програми.

Іншою критичною проблемою є «нова помилка», природу якої модель раніше не бачила або якої не зустрічалась в наборі даних для навчання ВММ [5, 15, 18]. Дана проблема може виникнути в реальних проєктах, коли використовуються нові фреймворки, мови програмування або навіть при застосуванні вже відомих технологій але в унікальних комбінаціях [6, 19]. Хоча ВММ може виконати поверхневий аналіз та намагатиметься вирішити проблему використовуючи певні шаблони та знайомі зв'язки, до яких можна звести нову помилку, зазвичай цього може бути недостатньо аби виконати ефективний аналіз, визначити дефект та запропонувати доречні зміни. Для вирішення цієї проблеми досліджуються підходи, що дозволяють ВММ отримувати доступ до зовнішніх знань, наприклад, через бази знань [14] або шляхом взаємодії з розробником для отримання додаткової інформації [13, 15, 16].

Враховуючи наведені вище проблеми зокрема перспективним є напрямок досліджень, відмінний від повного автоматичного виконання налагодження, спрямований на використання ВММ в парі з експертом-розробником [15, 16]. Даний напрямок отримав розвиток на базі розробок інтегрованих у середовища розробки агентів-помічників, які надають розробникам змогу інтерактивно та покроково виконувати налагодження, також надаючи змогу вести діалоги відносно тієї чи іншої ділянки коду чи отриманої під час виконання програми [16]. Як вже згадувалось у роботі, ВММ, досі є недосконалими, зокрема через недостатність знань відносно тієї чи іншої технології, мови програмування, а також відсутністю глибинного розуміння контексту. Водночас, синергія між розробником та ВММ може зменшити ці проблеми, але за умови експертизи розробника [19]. Як показують дослідження і звіти, загалом при роботі з ВММ, в середньому показники ефективності роботи покращуються, серед досвідчених працівників так і серед новачків, пришвидшуючи їх навчання [19]. Однак є й низка недоліків. В першу чергу, це необхідність додаткових верифікацій, особливо серед початківців, адже при навчанні через недостачу знань або експертизи студенти можуть використовувати частково вірні запропоновані ВММ рішення надто довіряючи

їм, що зокрема відбувається доволі часто [19]. Іншою ж проблемою є поміч у роботі із складними логічними завданням, яка все ж досі залишається несуттєвою (зазвичай це пов'язано із згаданою вже обмеженістю контекстного розуміння моделями) [19] та потребує ретельного припрацювання експертом шляхом надання додаткової інформацію про контекст власноруч.

Висновки. Звертаючи увагу на наведені в даній роботі аспекти, маємо, що застосування ВММ для автоматичного налагодження залишається обмеженим та потребує суттєвих покращень через низку причин: обмеження контекстного вікна ВММ (неможливість працювати з великими, реальними проєктами) – потребує додаткового розбиття та індексації, невідомі помилки (неможливість ефективно визначити дефект та запропонувати його виправлення) - потребує розвитку інтеграції окремих баз знань та просунутих стратегій пошуку. Також одним із вагомих факторів є власне обмеження моделей, однак вже сьогодні можна покращити процеси налагодження за рахунок інтеграцій ВММ у середовища розробки у вигляді інтерактивних агентів-помічників, які мають працювати в парі з розробником-експертом. Даний факт в свою чергу засвідчує необхідність не тільки розробляти і покращувати можливості ВММ, а й навчати та розвивати навички налагодження у студентів, для ефективної кооперації агентів та розробників.

ЛІТЕРАТУРА / REFERENCES

1. A survey on large language models for software engineering / Q. Zhang et al. arXiv.org. URL: https://doi.org/10.48550/arXiv.2312.15223 (дата звернення: 27.03.2025).
2. Large language models for software engineering: survey and open problems / A. Fan et al. 2023 IEEE/ACM international conference on software engineering: future of software engineering (icse-fose), Melbourne, Australia, 14–20 May 2023. 2023. URL: https://doi.org/10.1109/icse-fose59343.2023.00008 (дата звернення: 25.03.2025).
3. A comprehensive survey of ai-driven advancements and techniques in automated program repair and code generation / A. Anand et al. arXiv.org e-Print archive. URL: https://doi.org/10.48550/arXiv.2411.07586 (дата звернення: 25.03.2025).
4. Zhang Y. Chain of Agents: large language models collaborating on long-context tasks. Google Research - Explore Our Latest Research in Science and AI. URL: https://research.google/blog/chain-of-agents-large-language-models-collaborating-on-long-context-tasks/ (дата звернення: 25.03.2025).

5. A deep dive into large language model code generation mistakes: what and why? / Q. Chen et al. arXiv.org. URL: https://doi.org/10.48550/arXiv.2411.01414 (дата звернення: 26.03.2025).
6. LLMs love python: a study of llms' bias for programming languages and libraries / L. Twist et al. arXiv.org e-Print archive. URL: https://arxiv.org/html/2503.17181v1#S6 (дата звернення: 27.03.2025).
7. Teaching large language models to self-debug / X. Chen et al. arXiv.org. URL: https://doi.org/10.48550/arXiv.2304.05128 (дата звернення: 26.03.2025).
8. Training LLMs to better self-debug and explain code / N. Jiang et al. Amazon Science. URL: https://www.amazon.science/publications/training-llms-to-better-self-debug-and-explain-code (дата звернення: 26.03.2025).
9. Zhong L., Wang Z., Shang J. Debug like a human: a large language model debugger via verifying runtime execution step by step. Findings of the association for computational linguistics ACL 2024, Bangkok, Thailand and virtual meeting. Stroudsburg, PA, USA, 2024. P. 851–870. URL: https://doi.org/10.18653/v1/2024.findings-acl.49 (дата звернення: 26.03.2025).
10. InferFix: End-to-End Program Repair with LLMs / M. Jin et al. ESEC/FSE '23: 31st ACM joint european software engineering conference and symposium on the foundations of software engineering, San Francisco CA USA. New York, NY, USA, 2023. URL: https://doi.org/10.1145/3611643.3613892 (дата звернення: 26.03.2025).
11. Extending context window of large language models via semantic compression / W. Fei et al. Findings of the association for computational linguistics ACL 2024, Bangkok, Thailand and virtual meeting. Stroudsburg, PA, USA, 2024. P. 5169–5181. URL: https://doi.org/10.18653/v1/2024.findings-acl.306 (дата звернення: 26.03.2025).
12. Shiraishi M., Shinagawa T. Context-aware code segmentation for c-to-rust translation using large language models. arXiv.org e-Print archive. URL: https://arxiv.org/html/2409.10506v1#S8 (дата звернення: 26.03.2025).
13. Enhancing the code debugging ability of llms via communicative agent based data refinement / W. Yang et al. arXiv.org e-Print archive. URL: https://arxiv.org/html/2408.05006v1 (дата звернення: 27.03.2025).
14. KBLaM: knowledge base augmented language model / X. Wang et al. arXiv.org. URL: https://doi.org/10.48550/arXiv.2410.10450 (дата звернення: 27.03.2025).
15. Haque M. A. LLMs: a game-changer for software engineers?. arXiv.org. URL: https://doi.org/10.48550/arXiv.2411.00932 (дата звернення: 27.03.2025).
16. CodeAgent: enhancing code generation with tool-integrated agent systems for real-world repo-level coding challenges / K. Zhang et al. Proceedings of the 62nd annual meeting of the association for computational linguistics (volume 1: long papers), Bangkok, Thailand. Stroudsburg, PA, USA, 2024. P. 13643–13658. URL: https://doi.org/10.18653/v1/2024.acl-long.737 (дата звернення: 27.03.2025)
17. Alibaba lingmaagent: improving automated issue resolution via comprehensive repository exploration / Y. Ma et al. arXiv.org. URL: https://doi.org/10.48550/arXiv.2406.01422 (дата звернення: 27.03.2025).

18. Tarassow A. The potential of LLMs for coding with low-resource and domain-specific programming languages. arXiv.org. URL: https://doi.org/10.48550/arXiv.2307.13018 (дата звернення: 27.03.2025).
19. Human-AI experience in integrated development environments: a systematic literature review / A. Sergeyuk та ін. arXiv.org e-Print archive. URL: https://arxiv.org/html/2503.06195v1#S5 (дата звернення: 01.04.2025).

MODERN TRENDS AND CHALLENGES IN DEBUGGING SOFTWARE BASED ON LARGE LANGUAGE MODELS

Andrii Zavhorodnii, Oleksandr Ivanov

Abstract. *This report presents the results of research on the application of large language models (LLMs) in the field of software debugging. The current state of research in this area is examined, including both promising directions and existing problems that limit the widespread practical application of such model usage. Various approaches to using LLMs in the debugging process are analyzed. Particular attention is paid to the integration of LLMs into development environments as interactive assistants that work in close collaboration with the developer. It is argued that the effective use of LLMs for debugging requires a comprehensive approach that considers both the development of the models themselves and the improvement of developers' skills to ensure productive human-AI interaction. The work aims to identify optimal ways of applying LLMs in the field of software debugging, considering current technological capabilities and the needs of developers.*

Keywords: *Large Language Models (LLMs), software debugging, automated debugging, interactive assistants, development environments, human-AI interaction, LLM limitations, software developers.*