УДК 004.42:004.8
DOI https://doi.org/10.32782/tnv-tech.2024.5.8

КОМУНІКАТИВНА ПЛАТФОРМА ДЛЯ ПОБУДОВИ ТА ДОСЛІДЖЕННЯ МЕТОДІВ ГЕНЕРАЦІЇ ПРОГРАМ НА ОСНОВІ НАТУРАЛЬНИХ МОВ ІЗ ВИКОРИСТАННЯМ ШТУЧНОГО ІНТЕЛЕКТУ

Свистунов А. О. *– аспірант, асистент кафедри теорії*
та технології програмування
Київського національного університету імені Тараса Шевченка
ORCID ID: 0009-0005-4536-5083

Роботу присвячено методам генерації програм з використанням методів штучного інтелекту на основі натуральних мов. Пропонується використання комунікативної платформи для класифікації та побудови методів генерації програм на основі натуральних мов. Важливим аспектом для генерації програм є мультимодальність засобів специфікації – сприйняття відповідним засобом генерації нечітких умов у різних формах, наприклад, у запитах натуральними мовами, діаграмах, таблицях, тощо.

В роботі розглядаються підходи, що базуються на використанні засобів генеративного штучного інтелекту, в тому числі мультимодальних, з метою моделювання комунікативних систем. Ключем до використання комунікативної платформи для опису та дослідження засобів штучного інтелекту для генерації програм на основі натуральних мов є комбінування реалізацій складових частин комунікативної системи обміну інформації, зокрема методу опису предметної області, задання суб'єкта-ініціатора та суб'єкта-обробника.

Таким чином, пропонується класифікація на основі способів задання об'єктів предметної області, способів задання мети обробки (програми) та вибору моделі суб'єкта-обробника та способом задання його внутрішніх процедур. Мета обробки може бути задана або неявно у вигляді вимог до вихідних об'єктів, або явно у вигляді кроків виконання необхідних перетворень. В термінах комунікативної інформатики обидва формати задання мети подаються як спеціальний інформаційний об'єкт – програма.

В статті описуються в термінах комунікативної платформи методи на основі генерації програмного коду та методи на основі інтелектуальних агентів та змішані підходи, згортку комунікативних систем, де ШІ-агент є одночасно і суб'єктом-обробником, і суб'єктом-ініціатором, що задає дескриптивні системи для виконавця-обробника програмного коду, зокрема, саморефлексійні підходи (Reasoning and Acting) та архітектура Artificial Intelligence Operating System. Зокрема мультиагентна архітектура AgentCoder, є еталонним рішенням згідно оцінок на датасетах для задач генерації програмного коду HumanEval та MBPP.

Ключові слова: *генерація програм, генерація коду, великі мовні моделі, комунікативний процес, комунікативні системи.*

Svystunov A. O. Communicative platform for classifying and constructing program generation methods based on natural language using artificial intelligence

The paper is devoted to the methods of generating programs using artificial intelligence based on natural languages. It is proposed that a communicative platform be used to classify and construct methods of generating programs based on natural languages. An essential aspect of generating programs is the multimodality of the means of specification – the perception of the appropriate means of generating fuzzy conditions in various forms, for example, in natural language queries, diagrams, tables, etc.

The work considers approaches based on generative artificial intelligence tools, including multimodal ones, to model communication systems. The key to using a communicative platform for the description and research of artificial intelligence tools for generating programs based on natural languages is the combination of implementations of the components of the communicative system of information exchange, in particular, the method of describing the subject area and the assignment of the subject-initiator and subject-processor.

Thus, a classification is proposed based on the methods of specifying the objects of the subject area, the methods of specifying the purpose of the processing (program), and the selection of the model of the processing entity, and the method of specifying its internal procedures. The

purpose of processing can be specified either implicitly in the form of requirements for source objects or explicitly in the form of steps to perform the necessary transformations. In terms of communicative informatics, both formats of setting a goal are presented as a special information object – a program.

The article describes, in terms of the communication platform, methods based on the generation of software code and methods based on intelligent agents and mixed approaches, a convolution of communication systems, where the AI agent is both a processing subject and an initiating subject that sets descriptive systems for the executor-processor of the program code, in particular, self-reflective approaches? Like ReAct (Reasoning and Acting) and the architecture of the Artificial Intelligence Operating System. In particular, the multi-agent architecture of AgentCoder is the state of the art solution according to benchmarks on datasets for the tasks of generating the software code of HumanEval and MBPP.

Key words: *program generation, code generation, large language models, communicative process, communicative systems.*

Генерація програм належить до сфери інтересів багатьох галузей, зокрема, розробки програмного забезпечення, освіти, безпеки, тощо. Можливість автоматичного створення та модифікації програм призведе до зменшення витрат на розробку та впровадження, а також дозволить створювати програмні рішення залучаючи менше технічних спеціалістів.

Важливим аспектом для генерації програм є мультимодальність засобів специфікації – сприйняття відповідним засобом генерації нечітких умов у різних формах, наприклад, у запитах натуральними мовами, діаграмах, таблицях, тощо.

Суттєвим кроком до розв'язання задачі генерації програм стали дослідження та розробки в галузі генеративного штучного інтелекту, зокрема, великих мовних моделей (на англ. "Large Language Models "LLM"). Великі мовні моделі дозволяють генерувати текст (чи артефакти інших форматів), який може бути або кодом програми, або перетворенням вхідних даних на основі певної інструкції. Прикладами таких моделей є GPT-4 [1], Llama 2 [2], StarCoder 2 [3], Mistral 7B [4] та інші.

Відповідно до розвитку засобів машинного навчання в аспектах генерації програмного коду, постає необхідність у формалізації задачі генерації програм з точки зору теорії програмування. Така формалізація дозволить об'єднати існуючі методи під одну теоретичну базу, а також надати можливості до проєктування нових методів.

В рамках даної роботи запропоновано формулювання теоретичних засад генерації програм з використанням штучного інтелекту на основі комунікативної платформи інформатики [5].

Постановка задачі. Об'єктом комунікативної інформатики є поняття комунікативної системи обміну інформації, що складається з [5]:

- **опису предметної області** з певною сукупністю інформаційних об'єктів;
- **суб'єкта-ініціатора**, що формує і передає вхідну інформацію суб'єкту обробнику та приймає від нього певну вхідну інформацію;
- **суб'єкта-обробника**, що приймає вхідну інформацію, аналізує її та повертає як вихідну суб'єкту-ініціатору.

Головним призначенням комунікативних систем є встановлення зв'язку між даними предметної області – вхідними та вихідними. Життєвий цикл комунікативної системи розпочинається з запиту на обробку певних вхідних інформаційних об'єктів від ініціатора обробнику та містить мету обробки. Мета обробки може бути задана або неявно у вигляді вимог до вихідних об'єктів, або явно у вигляді кроків виконання необхідних перетворень. В термінах комунікативної інформатики обидва формати задання мети подаються як спеціальний інформаційний об'єкт – **програма**. Обробник має власні внутрішні інформаційні об'єкти, що є прообразами зовнішніх об'єктів ПрО.

Метою методів генерації програм на основі натуральної мови є перетворення вхідних специфікацій натуральною мовою, що подані у форматі *гіпертексту* (текст, діаграми, таблиці, тощо) у стан комунікативної системи для виконання перетворень, що відповідають специфікації. Метою використання засобів штучного інтелекту в даних методах є задання (генерація) дескриптивних систем вхідних даних (формат, який приймає програма на вхід), вихідних даних (формат результату перетворень), запиту (опису перетворень) та внутрішню мову обробника (стан обробника).

Загалом, методи генерації програм на основі натуральної мови можна розділити за:

- способом задання об'єктів предметної області;
- способом задання мети обробки (програми);
- вибором моделі суб'єкта-обробника та способом задання його внутрішніх процедур.

Далі у роботі використовуватиметься саме ця класифікація.

Методи на основі генерації програмного коду. Засоби генерації програмного коду охоплюють широкий спектр задач і об'єднують засоби машинного навчання, глибокого навчання та обробки натуральної мови для перетворень, результатом яких є програмний код певною мовою програмування.

Більшість засобів, що підпадають під цей термін, не можуть бути самостійно використані для генерації програм на основі натуральної мови, бо засоби для вирішення задач, наприклад, доповнення коду, обфускації/деобфускації коду, автоматичного виправлення коду, міграції коду, тощо, приймають на вхід програмний код [6]. Варто зазначити, що, хоча і обмежено, але код може містити специфікацію натуральною мовою, але у вигляді коментаря, або іншого синтаксично коректного способу.

Натомість, засоби синтезу програм, як очевидно з визначення, можуть бути застосовані для генерації програм на основі запитів натуральної мови. Методи на основі засобів (моделей машинного навчання) для генерації програмного коду відрізняються використанням інтерпретатора (або компілятора та обчислювального пристрою) в якості *суб'єкта-обробника*. *Предметна область* моделюється використовуючи в якості дескриптивної системи певну визначену мову програмування, яка також використовується для задання мети обчислень. Результатом виконання моделі синтезу програм є текст програмного коду, який може бути запущений суб'єктом-обробником.

Хоча даний підхід є природнім, бо за технологічною складовою повторює процес кодування, використання даного підходу містить суттєві недоліки.

Однією з головних проблем є забезпечення синтаксичної та семантичної коректності згенерованого коду. Датасети для тренування та оцінки рішень для генерації програмного коду, такі як HumalEval [7] та MBPP [8], містять лише засоби для тестування цільових функцій, без засобів для формальної верифікації та загалом перевірки коду на відповідність вимогам. Генерація коду може відбуватись рекурсивно в декілька етапів, таким чином покращуючи якість коду [8], проте на сьогодні не існує надійного механізму уникнення синтаксичних помилок та логічних неточностей у згенерованому програмному коді.

Ще одним викликом є проблема узагальнення. Моделі можуть показувати задовільні метрики точності на прикладах з тренувального датасету, але їх результат значно погіршується на нових, невідомих даних. Ця проблема загострюються обмеженістю контекстного вікна та уваги моделі, що не дозволяє тренувати моделі на великих кодових базах [3].

Методи на основі ШІ-агентів. Common Sence Reasoning (з англ. "Міркування на основі здорового глузду") є властивістю сучасних великих мовних моделей і використовується в якості оцінки когнітивних здібностей агентів штучного інтелекту. Для цього моделі тестують [1; 2; 4] на різних задачах від розв'язання шкільних іспитів [9] до аналізу повсякденних ситуацій [10].

Дана властивість знайшла широке застосування у створенні інтелектуальних агентів на основі великих мовних моделей. В якості інтелектуального агента, або ШІ-агента, далі вважатимемо сутності, що здатні сприймати стан навколишнього середовища та змінювати його.

Універсальність ШІ-агентів на основі великих мовних моделей дозволяє використовувати їх в комунікативній системі в якості суб'єкта-обробника, що може виконувати перетворення вхідних даних за певною специфікацією. В цьому випадку, задача зводиться до коректного формулювання мети перетворення, яку має здійснити агент над вхідними даними предметної області. Програмою в даному методі виступає аугментований контекст, що описує специфікацію предметної області натуральною мовою та бажане перетворення. Перетворення може бути задане неявно ("zero-shot" підхід), так і явно у вигляді плану або протоалгоритму ("few-shot" та "chain-of-thoughts" підхід). Інтерфейс спілкування з зовнішнім світом агента, його внутрішня процедура як обробника, також може бути аугментована.

Саморефлексивний підхід до побудови агентів-обробників може бути використаний для покращення результатів перетворень. Підхід до побудови ReAct (Reasoning and Acting, з англ – "Міркування та дії") дозволяє агенту багатокроково планувати та реалізовувати складні перетворення [11]. Таким чином, обробник стає своїм же ініціатором. Окрім цього, для виконання та оцінки перетворень може бути використаний ансамбль агентів довільної топології.

Якісною відміною від методів на основі генерації коду є використання натуральної мови в якості дескриптивної системи, як для задання мети обчислень, так і для задання внутрішньої процедури суб'єкта-обробника. Такий підхід дозволяє уникнути проблем з обмеженістю навчального датасету та проблем з синтаксисом та семантикою згенерованих програм. Натомість, постає проблема з відтворюваністю результатів перетворень через недетерміновану природу виходу великих мовних моделей.

Ще однією проблемою, що випливає з недетермінованості, є втрата інформації при виконанні перетворень, зокрема, арифметичних операцій [12].

Змішаний підхід. Розглянуті до цього методи на основі засобів генерації коду та інтелектуальних агентів відрізнялись природою внутрішньої ДС суб'єкта-обробника. Змішаний підхід до генерації програм з використанням методів штучного інтелекту на основі натуральної мови полягає у комбінації даних підходів і дозволяє балансувати їх переваги та недоліки. По суті, маємо згортку комунікативних систем, де ШІ-агент є одночасно і суб'єктом-обробником, і суб'єктом-ініціатором, що задає дескриптивні системи для виконавця-обробника програмного коду.

Ця ідея лежить в основі архітектури AI OS (Artificial Intelligence Operating System, з англ. "Операційна система на основі штучного інтелекту") [13]. В рамках даної архітектури, велика мовна модель виступає головним обчислювальних елементом, використовуючи інтерпретатор певної мови програмування для окремих задач. Таким чином, частина перетворень може бути делегована готовим програмним рішенням.

Це відкриває простір до варіацій рішень – комбінуючи інтелектуальних агентів та засоби для виконання програмного коду, змішані архітектури досягають найвищих оцінок точності у задачах синтезу програм. Зокрема, мультиагентна архітектура AgentCoder, що реалізує наведений вище підхід є еталонним рішенням згідно оцінок на датасетах HumanEval та MBPP з метрикою точності у 96.3% та 91.8% відповідно на основі моделі GPT-4, досягаючи приросту точності мінімум 39.4% [14]. В залежності від обраної моделі та датасету, середній приріст точності коливається від 39.4% для GPT-3.5-turbo на HumanEval до 277.4% на Claude-Instant-1 на MBPP.

Також, змішаний підхід може бути використаний для підвищення надійності та прозорості процесу генерації шляхом перевикористання верифікованого та/або попередньо згенерованого коду [15].

Висновки. В роботі було запропоновано використання комунікативної платформи для класифікації та побудови методів генерації програм. Було розглянуто підходи на основі генерації коду та інтелектуальних агентів з точки зору моделювання комунікативних систем засобами штучного інтелекту на основі натуральної мови. Дана платформа може бути використаною для дослідження, порівняння та побудови архітектур прикладних програмних засобів для генерації програм з використанням методів ШІ.

СПИСОК ВИКОРИСТАНОЇ ЛІТЕРАТУРИ:

1. OpenAI, Achiam J., Adler S., Agarwal S. та ін. GPT-4 Technical Report. 2024. arXiv:2303.08774 [cs.CL]. URL: https://arxiv.org/abs/2303.08774.

2. Touvron H., Martin L., Stone K., Albert P. та ін. Llama 2: Open Foundation and Fine-Tuned Chat Models. 2023. arXiv:2307.09288 [cs.CL]. URL: https://arxiv.org/abs/2307.09288.

3. Lozhkov A., Li R., Ben Allal L., Cassano F. та ін. StarCoder 2 and The Stack v2: The Next Generation. 2024. arXiv:2402.19173 [cs.SE]. URL: https://arxiv.org/abs/2402.19173.

4. Jiang A.Q., Sablayrolles A., Mensch A., Bamford C. та ін. Mistral 7B. 2023. arXiv:2310.06825 [cs.CL]. URL: https://arxiv.org/abs/2310.06825.

5. Zubenko V. On the communicative informatics // Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics. 2013. № 2. P. 151–156.

6. Le T.H.M., Chen H., Babar M.A. Deep Learning for Source Code Modeling and Generation: Models, Applications, and Challenges // ACM Computing Surveys. 2020. Vol. 53, № 3. P. 1–38. DOI: 10.1145/3383458. URL: http://dx.doi.org/10.1145/3383458.

7. Chen M., Tworek J., Jun H., Yuan Q., Ponde de Oliveira Pinto H., Kaplan J., та ін. Evaluating Large Language Models Trained on Code. 2021. arXiv:2107.03374 [cs.LG]. URL: https://arxiv.org/abs/2107.03374.

8. Austin J., Odena A., Nye M., Bosma M., Michalewski H., та ін. Program Synthesis with Large Language Models. 2021. arXiv:2108.07732 [cs.PL]. URL: https://arxiv.org/abs/2108.07732.

9. Cobbe K., Kosaraju V., Bavarian M., Chen M., Jun H., та ін. Training Verifiers to Solve Math Word Problems. 2021. arXiv:2110.14168 [cs.LG]. URL: https://arxiv.org/abs/2110.14168.

10. Zellers R., Holtzman A., Bisk Y., Farhadi A., Choi Y. HellaSwag: Can a Machine Really Finish Your Sentence? 2019. arXiv:1905.07830 [cs.CL]. URL: https://arxiv.org/abs/1905.07830.

11. Yao S., Zhao J., Yu D., Du N., Shafran I., та ін. ReAct: Synergizing Reasoning and Acting in Language Models. 2023. arXiv:2210.03629 [cs.CL]. URL: https://arxiv.org/abs/2210.03629.

12. Xia S., Li X., Liu Y., Wu T., Liu P. Evaluating Mathematical Reasoning Beyond Accuracy. 2024. arXiv:2404.05692 [cs.CL]. URL: https://arxiv.org/abs/2404.05692.

13. Mei K., Li Z., Xu S., Ye R., Ge Y., Zhang Y. AIOS: LLM Agent Operating System. 2024. arXiv:2403.16971 [cs.OS]. URL: https://arxiv.org/abs/2403.16971.

14. Huang D., Bu Q., Zhang J.M., Luck M., Cui H. AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation. 2024. arXiv:2312.13010 [cs.CL]. URL: https://arxiv.org/abs/2312.13010.

15. Koziolek H., Grüner S., Hark R., Ashiwal V., Linsbauer S., Eskandani N. LLM-based and Retrieval-Augmented Control Code Generation // In Proceedings of 1st International Workshop on Large Language Models for Code (LLM4Code'24). 2024. Association for Computing Machinery (ACM).

REFERENCES:

1. OpenAI, Achiam, J., Adler, S., Agarwal, S., & et al. (2024). *GPT-4 Technical Report*. arXiv. https://arxiv.org/abs/2303.08774

2. Touvron, H., Martin, L., Stone, K., Albert, P., & et al. (2023). *Llama 2: Open Foundation and Fine-Tuned Chat Models*. arXiv. https://arxiv.org/abs/2307.09288

3. Lozhkov, A., Li, R., Ben Allal, L., Cassano, F., & et al. (2024). *StarCoder 2 and The Stack v2: The Next Generation*. arXiv. https://arxiv.org/abs/2402.19173

4. Jiang, A. Q., Sablayrolles, A., Mensch, A., Bamford, C., & et al. (2023). *Mistral 7B*. arXiv. https://arxiv.org/abs/2310.06825

5. Zubenko, V. (2013). On the communicative informatics. *Bulletin of Taras Shevchenko National University of Kyiv. Series Physics & Mathematics, 2*, 151–156.

6. Le, T. H. M., Chen, H., & Babar, M. A. (2020). Deep learning for source code modeling and generation: Models, applications, and challenges. *ACM Computing Surveys, 53*(3), 1–38. https://doi.org/10.1145/3383458

7. Chen, M., Tworek, J., Jun, H., Yuan, Q., Ponde de Oliveira Pinto, H., Kaplan, J., & et al. (2021). *Evaluating Large Language Models Trained on Code*. arXiv. https://arxiv.org/abs/2107.03374

8. Austin, J., Odena, A., Nye, M., Bosma, M., Michalewski, H., & et al. (2021). *Program Synthesis with Large Language Models*. arXiv. https://arxiv.org/abs/2108.07732

9. Cobbe, K., Kosaraju, V., Bavarian, M., Chen, M., Jun, H., & et al. (2021). *Training Verifiers to Solve Math Word Problems*. arXiv. https://arxiv.org/abs/2110.14168

10. Zellers, R., Holtzman, A., Bisk, Y., Farhadi, A., & Choi, Y. (2019). HellaSwag: Can a machine really finish your sentence? *arXiv*. https://arxiv.org/abs/1905.07830

11. Yao, S., Zhao, J., Yu, D., Du, N., Shafran, I., & et al. (2023). *ReAct: Synergizing Reasoning and Acting in Language Models*. arXiv. https://arxiv.org/abs/2210.03629

12. Xia, S., Li, X., Liu, Y., Wu, T., & Liu, P. (2024). *Evaluating Mathematical Reasoning Beyond Accuracy*. arXiv. https://arxiv.org/abs/2404.05692

13. Mei, K., Li, Z., Xu, S., Ye, R., Ge, Y., & Zhang, Y. (2024). *AIOS: LLM Agent Operating System*. arXiv. https://arxiv.org/abs/2403.16971

14. Huang, D., Bu, Q., Zhang, J. M., Luck, M., & Cui, H. (2024). *AgentCoder: Multi-Agent-based Code Generation with Iterative Testing and Optimisation*. arXiv. https://arxiv.org/abs/2312.13010

15. Koziolek, H., Grüner, S., Hark, R., Ashiwal, V., Linsbauer, S., & Eskandani, N. (2024). LLM-based and retrieval-augmented control code generation. In *Proceedings of 1st International Workshop on Large Language Models for Code (LLM4Code'24)*. Association for Computing Machinery (ACM).