УДК 004.912

С.Г. Букина, С.С. Харченко

Набор данных для выявления искусственно сгенерированного исходного кода

Современные генеративные языковые модели активно используются для автоматической генерации исходного кода, что приводит к необходимости разработки методов его обнаружения. Однако создание наборов данных для определения сгенерированного кода остается затруднительной задачей. В данной работе проводится анализ существующих наборов данных с выявлением их ограничений. Разработан авторский набор данных, включающий решения задач в виде кода на языке программирования Python, написанные людьми и сгенерированные современными языковыми моделями. Проведена экспериментальная оценка с использованием методов машинного обучения. Результаты демонстрируют перспективность предложенного набора, но указывают на необходимость его дальнейшего расширения или же проведения новых экспериментов для подбора наилучшей модели.
Ключевые слова: исходный код, машинное обучение, языковые модели, набор данных, классификация кода.
DOI: 10.21293/1818-0442-2025-28-2-106-110

Современные генеративные модели на основе трансформеров активно используются для автоматической генерации кода, что находит применение в разработке программного обеспечения, тестировании и образовательных задачах. Согласно опросу [1] за 2023 г., 44% респондентов используют инструменты искусственного интеллекта в процессе разработки, а 26% планируют в ближайшее время. Согласно опросу [2] за 2024 г., 76% всех респондентов используют или планируют использовать инструменты искусственного интеллекта в процессе разработки, что больше, чем в прошлом году (70%), и большее число разработчиков используют инструменты искусственного интеллекта (62 против 44%). Наряду с преимуществами генерация кода несет ряд угроз: внедрение уязвимостей, снижение качества программных решений, сложность контроля авторства и возможные правовые риски. Возникает необходимость исследования возможности различения искусственно сгенерированного и написанного людьми исходного кода. Практическое значение развития данной темы имеется для целого спектра сценариев:

1. Выявление сгенерированных решений в образовательных учреждениях для оценки навыков студентов.

2. Обнаружение потенциально опасных паттернов или неэффективных конструкций до внедрения исходного кода в системы.

3. Проверка кода, представленного как результат работы разработчика (при приеме на работу, в рамках аудита качества кода в компании) для подтверждения авторства и понимания.

4. Понимание доли и характера сгенерированного кода в проекте для оценки инструментов разработчика (ИИ-ассистенты).

На сегодняшний день, несмотря на активное развитие генеративных моделей, развитие наборов данных для задач выявления сгенерированного кода отстает. Существующие наборы данных в открытом доступе, как правило, не включают сгенерированные образцы. Цель данной статьи – провести анализ наборов данных для определения сгенерированного кода, выделить ограничения и предложить авторский набор данных, который, несмотря на свои ограничения, может служить инструментом для дальнейших исследований.

Анализ существующих наборов

Для анализа существующих подходов к созданию наборов данных для выявления сгенерированного исходного кода была составлена сравнительная таблица, в которой представлены характеристики наборов данных, используемых в научных работах. В таблице отражены источники кода, написанного людьми, модели, используемые для генерации образцов, размеры наборов данных и языки программирования.

В работах [3–11] используются различные подходы к формированию наборов данных: в одних работах за основу берутся наборы данных, в которых присутствуют задачи с соревновательных платформ вместе с решениями людей, в других – образцы исходного кода собираются из репозиториев с открытым исходным кодом на Github. Для начала надежным вариантом для генерации решений являются задачи с соревновательных платформ, поскольку задается четкое условие. В случае с репозиториями авторы использовали для генерации кода стратегии «Перевод кода», «Перевод функций», «Функциональная настройка».

Первая стратегия предполагает предоставление генеративной модели фрагмента кода, написанного на одном языке программирования, и выдачу задания сгенерировать эквивалентный по функциональности код на другом языке программирования.

Вторая стратегия подразумевает, что генеративной модели сперва необходимо проанализировать и обобщить функциональное описание из выданного фрагмента кода, а затем сгенерировать код на том же языке программирования на основе составленного ранее описания.

Третья стратегия позволяет генеративной модели генерировать код на основе уже существующих функциональных описаний, полученных из соревнований по программированию, учебников и других

типичных задач. Также в случае отбора кода из репозиториев были выполнены следующие шаги по очистке набора данных: удаление учетных записей организаций, разветвленных хранилищ и дубликатов; временная сегментация – отбор кода, опубликованного до 2022 г.; удаление репозиториев для совместной работы. Все наборы, за исключением набора авторов работы [4] являются закрытыми.

Сравнительный анализ наборов данных

Работа	Источники кода, написанного людьми	Генеративные модели	Размер набора	Язык программирования
[3]	HumanEval, MBPP, APPS	CodeGen-6B, InCoder-6B, PolyCoder-2.7B, GPTNeo-2.7B и CodeT5-base (220M)	APPS: 8000 (по 4000 на класс). MBPP: 1542 (по 771 на класс). HumanEval: 262 (по 131 на класс)	C, C++ и C#
[4]	APPS, CCF, CC, HAEA, HED, MBPPD, MTrajk	gpt 3.5-turbo	31 400 (по 15700 на класс)	Python
[5]	CodeWorkout	GPT 3.5	3 162 решения от студентов. 3 000 решений от модели	Java
[6]	CodeChef	GPT 3.5	322 образца (по 161 на класс)	Python
[7]	Github, Google Code Jam	GPT-3.5, GPT-4	300 образцов (по 150 на класс)	Java и C++
[8]	Github	GPT 3.5	По 601 образцу на класс	Java
[9]	CodeNet	GPT 3.5	215 образцов от людей. 327 образцов от модели	C
[10]	APPS	Gemma 7B, Mixtral 8X7B и CodeLlama 70B, GPT-4	272 образца на класс	Java
[11]	CodeNet и AIGCode	OpenAI text-davinci-003 (GPT-3.5)	5 214 сгенерированных решений (по 869 на каждый язык). Число образцов класса решений людей неизвестно	C, C++, C#, Java, JavaScript и Python

Из данных таблицы можно заключить, что в каждом наборе присутствуют те или иные ограничения:

1. Использование одной модели генерации: большинство работ используют только одну модель (чаще всего GPT-3.5), что ограничивает разнообразие сгенерированных образцов и не учитывает особенности других генеративных моделей.

2. Небольшой размер наборов: во многих работах размер набора данных не превышает нескольких тысяч образцов.

3. Ограниченное разнообразие задач: наборы данных часто содержат задачи только одного типа.

4. Наборы данных не обновляются с появлением новых генеративных моделей, что снижает их актуальность.

5. Ограниченное разнообразие языков программирования.

Таким образом, при создании собственного набора необходимо также учитывать ограничения при подборе и использовании алгоритмов машинного обучения. Учитывая развитие языковых моделей, появляется потребность в новых наборах данных, ориентированных на конкретные языки программирования и генеративные модели.

Авторский набор данных

Авторский набор данных представляет собой экспериментальный инструмент, разработанный для исследования возможности различения искусственно сгенерированного и написанного людьми исходного кода. Набор данных включает фрагменты кода на языке Python, как написанные людьми, так и сгенерированные современными языковыми моделями.

В качестве источника решений людей был выбран набор данных APPS [12], поскольку он сформирован в 2021 г., до широкого распространения и использования генеративных моделей (авторы других работ устанавливали крайнюю границу в 2022 г.). Также данный набор содержит задачи разного уровня сложности (начальный, собеседование, соревнование). Таким образом, APPS является подходящим набором в качестве отправной точки формирования набора данных по исследуемой теме.

Из исходного набора APPS было отобрано 500 задач, равномерно распределенных по уровням сложности: 167 задач начального уровня, 167 задач уровня собеседования и 166 задач соревновательного уровня. Для каждой задачи было отобрано до 5 уникальных решений, написанных людьми, что в итоге составило 2 238 образцов.

Для создания сгенерированных образцов были выбраны четыре современные языковые модели, исходя из версии и доступности: GPT 4o-mini, Yandex GPT 4 PRO RC, Nvidia/Llama 3.1 Nemotron 70B Instruct HF, Qwen 2.5 72B Instruct.

Каждая модель использовалась для генерации решений для всех 500 задач. Процесс сбора данных осуществлялся вручную через интерфейсы чатов моделей. Для каждой задачи условие копировалось и вставлялось в поле ввода модели с инструкцией предоставить решение на Python и принять определенную роль в зависимости от сложности задачи. В случае если модель выдавала некорректный ответ (например, текстовое объяснение вместо кода), запрос повторялся до получения удовлетворительного результата.

В результате было собрано по 500 решений от каждой модели, что составило 2 000 сгенерированных образцов. Распределение решений по уровню сложности задач следующее: 1 356 решений на соревновательный уровень, 1 408 – на уровень собеседования, 1 474 – на начальный уровень.

В целом на класс решений людей приходится 52,8% образцов, на класс решений моделей – 47,2%. На рис. 1 представлена доля решений от каждого из источников.

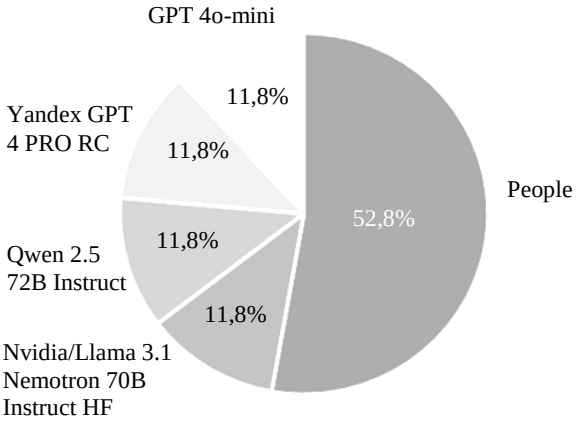

Рис. 1. Доля решений по источникам исходного кода

Набор данных структурирован следующим образом:

1. ID: Уникальный идентификатор записи.
2. Source_url: Ссылка на источник задачи.
3. Difficulty: Уровень сложности задачи (начальный, собеседование, соревнование).
4. Problem: Условие задачи.
5. Solution: Решение в виде кода на Python.
6. Source: Автор решения (0 – человек, 1–4 – модель генерации).
7. Class_label: Метка класса (0 – человек, 1 – модель).

Для оценки применимости набора данных были проведены выбраны два классических алгоритма машинного обучения: логистическая регрессия и метод опорных векторов. Данные модели позволяют без значительных затрат вычислительных ресурсов провести обучение и проанализировать результат на этапе разработки набора данных, оценить выделенные моделями признаки. Для векторизации кода использовался метод TF-IDF, позволяющий выделить информативные элементы кода и снизить вес часто встречающихся конструкций. Набор был разделен на обучающую выборку (80% образцов) и тестовую (20% образцов). На тренировочной выборке была применена 10-кратная кроссвалидация для обучения моделей с различными гиперпараметрами, где выбор оптимальных гиперпараметров проводился с использованием GridSearchCV. Модели с оптимальными гиперпараметрами были обучены на всей обучающей выборке. Финальная оценка проводилась на тестовой выборке, результаты представлены далее.

Логистическая регрессия: Accuracy 89%; F1-мера 90% для класса «Человек» и 88% для класса «Модель».

На рис. 2 представлена матрица ошибок для логистической регрессии.

Метод опорных векторов (SVC): Accuracy 89%, F1-мера 90% для класса «Человек» и 87% для класса «Модель». На рис. 3 представлена матрица ошибок для SVC.

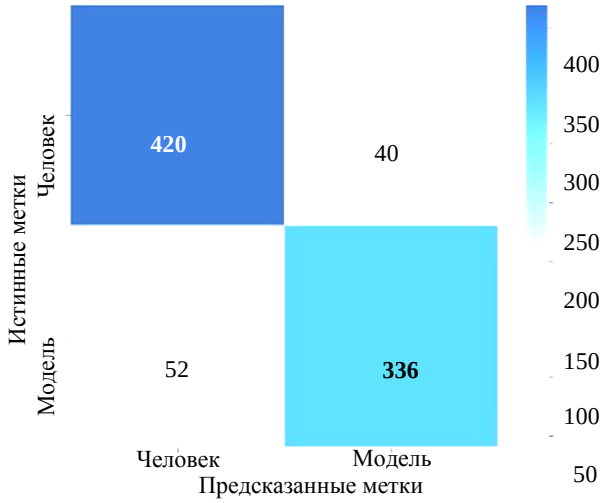

Рис. 2. Матрица ошибок, логистическая регрессия

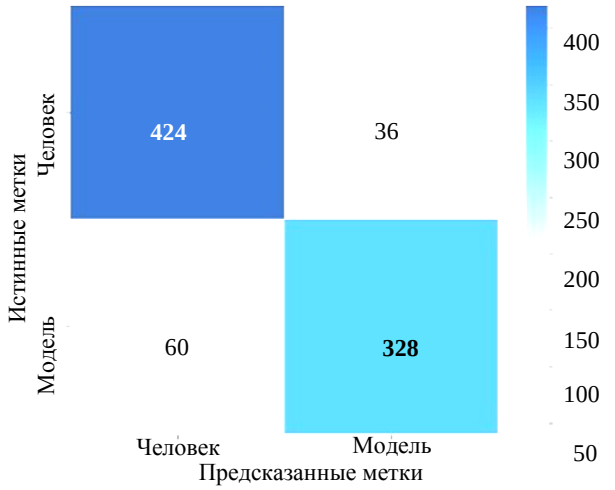

Рис. 3. Матрица ошибок SVC

Подбор гиперпараметров статистически значимо улучшил метрики при уровне значимости $p = 0,05$ ($p = 0,0019$ для F1 и Accuracy у логистической регрессии; $p = 0,048$ и $p = 0,037$ соответственно для SVC по критерию Уилкоксона).

Результаты экспериментов подтверждают, что авторский набор данных может быть использован для обучения и тестирования моделей классификации. Однако асимметрия в ошибках (модель чаще ошибается при классификации сгенерированного кода) указывает на необходимость дальнейшего улучшения моделей классификации и экспериментов с признаками кода. Авторский набор данных представляет собой важный шаг в развитии исследований по определению сгенерированного исходного кода. Несмотря на свои ограничения, он предоставляет возможность тестирования новых методов классификации и является отправной точкой для дальнейших работ. В будущем планируется расширение набора данных за счет включения большего числа образцов от каждой модели генерации для их отдельного рассмотрения, выбор фрагментов, отличных от алгоритмических задач, уже присутствующих в наборе, работа над рассмотрением новых признаков кода (например, включение структурных особенностей, метрик Холстеда, эксперименты со словесными n-граммами).

Основные сложности при создании набора

Формирование репрезентативного набора данных является одной из ключевых задач в рамках представленной научной темы: с развитием языковых моделей имеется потребность в разнообразии наборов – вариативность версий моделей, актуальность набора по отношению к используемым на момент исследования моделям, включение задач разного типа и уровня сложности, включение кода, написанного разными разработчиками. В данном разделе рассмотрены основные трудности, осложняющие работу над данной темой и требующие реагирования.

Современные языковые модели эволюционируют и демонстрируют рост качества генерации кода, приближая его к человеческому стилю. Исследования показали, что модели семейства Codex улучшили стилистическое и функциональное соответствие кода человеческим решениям и способны генерировать функционально корректный код, трудно отличимый от написанного разработчиками [13]. Этот прогресс создает проблему: любой статичный набор быстро устаревает и теряет свою эффективность. Решением могло бы стать внедрение динамически обновляемых наборов, но подход к формированию таких наборов данных для различения сгенерированного кода пока отсутствует.

Следующей сложностью является вариативность стиля написания кода среди разработчиков. Даже внутри одного проекта могут наблюдаться различия в подходах к именованию переменных, организации кода и использованию структур данных [14]. Модели могут адаптировать стиль написания кода к конкретному разработчику, что затрудняет классификацию. В связи с этим модели, обученные на одном наборе данных, могут оказаться недостаточно надежными при применении к реальным программным проектам.

На текущий момент большинство доступных наборов данных либо закрыты для академического использования, либо охватывают лишь узкий спектр задач, что сужает их применимость к реальным сценариям. Например, наборы данных APPS и HumanEval фокусируются на алгоритмических задачах, что не отражает реальное разнообразие промышленного кода. Как показано в таблице, многие наборы включают лишь один язык программирования из перечня распространенных языков, отсутствуют наборы с иными языками программирования.

Также несбалансированность распределения образцов по классам «человек» и «модель» в обучающих данных может приводить к смещению предсказаний классификаторов, т.е. при преобладании образцов от одной генеративной модели (например, GPT-3.5) алгоритмы могут терять способность детектиро-вать код от других моделей (например, CodeLlama). Кроме того, ошибки генерации, такие как избыточное использование шаблонных конструкций, могут стать легко предсказуемыми признаками для классификаторов, снижая их устойчивость к новым версиям моделей, что подчеркивает необходимость разработки новых наборов с обновленными версиями моделей [15].

Таким образом, создание набора данных для различения сгенерированного кода остается затруднительной задачей. В данной работе проведен анализ существующих наборов, выявлены их основные ограничения. Предложенный авторский набор данных представляет собой вклад в развитие данной темы. Он включает решения, написанные людьми, и код, сгенерированный современными языковыми моделями, к задачам различного уровня сложности.

Экспериментальные результаты, полученные с использованием методов машинного обучения, показывают, что модели способны классифицировать код в рамках представленного набора.

Дальнейшие исследования следует направить на работу с выделением различных признаков исходного кода и эксперименты с другими моделями машинного обучения. Кроме того, перспективным направлением является разработка наборов данных под конкретную тематическую направленность (например, веб-разработка, data science) и языки программирования.

Литература

1. Hype or not? AI’s benefits for developers explored in the 2023 Developer Survey [Электронный ресурс]. – URL: https://stackoverflow.blog/2023/06/14/hype-or-not-developers-have-something-to-say-about-ai/ (дата обращения: 16.09.2024).

2. 2024 Developer Survey [Электронный ресурс]. – URL: https://survey.stackoverflow.co/2024/ai/ (дата обращения: 16.09.2024).

3. The «Code» of Ethics: A Holistic Audit of AI Code Generators / W. Ma, Y. Song, M. Xue, S. Wen, Y. Xiang // IEEE Transactions on Dependable and Secure Computing. – 2024. – Vol. 21, No. 5. – P. 4997–5013.

4. ChatGPT Code Detection: Techniques for Uncovering the Source of Code / M. Oedingen, R. Denz, R. Engelhardt, M. Hammer // AI Journal. – 2024. – Vol. 5, No. 3. – P. 1066–1094.

5. Detecting ChatGPT-Generated Code Submissions in a CS1 Course Using Machine Learning Models / M. Hoq, Y. Shi, J. Leinonen, D. Babalola // SIGCSE 2024: Proceedings of the 55th ACM Technical Symposium on Computer Science Education. – Portland, Oregon, United States, 2024. – Vol. 1. – P. 526–532.

6. Whodunit: Classifying Code as Human Authored or GPT-4 Generated – A case study on CodeChef problems / O.J. Idialu, N.S. Mathews, R. Maipradit, J.M. Atlee // Proceedings of the 21st International Conference on Mining Software Repositories. – Lisbon, Portugal, 2024. – P. 394–406.

7. Discriminating Human-authored from ChatGPT-Generated Code Via Discernable Feature Analysis / K. Li, S. Hong, C. Fu, Y. Zhang, M. Liu // IEEE 34th International Symposium on Software Reliability Engineering Workshops. – Florence, Italy, 2023. – P. 120–127.

8. GPTSniffer: A CodeBERT-based classifier to detect source code written by ChatGPT / P. Nguyen, J. Rocco, C. Sipio, R. Rubei // Journal of Systems and Software. – 2024. – Vol. 214. – P. 112059.

9. Bukhari S.A. Issues in Detection of AI-Generated Source Code: The Requirements for the degree of Masters of Science. – Calgary, 2024. – 102 p.

10. Sjoerd S. The Detection of AI Generated Coding Content: The Requirements for the degree of Masters of Science. – Utrecht, 2024. – 75 p.

11. Xu Z. Detecting AI-Generated Code Assignments Using Perplexity of Large Language Models / Z. Xu, V. Sheng // Proceedings of the AAAI Conference on Artificial Intelligence. – Vancouver, Canada, 2024. – P. 23155–23162.

12. Набор данных APPS на платформе Hugging Face [Электронный ресурс]. – URL: https://huggingface.co/datasets/codeparrot/apps (дата обращения: 24.10.2024).

13. Chen M. Evaluating Large Language Models Trained on Code / M. Chen, J. Tworek, H. Jun [Электронный ресурс]. – URL: https://arxiv.org/abs/2107.03374 (дата обращения: 16.02.2025).

14. Allamanis M. A Survey of Machine Learning for Big Code and Naturalness / M. Allamanis, E. Barr, P. Devanbu // ACM Computing Surveys (CSUR). – 2018. – Vol. 51, No. 81. – P. 1–37.

15. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models / P. Vaithilingam, T. Wu, E. Glassman // Proceedings of the CHI Conference on Human Factors in Computing Systems Extended Abstracts – 2022. – No. 332. – P. 1–7.

Букина София Германовна
Студентка каф. безопасности информационных систем (БИС) Томского государственного ун-та систем управления и радиоэлектроники (ТУСУР)
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7-952-183-45-99
Эл. почта: sofiabukina1714@gmail.com

Харченко Сергей Сергеевич
Канд. техн. наук, доцент каф. БИС ТУСУРа
Ленина пр-т, 40, г. Томск, Россия, 634050
Тел.: +7 (382-2) 70-15-29
Эл. почта: kss@fb.tusur.ru

Поступила в редакцию: 08.05.2025.
Принята к публикации: 21.06.2025.

Bukina S.G., Kharchenko S.S.
Dataset for Detecting AI-Generated Source Code

Modern generative language models are actively used for automated source code generation, necessitating the development of detection methods. However, the creation of datasets for identifying machine-generated code remains a challenging task. This paper analyzes existing datasets, identifying their limitations. An original dataset is developed, comprising Python programming language solutions to problems written by humans and generated by state-of-the-art language models. Experimental evaluation is conducted using machine learning methods. The results demonstrate the promise of the proposed dataset while indicating the need for its further expansion or conducting new experiments to identify the optimal model.
Keywords: source code, machine learning, language models, dataset, code classification.
DOI: 10.21293/1818-0442-2025-28-2-106-110

References

1. Hype or not? AI's benefits for developers explored in the 2023 Developer Survey. Available at: https://stackoverflow.blog/2023/06/14/hype-or-not-developers-have-something-to-say-about-ai/ (Accessed: 16 September 2024).

2. 2024 Developer Survey. Available at: https://survey.stackoverflow.co/2024/ai/ (Accessed: 16 September 2024).

3. Ma W., Song Y., Xue M., Wen S., Xiang Y. The «Code» of Ethics: A Holistic Audit of AI Code Generators. *IEEE Transactions on Dependable and Secure Computing*, 2024, vol. 21, no. 5, pp. 4997–5013.

4. Oedingen M., Denz R., Engelhardt R., Hammer M. ChatGPT Code Detection: Techniques for Uncovering the Source of Code. *AI Journal*, 2024, vol. 5, no. 3, pp. 1066– 1094.

5. Hoq M., Shi Y., Leinonen J., Babalola D. *Detecting ChatGPT-Generated Code Submissions in a CS1 Course Using Machine Learning Models*. Proceedings of the 55th ACM Technical Symposium on Computer Science Education, Portland, Oregon, United States, 2024, vol. 1, pp. 526–532.

6. Idialu O.J., Mathews N.S., Maipradit R., Atlee J.M. *Whodunit: Classifying Code as Human Authored or GPT-4 Generated - A case study on CodeChef problems*. Proceedings of the 21st International Conference on Mining Software Repositories, Lisbon, Portugal, 2024, pp. 394–406.

7. Li K., Hong S., Fu C., Zhang Y., Liu M. *Discriminating Human-authored from ChatGPT-Generated Code Via Discernable Feature Analysis*. IEEE 34th International Symposium on Software Reliability Engineering Workshops, Florence, Italy, 2023, pp. 120–127.

8. Nguyen P., Rocco J., Sipio C., Rubei R. GPTSniffer: A CodeBERT-based classifier to detect source code written by ChatGPT. *Journal of Systems and Software*, 2024, vol. 214, 112059.

9. Bukhari S.A. *Issues in Detection of AI-Generated Source Code: The Requirements for the degree of Masters of Science*, Calgary, 2024, 102 p.

10. Sjoerd S. *The Detection of AI Generated Coding Content: The Requirements for the degree of Masters of Science*, Utrecht, 2024, 75 p.

11. Xu Z., Sheng V. *Detecting AI-Generated Code Assignments Using Perplexity of Large Language Models*. Proceedings of the AAAI Conference on Artificial Intelligence, Vancouver, Canada, 2024, pp. 23155–23162.

12. Dataset APPS on Hugging Face. Available at: https://huggingface.co/datasets/codeparrot/apps (Accessed: 24 October 2024).

13. Chen M., Tworek J., Jun H. Evaluating Large Language Models Trained on Code. Available at: https://arxiv.org/abs/2107.03374 (Accessed: 16 February 2025).

14. Allamanis M., Barr E., Devanbu P. A Survey of Machine Learning for Big Code and Naturalness. *ACM Computing Surveys (CSUR)*, 2018, vol. 51, no. 81, pp. 1–37.

15. Vaithilingam P., Wu T., *Glassman E. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools Powered by Large Language Models*. Proceedings of the CHI Conference on Human Factors in Computing Systems Extended Abstracts, 2022, no. 332, pp. 1–7.

Sofia S. Bukina
Student, Department of Information Systems Security (ISS), Tomsk State University of Control Systems and Radioelectronics (TUSUR)
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7-952-183-45-99
Email: sofiabukina1714@gmail.com

Sergei S. Kharchenko
Ph.D in Engineering, Associate Professor, ISS TUSUR
40, Lenin pr., Tomsk, Russia, 634050
Phone: +7 (382-2) 70-15-29
Email: kss@fb.tusur.ru

Received: 08.05.2025.
Accepted: 21.06.2025.