УДК 004.8:004.42

СРАВНИТЕЛЬНЫЙ АНАЛИЗ ИИ-АССИСТЕНТОВ ПРОГРАММИСТА ПРИ РАЗРАБОТКЕ УЧЕБНЫХ ПРОЕКТОВ

COMPARATIVE ANALYSIS OF AI ASSISTANTS FOR PROGRAMMERS IN EDUCATIONAL PROJECT DEVELOPMENT

А.С. Донских, студентка I курса гр. ПИб-251.3
А.В. Ионина, к.т.н., доцент кафедры ИТиЭД,
Филиал Федерального Государственного бюджетного образовательного учреждения высшего образования «Кузбасский государственный технический университет имени Т.Ф. Горбачева» в г. Новокузнецке, г. Новокузнецк

Аннотация: в статье проводится сравнительный анализ качества генерации программного кода современными ИИ-ассистентами (ChatGPT-4o, Copilot, Google-gemini-1.5-pro, YandexGPT) при разработке учебных проектов студентами направления «Программная инженерия». Для исследования разработано клиент-серверное приложение, выполнившее 560 запросов к каждой модели на трёх языках программирования (Python, C++, Java). Всего проанализировано 11 760 фрагментов кода. Выявлены лидеры по точности генерации, а также технические ограничения моделей. Сделаны рекомендации по использованию ИИ-ассистентов в образовательном процессе.

Ключевые слова: искусственный интеллект, генерация кода, ИИ-ассистенты, ChatGPT, Copilot, YandexGPT, программная инженерия, учебное проектирование.

Abstract: the article provides a comparative analysis of the quality of program code generation by modern AI assistants (ChatGPT-4o, Copilot, Google-gemini-1.5-pro, YandexGPT) in the development of educational projects by students of the Software Engineering direction. A client-server application was developed for the study, which executed 560 requests to each model in three programming languages (Python, C++, Java). A total of 11,760 code fragments were analyzed. The leaders in generation accuracy were identified, as well as technical limitations of the models. Recommendations for the use of AI assistants in the educational process are made.

Keywords: artificial intelligence, code generation, AI assistants, ChatGPT, Copilot, YandexGPT, software engineering, educational design.

С постоянным развитием компьютерных технологий и увеличением сложности программного обеспечения вопросы эффективной генерации программного кода становятся все более актуальными. Использование

современных моделей и методов искусственного интеллекта для автоматизации процесса создания программного кода является одним из перспективных направлений исследований в области программной инженерии [1–3]. С появлением и развитием нейронных сетей и глубокого обучения возможности автоматизации процесса создания программного кода значительно расширились. Глубокое обучение предоставляет мощные инструменты для генерации кода: с помощью различных архитектур, таких как рекуррентные нейронные сети (RNN), сверточные нейронные сети (CNN) и трансформеры, достигаются высокие результаты в этой области [4, 5].

В контексте учебного проектирования, где студенты только осваивают принципы построения сложных систем, критически важна не только функциональность, но и понятность, обучаемость и соответствие академическим стандартам генерируемого кода. Использование ИИ-ассистентов в учебном процессе может как ускорить разработку, так и сформировать неверные паттерны, если не контролировать качество результатов [3]. Несмотря на обилие работ по генерации кода, недостаточно исследований, сфокусированных именно на сравнении бесплатных и условно-бесплатных версий моделей для решения типовых учебных задач по программной инженерии.

Целью данного исследования является оценка качества применения моделей ИИ при генерации программного кода, предназначенного для разработки сложных программных систем студентами направления подготовки «Программная инженерия». Для достижения цели были поставлены следующие задачи: выполнить анализ научной литературы по технологиям генерации кода с помощью ИИ; определить перечень моделей для исследования их эффективности; разработать автоматизированное клиент-серверное приложение для оценивания эффективности исследуемых моделей.

В данной работе рассмотрены бесплатные и условно-бесплатные версии моделей ChatGPT-4o, Copilot, Google-gemini-1.5-pro и YandexGPT. Выбор обусловлен их популярностью среди разработчиков и студентов, доступностью, а также различиями в архитектурных подходах к генерации кода.

Для исследования моделей ИИ был сформулирован единообразный запрос на русском языке: «Написать контейнер типа priority_queue (очередь с приоритетом) с методами push, top. Организовать обработку исключительных ситуаций». Выбор данного задания обусловлен тем, что реализация priority_queue требует понимания объектно-ориентированного программирования, работы с шаблонами (в C++), управления памятью и обработки исключений, что позволяет комплексно оценить способность модели генерировать корректный и качественный код [6].

Дополнительные условия включали: написание программы на языке C++; генерацию необходимых значений случайным образом (пользователь вводит только размер очереди); вывод значений для визуального представления состояния структуры данных. Пример формулировки запроса и фрагмент сгенерированного кода для модели ChatGPT-4o представлены в таблице 1.

Таблица 1 – Пример запроса и результата генерации для модели ChatGPT-4o

Формулировка запроса	**Сгенерированный код**
Основное условие: Написать контейнер типа priority_queue (очередь с приоритетом) с методами push, top. Организовать обработку исключительных ситуаций. Дополнительные условия: 1. Написать программу на языке C++. 2. Генерировать необходимые значения случайным образом. Пользователь вводит только размер очереди. 3. Выводить значения для визуального представления.	#include <iostream> #include <vector> #include <queue> #include <stdexcept> class PriorityQueue { private: std::priority_queue<int, std::vector<int>, std::greater<int>> pq; size_t capacity; public: PriorityQueue(size_t cap) : capacity(cap) { std::cout << "Конструктор: создана очередь\n"; } void push(int value) { if (pq.size() >= capacity) throw std::overflow_error("Очередь заполнена"); pq.push(value); std::cout << "Добавлено: " << value << "\n"; display(); } int top() const { if (pq.empty()) throw std::underflow_error("Очередь пуста"); return pq.top(); } void display() const { auto temp = pq; std::cout << "Состояние: "; while (!temp.empty()) { std::cout << temp.top() << " "; temp.pop(); } std::cout << "\n"; } };

В ходе исследования каждой модели было отправлено по 560 запросов на генерацию кода для каждого из трёх языков программирования: Python, C++ и Java. Общее количество проанализированных фрагментов кода составило 11 760. Результаты анализа представлены в таблице 2.

Таблица 2 – Результаты тестирования моделей ИИ

Модель	**Язык**	**Правильные, %**	**Незапущенные, %**	**Неправильные %**	**Незначительные ошибки, %**
ChatGPT-o1-mini	Python	99,29	0,00	0,36	0,36
	C++	93,21	0,18	0,18	6,43
	Java	92,50	0,89	0,71	5,89
ChatGPT-4	Python	96,25	0,54	0,71	2,50
	C++	93,21	0,71	0,89	5,18
	Java	89,82	1,79	1,43	6,96
ChatGPT-	Python	95,89	0,36	0,36	3,39

4o	C++	94,64	0,36	0,54	4,46
	Java	89,64	2,14	2,14	6,07
Google-gemini-1.5-pro	Python	76,79	0,18	0,71	22,32
	C++	76,07	0,36	0,89	22,68
	Java	72,68	1,43	2,14	23,75
Google-gemini-1.5-flash	Python	75,54	0,36	1,07	23,04
	C++	73,57	0,36	0,89	25,18
	Java	67,86	2,68	2,86	26,61
YandexGPT	Python	57,50	3,93	2,86	35,71
	C++	56,79	2,32	3,21	37,68
	Java	52,50	4,46	5,36	37,68
Copilot	Python	85,71	0,54	0,54	13,21
	C++	83,21	0,89	1,07	14,82
	Java	80,89	2,86	1,96	14,29

Проведённое исследование выявило значительные различия в качестве генерации кода между моделями ИИ. Лидером стала ChatGPT-o1-mini с более чем 99 % правильных ответов для Python и свыше 92 % для C++ и Java, тогда как ChatGPT-4 и ChatGPT-4o показали результаты от 89 до 96 %. Аутсайдерами оказались YandexGPT (52–57 %) и Google-gemini-1.5-flash (67–75 %).

Для всех моделей наблюдалась зависимость от языка программирования: лучшие результаты достигнуты на Python благодаря простоте синтаксиса и большому объёму обучающих данных [7], тогда как строгая типизация Java и C++ предъявляет более высокие требования к моделям. Высокий процент «незначительных ошибок» у Google-gemini и YandexGPT (до 38 %), таких как отсутствие модификатора const, игнорирование стиля кодирования и проверок указателей, свидетельствует о недостаточном обучении лучшим практикам, что критично для учебного процесса, где студенты могут некритично перенимать неверные паттерны. Copilot занял промежуточное положение (80–85 %), будучи удобным для автодополнения, но уступая в генерации законченных решений.

Анализ полученных данных также показал, что даже у моделей-лидеров наблюдается корреляция между сложностью языка и количеством логических ошибок. Например, в Java-версиях часто возникали проблемы с некорректной реализацией дженериков, а в C++ – с утечками памяти при отсутствии виртуальных деструкторов. Это подчеркивает необходимость человеческого контроля за финальным кодом, особенно в образовательной среде, где студент только формирует навыки безопасного программирования.

В процессе исследования были выявлены следующие технические ограничения исследуемых моделей: YandexGPT имеет ограничение на количество символов в ответе – 1000. Большинство сгенерированных фрагментов кода не умещались в этот лимит, что требовало отправки дополнительных запросов для продолжения генерации. Кроме того,

существует ограничение на количество запросов в день, после превышения которого использование модели становится невозможным. Модели семейства ChatGPT (o1-mini, 4, 4o) имеют ограничение на количество запросов в день. После достижения этого лимита модели автоматически переключаются на более раннюю версию, которая не имеет ограничений, но генерирует ответы более низкого качества, что необходимо учитывать при интерпретации результатов. Copilot имеет ограничение на количество сообщений в одном диалоге, равное 30. После достижения этого лимита необходимо начинать новый диалог, что может быть неудобно при работе над крупными проектами.

Проведённое исследование показало, что современные модели ИИ обладают значительным потенциалом для автоматической генерации программного кода, что важно при разработке сложных программных систем. Модель ChatGPT-o1-mini уже сейчас может быть рекомендована для использования в образовательных и профессиональных целях при обязательной проверке кода преподавателем, в то время как Google-gemini и YandexGPT без тщательного контроля использовать не рекомендуется, а Copilot эффективен лишь как вспомогательный инструмент для автодополнения.

Обучение будущих программных инженеров применению ИИ-моделей в профессиональной деятельности может значительно повысить эффективность и качество их работы. Перспективным направлением дальнейших исследований видится разработка методик оценки не только синтаксической корректности, но и семантической сложности генерируемого кода, а также его соответствия современным стандартам качества программного обеспечения (SOLID, DRY, KISS). Это позволит формировать у студентов более глубокое понимание того, как эффективно интегрировать ИИ-инструменты в реальный процесс промышленной разработки.

Список литературы

1. Тимофеев А. Н. Разработка подхода к повышению качества генерации программного кода большими языковыми моделями / А. Н. Тимофеев, С. С. Михайлова // Нейрокомпьютеры: разработка, применение. – 2024. – Т. 26, № 4. – С. 18–26.
2. Резуник Л. Подход к созданию сервиса генерации программного кода мобильных приложений с использованием больших языковых моделей / Л. Резуник, Д. В. Александров // ИТ-Стандарт. – 2024. – № 4. – С. 34–41.
3. Вишнеков А. В. Особенности учебного процесса подготовки IT-специалистов в условиях возрастания возможностей генеративного искусственного интеллекта / А. В. Вишнеков, Е. А. Ерохина, Е. М.

Иванова, Н. К. Трубочкина // Инженерное образование. – 2023. – № 34. – С. 123–135.
4. Широков И. Б. Анализ технологий глубокого обучения с подкреплением для систем машинного зрения / И. Б. Широков, С. В. Колесова, В. А. Кучеренко, М. Ю. Серебряков // Известия Тульского государственного университета. Технические науки. – 2022. – № 10. – С. 118–120.
5. Vaswani A. Attention Is All You Need / A. Vaswani, N. Shazeer, N. Parmar [et al.] // Advances in Neural Information Processing Systems. – 2017. – Vol. 30. – P. 5998–6008.
6. Павловская Т. А. C/C++. Программирование на языке высокого уровня / Т. А. Павловская. – Санкт-Петербург: Питер, 2003. – 461 с. – ISBN 5-94723-568-4.
7. Brown T. B. Language Models are Few-Shot Learners / T. B. Brown, B. Mann, N. Ryder [et al.] // arXiv preprint arXiv:2005.14165. – 2020.