УДК 004.891:351

5.2.2. Математические, статистические и инструментальные методы в экономике (физико-математические науки, экономические науки)

АВТОМАТИЧЕСКОЕ ГЕНЕРИРОВАНИЕ КОДА С ИСПОЛЬЗОВАНИЕМ БОЛЬШИХ ЯЗЫКОВЫХ МОДЕЛЕЙ (LLM): ОГРАНИЧЕНИЯ И ПЕРСПЕКТИВЫ

Кушнир Надежда Владимировна
старший преподаватель кафедры информационных систем и программирования
РИНЦ-SCIENCE INDEX SPIN-код=6951-4012
kushnir.06@mail.ru
ФГБОУ ВО "Кубанский государственный технологический университет",
350020, улица Московская, 2, Краснодар, Россия

Зайков Владимир Полиевктович
доктор экономических и кандидат технических наук, профессор кафедры информационных систем и программирования
РИНЦ-SCIENCE INDEX SPIN-код=9531-8128
zaykovv@yandex.ru
ФГБОУ ВО "Кубанский государственный технологический университет",
350020, улица Московская, 2, Краснодар, Россия

Литовка Наталья Васильевна
старший преподаватель кафедры информационных систем и программирования
РИНЦ-SCIENCE INDEX SPIN-код=9446-1581
natalilitovkav@yandex.ru
ФГБОУ ВО "Кубанский государственный технологический университет",
350020, улица Московская, 2, Краснодар, Россия

Яцкевич Евгений Сергеевич
магистрант 2 курса
РИНЦ-SCIENCE INDEX SPIN-код=9985-2763
es_yatskevich@internet.ru
ФГБОУ ВО "Кубанский государственный технологический университет",
350020, улица Московская, 2, Краснодар, Россия

В данной статье рассматривается текущее состояние генерации кода на основе LLM, освещаются как его практические достижения, так и присущие ему ограничения. Данное исследование подчеркивает важность сочетания технологических инноваций со строгим обеспечением качества и этическими соображениями. В конечном счете, хотя LLM обладают огромными перспективами в плане расширения возможностей разработчиков-людей и демократизации доступа к программированию, их успешное внедрение требует продолжения

UDC 004.891:351

5.2.2. Mathematical, statistical and instrumental methods in economics (physical and mathematical sciences, economic sciences)

AUTOMATIC CODE GENERATION USING LARGE LANGUAGE MODELS (LLM): LIMITATIONS AND PROSPECTS

Kushnir Nadezhda Vladimirovna
Senior Lecturer at the Department of Information Systems and Programming
RSCI-SCIENCE INDEX SPIN code=6951-4012
kushnir.06@mail.ru
Kuban State Technological University, 350020, Moskovskaya, 2, Krasnodar, Russia

Zaikov Vladimir Polievktovich
Doctor of Economics and Candidate of Technical Sciences, Professor of the Department of Information Systems and Programming
РИНЦ-SCIENCE INDEX SPIN-code 9531-8128
zaykovv@yandex.ru
Kuban State Technological University, 350020 Moskovskaya, 2, Krasnodar, Russia

Litovka Natalia Vasilyevna
Senior Lecturer at the Department of Information Systems and Programming
РИНЦ-SCIENCE INDEX SPIN-code=9446-1581
natalilitovkav@yandex.ru
Kuban State Technological University, 350020 Moskovskaya, 2, Krasnodar, Russia

Yatskevich Evgeniy Sergeevich
2nd year Master's student
РИНЦ-SCIENCE INDEX SPIN-code=9985-2763
es_yatskevich@internet.ru
FGBOU VO "Kuban State Technological University", 350020, Moskovskaya, 2, Krasnodar, Russia

The current article examines the current state of LLM-based code generation, highlights both its practical achievements and its inherent limitations. This study highlights the importance of combining technological innovation with rigorous quality assurance and ethical considerations. Ultimately, although LLMs have great prospects in terms of empowering human developers and democratizing access to programming, their successful implementation requires continued research and responsible implementation in order to eliminate existing limitations and unlock their full potential

исследований и ответственного внедрения, чтобы устранить существующие ограничения и полностью раскрыть их потенциал

Ключевые слова: LLM, МОДЕЛЬ, ИСКУССТВЕННЫЙ ИНТЕЛЛЕКТ, РАЗРАБОТКА, ПРОГРАММИРОВАНИЕ

Keywords: LLM, MODEL, ARTIFICIAL INTELLIGENCE, DEVELOPMENT, PROGRAMMING

http://dx.doi.org/10.21515/1990-4665-210-023

Введение

Стремительное развитие искусственного интеллекта (ИИ) стало катализатором преобразующих достижений в различных областях, и разработка программного обеспечения стала заметным бенефициаром этих инноваций. Среди наиболее новаторских разработок — применение больших языковых моделей (LLM) для автоматической генерации кода — парадигма, которая использует архитектуры глубокого обучения для синтеза исполняемого кода из описаний на естественном языке или неполных фрагментов кода. Программы LLM, такие как OpenAI Codex, GitHub Copilot и Google AlphaCode, продемонстрировали беспрецедентное мастерство в создании синтаксически корректного кода, автоматизации повторяющихся задач программирования и даже преобразовании высокоуровневых постановок задач в функциональные реализации. Эти возможности проистекают из их обучения работе с обширными хранилищами открытого исходного кода, что позволяет им изучать сложные шаблоны и соглашения, присущие различным языкам программирования и фреймворкам.

Несмотря на их выдающиеся достижения, внедрение LLM в процесс генерации кода поднимает важные вопросы, касающиеся их надежности, контекстуального понимания и более широкого применения в практике разработки программного обеспечения. Текущие исследования выявляют существенные ограничения, в том числе склонность к созданию

семантически некорректного или неэффективного кода, трудности в обработке сложной логики и крайних случаев, а также уязвимости, связанные с внедрением уязвимостей в систему безопасности, таких как атаки путем внедрения или неправильный контроль доступа. Кроме того, этические соображения, связанные с правами интеллектуальной собственности, происхождением данных и алгоритмической предвзятостью, подчеркивают необходимость тщательного изучения наборов данных и методологий, лежащих в основе этих моделей. По мере того, как LLM все больше проникают в профессиональные и образовательные учреждения, становится необходимым решать эти проблемы, одновременно изучая стратегии повышения их надежности и адаптивности.

Основные возможности LLM в области генерации кода

Интеграция больших языковых моделей (LLM) в сферу разработки программного обеспечения привела к изменению парадигмы концептуализации, написания и оптимизации кода. Эти модели, основанные на сложных нейронных архитектурах, таких как transformers, продемонстрировали ряд возможностей, которые соответствуют требованиям современных задач программирования. Далее рассматриваются ключевые функциональные возможности LLM в области генерации кода.

Владение синтаксисом: создание корректного и структурированного кода. Одна из самых сильных сторон LLM заключается в способности создавать синтаксически корректный код на нескольких языках программирования. Эта способность возникает благодаря обучению работе с обширными массивами исходного кода, что позволяет усваивать грамматические правила и структурные паттерны различных парадигм

программирования — от объектно-ориентированных языков, таких как Java и Python, до функциональных языков, таких как Haskell.

Архитектура transformer, которая составляет основу LLM, превосходно отражает долгосрочные зависимости и иерархические отношения внутри последовательностей. В контексте кода это означает понимание вложенных структур (например, циклов, условных выражений и определений функций) и соблюдение правил синтаксиса, специфичных для конкретного языка.

Исследования, в которых оценивались такие модели, как Codex и AlphaCode, показали высокие показатели точности при генерации синтаксически корректных фрагментов кода для простых и умеренно сложных задач. Например, GitHub Copilot часто создает корректные реализации для распространенных программных конструкций, таких как понимание списков или вызовы API [1].

Однако, хотя синтаксическая корректность является важной вехой, она представляет собой только один аспект качества кода. Более сложной задачей остается приведение сгенерированного кода в соответствие с предполагаемой функциональностью.

Преобразование естественного языка в программный код: отличительной особенностью передовых LLM является способность переводить описания на естественном языке в исполняемый код. Эта возможность устраняет разрыв между нетехническими заинтересованными сторонами (например, менеджерами по продуктам или экспертами в предметной области) и разработчиками, обеспечивая более доступную коммуникацию и сотрудничество.

Двухмодальное обучение LLM, при котором они работают как с текстом на естественном языке, так и с исходным кодом, облегчает

межпредметную передачу знаний. Изучая сопоставления между описательными подсказками и соответствующими реализациями кода, эти модели приближают когнитивные процессы, связанные с интерпретацией спецификаций.

Оценки с использованием таких тестов, как HumanEval и MBPP (в основном, для решения базовых задач на Python), демонстрируют, что современные модели могут генерировать функциональный код на основе кратких формулировок задач. Например, получив запрос «Напишите функцию для вычисления факториала числа», LLM может надежно реализовать рекурсивную или итеративную реализацию [2].

Завершение кода и контекстные подсказки: LLM отлично справляются с доработкой кода в режиме реального времени и контекстными предложениями в интегрированных средах разработки (IDE). Эта функциональность не только ускоряет рабочие процессы разработки, но и снижает когнитивную нагрузку за счет автоматизации рутинных решений.

Авторегрессионный характер LLM позволяет прогнозировать последующие токены на основе предыдущего контекста, эффективно расширяя частичные фрагменты кода до полных выражений или блоков. Кроме того, тонкая настройка наборов данных, специфичных для предметной области, повышает способность адаптироваться к соглашениям и библиотекам, специфичным для проекта.

Было доказано, что такие инструменты, как GitHub Copilot и Amazon CodeWhisperer, повышают производительность разработчиков, предлагая соответствующие рекомендации по именам переменных, сигнатурам методов и даже целым функциям. Исследование, проведенное GitHub, показало, что разработчики, использующие Copilot, выполняли задачи на 55% быстрее по сравнению с ручным кодированием [3]. На

рисунке 1 представлен график, отображающий прямую зависимость эффективности работы с применением LLM.

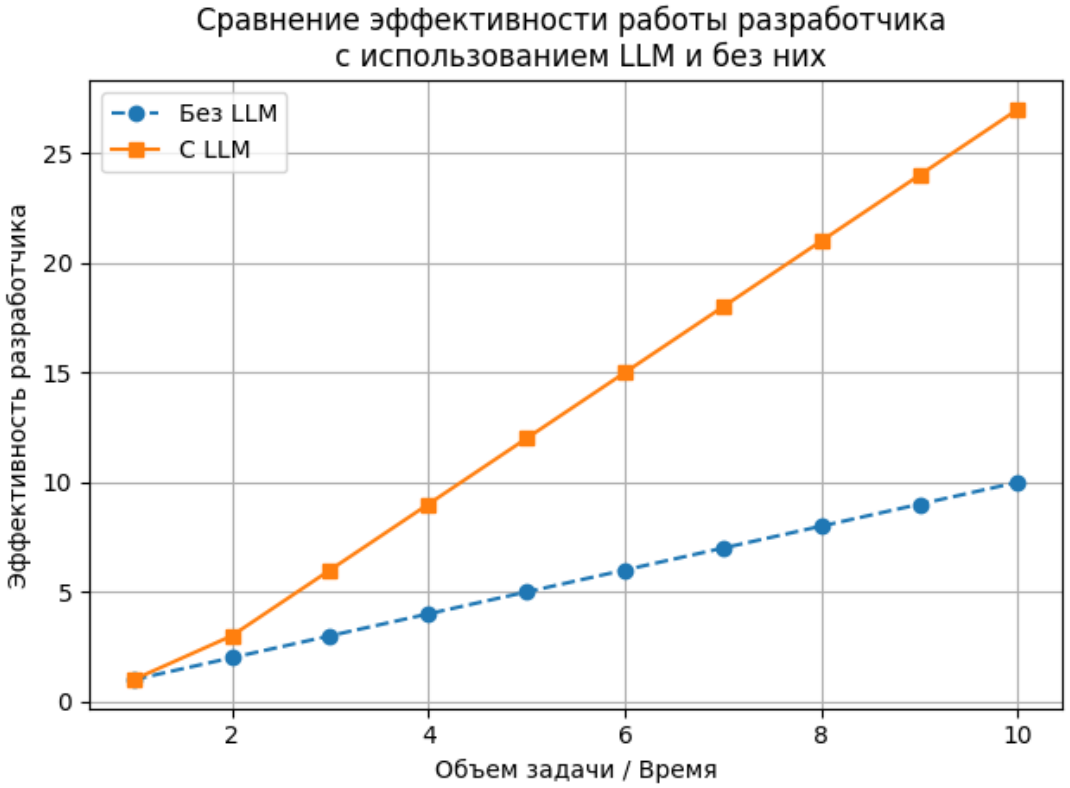

Рисунок 1 — График сравнения эффективности работы разработчика с использованием LLM и без них

Автоматизация повторяющихся задач: повторяющиеся задачи по кодированию, такие как написание стандартного кода, реализация CRUD-операций или настройка API, являются основными кандидатами для автоматизации с помощью LLM. Избавившись от этих рутинных действий, разработчики могут сосредоточиться на разработке более высокого уровня и решении проблем.

Возможности LLM по распознаванию образов позволяют идентифицировать повторяющиеся шаблоны и идиоматические выражения в кодовых базах. Например, модель может распознавать общие шаблоны

запросов к базе данных или конечные точки RESTful API и создавать эквивалентные реализации.

На практике LLM доказали свою эффективность в автоматизации таких задач, как создание каталогов проекта, инициализация файлов конфигурации и создание модульных тестов. Это привело к ощутимому повышению эффективности на ранних стадиях разработки.

Хотя такая автоматизация упрощает рабочие процессы, она также вызывает опасения по поводу чрезмерной зависимости от сгенерированного кода без надлежащего анализа, что может привести к появлению незначительных ошибок или неэффективности.

Многоязычная поддержка и межъязыковая транспиляция: современные программные системы часто используют несколько языков программирования, что требует полной функциональной совместимости между ними. LLM удовлетворяют эту потребность, поддерживая многоязычную генерацию кода и облегчая межъязыковую транспиляцию.

Система многозадачного обучения, используемая LLM, позволяет обобщать информацию на разных языках, выявляя общие абстракции и синтаксические параллели. Например, такие понятия, как циклы, условные выражения и структуры данных, имеют аналогичное представление в разных языках, что позволяет моделям эффективно сопоставлять их [4].

Демонстрация того, как LLM преобразуют скрипты Python в эквиваленты JavaScript или переводят SQL-запросы в операции с фреймворками данных Pandas, демонстрирует их универсальность. Такие возможности особенно ценны при модернизации устаревших систем и в средах программирования на разных языках.

Помощь в отладке и исправлении ошибок: помимо создания нового кода, LLM могут помочь в отладке существующих реализаций, выявляя

потенциальные проблемы и предлагая их исправления. Эта функциональность позволяет им широко использовать шаблоны, подверженные ошибкам, и стратегии их устранения, представленные в обучающих данных.

Предварительно обученные модели кодируют скрытые представления распространенных ошибок кодирования, таких как ошибки «не по порядку», разыменования нулевых указателей и несоответствия типов. При наличии ошибочного кода они могут определить вероятные причины и предложить корректирующие действия [5].

Экспериментальные оценки показывают, что LLM в определенных сценариях превосходят линтеры, основанные на правилах, предлагая более детальную диагностику, адаптированную к конкретным условиям. Например, модели могут рекомендовать провести рефакторинг чрезмерно сложной логики или оптимизировать неэффективные алгоритмы.

Заключение. Ключевые возможности Large Language Models (LLM) в области генерации кода представляют собой значительный шаг вперед в автоматизации и расширении процессов разработки программного обеспечения. От создания синтаксически точного кода и перевода естественного языка в исполняемые программы до автоматизации повторяющихся задач и помощи в отладке — эти модели продемонстрировали огромный потенциал для повышения производительности разработчиков и доступности программирования. Однако эффективность моделей сдерживается присущими им ограничениями, такими как проблемы с обеспечением семантической корректности, контекстуальной осведомленности и устойчивости к уязвимостям системы безопасности. По мере развития отрасли решающее значение будет иметь устранение этих недостатков с помощью усовершенствованных архитектур моделей, тонкой настройки в

зависимости от предметной области и гибридных подходов, которые интегрируют символические рассуждения. Сочетая инновации со строгой оценкой, LLM могут превратиться в надежные инструменты, которые не только оптимизируют рабочие процессы программирования, но и открывают новые возможности для сотрудничества человека и искусственного интеллекта в разработке программного обеспечения.

ЛИТЕРАТУРА

1. Остин Дж., Одена А., Най М. и др. Синтез программ с использованием больших языковых моделей [Электронный ресурс] / Дж. Остин, А. Одена, М. Най и др. – arXiv preprint arXiv:2108.07732, 2021. – Режим доступа: https://arxiv.org/abs/2108.07732 (дата обращения: 10.04.2025).

2. Ли Ю., Чой Д., Чунг Дж. и др. Генерация кода на уровне соревнований с помощью AlphaCode / Ю. Ли, Д. Чой, Дж. Чунг и др. // Science. – 2022. – Т. 378, № 6624. – С. 1092-1097.

3. Алламанис М., Барр Э. Т., Деванбу П., Саттон К. Обзор машинного обучения для анализа больших объемов кода и его "естественности" / М. Алламанис, Э. Т. Барр, П. Деванбу, К. Саттон // ACM Computing Surveys (CSUR). – 2018. – Т. 51, № 4. – С. 1-37.

4. Гудфеллоу И., Бенжио И., Курвиль А. Глубокое обучение / И. Гудфеллоу, И. Бенжио, А. Курвиль. – М.: МИТ Пресс, 2016. – 800 с.

5. Чен М., Творек Дж., Джун Х. и др. Оценка больших языковых моделей, обученных на коде [Электронный ресурс] / М. Чен, Дж. Творек, Х. Джун и др. – arXiv preprint arXiv:2107.03374, 2021. – Режим доступа: https://arxiv.org/abs/2107.03374 (дата обращения: 10.04.2025).

REFERENCES

1. Austin J., Audena A., Nai M. et al. Program synthesis using large language models [Electronic resource] / J. Austin, A. Oden, M. Nye, et al. – arXiv preprint arXiv:2108.07732, 2021. – Access mode: https://arxiv.org/abs/2108.07732 (date of request: 04/10/2025).

2. Lee Y., Choi D., Chung J. et al. Code generation at the competition level using AlphaCode / Yu. Li, D. Choi, J. Chung et al. // Science. – 2022. – Vol. 378, No. 6624. – pp. 1092-1097.

3. Allamanis M., Barr E. T., Devanbu P., Sutton K. Review of machine learning for analyzing large amounts of code and its "naturalness" / M. Allamanis, E. T. Barr, P. Devanbu, K. Sutton // ACM Computing Surveys (CSUR). – 2018. – Vol. 51, No. 4. – pp. 1-37.

4. Goodfellow I., Bengio I., Courville A. Deep learning / I. Goodfellow, I. Bengio, A. Courville, Moscow: MIT Press, 2016, 800 p.

5. Chen M., Creator J., June H. and others . Evaluation of large language models trained on code [Electronic resource] / M. Chen, J. Tworek, H. Jun et al. – arXiv preprint arXiv:2107.03374, 2021. – Available at: https://arxiv.org/abs/2107.03374 (date of access: 04/10/2025).