УДК 004.89

СРАВНИТЕЛЬНЫЙ АНАЛИЗ LLM ДЛЯ РАЗЛИЧНЫХ ДЛИН КОНТЕКСТА ПРИ ГЕНЕРАЦИИ И ОБСЛУЖИВАНИИ КОДА

♦♦♦♦♦

COMPARATIVE ANALYSIS OF LLMS FOR DIFFERENT CONTEXT LENGTHS IN CODE GENERATION AND MAINTENANCE

Можаровский Евгений Александрович
бакалавр,
Московский государственный университет
имени М.В. Ломоносова
mozharovsky_ea@rambler.ru

Аннотация. В данной статье проводится сравнительный анализ различных больших языковых моделей (large language models, LLM) в контексте генерации и поддержки программного кода при различных длинах контекста. Исследуются преимущества и недостатки моделей при обработке коротких и длинных контекстов, а также проводится оценка их эффективности в задачах кодогенерации и сопровождения программного обеспечения. Рассматриваются такие аспекты как точность предсказаний, скорость создания программного кода, масштабируемость и способность модели учитывать длинные зависимости в коде.

Ключевые слова: большие языковые модели (LLMs), генерация кода, сопровождение кода, длина контекста, GPT-4, Codex.

Mozharovsky Evgeny Alexandrovich
Bachelor,
Lomonosov Moscow State University
mozharovsky_ea@rambler.ru

Annotation. In this paper a comparative analysis of different large language models (LLMs) in the context of program code generation and maintenance at different context lengths is carried out. The advantages and disadvantages of the models in processing short and long contexts are investigated, and their effectiveness in the tasks of code generation and software maintenance is evaluated. Aspects such as prediction accuracy, code generation speed, scalability, and the ability to simulate for a long time depending on the code are considered.

Keywords: large language models (LLMs), code generation, code maintenance, context length, GPT-4, Codex.

Введение

Большие языковые модели (англ. Large Language Models – LLM) – это класс искусственного интеллекта (ИИ), который может понимать, интерпретировать и генерировать тексты. Основываясь на обучении и на наборах данных с миллиардами параметров, LLM способны создавать нужную информацию на уровне, идентичном человеческому. Это отличает их от традиционных моделей машинного обучения, применение которых ориентировано на помощь людям в задачах прогнозирования, а не на создание контента. Согласно прогнозу [1], мировой рынок LLM вырастет с 6,4 млрд долларов США в 2024 году до 36,1 млрд долларов США к 2030 году.

Развитие LLM за последние годы привело к значительным изменениям в области автоматизации программирования. Такие модели, такие как GPT-4 и Codex, демонстрируют высокие результаты в задачах генерации и сопровождения кода, что делает их важными инструментами для разработчиков. Несмотря на их потенциал, остается ряд вопросов, связанных с эффективностью при обработке контекстов разной длины.

Проблема длины контекста является одной из ключевых при работе с кодом, так как программное обеспечение часто включает сложные зависимости и структуры, которые требуют глубокого анализа и понимания. Короткие контексты, например, фрагменты кода длиной в несколько строк, могут быть легко обработаны большинством моделей. Однако при увеличении длины контекста, когда необходимо учитывать десятки или сотни строк кода, производительность и точность моделей могут существенно снижаться. Целью данной работы является проведение сравнительного анализа LLM при различных длинах контекста в задачах генерации и сопровождения кода.

Основная часть. Использование LLM для кодирования

LLM находят широкое применение в различных сферах деятельности, таких как образование, финансы, научные исследования и медицина [2]. Появление таких моделей открыло новую область программирования с использованием ИИ, предлагая разработчикам средства для оптимизации процессов кодирования и более эффективного

решения сложных проблем. LLM можно использовать для широкого спектра задач, таких как генерация и анализ кода, помощь с отладкой, рефакторингом и написанием тестовых примеров.

Одна из самых популярных LLM – **GPT-4**, которая была создана компанией OpenAI (США). Несмотря на то, что GPT-4 и ее предшественники GPT-2 и GPT-3 не являются моделями, разработанными специально в качестве помощника по кодированию, они хорошо справляются с разными задачами: генерация блоков кода, написание тестовых примеров и отладка ошибок. GPT-4 был обучен на основе данных, которые охватывают множество различных языков программирования и методов кодирования. Это позволяет модели понять широкий спектр логических потоков, правил синтаксиса и парадигм программирования, используемых разработчиками [3].

Еще одна модель от OpenAI – **Codex**. Она обучается на подмножестве публичных репозиториев Python на Github для генерации кода из документальных строк. Codex генерирует 100 версий программы методом повторной выборки для заданного описания, что позволяет получить рабочее решение для большинства проблем, прошедших модульные тесты.

Модель **GitHub Copilot** была создана GitHub (США) и является специально обученной LLM с использованием данных для помощи разработчикам с целью повышения эффективности и производительности. Построена на основе модели GPT-4. Обучается с использованием данных из общедоступных репозиториев кода, включая сам GitHub.

Модель американской компании Meta (США) **CodeLlama** является современной LLM, предназначенной для кодогенерации и задач на естественном языке, связанных с кодом. Это специализированная версия Llama 2, созданная путем ее глубокого обучения на наборах данных, специфичных для кода, с более длительной выборкой большего количества информации.

CodeT5 представляет собой модель, разработанную Google AI (США), которая улучшает понимание и генерацию кода с использованием идентификаторов, назначаемых разработчиком. Такой метод включает в себя задачу предварительного обучения для различения токенов кода, которые являются идентификаторами. Модель охватывает задачу двойной генерации, в которой используются комментарии к коду, написанные специалистом.

Анализ эффективности LLM при различных длинах контекста и стратегии их повышения

Сравнительный анализ предполагает оценку моделей по нескольким критериям: генерация кода (учитывая описание проблемы, модель генерирует соответствующий код), сопровождение (модель предлагает улучшения или исправления существующего кода) и изменчивость длины контекста. Метрики для оценки включают в себя точность, эффективность и масштабируемость.

Для анализа производительности моделей при работе с **короткими контекстами** (фрагментами кода длиной до 20 строк) используют стандартные тестовые наборы данных, включающие типичные задачи кодирования, такие как написание функций, исправление ошибок и создание тестов. Языковые модели GPT-4, Codex, Copilot и другие LLM, как правило, показывают высокую точность предсказаний и скорость генерации кода [4]. Они также успешно справляются с задачами, требующими понимания синтаксиса и семантики программного кода, что особенно важно при написании и исправлении функций. Codex, GPT-4 и Copilot демонстрируют отличные результаты, особенно в задачах, связанных с автоматизацией тестирования и генерацией вспомогательных функций. Важно отметить, что перечисленные модели показывают схожую производительность при обработке коротких контекстов, что делает их взаимозаменяемыми в этих сценариях.

При увеличении длины контекста до 50–100 строк кода возникают дополнительные сложности, связанные с необходимостью учета более глубоких зависимостей и контекста выполнения [5]. Для оценки производительности моделей в этих условиях используют задачи, включающие написание более сложных функций и классов, а также рефакторинг кода. Часто для LLM используют точную настройку – процесс исполь-

зования предварительно обученной модели и ее дальнейшей подготовки на наборе данных для конкретной предметной области. При анализе обученных и обычных языковых моделей [6] было выявлено, что точная настройка позволяет повысить точность данных при увеличении длины контекста (рис. 1).

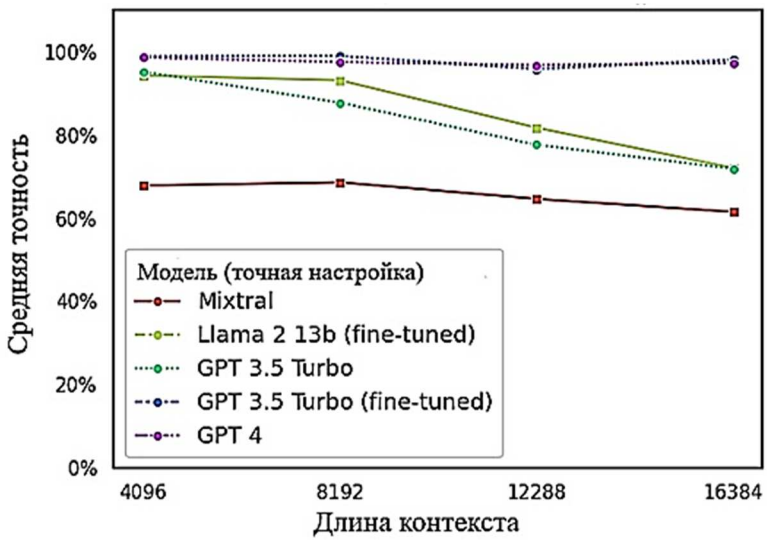

Рисунок 1 – Точность модели в зависимости от длины контекста

Улучшение модели Llama 2 с исходной длиной в 4 тыс токенов за счет точной настройки данных для длины контекста до 16 тыс токенов значительно повышает ее качество, превосходя производительность немодифицированного GPT-3.5. GPT-4 может поддерживать длину контекста до 128 тыс токенов и обеспечивает оптимальную производительность для всех протестированных длин контекста до 16 тыс.

Длинные контексты. Для анализа работы моделей с контекстами свыше 100 строк кода используют сложные сценарии, такие как написание модулей и компонентов, а также интеграция различных частей программного обеспечения. Эти задачи требуют не только понимания синтаксиса и семантики, но и способности модели учитывать долгоиграющие зависимости и структурные особенности кода.

При проведении тестов на выборке LLM с длиной контекста от 2 до 100 тыс токенов американские исследователи [7] выявили, что модели постоянно не могут точно воспроизвести информацию в середине контекста. Однако они лучше справлялись с обработкой данных в начале контекста (первичное смещение) и в конце (смещение новизны).

В американском исследовании LLM [8], GPT-4 столкнулась с трудностями при обработке длинных контекстов, что проявилось в снижении точности результатов и увеличении времени выполнения задач. Это связано с ограничениями модели по объему памяти и способности удерживать контекст на протяжении большого количества строк кода. Codex показывает лучшие результаты, сохраняя более стабильную производительность и точность при работе с длинными фрагментами кода, что обусловлено улучшенной архитектурой модели и оптимизацией для обработки долгосрочных зависимостей.

Сравнение производительности языковых моделей при различных длинах контекста показало, что все LLM обладают высокой точностью и эффективностью при работе с короткими и средними контекстами. Однако при увеличении длины контекста точно настроенные модели демонстрирует более стабильные результаты, что делает данную модификацию предпочтительным выбором для задач, связанных с обработкой больших объемов кода.

Анализ демонстрирует, что LLM обладают значительным потенциалом для автоматизации процессов программирования. Однако их производительность зависит от длины контекста, что следует учитывать при выборе инструмента для конкретных за-

дач. Необходимость дальнейшего совершенствования LLM для работы с длинными контекстами приводит разработке новых методов оптимизации. Точность и масштабируемость LLM в задачах генерации кода и обслуживания можно повысить с помощью нескольких стратегий (табл. 1).

Таблица 1 – Стратегии повышения эффективности LLM в кодировании [9]

Стратегия	Метод	Описание
Увеличение данных	Расширение кода	Преобразование и мутация кода для создания дополнительных обучающих примеров могут помочь модели изучить больше шаблонов кодирования
	Синтетические данные	Создание примеров синтетического кода, имитирующих реальные сценарии кодирования, может дополнить существующие наборы данных
Точная настройка с использованием конкретных данных	Наборы данных предметной области	Точная настройка наборов данных, специфичных для конкретных областей (например, веб-разработка, наука о данных), может повысить производительность модели в этих областях
	Данные по обслуживанию	Включение наборов данных, таких как задачи рефакторинга и отладки, помогает модели понять общие практики обслуживания
	Постоянное обучение	Реализация стратегий, при которых модель обновляется, может поддерживать ее актуальность с учетом новейших практик и технологий кодирования
Расширение кодовых данных предварительного обучения	Более широкий языковой охват	Включение кода из широкого спектра языков программирования и парадигм может помочь моделям лучше обобщать различные задачи
	Качество кода	Использование высококачественных, хорошо документированных баз кода гарантирует, что модели изучат лучшие практики и стандарты кодирования
	Реальные проекты	Включение данных из проектов, репозиториев с открытым исходным кодом и приложений может улучшить понимание сценариев кодирования
Понимание иерархии и структуры	Абстрактные синтаксические деревья (AST).	Использование AST может предоставить модели структурированное представление кода, способствуя лучшему пониманию и генерации

Повышение точности и масштабируемости LLM при создании и обслуживании кода требует многогранного подхода, который включает в себя ряд стратегий. Разработчики могут применять данные методы комплексно, чтобы создавать более эффективные и надежные LLM. Это позволит значительно расширить возможности использования языковых моделей в проектах разработки программного обеспечения и повысить их эффективность в задачах генерации и сопровождения кода.

Выводы

Сравнительный анализ LLM в задачах генерации и поддержки программного кода при различных длинах контекста позволил выявить особенности и ограничения языковых моделей. Большинство LLM демонстрирует высокую точность и гибкость в работе с короткими и средними контекстами. Однако при увеличении длины контекста их производительность может снижаться, что требует дополнительных методов оптимизации и повышения эффективности обработки длинных зависимостей в коде. Важность выбора подходящей модели может зависеть от конкретных целей проекта. Для задач, требующих обработки большого объема кода и учета длинных зависимостей, возможно использование различных специализированных техник.

Литература

1. Large Language Model (LLM) Market worth $36.1 billion by 2030 / MarketsandMarkets. – URL : https://www.marketsandmarkets.com/PressReleases/large-language-model-llm.asp (date of application 07.07.2024).
2. Андрейченко А.Е. Перспективы применения больших языковых моделей в здравоохранении / А.Е. Андрейченко, А.В. Гусев // Национальное здравоохранение. – 2023. – № 4(4). – С. 48–55.

3. Малыхин Н.И. Оптимизация процесса валидации данных в документах с использованием Java Reflection и Apache Commons / Н.И. Малыхин // Актуальные исследования. – 2019. – № 1(1).
4. A Comprehensive Overview of Large Language Models / H. Naveed, A.U. Khan, S. Qiu [et al.] // arXiv:2307.06435. Preprint. 2024.
5. Тюменцев Д.В. Автоматизация тестирования в Devops: подходы и лучшие практики / Д.В. Тюменцев // Международный журнал гуманитарных и естественных наук. – 2024. – № 2-2(89). – С. 156–159.
6. Fine-tuning LLMs for longer context and better RAG systems, 2024 / Anyscale. – URL : https://www.anyscale.com/blog/fine-tuning-llms-for-longer-context-and-better-rag-systems (date of application 10.07.2024).
7. Lost in the Middle: How Language Models Use Long Contexts / N.F. Liu, K. Lin, J. Hewitt [et al.] // TACL. – 2023. – 18 p.
8. Evaluating Large Language Models Trained on Code / M. Chen, J. Tworek, H. Jun [et al.] // arXiv preprint arXiv:2107.03374. – 2021.
9. Framework for evaluating code generation ability of large language models / S. Yeo, Y.S. Ma, S.C. Kim [et al.] // ETRI Journal. – 2024. – Vol. 46(1). – P.106–117.

References

1. Large Language Model (LLM) Market worth $36.1 billion by 2030 / MarketsandMarkets. – URL : https://www.marketsandmarkets.com/PressReleases/large-language-model-llm.asp (date of application 07.07.2024).
2. Andreichenko A.E. Prospects for the Use of Large Language Models in Healthcare / A.E. Andreichenko, A.V. Gusev // National Healthcare. – 2023. – № 4(4). – P. 48–55.
3. Malykhin N.I. Optimization of Data Validation Processes in Documents Using Java Reflection and Apache Commons / N.I. Malykhin // Relevant Research. – 2019. – № 1(1).
4. A Comprehensive Overview of Large Language Models / H. Naveed, A.U. Khan, S. Qiu [et al.] // arXiv:2307.06435. Preprint. 2024.
5. Tyumentsev D.V. Automation of Testing in DevOps: Approaches and Best Practices / D.V. Tyumentsev // International Journal of Humanities and Natural Sciences. – 2024. № 2-2(89). – P. 156–159.
6. Fine-tuning LLMs for longer context and better RAG systems, 2024 / Anyscale. – URL : https://www.anyscale.com/blog/fine-tuning-llms-for-longer-context-and-better-rag-systems (date of application 10.07.2024).
7. Lost in the Middle: How Language Models Use Long Contexts / N.F. Liu, K. Lin, J. Hewitt [et al.] // TACL. – 2023. – 18 p.
8. Evaluating Large Language Models Trained on Code / M. Chen, J. Tworek, H. Jun [et al.] // arXiv preprint arXiv:2107.03374. – 2021.
9. Framework for evaluating code generation ability of large language models / S. Yeo, Y.S. Ma, S.C. Kim [et al.] // ETRI Journal. – 2024. – Vol. 46(1). – P. 106–117.