DOI 10.37539/2949-1991.2026.36.1.011

Толстых Роман Андреевич, Магистрант,
АНО ВО "Российский новый университет"
Tolstykh Roman Andreevich, Master's Student,
ANO VO "Russian New University"

ТРАССИРУЕМАЯ RAG-ГЕНЕРАЦИЯ ФРОНТЕНД-КОДА: СЛОЙ ОБОСНОВАНИЙ И ДОПУЩЕНИЙ КАК МЕХАНИЗМ ПРОВЕРЯЕМОСТИ ВЫВОДА LLM
TRACEABLE RAG-BASED FRONTEND CODE GENERATION: EVIDENCE/ASSUMPTIONS LAYER AS A MECHANISM FOR LLM OUTPUT VALIDATION

Аннотация. Системы генерации кода, построенные на архитектуре с использованием механизма Retrieval-Augmented Generation (RAG), приобретают все большее распространение и становятся стандартом. Однако ограничение таких систем заключается в том, что они функционируют как "черные ящики", скрывая происхождение решений LLM. Настоящая работа предлагает методологию Трассируемый RAG, которая внедряет проверяемый слой, позволяющий классифицировать каждое решение, принятое LLM при генерации, либо как аргументированное источником, либо как допущение с независимой оценкой уверенности. Данная методология обеспечивает проверяемость и объяснимость процесса генерации кода.

Abstract. Retrieval-Augmented Generation (RAG)-based systems are becoming the standard for code generation systems. A fundamental limitation of such systems is their lack of explainability, which stems from the obscured provenance of LLM outputs. The paper proposes the Traceable RAG methodology, which integrates an evidence/assumptions verification layer to classify every LLM output. This ensures the traceability and explainability of LLM-based workflows.

Ключевые слова: RAG, Большие языковые модели, LLM, Retrieval-Augmented Generation, генерация кода.

Keywords: RAG, Large Language Models, LLM, Retrieval-Augmented Generation, code generation.

Введение

В настоящее время разработка ПО претерпевает кардинальные изменения, связанные с внедрением в процесс больших языковых моделей и инструментов, использующих механизмы RAG [1]. Уже сейчас решения для автоматизации написания кода движутся в сторону автономной генерации целых программных модулей на основе мультимодальных спецификаций и не ограничиваются лишь классическим автодополнением.

При рассмотрении возможности применения генеративного ИИ в области фронтенд-разработки можно выявить ряд особенностей, отличающих эту область от генерации бэкенд-кода. Артефакты, возникающие в результате программирования пользовательских интерфейсов, обладают высоким уровнем гетерогенности. Код должен одновременно соответствовать функциональным требованиям, правилам и стандартам используемой дизайн-системы, нормам доступности, требованиям производительности и безопасности [3]. Галлюцинации LLM при генерации кода в данном случае становятся критичными, поскольку они ведут к фактическим ошибкам пользовательских путей и визуальным регрессиям [4].

Центральной проблемой систем, архитектура которых включает механизмы RAG, является отсутствие трассируемости. Механизм традиционных RAG-конвейеров подразумевает конкатенацию релевантных документов в контекстное окно, но функционирует

непрозрачно, скрывая атрибуцию тех или иных решений LLM [5]. В результате разработчик, использующий систему, при получении результата не может детерминировано установить, почему для генерации кода LLM выбрала конкретный паттерн или алгоритм.

В настоящей работе предлагается концептуальная модель Трассируемый RAG, которая включает механизм проверки связи между генерируемыми артефактами и исходными требованиями посредством введения промежуточного слоя обоснований и допущений.

Методы

Предпосылками для разработки методологии Трассируемого RAG стали исследования в области атрибуции утверждений, оценки вероятности и управления дрейфом знаний. В сложных инженерных задачах, где предпочтительнее непараметрические методы, которые позволяют проверять результат относительно внешних источников, атрибуция стабильно движется от простой верификации ссылок на документ к проверке отдельных фактов [6]. В этом контексте следует рассмотреть фреймворк Claimify, который реализует разбиение ответа на отдельные утверждения с целью повышения точности верификации [7, 8]. Для оценки того, насколько сгенерированный ответ основан на предоставленном контексте, используются метрики Faithfulness (достоверность) и Groundedness (обоснованность) [9, 10]. Однако адаптация этих концептов и метрик к предметной области генерации фронтенд-кода требует высокой согласованности функциональной, визуальной и структурной составляющих.

Стоит отметить, что разработка пользовательских интерфейсов характеризуется высоким уровнем волатильности [11]. Это обусловлено коротким жизненным циклом JavaScript решений: выпускаются новые библиотеки, которые задают собственные архитектурные паттерны, фреймворки выпускают мажорные обновления, которые нарушают обратную совместимость. Таким образом, на сегодняшний день невозможно выделить единый устоявшийся подход к написанию фронтенд-кода. В связи с этим долгосрочное хранение знаний о предметной области в векторном индексе становится малоэффективным, поскольку происходит рассинхронизация между содержимым индекса и актуальным состоянием внешнего мира. Такое явление называется дрейфом базы знаний (Knowledge Base Drift) [12]. Оно демонстрирует, что при устаревании индекса более чем на 6 месяцев качество RAG-ответов падает на 15-25%. Управление этим процессом является ключевым преимуществом RAG перед fine-tuning [13], однако требует дополнительных инженерных практик.

Таким образом, для поддержания качества генерации кода в условиях быстрых изменений условий внешнего мира необходим механизм, который будет адаптирован не только к оперативному обновлению контекста, но и к постоянной внутренней проверке решений актуальному состоянию.

Адаптируя эти идеи для предметной области генерации кода, в настоящей работе предлагается модель, где утверждением считается атомарное архитектурное решение, например, о выборе API, алгоритма или компонента. Решения извлекаются из абстрактного синтаксического дерева (AST) сгенерированного артефакта. При этом учитываются импорты, как библиотек, так и других компонентов, типы свойств и состояний, правила валидации, CSS-классы и стили, а также функции и обработчики. В качестве доказательства рассматривается проверяемая связь решения с контекстом, представленная в виде пары (решение, фрагмент контекста), подтверждающая, что решение обосновано источником. Допущением является решение, для которого не найдено доказательство. Допущения неизбежны и обуславливаются неопределенностью или новизной требований. Для оценки надежности решения, независимо от наличия доказательств, вводится понятие уверенности, которое классифицируется как высокое, среднее или низкое на основе консистентности между независимыми генерациями.

Практическая реализация процесса Трассируемого RAG представлена последовательной цепочкой действий. На первом этапе из версионированной векторной базы

знаний извлекаются наиболее релевантные фрагменты с метаданными. На следующем этапе, перед генерацией кода, модель формирует структурированный план, который представляет из себя список ключевых решений. Затем следует этап атрибуции, на котором происходит связывание решений с конкретными фрагментами извлеченного на первом этапе контекста. В случае, если для решения не найдена ссылка, такое решение помечается как предположение с указанием уровня уверенности, который основывается на сходстве между несколькими генерациями. Наконец, на основе классифицированного списка решений происходит генерация кода. На финальном этапе также формируются выходные артефакты, которые помимо самого кода включают спецификацию и отчёт о трассируемости в машиночитаемом виде.

Результаты

Оценку практической эффективности модели проводили на выборке из 10 задач по генерации UI-компонентов. Элементы интерфейса в задачах были представлены шестью формами (регистрационная, контактная, поисковая, настроек, обратной связи, платежная) и четырьмя таблицами (каталог товаров, отчет по метрикам, список пользователей, журнал событий). База знаний содержала документацию по React, дизайн токены и гайдлайны по доступности. В ходе эксперимента проводилось сравнение классического RAG-конвейера и Трассируемого RAG со слоем доказательств/допущений с предварительной декомпозицией решений и их классификацией. С целью оценки воспроизводимости выполнялись 5 независимых генераций каждой задачи.

Для качественной оценки эффективности трассируемого RAG были введены метрики, которые характеризуют покрытие доказательствами и согласованность кода со спецификацией. Информация о метриках представлена в Таблице 1.

Таблица 1.

Метрики оценки Трассируемого RAG

Метрика	**Что измеряет**	**Способ измерения**
Unsupported Decision Rate (UDR)	Доля решений без доказательств	Отношение числа предположений к общему числу решений
Evidence Coverage (EC)	Покрытие доказательствами	Сумма связей между решениями и извлеченными фрагментами из базы знаний деленная на количество решений
Spec-Code Consistency (SCC)	Согласованность спецификации и кода	Рассчитывается через структурное сопоставление: для каждого поля спецификации проверяется наличие соответствующего элемента в AST кода
Reproducibility (REP)	Воспроизводимость	Сходство Жаккара между нормализованными AST, полученными в нескольких независимых генерациях

Результаты, полученные в ходе эксперимента сведены в Таблицу 2. В среднем на задачу извлекалось 14,3 решения для форм и 11,7 решения для таблиц. В случае Трассируемого RAG произошло значительное снижение доли неподтверждённых решений: UDR уменьшился с 0,47 ± 0,11 до 0,23 ± 0,08. Покрытие доказательствами выросло с 0,53 ± 0,12 до 0,81 ± 0,09, а согласованность спецификации и кода с 0,71 ± 0,14 до 0,89 ± 0,07. Воспроизводимость нормализованных AST выросла с 0,64 ± 0,18 до 0,82 ± 0,11. Данные результаты свидетельствуют о том, что явная декомпозиция решений и целенаправленный поиск доказательств повышают соответствие кода спецификации и снижают стохастичность при генерации кода LLM.

Таблица 2.

Результаты эксперимента

Метрика	Классический RAG	Трассируемый RAG
UDR (↓ лучше)	0,47 ± 0,11	0,23 ± 0,08
EC (↑ лучше)	0,53 ± 0,12	0,81 ± 0,09
SCC (↑ лучше)	0,71 ± 0,14	0,89 ± 0,07
REP (↑ лучше)	0,64 ± 0,18	0,82 ± 0,11

Полученные результаты демонстрируют, что Трассируемый RAG не только улучшает ключевые показатели качества кода, но и превращает саму генерацию в контролируемый и проверяемый процесс, в котором каждое решение имеет документально подтвержденное происхождение.

Заключение

В работе предложена методология Трассируемого RAG для генерации фронтенд-кода, которая включает проверяемый слой доказательств и допущений. Были введены ключевые понятия, такие как решение, доказательство и допущение, а также предложены основные метрики оценки трассируемости RAG решений. Пилотный эксперимент на 10 задачах продемонстрировал значительное снижение процента неподтвержденных решений и рост соответствия кода спецификации.

Тем не менее, методология обладает некоторыми ограничениями, которые включают зависимость от качества поиска, повышенную латентность, обусловленную этапом декомпозиции, и необходимость поддержания актуальности базы знаний. Дальнейшие исследования будут направлены на переход к точечной атрибуции и интеграции с инструментами автоматической верификации.

Список литературы:

1. Lewis P., Perez E., Piktus A. и др. Retrieval-Augmented Generation for Knowledge-Intensive NLP Tasks // Advances in Neural Information Processing Systems. – 2020. – Vol. 33. – P. 9459-9474. DOI: 10.48550/arXiv.2005.11401.

2. Chen M., Tworek J., Jun H. и др. Evaluating Large Language Models Trained on Code // arXiv:2107.03374 [cs.CL]. – 2021. – (Preprint). URL: https://arxiv.org/abs/2107.03374 (дата обращения: 19.01.2026).

3. Vaithilingam P., Zhang T., Glassman E. L. Expectation vs. Experience: Evaluating the Usability of Code Generation Tools // In CHI Conference on Human Factors in Computing Systems Extended Abstracts (CHI ’22 Extended Abstracts); 29 апреля – 5 мая 2022 г.; Нов-Орлеан (США). – Нью-Йорк: ACM, 2022. – С. 1-7. DOI: 10.1145/3491101.3519665.

4. Ji Z., Lee N., Frieske R. и др. Survey of Hallucination in Natural Language Generation // ACM Computing Surveys. – 2023. – Vol. 55, no. 12. – P. 1-38. DOI: 10.1145/3571730.

5. Gao Y., Xiong Y., Gao X. и др. Retrieval-Augmented Generation for Large Language Models: A Survey // arXiv:2312.10997 [cs.CL]. – 2023. – (Preprint). URL: https://arxiv.org/abs/2312.10997 (дата обращения: 19.01.2026).

6. Gao L., Dai Z., Pasupat P. и др. RARR: Researching and Revising What Language Models Say, Using Language Models // In Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (ACL 2023). – 2023. – P. 16477-16508. DOI: 10.18653/v1/2023.acl-long.910.

7. Weller O., Marone M., Weir N. и др. Claimify: Extracting High-Quality Claims from Language Model Outputs // Microsoft Research. – 2024. – (Блог компании; исследование принято на ACL 2025). URL: https://www.microsoft.com/en-us/research/blog/towards-effective-extraction-and-evaluation-of-factual-claims/ (дата обращения: 19.01.2026).

8. Bohnet B., Pham V. Q., Berber Sardinha T. и др. Attributed Question Answering: Evaluation and Modeling for Attributed Large Language Models // arXiv:2212.08037 [cs.CL]. – 2022. – (Preprint). DOI: 10.48550/arXiv.2212.08037.

9. Es S., James J., Espinosa-Anke L., Schockaert S. RAGAS: Automated Evaluation of Retrieval Augmented Generation // In Proceedings of the 18th Conference of the European Chapter of the Association for Computational Linguistics: System Demonstrations (EACL 2024). – 2024. – P. 150-163. DOI: 10.18653/v1/2024.eacl-demo.16.

10. Liu N. F., Zhang T., Liang P. Evaluating Verifiability in Generative Search Engines // Findings of EMNLP. – 2023. – P. 7001-7025. DOI: 10.48550/arXiv.2304.09848.

11. Coblenz M., Aldrich J., Myers B. A., Sunshine J. Interdisciplinary Programming Language Design // In Proceedings of the 2018 ACM SIGPLAN International Symposium on New Ideas, New Paradigms, and Reflections on Programming and Software (Onward! '18); 8 ноября 2018 г.; Бостон (США). – Нью-Йорк: ACM, 2018. – С. 133-146. DOI: 10.1145/3276954.3276965.

12. Kandpal N., Deng H., Roberts A. и др. Large Language Models Struggle to Learn Long-Tail Knowledge // In Proceedings of the 40th International Conference on Machine Learning (ICML 2023). – 2023. – P. 15696-15707. URL: https://proceedings.mlr.press/v202/kandpal23a.html (дата обращения: 19.01.2026).

13. Ovadia O., Brief M., Mishaeli M., Elisha O. Fine-Tuning or Retrieval? Comparing Knowledge Injection in LLMs // arXiv:2312.05934 [cs.CL]. – 2023. – (Preprint). DOI: 10.48550/arXiv.2312.05934.