5.1. RAG vs. Fine-tuning

Цель и контекст

Внедрение ИИ-ассистентов в разработку кода часто начинается с вопроса: что проще — заставить модель «знать» кодовую базу через дообучение или снабдить её контекстом во время запроса? В этой секции сравним два подхода — Retrieval-Augmented Generation (RAG) и `fine-tuning` — с точки зрения задач AI-разработчика, работающего с исходным кодом.

Retrieval-Augmented Generation (RAG)

«Открытая книга»: RAG — это стратегия, при которой модель получает доступ к актуальной информации извне, например к вашей кодовой базе, и генерирует ответ, опираясь на эти факты. Исследования по RAG показывают, что эта техника эффективно интегрирует свежие данные, борется с галлюцинациями и повышает качество ответа, особенно в специализированных доменах[1].

Составные модули:

Типичный RAG-конвейер включает классификацию запроса, извлечение (`retrieval`) релевантных документов, переранжирование, упаковку (`repacking`) и суммаризацию[1]. Такой подход требует больше инженерных усилий, чем использование модели «из коробки», но его легко поддерживать — для улучшения качества достаточно расширять и улучшать источники для поиска[1].

Преимущества для кода:

- **Минимизация галлюцинаций.** Внешний поиск снабжает модель точным контекстом и уменьшает вероятность выдуманных ответов. Лилиан Вэн отмечает, что галлюцинации — это ответы, не основанные на контексте или знании о мире[2], и что именно RAG позволяет снабдить модель поддерживающей информацией[2].
- **Актуальность.** Кодовая база и документация постоянно меняются. RAG позволяет подключить актуальные файлы или репозитории, не требуя переобучения модели, что ускоряет внедрение и обновление[1].
- **Локализация кода.** При задачах вроде «почему функция падает?» или «как изменить метод» модель может извлекать точные фрагменты кода, README и тесты, анализировать их и выдавать полезные советы.

Недостатки:

- **Инженерная сложность.** Качество RAG сильно зависит от выбора алгоритмов поиска, размера и структуры «кусков» (`chunks`), качества векторных эмбеддингов и методик переупаковки[1]. Плохо настроенный конвейер может привести к пропуску нужных фрагментов.
- **Задержка.** В цепочке появляются поиск и обработка данных, что увеличивает время ответа. Оптимальные настройки — компромисс между точностью и скоростью.

Оценка качества RAG:

- **RAGAS и DeepEval.** Фреймворк `DeepEval` предназначен для тестирования LLM-систем и включает ряд готовых метрик для RAG: `Answer Relevancy`, `Faithfulness`, `Contextual Recall/Precision` и `RAGAS`[3]. DeepEval позволяет запускать эти метрики локально, писать `unit-` для RAG-пайплайнов и отслеживать эволюцию модели[3].
- **Метрики для кода.** Для задач с кодом важно проверять, что ответ содержит корректные ссылки на строки кода («contextual recall») и что итоговая рекомендация соответствует `retrieved`-контексту («faithfulness»). Эти метрики помогают избежать галлюцинаций при генерации патчей или советов.

Fine-tuning (Дообучение)

Суть: `Fine-tuning` — это обучение модели на ваших данных с изменением её параметров. В классическом виде дообучение требует большого объёма GPU-памяти и долгих тренировок. Методика `LoRA` предлагает обучать лишь низкоранговые матрицы в слоях трансформера, что сокращает число обучаемых параметров в 10 000 раз и уменьшает требования к памяти в 3 раза[4]. При этом достигается сопоставимое качество по сравнению с полной настройкой модели.

Проблемы:

- **Затраты и ресурсы.** Даже с `LoRA/QLoRA` для дообучения частично замороженной модели требуются мощные GPU и качественный датасет. Поддерживать отдельную `fine-tuned`-модель под каждую кодовую базу дорого и трудно масштабировать.
- **Узость знаний и переобучение.** Дообучение прививает модели знания из конкретной выборки. Сырая или ограниченная выборка кода может привести к переобучению и потере общих навыков. Исследования показывают, что простое дообучение на новой информации сначала снижает частоту галлюцинаций, но затем она вновь растёт[5].
- **Галлюцинации и ошибки.** Лилиан Вэн отмечает, что модели учатся примерам с новой информацией медленнее, из-за чего дообучение на фактах может усиливать склонность к галлюцинациям[2].

Когда полезно:

- **Стилизация и формат.** Дообучение оправдано, если нужно адаптировать стиль генерации (например, соблюдать внутренние стандарты `code review`) или создать локальный продукт на базе маленькой модели, когда нет доступа к API.

- **Специализированные языки.** Если кодовая база использует редкий DSL (`domain-specific language`) или содержит сложные доменные правила, краткое дообучение может улучшить качество синтаксиса и семантики.

Этичный и технико-экономический контекст: по мере появления новых, более качественных базовых моделей, `fine-tuned`-варианты быстро устаревают. После обновления базовой модели приходится пересматривать все кастомные веса, что увеличивает затраты на поддержку.

Выводы и рекомендации для AI-разработчиков

- **Выбор метода:** для задач сопровождения и анализа кода предпочтителен **RAG**. Он позволяет выдавать свежие и подкреплённые ссылками ответы, не модифицируя параметры модели. `Fine-tuning` выгодно использовать лишь для адаптации стиля или обучения маленьких локальных моделей при отсутствии API.
- **Инвестиции в данные:** вместо дорогого дообучения сосредоточьтесь на подготовке знаний для `retrieval`-слоя: индексируйте код, документацию, тесты и логи, продумайте схему разбиения на фрагменты и присвойте каждому полезные метаданные. Хорошая база и качественные эмбеддинги существенно повышают качество ответов.
- **Контроль качества:** внедряйте автоматическую проверку RAG-ответов. Метрики вроде `Answer Relevancy`, `Contextual Recall` и `Faithfulness`[3] помогают количественно оценивать, насколько хорошо модель использует `retrieved`-контекст и соответствует ему. Инструменты `DeepEval` позволяют интегрировать эти проверки в CI/CD и писать `unit`-для задач на код[3].
- **Продолжайте исследовать:** RAG-пайплайны постоянно развиваются, включая мультимодальные `retrieval`-модели и стратегии «retrieval as generation»[1]. Современные методы пост-обучения (RLHF, DPO и т.д.) помогают снизить галлюцинации, но не отменяют необходимость контроля качества. Совмещая RAG и аккуратные ограничения на модель, можно получить мощного ассистента для инженеров.

Источники

[1] Источник: arxiv.org
[2] Источник: lilianweng.github.io
[3] Источник: raw.githubusercontent.com
[4] Источник: arxiv.org
[5] Источник: cdn.openai.com