Методология оценки качества контекстного извлечения в Scaffold

Цель исследования

Целью является построение стратегии оценки качества работы сервиса Scaffold, задача которого не генерировать код, а извлекать релевантный контекст из кодовой базы и передавать его в LLM для дальнейшей генерации ответа.

Наш пайплайн проверки

1. Используем заранее подготовленный синтетический проект: https://github.com/Beer-Bears/scaffold-testgroup.
2. Формулируем конкретный вопрос к коду, например: Почему не работает connect_db().
3. Вручную составляем эталон: список релевантных фрагментов и зависимостей, необходимых для ответа.
4. Отправляем запрос в Scaffold, получаем от него графово-векторный контекст.
5. Сравниваем возвращённый контекст с эталоном по метрикам.
6. Записываем результаты, формируем таблицу и делаем анализ качества.

Автоматическое тестирование графогенерации

- Используем заранее заготвленный репозиторий с заранее известной архитектурой.
- Пишем скрипт, сравнивающий построенный граф с эталоном.
- Делаем отчет: где несовпадения, недостающие узлы или связи.
- Эти тесты можно попробовать запускать при каждом коммите в пайплайне CI/CD.

Используемые метрики

Метрика	Что измеряет	Зачем нужна
Precision@k	Точность поиска	Сколько из найденных фрагментов действительно релевантны. Пример: после запроса сервис вернул 3 фрагмента, а полезны из них только два. Если параметр низкий, значит, он выдает много мусора.

Recall@k	Полнота поиска	Сколько из всех нужных фрагментов были найдены системой. Сколько из всех нужных удалось найти. Если recall низкий, значит, сервис не находит важное и потом llm не сможет ответить правильно.
F1@k	Баланс между Precision и Recall	Усреднённая метрика: $2 \cdot \frac{P \cdot R}{P+R}$. Бывает, что precision параметр высокий, а recall низкий. F1 показывает общую полезность результата
MRR (опц.)	Позиция первого релевантного фрагмента	Показывает, насколько быстро наш сервис находит полезные нам фрагменты: чем выше, тем быстрее система возвращает полезный ответ.
Cosine Similarity (Answer Semantic Similarity)	Семантическое сходство (вектора)	Строит вектора для всех кусков кода и потом смотрим насклько вектора найденного похожи с нужным. Таким образом показывает оценку смысла

Также есть еще Ragas для помощи в оценке: "Библиотека Ragas значительно упрощает процесс оценки RAG-систем по вышеперечисленным метрикам. С её помощью разработчики могут создать набор данных, содержащий вопросы, ответы и соответствующие контексты, а затем оценить их по нескольким метрикам одновременно."

Аналоги

- GitHub Copilot - Передает llm код для работы, но использует лишь ближайший код в файле, не строит граф всего года и не может видеть весь проект целиком

- Sourcegraph Cody - AI помощник основанный на семантическом поиске. Строит индекс всего проекта и использует его для поиска. Cody will use semantic search to retrieve files from your codebase and use context from those files to answer your questions. "Uses advanced search and codebase context to help you write and fix code."

- Cursos - Похож на Copilot, но заточен под контекст всего проекта. Он индексирует проект и отвечает на вопросы о коде. Он не имеет граф или явных зависимостей, так как все хранить-ся в embeding

- Continue.dev - Open-source альтернатива Copilot с кастомными ретриверам. Все о поиске контекста здесь https://docs.continue.dev/autocomplete/context-selection, https://docs.continue.dev/autocomplete/how-it-works

- Refact.ai - работает очень близко как scaffold тоже помогает искать эффективно контекст. Также работает с граф кодом и делает семантический поиск. https://refact.ai/blog/2024/rag-in-refact-ai-technical-details/?utm_source=chatgpt.com - показаны технические детали работы

Идеи монетизации

- Подписка для команд — Scaffold можно будет использовать как сервис: подключаешь репу, он строит граф кода, и ты можешь делать запросы через API. За доступ к этому — подписка.
- Интеграция с IDE — можно сделать расширение для VSCode или других редакторов. Базовые функции — бесплатно, а вот расширенный контекст, история, и поиск по зависимостям — за деньги.
- Продажа компаниям — некоторые команды могут захотеть установить Scaffold у себя локально, особенно если код приватный. Это можно будет продавать как on-premise решение.
- Инструмент для ревью кода — можно встроить Scaffold в CI/CD: когда разработчик делает pull request, система сама подтягивает нужный контекст и помогает ревьюеру.
- Маркетплейс — если появятся разработчики, которые захотят писать свои плагины под Scaffold (например, для анализа ошибок или метрик), можно сделать магазин плагинов.
- API-доступ для агентов — если появятся LLM-ассистенты или внешние системы, которым нужно быстро искать контекст по коду, они смогут использовать наш API.
- Данные как продукт — можно будет продавать выгрузку графа проекта или делать кастомные отчёты/аналитику по коду для больших команд.

Источники

- RAG Evaluation Metrics Explained
  https://medium.com/%40med.el.harchaoui/rag-evaluation-metrics-explained-a-complete-guide-dbd7a3b
- How to Evaluate Your RAG System?
  https://www.vellum.ai/blog/how-to-evaluate-your-rag-system
- Оцениваем RAG-пайплайны
  https://habr.com/ru/articles/778166/
- Retrieval-Augmented Generation (RAG): Последние Исследования и Вызовы
  https://alimbekov.com/retrieval-augmented-generation-rag-Р»Р«СЃРњРҳРУР„РчРҳ-РчСЃСЃРњРҳРУ