AI TALENT HUB

Large Language Models for Code Documentation Generation

Работа студента Яблочникова Вячеслава,
НИУ ВШЭ, 3 курс

ИТМО NAPOLEON IT

Цель проекта и целевая аудитория

Большие языковые модели (LLMs) в последнее время используются не только в качестве chat-ботов или ассистентов, но и также для генерации кода и проверки фактов (fact-checking). Хорошо известно, что моделям легче (они это делают точнее) проверять или объяснять куски кода, чем генерировать их самостоятельно. И качество их работы значительно лучше, чем если бы это делал человек (https://arxiv.org/abs/2304.03938v1).

- **Проблема**: Отсутствие качественной и информативной документации к коду затрудняет разработку, сопровождение и совместную работу над проектами. Зачастую разработчики ленятся понятно писать и качественно объяснять созданный ими код.
- **Целевая аудитория**: Разработчики, команды по сопровождению ПО, студенты, начинающие программисты.

- **Цель проекта**: Создать инструмент, использующий LLM для автоматической генерации понятной и полезной документации к коду, а также базовых примеров его использования.

AI TALENT HUB

2024

01.
Research and solutions

Поиск и описание решения

Чтобы создать документацию, LLM должна уметь понимать код, объяснять код, а затем, форматировать его в легкую, понятную, разборчивую и удобную для использования форму. Важнее, что бы эта программа/приложение работала точно, причем не обязательно быстро. Для оценки качества можно использовать несколько основных метрик:

- **Accuracy**, **Relevance**: правильно ли описан код или нет, и насколько точно это соответствует действительности (not go off-topic).
- **Completeness**: насколько полно освещены важные аспекты кода. Настройка специальных чеклистов для файлов, функций и встроенной документации.
- **Understandability**, **Readability**: насколько легко читателю понимать документацию.
- **Time:** время, затраченное на генерацию

Нужно определиться на каком уровне абстракции требуется генерировать документацию: строчные комментарии (inline level), на уровне функций или классов, для отдельных файлов или каталогов.

Источники: https://arxiv.org/pdf/2312.10349.pdf, https://arxiv.org/pdf/2310.16673.pdf

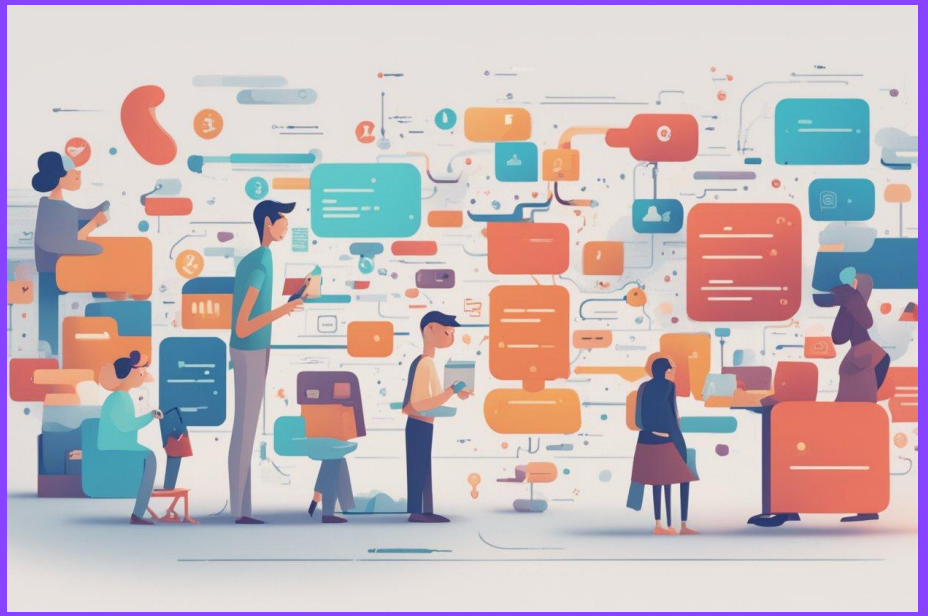
Архитектура и технологии

Для достижения этого нужно использовать техники ***promt-engineering***, результаты которых подаются в модель. В промте следует указать: **context, task, format.** Также можно использовать различные техники внутри одной модели: **zero-shot, few-shot и instruction fine-tuning strategies**. Всё описанное выше будет немного отличаться для разных уровней генерации абстракции.

Для выбора модели важны не только перечисленные ранее критерии, но соотношение цена-качество-размер, а также необходимость использовать open-source или закрытые модели тоже. Вот список некоторых из них: **GPT-3.5, GPT-4, Bard, LLama2, Starchat.** Первые три являются closed-source, а последние – open-source.

Данные удобнее всего брать из github (у bigtech компаний / open-source) с хорошим описанием кода (например, scikit-learn и PyTorch). Или добавлять базу знаний для дообучения или создания RAG систем. Можно использовать дополнительные открытые датасеты (пары: код, объяснение), такие как IRSE dataset, CoNaLa train dataset.
Обученную модель можно интегрировать (в качестве плагина) в современные IDE.

02.
Impact on users



Потенциальное влияние

Важность документации невозможно переоценить. Она особенна нужно в больших компаниях.

01
Этот проект может значительно улучшить процесс разработки, ускорив создание документации к коду.

02
Поможет новичкам быстрее разобраться в больших проектах и улучшит коммуникацию внутри команды.

03
Улучшение процесса разработки может привести к более быстрому выпуску программного обеспечения, что в конечном итоге может улучшить жизнь пользователей.

04
В долгосрочной перспективе такой формат может стать стандартом для автоматической генерации документации.

Вызовы и ограничения

01
Качество: Необходимо обеспечить высокое качество генерируемой документации.

02
Языковая специфика: Модели должны поддерживать разные языки программирования.

03
Конфиденциальность: Обработка закрытого кода может вызвать проблемы с конфиденциальностью.

04
Точность: Модели могут иногда генерировать неточные описания. Что необходимо верифицировать отдельными чекерами.

05
Сложность архитектуры: Необходимо балансировать между точностью, скоростью и ресурсами.

06
Обучение и обновление: Модели нужно регулярно обучать на новых данных. Технические изменения в языках программирования также могут повлиять на качество генерации.

Проект по автоматической генерации документации к коду может

улучшить процесс разработки,

облегчить коммуникацию внутри команды и

повысить качество программного обеспечения

Это позволит разработчикам **быстрее создавать и поддерживать код**, а также сделает программирование более доступным для всех. 🌟🚀

Контакты

Яблочников Вячеслав

Студент, НИУ ВШЭ

GitHub: https://github.com/slava-qw

TG: https://t.me/wertlonqwert

просто прикольная картинка

AI TALENT HUB ИТМО NAPOLEON IT