RAG and Embedders

Natural Language Processing

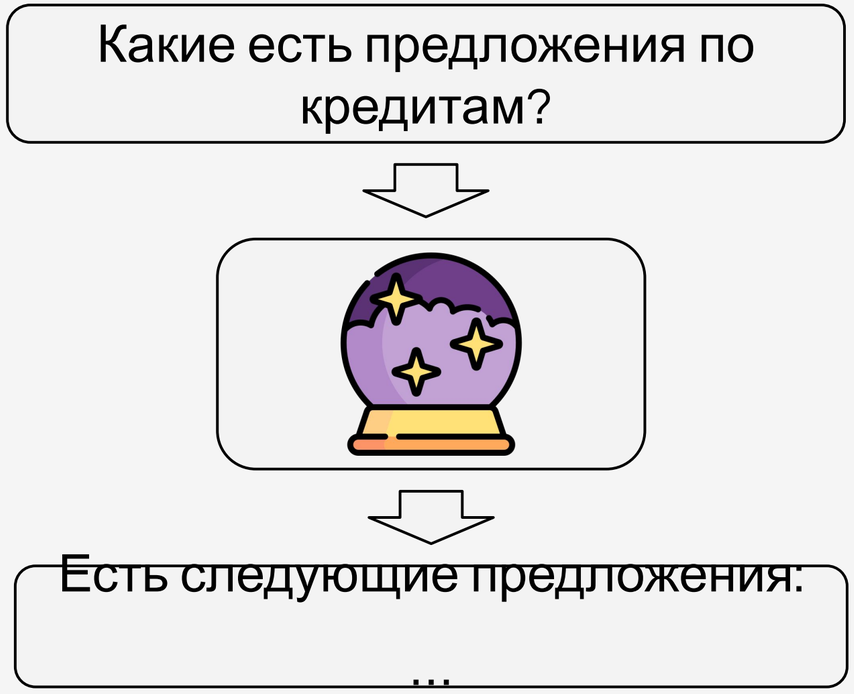
Ника Зыкова, 2025/12/04

Чего не хватает?

У всех подходов, которые мы обсуждали ранее, есть несколько ограничений:

- LLM ограничены контекстным окном: для слишком длинных данных модель скорее всего забудет начало;
- Большая часть данных не входит в обучение (документы компании, база знаний, PDF, почта и т. д.);
- LLM чаще может «галлюцинировать» при генерации без контекста.

Нужен механизм поиска и точной выдачи релевантной информации → эмбеддинги и RAG (Retrieval-Augmented Generation).

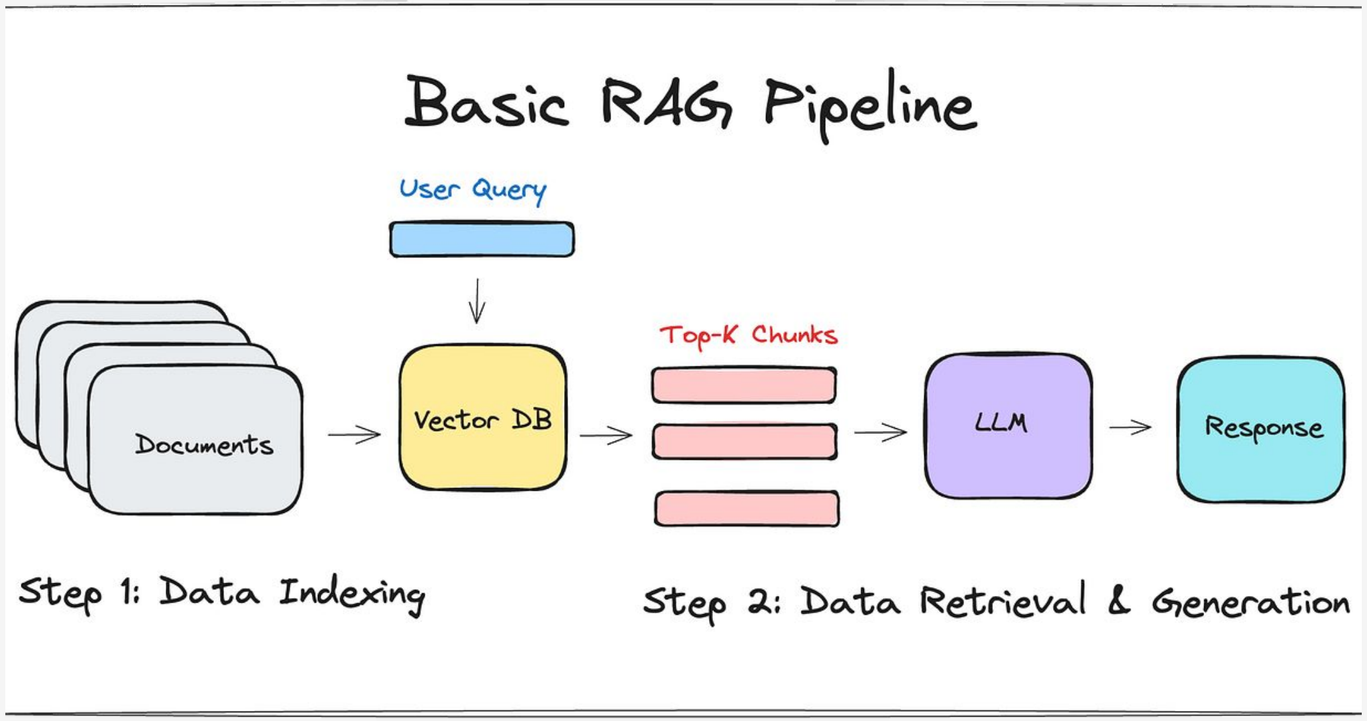


RAG

Идея: давайте будем подавать на вход генерации кусочки из базы знаний.

RAG состоит из нескольких частей:

- База знаний: набор текстов с важной информацией;
- Retriever: позволяет находить релевантные куски из базы;
- (?) Reranker - переоценивает важность кусков (опциональная часть);
- LLM: генерирует ответ на основе найденного.



https://github.com/enricollen/rag-conversational-agent

RAG: база знаний

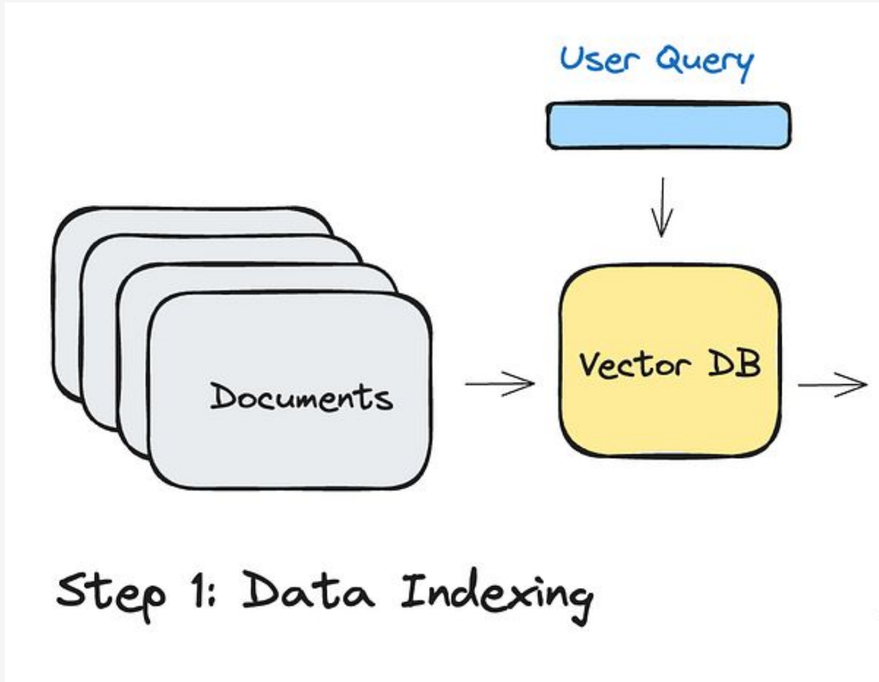
Дано: большой набор текстов для формирования базы знаний.

Как строится база:

- Каждый текст разделяем на чанки (чаще всего 300-500 токенов), иногда с перекрытиями в 20-30%;
- Чанки векторизуем с помощью эмбеддера и кладем в векторную БД.

Зачем чанки:

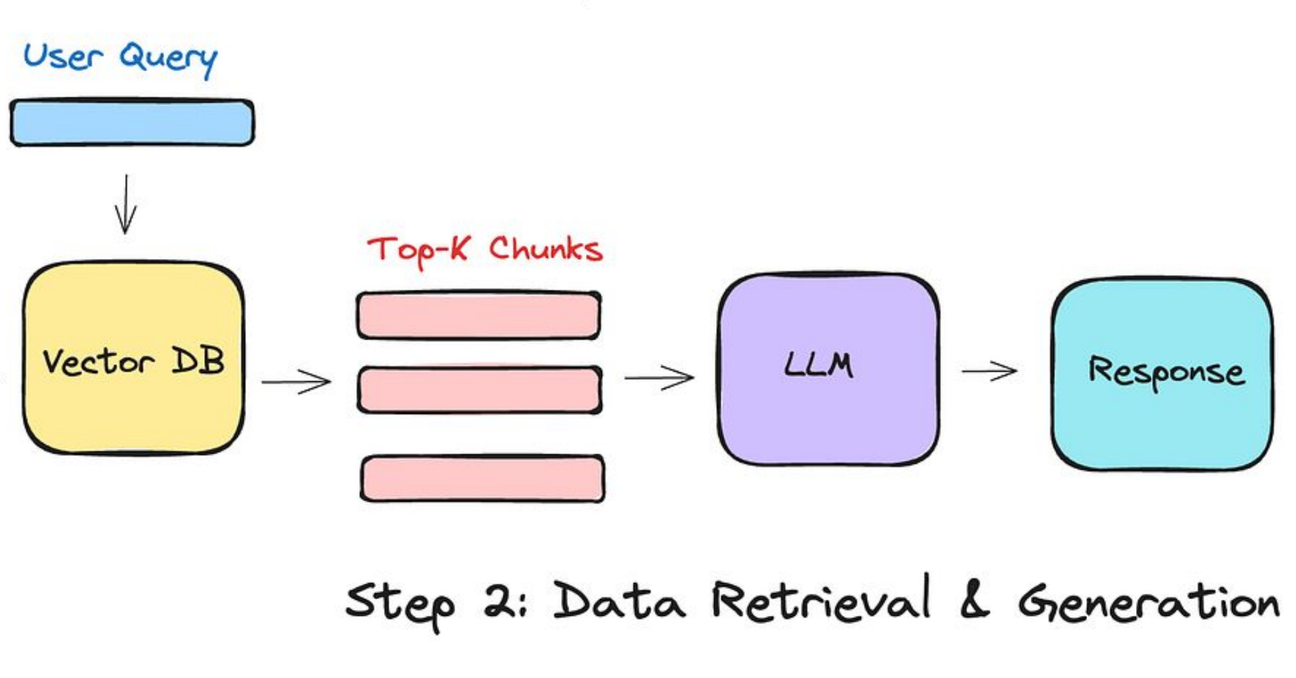
- Поиск будет более точным: выбираем важный кусочек текста, а не весь текст;
- Занимает меньше места в контексте модели.



https://github.com/enricollen/rag-conversational-agent

RAG: алгоритм работы

1. Пришедший запрос векторизуем тем же эмбеддером, которым векторизовали БД;
2. Ищем к нему top-k ближайших чанков из базы (чаще всего через косинусное или евклидово расстояние);
3. Опционально: используем отдельную модель чтобы переставить top-k более правильно (реранкер);
4. Подаем найденное на вход LLM вместе с промптом для генерации ответа.



https://github.com/enricollen/rag-conversational-agent

RAG: плюсы и минусы

Плюсы

- Актуальность данных: надо следить только за чанками, без переобучения модели;
- Контролируемость: можно точно знать, откуда взялся ответ;
- Меньше галлюцинаций;
- (часто) Не требует дообучения LLM;
- Гибкость: можно улучшать отдельные части (ретривер, реранкер, LLM);
- Масштабируемость: увеличиваем объем данных без увеличения моделей;
- Безопасность: внутренние документы компании не становятся частью параметров модели.

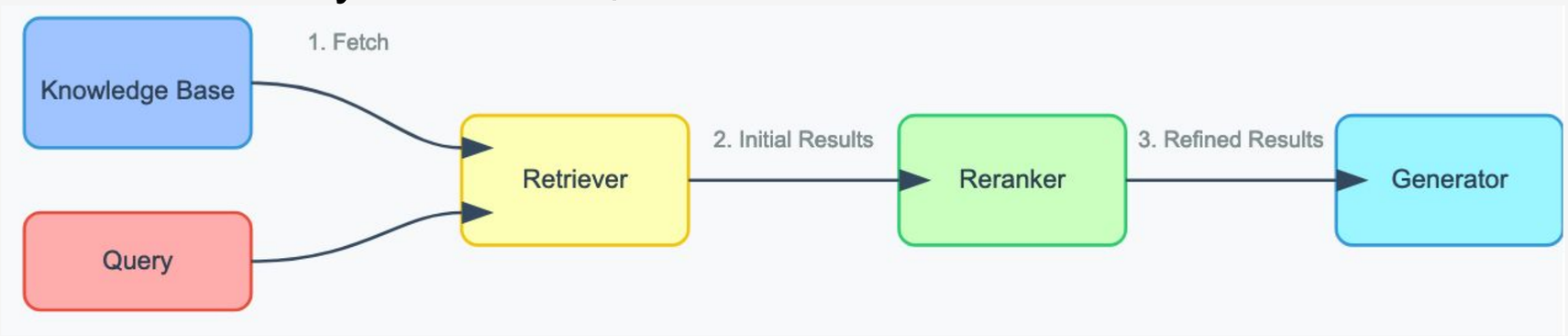
Минусы

- Сильно зависит от качества поиска: большая качества RAG = качество retrieval;
- Зависимость от chunking;
- Усложнённая архитектура;
- Нестабильность качества при обновлении данных;
- Плохие данные = плохой RAG;
- Может проигрывать fine-tuning там, где важно поведение модели (но тогда просто делается FT для RAG).

RAG: что улучшать

При разработке RAG’а базового качества, которое дает пара из предобученных эмбеддера и LLM, часто оказывается недостаточно. Что в таком случае можно улучшить:

- Дообучить эмбеддер на ваших данных: более качественный поиск релевантных чанков чаще всего сильно растит метрики;
- Дообучить генератор (LLM): актуально только, если LLM не соблюдает формат и/или совсем плохо работает с информацией;
- Добавить реранкер: отдельная модель, которая еще раз проверит уже найденные чанки (позволит выкинуть лишнее).



https://www.salesforce.com/blog/llamarank/

Эмбеддер

В качестве эмбеддера может быть любой способ векторизации: от Tf-Idf до современных Transformers-based эмбеддеров.

Но чаще всего используются: BERT (и его родственники), E5, MiniLM, GTE, Sentence-Transformers и многие другие.

Еще бывают разные retrieval стратегии:

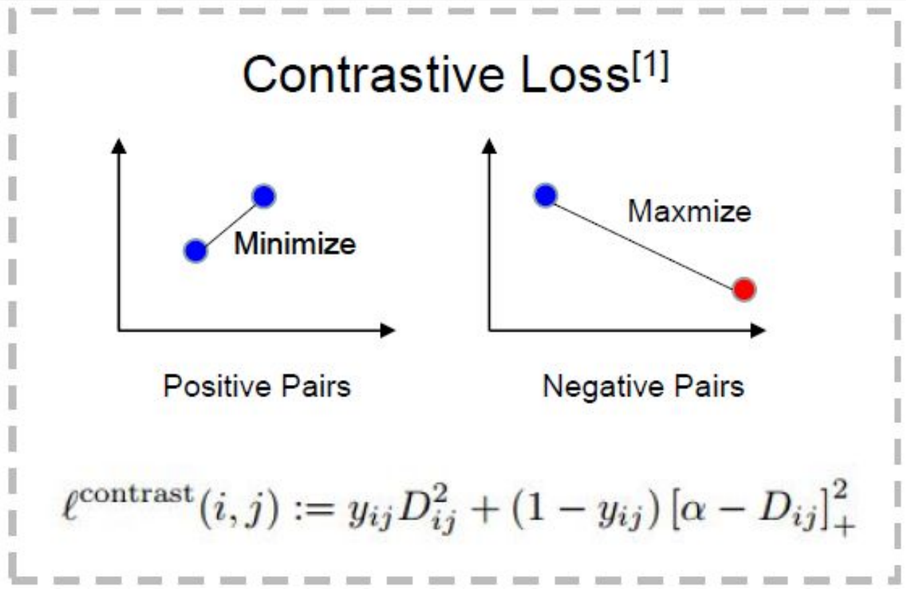
- Vector search: стандартный векторный поиск;
- Hybrid search (BM25 + embeddings): добавляем BM25 (улучшенный Tf-Idf) для более правильного вектора всего чанка;
- Reranking (Cross-Encoder): пересортировка после первичного поиска;
- Multi-step retrieval: много раз делаем retrieval, чтобы уточнять.

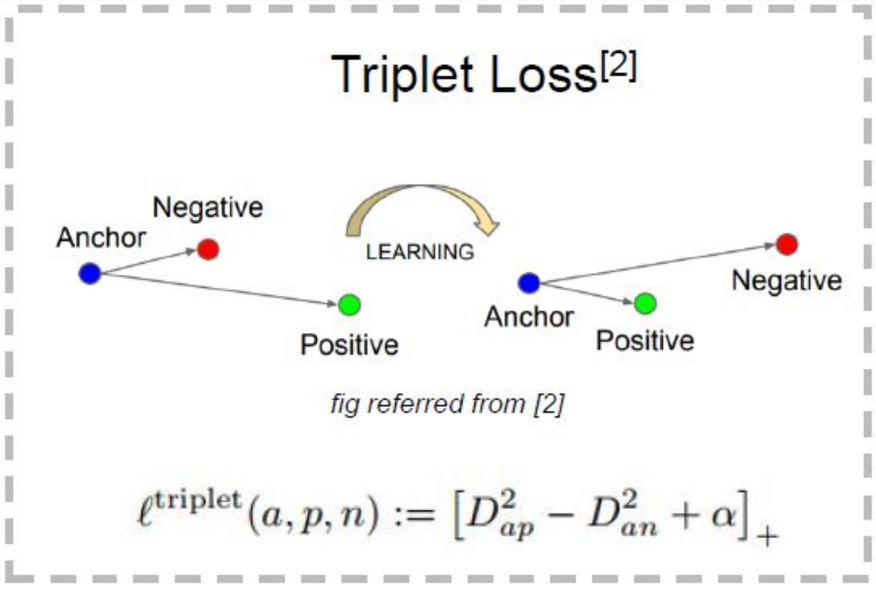
https://inixindojogja.co.id/5-tools-ai-llm-open-source-yang-mengubah-cara-manusia-berinteraksi-dengan-teknologi/

Эмбеддер: дообучение

Если retrieval дает много неверных чанков или часто не находит нужные, то имеет смысл дообучить эмбеддер. Что используется:

- Contrastive Learning: contrastive / triplet loss;
- Cosine similarity loss;
- Distillation Loss: предсказываем вероятности от модели-учителя.





https://programmersought.com/article/70461401080/

Оценка качества

RAG

- Доля правильных ответов без галлюцинаций;
- Дифференцированная оценка по типам ошибок: критичные / не критичные и т.д.;
- Устойчивость к нерелевантным данным: доля галлюцинаций в таком случае;
- Обоснованность ответов: правда ли генерация основана на чанках;
- Релевантность ответов;
- ROUGE / BLEU / METEOR: метрики для нечеткого сравнения текстов;
- Human Evaluation.

Embedders

- Метрики ранжирования: насколько часто мы находим в топ-1, топ-2, …, топ-k (не)релевантный вариант;
- Метрики близости: оцениваем как изменилась после дообучения средняя близость между запросом и релевантным чанком;
- Качество кластеризации: насколько хорошо выделяются кластера на новых эмбеддингах;
- Proxy задачи: любая down-stream задача (часто классификация).

https://programmersought.com/article/70461401080/