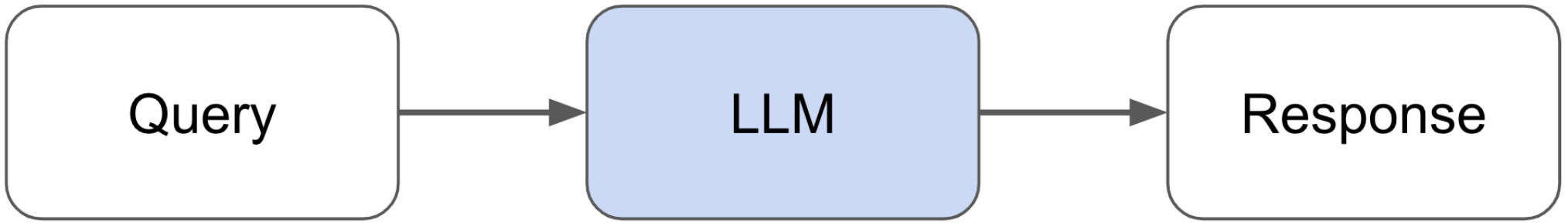
RAG

LLM



RAG

RAG (Retrieval Augmented Generation) — набор техник для подачи в модель дополнительных знаний, в основном через поиск релевантных текстов и их добавление в промпт. Например, модель, обученная на данных до 2023 года, не знает о фичах айфона 2025 года. Но если мы подадим ей на вход текст с описанием нового флагмана Apple — она применит знания оттуда и не выдумает своей ответ (не будет галлюцинировать). От качества обучающего корпуса модели зависит степень ее обращения к поданному тексту.

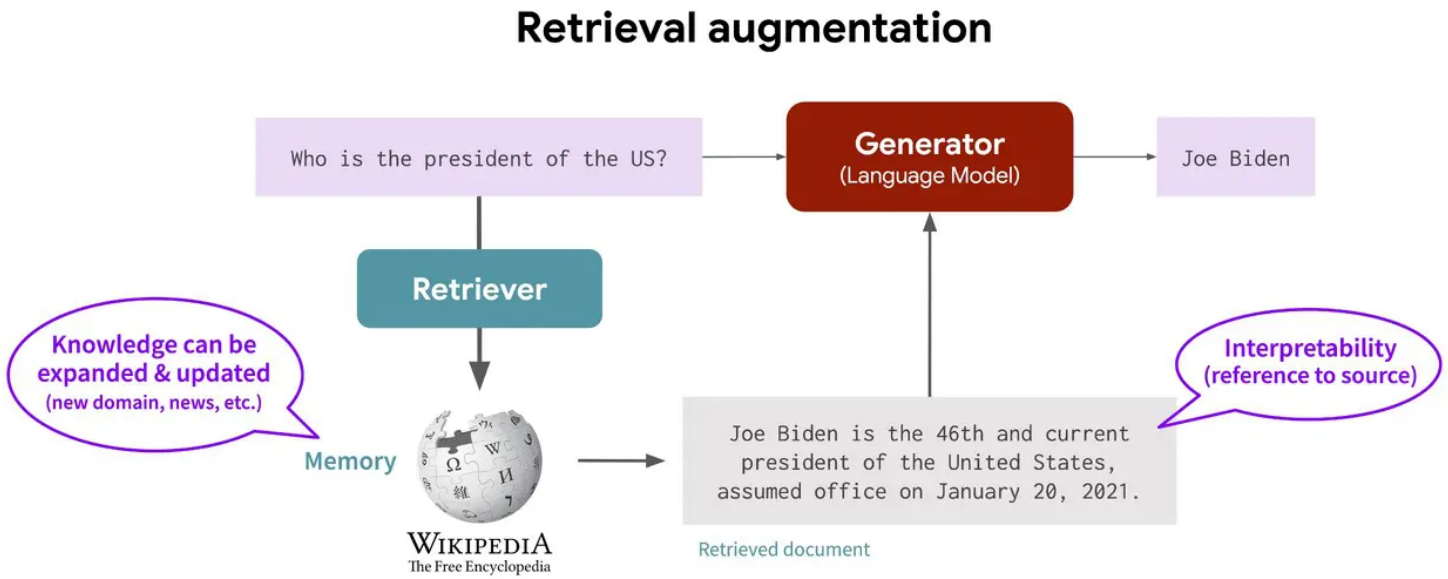
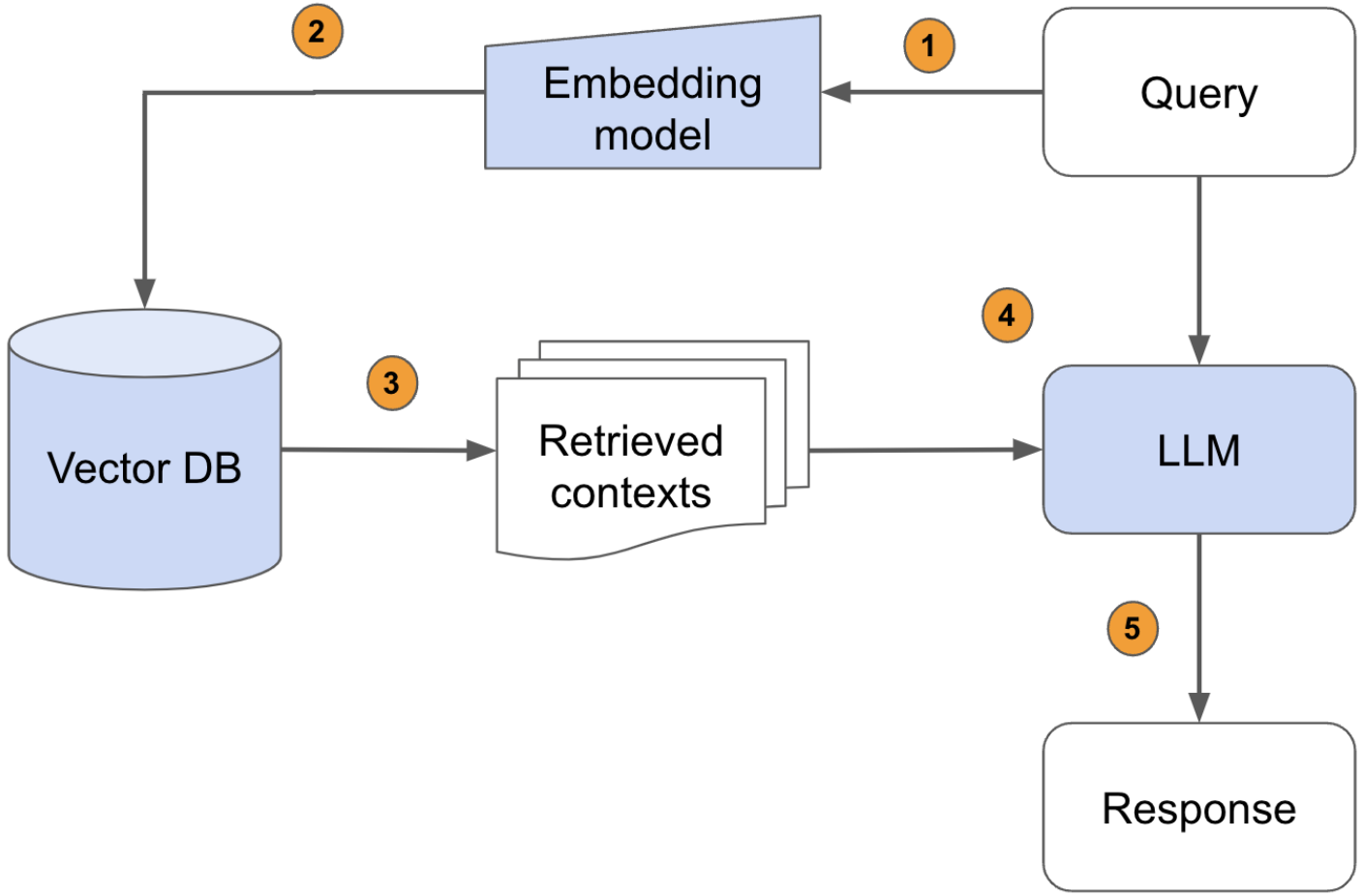
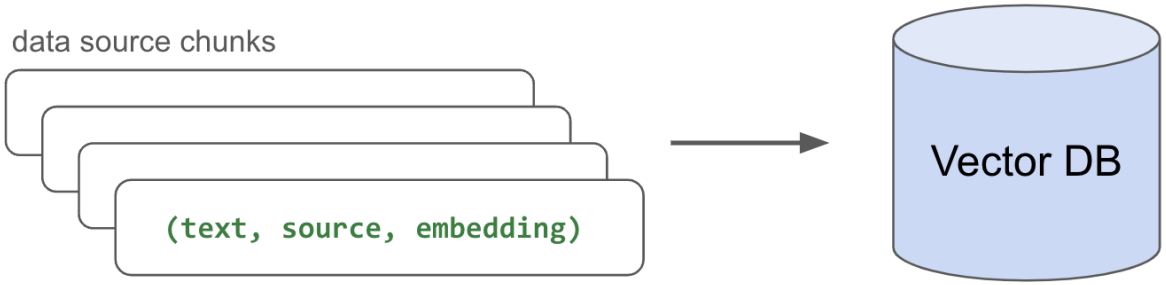


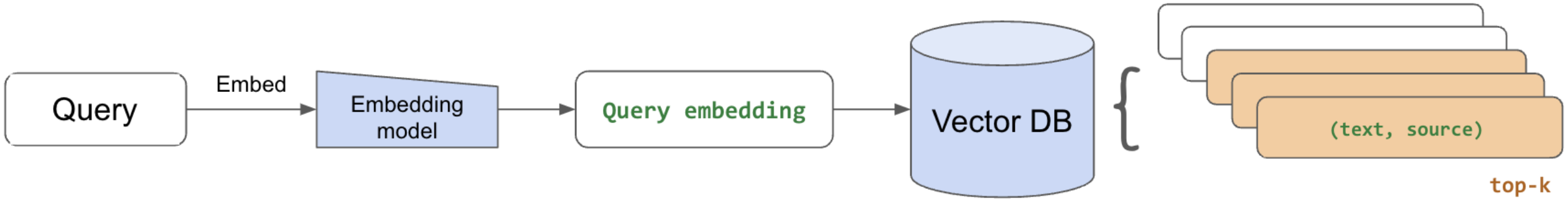
Схема RAG


2
Embedding model
1
Query
Vector DB
3
Retrieved contexts
4
LLM
5
Response

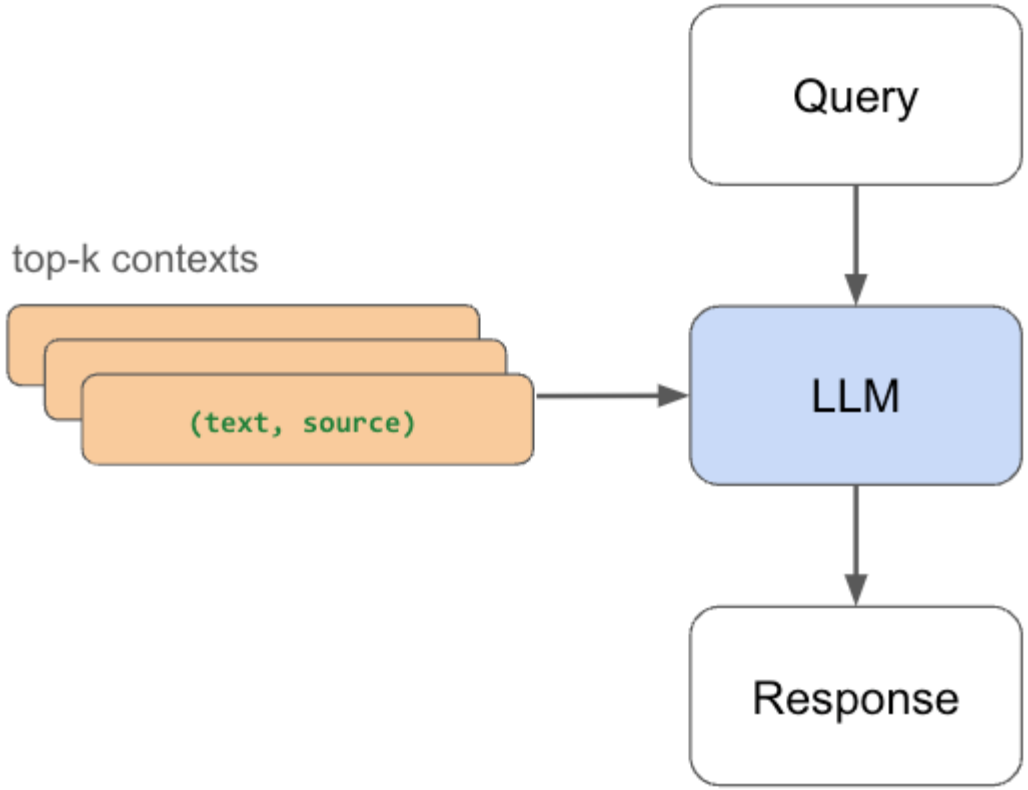
Создание векторной БД (Vector DB creation)



Запрос в векторную БД (Query Retrieval)

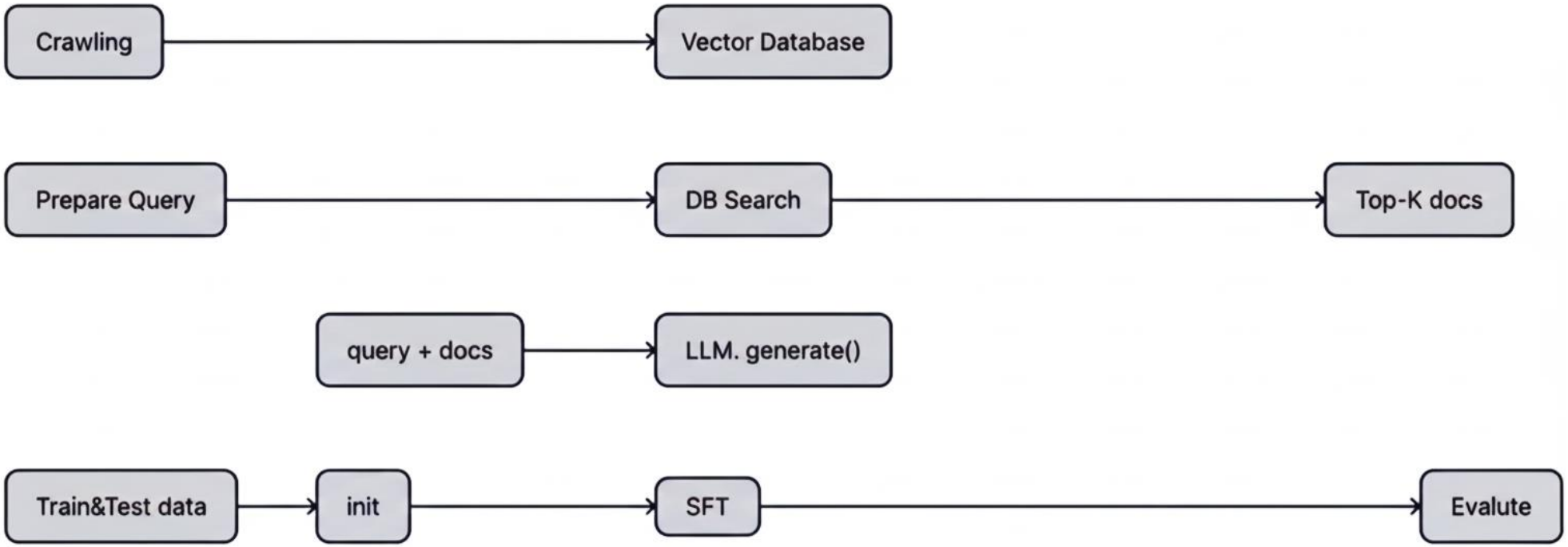


Генерация ответа (Response Generation)



Улучшение качества работы RAG

Классический RAG

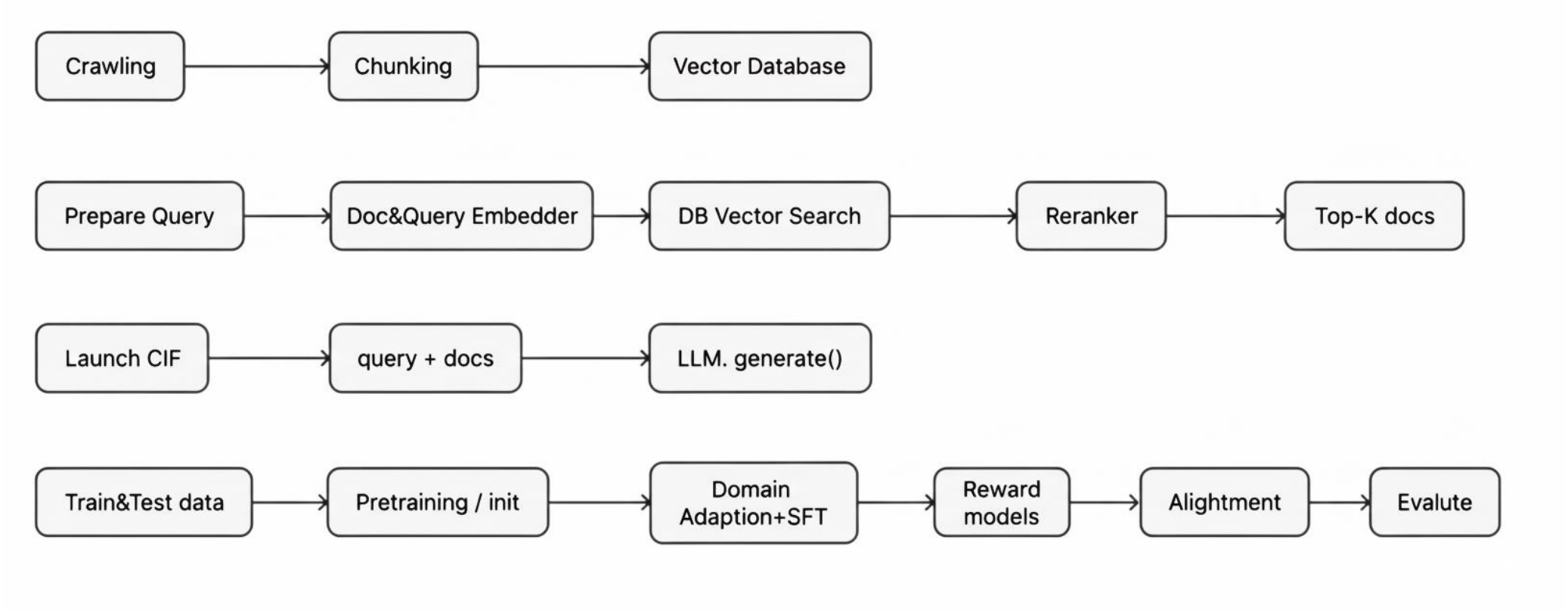


Типичная первая версия RAG

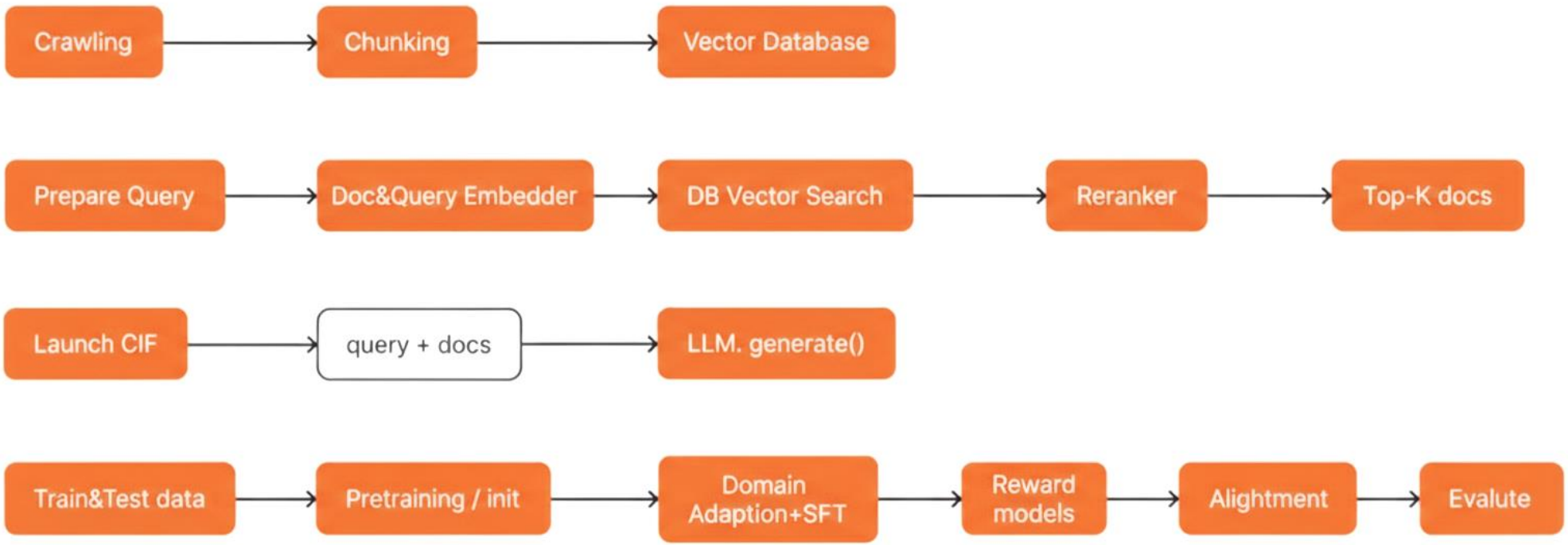
Типичная первая версия RAG

- Галлюцинации
- Неактуальные ответы
- Отсутствие прямого ответа на вопрос
- Зацикливания
- Неподходящий стиль ответа (длинные, сухие и т.д.)
- Неграмотные ответы
- Неэтичные ответы
- На вопрос не нужно было отвечать генеративно
- RAG-система не выдерживает нагрузку
- И много чего еще

Классический RAG чуть подробнее

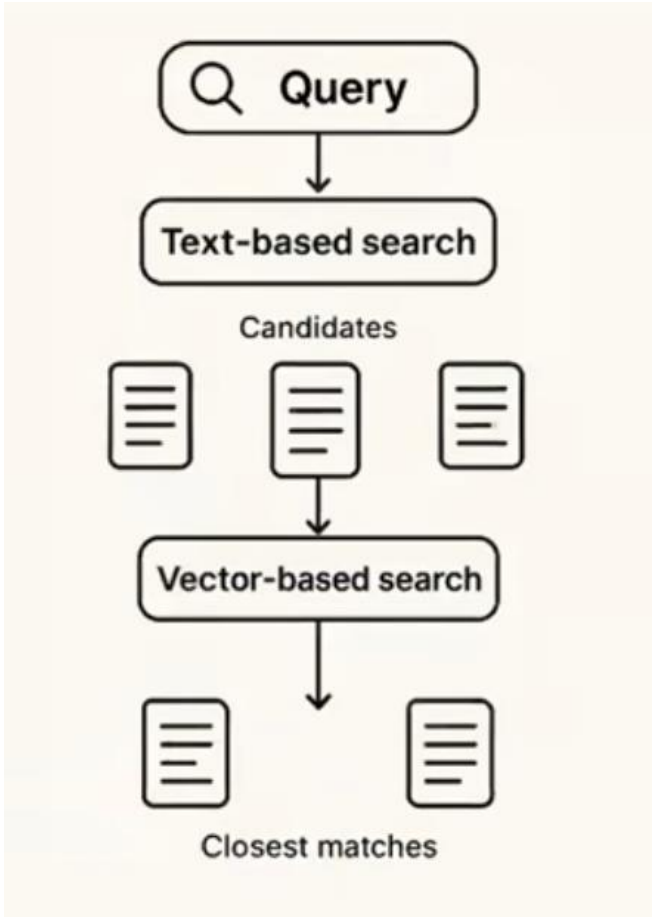


Что можно сделать не так



Поиск

- Нет релевантных документов нет генеративного ответа, даже с хорошим генератором
- Поиск:
  - Полнотекстовый: поиск по точному или частичному совпадению слов в тексте (TF-IDF, BM25)
  - Векторный: поиск по смыслу, основанный на векторных представлениях (эмбеддингах).
- На практике часто используют иерархический:
  - отбираем кандидатов с помощью полнотекстового поиска
  - находим наиболее близкие векторным



Поиск чего?

- Проблема: в базе данных много «плохих» документов, они попадают в источники RAG
- Часть удаляем как мусор и дубли: классификатор
- Часть очищаем:
  - Пишем свой парсер для проблемных источников: PDF, Word, специфичные для домена сайты
  - Удаляем служебную разметку
  - Scrapy, BeautifulSoup, API-интеграции

Поиск чего?

- Плохое чанкирование
  - Добавляем перекрытие
  - По кол-ву символов/токенов
  - По абзацам, заголовкам, маркерам
  - По семантике
  - Связь сущностей
- Сразу подумаем про БД
  - Хранить текст, вектор, мета-дату, ключевые слова
  - Инструменты: FAISS, Weaviate, Pinecone, Chroma и др.

Эмбеддер

Бейзлайн – предобученный BERT с hugginface

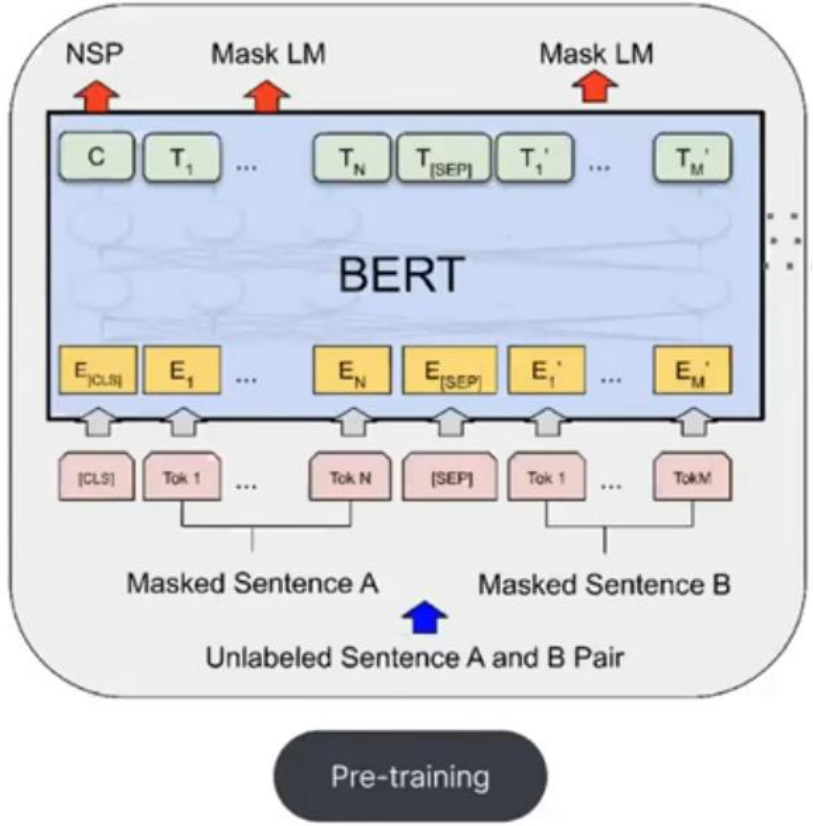
- Может плохо работать на вашем домене
- Может не уметь сопоставлять семантику запроса и документов (у них всегда разное распределение)

Эмбеддер

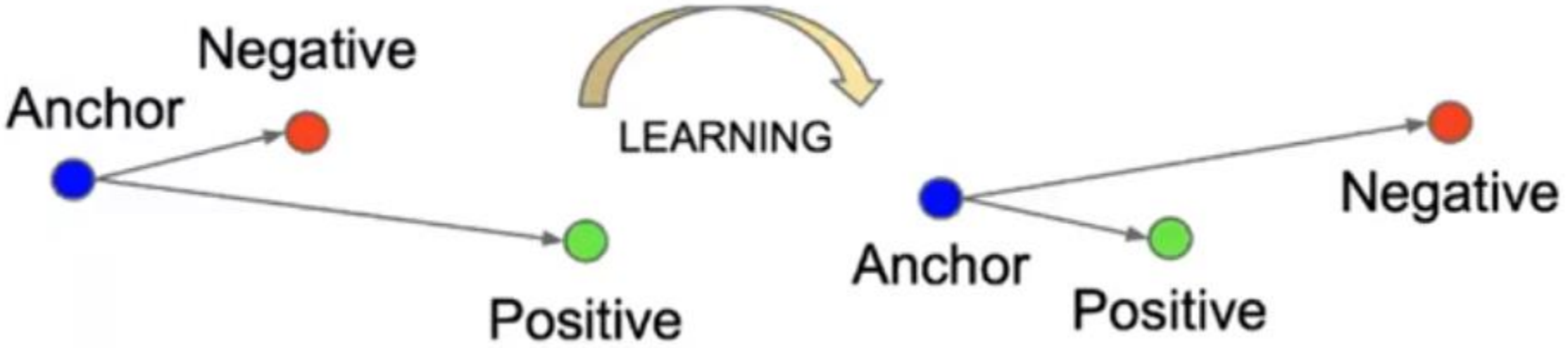
- Проблема:

Ближайшие по эмбеддингу тексты совершенно не близки

- Как улучшить:
  - Предобучить/дообучить BERT на вашем домене
  - Дообучить BERT на задачу сопоставления релевантных запросов и документов
  - Contrastive loss, triplet loss, двухбашенные архитектуры



Triplet & Contrastive



$$\mathcal{L}_{\text{triplet}}(\mathbf{x}, \mathbf{x}^+, \mathbf{x}^-) = \sum_{\mathbf{x} \in \mathcal{X}} \max\left(0, \|f(\mathbf{x}) - f(\mathbf{x}^+)\|_2^2 - \|f(\mathbf{x}) - f(\mathbf{x}^-)\|_2^2 + \epsilon\right)$$

$$\mathcal{L}_{\text{cont}}(\mathbf{x}_i, \mathbf{x}_j, \theta) = \mathbb{1}[y_i = y_j] \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2^2 + \mathbb{1}[y_i \neq y_j] \max(0, \epsilon - \|f_\theta(\mathbf{x}_i) - f_\theta(\mathbf{x}_j)\|_2)^2$$

Генератор

Бейзлайны:

промпт + API

- Может быть дороже, чем собственное решение
- Сложно подстраивать под собственные продуктовые требования

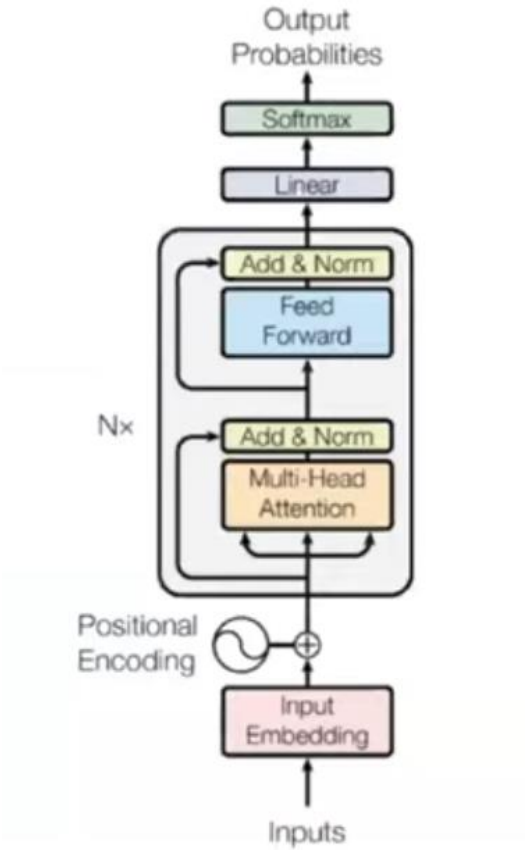
opensource-модель

- Из коробки скорее всего будет плохое качество
- Могут отсутствовать важные продуктовые свойства ответов

Бредовые генерации

Прокачать базовую модель

- Взять более «умный» инит: вместо llama-1 и gpt-2 лучше взять qwen2-2.5 или deepseek
- В частности, можно взять большую модель, дообучить, затем дистиллировать в меньшую
- При необходимости: адаптировать ее под домен и сделать качественный SFT
- Цель: добиться того, чтобы модель понимала, какую задачу решаем



Адаптация под домен

Как сделать эффективнее под язык:

- заменить токенизатор
- заменить эмбеддинги случайными
- заморозить веса, учить только эмбеддинги
- разморозить веса, учить всю сеть

SFT

- собрать корпус специфичных для домена и задачи текстов хорошего качества (синтетика с фильтрами или авторы) - например, юриспруденция
- аккуратно дообучить, контролируя деградацию на других доменах

```
Qwen2ForCausalLM(
  (model): Qwen2Model(
    (embed_tokens): Embedding(151936, 896)
    (layers): ModuleList(
      (0-23): 24 x Qwen2DecoderLayer(
        (self_attn): Qwen2Attention(
          (q_proj): Linear(in_features=896, out_features=896, bias=True)
          (k_proj): Linear(in_features=896, out_features=128, bias=True)
          (v_proj): Linear(in_features=896, out_features=128, bias=True)
          (o_proj): Linear(in_features=896, out_features=896, bias=False)
        )
        (mlp): Qwen2MLP(
          (gate_proj): Linear(in_features=896, out_features=4864, bias=False)
          (up_proj): Linear(in_features=896, out_features=4864, bias=False)
          (down_proj): Linear(in_features=4864, out_features=896, bias=False)
          (act_fn): SiLU()
        )
        (input_layernorm): Qwen2RMSNorm((896,), eps=1e-06)
        (post_attention_layernorm): Qwen2RMSNorm((896,), eps=1e-06)
      )
    )
    (norm): Qwen2RMSNorm((896,), eps=1e-06)
    (rotary_emb): Qwen2RotaryEmbedding()
  )
  (lm_head): Linear(in_features=896, out_features=151936, bias=False)
)
```

Нет нужных свойств

Задача ответа по источникам решается, но:

• Низкая достоверность:

галлюцинации, обобщения, нет ссылок на источники

• Ответы слишком длинные/короткие

• Всегда ссылаемся только на 1 источник

• Неудачная структура ответа

• Язык слишком "сухой"

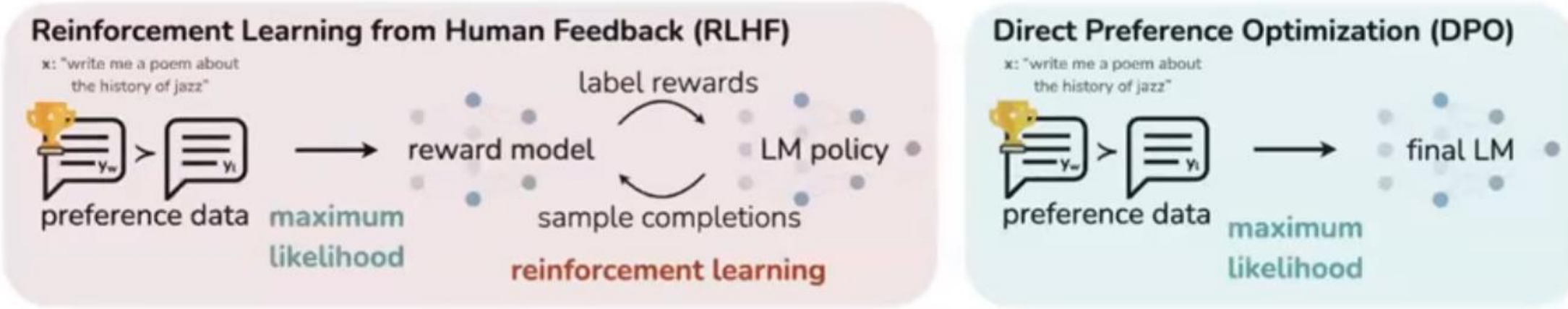
• Неэтичные ответы

Alignment

Как улучшить на этой стадии:

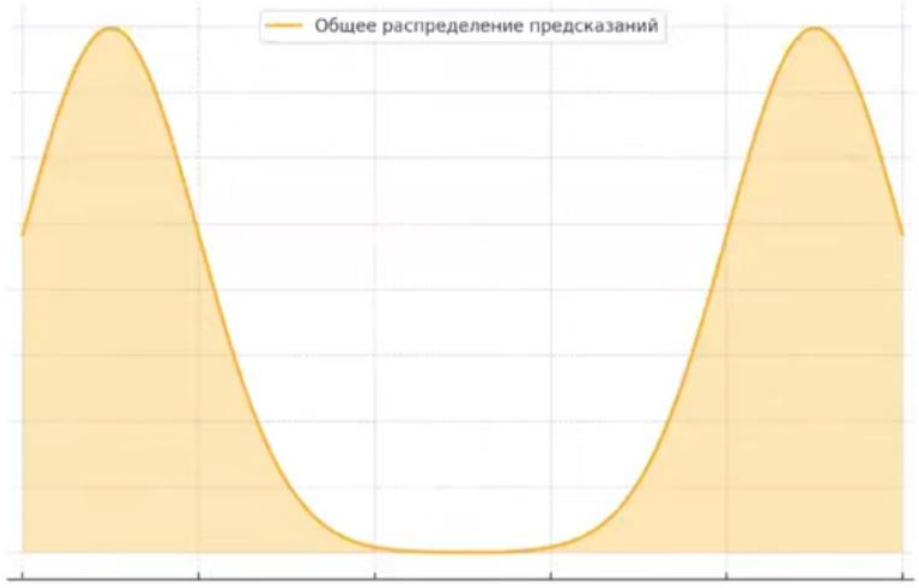
- Выбрать правильные методы: RS, DPO, GRPO, PPO, ...
- Правильно использовать: гипер-параметры, формулы, сходимость
- Правильные данные: кол-во vs кач-во (LIMA)
- Если используете реворд-модели — правильные реворд-модели, правильно их обучая и тестируя на правильных данных

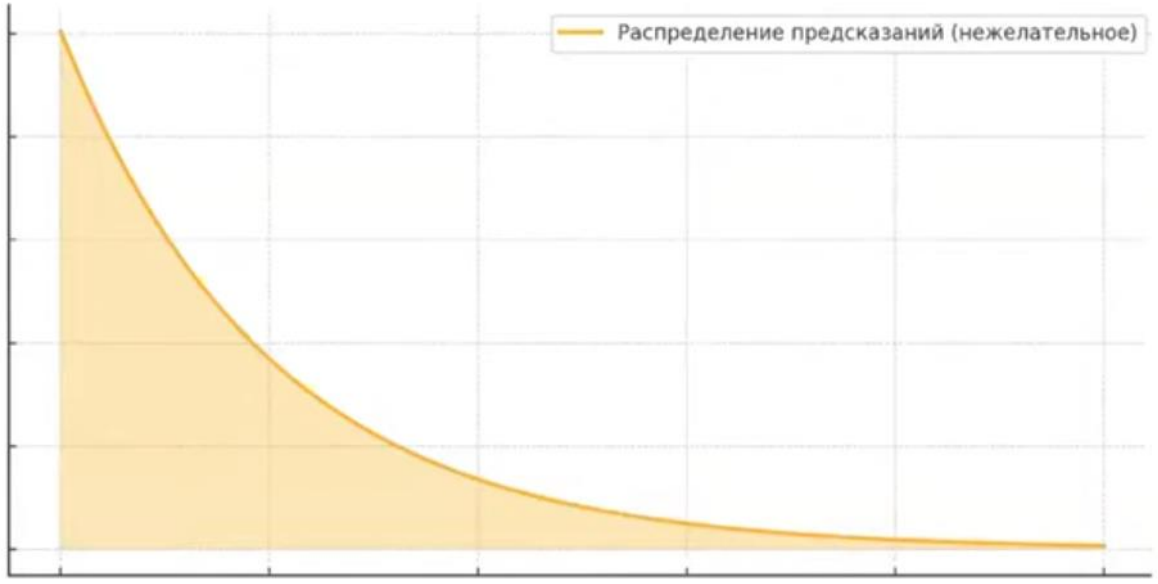
Alignment – DPO



$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l)\sim\mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w \mid x)}{\pi_{\text{ref}}(y_w \mid x)} - \beta \log \frac{\pi_\theta(y_l \mid x)}{\pi_{\text{ref}}(y_l \mid x)} \right) \right]$$

Alignment


Общее распределение предсказаний


Распределение предсказаний (нежелательное)

Данные

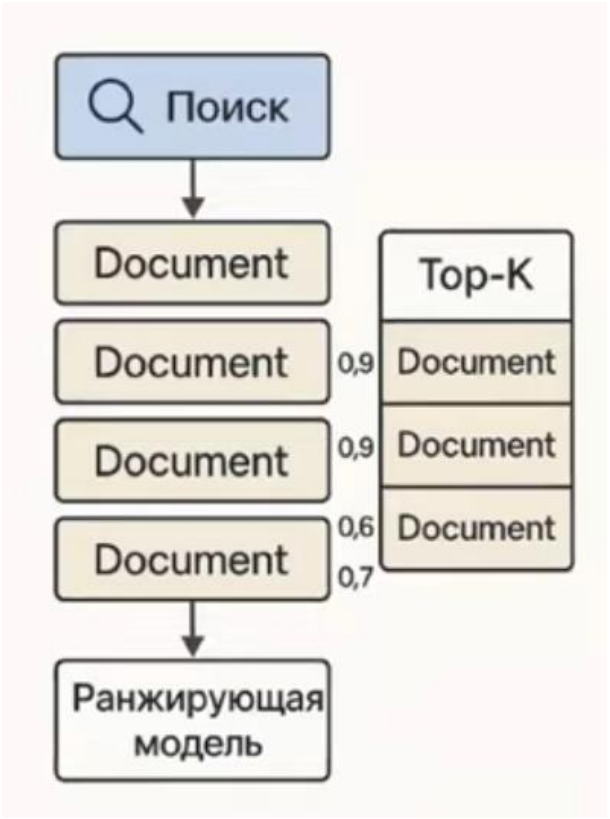
- Представим, что у нас нет размеченных данных для классификации, какой из двух ответов лучше
- Мы решили их разметить с помощью асессоров
- Какие могут возникнуть проблемы?

Данные

- Настройте процесс разметки: люди или LLM
- Инвестируйте в инструкцию
- Следите за качеством и согласованностью разметок
- Никуда без Active Learning!
- Не проливайте тест в трейн
- Стартуйте с малого и итеративно улучшайтесь
- Смотрите в данные: кластеризация и анализ

Реранкер

- Поиск выдает релевантные документы, но не в порядке их полезности для ответа
- Нужен самый полезный top-K
- Ранжирующая модель: можно моделировать качество ответа (например, реворды)



А что еще

- Не всегда на запрос нужно генерировать ответ: классификатор запуска, фильтры, заглушки
- Оптимизации: дистилляция, квантование, speculative decoding
- Ответы на частотники можно предподсчитать
- Собирайте фидбек от пользователей, используйте в алайнменте (через реворд или напрямую)

Использованные материалы

- Статья по построению RAG
- Открытая лекция DeepSchool по RAG