Retrieval-Augmented Generation (RAG)

Проблемы генерации LLM

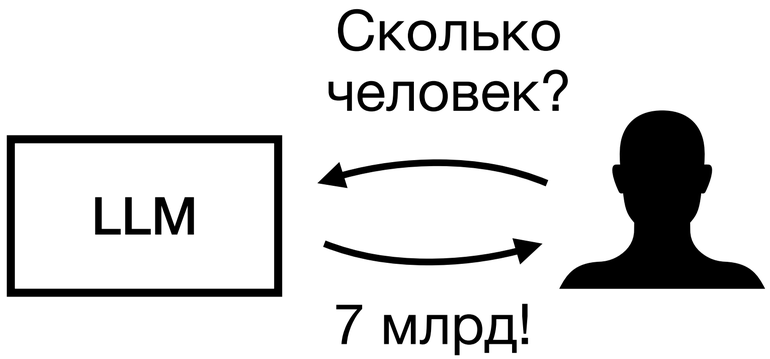
Хотим узнать ответ на вопрос

- Сколько человек на земле?

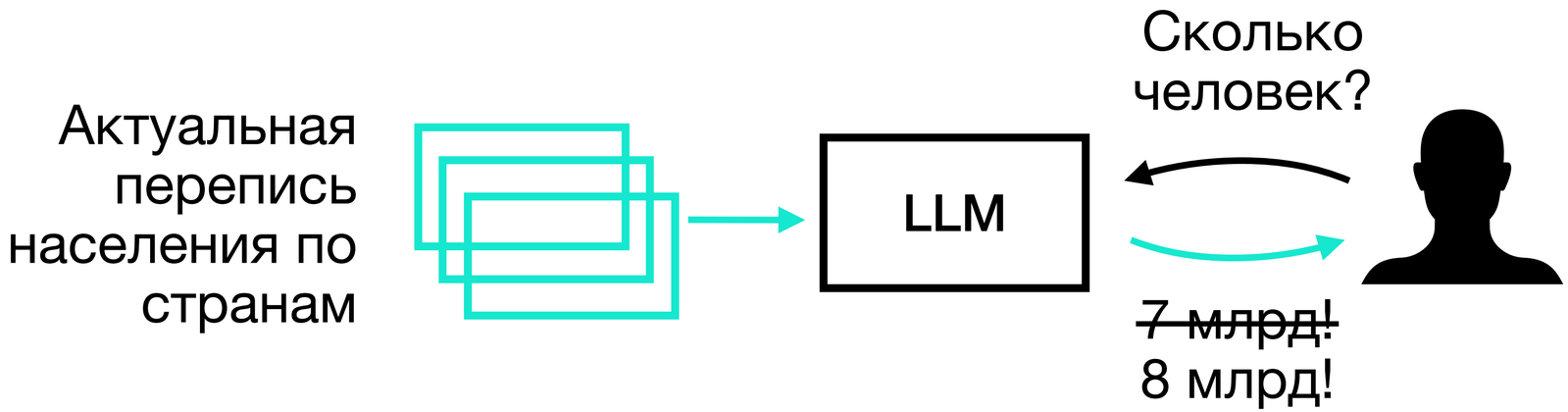
Что может пойти не так?

- Модель может галлюцинировать
- Информации не было в обучающем корпусе
- Информация устарела

Проблемы генерации LLM



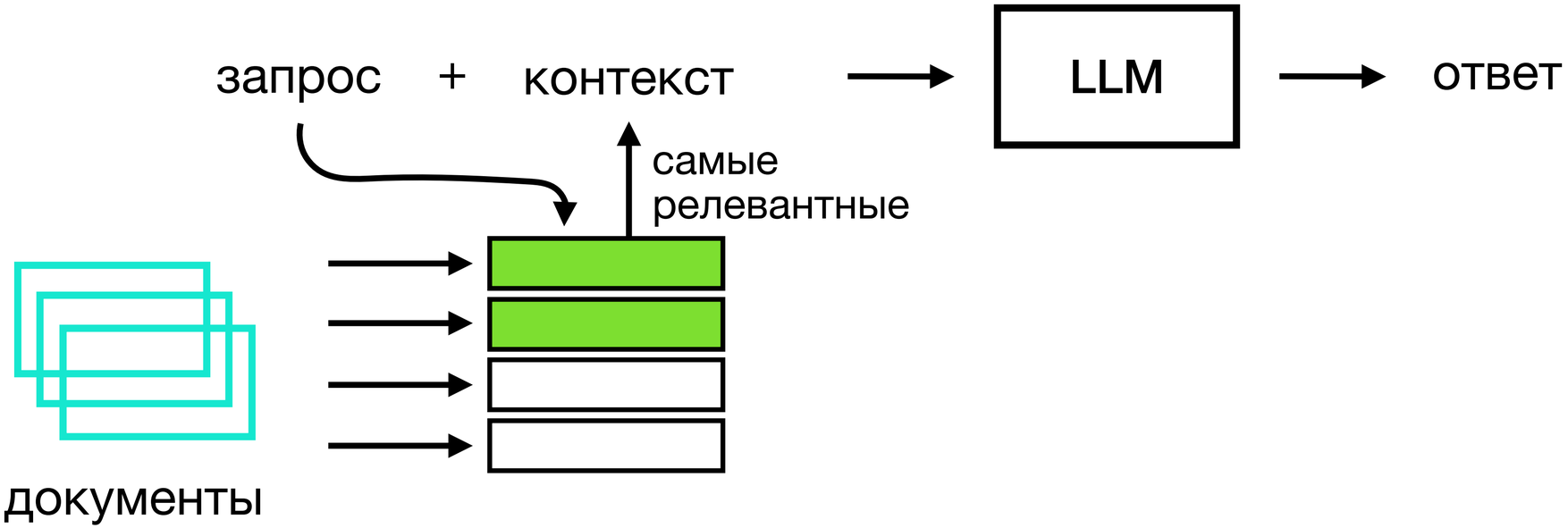
Проблемы генерации LLM



Retrieval-Augmented Generation (RAG)

Стандартный способ применения

Retrieval-Augmented Generation

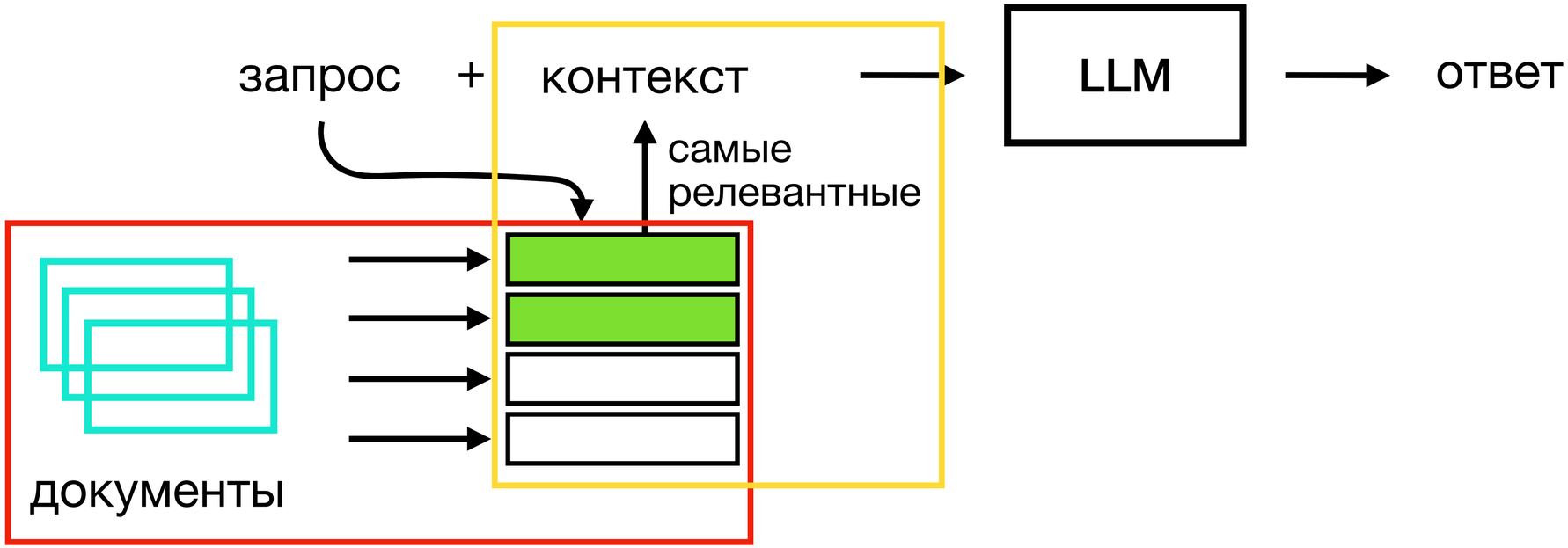


Когда использовать RAG

- **Поисковик по документации**
- **Юридическая помощь по определенной теме**
- **Применения в сфере образования**
  - Добавление учебных пособий в базу данных
- **Персонализированные рекомендации**
  - Сохранение всей релевантной информации о пользователе
- **Поддержка клиентов**
  - База данных хранит всю информацию об услугах компании

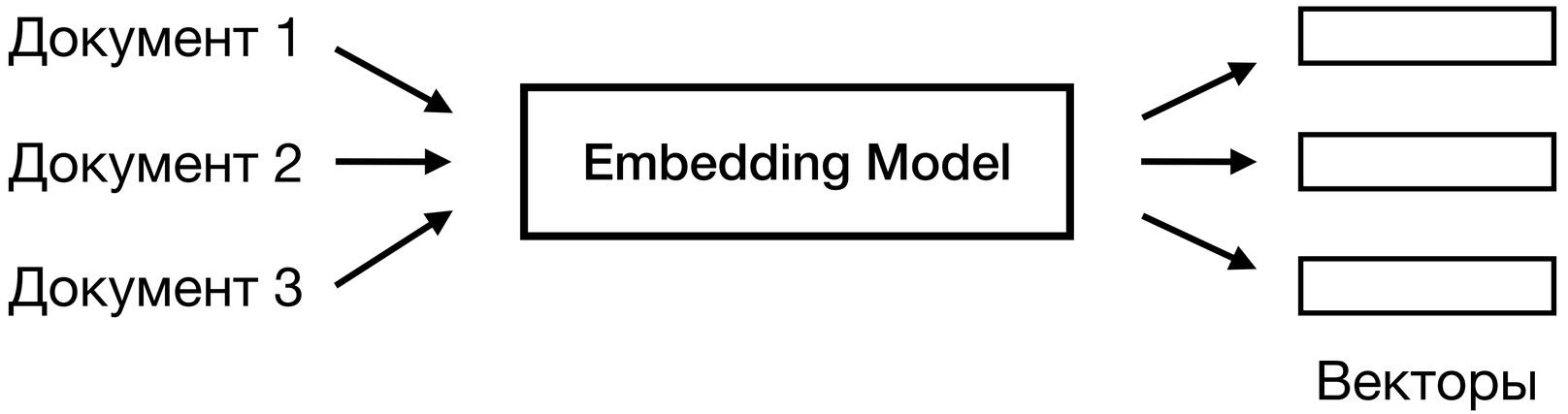
План

- Как хранить документы?
- Как выбирать наиболее релевантные запросу?



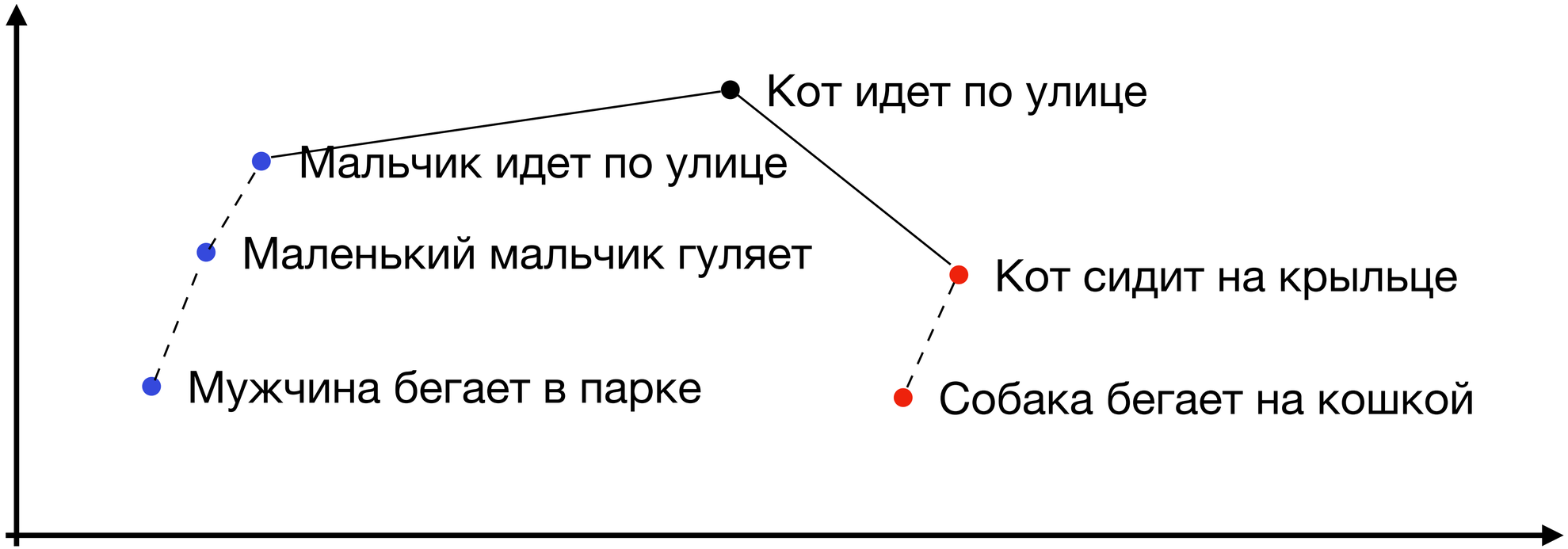
Векторные базы данных

Способ хранения набора текстов с возможностью семантического поиска



Векторные базы данных

Способ хранения набора текстов с возможностью семантического поиска



Кодирование документов

Кодировать документ целиком – плохая идея!

- Придется подавать весь документ в контекст, а размер контекста ограничен.
- Скорость и затраты по памяти квадратично зависят от размера контекста.

- Документ может содержать противоречивую информацию.
  - Населения земли за разные годы.

Лучше разбивать каждый документ на несколько кусков

Способы разбиения документов

- Разбиение с фиксированным размером
- Рекурсивное разбиение
- Семантическое разбиение

Разбиение с фиксированным размером

Делим документ на куски с фиксированным числом символов или токенов.
Добавляем наслоение кусков, чтобы важный текст не разбивался на разные куски.

Летним утром на опушке леса резвились три поросенка. Они прыгали по лужам и пели так беззаботно, будто лето никогда не закончится. Только вот по вечерам стало холодать, братья мерзли и самый младший предложил построить крепкий, теплый дом.

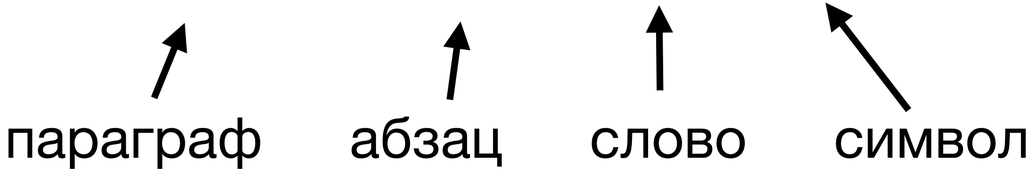
Мысль о том, что пора прекращать игры, так расстроила двух других поросят, что строительство было решено отложить на попозже. И только когда холод стал совсем нестерпим, началась наша история.

Старший поросенок так торопился побыстрее вернуться к играм, что построил не дом, а маленькую хижину из соломы.

Рекурсивное разбиение

Учитывает структуру текста

Вводим ограничение куска текста по длине и несколько уровней разбиения
Стандартный набор: `['\n\n', '\n', ' ', '']`



Если параграф не влезает в ограничение, делим его на абзацы
Если абзац не влезает – делим по словам
В конце делим по символам

Рекурсивное разбиение

Летним утром на опушке леса резвились три поросенка. Они прыгали по лужам и пели так беззаботно, будто лето никогда не закончится. Только вот по вечерам стало холодать, братья мерзли и самый младший предложил построить крепкий, теплый дом.

Мысль о том, что пора прекращать игры, так расстроила двух других поросят, что строительство было решено отложить на попозже. И только когда холод стал совсем нестерпим, началась наша история.

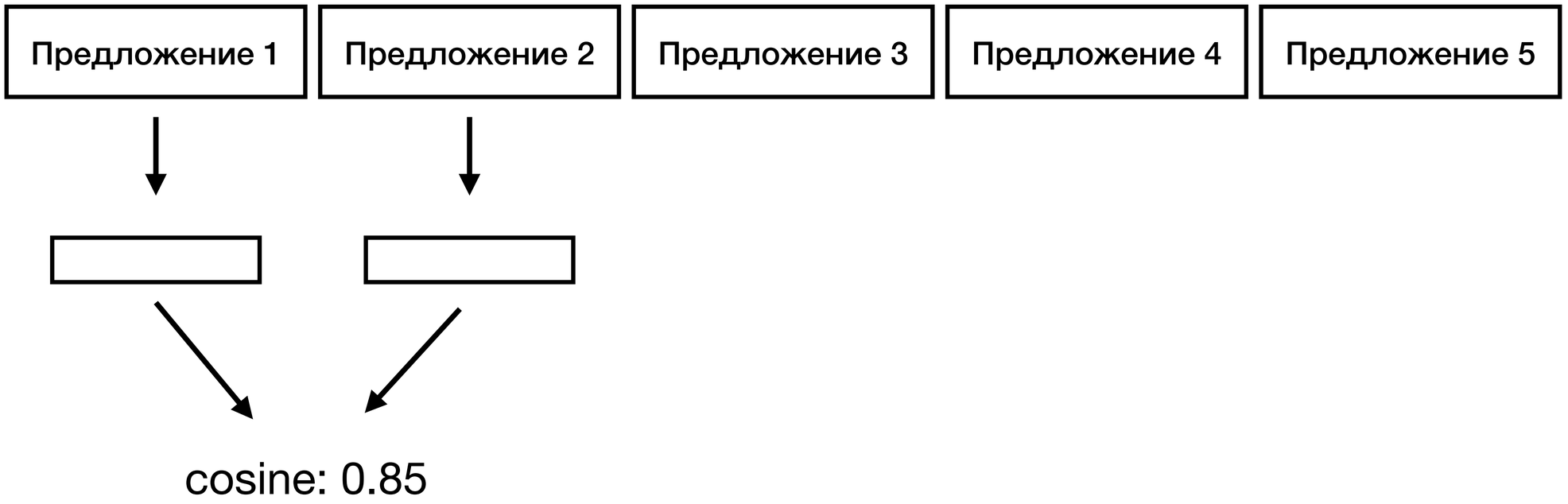
Старший поросенок так торопился побыстрее вернуться к играм, что построил не дом, а маленькую хижину из соломы.

Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.

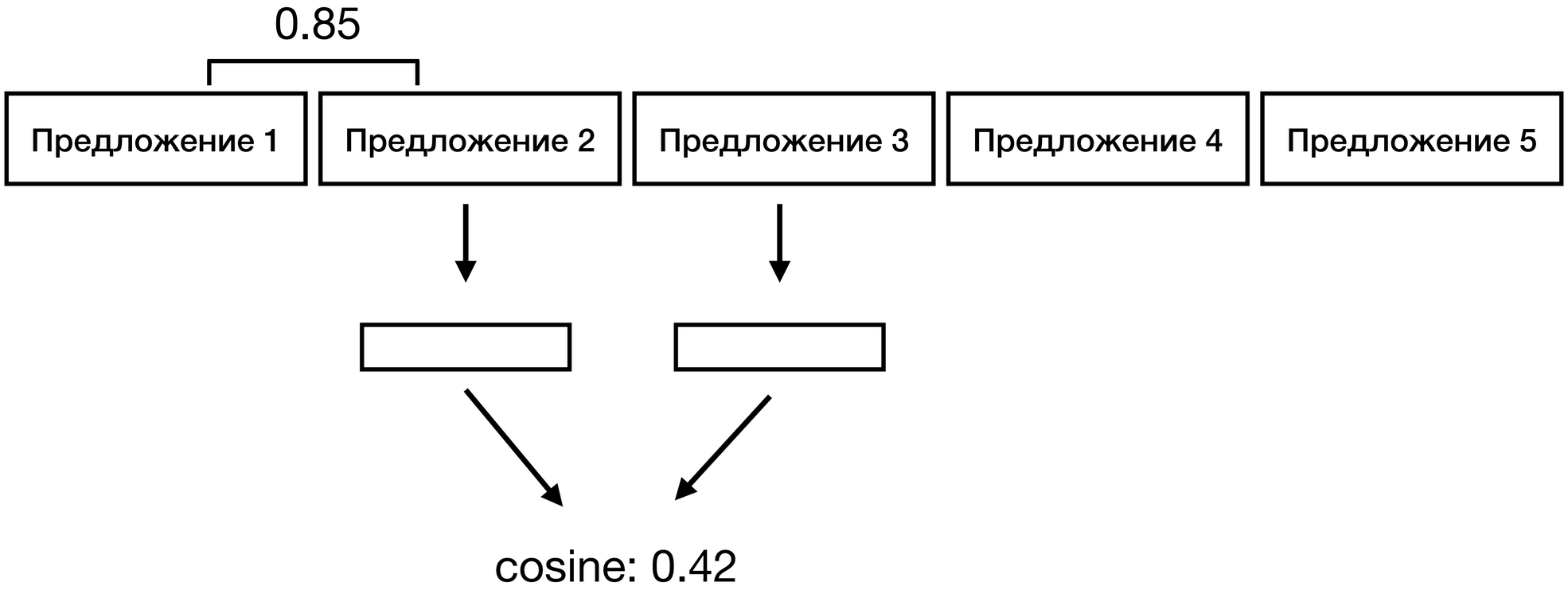
Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.



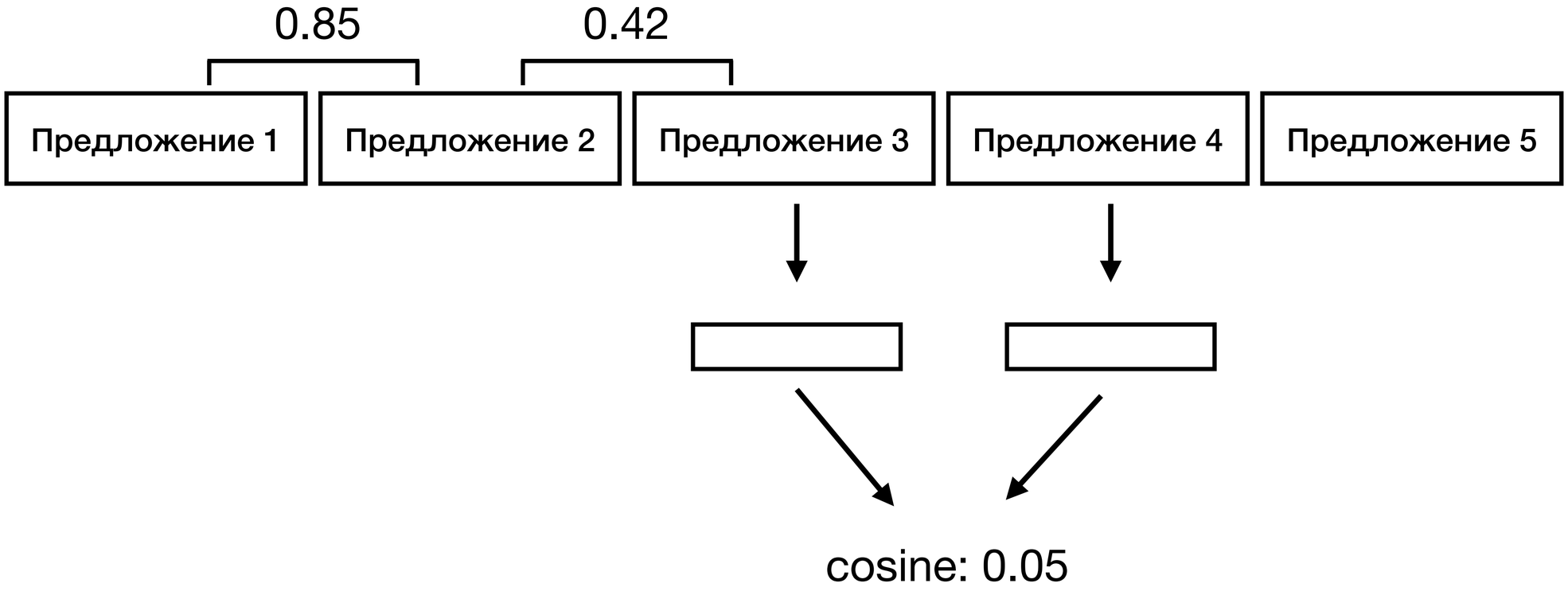
Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.



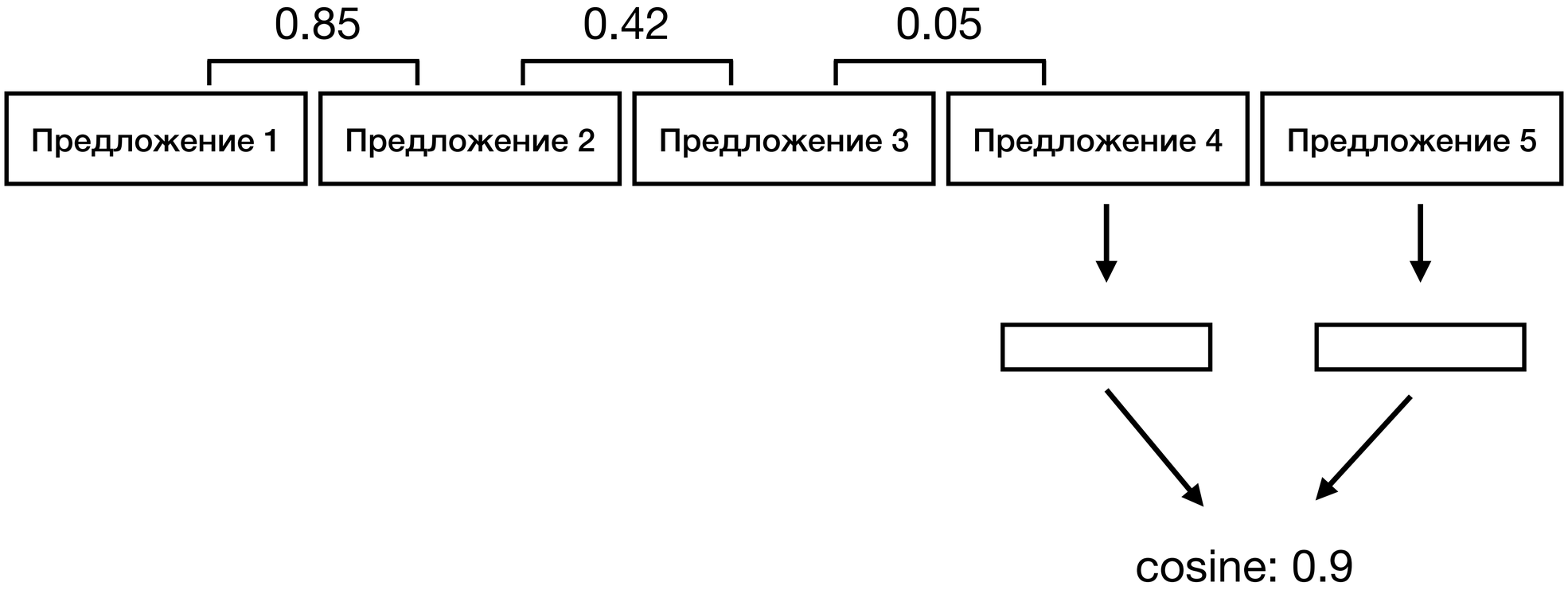
Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.



Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.



Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.

Семантическое разбиение

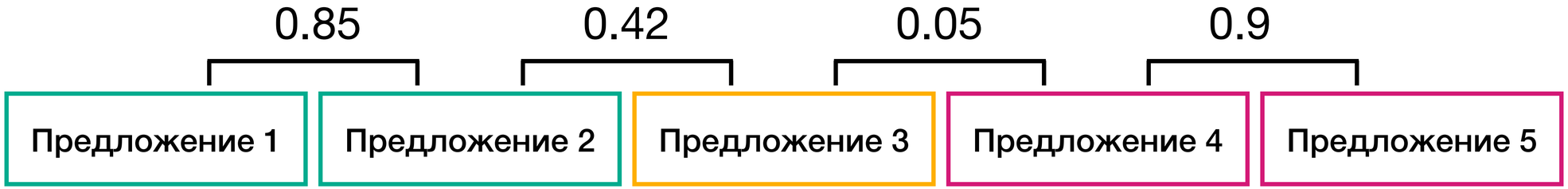
Находит разбиение, объединяя похожие предложения в один кусок.

Выбираем порог в зависимости от того, сколько предложений хотим объединить

Квантиль 50% – 0.85

Семантическое разбиение

Находит разбиение, объединяя похожие предложения в один кусок.



Выбираем порог в зависимости от того, сколько предложений хотим объединить

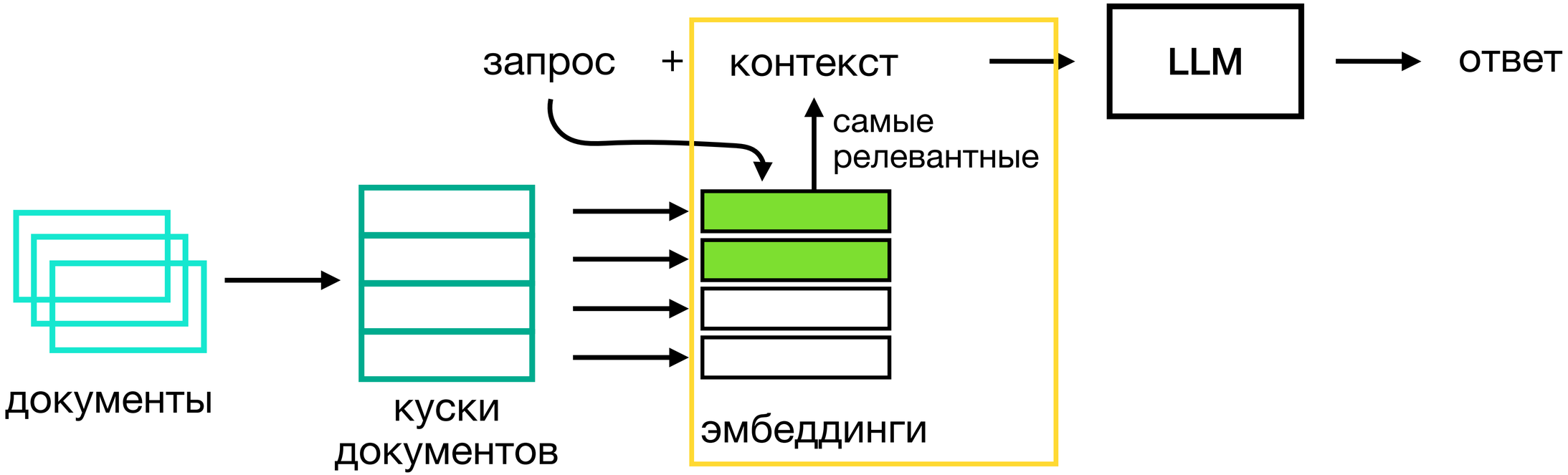
Квантиль 50% – 0.85

Объединяем предложения с похожестью не меньше 0.85

План

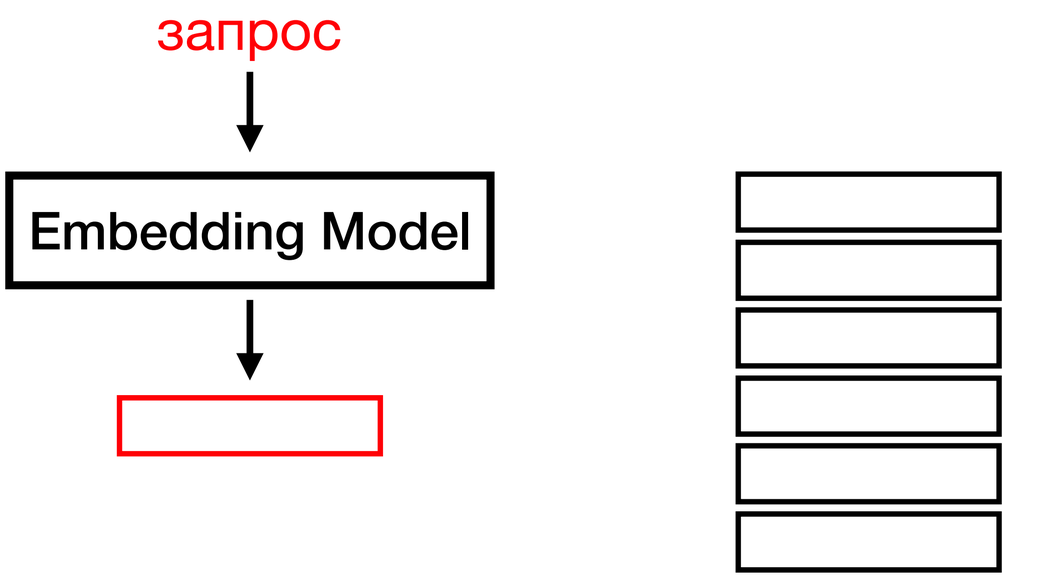
- Разобрались как хранить документы
- Как выбирать наиболее релевантные запросу?

  Retrieval-Augmented Generation



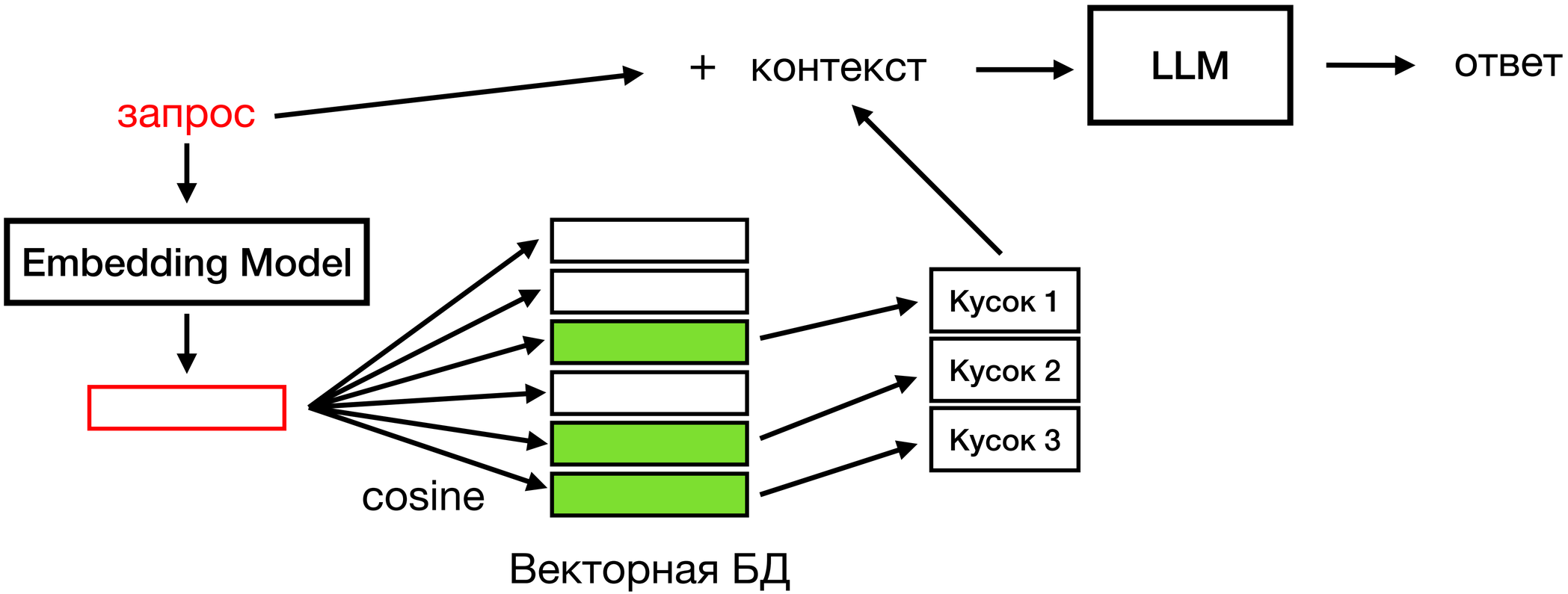
Как выбирать куски?
Retrieval

Наивный подход



Как выбирать куски?
Retrieval

Наивный подход



Проблемы наивного подхода

1. Необходимо считать похожесть каждого куска текста на запрос
   Может работать долго с большими БД

2. При сжатии текста в эмбеддинг информация теряется
   Из-за этого похожесть оценивается недостаточно хорошо

3. Извлекаются только куски документов
   Поэтому контекста может не хватить

4. Если формулировка пользователя неточна, то нужные документы не найдутся

Locality Sensitive Hashing

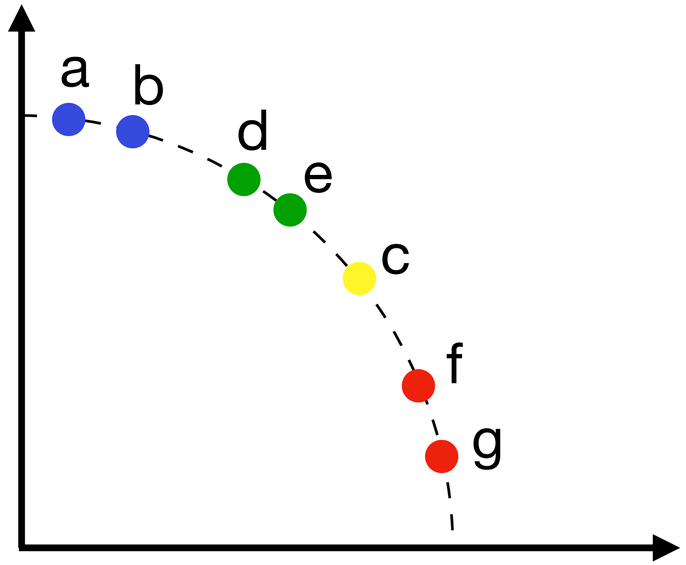
LSH – самый популярный метод ускорение поиска релевантных документов.

Идея построена на объединении похожих документов в одну группу

При поиске документов достаточно будет найти наиболее релевантную группу

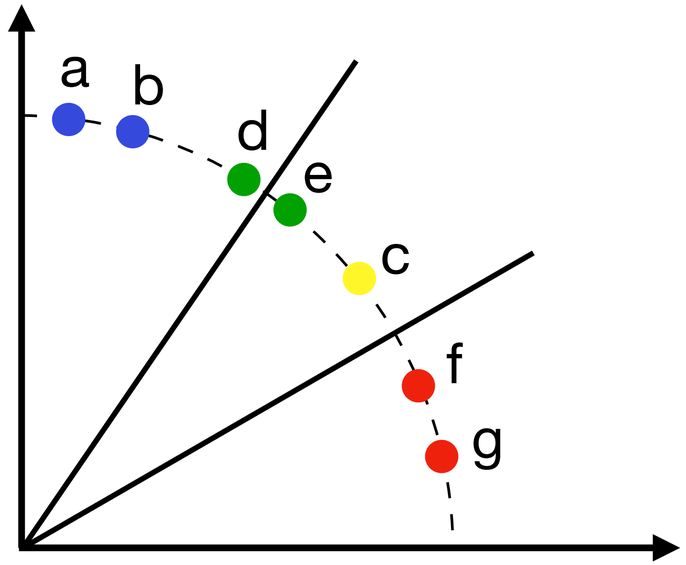
LSH: как работает

- Имеем набор отнормированных векторов из векторной БД
- Зададим хэш-функцию, сопоставляющую хэш каждому вектору



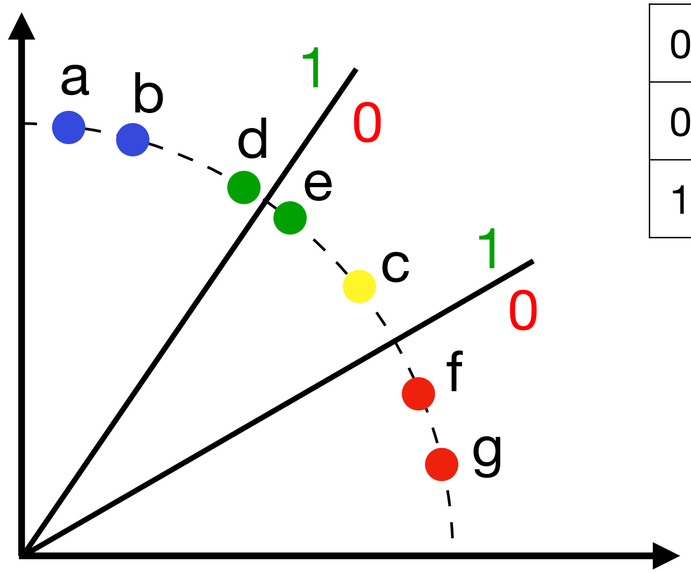
LSH: как работает

- Имеем набор отнормированных векторов из векторной БД
- Зададим хэш-функцию, сопоставляющую хэш каждому вектору
- Генерируем случайно n разделяющих гиперплоскостей



LSH: как работает

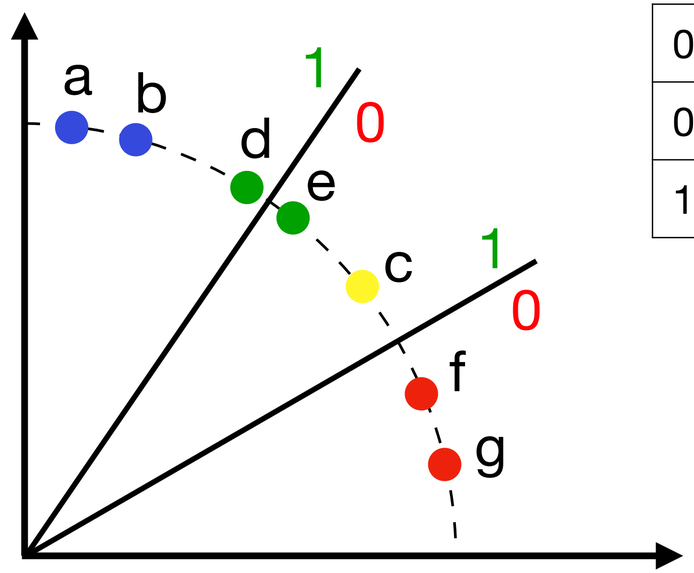
- Имеем набор отнормированных векторов из векторной БД
- Зададим хэш-функцию, сопоставляющую хэш каждому вектору
- Генерируем случайно n разделяющих гиперплоскостей
- Записываем в хэш 0, если точка лежит ниже плоскости и 1, если выше



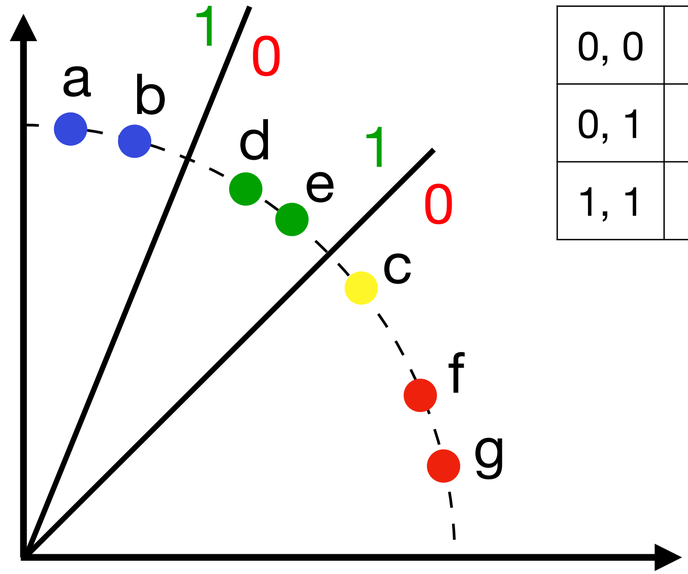
0, 0	f, g
0, 1	e, c
1, 1	a, b, d

LSH: как работает

- Имеем набор отнормированных векторов из векторной БД
- Зададим хэш-функцию, сопоставляющую хэш каждому вектору
- Генерируем случайно n разделяющих гиперплоскостей
- Записываем в хэш 0, если точка лежит ниже плоскости и 1, если выше
- Повторяем процедуру k раз
- При поиске извлекаем **все** векторы, которые оказались в группе с новым вектором



0, 0	f, g
0, 1	e, c
1, 1	a, b, d

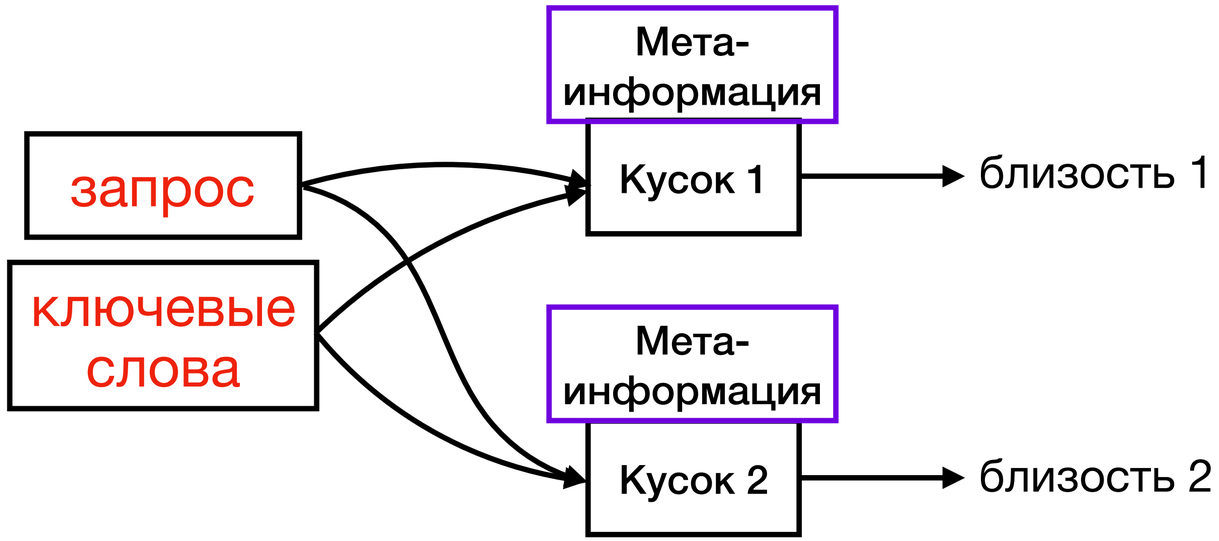


0, 0	c, f, g
0, 1	d, e
1, 1	a, b

Дополнительная информация

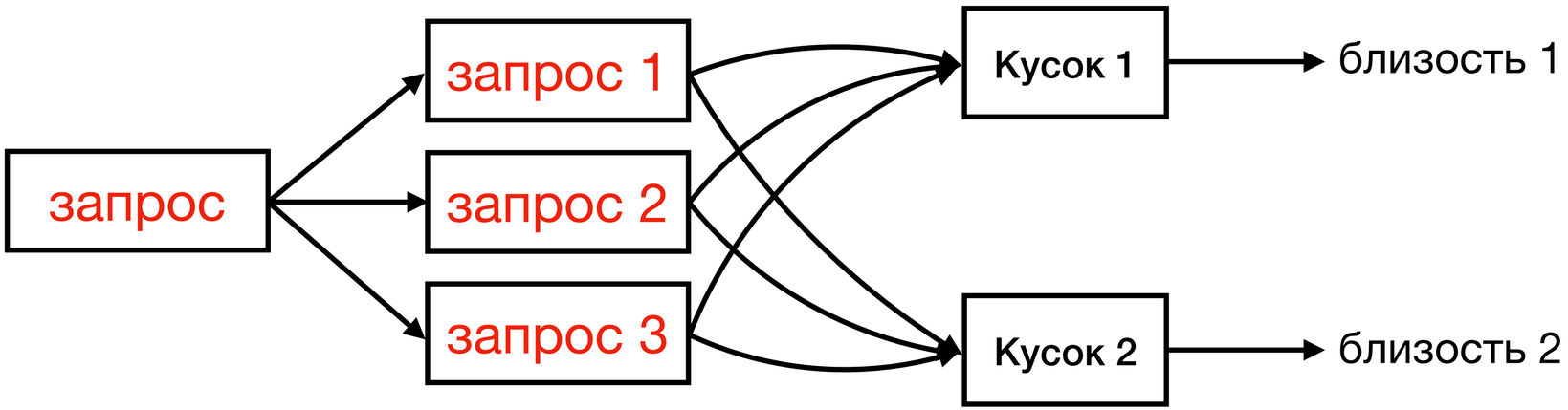
Для оценки близости можно использовать дополнительную информацию:

- Ключевые слова: NER, LLM
- Мета-информация у документов:
  - название предмета для учебного пособия
  - жанр фильма
  - цена товара



Исправление запроса

- Просим LLM переформулировать запрос несколько раз
- Оцениваем близость куска текста к каждому запросу

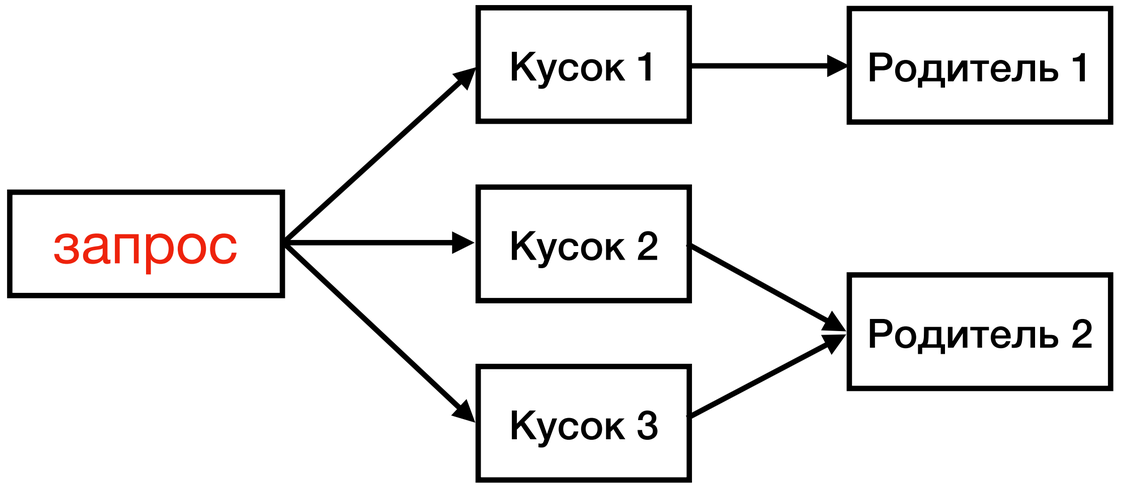


Родительские куски текста

- Разбиваем документ на большие куски (родители)
- Затем каждый кусок на куски поменьше (дети)

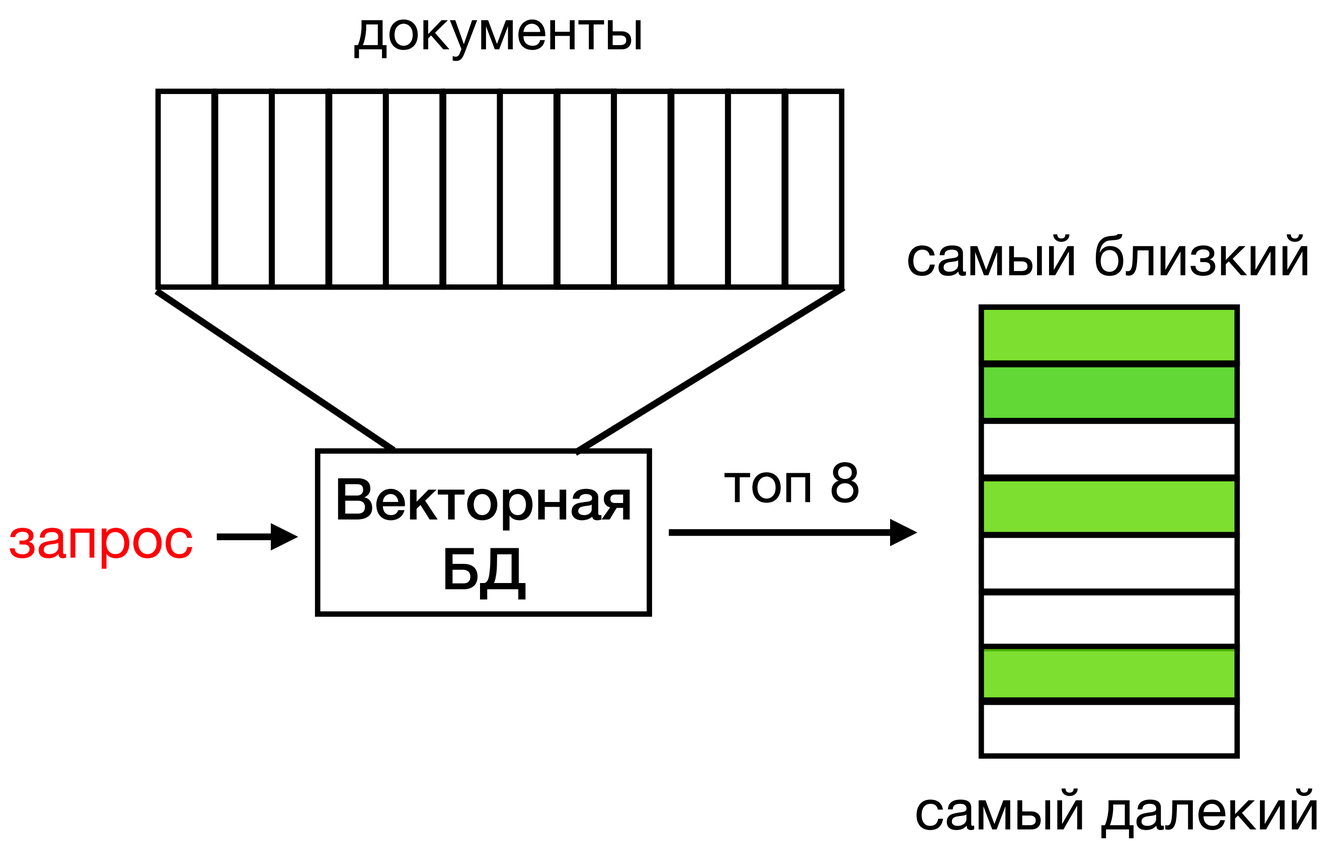
- При формировании контекста вместо маленьких кусков берем их родителей

- Так мы не теряем информацию при сжатии текста и
  в контекст LLM попадает больше релевантного текста



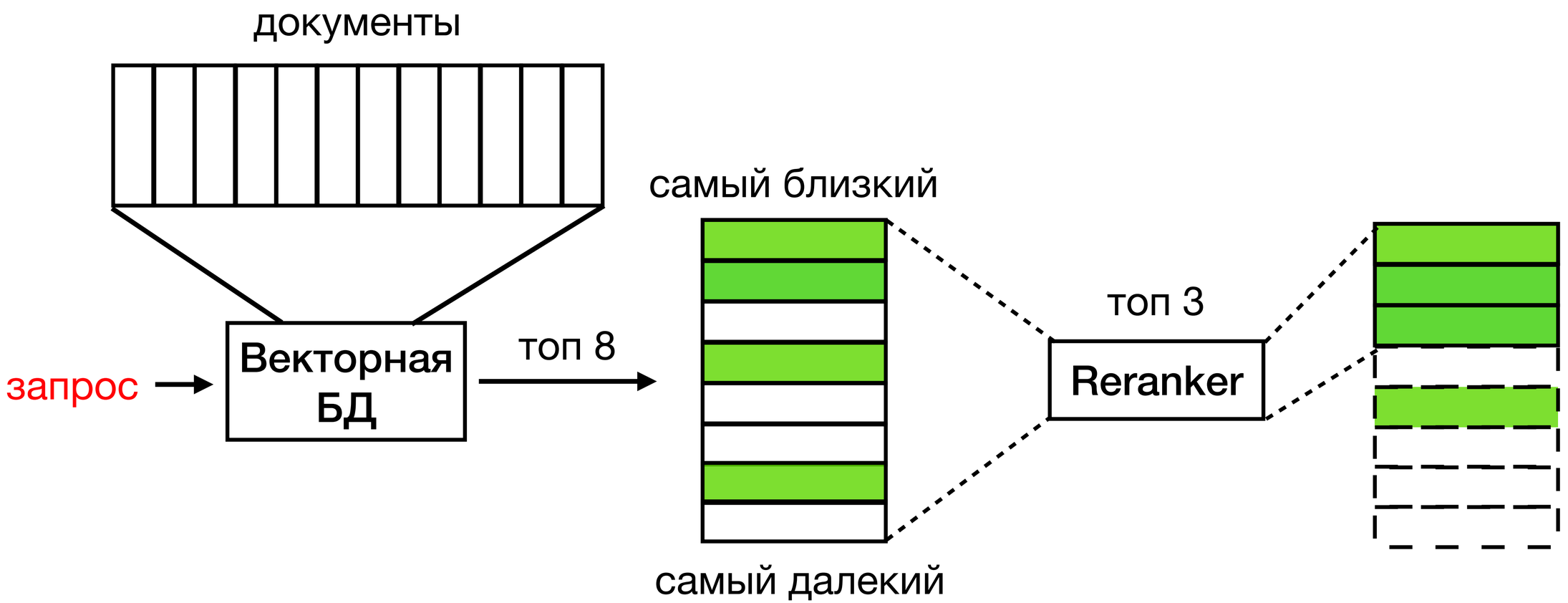
Переранжирование

При сжатии текста в эмбеддинг теряется информация
Из-за этого ранжирование оказывается неоптимальным.



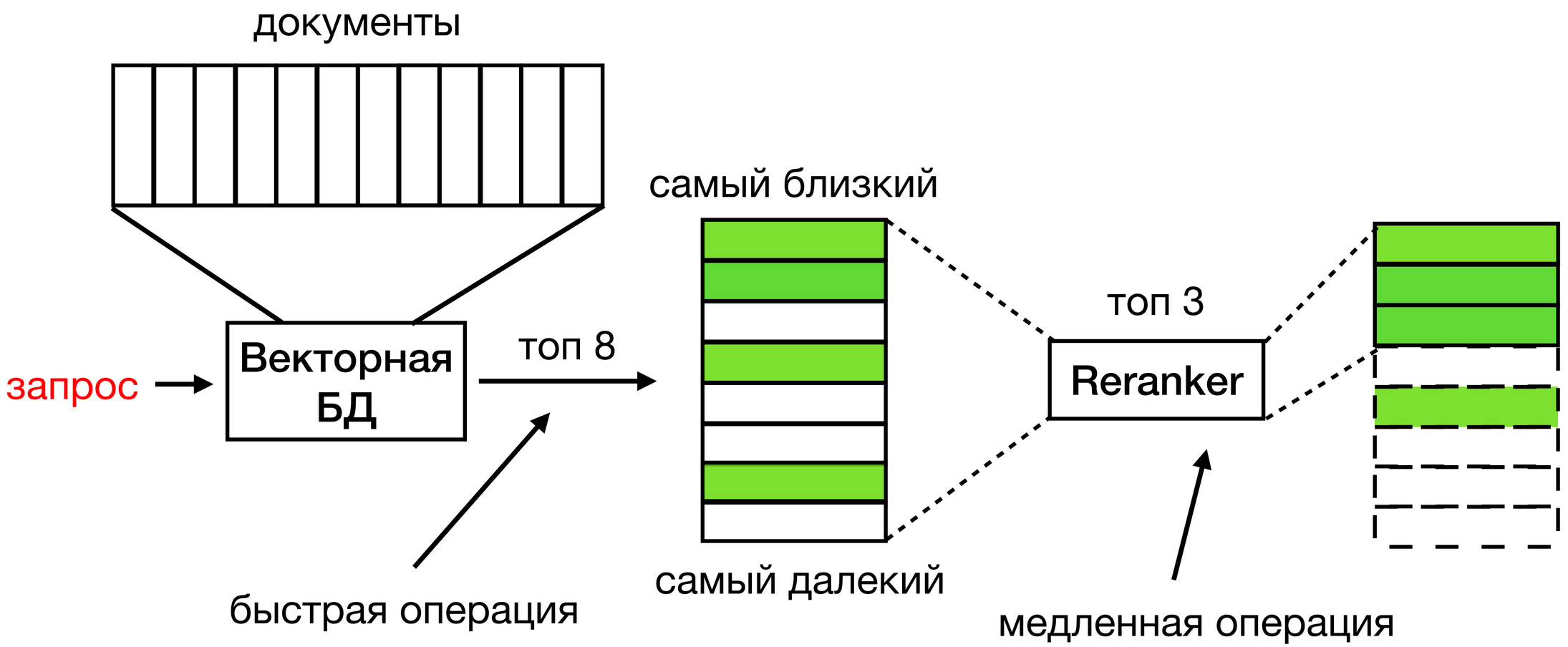
Переранжирование

При сжатии текста в эмбеддинг теряется информация
Из-за этого ранжирование оказывается неоптимальным.



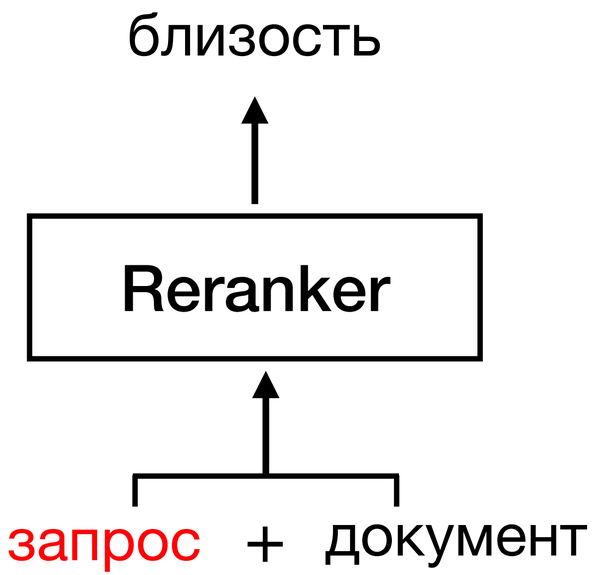
Переранжирование

При сжатии текста в эмбеддинг теряется информация
Из-за этого ранжирование оказывается неоптимальным.



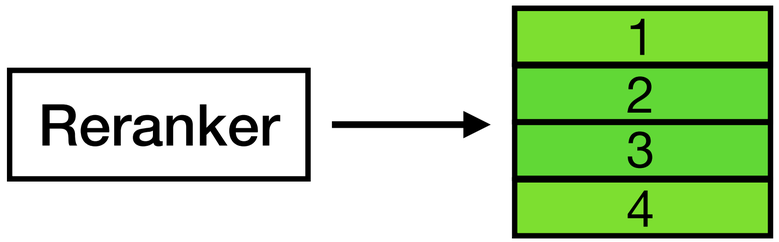
Reranker

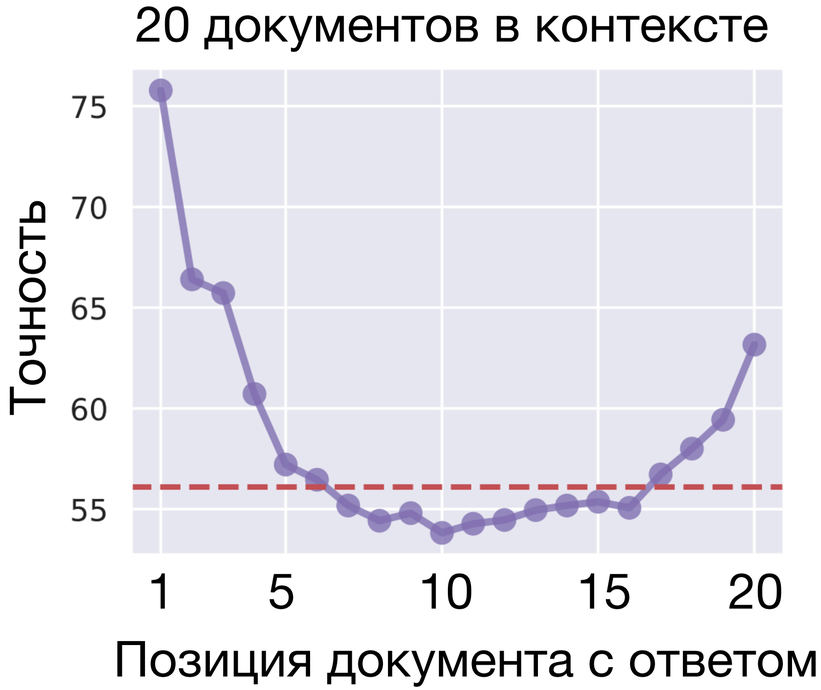
Reranker – трансформераная модель, оценивающая близость двух текстов
Например, BERT, который обучался на задачу Next Sentence Prediction



Порядок подачи документов в контекст

Подавать документы в порядке ранжирования – плохая идея!



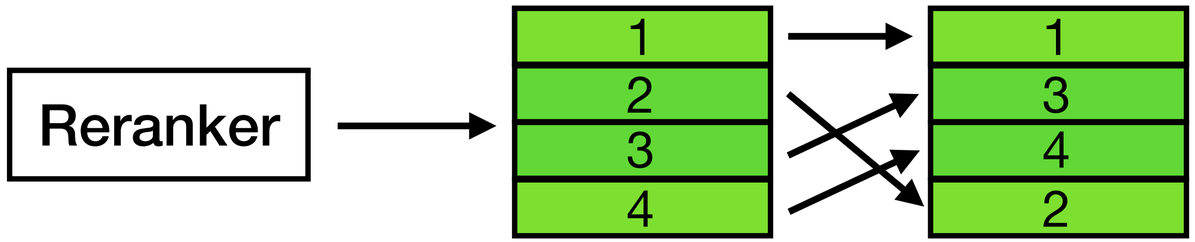


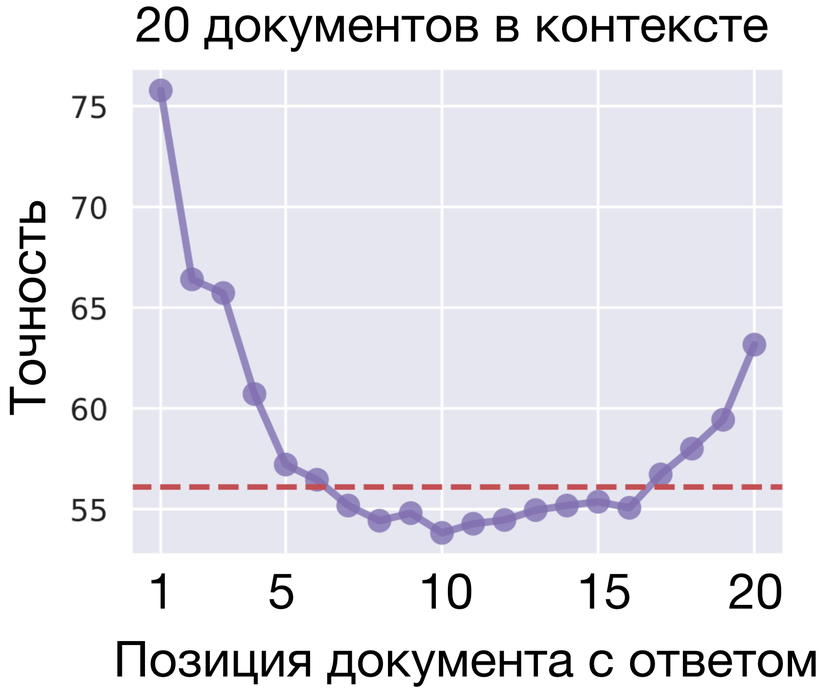
https://arxiv.org/pdf/2307.03172

Порядок подачи документов в контекст

Подавать документы в порядке ранжирования – плохая идея!

Самую важную информацию нужно класть по краям





https://arxiv.org/pdf/2307.03172

Оценка качества RAG

При оценке качестве все составляющие тестируется отдельно:

- Качество извлечения релевантных кусков текста
- Качество генерации текста

Оценка качества RAG

При оценке качестве все составляющие тестируется отдельно:

- Качество извлечения релевантных кусков текста
- Качество генерации текста

Требуется:

- Устойчивость к нерелевантной информации в контексте
- Возможность отказаться от ответа, если знаний недостаточно
- Умение агрегировать информацию из разных источников
- Умение выявлять неверную информацию в контексте

Оценка качества RAG

При оценке качестве все составляющие тестируется отдельно:

- Качество извлечения релевантных кусков текста
- Качество генерации текста

Требуется:

- Устойчивость к нерелевантной информации в контексте
- Возможность отказаться от ответа, если знаний недостаточно
- Умение агрегировать информацию из разных источников
- Умение выявлять неверную информацию в контексте

Все метрики используют LLM

Качество извлечения кусков текста

- **Retrieval Precision** – доля текстов, соответствующих запросу

$$\text{Retrieval Precision} = \frac{\text{Число выбранных релевантных текстов}}{\text{Число выбранных текстов}}$$

- **Retrieval Recall** – доля правильно выбранных текстов от всех, соответствующих запросу

$$\text{Retrieval Precision} = \frac{\text{Число выбранных релевантных текстов}}{\text{Число релевантных текстов}}$$

- Метрики ранжирования: mAP, MRR

Качество генерации текста

- **Answer Relevancy** – насколько ответ соответствует запросу

$$\text{Answer Relevancy} = \frac{\text{Число релевантных утверждений}}{\text{Число утверждений}}$$

- **Faithfulness** – проверка правильности утверждений LLM

$$\text{Faithfulness} = \frac{\text{Число верных утверждений}}{\text{Число утверждений}}$$

Использование параллельных датасетов

Можно использовать размеченные датасеты с правильными ответами

- SQuAD (Stanford Question Answering Dataset)

Бенчмарки:

- CRAG – 3 разные по сложности задачи с наиболее разнообразными вопросами
- FreshQA – 600 вопросов, разделенных на 4 категории
- Natural Questions – 7842 вопроса на основе википедии

Инструменты для RAG

Сборник LLM

Transformers

Векторные базы данных

qdrant

Chroma

Pinecone

milvus

Поиск текста по БД

elasticsearch

Библиотеки для интеграции

LangChain

LlamaIndex

Поиск ближайших векторов

FAISS

Annoy