

Методы построения отраслевых LLM-систем

на примере чат-бота по
корпоративной документации



Константин Архипенко

Kaspersky



Светлана Ищенко

ИСП РАН



Денис Хайбуллин

ИСП РАН

Насколько успешны GenAI и LLM в бизнесе?

2 из 39

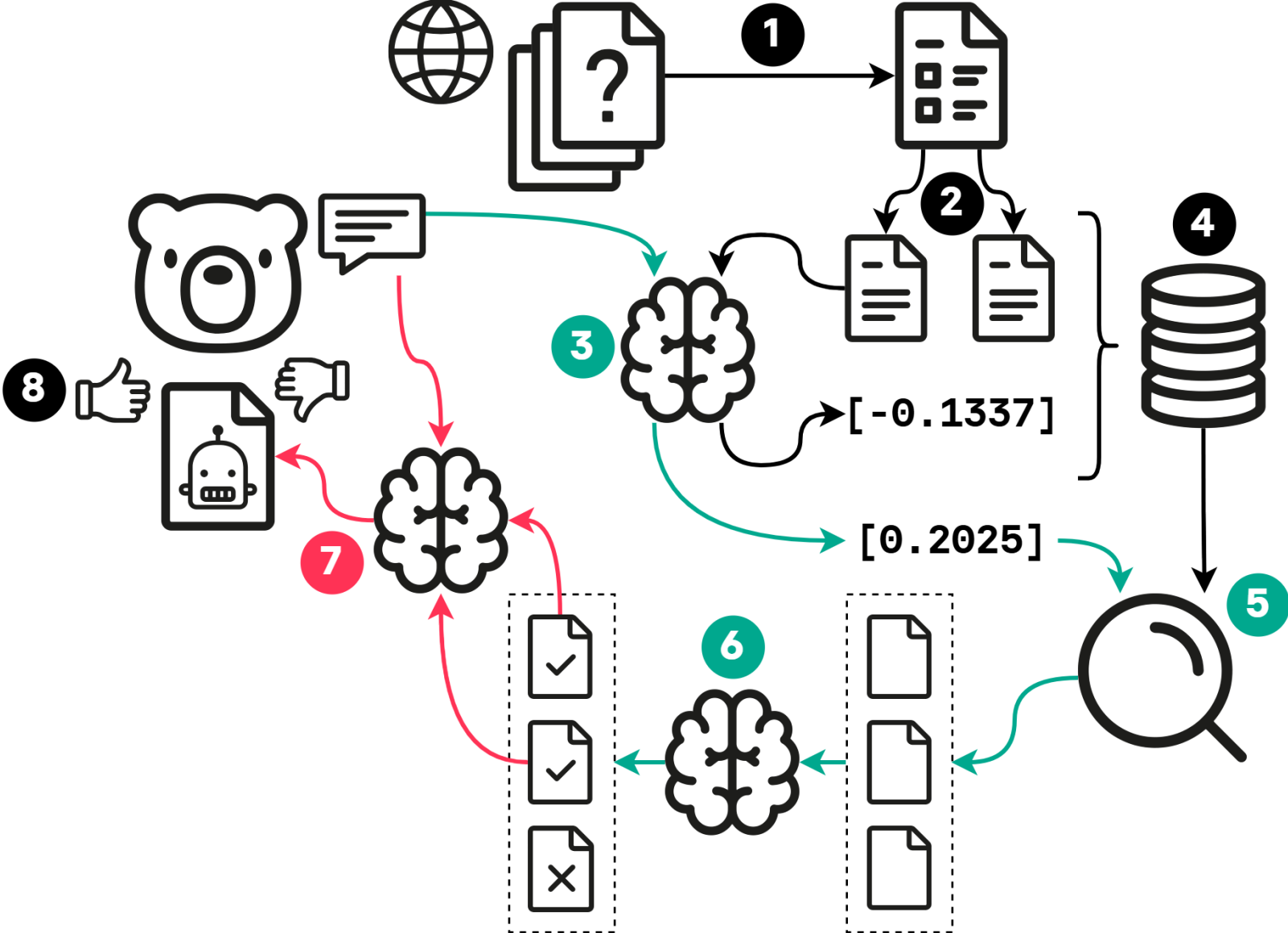
- Только **5%** пилотных проектов GenAI превращаются в продукты, существенно увеличивающие выручку предприятия
- Причина: большие модели сложно адаптировать к требованиям бизнеса:
 - интеграция с узкоспециализированными, постоянно меняющимися данными
 - минимизация галлюцинаций
 - информационная безопасность (все три свойства)
 - производительность и масштабируемость

MIT

The GenAI Divide


**STATE OF AI IN
BUSINESS 2025**

Retrieval-augmented generation (RAG)




- 1** Parsing
- 2** Chunking
- 3** Embedding
- 4** Indexing
- 5** Search
- 6** Reranking
- 7** Generation
- 8** Evaluation

Галлюцинации


Peter Bogdanovich's most famous association with car crashes is **not from a movie he directed or starred in**, but from a real-life accident that he was involved in. He was driving and in a car crash that killed his wife, Polly Platt, and two other people. 

Elaboration:


Real-Life Tragedy:

Peter Bogdanovich was in a car accident in 1979 that resulted in the deaths of Polly Platt, his wife and mother of his two children, and two other passengers. Bogdanovich himself suffered serious injuries. 


Not a Character in a Movie:

While Bogdanovich was a director and actor, and some of his films involved car chases or accidents, it was this tragic real-life event that is most associated with him and car crashes. 

The Last Picture Show:

One of Bogdanovich's most well-known films, "The Last Picture Show," does include a car accident, but it is part of the narrative and not directly related to his personal life. 

Career:

Bogdanovich was known for his work as a director, actor, and film critic. He directed and acted in several films, including "What's Up, Doc?" and "The Last Picture Show," among others. 

<https://www.reddit.com/r/BetterOffline/comments/1kljkh/>

Безопасность



<https://www.reuters.com/investigates/special-report/ai-chatbots-cyber>

Reuters and a Harvard University researcher used top chatbots to plot a simulated phishing scam – from composing emails to tips on timing – and tested it on 108 elderly volunteers. The bots' persuasive performance shows how AI is arming criminals for industrial-scale fraud.

Yes, GitHub's Copilot can Leak (Real) Secrets

Researchers successfully extracted valid hard-coded secrets from Copilot and CodeWhisperer, shedding light on a novel security risk associated with the proliferation of secrets.

<https://blog.gitguardian.com/yes-github-copilot-can-leak-secrets>

Почему RAG?

RAG позволяет обрабатывать динамические отраслевые данные.

Без RAG нам придется выбирать из следующего:

- использовать LLM как есть
 - несоответствие данных pre-training и fine-tuning предметной области → **галлюцинации**
- подавать все отраслевое знание как часть контекста LLM
 - переполнение контекста — **context rot**
- дообучение (fine-tuning) LLM
 - в большей степени пригодно для обучения **структуре ответов** LLM, а не **новому знанию**
 - для второй цели само по себе **ненадежно**
 - катастрофическое забывание (catastrophic forgetting)

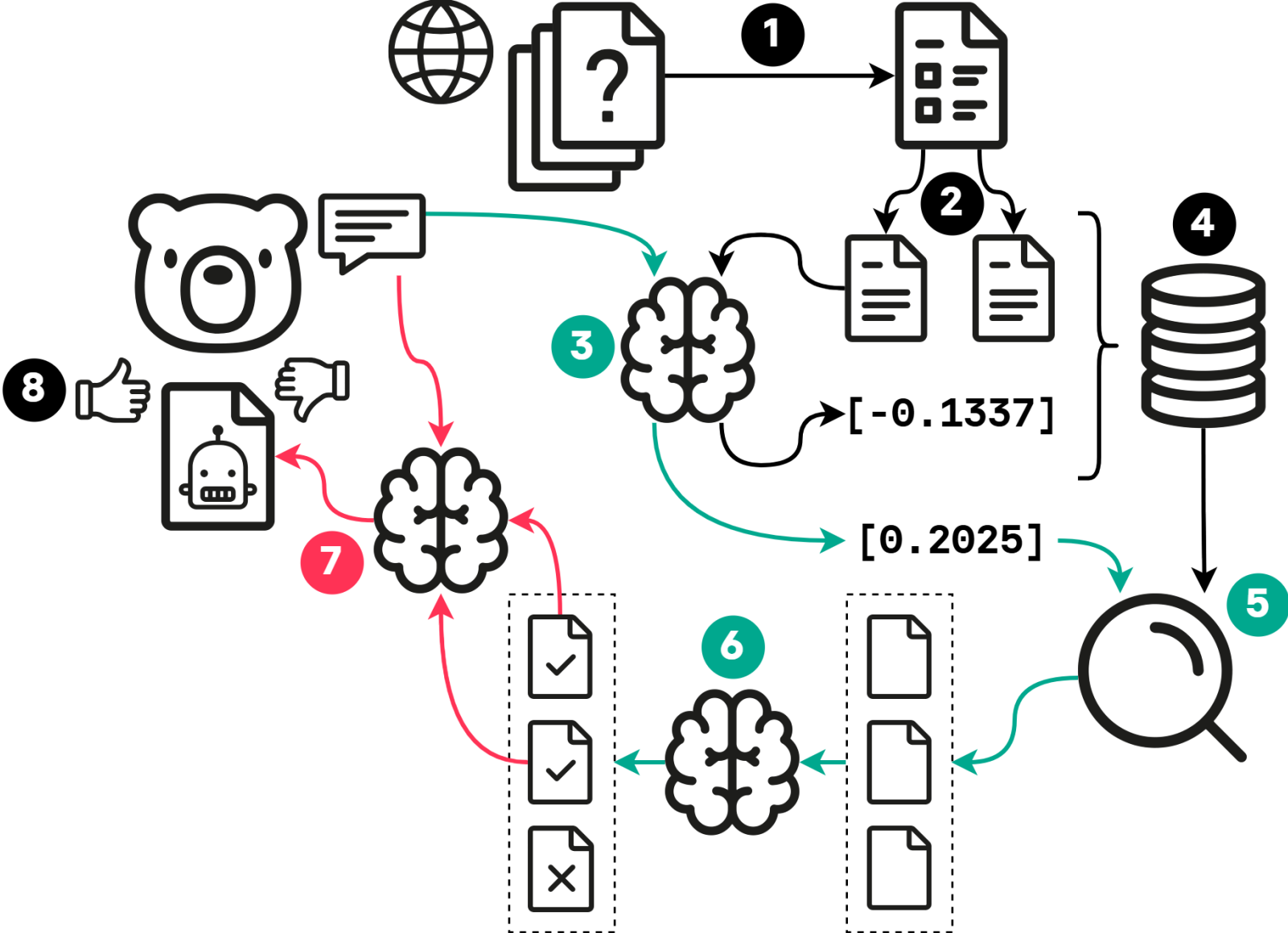
Почему RAG? [2]

RAG меньше галлюцинирует и более безопасен.

- LLM в составе RAG опирается на **базу документов**, а не собственные «знания»
- Собственные «знания» LLM черпает из Сети:
 - веб-сайты, социальные сети
 - телеметрия ОС и приложений...поэтому они могут включать:
 - **зловредную информацию** (призывы к насилию, 18+, запрещенные вещества)
 - **конфиденциальную информацию**
- RAG снижает нагрузку на инфраструктуру: можно использовать не самые большие LLM и только по назначению

- Сделали PoC чат-бота по документации к продуктам Kaspersky
 - 63 продукта, 193 справки (ru + en), 40209 страниц
- Столкнулись с вызовами:
 - для парсинга и чанкинга базы документов существующие инструменты оказались неэффективны, поэтому мы написали свой
 - оценка качества RAG — открытая научная проблема; существующие метрики и тестовые датасеты далеки от реальных пользователей
 - в нашей задаче датасет для оценки качества отсутствовал, поэтому мы его успешно сгенерировали
 - развернуть ИИ-компоненты (те, которые мы хотели) чат-бота оказалось нетривиально:
 - обходили баги и ограничения существующего ПО
 - писали свои системные промпты

Retrieval-augmented generation (RAG)



- 1** Parsing
- 2** Chunking
- 3** Embedding
- 4** Indexing
- 5** Search
- 6** Reranking
- 7** Generation
- 8** Evaluation



Поиск

Что нового

Введение

+ О KasperskyOS Community Edition

+ Обзор KasperskyOS

Краткое руководство разработчика решений на базе KasperskyOS

— Начало работы

Использование Docker-контейнера

Установка и удаление

Настройка среды разработки

+ Расширение KasperskyOS SDK Extension for Visual Studio Code

+ Сборка и запуск примеров

— Разработка под KasperskyOS

+ Запуск процессов

+ Файловые системы и сеть

KasperskyOS API > Коды возврата

Коды возврата

Общие сведения

В решении на базе KasperskyOS коды возврата функций различных API (например, API библиотек [libkos](#) и [kdf](#), драйверов, транспортного кода, прикладного ПО) имеют тип 32-битного знакового целого числа. Этот тип определен в заголовочном файле `sysroot-*-kos/include/rtl/retcode.h` из состава KasperskyOS SDK так:

```
typedef __INT32_TYPE__ Retcode;
```

Множество кодов возврата состоит из кода успеха со значением 0 и кодов ошибок. Код ошибки интерпретируется как структура данных, формат которой описан в заголовочном файле `sysroot-*-kos/include/rtl/retcode.h` из состава KasperskyOS SDK. Этот формат предусматривает наличие нескольких полей, которые содержат не только сведения о результатах вызова функции, но и следующую дополнительную информацию:

- Флаг в поле `Customer`, сигнализирующий о том, что код ошибки определен разработчиками решения на базе KasperskyOS, а не разработчиками ПО из состава KasperskyOS SDK.

Благодаря флагу в поле `Customer` разработчики решения на базе KasperskyOS и разработчики ПО из состава KasperskyOS SDK могут определять коды ошибок из непересекающихся множеств.

```
<div class="cont">
<h2 id="t233386" class="heading2">Коды возврата</h2>
<p class="introheading">Общие сведения</p>
<p class="bodytext">В решении на базе KasperskyOS коды возврата функций различных API
(например, API библиотек <code class="programcharacterkos"><a id="h226099"
class="crossreferencetemplate" href="libkos.htm" target="_self">libkos</a></code> и
<code class="programcharacterkos">kdf</code>, драйверов, транспортного кода, прикладного
ПО) имеют тип 32-битного знакового целого числа. Этот тип определен в заголовочном файле
<code class="programcharacterkos">sysroot-*-kos/include/rtl/retcode.h</code> из состава
KasperskyOS&nbsp;SDK так:</p>
<p class="programckos">typedef __INT32_TYPE__ Retcode;</p>
<p class="bodytext">Множество кодов возврата состоит из кода успеха со значением <code
class="programcharacterkos">0</code> и кодов ошибок. Код ошибки интерпретируется как
структура данных, формат которой описан в заголовочном файле <code
class="programcharacterkos">sysroot-*-kos/include/rtl/retcode.h</code> из состава
KasperskyOS&nbsp;SDK. Этот формат предусматривает наличие нескольких полей, которые
содержат не только сведения о результатах вызова функции, но и следующую дополнительную
информацию:</p>
<ul class="listbullet1"><li class="listbullet1">Флаг в поле <code
class="programcharacterkos">Customer</code>, сигнализирующий о том, что код ошибки
определен разработчиками решения на базе KasperskyOS, а не разработчиками ПО из состава
KasperskyOS&nbsp;SDK.<p class="listcontinue1">Благодаря флагу в поле <code
class="programcharacterkos">Customer</code> разработчики решения на базе KasperskyOS и
разработчики ПО из состава KasperskyOS&nbsp;SDK могут определять коды ошибок из
непересекающихся множеств.</p>
</li> ...
```



Чем это распарсить?

Особенности нашей базы:

- Специфические HTML-элементы: всплывающие подсказки, сворачиваемые секции
- Не всегда можно доверять именам HTML-тегов
- DOM-дерево **мутирует** в браузере
- Мы берем **исходные** .htm-файлы



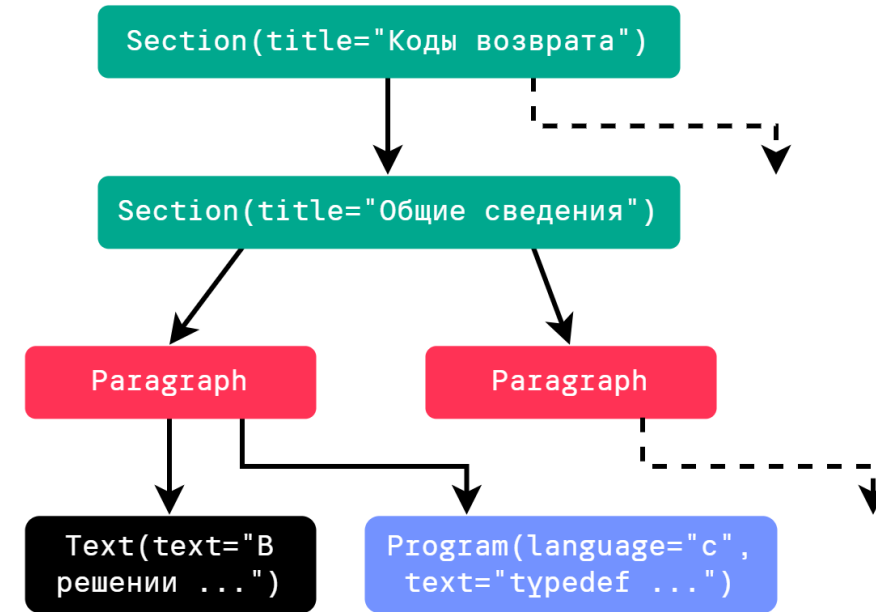
Docling (IBM)

- Считается самым качественным парсером документов
 - форматы PDF, DOCX, HTML и др.
- Использован победителем Enterprise RAG Challenge
<https://abdullin.com/ilya/how-to-build-best-rag>
- Строит **дерево** документа
 - секции, абзацы, списки, таблицы
- Имеет **иерархический чанкер** для RAG
- Мы попробовали запустить парсер Docling:
 - время парсинга страницы KOS CE «Примеры» — **0,66 с**
 - время нашего парсера для этой страницы — **0,02 с**
 - плохо переносит особенности нашей базы, не распознает листинги программ

Наш парсер документации Kaspersky



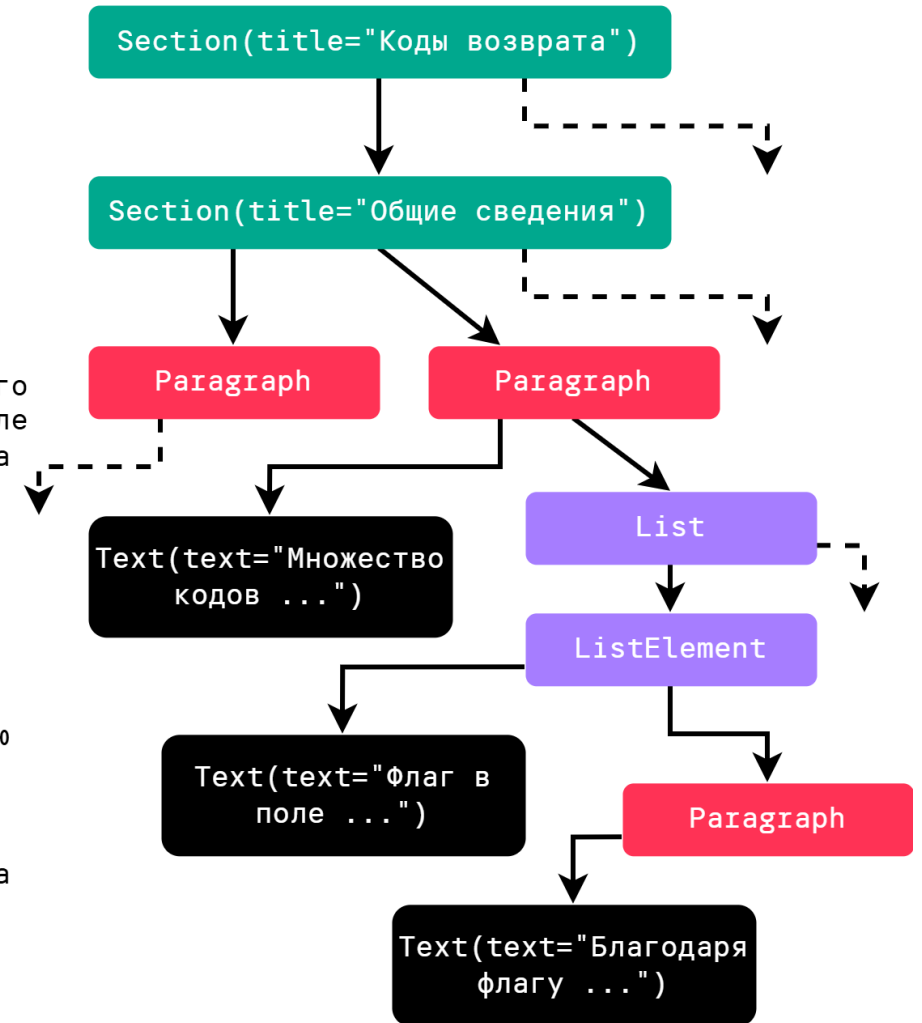
```
<div class="cont">
<h2 id="t233386" class="heading2">Коды возврата</h2>
<p class="introheading">Общие сведения</p>
<p class="bodytext">В решении на базе KasperskyOS коды возврата функций различных API
(например, API библиотек <code class="programcharacterkos"><a id="h226099"
class="crossreferencetemplate" href="libkos.htm" target="_self">libkos</a></code> и
<code class="programcharacterkos">kdf</code>, драйверов, транспортного кода, прикладного
ПО) имеют тип 32-битного знакового целого числа. Этот тип определен в заголовочном файле
<code class="programcharacterkos">sysroot-*-kos/include/rtl/retcode.h</code> из состава
KasperskyOS&nbsp;SDK так:</p>
<p class="programckos">typedef __INT32_TYPE__ Retcode;</p>
<p class="bodytext">Множество кодов возврата состоит из кода успеха со значением <code
class="programcharacterkos">0</code> и кодов ошибок. Код ошибки интерпретируется как
структура данных, формат которой описан в заголовочном файле <code
class="programcharacterkos">sysroot-*-kos/include/rtl/retcode.h</code> из состава
KasperskyOS&nbsp;SDK. Этот формат предусматривает наличие нескольких полей, которые
содержат не только сведения о результатах вызова функции, но и следующую дополнительную
информацию:</p>
<ul class="listbullet1"><li class="listbullet1">Флаг в поле <code
class="programcharacterkos">Customer</code>, сигнализирующий о том, что код ошибки
определен разработчиками решения на базе KasperskyOS, а не разработчиками ПО из состава
KasperskyOS&nbsp;SDK.<p class="listcontinue1">Благодаря флагу в поле <code
class="programcharacterkos">Customer</code> разработчики решения на базе KasperskyOS и
разработчики ПО из состава KasperskyOS&nbsp;SDK могут определять коды ошибок из
непересекающихся множеств.</p>
</li> ...
</div>
```



Наш парсер документации Kaspersky [2]



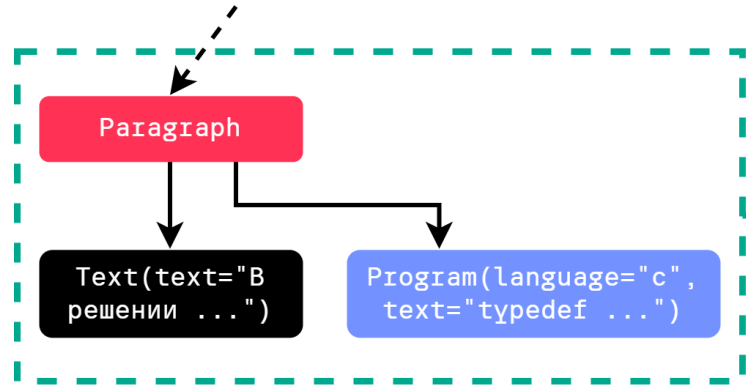
```
<div class="cont">
<h2 id="t233386" class="heading2">Коды возврата</h2>
<p class="introheading">Общие сведения</p>
<p class="bodytext">В решении на базе KasperskyOS коды возврата функций различных API
(например, API библиотек <code class="programcharacterkos"><a id="h226099"
class="crossreferencetemplate" href="libkos.htm" target="_self">libkos</a></code> и
<code class="programcharacterkos">kdf</code>, драйверов, транспортного кода, прикладного
ПО) имеют тип 32-битного знакового целого числа. Этот тип определен в заголовочном файле
<code class="programcharacterkos">sysroot-*-kos/include/rtl/retcode.h</code> из состава
KasperskyOS&nbsp;SDK так:</p>
<p class="programckos">typedef __INT32_TYPE__ Retcode;</p>
<p class="bodytext">Множество кодов возврата состоит из кода успеха со значением <code
class="programcharacterkos">0</code> и кодов ошибок. Код ошибки интерпретируется как
структура данных, формат которой описан в заголовочном файле <code
class="programcharacterkos">sysroot-*-kos/include/rtl/retcode.h</code> из состава
KasperskyOS&nbsp;SDK. Этот формат предусматривает наличие нескольких полей, которые
содержат не только сведения о результатах вызова функции, но и следующую дополнительную
информацию:</p>
<ul class="listbullet1"><li class="listbullet1">Флаг в поле <code
class="programcharacterkos">Customer</code>, сигнализирующий о том, что код ошибки
определен разработчиками решения на базе KasperskyOS, а не разработчиками ПО из состава
KasperskyOS&nbsp;SDK.<p class="listcontinue1">Благодаря флагу в поле <code
class="programcharacterkos">Customer</code> разработчики решения на базе KasperskyOS и
разработчики ПО из состава KasperskyOS&nbsp;SDK могут определять коды ошибок из
непересекающихся множеств.</p>
</li> ...
</div>
```



Иерархический чанкинг

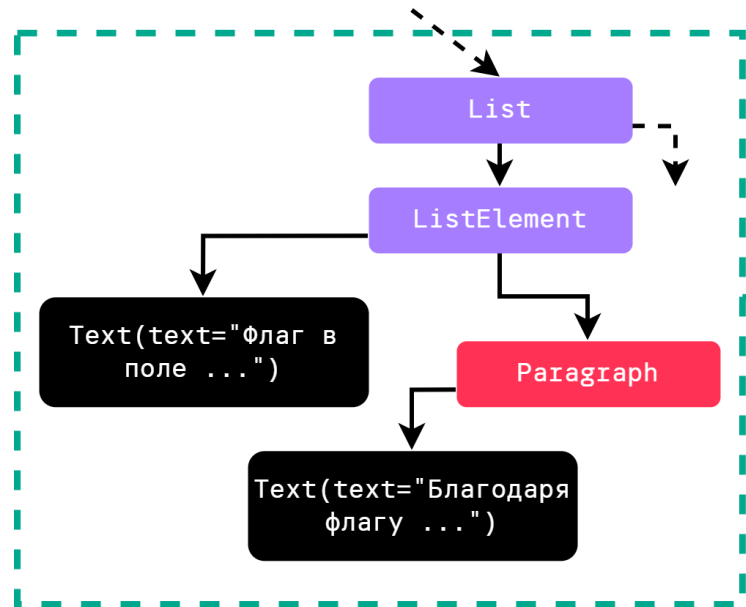
- Нам нужны **маленькие** чанки, чтобы отвечать на вопросы пользователя, касающиеся единичных фактов
- Нам нужны **большие** чанки для суммаризации и логических выводов
- Построим **иерархию** чанков, где маленькие являются частью больших
 - дублирование информации, которое нужно учесть на этапе поиска (retrieval)
- В каких местах разбивать документы?
 - простой подход (LlamaIndex): двойной `\n`, границы предложений
 - в Docling и у нас: для **каждой вершины** дерева (результата парсера) строится свой чанк
 - конвертация в Markdown поддерева с корнем в этой вершине

Иерархический чанкинг [2]



В решении на базе KasperskyOS коды возврата функций различных API (например, API библиотек libkos и `kdf`, драйверов, транспортного кода, прикладного ПО) имеют тип 32-битного знакового целого числа. Этот тип определен в заголовочном файле `sysroot-*-kos/include/rtl/retcode.h` из состава KasperskyOS SDK так:

```
```c
typedef __INT32_TYPE__ Retcode;
```
```



* Флаг в поле `Customer`, сигнализирующий о том, что код ошибки определен разработчиками решения на базе KasperskyOS, а не разработчиками ПО из состава KasperskyOS SDK. Благодаря флагу в поле `Customer` разработчики решения на базе KasperskyOS и разработчики ПО из состава KasperskyOS SDK могут определять коды ошибок из непересекающихся множеств.

* Глобальный идентификатор кода ошибки в поле `Space`. Глобальные идентификаторы позволяют определять непересекающиеся множества кодов ошибок. Коды ошибок могут быть общими и специфичными. Общие коды ошибок могут использоваться в API любых компонентов решения и в API любых составных частей компонентов решения (например, драйвер или VFS могут быть составной частью компонента решения). Специфичные коды ошибок используются в API одного или нескольких компонентов решения или в API одной или нескольких составных частей компонентов решения.

Например, идентификатору `RC_SPACE_GENERAL` соответствуют коды общих ошибок, идентификатору `RC_SPACE_KERNEL` соответствуют коды ошибок ядра, идентификатору `RC_SPACE_DRIVERS` соответствуют коды ошибок драйверов.

* Локальный идентификатор кода ошибки в поле `Facility`. Локальные идентификаторы позволяют определять непересекающиеся подмножества кодов ошибок в рамках множества кодов ошибок, которые соответствуют одному глобальному идентификатору. Например, множество кодов ошибок с глобальным идентификатором `RC_SPACE_DRIVERS` включает непересекающиеся подмножества кодов ошибок с локальными идентификаторами `RC_FACILITY_I2C`, `RC_FACILITY_USB`, `RC_FACILITY_BLKDEV`.

результат — более 800 тыс. чанков размером от 32 до 2048 токенов

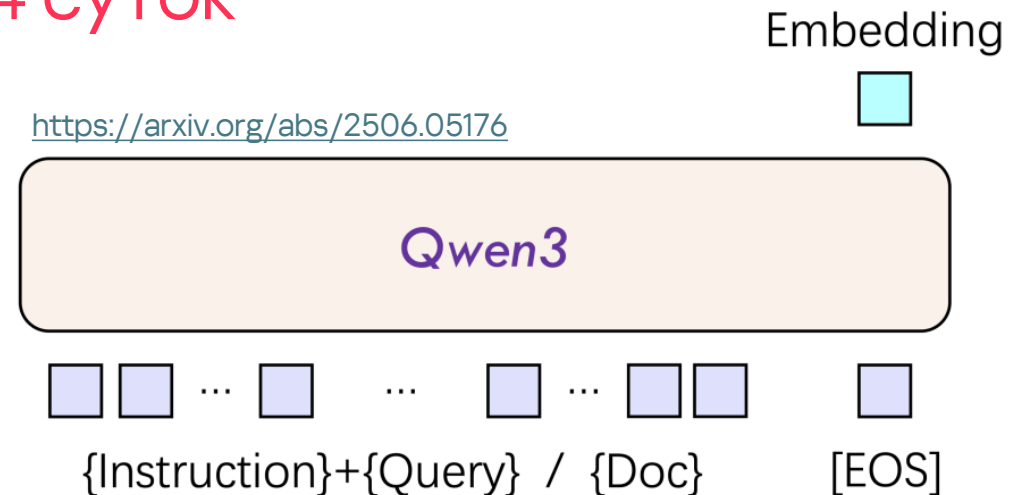
- Перевод чанков в вещественное векторное пространство
- Используем модель **Qwen3-Embedding-8B**:
 - основана на Qwen3-8B-Base — трансформере-декодере, прошедшем pre-training
 - векторы размера **4096** — последние скрытые состояния для токена [EOS]
 - контрастное обучение на парах «запрос-документ», в основном синтетических
- Время векторизации нашей базы **~4 суток**

В решении на базе KasperskyOS коды возврата функций различных API (например, API библиотек libkos и `kdf`, драйверов, транспортного кода, прикладного ПО) имеют тип 32-битного знакового целого числа. Этот тип определен в заголовочном файле `sysroot-*
kos/include/rtl/retcode.h` из состава KasperskyOS SDK так:

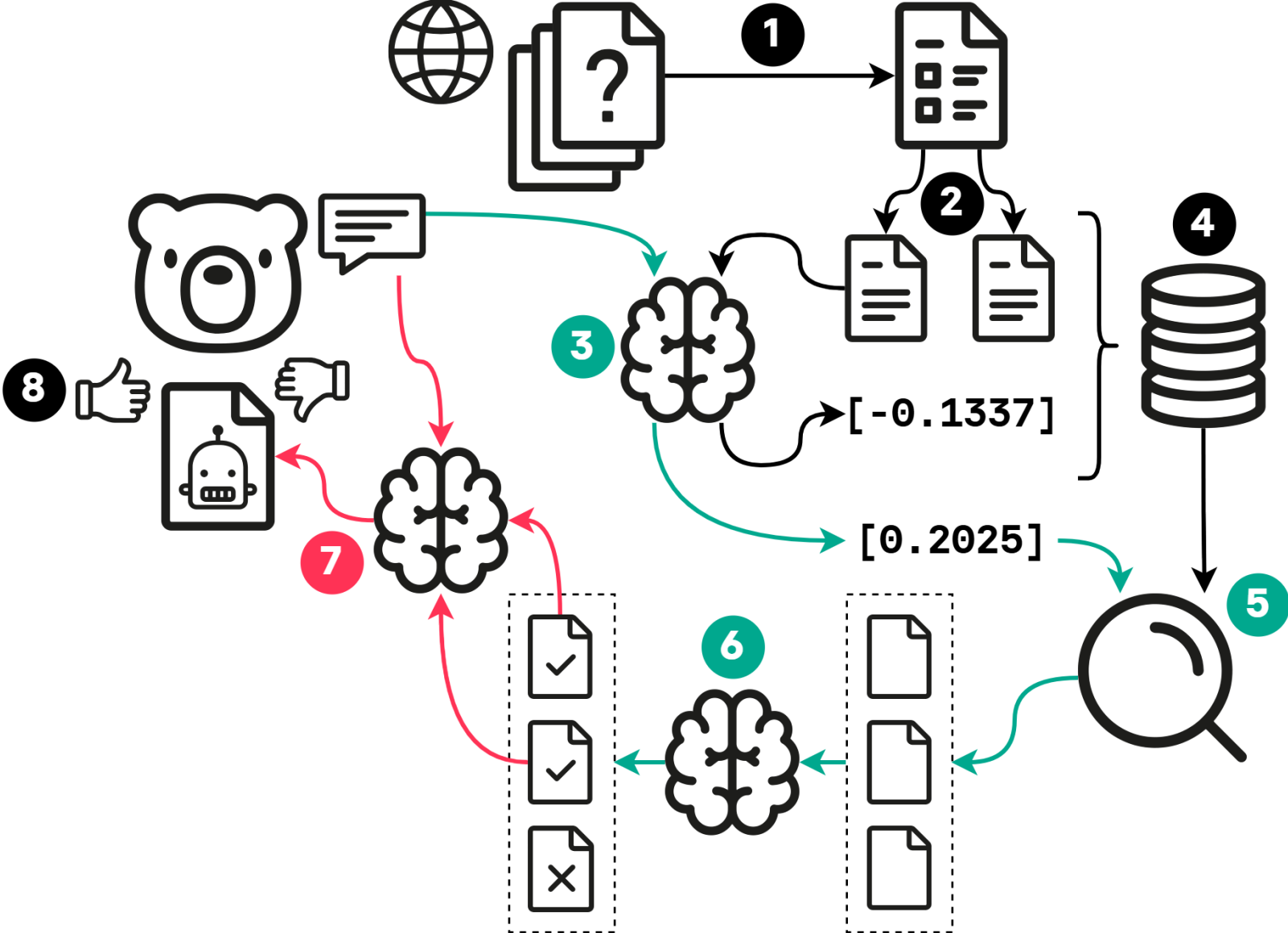
```
...  
typedef __INT32_TYPE__ Retcode;  
...
```



[0.01682838, -0.013170436, ..., -0.008415891, -0.017338008]



Retrieval-augmented generation (RAG)



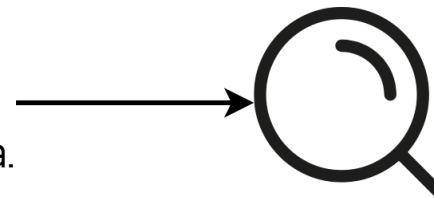
- 1** Parsing
- 2** Chunking
- 3** Embedding
- 4** Indexing
- 5** Search
- 6** Reranking
- 7** Generation
- 8** Evaluation


- Сравнение эмбедингов вопроса пользователя и чанков
- Быстрый (приближенный) поиск k ближайших соседей
 - метрика близости — косинус между эмбедингами
 - мы используем довольно большое значение k (до 50)
 - векторная БД — Qdrant (<https://qdrant.tech>)
- **Не используем** полнотекстовый поиск (BM25)
- **Автоматическое слияние** (auto-merging) чанков:
 - если более 50% текста вершины-чанка попало в top- k в виде чанков-потомков, то **заменяем** этой вершиной всех потомков
 - модификация `AutoMergingRetriever` из LlamaIndex (в нашем случае чанки-братья могут быть существенно разной длины)






Перечисли заголовочные файлы KasperskyOS, относящиеся к кодам возврата.




 Множество кодов возврата состоит из кода успеха со значением 0 и кодов ошибок. Код ошибки интерпретируется как структура данных, формат которой описан в заголовочном файле ``sysroot-*-kos/include/rtl/retcode.h`` из состава KasperskyOS SDK.

0.9995
Paragraph

 Макросы ``RC_GET_CUSTOMER()``, ``RC_GET_SPACE()``, ``RC_GET_FACILITY()`` и ``RC_GET_CODE()``, определенные в заголовочном файле ``sysroot-*-kos/include/rtl/retcode.h`` из состава KasperskyOS SDK, позволяют читать поля структуры кода ошибки.

0.9975
Paragraph

 Макросы ``RETCODE_HR_PARAMS()`` и ``RETCODE_HR_FMT()``, определенные в заголовочном файле ``sysroot-*-kos/include/rtl/retcode_hr.h`` из состава KasperskyOS SDK, используются для форматированного вывода сведений об ошибке.

0.9944
Paragraph

Section

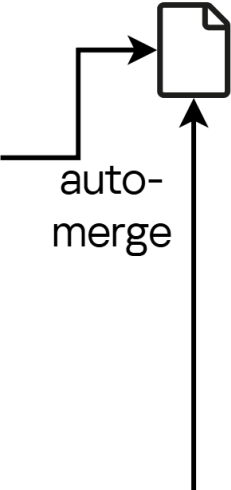
Paragraph Paragraph

Чтение полей структуры кода ошибки

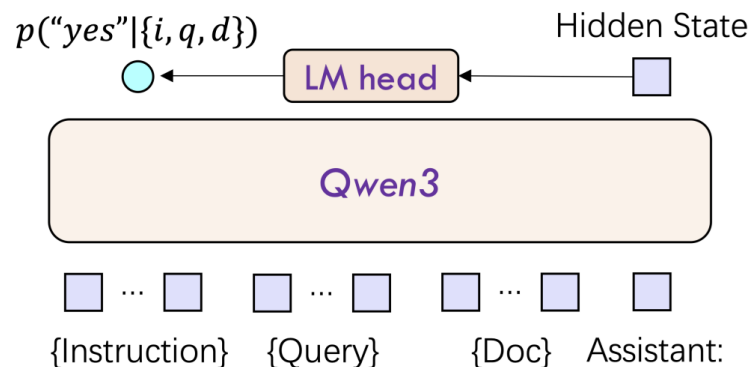
Макросы ``RC_GET_CUSTOMER()``, ``RC_GET_SPACE()``, ``RC_GET_FACILITY()`` и ``RC_GET_CODE()``, определенные в заголовочном файле ``sysroot-*-kos/include/rtl/retcode.h`` из состава KasperskyOS SDK, позволяют читать поля структуры кода ошибки.

Макросы ``RETCODE_HR_PARAMS()`` и ``RETCODE_HR_FMT()``, определенные в заголовочном файле ``sysroot-*-kos/include/rtl/retcode_hr.h`` из состава KasperskyOS SDK, используются для форматированного вывода сведений об ошибке.

auto-merge



- Reranker — ИИ-модель, оценивающая релевантность чанка вопросу пользователя
 - оценка более точная, чем сравнение эмбеддингов, за счет **совместного** анализа чанка и вопроса
 - диапазон релевантности обычно $[0, 1]$, где 1 — «да», 0 — «нет»
- Мы используем **Qwen3-Reranker-4B**:
 - отличие от большинства реранкеров — она является **декодером** (использует causal attention), основываясь на Qwen3-4B-Base
 - сравнение логитов токенов «yes» и «no» — выходов Language modeling head



$$\text{score}(q, d) = \frac{e^{P(\text{yes}|I, q, d)}}{e^{P(\text{yes}|I, q, d)} + e^{P(\text{no}|I, q, d)}}$$

Реранкинг [2]



Чтение полей структуры кода ошибки

Макросы ``RC_GET_CUSTOMER()``, ``RC_GET_SPACE()``, ``RC_GET_FACILITY()`` и ``RC_GET_CODE()``, определенные в заголовочном файле ``sysroot-*-kos/include/rtl/retcode.h`` из состава KasperskyOS SDK, позволяют читать поля структуры кода ошибки.

Макросы ``RETCODE_HR_PARAMS()`` и ``RETCODE_HR_FMT()``, определенные в заголовочном файле ``sysroot-*-kos/include/rtl/retcode_hr.h`` из состава KasperskyOS SDK, используются для форматированного вывода сведений об ошибке.

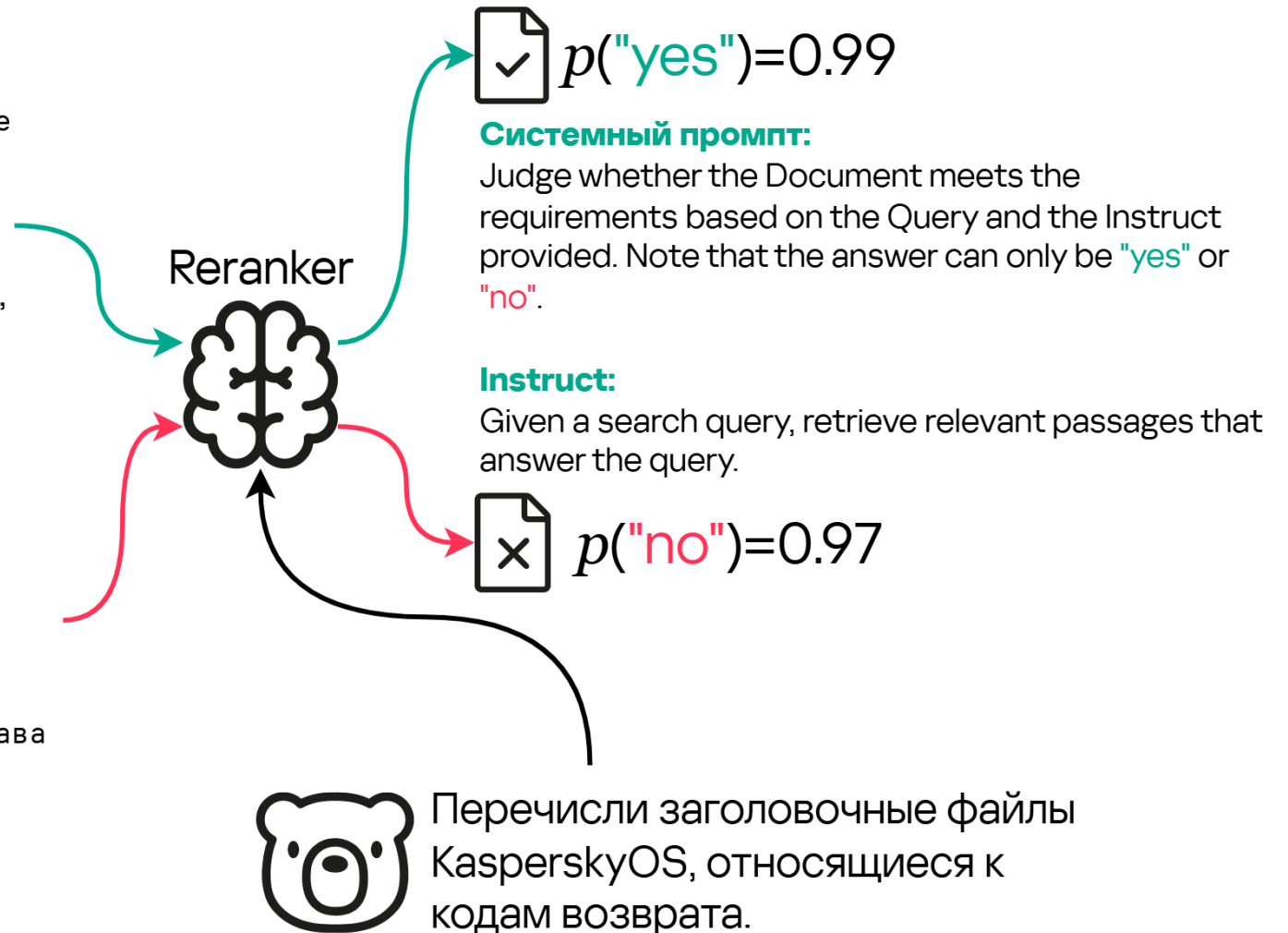


API KasperskyOS

KasperskyOS предоставляет следующие API:

- * Собственные API:
 - высокоуровневые API (заголовочные файлы в директории ``sysroot-*-kos/include/kos`` из состава KasperskyOS Community Edition);
 - низкоуровневые API (заголовочные файлы в директории ``sysroot-*-kos/include/coresrv`` из состава KasperskyOS Community Edition).

...



Генерация ответа



Множество кодов возврата состоит из кода успеха со значением 0 и кодов ошибок. Код ошибки интерпретируется как структура данных, формат которой описан в заголовочном файле `sysroot-*-kos/include/rtl/retcode.h` из состава KasperskyOS SDK.



Чтение полей структуры кода ошибки

Макросы `RC_GET_CUSTOMER()`, `RC_GET_SPACE()`, `RC_GET_FACILITY()` и `RC_GET_CODE()`, определенные в заголовочном файле `sysroot-*-kos/include/rtl/retcode.h` из состава KasperskyOS SDK, позволяют читать поля структуры кода ошибки.

Макросы `RETCODE_HR_PARAMS()` и `RETCODE_HR_FMT()`, определенные в заголовочном файле `sysroot-*-kos/include/rtl/retcode_hr.h` из состава KasperskyOS SDK, используются для форматированного вывода сведений об ошибке.

Context information is below.

`{context_str}`

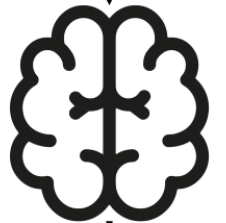
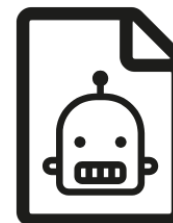
Given the context information and not prior knowledge, answer the query.

Query: `{query_str}`

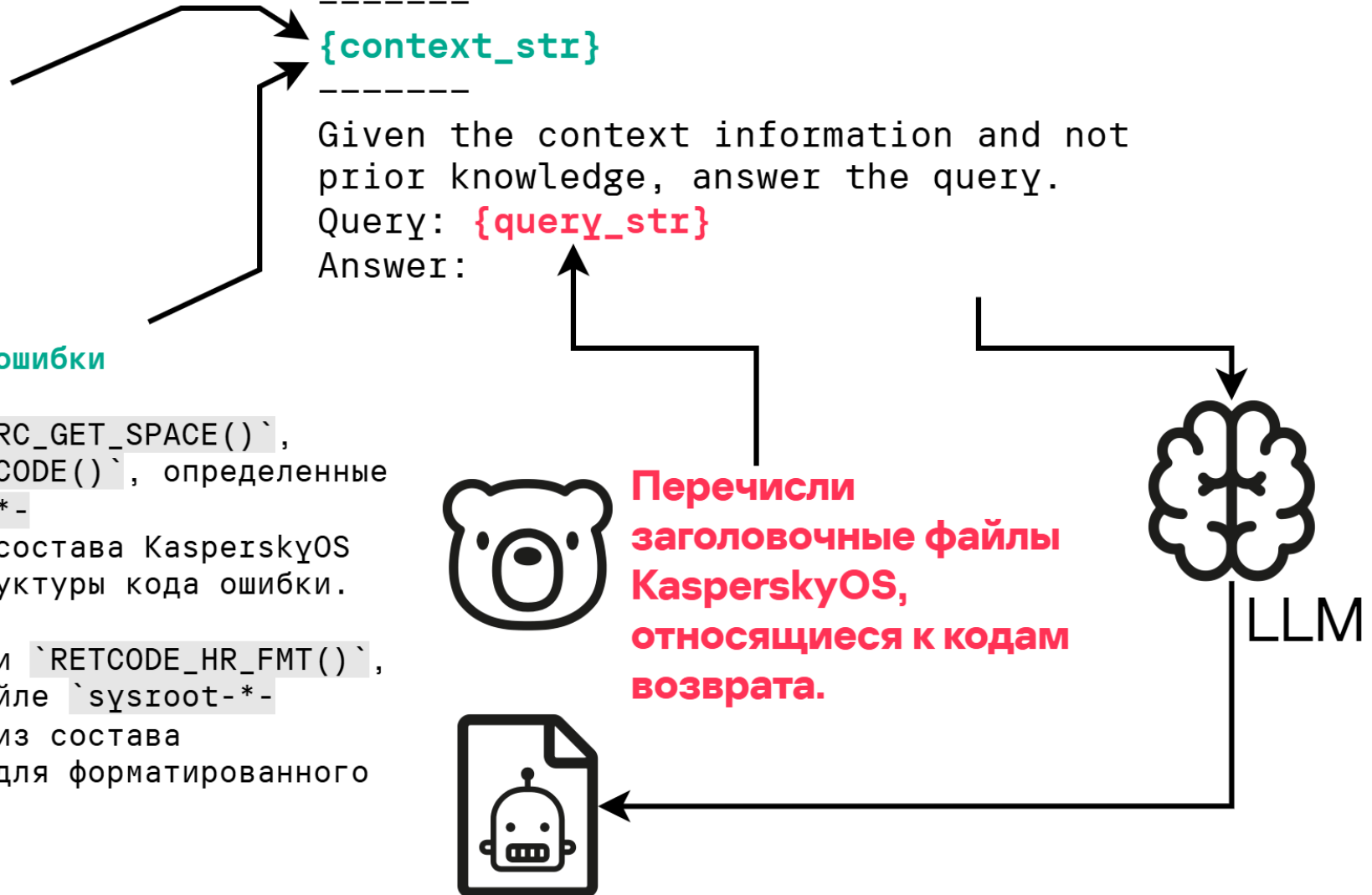
Answer:



Перечисли заголовочные файлы KasperskyOS, относящиеся к кодам возврата.



LLM



Генерация ответа [2]

- LLM для генерации — [Qwen3-30B-A3B-Instruct-2507](#) (MoE)
- top-*p* семплирование с температурой 0,1–0,2
- [Собственные системные промпты](#) (ru + en):

Ты – ИИ-ассистент, который отвечает пользователю только на основе предоставленных текстов из документов о продукте **{product}** компании **Лаборатория Касперского**.

Если вопрос не связан с документацией Касперского и уходит от темы, пиши, что ты отвечаешь только по документации.

Если в контексте нет точной информации для ответа – честно сообщи об этом и предложи возможные варианты поиска.

Говори только на русском языке и только о документации. Если будут говорить и спрашивать не по твоей теме, отвечай, что это не твой профиль задач.

Твои правила:

1. Говори только на русском языке.
2. Используй только факты из переданного контекста.
3. Не придумывай информацию и не делай предположений, если их нет в источнике.
4. Пиши понятно и лаконично, в деловом, но дружелюбном стиле.
5. Если вопрос не связан с тематикой или выходит за рамки предоставленных данных – скажи, что информации об этом нет.
6. Никогда не вставляй внутренние рассуждения, “think”-блоки или скрытые размышления.

Формат ответа:

- Чёткий, структурированный текст (по пунктам, если нужно).
- Без лишних фраз, шаблонов и рассуждений о процессе поиска.



Для чего нужен Event Processor?



Служба **Event Processor** в Kaspersky MLAD предназначена для:

- Обработки событий, поступающих от объектов мониторинга и от службы Anomaly Detector.
- Выявления паттернов и аномальных последовательностей событий.
- Формирования **эпизодов** — последовательностей событий, ограниченных по времени и/или количеству.
- Распознавания новых и повторяющихся (стабильных) событий и паттернов с использованием механизма **внимания**.
- Обработки инцидентов, зарегистрированных службой Anomaly Detector, как событий (при соответствующей настройке).
- Поддержки **режима сна** для повторной обработки истории событий с целью улучшения качества и структуры выявленных паттернов.
- Обработки только **категориальных данных**, преобразуемых в строковый тип.

Служба работает в двух режимах:

- **Основной режим** — обработка входящего потока событий в виде эпизодов.
- **Режим сна** — повторный анализ ранее обработанных данных за более длительные интервалы времени.

Для корректной работы требуется настройка конфигурационного файла, параметров внимания и отображения параметров событий.



Хорошо, как выполнить эту настройку?



Настройка **Event Processor** в Kaspersky MLAD включает следующие шаги:

1. **Настройка конфигурации внимания:**

Программная реализация и «железо»

25 из 39

- 2x NVIDIA A100 80GB
- vLLM в Docker-контейнерах для сервинга **всех** моделей
 - **баг** при сервинге Qwen3-Embedding-8B: <https://github.com/vllm-project/vllm/issues/25991>
 - из реранкеров поддерживает **только кросс-энкодеры**, для поддержки Qwen3-Reranker-4B **встраивали свой код** в vLLM
 - все модели развернуты в **bf16** (без квантизации)
- Библиотеки Python:
 - BeautifulSoup (основа нашего парсера)
 - LlamaIndex (https://github.com/run-llama/llama_index)
- Собственный демонстрационный UI на базе Streamlit

- Собрали proof-of-concept
- **Пока не пробовали** сложные «агентные» подходы — только успешно кастомизировали базовые этапы RAG
- Конечно, впереди много **улучшений**:
 - учет связей/ссылок между страницами документации
 - обработка неоднозначных, противоречивых, **зловредных** запросов пользователя
 - контролируемое привлечение собственных «знаний» LLM
 - добавление визуальной модальности и др.
- Но чтобы понять, что что-то улучшилось, нужна **оценка качества**

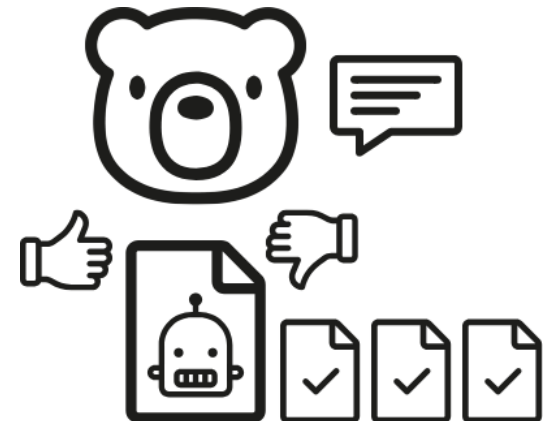
Оценка качества RAG

27 из 39

- Для оценки требуются **вопрос пользователя**, **контекст** (чанки) и выданный системой **ответ**
- Как правило, нужен также **эталонный** (reference) ответ
 - бывает и reference-free оценка: в Ragas ответ системы оценивается за счет **генерации вопросов** к нему и сравнения с вопросом пользователя
 - но отсутствие reference **не дает** убедиться, действительно ли дан **полный** ответ с извлечением максимума пользы из контекста
- Проблема: где взять вопросы и reference?
 - одно из решений — статья Know Your RAG и инструмент Docling SDG

<https://arxiv.org/abs/2411.19710>

<https://github.com/docling-project/docling-sdg>



- Входные данные — отрывки (passages) документации
 - мы ограничиваем размер отрывков **2048** токенами; берем целые страницы либо их наиболее крупные чанки
- К каждому отрывку генерируются **три** вопроса
 - один типа **fact_single** (единичный факт)
 - один типа **summary**
 - один типа **reasoning**
- Для каждой пары «отрывок-вопрос» генерируется **reference**
 - свой системный промпт для каждого типа вопросов
- Для каждой тройки «отрывок-вопрос-reference» выполняется **критика** по 8 критериям
- Мы взяли **Llama-3.3-70B-Instruct** для всех шагов



Генерация вопросов: СИСТЕМНЫЙ ПРОМПТ

29 из 39

I will provide you a text passage. I need you to generate three questions that must be answered only with information contained in this passage, and nothing else.

The first question is of type "**fact_single**", which means that the answer to this question is a **simple, single piece of factual information** contained in the context.

The second question is of type "**summary**", which means that the answer to this question **summarizes different pieces of factual information** contained in the context.

The third question is of type "**reasoning**", which is a question that **requires the reader to think critically and make an inference or draw a conclusion** based on the information provided in the passage.

Make sure that the three questions are different.

...

+ наш перевод на русский язык

Генерация reference: системный промпт

A "reasoning" answer is an answer with the following properties:

- It contains a summary of multiple pieces of information.
- It clearly states an inference or conclusion that can be drawn from a passage.

I will provide you with a context and a question that should be answered with a "reasoning" answer.

This "reasoning" answer should only contain information from the context, and nothing else.

Context: `{context_str}`

Question: `{question_str}`

What is your answer? Do not say anything other than the answer.

Критерии для критики: `stand_alone`

You will be given a question.

Your task is to provide a 'total rating' representing `how context-independent this question is`.

Give your answer on a `scale of 1 to 5`, where `1` means that the question `depends on additional information` to be understood, and `5` means that the question `makes sense by itself`. For instance, if the question refers to a particular setting, like 'in the context' or 'in the document', the rating must be 1.

The questions can contain obscure technical nouns or acronyms and still be a 5: it must simply be clear to an operator with access to documentation what the question is about.

For instance, "What is the name of the checkpoint from which the ViT model is imported?" should receive a 1, since there is an implicit mention of a context, thus the question is not independent from the context.

...

Критерии для критики: оставшиеся семь

32 из 39

q_to_c_groundedness

You will be given a context and a sentence that should be a question. Your task is to provide a 'total rating' scoring how well one can answer the given question unambiguously with the given context.

q_usefulness

You will be given a question. Your task is to provide a 'total rating' representing how useful this question can be to a user with domain knowledge on the subject covered by the document collection.

c_clarity

You will be given a context. Your task is to provide a 'total rating' representing the clarity of the information contained in the context.

a_to_c_groundedness

You will be given a context, and a passage. Your task is to score how well the statements in the passage can be inferred from the context.

c_usefulness

You will be given a context. Your task is to provide a 'total rating' representing how useful this context can be to extract statements for a user with domain knowledge on the subject covered by the context collection.

q_feasibility

You will be given a context and a question. This context is extracted from a collection of passages, and the question will be used to find it. Your task is to provide scoring how well this context can be retrieved based on the specificity and pertinence of the question.

qa_tautology

You will be given a question and passage its answer. Your question is to judge whether this question and answer pair form a tautological exchange.

Пример: Kaspersky MLAD

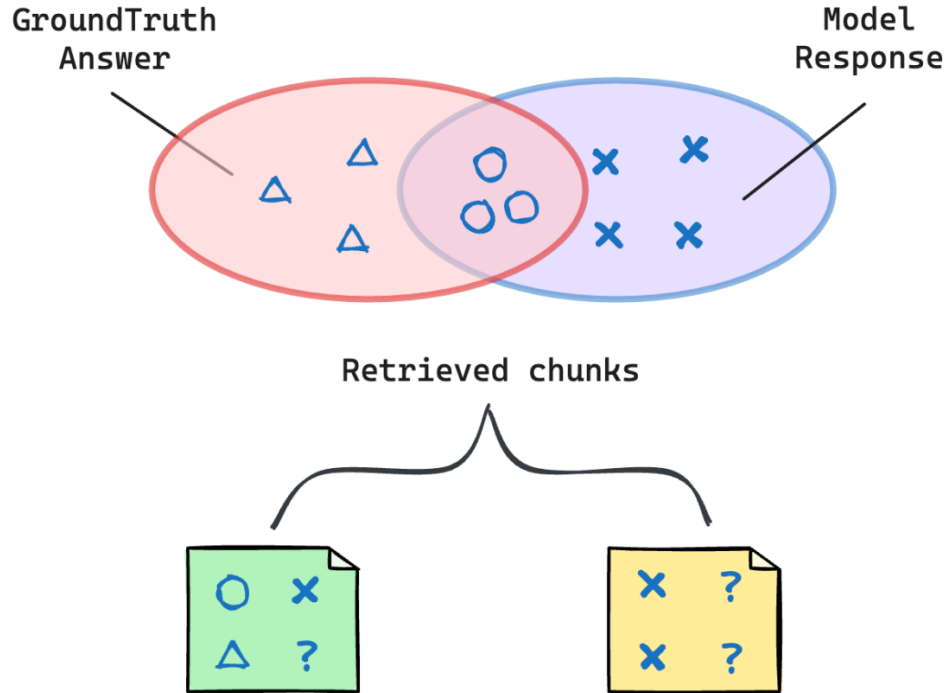
При работе с веб-интерфейсом Kaspersky MLAD в файлах cookie браузера пользователя хранятся следующие данные:

* Индивидуальные токены JSON Web Token для поддержки пользовательской сессии подключения к веб-интерфейсу программы. Индивидуальный токен хранится в файлах cookie браузера пользователя в течение периода неактивности пользователя, заданного при настройке параметров безопасности.

* Идентификатор запущенной сессии Grafana, если пользователь переходил к просмотру журналов программы. Идентификатор сессии Grafana хранится в файлах cookie браузера пользователя в течение 30 дней.

| Тип | Вопрос | Ответ |
|--------------------|---|--|
| fact_single | Что хранится в файлах cookie браузера пользователя при работе с веб-интерфейсом Kaspersky MLAD? | Индивидуальные токены JSON Web Token и идентификатор запущенной сессии Grafana. |
| reasoning | Почему веб-интерфейс Kaspersky MLAD хранит индивидуальные токены JSON Web Token и идентификатор сессии Grafana в файлах cookie браузера пользователя? | Веб-интерфейс Kaspersky MLAD хранит эти данные, чтобы поддерживать пользовательскую сессию и обеспечить доступ к журналам программы, а также для удобства пользователя, поскольку токены и идентификаторы хранятся на определенный период времени, позволяя пользователю не вводить данные заново. |

Метрики RAGChecker (<https://arxiv.org/abs/2408.08067>)



- correct claim
- ✘ incorrect claim
- △ missing claim
- ? irrelevant claim
- ∈ entailment
- |·| count
- relevant chunk
- irrelevant chunk

извлечение утверждений (claims) – троек «**субъект-предикат-объект**»

Overall Metrics

precision: $\frac{|\circ|}{|\circ| + |\times|}$

recall: $\frac{|\circ|}{|\circ| + |\triangle|}$

Retriever Metrics

context precision: $\frac{|\blacksquare|}{|\blacksquare| + |\yellowsquare|}$

claim recall: $\frac{|\circ \in \blacksquare| + |\triangle \in \blacksquare|}{|\circ| + |\triangle|}$

Generator Metrics

context utilization: $\frac{|\circ \in \blacksquare|}{|\circ \in \blacksquare| + |\triangle \in \blacksquare|}$

noise sensitivity: $\frac{|\times \in (\blacksquare \cup \yellowsquare)|}{|\circ| + |\times|}$

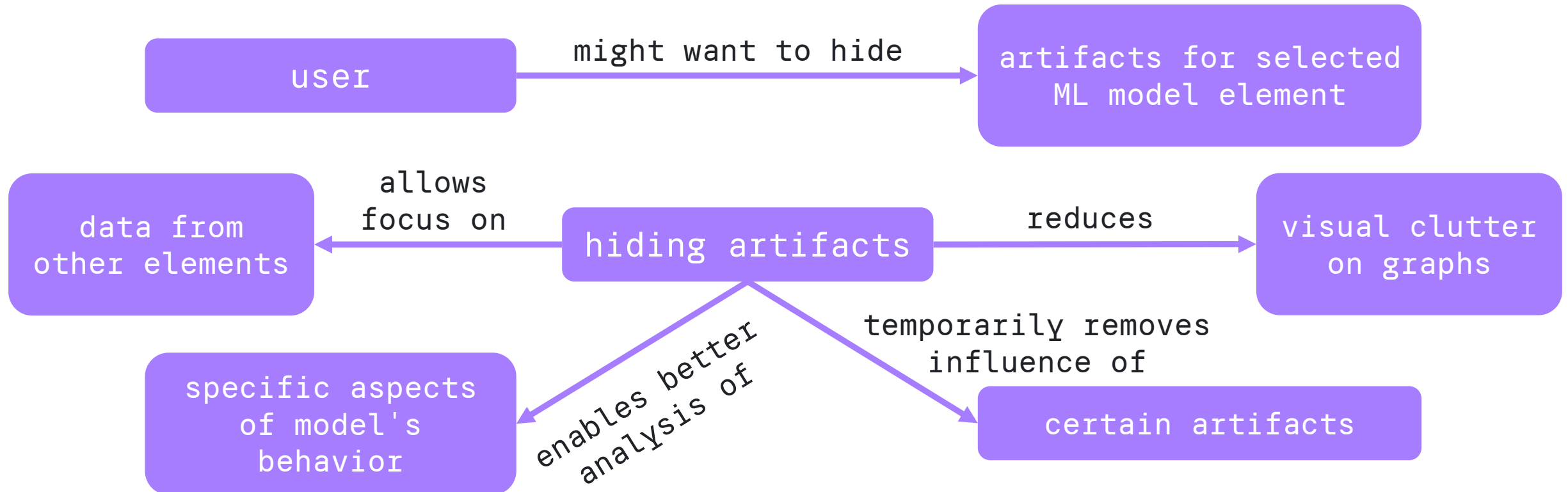
hallucination: $\frac{|\times \notin (\blacksquare \cup \yellowsquare)|}{|\circ| + |\times|}$

self-knowledge: $\frac{|\circ \notin (\blacksquare \cup \yellowsquare)|}{|\circ| + |\times|}$

faithfulness: $\frac{|\circ \in \blacksquare| + |\times \in (\blacksquare \cup \yellowsquare)|}{|\circ| + |\times|}$

Пример извлеченных LLM утверждений

A user might want to hide the artifacts for a selected ML model element to focus on the data or results from other elements, reduce visual clutter on the graphs, or better analyze specific aspects of the model's behavior by temporarily removing the influence of certain artifacts.



Результаты оценки RAGChecker

36 из 39

| Метрика | KICS for Networks | | KICS for Linux Nodes | | KUMA | | MLAD | |
|------------------------------|-------------------|-------|----------------------|-------|-------|-------|-------|-------|
| | ru | en | ru | en | ru | en | ru | en |
| Тестовых примеров | 334 | 371 | 321 | 297 | 301 | 360 | 504 | 789 |
| Precision ↑ | 0.731 | 0.634 | 0.719 | 0.644 | 0.711 | 0.599 | 0.637 | 0.638 |
| Recall ↑ | 0.837 | 0.874 | 0.783 | 0.786 | 0.812 | 0.817 | 0.779 | 0.861 |
| Claim recall ↑ | 0.852 | 0.912 | 0.797 | 0.824 | 0.821 | 0.833 | 0.809 | 0.901 |
| Context precision ↑ | 0.945 | 0.917 | 0.916 | 0.864 | 0.91 | 0.875 | 0.867 | 0.893 |
| Context utilization ↑ | 0.911 | 0.916 | 0.879 | 0.852 | 0.882 | 0.865 | 0.862 | 0.89 |
| Noise sensitivity (rel.) ↓ | 0.205 | 0.271 | 0.19 | 0.237 | 0.198 | 0.287 | 0.273 | 0.257 |
| Noise sensitivity (irrel.) ↓ | 0.009 | 0.019 | 0.014 | 0.042 | 0.031 | 0.043 | 0.033 | 0.038 |
| Hallucination ↓ | 0.054 | 0.076 | 0.074 | 0.076 | 0.056 | 0.071 | 0.057 | 0.066 |
| Self-knowledge ↓ | 0.037 | 0.014 | 0.043 | 0.019 | 0.032 | 0.021 | 0.023 | 0.022 |
| Faithfulness ↑ | 0.909 | 0.91 | 0.883 | 0.905 | 0.912 | 0.908 | 0.92 | 0.912 |

Результаты оценки RAGChecker [2]

37 из 39

- Для извлечения утверждений и проверки логического следования (entailment) мы использовали [Qwen3-30B-A3B-Instruct-2507](#) вместо LLaMA
 - в первую очередь из соображений производительности
 - также в нашем случае утверждения оказались качественнее, чем у LLaMA
- Текущие **ограничения** (limitations) оценки качества:
 - генерация тестовых примеров только по **одной** странице/чанку
 - тестирование reasoning-способностей пока очень наивное
 - пока не генерируем вопросы, на которые **заведомо невозможно** ответить (unanswerable)
 - пока не поддерживаем **multi-turn** взаимодействие с пользователем

- Внедрение LLM в прибыльные системы требует обхода фундаментальных **ограничений** LLM
 - LLM — это вероятностная модель языка, и сама по себе не может выступать базой отраслевого знания
 - Датасеты (и алгоритмы их построения) и метрики, оценивающие качество LLM, не всегда описывают реальное взаимодействие с пользователем
- **RAG — путь к такому внедрению**
 - Для оценки и обеспечения качества и доверенности RAG нужно понимать характер взаимодействия пользователя с системой
 - Характер взаимодействия, несомненно, свой для каждой системы, но всегда есть некоторые паттерны, которые требуют (научного) исследования

Спасибо за внимание 😊