

Как собрать своего AI-ассистента с RAG

и убедиться,
что он хорош



ПРИВЕТ!

Александр Волков



**Точка
банк**

ML Space

Evolution ML Inference

Foundation Models

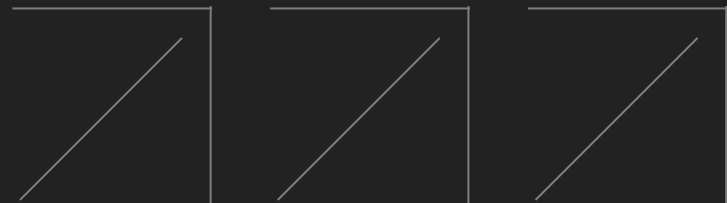
Evolution Managed RAG

Evolution AI Agents

ИИ-Ассистент

и другие

LLM-based-продукты



ПРИВЕТ!

Александр Волков



ML Space
Evolution ML
Foundation Models
Evolution Managed RAG
Evolution AI Agents

Точка
Секс

Evolution ML Inference

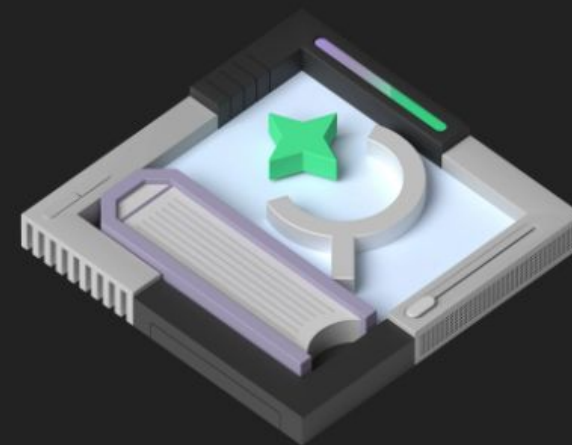
Foundation Models

Evolution AI Agents



Evolution Managed RAG

Управляемые RAG-системы на основе пользовательских данных



[Подключить](#)

[Документация сервиса](#)

[О сервисе](#) [Возможности сервиса](#) [Преимущества](#) [Ответы на вопросы](#) [Консультация](#)




Привет, я Гига-помощник. Ваш ИИ-помощник, который упрощает работу в облаке.

[Подробнее](#)

Описание сервиса


Evolution Managed RAG – облачный сервис для обогащения языковой модели вашими данными. Загружайте файлы в хранилище: сервис самостоятельно предоставит языковой модели актуальную информацию по вашим документам для генерации точных ответов. Сервис подойдет для чат-ботов, поддержки клиентов и других задач.

 Гига-помощник

ИИ-помощник, который упрощает работу в облаке

Новым пользователям — 4 000 ₽ бонусами на облачные ресурсы. ⓘ

Создавайте инфраструктуру с новым членом вашей команды, который знает все про облака.

 [Попробовать бесплатно](#)

Быстрее, проще, эффективнее
с Гига-помощником


[Виртуальная машина](#)

[Мониторинг](#)

[Документация](#)

[Co-pilot для консоли](#)

[Контроль затрат](#)

 [Отдел продаж Cloud.ru](#)



A Александр Г.

Я заказал покрывало, а подскажите, какой у него размер, не могу найти?

Подсказка оператору

A Оператор Мария

Здравствуйте! Все проверила и вижу, что в заказе размер покрывала составляет 220x240 см. Могу ли я еще чем-нибудь помочь?

A Александр Г.

Спасибо! А когда привезете?

Подсказка оператору

Редактировать текст

Ожидайте доставку сегодня 21.03 в интервале 14:30 – 16:00. Курьер свяжется с вами за полчаса.



какие доклады про llm будут в программе heisenbug spring 2026

25 марта 2026 г.

На конференции **Heisenbug Spring 2026** (27–28 апреля, Москва + онлайн) запланирован ряд докладов, посвященных применению больших языковых моделей (LLM) и ИИ в тестировании. Программа продолжает пополняться, но на данный момент подтверждены следующие выступления по теме

AI/LLM: Heisenbug 2026 Spring +2

Ключевые доклады по LLM и ИИ

- **LLM'изация тестирования в Яндексе: измеряем эффект от AI в команде из 1000+ QA-инженеров**
 - **Спикер:** Владислав Миронов (Яндекс).
 - **О чем:** Как измерить реальную пользу от GenAI и отделить «вау-эффект» от реальной ценности. Обсуждение метрик и кейсов: генерация тестовой

Задайте вопрос



11 сайтов

Schedule | Conference on QA & testing

April 27. Hall 1Hall 2Hall 3 00:30. Conversation. Opening of Heisenbug 2026...



Heisenbug 2026 Spring

Расписание | Конференция по тестированию

Heisenbug 2026 Spring | Расписание | Конференция по тестированию...



Heisenbug 2026 Spring

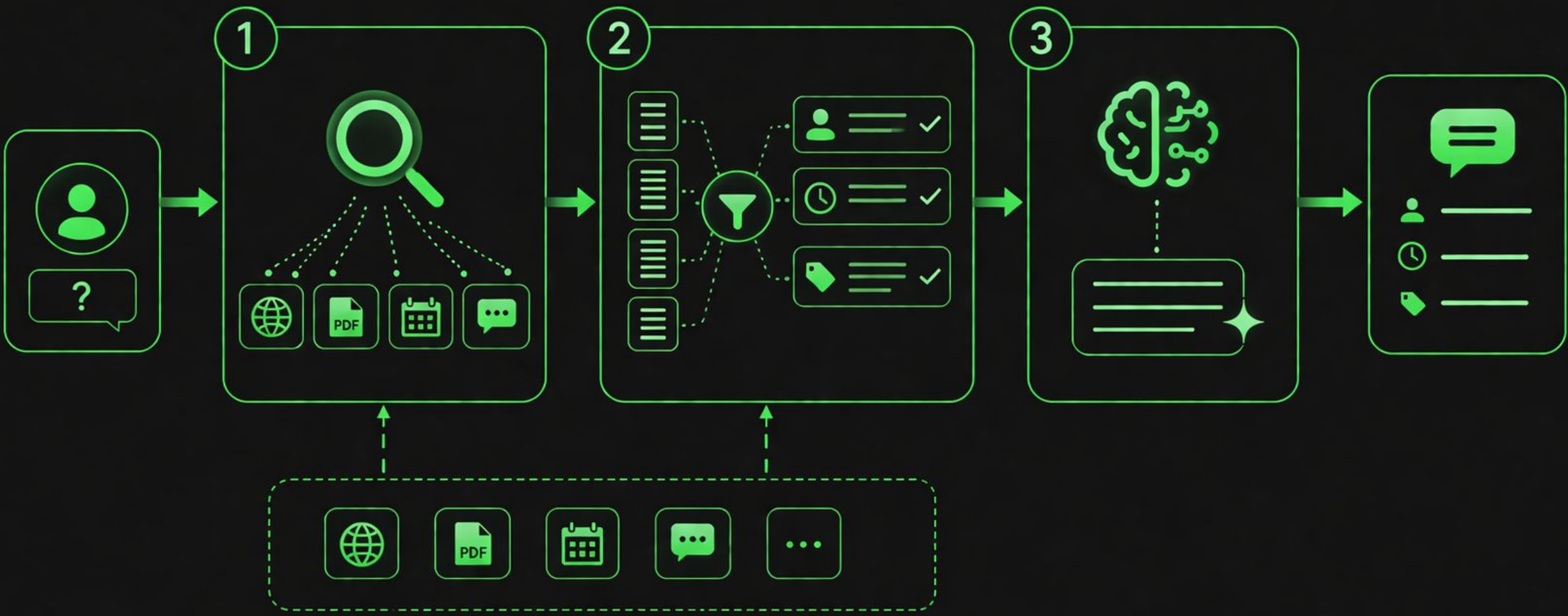
Heisenbug 2026 Spring | IT Conference on QA and Testing

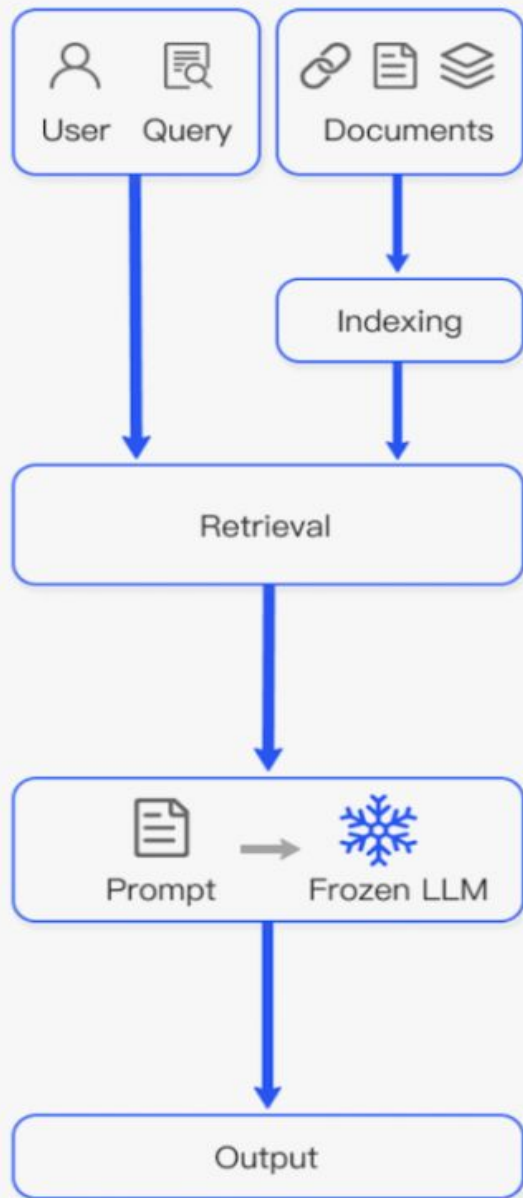
Aleksander Sharkov. Monitor Electric. Hall 1 RU. Keynote 28.04 / 01:00 – 02:00. Testin...



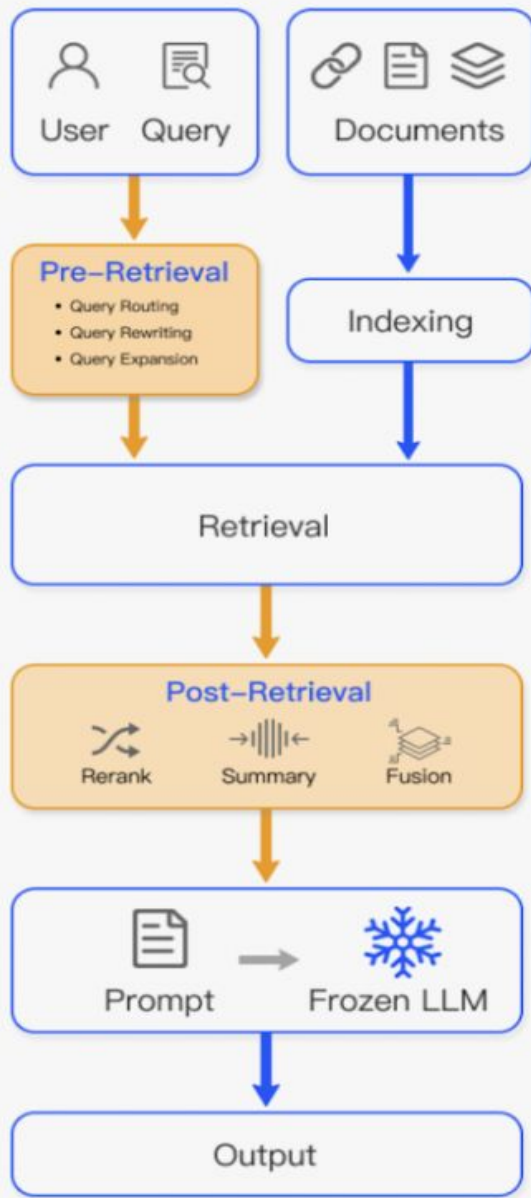
Heisenbug 2026 Spring

Показать все



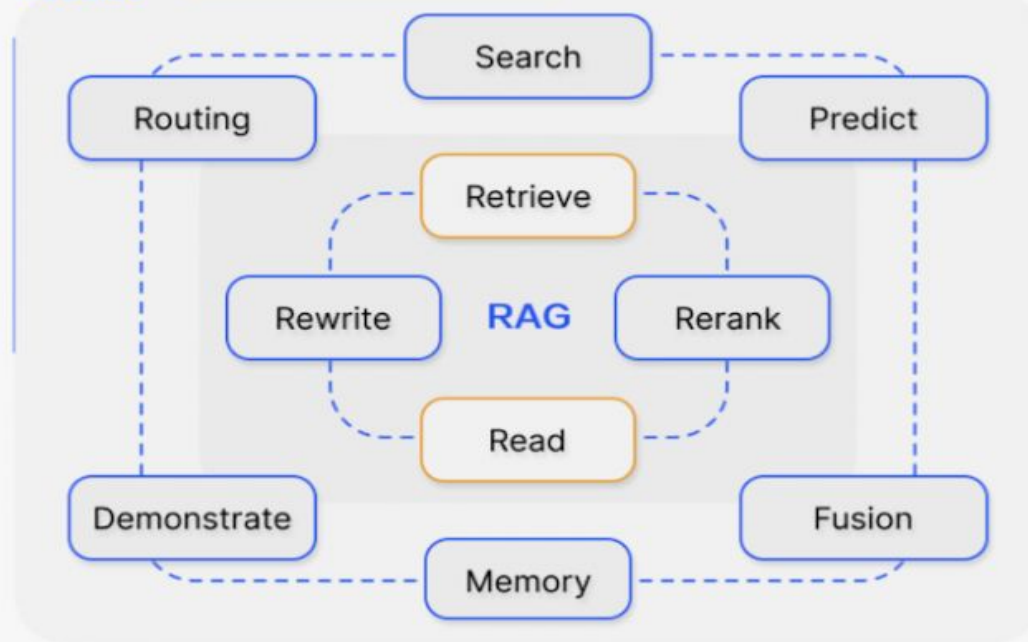


Naive RAG

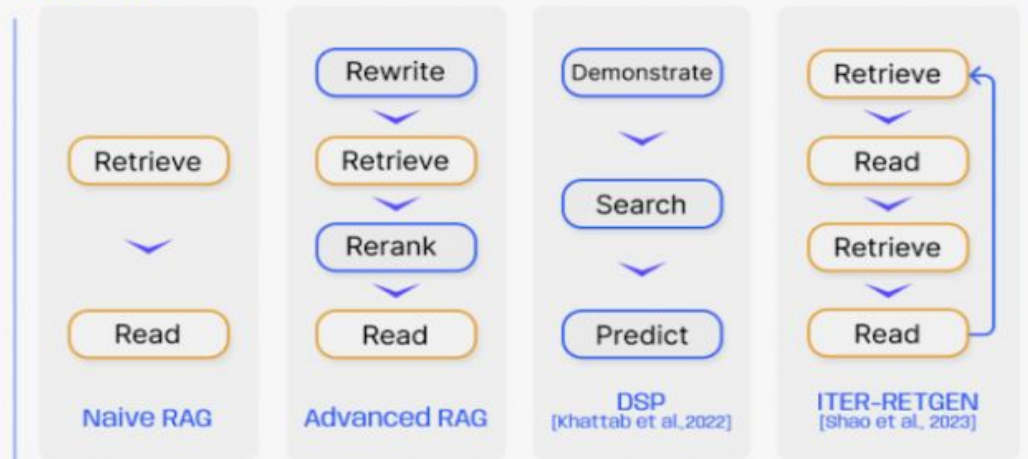


Advanced RAG

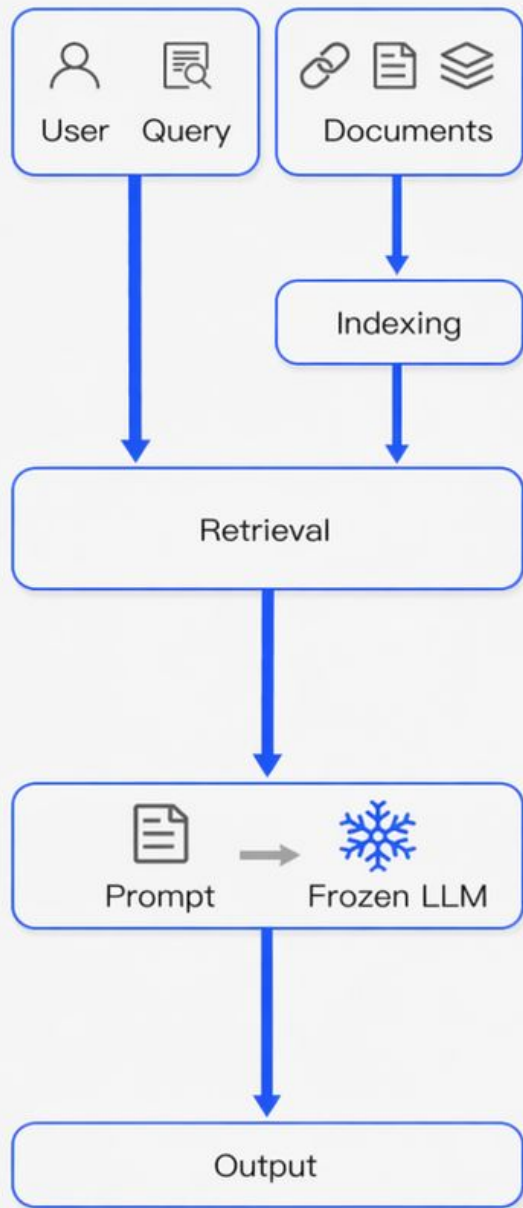
Modules



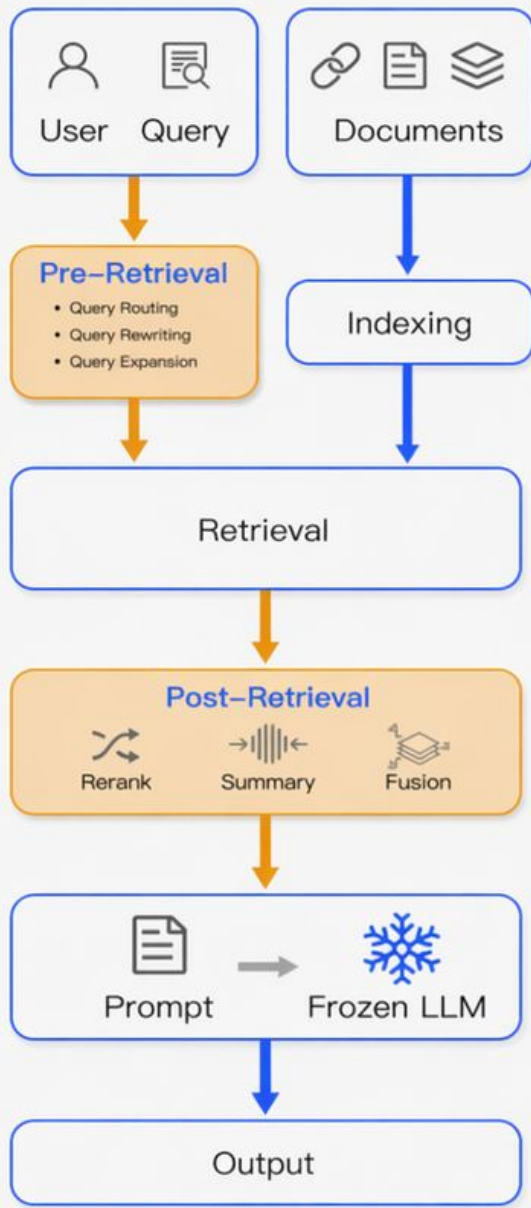
Patterns



Modular RAG



Naive RAG

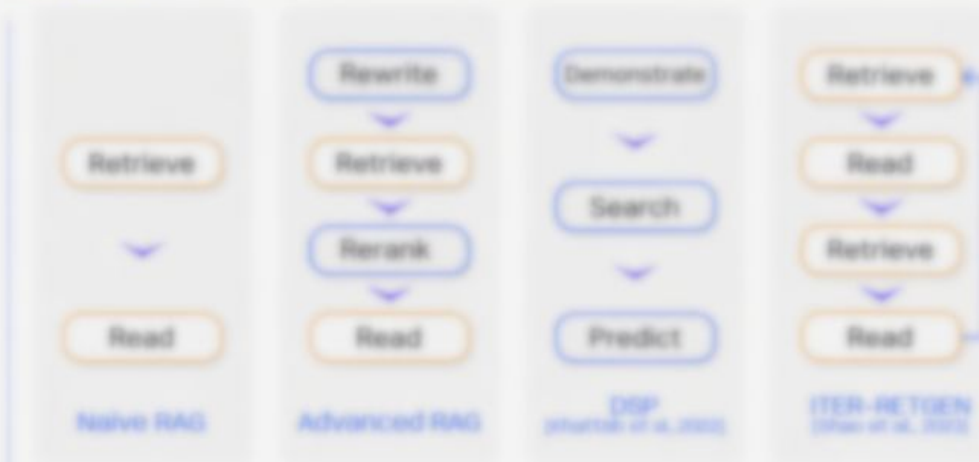


Advanced RAG

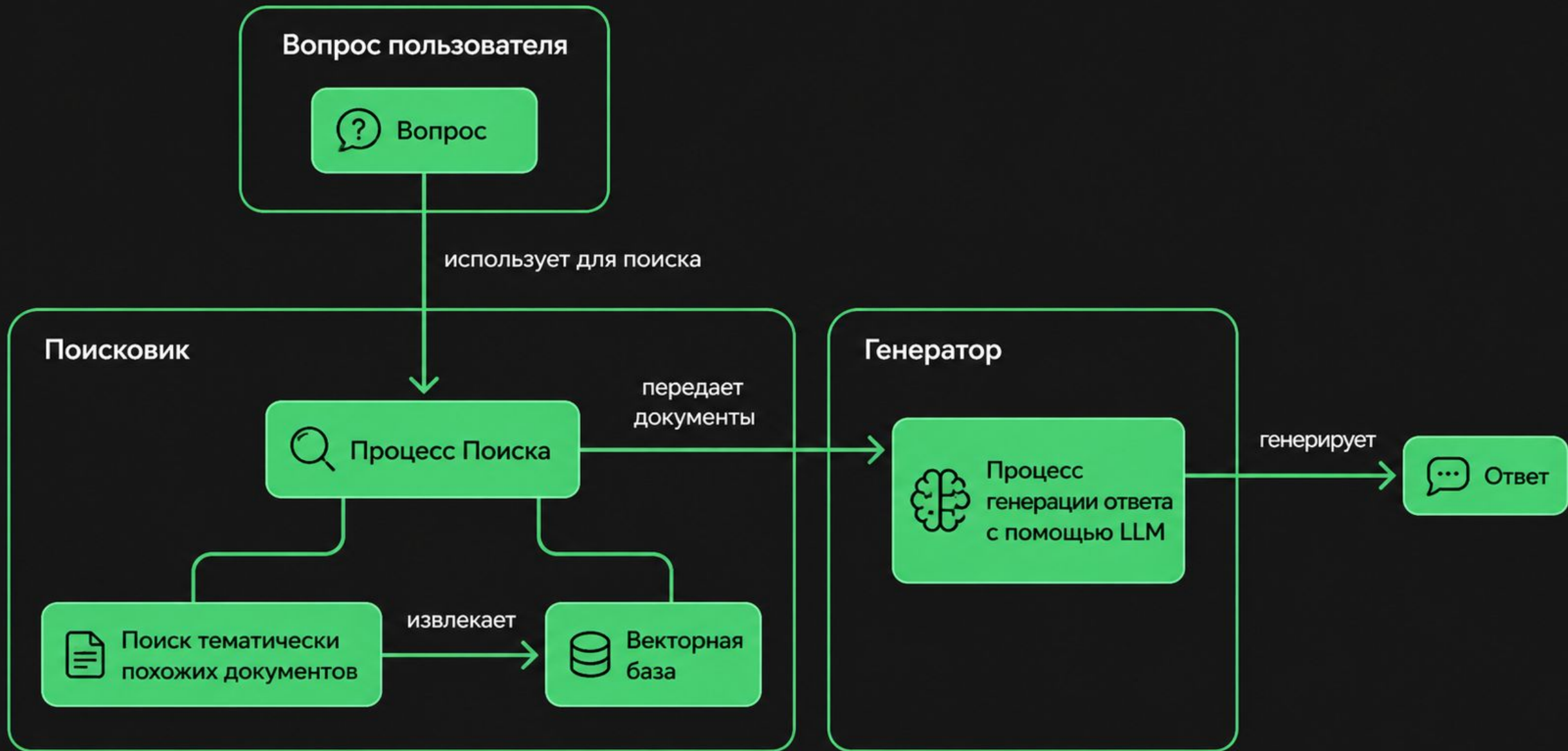
Modules

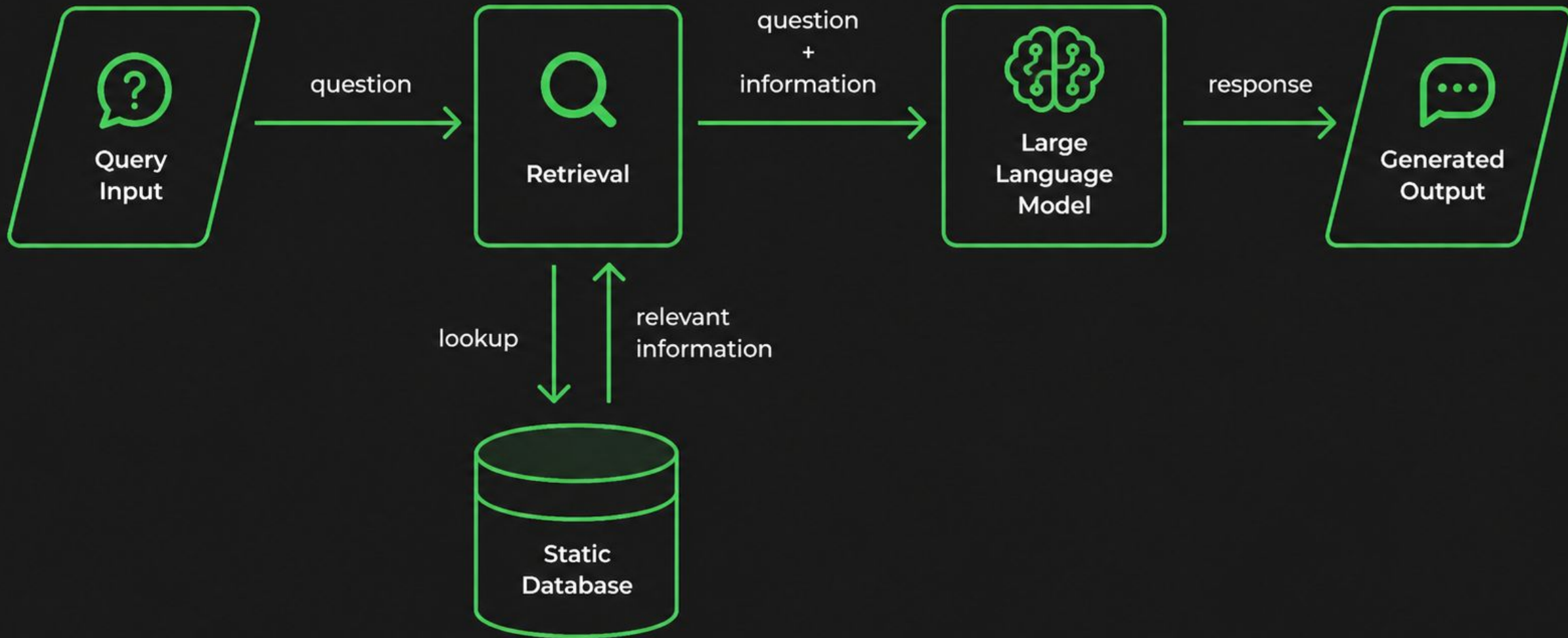


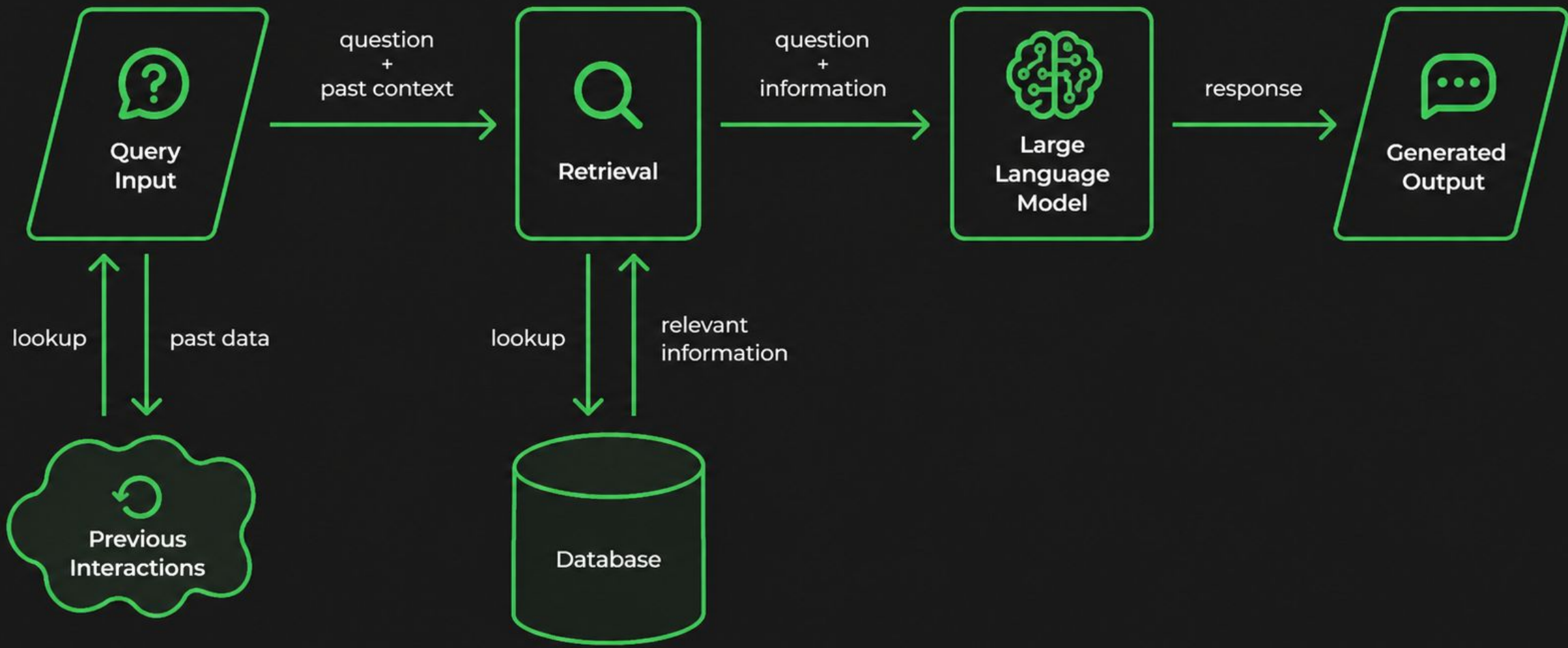
Patterns



Modular RAG

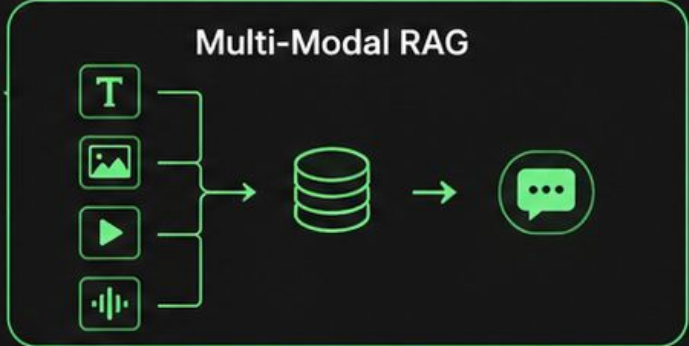
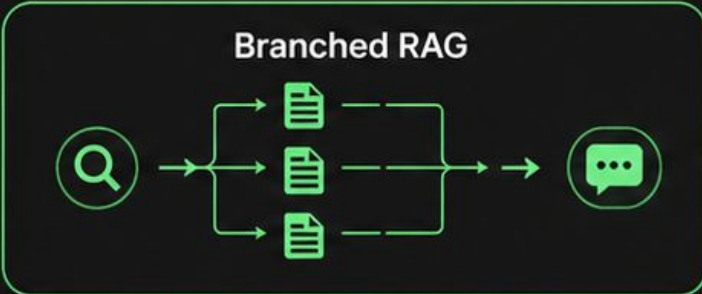








RAG





ДОКЛАД 27.04 / 16:30 – 17:15 (UTC+3)

AI

Tools/Frameworks

Зал 1



Материалы ▾

Оценить выступление 😊



Внедрение ИИ в небольших компаниях

Инженерный подход к внедрению ИИ в тестировании для небольших компаний, где нет лишнего времени и бюджета. Расскажу, как мы сэкономили время тестировщиков, и поделюсь готовым стартовым пакетом: моделями, сценариями и бесплатным инструментом.

Вы узнаете про Roo Code, поймете, как работает API LLM и как строятся агенты с помощью классических языков программирования и MCP.

В докладе будет много маленьких провалов и несколько больших побед.

Спикеры



Александр Шарков
Монитор Электрик

Приглашенные эксперты



Авенир Воронов
Veai



Сборка AI-ассистента
с RAG изнутри

1 часть

Проверка качества

2 часть



Сборка AI-ассистента с RAG изнутри

1 часть



AI-ассистент с RAG изнутри

Зачем RAG?

Пайплайн

Как собрать?





До:

“что-то слышал/а про РАК”

-

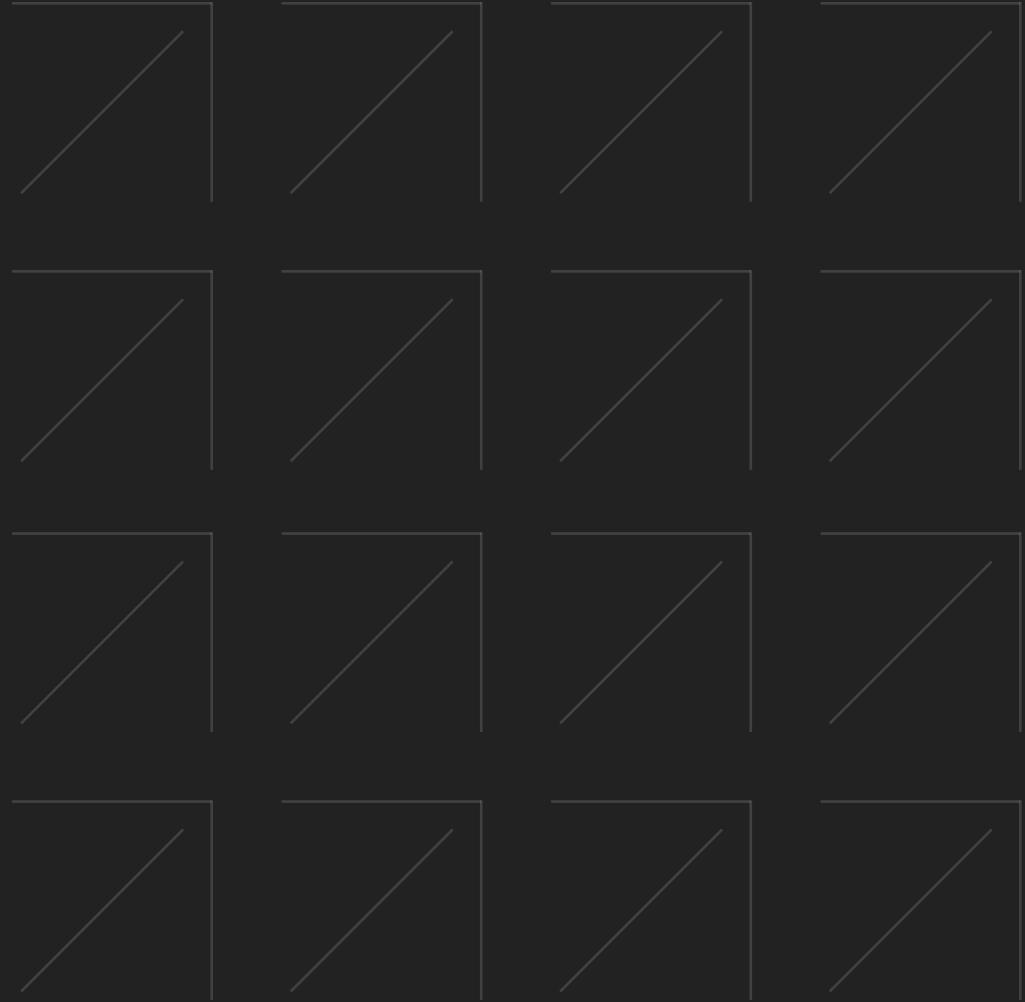
После:

“классно, я теперь знаю”

+

как устроен пайплайн
как собрать свой

+



До:

“что-то слышал/а про РАК”

—

После:

“классно, я теперь знаю”

+

как устроен пайплайн
как собрать свой

+

How to draw an owl

1.



2.

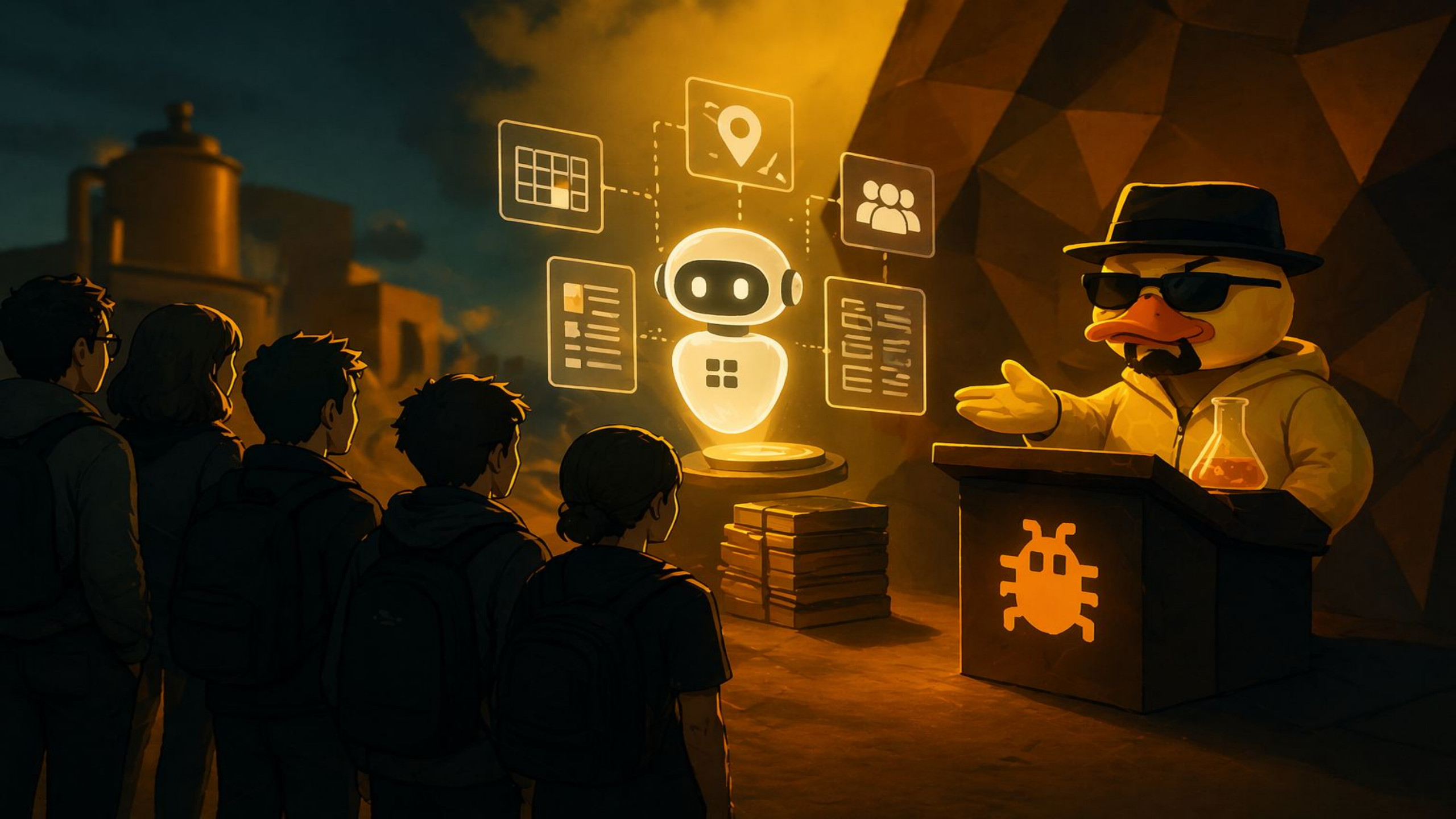


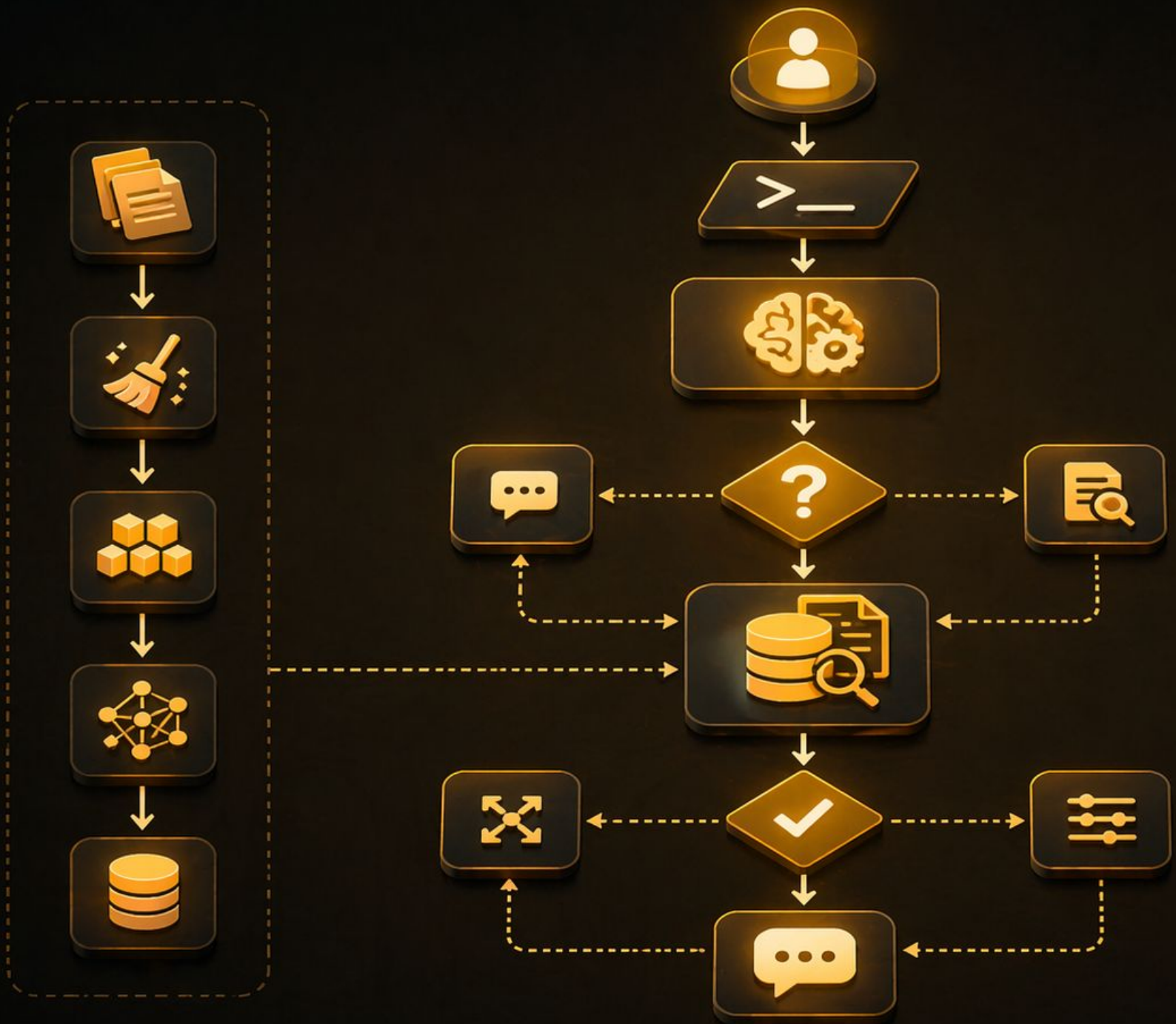
1. Draw some circles

2. Draw the rest of the fucking owl

Стартуем!









Ограничения LLM



Знания -> только до момента обучения





Ограничения LLM



Знания -> только до момента обучения

Нет доступа к актуальному контексту





Ограничения LLM



Знания -> только до момента обучения

Нет доступа к актуальному контексту

Ограниченное окно контекста





Ограничения LLM

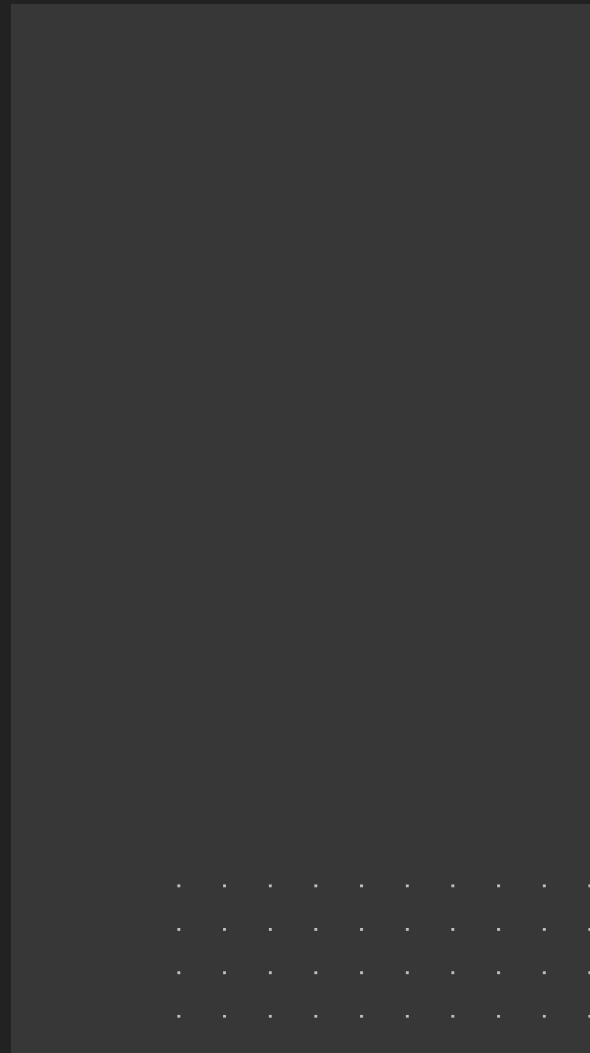


Знания -> только до момента обучения

Нет доступа к актуальному контексту

Ограниченное окно контекста

Дорогое дообучение





Ограничения LLM



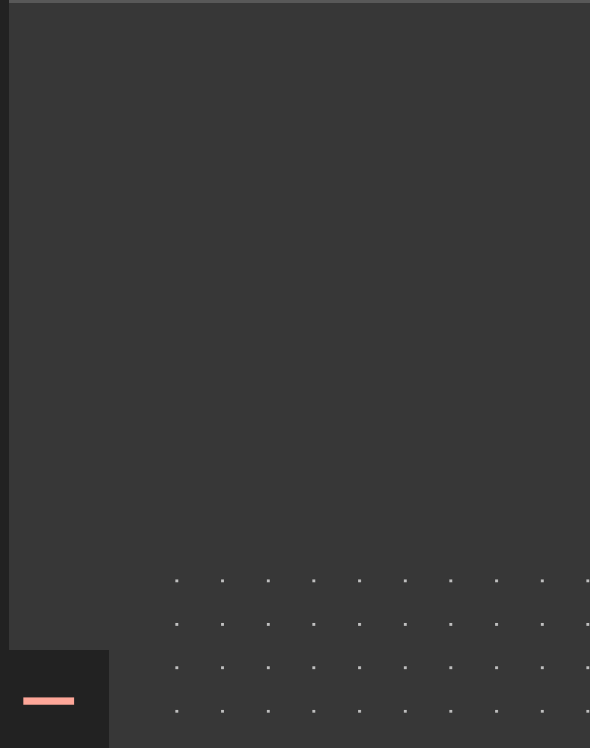
Знания -> только до момента обучения

Нет доступа к актуальному контексту

Ограниченное окно контекста

Дорогое дообучение

Нет встроенной "проверки правды"

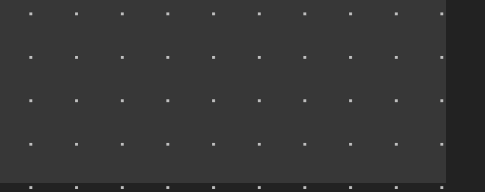




Как это проявляется?



Галлюцинации на доменных знаниях



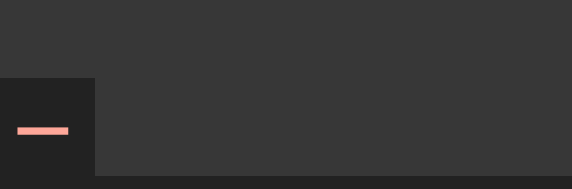


Как это проявляется?



Галлюцинации на доменных знаниях

Уверенные, но поверхностные ответы





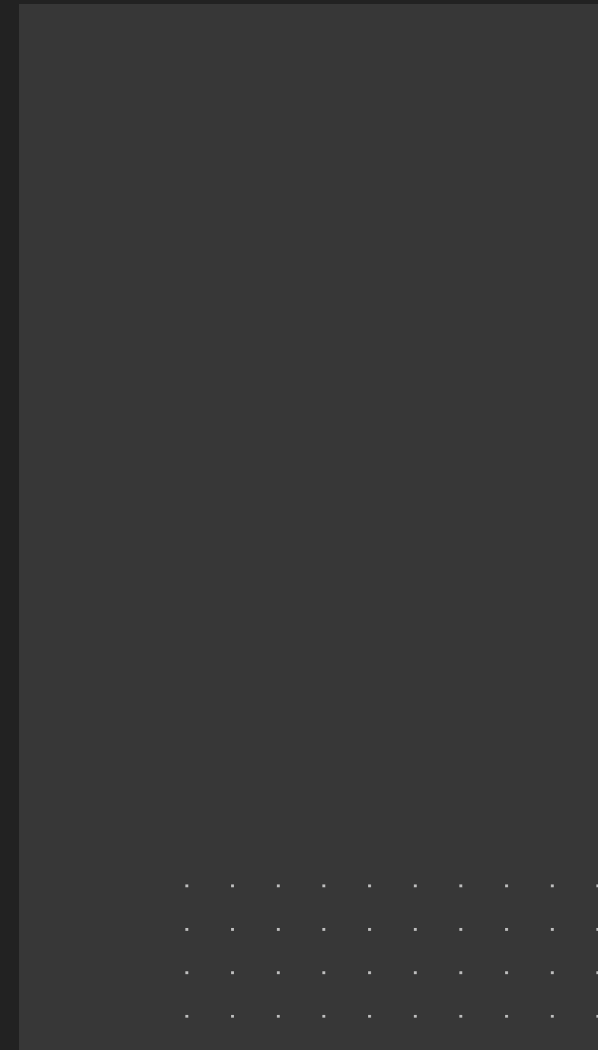
Как это проявляется?



Галлюцинации на доменных знаниях

Потеря контекста в длинных задачах

Уверенные, но поверхностные ответы





Как это проявляется?

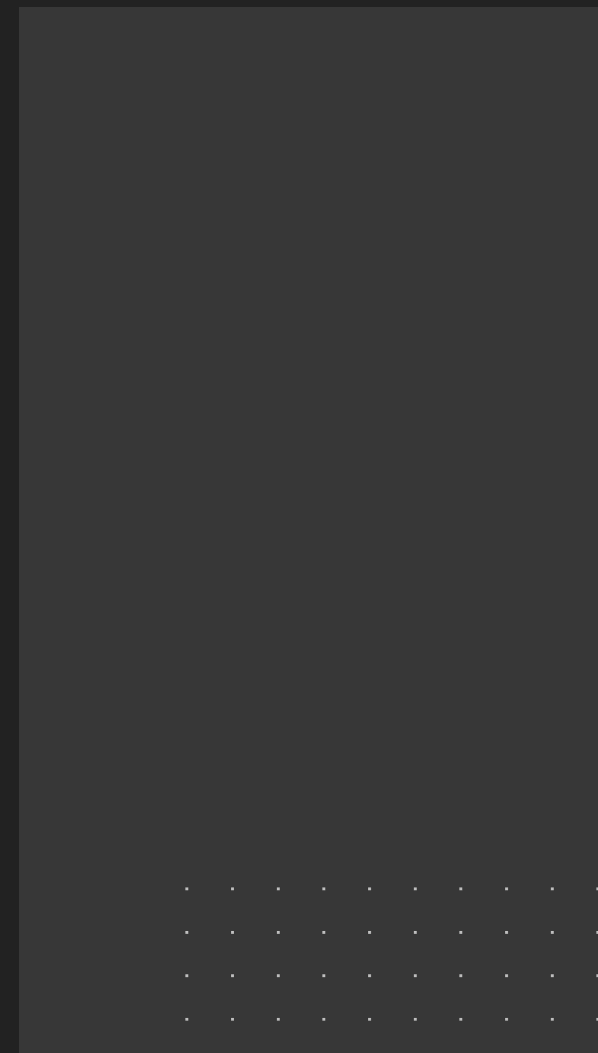


Галлюцинации на доменных знаниях

Потеря контекста в длинных задачах

Уверенные, но поверхностные ответы

Нерелевантные или общие ответы





**Наивный
подход -
отдать все на
откуп LLM**



Знания -> только до
момента обучения

Ограниченное окно
контекста

Нет встроенной
“проверки правды”

Нет доступа к
актуальному контексту

Дорогое дообучение



Что такое RAG?



Что такое RAG?

Retrieval Augmented Generation

Генерация, дополненная поиском



Что такое RAG?

Retrieval Augmented Generation

Генерация, дополненная поиском

Retrieve

Augmentation

Generation



Что такое RAG?

Retrieval Augmented Generation

Генерация, дополненная поиском

Retrieve – найти релевантные документы

Augmentation

Generation



Что такое RAG?

Retrieval Augmented Generation

Генерация, дополненная поиском

Retrieve – найти релевантные документы

Augmentation – дополнить запрос контекстом

Generation



Что такое RAG?

Retrieval Augmented Generation

Генерация, дополненная поиском

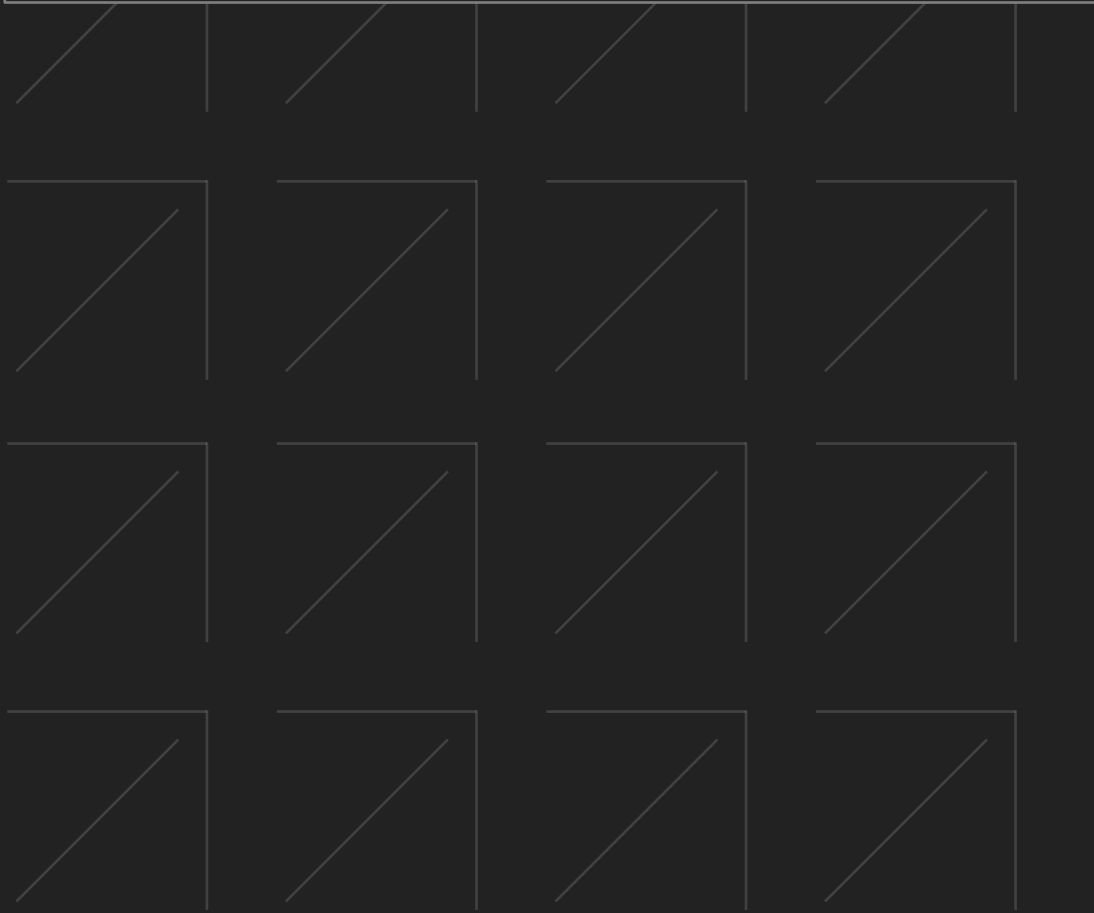
Retrieve – найти релевантные документы

Augmentation – дополнить запрос контекстом

Generation – сгенерировать ответ



Где полезно?



внутренние базы знаний




ассистенты по домену



документация и поддержка

Managed RAG

Документация [Создать базу знаний](#)**Создайте базу знаний**

Вы сможете интегрировать ее для решения ваших задач информационного поиска, отправляя HTTP-запросы.

Подробнее — в [документации](#).

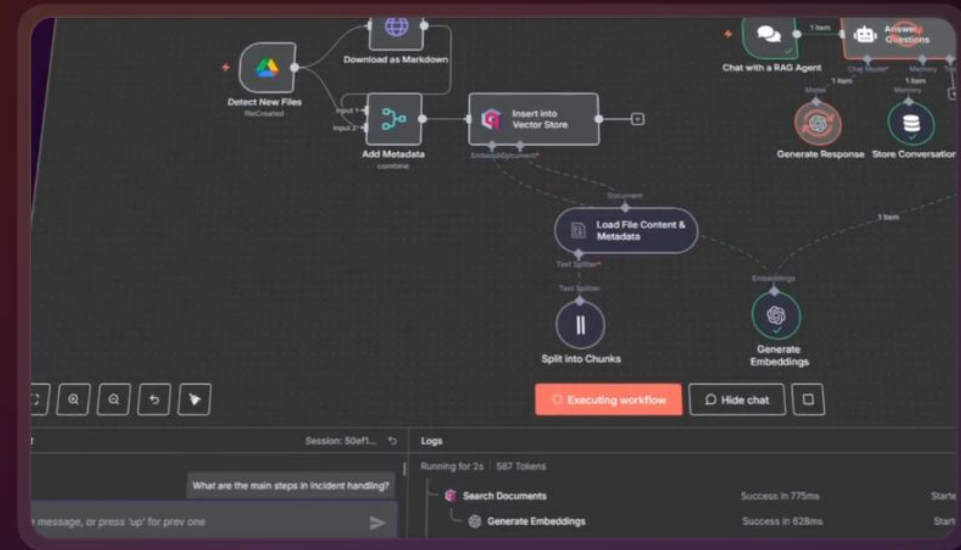
RAG in n8n

Take the drag out of building with RAG

Build complete RAG systems without manually stitching together different tools. n8n's visual builder and 500+ integrations let you handle everything — from ingestion, chunking, and embeddings to retrieval and memory — all in one place.

[Start building RAG systems*](#)[Talk to sales](#)

*14-day free trial. No credit card needed



⚡ Over 150k Github stars

⚡ Self host-able

⚡ SOC2 compliant

Laplace Transform Fundamentals and System Analysis

Analytics Share Settings PRO

Sources

+ Add Discover

Select all sources ✓

PDF Laplace Transform.p... ✓

Chat Refresh

The source material provides an extensive overview of the Laplace Transform (LT) and its applications in signal processing and system analysis. It first introduces key concepts in continuous and discrete-time signal analysis, such as the Fourier Series (FS), Fourier Transform (FT), and related transforms like the Z-Transform and Discrete-Time Fourier Transform (DTFT). A major focus is placed on the bilateral and unilateral LT, detailing its definition, properties (linearity, time shifting, time scaling), and use in solving differential equations and determining the transfer function $H(s)$ of a system. The text also covers the Region of Convergence (ROC), the relationship between the LT and F

Start typing... 1 source

What is the Laplace transform? What is the Fourier transform? What is $x(t)$?

Studio > Mindmap

Laplace Transform for System Analysis

Based on 1 source

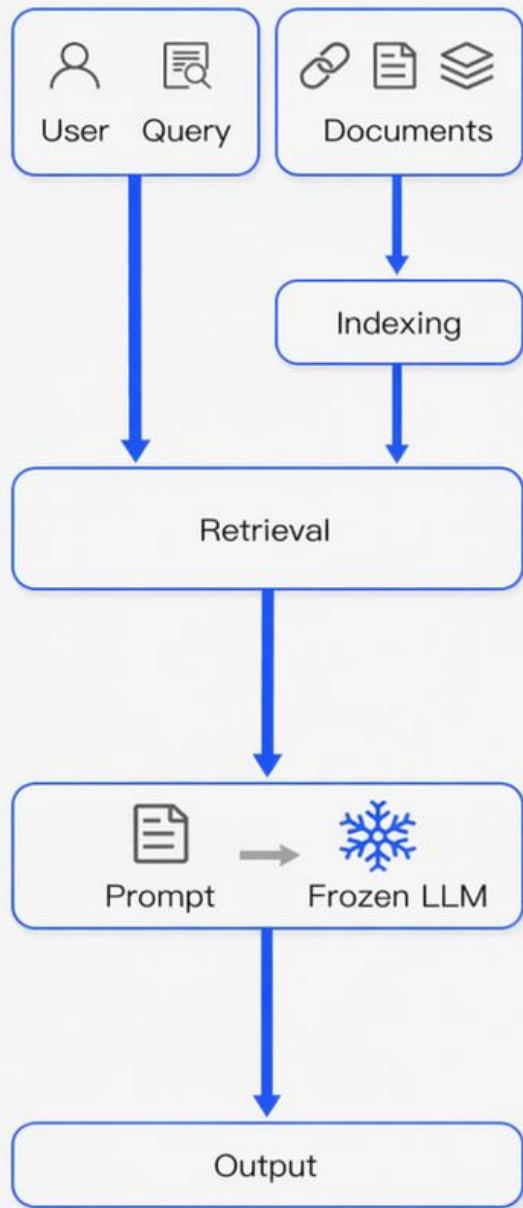
- Fast Fourier Transform (FFT)
- Sampling
- Linearity
- Time Shifting
- Time Scaling
- Differentiation in Time
- Multiplication by t
- Multiplication by e^{st} (Frequency Shifting)
- Convolution
- Properties of Laplace Transform
- Common LT Pairs and Functions
 - $\delta(t) \leftrightarrow 1$
 - $u(t) \leftrightarrow 1/s$
 - $t u(t) \leftrightarrow 1/s^2$
 - $e^{-at} u(t) \leftrightarrow 1/(s+a)$
 - $e^{-at} u(t) \leftrightarrow 1/(s+a)$
 - $e^{-at} u(t) \leftrightarrow 1/(s+a)$
 - $\cos(\omega t) u(t) \leftrightarrow s / (s^2 + \omega^2)$
 - $\sin(\omega t) u(t) \leftrightarrow \omega / (s^2 + \omega^2)$
- Transfer Function Definition
- Poles and Zeros
- Frequency Response Magnitude $M(\omega)$
- System Analysis and Transfer Function $H(s)$

Good content Bad content

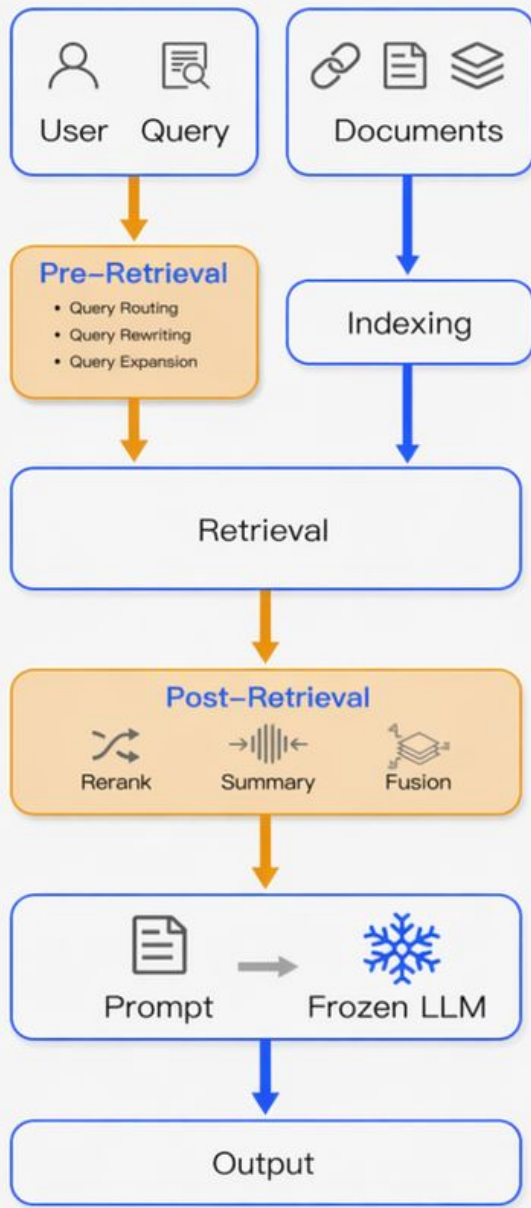
NotebookLM can be inaccurate; please double check its responses



Так, а что под капотом?



Naive RAG

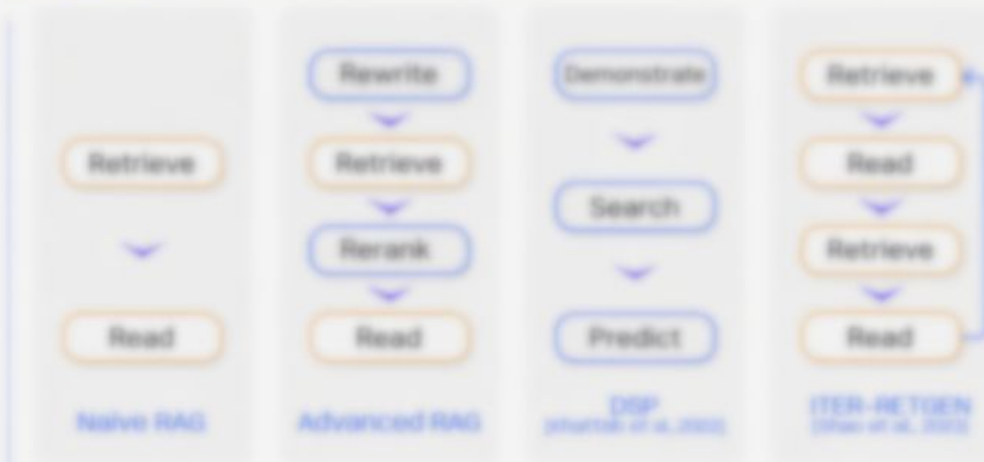


Advanced RAG

Modules



Patterns



Modular RAG

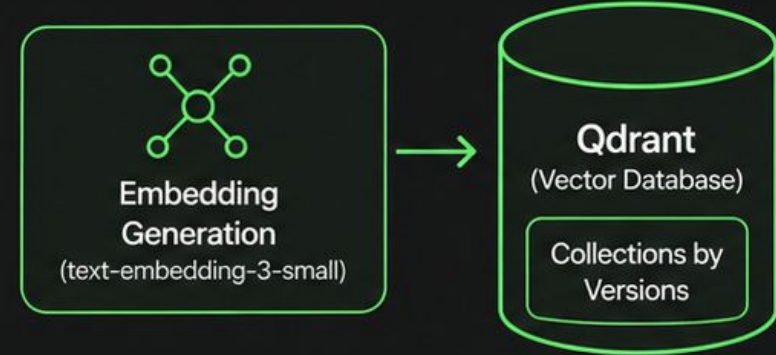
Stage 1: Data Preparation

(Start with original data sources)



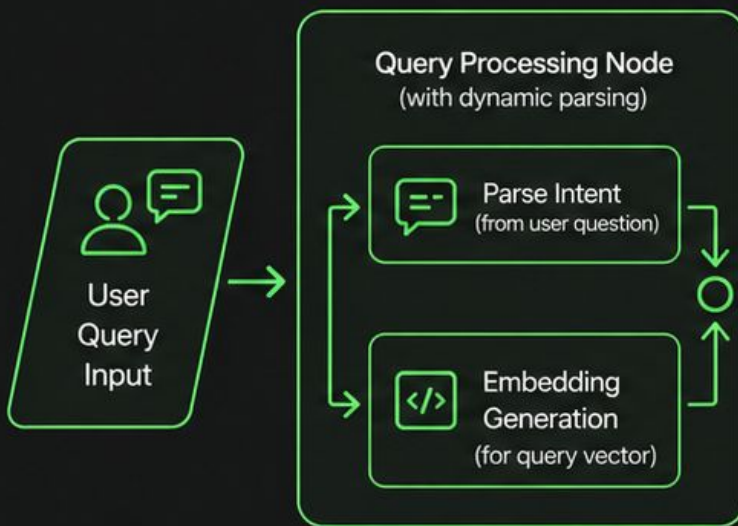
Stage 2: Embedding & Storage

(Focus on vectors)



Stage 3: Inquiry & Retrieval

(Start of the query path)



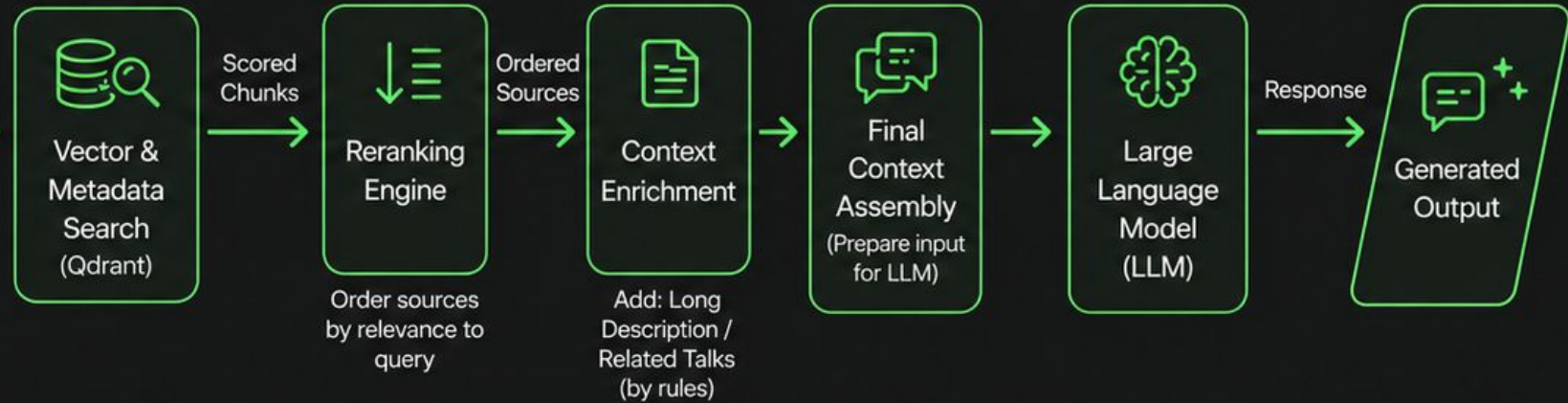
Search Request top-k + metadata filters

Search Request (top-k + filters)

Retrieved Chunks

Stage 4: Post-Processing & Context Synthesis

Stage 5: Generation & Output



Context + System Prompt + Query





ГОТОВИМ ДАННЫЕ



Stage 1: Data Preparation

(Start with original data sources)





27 апреля

11:45

12:15

Доклад

**LLM'изация тестирования в Яндексе: измеряем эффект от AI в команде из 1000+ QA-инженеров**

Как измерить реальный эффект GenAI в тестировании на масштабе более чем в тысячу QA-инженеров и отделить пользу от вау-эффекта? Покажем систему метрик и результаты внедрения: LLM для генерации тестовой документации, ускорение E2E-автотестов и AI-агенты для регрессов.

**Владислав Миронов**
Яндекс

AI

Best Practices

Доклад

**Как не превратить автотесты в объект ненависти: ред флаги в инфраструктуре**

CI может выглядеть стабильным, но это не всегда означает, что он работает эффективно.

Разберем, как по метрикам, мониторингам и структуре пайплайна понять реальное состояние инфраструктуры, где искать узкие места и почему многие проблемы скрываются не в тестах, а в конфигурации среды.

На примерах из практики покажем, какие решения действительно помогают ускорять прогоны и почему советы генеративного ИИ стоит проверять инженерным опытом.

**Иван Левиков****Даниил Смирнов**
Ozon Банк

Mobile

Best Practices

Automation

Доклад

**Системные баги под нагрузкой, или Как мы пытались найти баг в ядре ОС**

Расскажем, какие проблемы появлялись, как мы учились воспроизводить дефект, как наступали на детские грабли и чем это все закончилось (или не закончилось).

**Михаил Жилин**
Postgres Professional

Load Testing

Воркшоп









**Чем аналитик поможет тестировщику, а тестировщик — аналитику**

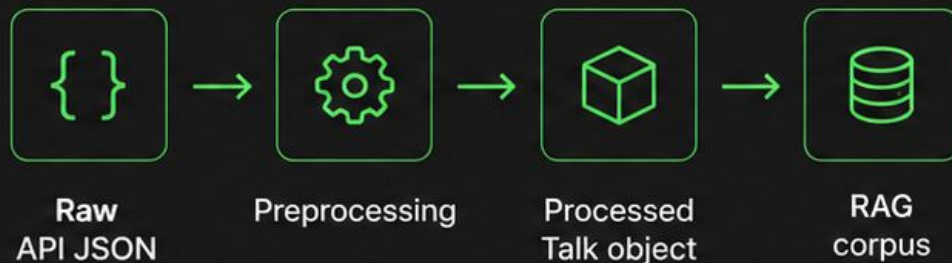
Ошибки встречаются на всех этапах разработки ПО. Чем раньше обнаружим, тем дешевле их исправить. На воркшопе тестировщики научатся выявлять ошибки в требованиях до этапа разработки. Тем самым сократят количество потенциальных ошибок, время и силы команды на их исправление.

12:15

<https://heisenbug.ru/talks/48ab465c0a7844f5b79551b6bb015816/>

Heisenbug talks = карточки докладов

-  title
-  short_description
-  long_description
-  speakers
-  topics
-  day / time / hall
-  talk_type
-  related_talks



Raw JSON

```
{
  "talk": {
    "id": "...",
    "name": {"ru": "..."},
    "shortDescription": {"ru": "<p>...</p>"},
    "longDescription": {"ru": "<p>...</p>"},
    "speakers": [...],
    "talkDay": 1,
    "talkStartTime": "...",
    "talkEndTime": "...",
    "topics": [{"AI"}, {"Best Practices"}],
    "type": "talk",
    "hall": {"ru": "Зал 1"}
  },
  "relatedTalks": [...],
  "_meta": {...}}
}
```

Processed JSON

```
{
  "id": "...",
  "title": "...",
  "short_description": "...",
  "long_description": "...",
  "long_description_useful": true,
  "speakers": [...],
  "topics": ["AI", "Best Practices"],
  "related_talks": [...],
  "day": 1,
  "date": "2026-04-27",
  "time_start": "09:15",
  "time_end": "10:00",
  "hall": "Зал 1",
  "talk_type": "talk"
}
```

Cleaning и de-noising



HTML -> plain text



нормализация
speaker fields



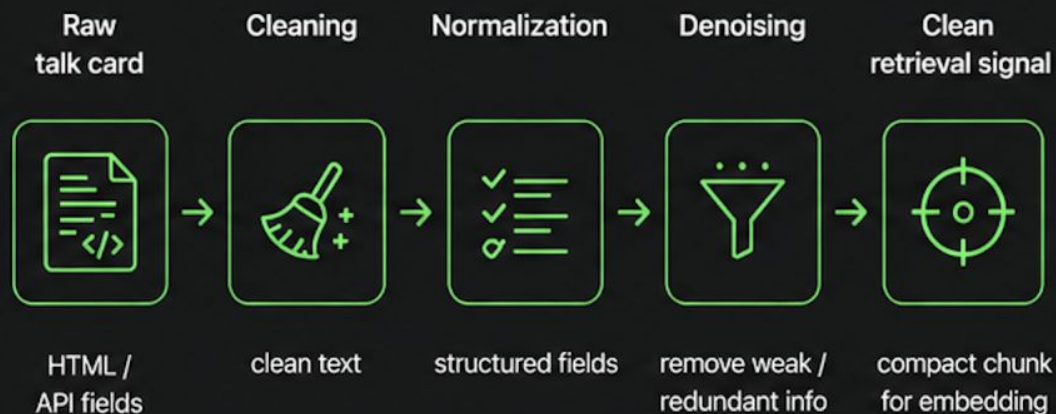
ISO time ->
date / time_start / time_end



related talks filtering



dedup short vs
long description



html_cleaner.py

- clean_html()
- remove HTML tags
- normalize spaces
- plain text output



preprocessor.py

- speakers: name / position / company / bio
- parse date and time
- normalized Talk object



related_filter.py

- topic overlap
- filter weak related talks
- rank by similarity



dedup_checker.py

- compare shortDescription vs longDescription
- keep long_description in data
- do not put it into embedding chunk by default



long_description сохраняется, но не идёт в embedding chunk

добавляется позже только при enrichment в v3 / v4

Cleaning и de-noising



HTML -> plain text



нормализация
speaker fields



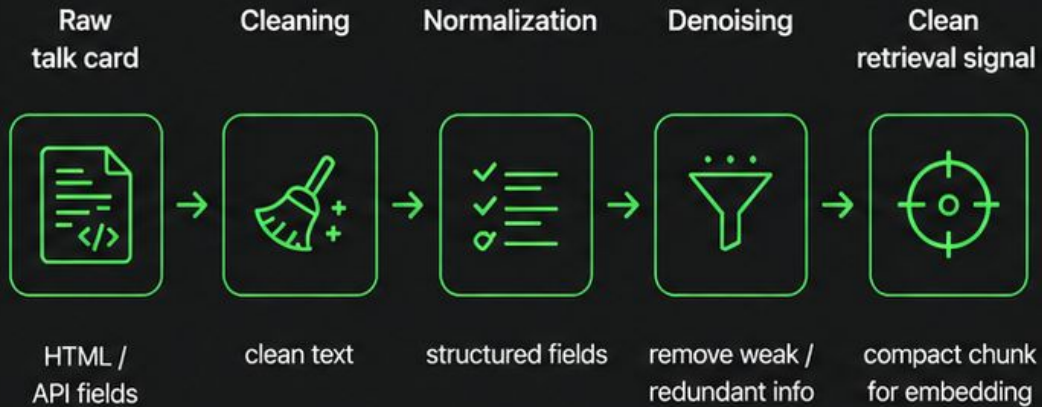
ISO time ->
date / time_start / time_end



related talks filtering



dedup short vs
long description



Качество RAG часто растёт
не от добавления текста,
а от **удаления лишнего текста**



html_cleaner.py

- clean_html()
- remove HTML tags
- normalize spaces
- plain text output



preprocessor.py

- speakers: name / position / company / bio
- parse date and time
- normalized Talk object



related_filter.py

- topic overlap
- filter weak related talks
- rank by similarity



dedup_checker.py

- compare shortDescription vs longDescription
- keep long_description in data
- do not put it into embedding chunk by default



long_description сохраняется, но не идёт в embedding chunk

добавляется позже только при enrichment в v3 / v4

Токены



что такое токенизация?

Посоветуй доклад
про тестирование LLM



Посоветуй доклад
про тестирование LLM



Токенов: 15
Символов: 37

Посоветуй доклад про тестирование LLM

GPT-4

Посоветуй доклад
про тестирование LLM



Посоветуй доклад про тестирование LLM

Токенов: 10
Символов: 37

GPT-5.X

Recommend a talk
on LLM testing



Токенов: 7
Символов: 31



Recommend a talk on LLM testing

GPT-5.X

Посоветуй доклад
про тестирование LLM



Посоветуй доклад про тестирование LLM

```
[54254, 182857, 36012, 18607, 12907,  
4964, 52523, 54616, 451, 19641]
```

Токенов: 10
Символов: 37

GPT-5.X

Чанки



MAXCHUNKTOKENS = 700

Чанк режется по токенам

- tiktoken
- gpt-4o-mini

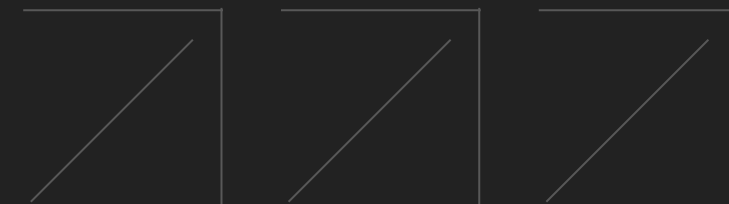
Чанки



Фиксированный размер

По смысловым границам

С перекрытием



Плохой сценарий



Хороший сценарий



1. Source data

Heisenbug talks как структурированные карточки докладов



- title
- short_description
- long_description
- speakers
- topics
- day / time / hall
- related_talks
- talk_type

```
{
  "talk": {
    "id": "...",
    "name": {"ru": "..."},
    "shortDescription": {"ru": "<p>...</p>"},
    "longDescription": {"ru": "<p>...</p>"},
    "speakers": [...],
    "talkDay": 1,
    "talkStartTime": "...",
    "talkEndTime": "...",
    "topics": [{"AI"}, {"Best Practices"}],
    "type": "talk",
    "hall": {"ru": "Зал 1"}
  },
  "relatedTalks": [...],
  "_meta": {...}
}
```

Источник: `parse_talks.py` -> `talks_raw.json`

2. Cleaning / De-noising

Точечная очистка и нормализация без сложного ML



- HTML -> plain text
- normalize speakers
- parse ISO timestamps
- filter related talks
- check long vs short redundancy

html_cleaner.py
clean_html()



preprocessor.py
normalize Talk



related_filter.py
topic similarity



dedup_checker.py
long_description_useful



De-noising здесь = убрать HTML-шум, слабые связи и лишнюю избыточность

raw API fields
(грязные данные)



clean text
(без HTML)



normalized Talk
object

3. Clean retrieval signal

Готовим компактный корпус для следующего этапа



```
{
  "id": "...",
  "title": "...",
  "short_description": "...",
  "long_description": "...",
  "long_description_useful": true,
  "speakers": [...],
  "topics": ["AI", "Best Practices"],
  "related_talks": [...],
  "day": 1,
  "date": "2026-04-27",
  "time_start": "09:15",
  "time_end": "10:00",
  "hall": "Зал 1",
  "talk_type": "talk"
}
```



long_description сохраняется, но **не идёт** в embedding chunk по умолчанию

Результат: `talks_processed.json`

1. Source → Documents

Карточки докладов Heisenbug превращаются в LangChain Document



Карточки
Heisenbug talks



LangChain
Document

```
from langchain_core.documents import Document

docs = [
    Document(
        page_content=talk["short_description"],
        metadata={
            "id": talk["id"],
            "title": talk["name"]["ru"],
            "speakers": talk["speakers"],
            "topics": [t[0] for t in talk["topics"]],
            "day": talk["talkDay"],
            "hall": talk["hall"]["ru"],
        }
    )
    for talk in talks
]
```

2. Cleaning / De-noising

Кастомный preprocessing перед embedding



HTML → plain text



normalize speakers



parse date / time



filter weak related talks



long_description
не идёт в embedding chunk

```
def clean_talk(talk):
    text = clean_html(talk["shortDescription"]["ru"])
    return Document(
        page_content=text,
        metadata={
            "id": talk["id"],
            "title": talk["name"]["ru"],
            "speakers": normalize_speakers(talk["speakers"]),
            "topics": [t[0] for t in talk["topics"]],
            "day": talk["talkDay"],
            "time_start": parse_time(talk["talkStartTime"]),
            "time_end": parse_time(talk["talkEndTime"]),
            "hall": talk["hall"]["ru"],
            "related_talks": filter_related(talk["relatedTalks"])
        }
    )
)
```

3. Chunking для retrieval

Разбиваем документы на чанки с учётом лимита токенов



Documents
(после cleaning)



Chunks
для embedding и retrieval

```
from langchain_text_splitters import RecursiveCharacterTextSplitter

splitter = RecursiveCharacterTextSplitter.\
    from_tiktoken_encoder(
        model_name="gpt-4o-mini",
        chunk_size=700,
        chunk_overlap=0,
    )

# split по токенам (~700 токенов на чанк)
chunks = splitter.split_documents(docs)
```


Семантическое пространство чанков Heisenbug talks

После cleaning, normalization и chunking близкие доклады группируются по смыслу



Проекция: UMAP (cosine)

Каждая точка = 1 chunk / карточка доклада

Векторизация и хранение

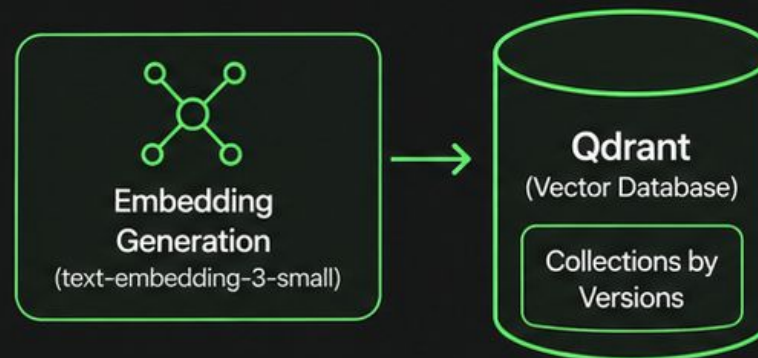
Stage 1: Data Preparation

(Start with original data sources)

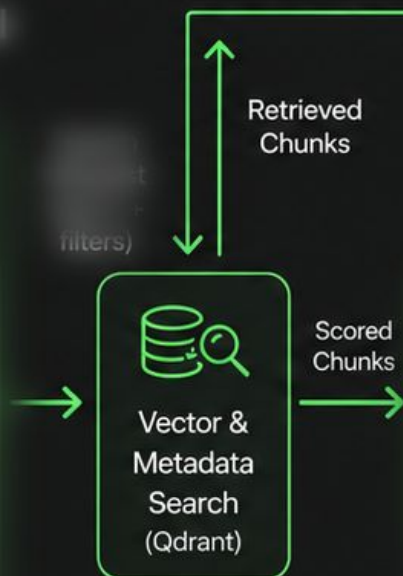


Stage 2: Embedding & Storage

(Focus on vectors)



Search Request top-k + metadata filters



Пример chunk

LLM'изация тестирования в Яндексе

Как измерить реальный эффект GenAI в тестировании на масштабе более чем в тысячу QA-инженеров и отделить пользу от вау-эффекта?

Спикеры: Владислав Миронов —
Руководитель общей инфраструктуры Фантеха — Яндекс

Топики: AI, Best Practices

День 1, 2026-04-27, 09:15–10:00, Зал 1

Связанные доклады:

- Как автоматизировать тестирование ИИ-агентов
- Тестирование ИИ-копилотов для разработки

Embedding model



- модель: text-embedding-3-small
- размерность: 1536
- batch processing
- embedding кодирует смысл, а не оригинальный текст

```
response = client.embeddings.create(
    model="text-embedding-3-small",
    input=batch,
    dimensions=1536,
)
```

Что получается

[0.021, -0.144, 0.087, ..., 0.312]



Что мы загружаем



talk_id
48ab465c0a7844f5b79551b6bb015816



embedding vector [1536]
[0.021, -0.144, 0.087, ...]



payload metadata

- title
- topics
- day
- speakers
- hall
- chunk_text
- has_useful_long_desc



Qdrant Point

```
{
  "id": "uuid5(heisenbug:talk:48ab...)",
  "vector": [0.021, -0.144, 0.087, ...],
  "payload": {
    "talk_id": "48ab465c0a7844f5b79551b6bb015816",
    "title": "LLM'изация тестирования в Яндексе...",
    "topics": ["AI", "Best Practices"],
    "day": 1,
    "speakers": ["Владислав Миронов"],
    "hall": "Зал 1",
    "chunk_text": "# LLM'изация тестирования...",
    "has_useful_long_desc": true
  }
}
```

src/vector_store/qdrant_store.py



Как устроена коллекция



- heisenbug_v1
- heisenbug_v2
- heisenbug_v3
- v4 использует ту же коллекцию, что и v3



vector size = 1536



distance = cosine



payload indexes:

- topics → keyword
- day → integer
- speakers → keyword

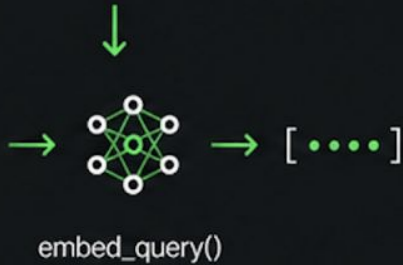
```
client.create_collection(
    collection_name=name,
    vectors_config=VectorParams(
        size=1536,
        distance=Distance.COSINE,
    ),
)
```

Запрос пользователя



Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

query



Поиск в Qdrant



```
results = client.query_points(
    collection_name=name,
    query=query_vector,
    query_filter=qdrant_filter,
    limit=top_k,
    with_payload=True,
)
```

Фильтры metadata в проекте

- day
- topics
- speakers (индекс есть)

Что получает retriever

- 1** LLM'изация тестирования в Яндексе... 0.91
 topics: AI, Best Practices | day: 1
 speakers: Владислав Миронов
- 2** Как автоматизировать тестирование ИИ-агентов... 0.87
 topics: AI, Testing, Automation | day: 1
 speakers: Алексей Петров
- 3** Тестирование ИИ-копилотов для разработки... 0.84
 topics: AI, Tools, Best Practices | day: 1
 speakers: Анна Смирнова

- semantic score показывает близость по смыслу
- payload metadata помогает фильтровать и объяснять результат
- дальше эти chunks идут в следующий этап RAG

Retrieval = semantic similarity + **metadata-aware filtering**

1. Embedding Generation

Chunk -> embedding vector



- input: chunk_text
- model: text-embedding-3-small
- dimensions: 1536
- batch embedding
- compact semantic representation

```
from openai import OpenAI

client = OpenAI()

response = client.embeddings.create(
    model="text-embedding-3-small",
    input=batch,
    dimensions=1536,
)
```



2. Qdrant storage

Векторы и payload metadata хранятся в коллекциях



Point

- id: uuid5(heisenbug:talk:{talk_id})
- vector: [1536]
- payload:
 - talk_id
 - title
 - topics
 - day
 - speakers
 - hall
 - chunk_text
 - has_useful_long_desc

```
client.create_collection(
    collection_name=name,
    vectors_config=VectorParams(
        size=1536,
        distance=Distance.COSINE,
    ),
)
```

Similarity metric:
cosine distance




Collections:

- heisenbug_v1
- heisenbug_v2
- heisenbug_v3
- v4 использует ту же коллекцию, что и v3

Metadata indexes:

- topics -> keyword
- day -> integer
- speakers -> keyword

 Metadata нужны не для семантики, а для точного управления retrieval

Каждый chunk превращается в числовой вектор



Embedding не хранит текст как есть, а кодирует смысл



chunk_text



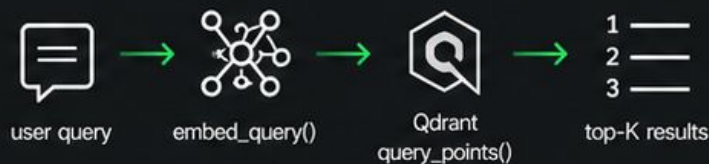
embedding model



1536-d vector

3. Semantic search + metadata filters

Запрос -> query embedding -> top-K retrieval



```
results = client.query_points(
    collection_name=name,
    query=query_vector,
    query_filter=qdrant_filter,
    limit=top_k,
    with_payload=True,
)
```

Фильтры в проекте:

- day
- topics
- speakers (индекс есть)

Но в runtime retrieval сейчас реально используются:

- day
- topics

Когда используются metadata

- если пользователь спрашивает «во второй день»
- если запрос про конкретную тему
- чтобы повысить precision
- чтобы сузить semantic top-K

1. Embeddings через LangChain

Chunk -> embedding vector



- input: chunk_text
- model: OpenAIEmbeddings (text-embedding-3-small)
- dimensions: 1536
- batch embedding
- compact semantic representation

```
from langchain_openai import OpenAIEmbeddings

embeddings = OpenAIEmbeddings(
    model="text-embedding-3-small",
    dimensions=1536,
)

vectors = embeddings.embed_documents(batch_texts)
```

Каждый chunk превращается в числовой вектор



Embedding не хранит текст как есть, а кодирует смысл



chunk_text OpenAIEmbeddings 1536-d vector

2. Хранение в Qdrant через LangChain

Векторы и метаданные сохраняются в коллекцию



```
from langchain_qdrant import QdrantVectorStore

vectorstore = QdrantVectorStore(
    client=client,
    collection_name="heisenbug_v3",
    embedding=embeddings,
    content_payload_key="chunk_text",
)
```

Структура точки в Qdrant

Point

- id: uuid5(heisenbug:talk:{talk_id})
- vector: [1536]
- payload:
 - talk_id
 - title
 - topics
 - day
 - speakers
 - hall
 - chunk_text
 - has_useful_long_desc

Создание коллекции

```
client.recreate_collection(
    collection_name="heisenbug_v3",
    vectors_config=VectorParams(
        size=1536,
        distance=Distance.COSINE,
    ),
)
```

Similarity metric:
cosine distance

Metadata indexes (payload indexes)

- topics -> keyword
- day -> integer
- speakers -> keyword



Collections:

- heisenbug_v1
- heisenbug_v2
- heisenbug_v3
- v4 использует ту же коллекцию, что и v3



Metadata (payload) не влияют на семантику, но позволяют фильтровать и управлять поиском

3. Поиск и фильтрация через LangChain

Запрос -> query embedding -> top-K results



```
from langchain_core.prompts import ChatPromptTemplate

query_vector = embeddings.embed_query(user_query)

results = vectorstore.similarity_search(
    query=user_query,
    k=top_k,
    filter=qdrant_filter, # metadata filters
)
```

Фильтры (metadata filters) в проекте

- day
- topics
- speakers (индекс есть)

Но в runtime retrieval сейчас реально используются:

- day
- topics

Когда используются metadata

- если пользователь спрашивает «во второй день»
- если запрос про конкретную тему
- чтобы повысить precision
- чтобы сузить semantic top-K

Пример фильтра (Qdrant filter через LangChain)

```
from langchain_qdrant import Filter, FieldCondition, MatchAny, MatchValue

qdrant_filter = Filter(must=[
    FieldCondition(key="day", match=MatchValue(value=2)),
    FieldCondition(key="topics", match=MatchAny(any=["AI", "Best Practices"])),
])
```

Запрос пользователя

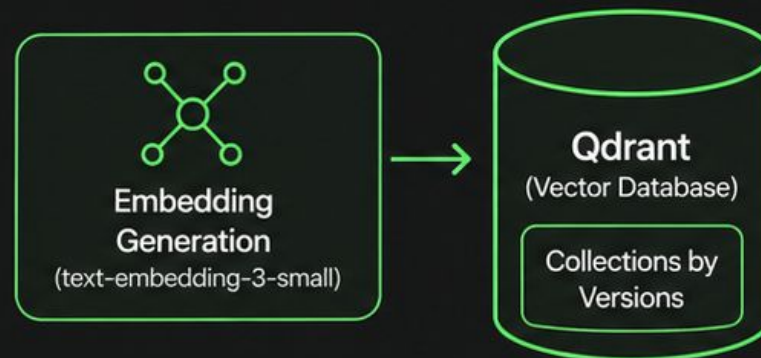
Stage 1: Data Preparation

(Start with original data sources)



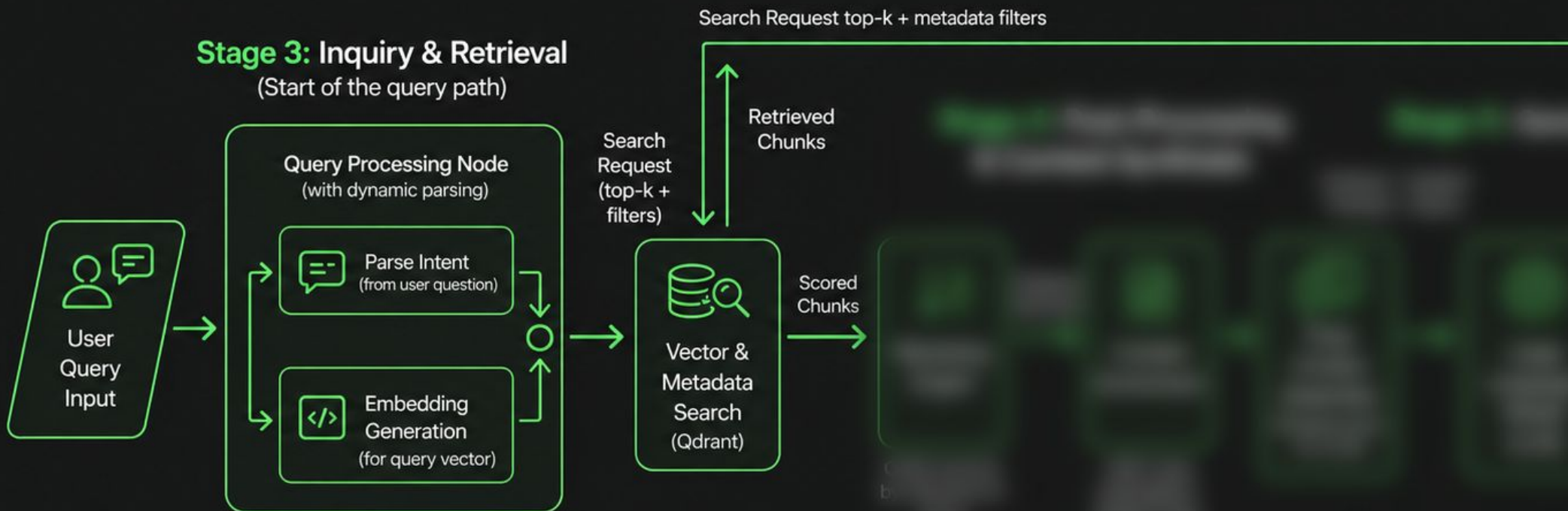
Stage 2: Embedding & Storage

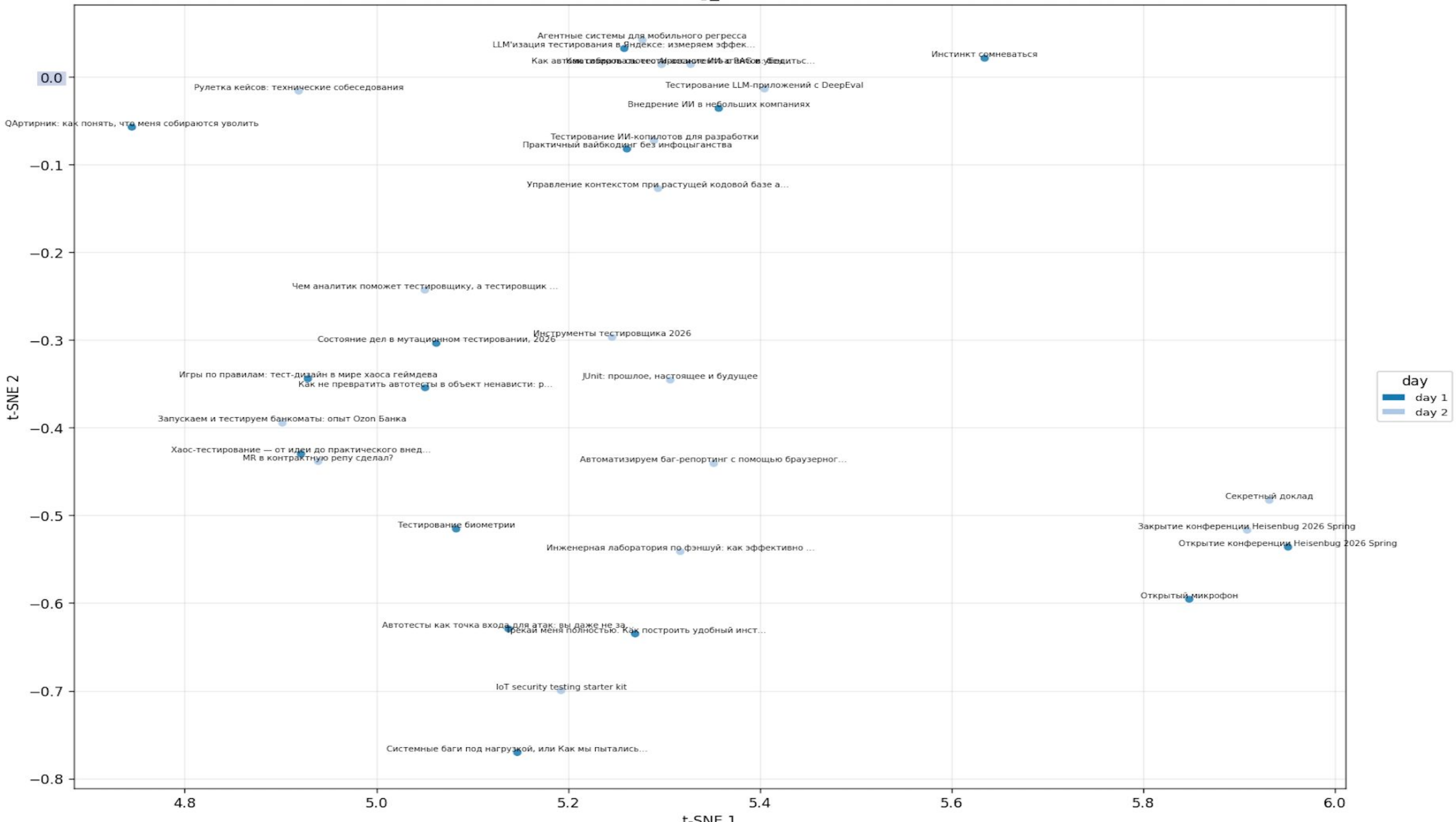
(Focus on vectors)



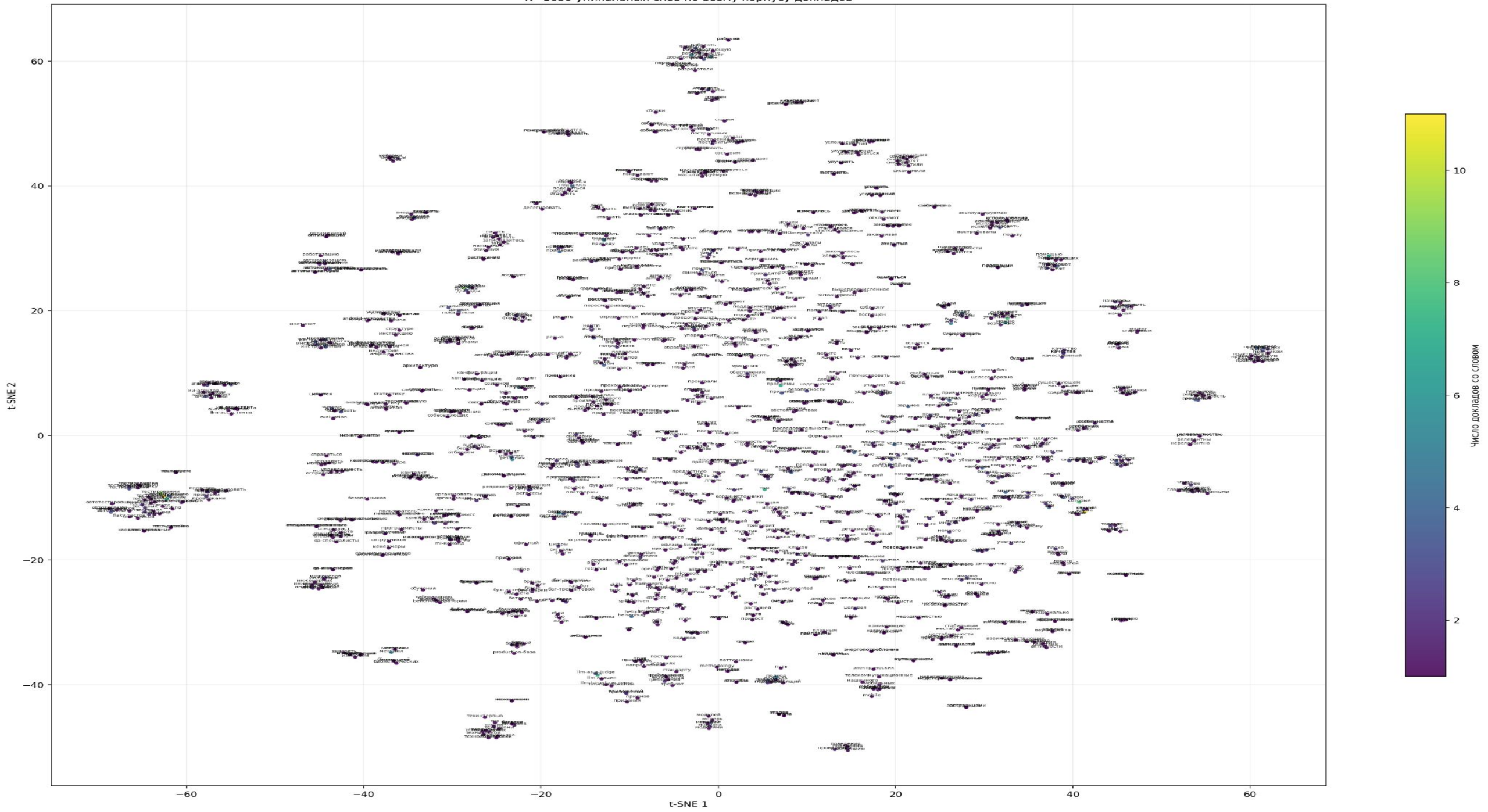
Stage 3: Inquiry & Retrieval

(Start of the query path)





Ключевые слова из описаний (title + short + long), позиция = семантика слова
N=1639 уникальных слов по всему корпусу докладов





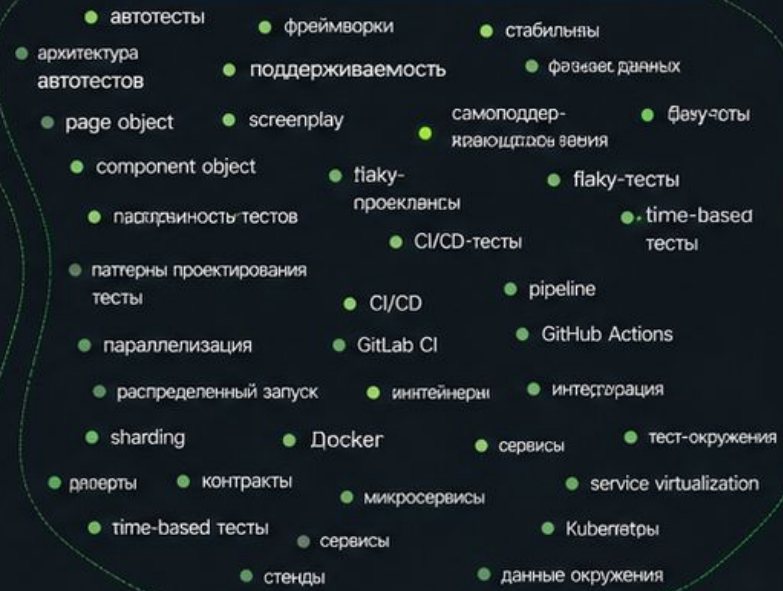
- Доклад / chunk (точка = один доклад)
- Дополнительный контент / соседний смысл
- 👤 Спикеры
- 🏢 Компании
- 🔗 Смысловой кластер (тема)
- Близость в embedding-пространстве (смысловая близость)

Semantic Y

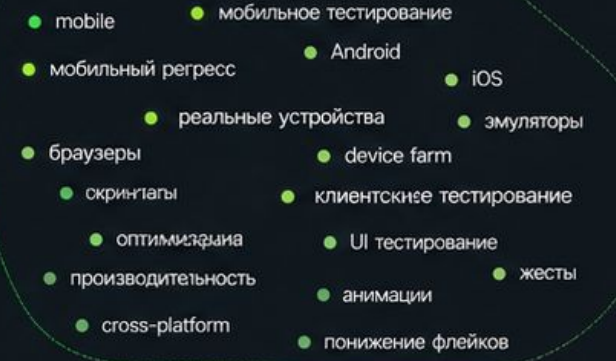
AI / GenAI / LLM / Агенты



Автоматизация / Автотесты / Инфраструктура



Mobile / Регресс / Клиентские платформы



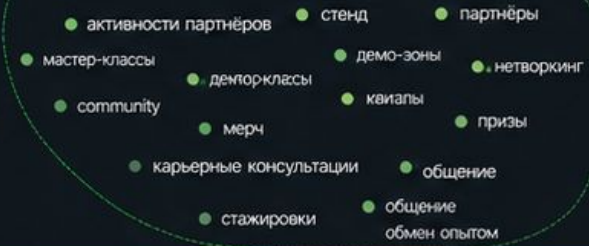
Спикеры / Компании / Доклады



Best Practices / QA Engineering



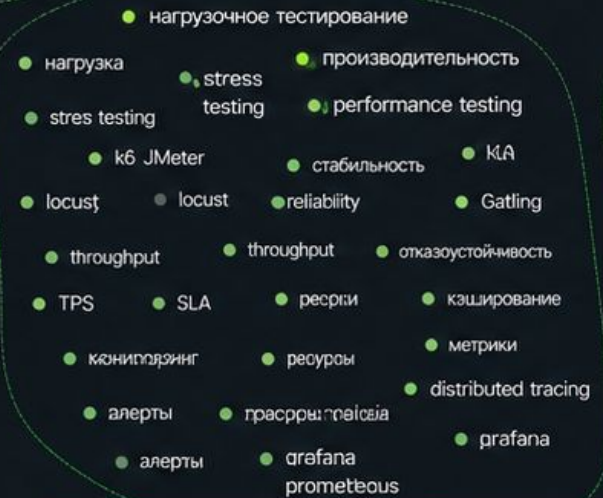
Активности партнёров / Community / Стенды



Security / Безопасность / IoT



Нагрузка / Производительность / Надёжность



● Ключевые термины / сущности
● Связанные термины
● Соседний контент

○ Смысловой кластер (тема)
— Близость в embedding-пространстве (смысловая близость)

Semantic X

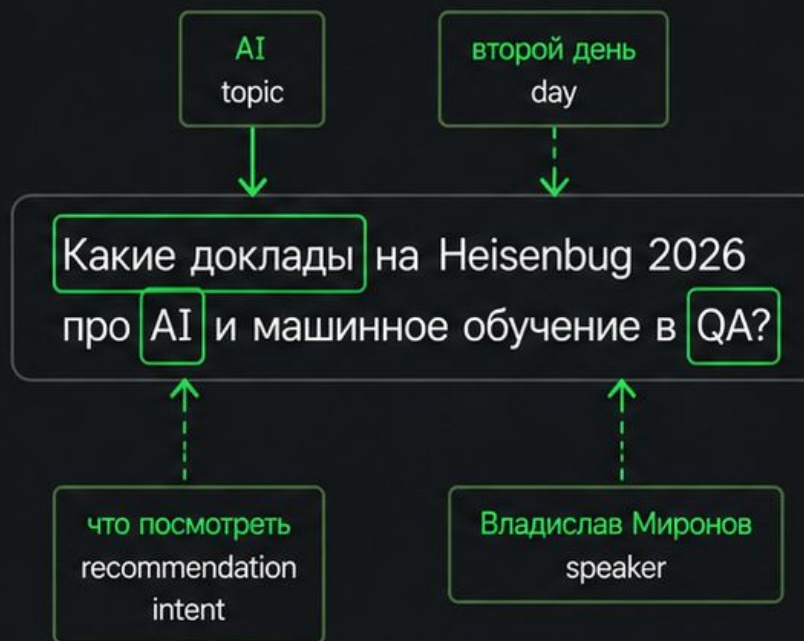
Примеры пользовательских вопросов

Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Что посмотреть про мобилки во второй день?

Какой доклад у Владислава Миронова?

Что скрыто внутри вопроса



Проблема: запрос пользователя неструктурирован



- свободный язык
- синонимы
- разные формулировки
- неявные ограничения
- нужен разбор перед retrieval



Пользовательский вопрос

Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?



query_analyzer.py

Модуль разбора intent

- known_topics
- known_speakers
- LLM parsing
- JSON output
- fallback parsing

```
intent = analyze_query(
    question,
    known_topics=...,
    known_speakers=...,
)
```



Структурированный intent (JSON)

```
{
  "topics": ["AI"],
  "day": null,
  "format_hint": "recommendation",
  "speaker": null
}
```

- **topics** -> фильтр по теме
- **"day"** -> ограничение по дню
- **format_hint** -> тип ответа
- **"speaker"** -> поиск по спикеру



Intent нужен не для генерации ответа,
а для **управления retrieval**

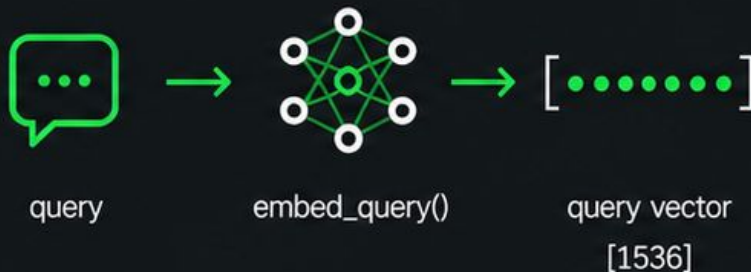
Пользовательский вопрос

Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Intent (из предыдущего шага)

```
{
  "topics": ["AI"]
}
```

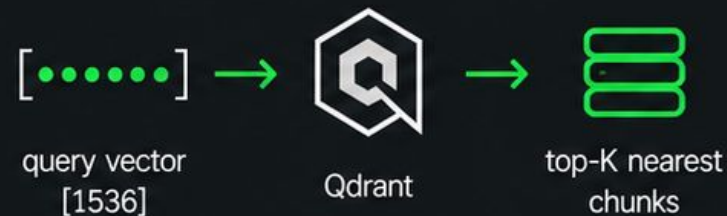
Построение query embedding



```
query_vector = embed_query(query)
```

- та же embedding-логика, что и для chunks
- query становится числовым вектором
- близкие по смыслу chunks находятся по cosine similarity

Semantic search в Qdrant



```
results = qdrant_search(
  version="v3",
  query_vector=query_vector,
  top_k=10,
  filters={"topics": ["AI"]}
)
```

- semantic similarity
- cosine distance
- top-K retrieval
- metadata filters



Semantic search отвечает за **смысловую близость**, metadata filters — за **управляемое сужение поиска**



user query

Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?



intent

```
{ "topics": ["AI"] }
```



query embedding

[1536]



Qdrant

semantic search + metadata filters



После поиска у нас есть top-K кандидатов

1

LLM'изация тестирования в Яндексe...

score

0.92

topics: AI Best Practices day: 1

speaker: Владислав Миронов

2

Как автоматизировать тестирование ИИ-агентов...

score

0.89

topics: AI day: 2

3

Тестирование ИИ-копилотов для разработки

score

0.87

topics: AI day: 2



Результаты отсортированы по semantic score (чем выше — тем ближе по смыслу)



Зачем нужны metadata

- сузить поиск по дню
- ограничить тему
- повысить precision
- объяснить, почему результат попал в top-K



Retrieval candidates -> дальше идут в reranking / enrichment / generation

1. Вопрос пользователя



Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Пользователь формулирует вопрос на естественном языке

- может быть тема
- может быть день
- может быть спикер
- может быть формат запроса

2. Parse Intent

LLM разбирает запрос в структурированный intent

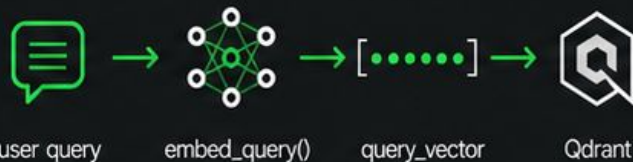
```
{
  "topics": ["AI"],
  "day": null,
  "format_hint": "recommendation",
  "speaker": null
}
```

- query_analyzer.py
- known_topics
- known_speakers
- JSON output
- fallback parsing

```
intent = analyze_query(
  question,
  known_topics=...,
  known_speakers=...,
)
```

3. Query Embedding + Search

Вопрос превращается в query vector и идёт в Qdrant



```
query_vector = embed_query(query)

results = qdrant_search(
  version="v3",
  query_vector=query_vector,
  top_k=10,
  filters={"topics": ["AI"]}
)
```

- semantic similarity
- cosine distance
- top-K retrieval
- metadata filters



Если в intent есть **day/topics**, retrieval сужается через metadata

4. Что находит retriever

На выходе — кандидаты, близкие по смыслу и подходящие по metadata

1 LLM'изация тестирования в Яндексе... 0.91

topics: AI, Best Practices | day: 1
speaker: Владислав Миронов

2 Как автоматизировать тестирование ИИ-агентов... 0.87

topics: AI | day: 2
speaker: Алексей Петров

3 Тестирование ИИ-копилотов для разработки 0.84

topics: AI | day: 2
speaker: Анна Смирнова

Это ещё не финальный ответ, а retrieval candidates

1. Вопрос пользователя



Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Пользователь формулирует вопрос на естественном языке

- может быть тема
- может быть день
- может быть спикер
- может быть формат запроса

2. Parse Intent (LangChain)

LLM разбирает запрос в структурированный intent

```
{
  "topics": ["AI"],
  "day": null,
  "format_hint": "recommendation",
  "speaker": null
}
```

```
from langchain_chat_models import ChatOpenAI
from pydantic import BaseModel, Field

class QueryIntent(BaseModel):
    topics: list[str] = Field(default_factory=list)
    day: int | None = None
    format_hint: str | None = None
    speaker: str | None = None

llm = ChatOpenAI(model="gpt-4o-mini", temperature=0)
structured_llm = llm.with_structured_output(QueryIntent)
intent = structured_llm.invoke({"question": question})
```



- LangChain: with_structured_output()
- Pydantic схема (QueryIntent)
- JSON-выход
- fallback parsing

3. Query Embedding + Search (LangChain)

Вопрос превращается в query vector и идёт в Qdrant



```
from langchain_openai import OpenAIEmbeddings
from langchain_qdrant import QdrantVectorStore
from qdrant_client import models
```

```
embeddings = OpenAIEmbeddings(
    model="text-embedding-3-small"
)

vector_store = QdrantVectorStore(
    client=qdrant_client,
    collection_name="heisenbug_v3",
    embedding=embeddings,
)

query_vector = embeddings.embed_query(question)

results = vector_store.similarity_search_by_vector(
    query_vector,
    k=10,
    filter=models.Filter(must=[
        models.FieldCondition(
            key="topics",
            match=models.MatchAny(any=["AI"])
        )
    ]),
    include_payload=True
)
```



- semantic similarity
- cosine distance
- top-K retrieval
- metadata filters

4. Что находит retriever (LangChain)

На выходе — кандидаты, близкие по смыслу и подходящие по metadata

1 LLM'изация тестирования в Яндексе... 0.91

topics: AI, Best Practices | day: 1
speaker: Владислав Миронов

2 Как автоматизировать тестирование ИИ-агентов... 0.87

topics: AI | day: 2
speaker: Алексей Петров

3 Тестирование ИИ-копилотов для разработки 0.84

topics: AI | day: 2
speaker: Анна Смирнова

Это ещё не финальный ответ, a retrieval candidates



Постобработка и сбор контекста



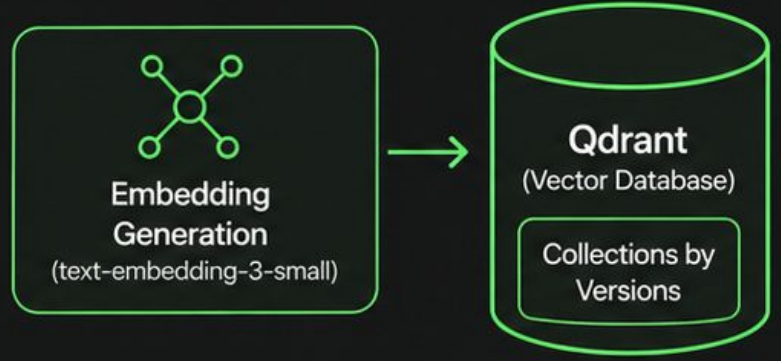
Stage 1: Data Preparation

(Start with original data sources)



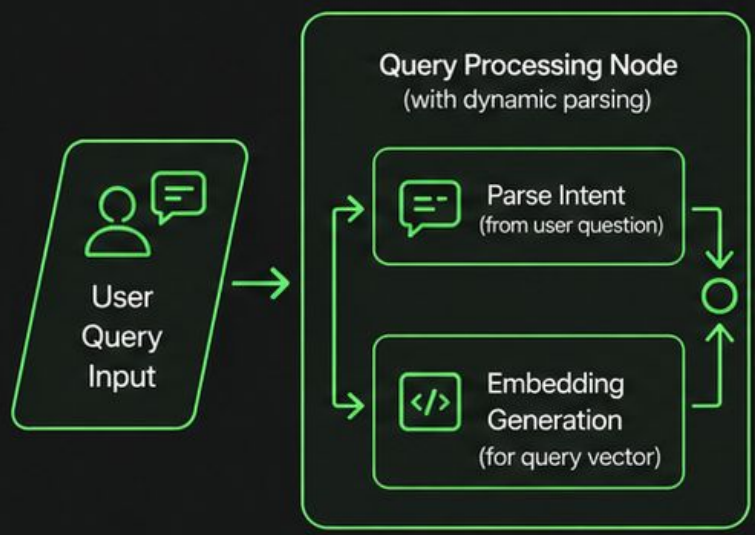
Stage 2: Embedding & Storage

(Focus on vectors)

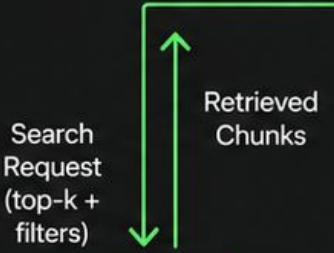


Stage 3: Inquiry & Retrieval

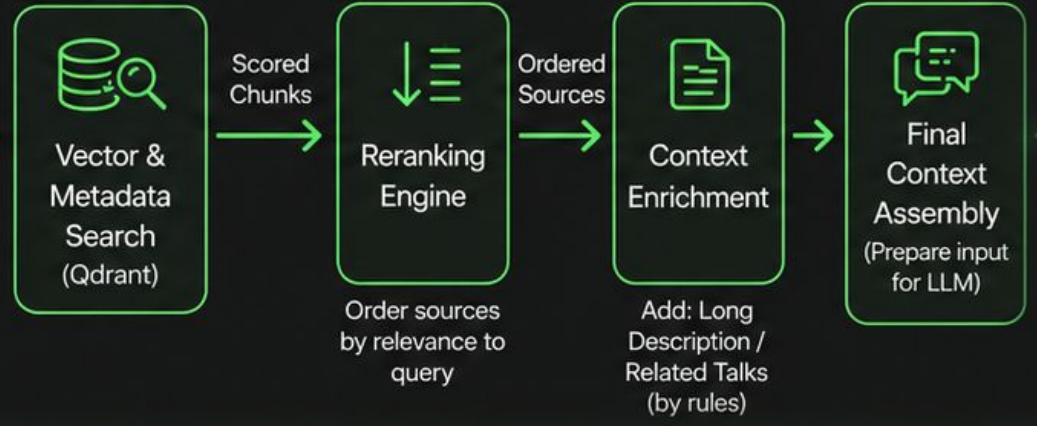
(Start of the query path)



Search Request top-k + metadata filters



Stage 4: Post-Processing & Context Synthesis



Context + Search Results + Prompt + LLM

Retrieval

Post-processing

Находит
ВОЗМОЖНЫЕ
ИСТОЧНИКИ

Делает их
ПОЛЕЗНЫМИ
для generation





Пользовательский
вопрос

Какие доклады на
Heisenbug 2026 про AI
и машинное обучение
в QA?



Retrieval уже выполнен
на предыдущем этапе



Qdrant



top-K
retrieved chunks



1

LLM'изация тестирования
в Яндексе...

score

0.92

topics: AI, Best Practices day: 1

2

Как автоматизировать
тестирование ИИ-агентов...

score

0.89

topics: AI day: 2

3

Тестирование ИИ-копилотов
для разработки

score

0.87

topics: AI day: 2

4

Как собрать своего
AI-ассистента с RAG...

score

0.84

topics: AI, RAG day: 1



Это ещё сырой набор
кандидатов

- порядок не идеален
- chunks ещё не готовы для prompt
- части контекста могут быть слишком короткими
- нужен дополнительный post-processing



Semantic retrieval находит
возможные источники,
но ещё не формирует
лучший контекст для LLM



До reranking

После semantic search

1

Тестирование ИИ-копилотов для разработки

score
0.87

topics: AI day: 2

2

LLM'изация тестирования в Яндексе...

score
0.92

topics: AI, Best Practices day: 1

3

Как автоматизировать тестирование ИИ-агентов...

score
0.89

topics: AI day: 2

4

Как собрать своего AI-ассистента с RAG...

score
0.84

topics: AI, RAG day: 1



LLM Reranker

- получает retrieved chunks
- получает parsed intent
- оценивает релевантность к текущему вопросу
- выбирает top-N

```
reranked = self.reranker.rerank(
    results,
    intent,
    top_n=min(top_k, RETIREVAL_TOP_N_RERANKED)
)
```

v3 / v4 pipeline



После reranking

Переупорядоченные кандидаты

cloud.ru

1

LLM'изация тестирования в Яндексе...

rerank_score
0.96

reason
наилучшее соответствие запросу по теме и контексту

topics: AI, Best Practices day: 1

2

Как автоматизировать тестирование ИИ-агентов...

rerank_score
0.93

reason
прямой фокус на автоматизации ИИ-агентов

topics: AI day: 2

3

Тестирование ИИ-копилотов для разработки

rerank_score
0.88

reason
связанная тема, чуть менее релевантно запросу

topics: AI day: 2

Retrieval

Отвечает
за recall

Reranking

Поднимает:

- precision
- MRR
- качество
контекста



Базовый chunk-контекст

- title**
LLM'изация тестирования в Яндексе
- short_description**
Как мы применяем LLM для ускорения и улучшения тестирования
- speakers**
Владислав Миронов
- topics**
AI, Best Practices
- schedule**
День 1, 11:00 – 11:45
- related talks (short)**
2 доклада

То, что уже было в retrieval

enricher.py

- если long_description_useful = true
- если format_hint подходит
- добавить long_description
- добавить related_talks_extended

```


if talk and talk.long_description_useful:
    wants_detail = intent.format_hint in (
        "detail", "comparison", None
    )
    
```

Dynamic enrichment по правилам

Enriched context


- chunk_text**
Как мы применяем LLM для ускорения и улучшения тестирования
- long_description** ДОБАВЛЕНО
Расширенное описание подхода, архитектуры, примеров использования, результатов и выводов из доклада (если long_description_useful = true)
- related_talks_extended** ДОБАВЛЕНО
Расширенный список релевантных докладов с пояснениями, почему они связаны с текущим вопросом

После enrichment: больше релевантных деталей для generation

 Before

Краткий контекст из индекса



 After

Расширенный контекст для generation

Упорядоченные и обогащённые источники

- 1** Доклад 1
- chunk_text
 - long_description
 - related talks (extended)

- 2** Доклад 2
- chunk_text
 - related talks (extended)

- 3** Доклад 3
- chunk_text

После reranking и enrichment

build_prompt()

```
prompt = build_prompt(
    question,
    enriched,
    include_long_desc=True,
    include_related=True,
)
```

Формируем final context для LLM

- ordered sources
- grounded facts
- links / ids / metadata
- единая структура prompt

Собранный final context

Вопрос:
Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Контекст:
[Доклад 1] ...
[Доклад 2] ...
[Доклад 3] ...
...

Instruction:
Отвечай только по контексту

Это уже вход в generation





1. Retrieved Chunks

На входе — top-K кандидаты из Qdrant

1 LLM'изация тестирования в Яндекссе... 0.92
score

topics: AI, Best Practices
day: 1
speaker: Владислав Миронов

2 Как автоматизировать тестирование ИИ-агентов... 0.89
score

topics: AI
day: 2
speaker: Алексей Петров

3 Тестирование ИИ-копилотов для разработки 0.87
score

topics: AI
day: 2
speaker: Анна Смирнова

Это результат retrieval из предыдущего этапа



2. Reranking

LLM пересортировывает кандидатов по релевантности к запросу

reranker.py
(LLM Reranker)

- вход: retrieved chunks + parsed intent
- оценивает релевантность уже к конкретному вопросу
- выбирает top-N
- улучшает precision / MRR

```
reranked = self.reranker.rerank(
    results,
    intent,
    top_n=min(top_k,
              RETRIEVAL_TOP_N_RERANKED)
)
```

v3 / v4: retrieval -> reranking

Semantic search даёт кандидатов, reranking уточняет порядок



3. Context Enrichment

К лучшим кандидатам добавляется дополнительный контекст по правилам

long_description (если полезно) + related_talks_extended (расширяющий контекст)

- long_description добавляется не всегда
- только если long_description_useful = true
- зависит от format_hint
- related talks добавляются как расширяющий контекст

short chunk + useful long description (опционально) + related talks (расширение)

enriched context

```
if talk and talk.long_description_useful:
    wants_detail = intent.format_hint in
        ("detail", "comparison", None)
```

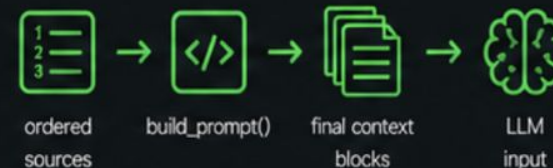
src/generation/enricher.py

long_description не участвует в embedding, но может усиливать generation



4. Final Context Assembly

Собираем финальный prompt для LLM из лучших и уже enriched источников



```
prompt = build_prompt(
    question,
    enriched,
    include_long_desc=True,
    include_related=True,
)
```

Пример финального контекста (фрагмент)

```
[Доклад 1]
• chunk_text
• long_description
• related talks

[Доклад 2]
• chunk_text
• ...
...
```



1. Retrieved Chunks

На входе — top-K кандидаты из Qdrant через LangChain

1 LLM'изация тестирования в Яндексе... 0.92
score

topics: AI, Best Practices day: 1
speakers: Владислав Мионов

2 Как автоматизировать тестирование ИИ-агентов... 0.89
score

topics: AI day: 2
speakers: Алексей Петров

3 Тестирование ИИ-копилотов для разработки 0.87
score

topics: AI day: 2
speakers: Анна Смирнова

Это результат retrieval из предыдущего этапа



2. Reranking (LangChain)

LangChain пересортировывает кандидатов по релевантности к запросу



ContextualCompressionRetriever с LLM Reranker

- вход: retrieved chunks + parsed intent
- оценивает релевантность к конкретному вопросу
- выбирает top-N
- улучшает precision / MRR

```
from langchain.retrievers import ContextualCompressionRetriever
from langchain.retrievers.document_compressors import LLMChainExtractor
```

```
compressor = LLMChainExtractor.from_llm(llm)
retriever = ContextualCompressionRetriever(
    base_retriever=vector_retriever,
    base_compressor=compressor
)

reranked_docs = retriever.get_relevant_documents(
    query=question
)
```

Используем LangChain: retrieval -> reranking



Semantic search даёт кандидатов, reranking уточняет порядок



3. Context Enrichment (LangChain)

К лучшим кандидатам добавляется дополнительный контекст по правилам

long_description (если полезно)

+ related_talks_extended (расширяющий контекст)

- long_description добавляется не всегда
- только если long_description_useful = true
- зависит от format_hint
- related talks добавляются как расширяющий контекст

short chunk

+ useful long_description (опционально)

+ related talks (расширение)

enriched context

```
from langchain.schema import Document
def enrich(doc: Document, talk, intent):
    if talk and talk.long_description_useful:
        wants_detail = intent.format_hint in (
            "detail", "comparison", None
        )
        if wants_detail:
            doc.page_content += "\n\n" + talk.long_description
    # related talks добавляются как контекст
    return doc
```



Реализуем через LangChain Document с метаданными и кастомной обработкой



4. Final Context Assembly (LangChain)

Собираем финальный промт для LLM из лучших и уже enriched источников



ordered documents



Stuff/MapReduce/Refine Chain



final context blocks



LLM input

```
from langchain.chains.combine_documents import create_stuff_documents_chain
from langchain.chains import create_retrieval_chain

document_chain = create_stuff_documents_chain(llm, prompt)
rag_chain = create_retrieval_chain(retriever, document_chain)

result = rag_chain.invoke({"input": question})
```

Пример финального контекста (фрагмент)

```
[Доклад 1]
chunk_text
long_description (если добавлена)
related talks
```

```
[Доклад 2]
chunk_text
related talks
...
```

Генерация ответа

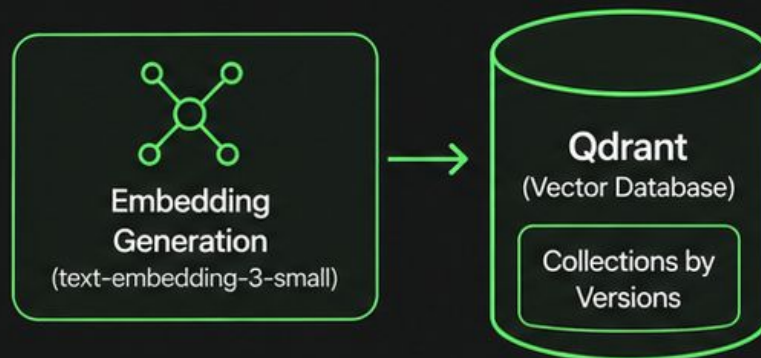
Stage 1: Data Preparation

(Start with original data sources)



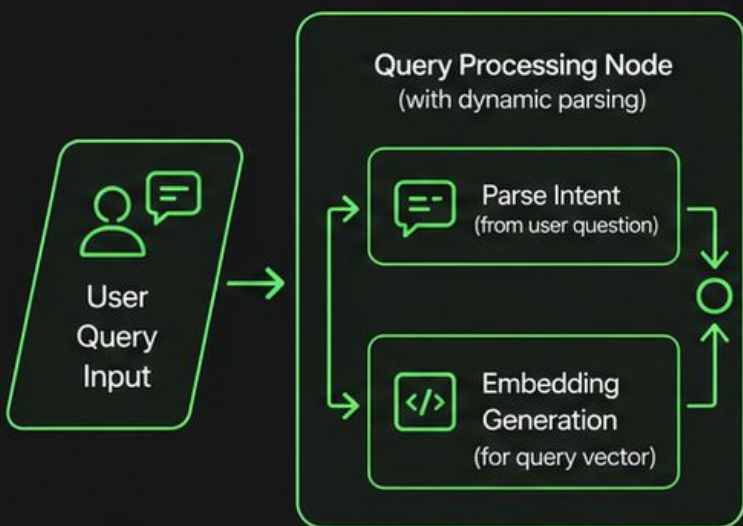
Stage 2: Embedding & Storage

(Focus on vectors)

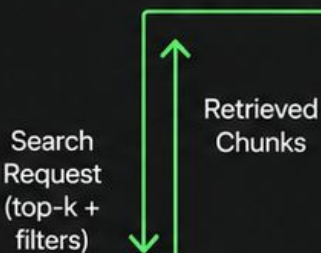


Stage 3: Inquiry & Retrieval

(Start of the query path)



Search Request top-k + metadata filters

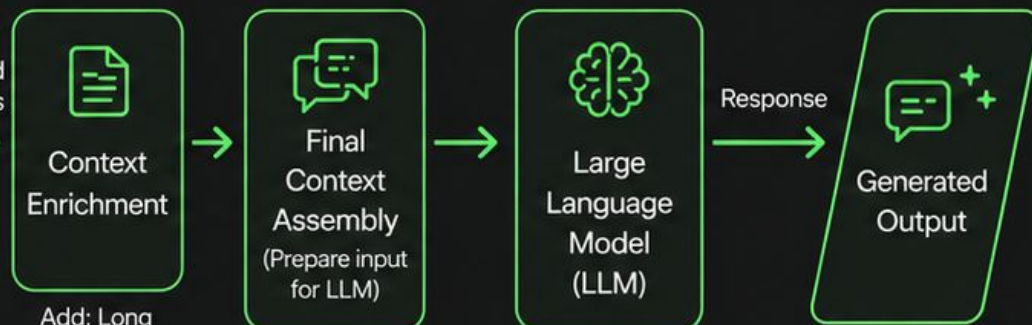


Stage 4: Post-Processing & Context Synthesis



Stage 5: Generation & Output

Context + System Prompt + Query



Stage 4: подготовка контекста



Это результат предыдущего Stage 4

Prompt / Context для LLM

Вопрос:

"Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?"

Контекст:

- [Доклад 1]
 LLM'изация тестирования в Яндексе...
 - short_description
 - long_description (если добавлен)
 - related talks

- [Доклад 2]
 Как автоматизировать тестирование ИИ-агентов...
 - ...

- [Доклад 3]
 Тестирование ИИ-копилотов для разработки
 - ...

Инструкции:

- отвечай только по контексту
- не выдумывай

Что входит в final context:

- question
- ordered sources
- enriched text
- grounded instructions

ВХОД

Final Context

Что это:
готовый prompt / context
после Stage 4
(сбран, упорядочен,
обогащён)

LLM

- model: gpt-4o-mini
- grounded generation
- temperature = 0.3



Generator

```
gen_result = self.generator.generate(
    prompt,
    [e.talk_id for e in enriched]
)
```

Generator stage в pipeline



ПРИНЦИПЫ GENERATION

- отвечает только по контексту

- не должен добавлять знания
"из головы"

- использует найденные
источники

Generated Answer

```
{
  "answer": "...",
  "sources": [ ... ],
  "cited_talk_ids": [
    "talk_001", "talk_002"
  ],
  "structured": true
}
```

- answer
- sources
- cited_talk_ids
- structured response



1 LLM'изация тестирования в Яндексе:
измеряем эффект от AI в команде
из 1000+ QA-инженеров

- Спикер: Владислав Миронов
- Время: День 1, 2026-04-27, 09:15-10:00
- Зал: 1
- Подробнее: <https://heisenbug.ru/talks/.../>

2 Как автоматизировать тестирование
ИИ-агентов: бенчмарки, метрики,
LLM-as-a-Judge

- Спикер: ...
- Время: День 2, 2026-04-28, 11:00-11:45
- Зал: 2
- Подробнее: <https://heisenbug.ru/talks/.../>

...



Product Output

- Telegram HTML
- русский язык
- ссылки на доклады
- grounded answer
- честный fallback
при недостатке данных



Если нужного доклада нет
в контексте, ассистент должен
прямо сказать, что данных
недостаточно



1. Final Context

На входе — prompt, собранный на предыдущем этапе

Вопрос:

Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Контекст:

[Доклад 1] ...
[Доклад 2] ...
[Доклад 3] ...
...

Инструкция:

Отвечай только по контексту. Если данных недостаточно — скажи об этом.

Результат Stage 4:
assembled prompt для LLM



ordered sources



enriched context



only grounded facts



2. LLM Generation

Модель отвечает только по переданному контексту

Generator / LLM

- model: gpt-4o-mini
- temperature = 0.3
- grounded generation
- без выхода за пределы контекста

```
gen_result = self.generator.generate(
    prompt,
    [e.talk_id for e in enriched]
)
```

Generation stage



LLM здесь не ищет новые знания, а формирует ответ из уже собранного контекста



3. Generated Answer

На выходе — уже не chunks, а готовый пользовательский ответ



LLM'изация тестирования в Яндексе: измеряем эффект от AI в команде из 1000+ QA-инженеров



Спикер: Владислав Миронов



Время: День 1, 2026-04-27, 09:15-10:00



Зал: 1



Подробнее: <https://heisenbug.ru/talks/...>

```
{
  "answer": "...",
  "cited_talk_ids": ["talk_001", "talk_002"],
  "sources": [
    {"talk_id": "talk_001", "title": "..."},
    {"talk_id": "talk_002", "title": "..."}
  ]
}
```



4. Output to User

Ответ возвращается в прикладном формате

Heisenbug Bot

Вот доклады на Heisenbug 2026 про AI и машинное обучение в QA:

1. LLM'изация тестирования в Яндексе: измеряем эффект от AI в команде из 1000+ QA-инженеров [Подробнее](https://heisenbug.ru/talks/...)
Спикер: Владислав Миронов
День 1, 2026-04-27, 09:15-10:00
Зал: 1
2. Как автоматизировать тестирование ИИ-агентов... [Подробнее](https://heisenbug.ru/talks/...)
День 2, 2026-04-28, 11:00-11:45
[Подробнее](https://heisenbug.ru/talks/...)

...

12:34 ✓



grounded output



product-ready formatting



source-aware response



result for user interface



Если в контексте недостаточно данных, модель должна честно это сказать



1. Final Context

На входе — prompt, собранный на предыдущем этапе

Вопрос:

Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?

Контекст:

[Доклад 1] ...
[Доклад 2] ...
[Доклад 3] ...
...

Инструкция:

Отвечай только по контексту. Если данных недостаточно — скажи об этом.

Результат Stage 4:
assembled prompt для LLM



ordered sources



enriched context



only grounded facts



2. LLM Generation

Модель отвечает только по переданному контексту

LangChain + LLM



LangChain



Chains



LLM

- model: gpt-4o-mini
- temperature = 0.3
- grounded generation
- без выхода за пределы контекста

```
from langchain.schema import HumanMessage
messages = [
    HumanMessage(content=prompt)
]
response = llm.invoke(messages)
# (или chain.invoke({...}))
```

Generation stage



LLM здесь не ищет новые знания, а формирует ответ из уже собранного контекста



3. Generated Answer

На выходе — уже не chunks, а готовый пользовательский ответ



LLM'изация тестирования в Яндексе: измеряем эффект от AI в команде из 1000+ QA-инженеров



Спикер: Владислав Миронов



Время: День 1, 2026-04-27, 09:15-10:00



Зал: 1



Подробнее: <https://heisenbug.ru/talks/...>

```
{
  "answer": "...",
  "cited_talk_ids": ["talk_001", "talk_002"],
  "sources": [
    {"talk_id": "talk_001", "title": "..."},
    {"talk_id": "talk_002", "title": "..."}
  ]
}
```



4. Output to User

Ответ возвращается в прикладном формате

Heisenbug Bot

Вот доклады на Heisenbug 2026 про AI и машинное обучение в QA:

1. **LLM'изация тестирования в Яндексе: измеряем эффект от AI в команде из 1000+ QA-инженеров**
Спикер: Владислав Миронов
День 1, 2026-04-27, 09:15-10:00
Зал: 1
<https://heisenbug.ru/talks/...>Подробнее
2. **Как автоматизировать* тестирование ИИ-агентов...**
День 2, 2026-04-28, 11:00-11:45
<https://heisenbug.ru/talks/...>Подробнее

... 12:34 ✓

- ✓ grounded output
- ✓ product-ready formatting
- ✓ source-aware response
- ✓ result for user interface



Если в контексте недостаточно данных, модель должна честно это сказать

Архитектура

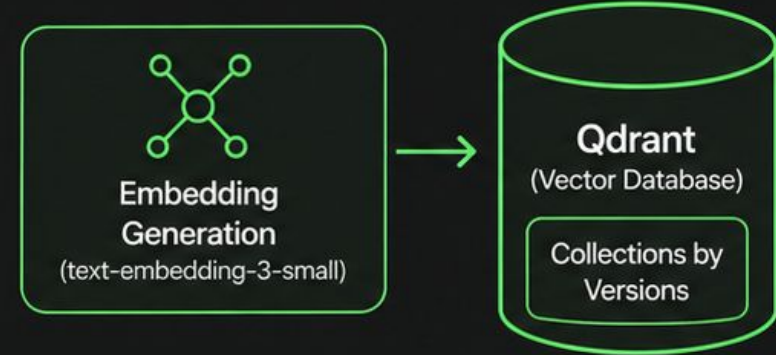
Stage 1: Data Preparation

(Start with original data sources)



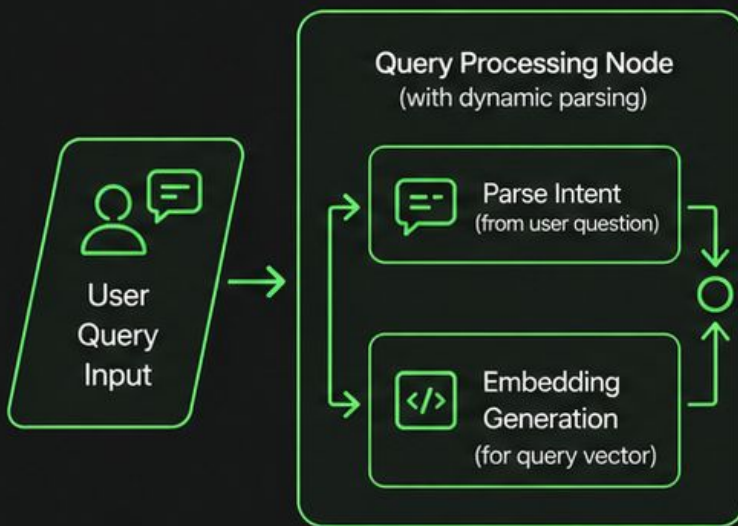
Stage 2: Embedding & Storage

(Focus on vectors)



Stage 3: Inquiry & Retrieval

(Start of the query path)



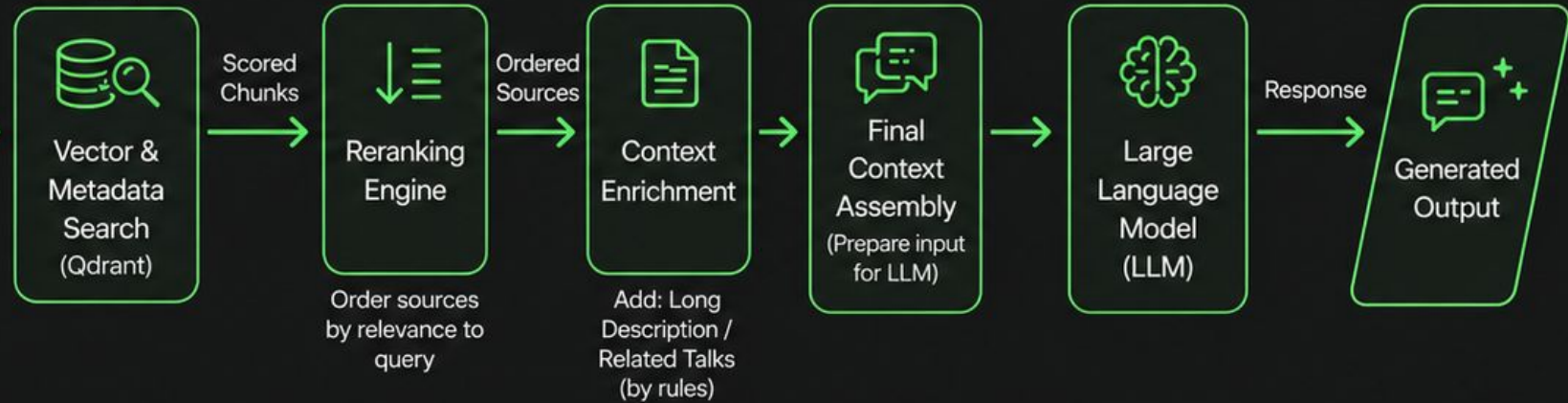
Search Request top-k + metadata filters

Search Request (top-k + filters)

Retrieved Chunks

Stage 4: Post-Processing & Context Synthesis

Stage 5: Generation & Output



Context + System Prompt + Query



Retrieve



Augment



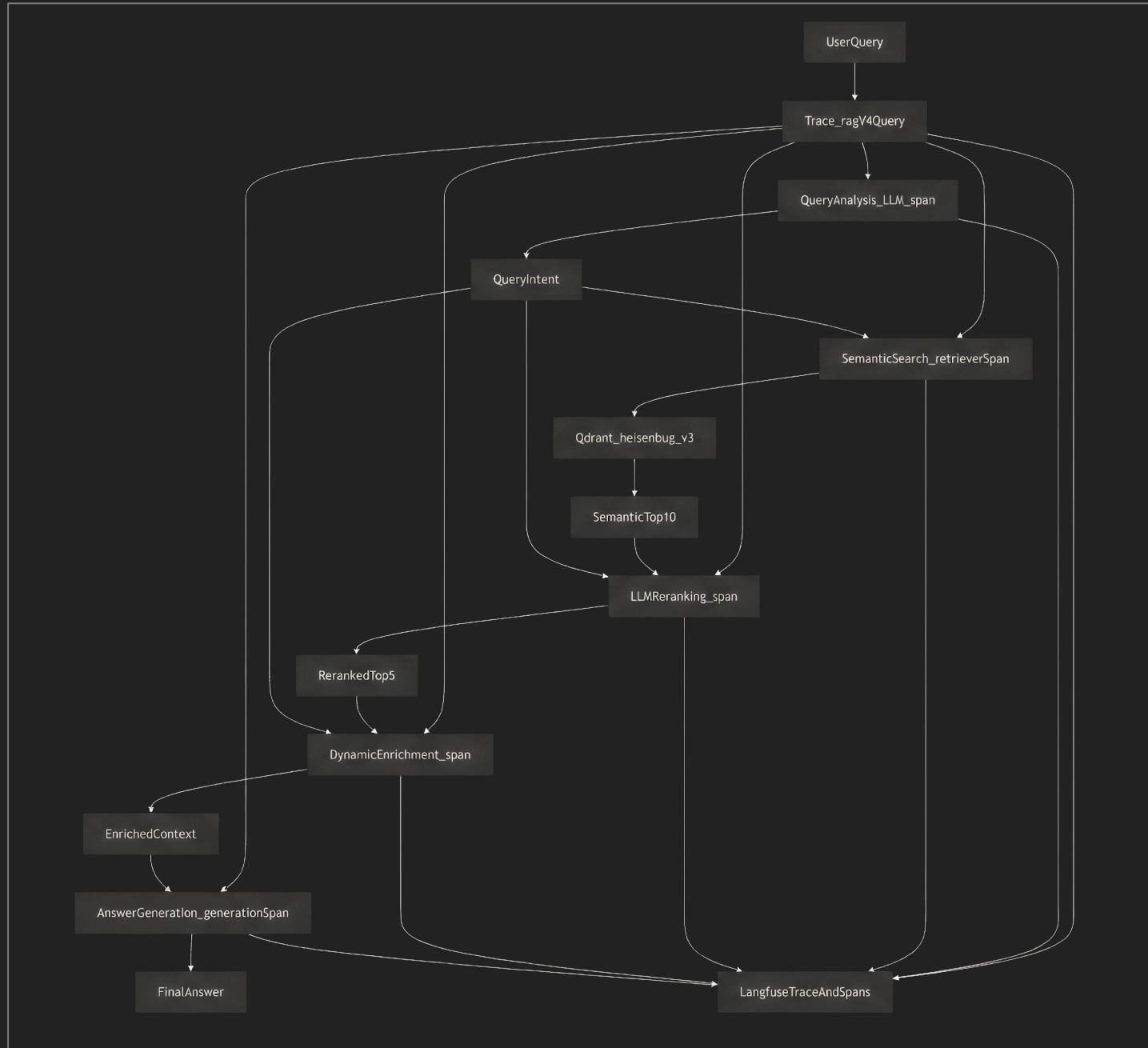
Generate

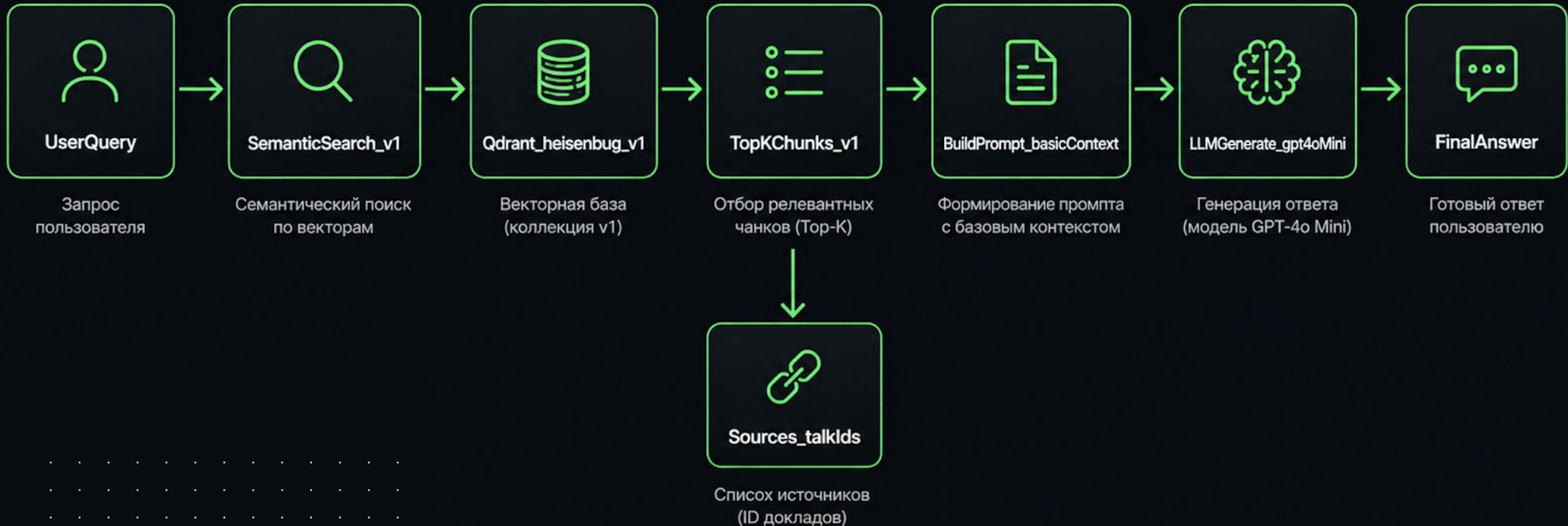


Evaluate

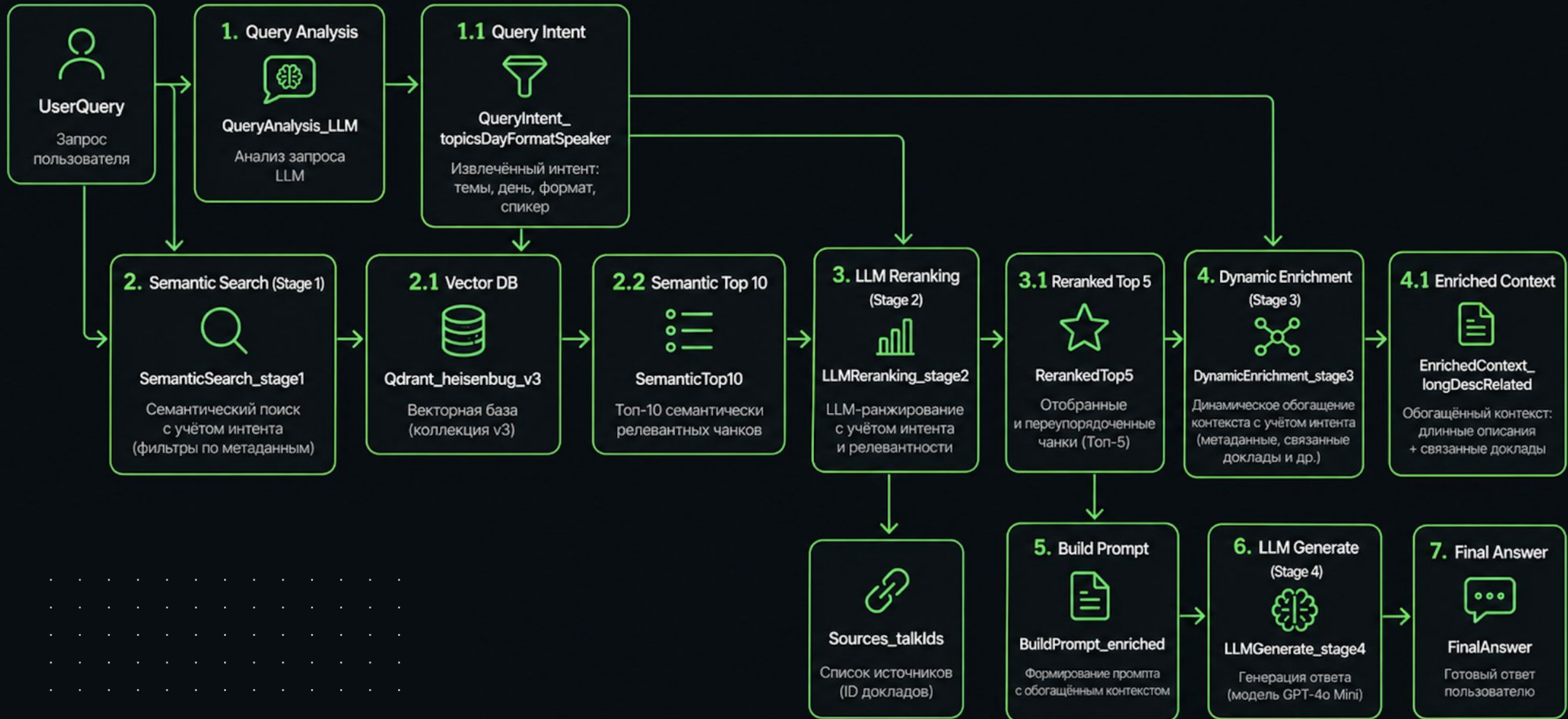


Iterate











 A large graphic featuring the text "Q&A" in a bold, sans-serif font. The "Q" and "A" are white, while the ampersand "&" is a vibrant green. The text is centered within a thin green circular outline. The background is dark green with decorative elements: concentric circles in the top-left, a grid of dots in the top-right, and wavy lines in the bottom-right. Scattered throughout are small green plus signs and circles.

Перерыв?

10-15 минут

Встречаемся в
`datetime.now()`
+
`timedelta(minutes=15)`





@hb2026_workshop_bot

@HB2026_WORKSHOP
_BOT

Стартуем!



Сборка AI-ассистента
с RAG изнутри

1 часть

Проверка качества

2 часть



Проверка качества

2 часть





ДОКЛАД 27.04 / 13:30 – 14:15 (UTC+3)

AI

Зал 1



Материалы ▾

Редактировать отзыв 😊



Как автоматизировать тестирование ИИ-агентов: бенчмарки, метрики, LLM-as-a-Judge

В процессах контроля качества AI-агентов уже сегодня назрел вопрос автоматизации. Классические автотесты плохо работают с недетерминированным поведением агентов, а ручная проверка не масштабируется. В свою очередь, LLM не только порождает проблемы, но и может помочь их решить, если делегировать проверку недетерминированных данных для LLM.

Автоматизация оценки качества через LLM-as-a-Judge и внедрение такого подхода в виде автотестов имеют как огромные плюсы, так и очень много нюансов, которые будут полезны всем, кто задумается: «А почему бы мне не написать автотесты на агента?»

Доклад будет полезен QA-инженерам, автоматизаторам и всем, кто столкнулся с задачей тестирования или разработки AI-агентов и задумался о регрессионном тестировании их поведения.

Спикеры

**Артем Симешин**

Сбер

Приглашенные эксперты

**Михаил Новотарский**

Сбер



Все остальное дорого и бесполезно (Полезно для богатых)



М/с «Футурама» (*Futurama*), созд. Мэтт Грейнинг, 1999–2013 / 2023–н.в.



Retrieve

Поиск релевантной информации



Augment

Дополнение контекста и данными



Generate

Генерация ответа ассистентом



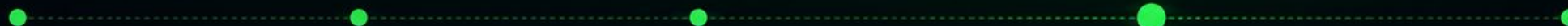
Evaluate

Оценка качества ответа и релевантности



Iterate

Улучшение и повторный цикл



- ★ Качество и точность
- ★ Релевантность контекста
- ★ Полезность для пользователя
- ★ Безопасность и достоверность

Проверка качества

Retrieval

Generation

Golden dataset

Q & A

Проверка качества

Retrieval vs Generation

Ключевые метрики

Golden dataset

LLM-as-a-Judge



ДОКЛАД 28.04 / 15:30 – 16:15 (UTC+3)

AI

Tools/Frameworks

Зал 2



Материалы ▾

Оценить выступление 😊



Тестирование LLM-приложений с DeepEval

Рассмотрим практику тестирования приложений, построенных на основе больших языковых моделей (LLM). Основной фокус — решение проблем оценки недетерминированных LLM-приложений с помощью специализированного инструмента DeepEval, который позволяет автоматизировать оценку качества работы LLM-приложений через подход LLM-as-a-Judge, а также применять различные метрики и тестовые сценарии. Доклад затронет основные концепции инструмента.

Также на примере кейса покажу, как интегрировать DeepEval в процесс разработки, чтобы оценивать релевантность ответов и контролировать качество системы.

Целевая аудитория — инженеры машинного обучения, разработчики AI-продуктов и QA-специалисты, сталкивающиеся с необходимостью обеспечения надежности и предсказуемости LLM-приложений.

Спикеры



Максим Максимов
red_mad_robot

Приглашенные эксперты



Софья Селезнева
Гранч



1. Retrieval может сломать всё раньше

Запрос



Что расскажет Владислав Миронов?



Retrieval (поиск по базе Heisenbug)



- Speaker распознан: Владислав Миронов ✓
- Нужный доклад не найден ✗
- В candidates попали нерелевантные chunks



Ответ ассистента



В предоставленном контексте нет информации о Владиславе Миронове.

2. Хороший текст не всегда grounded

Запрос



Что происходит в первый день с утра: открытие и кейноут?



Ответ ассистента



<h3>Открытие и кейноут – первый день</h3>



10:00 – 10:30 Открытие конференции Heisenbug
Приветственное слово и обзор программы.



10:30 – 11:30 Кейноут: Engineering the Unknown
Спикер: Владислав Миронов

<https://heisenbug.ru/talks/1234>

Проверка groundedness



Часть ответа подтверждена (спикер, доклад)

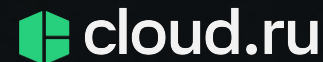


Часть ответа неполна (время может отличаться)



Формат ответа не совсем по задаче (нужен список)

3. Offline хорошо != online стабильно



Offline
(контролируемая среда)

VS



Online
(реальный мир)

- Golden dataset
- Фиксированные тесты
- Ожидаемые ответы

Метрики (offline)

Groundedness

0.94



Answer Relevance

0.93



Context Precision

0.91



Faithfulness

0.93



Всё выглядит отлично

- Реальные запросы
- Шумные формулировки
- Запросы про спикеров
- No-answer сценарии
- Distribution shift

Проблемы в продакшене



Рост no-answer запросов



Падение groundedness



Слайки ошибок на новых темах



Жалобы пользователей

Метрики

Recall@K

Precision@K

MRR

1. Что измеряет Recall@K



Доля ожидаемых докладов, которые попали в top-k

$$\text{Recall@K} = \frac{|\text{expected} \cap \text{retrieved}[:K]|}{|\text{expected}|}$$

- **expected** = какие доклады мы ожидали найти
- **retrieved[:K]** = что реально вернул retriever в top-k

2. Как считаем в проекте

```
def recall_at_k(retrieved_ids,
               expected_ids, k=5):

    if not expected_ids:
        return 1.0

    retrieved_set = set(retrieved_ids[:k])

    found = sum(1 for eid in expected_ids
               if eid in retrieved_set)

    return found / len(expected_ids)
```

3. Пример по программе



Вопрос:

Какие доклады про AI и тестирование?

Expected (эталон)

1	LLM'изация тестирования в Яндексе	✓
2	AI-агенты	✓
3	DeepEval	✓
4	ИИ-копилоты	✓
5	RAG-ассистент	✗

Retrieved top-5

1	LLM'изация тестирования в Яндексе	✓
2	AI-агенты	✓
3	DeepEval	✓
4	ИИ-копилоты	✓
5	Управление контекстом в автотестах	



Нашли 4 из 5



Recall@5 = 0.8

1. Что измеряет Precision@5



Доля релевантных документов среди **top-5** результатов

$$\text{Precision@K} = \frac{|\text{expected} \cap \text{retrieved}[:K]|}{K}$$

- чем выше precision, тем **меньше** шумных chunks в выдаче
- это метрика **чистоты retrieval**

2. Как считаем в проекте

```
def precision_at_k(retrieved_ids,
                  expected_ids, k=5):

    top_k = retrieved_ids[:k]

    if not top_k:
        return 0.0

    expected_set = set(expected_ids)

    relevant = sum(1 for rid in top_k
                  if rid in expected_set)

    return relevant / len(top_k)
```

3. Пример по программе



Вопрос:

Доклады про автоматизацию тестирования

Retrieved top-5

1	Управление контекстом в автотестах	✓	релевантные (3 из 5)
2	Агентные системы для мобильного регресса	✓	
3	Как не превратить автотесты в объект ненависти	✓	
4	Нерелевантный доклад	✗	шум (2 из 5)
5	Нерелевантный доклад	✗	



Precision@5 = 3 / 5 = 0.6

1. Что измеряет MRR



Обратный ранг первого релевантного результата

$$\text{MRR} = \frac{1}{\text{rank}(\text{first relevant})}$$

- если первый релевантный на 1 месте → MRR = 1.0
- если на 2 месте → 0.5
- если на 5 месте → 0.2
- если релевантного нет → 0



Чем выше MRR, тем выше шанс, что нужный документ попадёт в контекст модели

2. Как считаем в проекте

```
def mrr(retrieved_ids, expected_ids):  
    expected_set = set(expected_ids)  
  
    for i, rid in enumerate(retrieved_ids):  
        if rid in expected_set:  
            return 1.0 / (i + 1)  
  
    return 0.0
```

3. Пример по программе



A Вопрос: Какие доклады про AI и тестирование?

- 1 LLM'изация тестирования в Яндексе ✓
- 2 Управление контекстом в автотестах
- 3 Агентные системы для мобильного регресса
- 4 Как не превратить автотесты в объект ненависти
- 5 Мониторинг качества тестов в CI

→ первый релевантный на 1 месте

MRR = 1.0

B Вопрос: Какие доклады во второй день про безопасность?

- 1 Обзор инструментов тестирования
- 2 Безопасность в пайплайнах разработки ✓
- 3 Сканирование зависимостей
- 4 Threat modeling на практике
- 5 Управление секретами в CI/CD

→ первый релевантный на 2 месте

MRR = 0.5

C Вопрос: Что расскажет Владислав Миронов?

- 1 Открытие конференции Heisenbug
- 2 Ключевые тренды в тестировании
- 3 Управление контекстом в автотестах
- 4 Мониторинг тестовой инфраструктуры
- 5 Репортаж: как прошёл Heisenbug 2024

релевантного результата нет

MRR = 0.0

Пример запроса



Какие доклады про AI и тестирование?

Retrieval: top-5 результатов

1 LLM'изация тестирования в Яндексе ✓

2 Как автоматизировать тестирование ИИ-агентов ✓

3 Тестирование LLM-приложений с DeepEval ✓

4 Тестирование ИИ-копилотов для разработки ✓

5 Управление контекстом в автотестах с помощью ИИ ✗

Expected set (ожидаемые AI-доклады)

✓ LLM'изация тестирования в Яндексе

✓ Как автоматизировать тестирование ИИ-агентов

✓ Тестирование LLM-приложений с DeepEval

✓ Тестирование ИИ-копилотов для разработки

✗ RAG-ассистент для инженеров (на программе не найден)

не найден в top-5



релевантный (ожидаемый)



нерелевантный (шум)



ожидаемый, но не найден



Recall@5

Сколько нужных нашли

$\frac{|\text{expected } n \text{ retrieved}[:5]|}{|\text{expected}|}$

4 из 5 → **0.8**



Precision@5

Насколько top-5 чистый

$\frac{|\text{expected } n \text{ retrieved}[:5]|}{5}$

5

4 из 5 → **0.8**



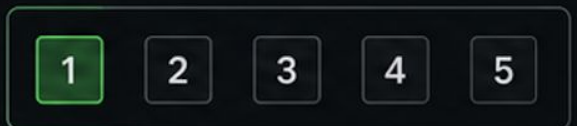
MRR

Насколько рано нашли первый правильный

1

$\text{rank}(\text{first relevant})$

первый релевантный на 1 месте → **1.0**



1. Что получает judge



input question

Какие доклады про AI и тестирование?



retrieval_context (top-5)

1 chunk 1: AI-агенты ✓

2 chunk 2: LLM'изация тестирования ✓

3 chunk 3: DeepEval ✓

4 chunk 4: ИИ-копилоты ✓

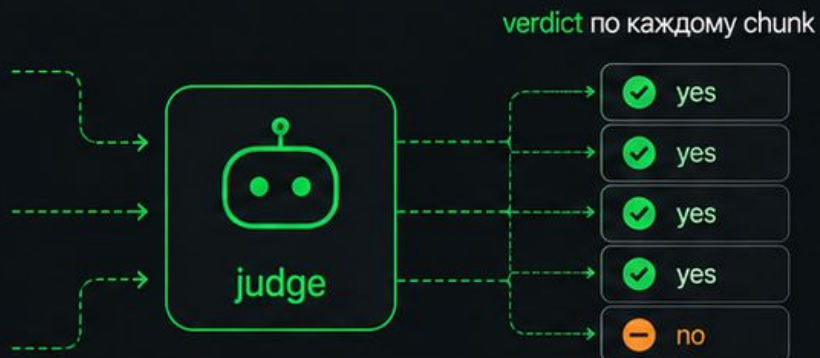
5 chunk 5: частично нерелевантный chunk ✗



expected_output

Подобрать несколько релевантных AI-докладов без шума

2. Как считает DeepEval



Judge-driven ranked precision

score = average precision по judge-verdicts

```
relevant_nodes_count = 0
sum_weighted_precision_at_k = 0.0

for k, is_relevant in enumerate(node_verdicts, start=1):
    if is_relevant:
        relevant_nodes_count += 1
        precision_at_k = relevant_nodes_count / k
        sum_weighted_precision_at_k += precision_at_k

score = sum_weighted_precision_at_k / relevant_nodes_count
```

`deepeval.metrics.contextual_precision`

3. Как используем в проекте

```
ctx_prec = ContextualPrecisionMetric(
    threshold=0.5,
    model=judge,
    async_mode=False,
)
```

`src/evaluation/deepeval_evaluator.py`



включаем ТОЛЬКО в extended metrics



нужен `expected_output`



полезно для оценки чистоты `retrieval_context`



помогает понять, много ли шума в `top-K`

1. Что получает judge



input question

Что происходит в первый день с утра: открытие и кейноут?



retrieval_context (top-K chunks)

1 opening chunk: информация об открытии конференции ✓

2 утренний chunk (частично): несколько утренних докладов, но без кейноута ✓

3 missing keynote chunk: информации о кейноуте нет в контексте ✗



expected_output

Назвать открытие и кейноут первого дня, показать утренние слоты

2. Как считает DeepEval

expected_output

Назвать открытие и кейноут первого дня, показать утренние слоты

разбиваем на ожидаемые смысловые элементы

- 1 Информация об открытии конференции
- 2 Информация о кейноуте первого дня
- 3 Утренние слоты (другие доклады)



judge

verdicts

✓ yes

✓ yes

✗ no



Contextual Recall = полнота покрытия ожиданий

$score = \text{justified_sentences} / \text{number_of_verdicts}$

```
justified_sentences = 0
for verdict in verdicts:
    if verdict.verdict.lower() == "yes":
        justified_sentences += 1
score = justified_sentences / number_of_verdicts
```

`deepeval.metrics.contextual_recall`

3. Как используем в проекте

```
ctx_rec = ContextualRecallMetric(
    threshold=0.5,
    model=judge,
    async_mode=False,
)
```

`src/evaluation/deepeval_evaluator.py`



включаем только в **extended metrics**



нужен **expected_output**



полезно для оценки полноты **retrieval_context**



особенно важно для **recommendation** и **schedule cases**



Обычные retrieval-метрики

Recall@3

Recall@5

Precision@5

MRR

Сравниваем `retrieved_ids`
с `expected_talk_ids`

$$\text{Recall@K} = |\text{expected} \cap \text{retrieved[:K]}| / |\text{expected}|$$

$$\text{Precision@K} = |\text{expected} \cap \text{retrieved[:K]}| / K$$

$$\text{MRR} = 1 / \text{rank}(\text{first relevant})$$

- ✓ deterministic
- ✓ стабильные
- ✓ хороши для regression
- ✓ меряют **exact** retrieval



VS



DeepEval retrieval-like metrics

contextual_precision

contextual_recall

Judge сравнивает
`retrieval_context` с `expected_output`

retrieval_context
(chunks / passages)

expected_output
(что нужно покрыть)

question
(входной запрос)








LLM judge

score
0.0 – 1.0

- ✓ semantic / judge-based
- ✓ гибче
- ✓ ближе к **perceived usefulness** context
- ✓ менее стабильны, чем exact metrics

Сравнение подходов к оценке retrieval

Критерий	Обычные retrieval-метрики (детерминированные)	DeepEval context-метрики (judge-based)
 Что лежит в основе	На основе идентификаторов (ID)	На основе полезности контекста
 Тип соответствия	Точное (идентичное совпадение)	Семантическое (смысловая оценка)
 Метод	Детерминированный (алгоритмический)	Судейская оценка моделью (LLM judge)
 Что оцениваем	Ранжирование и покрытие релевантных ID	Достаточность и полнота контекста для ожидаемого результата
 Стабильность	Высокая (стабильные)	Ниже (зависит от модели и промптов)
 Использование	Регрессия, мониторинг, алерты	Оценка качества контекста для ответа пользователю
 Чувствительность	К ранжированию и составу top-K	К смысловому покрытию ожиданий
 Идеально для	Оценки качества поиска по документам (точность и полнота)	Оценки полезности и достаточности контекста для ответа

Метрики

Faithfulness

Answer Relevancy

Hallucination

Кастомные метрики



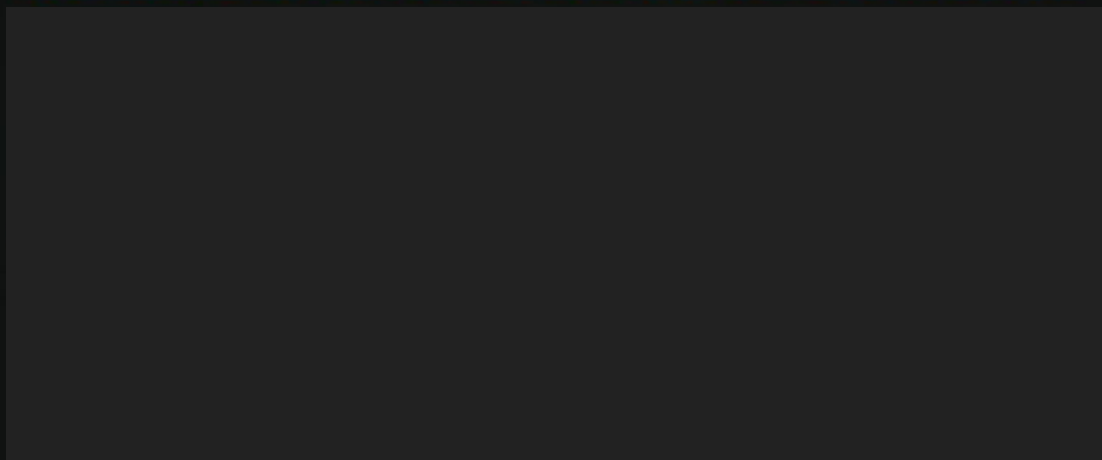
Faithfulness



Answer Relevancy



Hallucination





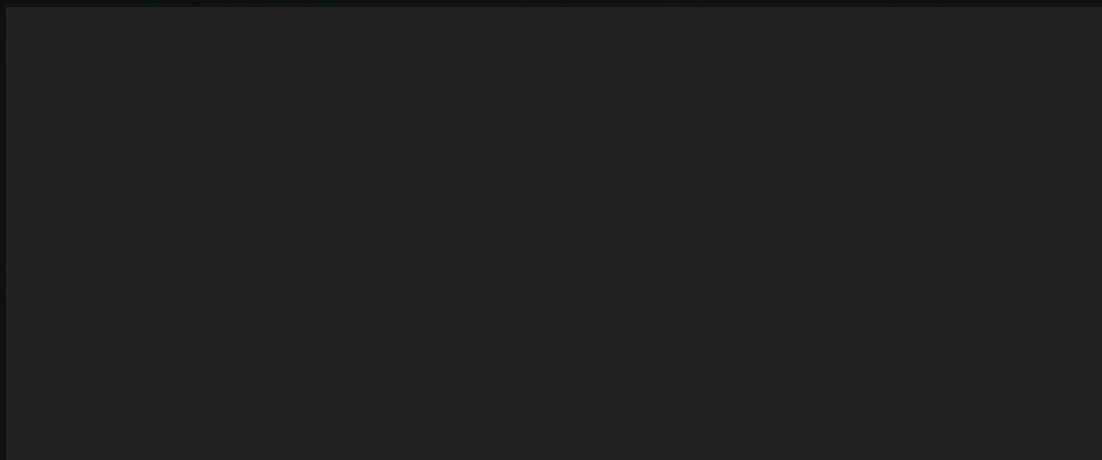
Faithfulness



Answer Relevancy



Hallucination



Applied Practicality



Technical Depth



Grounding Transparency



Activity Disambiguation

1. Входные данные



Вопрос

Есть доклады про геймдев и тест-дизайн в играх?



Retrieval context



Игры по правилам: тест-дизайн в игре хаоса геймдева

- Спикер: Анна Петрова
- Время: 11:30 – 12:20
- Зал: Зелёный зал

Доклад о тест-дизайне в игровых проектах: как работать с неопределённостью, выявлять скрытые механики и строить эффективные тест-кейсы в условиях хаоса и ограничений.

2. Как работает Faithfulness



Сгенерированный ответ

Да, в программе есть доклад «Игры по правилам: тест-дизайн в мире хаоса геймдева». Его проведёт Анна Петрова с 11:30 до 12:20 в Зелёном зале. Доклад о тест-дизайне в игровых проектах и работе в условиях хаоса и ограничений.



Фактический ответ
(actual_output)



Выделение утверждений
(claims)



Судья (LLM) сравнивает с retrieval context
(judge vs context)

Утверждения (claims)



В программе есть доклад «Игры по правилам: тест-дизайн в мире хаоса геймдева»



Спикер доклада — Анна Петрова



Время доклада — с 11:30 до 12:20



Доклад в Зелёном зале



Доклад о тест-дизайне в игровых проектах и работе в условиях хаоса и ограничений

Вердикт

Подтверждено

Подтверждено

Подтверждено

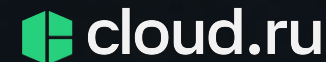
Подтверждено

Подтверждено



$$\text{Faithfulness} = \frac{\text{подтвержденные утверждения}}{\text{все утверждения}}$$

3. Результат



Faithfulness

1.0



Все ключевые утверждения в ответе подтверждаются retrieval-контекстом.

1. Входные данные



Вопрос

Доклады про автоматизацию тестирования



Ожидаемый интент (expected intent)

- ✓ automation
- ✓ несколько релевантных докладов
- ✓ без ухода в побочные темы

2. Как работает Answer Relevancy



Пример ответа системы

- Управление контекстом в автотестах с помощью ИИ
- Агентные системы для мобильного регресса
- Как не превратить автотесты в хрупкий легаси
- **Использование метавселенных в образовании**
- **Психология команды тестирования**



Фактический ответ системы (actual_output)



Выделение утверждений (statements)



Судья (LLM) сравнивает с вопросом (judge vs input)

Утверждения (statements)	Вердикт (релевантность к вопросу)
✓ Управление контекстом в автотестах с помощью ИИ	Релевантно
✓ Агентные системы для мобильного регресса	Релевантно
✓ Как не превратить автотесты в хрупкий легаси	Релевантно
− Использование метавселенных в образовании	Нерелевантно
− Психология команды тестирования	Слабо релевантно



Answer Relevancy =

$$\frac{\text{релевантные утверждения}}{\text{все утверждения}}$$

3. Результат

Итоговая оценка
answer_relevancy

0.294



Что это значит

Ответ частично отвечает на вопрос: есть релевантные доклады, но есть и нерелевантные/слабые утверждения.



Важно

- Faithfulness может быть высоким
- Но Answer Relevancy низкий, потому что часть ответа не отвечает на исходный вопрос пользователя

1. Входные данные



Вопрос

Что происходит в первый день с утра: открытие и кейноут?



Retrieval context (неполный)

Открытие конференции

- 09:00 – 09:30 — Регистрация участников
- 09:30 – 10:00 — Торжественное открытие конференции

Утренние элементы

- 10:00 – 10:45 — Доклад: Тренды в тестировании 2025
- 10:45 – 11:15 — Кофе-брейк



Контекст неполный: в нем нет информации о кейноуте первого дня.

2. Как работает Hallucination



Пример ответа системы

Утром первого дня проходит открытие конференции, затем доклад «Тренды в тестировании 2025», а после — кейноут «Будущее качества: ИИ и автотесты» от Алексея Иванова.
Далее — сессия вопросов и ответов с докладчиком.



Фактический ответ системы (actual_output)



Выделение фактических утверждений (factual assertions)



Судья (LLM) сравнивает с retrieved context (judge vs context)

Фактические утверждения	Вердикт (поддерживается контекстом?)
✔ Утром проходит открытие конференции	Поддерживается
✔ После открытия — доклад «Тренды в тестировании 2025»	Поддерживается
✔ После доклада — кофе-брейк	Поддерживается
✖ Есть кейноут «Будущее качества: ИИ и автотесты»	Не подтверждается
✖ Кейноут проводит Алексей Иванов	Не подтверждается
✖ Далее — сессия вопросов и ответов с докладчиком	Не подтверждается



Hallucination = $\frac{\text{неподтвержденные утверждения}}{\text{все утверждения}}$

3. Результат

Итоговая оценка hallucination

0.6



Что это значит

Ответ содержит факты, которых нет в retrieved context. Часть информации выдумана или не подтверждается имеющимся контекстом.



Важно

- Ответ звучит уверенно и логично.
- Но часть утверждений не опирается на retrieved context.

1. Входные данные



Вопрос

Какие доклады по теме AI есть во второй день?



Практический intent пользователя



Подобрать варианты докладов по теме AI



Спланировать второй день конференции



Получить next step — что смотреть дальше и почему



Цель: получить прикладную, готовую к использованию информацию, а не просто правильный ответ.

2. Пример ответа системы и критерии judge



Ответ системы (готов к использованию)

1

AI-агенты в тестировании: от идеи к продакшену

Анна Петрова 10:00 – 10:45 Зал: Альфа

[Ссылка на программу](#)

2

LLM в QA: практики и подводные камни

Иван Смирнов 11:00 – 11:45 Зал: Бета

[Ссылка на программу](#)

3

AI для автотестов: ускоряем регрессионное тестирование

Мария Иванова 14:00 – 14:45 Зал: Гамма

[Ссылка на программу](#)



Рекомендация: начните с доклада 1, затем переходите к 2 и 3. Они логично дополняют друг друга — от архитектуры к практике.



Критерии judge (Applied Practicality)

- ✓ Ответ опирается на реальные пункты программы (доклады, спикеры, время, зал)
- ✓ Рекомендации понятны и применимы для планирования дня
- ✓ Нет выдуманных залов, времени, спикеров или ссылок

3. Результат оценки

Итоговая оценка
applied_practicality



HIGH
(высокая)

Ответ можно использовать сразу.
Пользователь знает, что смотреть дальше.



Полезно

Ответ помогает решить практическую задачу



Действенно

Можно сразу применить для планирования



Обосновано контекстом

Полностью опирается на актуальную программу конференции

1. Входные данные



Вопрос

Посоветуй технические доклады по automation во второй день



Ожидаемый стиль ответа



Конкретика

Чёткие названия, спикеры, время, зал, темы докладов



Инженерные термины

autotest, CI, mutation testing, инфраструктура, регресс, flaky, observability



Содержательность

Ответ должен давать инженерную ценность, а не просто перечислять



Цель

Получить рекомендации, которые помогут инженеру выбрать и спланировать день.

2. Пример хорошего ответа и критерии judge



Пример ответа системы (готов к использованию)

1

Mutation testing в реальном проекте: опыт внедрения и результаты

Анна Петрова 11:00 – 11:45 Зал: Бета

Как мы внедрили mutation testing в CI на pytest + Stryker, покрытие мутаторами, пайплайн в GitLab CI, снижение escaped defects на 32%.

mutation testing CI pytest Stryker quality gates

2

Автотесты на UI: от стабильности к скорости

Иван Смирнов 14:00 – 14:45 Зал: Гамма

Оптимизация E2E на Playwright: параллелизация, shard-инг, retry-стратегии, борьба с flaky-тестами, ускорение прогонов в 3.5 раза.

Playwright E2E параллелизация flaky оптимизация

3

Инфраструктура для автотестов: масштабирование без боли

Мария Иванова 16:00 – 16:45 Зал: Альфа

Архитектура распределённой инфраструктуры для автотестов в Kubernetes, service virtualization, observability и отчётность.

инфраструктура Kubernetes service virtualization observability



Критерии judge (Technical Depth)



Использует уместную инженерную терминологию



Даёт достаточную техническую конкретику и детали



Избегает лишней «воды» и общих фраз



Соответствует техническому уровню запроса



Пример поверхностного ответа

Вот несколько докладов во второй день:

1. Про автотесты
2. Про инфраструктуру
3. Про качество кода

...



Пример технического ответа

Содержит термины, детали, инструменты, подходы, ожидаемые результаты и практическую пользу для инженера.

3. Результат оценки

Итоговая оценка
technical_depth



HIGH

(высокая)

Ответ технически насыщенный, конкретный и полезный для инженера.



Не просто релевантный

Ответ выходит за рамки формального соответствия теме.



Действительно инженерный

Содержит нужные термины, подходы и технические детали.



Практическая ценность

Помогает инженеру принять решение и спланировать день.

1. Входные данные



Вопрос

Расскажи про доклад про Kubernetes и service mesh на Heisenbug



Retrieval context

Подходящего доклада нет в извлечённом контексте.

Есть только близкие, но не те темы:

- Контейнеризация в продакшене без Kubernetes
- Сетевые политики в микросервисах
- Observability в распределённых системах
- Service discovery и балансировка нагрузки



Контекст не содержит доклада про Kubernetes и service mesh на Heisenbug.

2. Ответ ассистента и критерии judge



Ответ ассистента (пример хорошего ответа)

К сожалению, в доступном контексте нет информации о докладе про Kubernetes и service mesh на Heisenbug.

Я просмотрел извлечённые материалы и не нашёл упоминаний такого доклада.

Могу предложить близкие по теме доклады, связанные с контейнеризацией, сетевыми технологиями и архитектурой микросервисов — они могут быть полезны.



Критерии judge (Grounding Transparency)

- ✓ Не выдумывает несуществующие доклады, спикеров, время и залы
- ✓ Явно сообщает, что нужной информации нет в доступном контексте
- ✓ Не приписывает контексту то, чего в нём нет
- ✓ Использует осторожный и честный тон при отсутствии данных
- ✓ При необходимости предлагает безопасные альтернативы (близкие по теме)

✗ Нехороший ответ (галлюцинация)

«На Heisenbug будет доклад про Kubernetes и service mesh от Ивана Петрова в 14:00 в зале Альфа...»

vs

✓ Хороший ответ (прозрачность)

«Такого доклада нет в доступном контексте. Вот близкие темы, которые могут быть интересны...»

3. Результат оценки

Итоговая оценка
grounding_transparency

HIGH
(высокая)



Ответ честно отражает ограничения доступного контекста.



Это не значит, что другие метрики (например, полнота или практическая полезность) тоже высокие.



Честный отказ — это правильное поведение с точки зрения judge.

Важно различать

Нет данных

Честно сообщаем об ограничениях

Галлюцинация

Уверенно выдаём неподтверждённые факты

≠

1. Входные данные



Вопрос

Какие активности на стенде Ozon Tech на конференции?



Ожидаемый intent



Нужна активность партнёра (стенд Ozon Tech)



Не нужен доклад в зале (не про расписание)



Нужен корректный формат ответа: форматы, время, что можно получить/сделать

Важно различать

Доклад в зале



часть расписания конференции

≠

Активность на стенде



активность партнёра в зоне стендов

2. Пример хорошего ответа



Активности на стенде Ozon Tech

На конференции у Ozon Tech будет стенд с активностями для участников. Это не доклады в залах, а интерактивные форматы в зоне стендов.

Что можно сделать на стенде Ozon Tech:



Пообщаться с инженерами

Обсудите продукты, архитектуру и карьерные возможности.



Демозона

Посмотрите live-демо наших решений и инструментов.



Мини-квест

Пройдите короткий квест и получите мерч.



Розыгрыши и призы

Участвуйте в розыгрышах в течение дня.



Место: зона стендов

Время: оба дня конференции



Критерии judge (Activity Disambiguation)



Не путает доклад в зале и активность партнёра на стенде



Не выдаёт стендовые активности как доклады из расписания



Честно разделяет форматы и корректно описывает стендовые активности

3. Результат оценки

Итоговая оценка

activity_disambiguation

0.99

из 1.00



ОТЛИЧНО

(высокая оценка)

Ассистент корректно различил стендовую активность партнёра и не подменил её докладами из расписания.

Контраст для наглядности

Неправильно (подмена)



Выдал доклад из расписания вместо стендовой активности.

Правильно (различение)



Описал активности на стенде партнёра корректно.



1. Faithfulness

ВХОДНЫЕ ДАННЫЕ



actual_output
(ответ модели)



retrieval_context
(извлечённый контекст)

СХЕМА ОЦЕНКИ



ФОРМУЛА

$$\text{Faithfulness} = \frac{\text{supported_claims}}{\text{all_claims}}$$

КОД (DEEPEVAL INTERNALS)

```
score = faithfulness_count / number_of_verdicts
```



2. Answer Relevancy

ВХОДНЫЕ ДАННЫЕ



question
(вопрос пользователя)



actual_output
(ответ модели)

СХЕМА ОЦЕНКИ



ФОРМУЛА

$$\text{Answer Relevancy} = \frac{\text{relevant_statements}}{\text{all_statements}}$$

КОД (DEEPEVAL INTERNALS)

```
score = relevant_count / number_of_verdicts
```



3. Hallucination

ВХОДНЫЕ ДАННЫЕ



actual_output
(ответ модели)



retrieval_context
(контекст / факты)

СХЕМА ОЦЕНКИ



ФОРМУЛА

$$\text{Hallucination} = \frac{\text{unsupported_assertions}}{\text{all_assertions}}$$

КОД (DEEPEVAL INTERNALS)

```
score = hallucination_count / number_of_verdicts
```



Метрики подключаются так

```
faithfulness = FaithfulnessMetric(threshold=0.7, model=judge, async_mode=False)
```

```
relevancy = AnswerRelevancyMetric(threshold=0.7, model=judge, async_mode=False)
```

```
hallucination = HallucinationMetric(threshold=0.5, model=judge, async_mode=False)
```

Общая схема GEval



ФОРМУЛА НОРМАЛИЗАЦИИ

$$\text{normalized_score} = (\text{g_score} - \text{score_range_start}) / \text{score_range_span}$$



По умолчанию
извапзон 0..10



Judge возвращает
score + reason



В проекте
threshold = 0.5

КОД

```
GEval(  
    name="applied_practicality",  
    criteria="...",  
    evaluation_params=params,  
    model=m,  
    threshold=0.5,  
    async_mode=False,  
)
```

```
params = [  
    LLMTestCaseParams.INPUT,  
    LLMTestCaseParams.ACTUAL_OUTPUT,  
    LLMTestCaseParams.RETRIEVAL_CONTEXT,  
]
```



1. Applied Practicality

КРИТЕРИИ

- ✓ есть опора на реальные пункты программы
- ✓ понятные next steps
- ✓ нет выдуманных деталей расписания



2. Technical Depth

КРИТЕРИИ

- ✓ уместная техническая терминология
- ✓ достаточная конкретика
- ✓ без лишней воды



3. Grounding Transparency

КРИТЕРИИ

- ✓ не приписывает контексту лишнего
- ✓ честно говорит об ограничениях
- ✓ не скрывает отсутствие данных



4. Activity Disamiguation

КРИТЕРИИ

- ✓ отличает talk от sponsor_activity
- ✓ не подменяет стенд докладом
- ✓ корректно разделяет форматы

Search

- Safety >
- Non-LLM >
- Images >
- Others >

Prompt Optimization

- Introduction
- Algorithms >

Synthetic Data Generation

- Introduction
- Generate Goldens >
- Conversation Simulator

Benchmarks

- Introduction
- Available Benchmarks >

Others

- CLI Settings
- Environment Variables
- Troubleshooting
- FAQ
- Data Privacy
- Miscellaneous

Generate Goldens

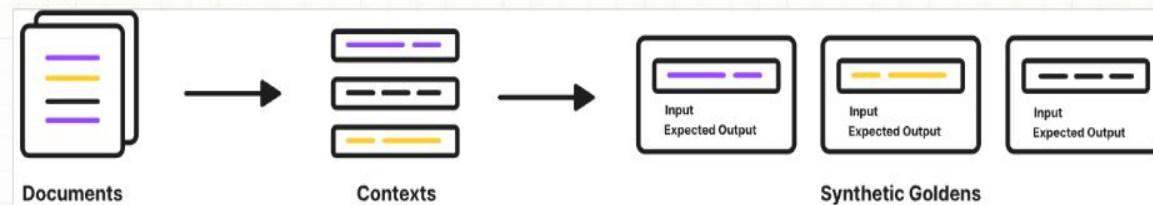
Generate Goldens From Documents

Copy Markdown

Open ▾

If your application is a Retrieval-Augmented Generation (RAG) system, generating Goldens from documents can be particularly useful, especially if you already have access to the **documents that make up your knowledge base**.

By simply providing these documents, `deepeval`'s Synthesizer will automatically handle generating the relevant contexts needed for synthesizing test Goldens.



💡 DID YOU KNOW?

The only difference between the `generate_goldens_from_docs()` and `generate_goldens_from_contexts()` method is `generate_goldens_from_docs()` involves an additional [context construction step](#).

On this page

Prerequisites

Generate Your Goldens

Customize Context Construction

How Does Context Construction Work?

Document Parsing

Context Selection

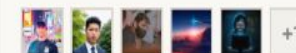


Collaborate in Confident Cloud

Review evals, traces, annotate, manage datasets, and version prompts.

Launch Platform

Contributors



Last updated on 26.04.2026

Find us on Github 🌟 15.0k

Join Community

Собираем датасет DeepEval

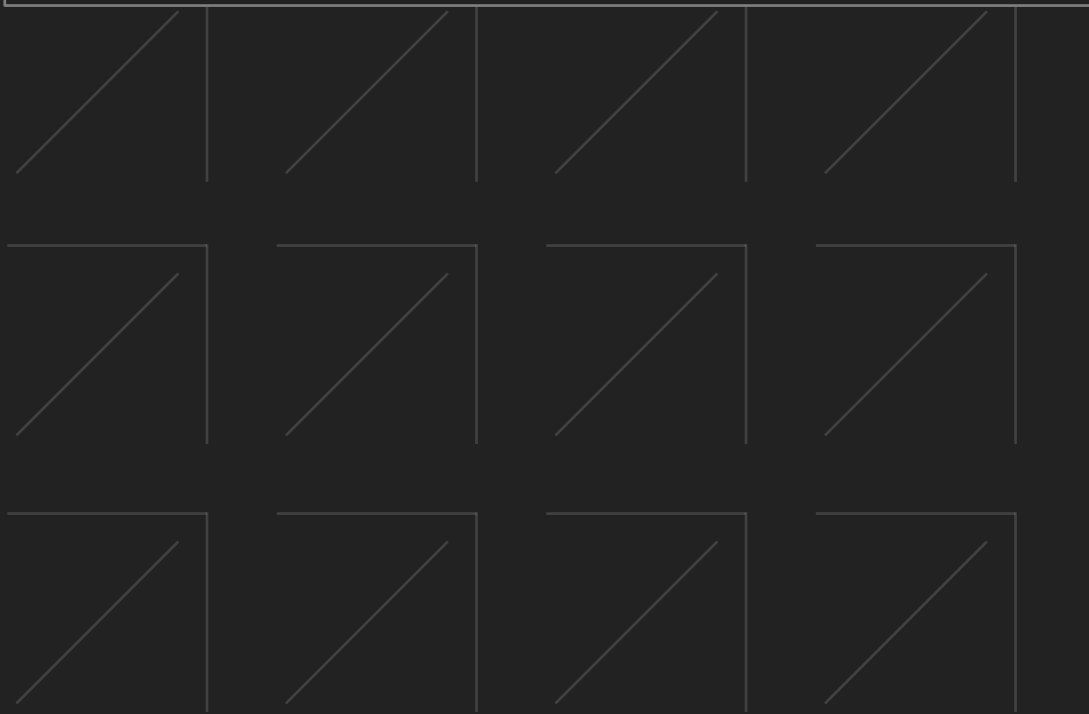
From Documents

From Contexts

From Scratch

From Goldens

DeepEval Симулятор



🔍 Search

Safety >

Non-LLM >

Images >

Others >

🦄 Prompt Optimization

Introduction

Algorithms >

📄 Synthetic Data Generation

Introduction

Generate Goldenes >

From Documents

From Contexts

From Scratch

From Goldenes

Conversation Simulator

🕒 Benchmarks

Introduction

Available Benchmarks >

🔗 Others

CLI Settings

Environment Variables

Conversation Simulator

📄 Copy Markdown

Open ▾

deepeval's `ConversationSimulator` allows you to simulate full conversations between a fake user and your chatbot, unlike the [synthesizer](#) which generates regular goldenes representing single, atomic LLM interactions.

📄 main.py

```

from deepeval.test_case import Turn
from deepeval.simulator import ConversationSimulator
from deepeval.dataset import ConversationalGolden

# Create ConversationalGolden
conversation_golden = ConversationalGolden(
    scenario="Andy Byron wants to purchase a VIP ticket to a cold play concert.",
    expected_outcome="Successful purchase of a ticket.",
    user_description="Andy Byron is the CEO of Astronomer.",
)

# Define chatbot callback
async def chatbot_callback(input):
    return Turn(role="assistant", content=f"Chatbot response to: {input}")

# Run Simulation
simulator = ConversationSimulator(model_callback=chatbot_callback)

```

☰ On this page

| Create Your First Simulator

Model callback

Simulate A Conversation

Advanced Usage

Lifecycle hooks

Existing turns

Evaluate Simulated Turns

**Collaborate in Confident Cloud**

Review evals, traces, annotate, manage datasets, and version prompts.

Launch Platform

👤 Contributors



Last updated on 26.04.2026

🔗 Find us on Github 🌟 15.0k

👤 Join Community

Что генерирует Simulator



Итог



Получаете готовые многоходовые тест-кейсы для оценки диалога end-to-end.

Дальше



Оценивайте их любыми conversational-метриками.

Пример

```
from deepeval.test_case import Turn
from deepeval.simulator import ConversationSimulator
from deepeval.dataset import ConversationalGolden

# 1. Описываем сценарий
golden = ConversationalGolden(
    scenario="Andy Byron хочет купить VIP-билет на концерт Coldplay",
    expected_outcome="Успешная покупка билета",
    user_description="Andy Byron — CEO компании Astronomer",
)

# 2. Колбэк вашего чат-бота
async def chatbot_callback(input: str, turns, thread_id: str) -> Turn:
    response = await your_chatbot(input, turns, thread_id)
    return Turn(role="assistant", content=response)

# 3. Запуск симуляции
simulator = ConversationSimulator(model_callback=chatbot_callback)
test_cases = simulator.simulate(
    conversational_goldens=[golden],
    max_user_simulations=10 # макс. число циклов user↔assistant
)

# 4. Результат — многоходовые тест-кейсы для оценки
print(test_cases[0].turns) # список turns диалога
```

Собираем датасет ~~DeepEval~~




ручками

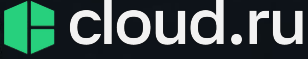
требования

эксперты

пользователи

GOLDEN DATASET

	golden_dataset_program.json программа конференции	v1.3.2
	golden_dataset_activities.json активности и стенды	v1.2.0
	golden_dataset_mixed_hard.json сложные multi-constraint вопросы	v1.1.0
	golden_dataset_adversarial.json out-of-scope / no-answer / adversarial	v1.0.1
	golden_dataset_rag_deepeval.json synthetic dataset через DeepEval	v1.0.0

Что содержит один вопрос в golden dataset 

```
{  
  "question": "Какие доклады по AI во второй день?",  
  "expected_talk_ids": ["talk_102", "talk_245"],  
  "expected_topics": ["AI", "LLM", "RAG"],  
  "difficulty": "medium",  
  "intent": "find_talks",  
  "expected_output": "Список докладов с названиями,  
  спикерами, временем, залом и ссылками.",  
  "tags": ["program", "ai", "day2"]  
}
```

Ключевые поля

- ✔ **expected_talk_ids** — эталонные ID докладов (если применимо)
- ✔ **expected_topics** — ожидаемые темы/теги
- ✔ **expected_output** — эталонный ответ или требования к ответу
- ✔ **difficulty** — уровень сложности: easy / medium / hard
- ✔ **intent** — намерение пользователя (например, find_talks, booth_info)
- ✔ **tags** — дополнительные теги для фильтрации и анализа



1. Topic / recommendation

- Какие доклады на Heisenbug 2026 про AI и машинное обучение в QA?
- что глянуть про мобилки и тестирование приложений?
- Есть ли что-то про безопасность во второй день?



2. Activities / sponsor

- Какие активности на стенде Ozon Tech на конференции?
- Что делает YADRO на стенде — какие форматы?
- Есть ли у партнёров активности про open source?



3. Mixed hard / multi-constraint







- Во второй день хочу углубиться в AI-агентов и LLM-as-a-Judge — что выбрать?
- Нужны и мобильный регресс, и истории про железо — собери трек на два дня
- Где в программе лучше пересечься людям с интересом к AI и mobile?



4. Adversarial / no-answer


- Забронируй мне отель рядом с площадкой и трансфер
- Какой номер телефона организаторов?
- Расскажи про доклад про Kubernetes и service mesh на Heisenbug

1. Основа корпуса

-  **talks**
доклады и описания
-  **activities**
активности и стенды
-  **topics**
темы и теги
-  **speakers**
спикеры
-  **days**
дни конференции
-  **talk_type**
тип доклада



2. Rule-based generation

-  **стратификация по topic-тегам**
покрытие тем равномерно
-  **выбор по day**
вопросы по дням и их комбинациям
-  **speaker_lookup**
запросы про спикеров и их доклады
-  **keyword / RAG cases**
поиск по ключевым словам и смыслу
-  **hard mixed queries**
мульти-constraint и комбинации условий
-  **adversarial / out-of-scope cases**
некорректные и вне области кейсы

индексируем корпус для правил





```
by_topic = build_index(talks, key="topics")
```

```
by_day = build_index(talks, key="day")
```

```
speaker_to_talks = build_speaker_index(talks)
```



3. Synthetic generation

-  **DeepEval Synthesizer**
используем LLM для генерации вопросов и ожиданий
-  **generate_goldens_from_contexts**
генерация на основе контента докладов, активностей, тегов
-  **expected_output generation**
формируем эталонные ответы и связанные сущности (talk_ids, topics и т.д.)
-  **synthetic natural-language questions**
естественные вопросы разных типов и уровней сложности

GOLDEN DATASET: ПОКРЫТИЕ

1



Intent coverage

intent types: recommendation / detail / schedule / comparison / speaker / activity

2



Topic coverage

topics: AI / Mobile / Automation / Security / ...

3



Constraint coverage

constraints: day / speaker / multi-constraint / no-answer

4



Difficulty coverage

difficulty: easy / medium / hard

5



Real-query coverage

realism: доля вопросов из реальных пользовательских формулировок



$$\text{Intent coverage} = \frac{\text{covered_intents}}{\text{required_intents}} * 100\%$$



$$\text{Topic coverage} = \frac{\text{topics_present}}{\text{required_topics}} * 100\%$$



$$\text{Constraint coverage} = \frac{\text{covered_constraint_types}}{\text{total_constraint_types}} * 100\%$$



$$\text{Difficulty coverage} = \text{balanced distribution across easy / medium / hard}$$



$$\text{Real-query coverage} = \frac{\text{real_queries_in_dataset}}{\text{total_questions}} * 100\%$$



$$\text{coverage_total} = 0.25 * \text{intent} * 0.25 * \text{topic} * 0.20 * \text{constraint} * 0.15 * \text{difficulty} * 0.15 * \text{realism}$$

GOLDEN DATASET: ВЕРСИОНИРОВАНИЕ

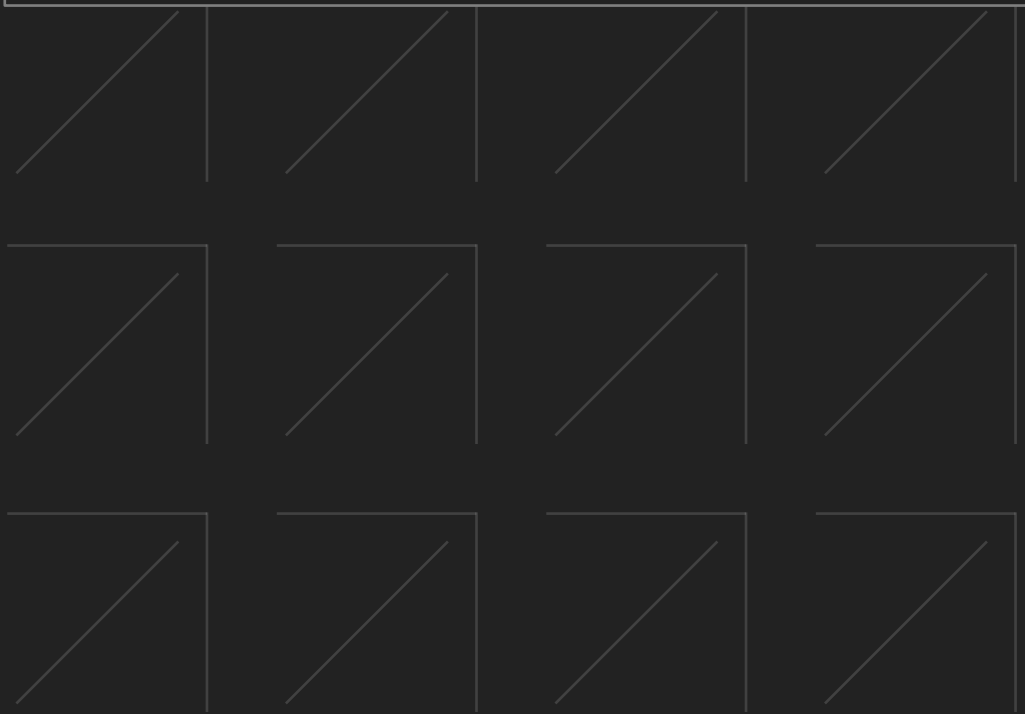


Сторонний эксперт



Кадр из м/ф «Вовка в Тридевятом царстве», реж. Б. Степанцев,
киностудия «Союзмультфильм», 1965 г.

Сторонний эксперт



Ошибается

Оценка зависит от критерия

Сторонний эксперт

Ошибается

Оценка зависит от критерия

Ускоряет

Покрывает большие объемы



1. Prompt sensitivity

оценка зависит от формулировки критерия



2. Variance

на похожих кейсах возможны разные баллы



3. Bias

модель может переоценивать гладкий текст и недооценивать краткий, но точный ответ



4. False confidence

judge может уверенно объяснять спорную или неверную оценку



Поэтому Judge LLM

нужно использовать вместе с:



golden dataset



retrieval metrics



ручным case review



version comparison

Плохой prompt

Пример:

“Оцени, хороший ли это ответ.
Учитывай качество, полезность и точность.”



слишком расплывчато



нет явных критериев



нет границ между score 0 и 1



Результат:

неустойчивая и субъективная оценка

Хороший prompt

Пример:

“Оцени, насколько ответ практически применим
для участника конференции.
Смотри:
1. есть ли конкретные рекомендации
2. есть ли полезные детали
3. избегает ли ответ общих фраз
4. опирается ли на факты из контекста
Верни score от 0 до 1”



критерии конкретны



есть операционализация



score становится объяснимым



Результат:

устойчивая и объяснимая оценка

Проверим?



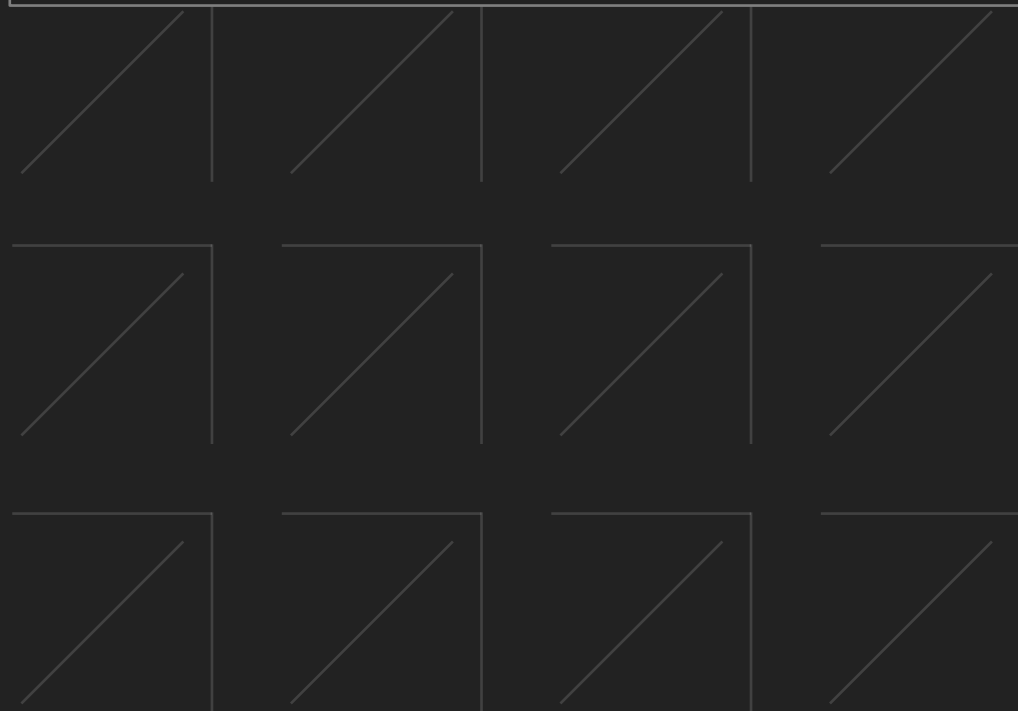
Кадр из м/ф «Вовка в Тридевятом царстве», реж. Б. Степанцев, киностудия «Союзмультфильм», 1965 г.

Проверим?

Хороший ответ

Неполный ответ

Плохой ответ





Хороший ответ + хорошие оценки

ВОПРОС

Какие доклады про AI и тестирование?

ОТВЕТ АССИСТЕНТА (ФРАГМЕНТ)



LLM'изация тестирования в Яндексе:
измеряем эффект от AI в команде из 1000+ QA-инженеров
Спикер: Владислав Миронов — Яндекс



Тестирование ИИ-копилотов для разработки
Спикер: Авенир Воронов — Veai



Как автоматизировать тестирование ...
Спикер: ...



Подобрано минимум 3 релевантных доклада по теме AI и тестирования, с названиями и спикерами. Без нерелевантных тем.

ОЦЕНКИ

	Faithfulness	-----	1.000
	Answer Relevancy	-----	1.000
	Hallucination (чем ниже, тем лучше)	-----	0.000
	Contextual Precision	-----	1.000
	Contextual Recall	-----	1.000
	GEval: Applied Practicality	-----	0.994
	GEval: Technical Depth	-----	0.936
	GEval: Grounding Transparency	-----	1.000
	GEval: Activity Disambiguation	-----	0.953



Какие доклады про **AI** и **тестирование**?



Practicality

0.994



Technical Depth

0.936



Grounding
Transparency

1.000



Activity
Disambiguation

0.953



ВОПРОС

Есть доклады про геймдев и тест-дизайн в играх?

КРИТЕРИИ (ОЖИДАЕТСЯ)

- expected_talk_ids: ['36fb78c7fd541f28fe442c4304caf45']
- expected_topics: ['GameDev']
- Нужно подтвердить наличие доклада про геймдев
- Нужно назвать доклад: «Игры по правилам: тест-дизайн в мире хаоса геймдева»

ОТВЕТ АССИСТЕНТА (ФРАГМЕНТ)

Игры по правилам: тест-дизайн в мире хаоса геймдева

Спикер: Алексей Симкин
Belarus TDC

2026-04-27, 10:30-11:15, Зал 2

Доклад посвящен систематизации подхода к тестированию в геймдеве и использованию методов тест-дизайна для игровых механик.

Точный match по intent, теме и ожидаемому докладу

ОЦЕНКИ

	Faithfulness	1.000
	Answer Relevancy	1.000
	Hallucination	0.000
	Contextual Precision	1.000
	Contextual Recall	1.000
	GEval: Applied Practicality	0.920
	GEval: Technical Depth	0.883
	GEval: Grounding Transparency	0.992
	GEval: Activity Disambiguation	0.775



Точный, но неполный ответ



ВОПРОС

доклады про автоматизацию тестирования



КРИТЕРИИ (ОЖИДАЕТСЯ)

- expected_talk_ids: 5 докладов
- expected_topics: ['Automation']
- Нужно перечислить несколько докладов по automation (CI, автотесты, регресс), без ухода в нерелевантные темы.



ОТВЕТ АССИСТЕНТА (ФРАГМЕНТ)

1

Как не превратить автотесты в объект ненависти: ред флаги в инфраструктуре

Спикеры: Иван Левиков; Даниил Смирнов — Ведущий инженер по автоматизации тестирования — Ozon Банк
День 1, 2026-04-27, 09:15–10:00, Зал: 2

[Подробнее](#)

2

Хаос-тестирование — от идеи до практического внедрения

Спикеры: Александр Фадеев — TBD — Т-Банк; Иван Горбацевич — TBD — Т-Банк
День 2, 2026-04-28, 13:45–14:30, Зал: 2

[Подробнее](#)

...

ОЦЕНКИ



Faithfulness ----- 1.000



Answer Relevancy ----- 0.750



Hallucination (чем ниже, тем лучше) ----- 0.600



Contextual Precision ----- 0.756



Contextual Recall ----- 0.600



GEval: Applied Practicality ----- 0.891



GEval: Technical Depth ----- 0.794



GEval: Grounding Transparency ----- 0.954



GEval: Activity Disambiguation ----- 0.837



ВОПРОС

Что происходит в первый день с утра: открытие и кейноут?



КРИТЕРИИ (ОЖИДАЕТСЯ)

- expected_talk_ids: 4 ожидаемых источника
- Нужно назвать открытие и кейноут первого дня
- Нужно показать утренние слоты
- Нельзя подменять другими форматами



ОТВЕТ АССИСТЕНТА (ФРАГМЕНТ)

1. Открытие конференции Heisenbug 2026 Spring
07:30-07:45, Зал 1
2. Чем аналитик поможет тестировщику,
а тестировщик — аналитику
09:15-11:15



ОЦЕНКИ



Faithfulness 1.000



Answer Relevancy 0.700



Hallucination 0.600



Contextual Precision 0.700



Contextual Recall 0.667



GEval: Applied Practicality 0.857



GEval: Technical Depth 0.793



GEval: Grounding Transparency 0.854



GEval: Activity Disambiguation 0.553

✘ Провал, плохой ответ



ВОПРОС

что расскажет Владислав Миронов?



КРИТЕРИИ (ОЖИДАЕТСЯ)

- expected_talk_ids: ['48ab465c0a7844f5b79551b6bb015816']
- expected_topics: []
- Нужно назвать доклад Владислава Миронова и кратко описать тему AI в тестировании на масштабе.



ОТВЕТ АССИСТЕНТА (ФРАГМЕНТ)

“ К сожалению, в предоставленном контексте нет информации о Владиславе Миронове и его докладе.

ОЦЕНКИ



Faithfulness ----- 1.000



Answer Relevancy ----- 0.000



Hallucination (чем ниже, тем лучше) ----- 1.000



Contextual Precision ----- 0.333



Contextual Recall ----- 1.000



GEval: Applied Practicality ----- 0.626



GEval: Technical Depth ----- 0.410



GEval: Grounding Transparency ----- 1.000



GEval: Activity Disambiguation ----- 0.549



ВОПРОС

Расскажи про доклад про Kubernetes и service mesh на Heisenbug



КРИТЕРИИ (ОЖИДАЕТСЯ)

- expected_talk_ids: []
- expected_topics: []
- Если такого доклада нет в программе, нужно честно сказать об этом
- Нужно предложить близкие темы, а не просто оборвать ответ



ОТВЕТ АССИСТЕНТА (ФРАГМЕНТ)

“ Извините, но информации о докладе про Kubernetes и service mesh на конференции Heisenbug 2026 Spring нет.



Ответ слишком слабый: нет полезного продолжения, нет предложения близких тем



ОЦЕНКИ



Faithfulness 0.000



Answer Relevancy 0.000



Hallucination 1.000



Contextual Precision 0.000



Contextual Recall 1.000



GEval: Applied Practicality 0.270



GEval: Technical Depth 0.357



GEval: Grounding Transparency 0.994



GEval: Activity Disambiguation 0.534



Начало 12:45:32 | Длительность 6.21s | Модель gpt-4.1 | Пользователь @user_123 | Сессия 9c1e2... | Версия prod-v2.14.3

Трейс	События	Время	Длительность	Статус
01	user_message Пользователь отправил сообщение в Telegram	12:45:32	120ms	✓
02	rewrite_with_history Учет контекста диалога, переформулировка запроса	12:45:32	310ms	✓
03	intent_parsing Определение интента и извлечение фильтров	12:45:32	180ms	✓
04	retrieval_qdrant Векторный поиск (top-k)	12:45:32	690ms	✓
05	reranking Переранжирование результатов	12:45:33	420ms	✓
06	enrichment Дозагрузка связанных докладов и longDescription	12:45:33	610ms	⚠
07	prompt_assembly Формирование финального промпта	12:45:34	210ms	✓
08	llm_generation Генерация ответа моделью	12:45:34	2.87s	✓
09	final_answer Финальный ответ	12:45:37	90ms	✓
10	response_to_user Ответ отправлен пользователю	12:45:37	80ms	✓

04 retrieval_qdrant 12:45:32 690ms

Запрос к векторной БД

top_k: 20 | score_threshold: 0.28 | must: topic=GameDev | must: type=talk

Результаты (top 5)

id	score	metadata
36bfb78c7f...	0.842	talk GameDev 2026
a7f21c9d3a...	0.798	talk QA 2026
1d8e3b4c5f...	0.763	talk GameDev 2025
9c4d2e1f7b...	0.712	talk Tools 2026
7e9a0b1c2d...	0.689	talk GameDev 2024

▼ Запрос

```
{ "vector": [...],
  "top_k": 20,
  "filters": { "must": [{"key": "topic", "value": "GameDev"}, ...] }
```

[Показать больше](#)

06 enrichment 12:45:33 610ms

Добавлено полей

related_talks (3) | longDescription | speakers | schedule

Источник данных

Postgres (conference_db)

Оценка | Логи и метрики

Итоговая оценка ответа

0.87 Хорошо

Ответ релевантен и основан на контексте

12:45:38 auto-eval-v1.8

Метрики Правила

Retrieval quality 0.92

Groundedness 0.86

Consistency 0.81

Hallucination risk 0.07

Completeness 0.65

Детали проверок

- ✓ Ответ опирается на предоставленный контекст ПРОЙДЕНО
- ✓ Факты не противоречат контексту ПРОЙДЕНО
- ⚠ Найдены утверждения без источников ЧАСТИЧНО
- ⚠ Возможные упущения по ключевым аспектам ЧАСТИЧНО
- ⚠ Потенциальные галлюцинации ПРОЙДЕНО

Пользовательский запрос

Есть доклады про геймдев и тест-дизайн в играх?

Переписанный запрос

Найди доклады на тему GameDev и тест-дизайна в играх

Финальный промпт (сокращено) [Показать полный промпт](#)

Системная инструкция...
Контекст:
- Доклад: «Игры по правилам: тест-дизайн в мире хаоса геймдева» ...
- Доклад: «Инструменты для тестирования игровых механик» ...

Пользовательский запрос:
Найди доклады на тему GameDev и тест-дизайна в играх.
Ответь кратко и по существу, опираясь только на контекст.

Ответ модели

Да, есть доклад:
«Игры по правилам: тест-дизайн в мире хаоса геймдева»
Спикер: Алексей Симкин
Belarus TDC | 2026-04-27, 10:30–11:15 | Зал 2

Доклад посвящен систематизации подхода к тестированию в геймдеве и использованию методов тест-дизайна для игровых механик.

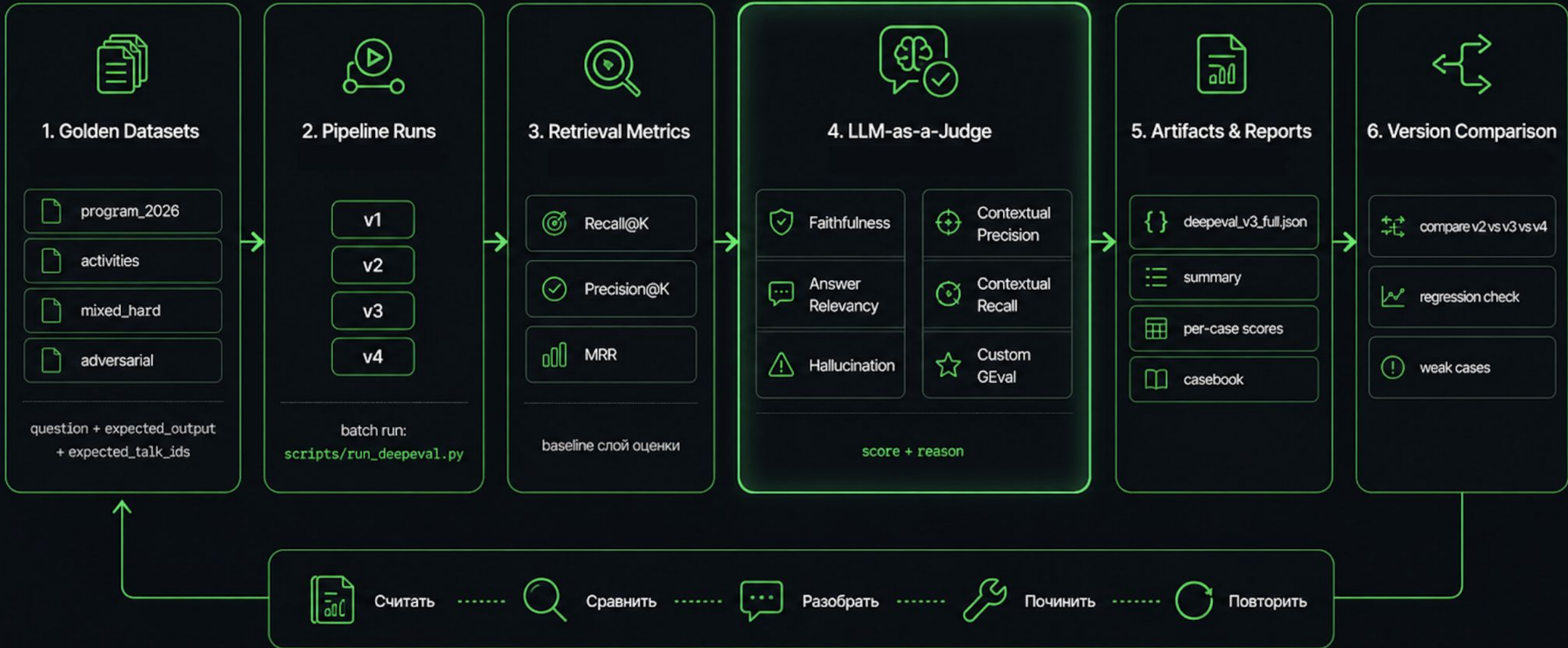
[Показать больше](#)

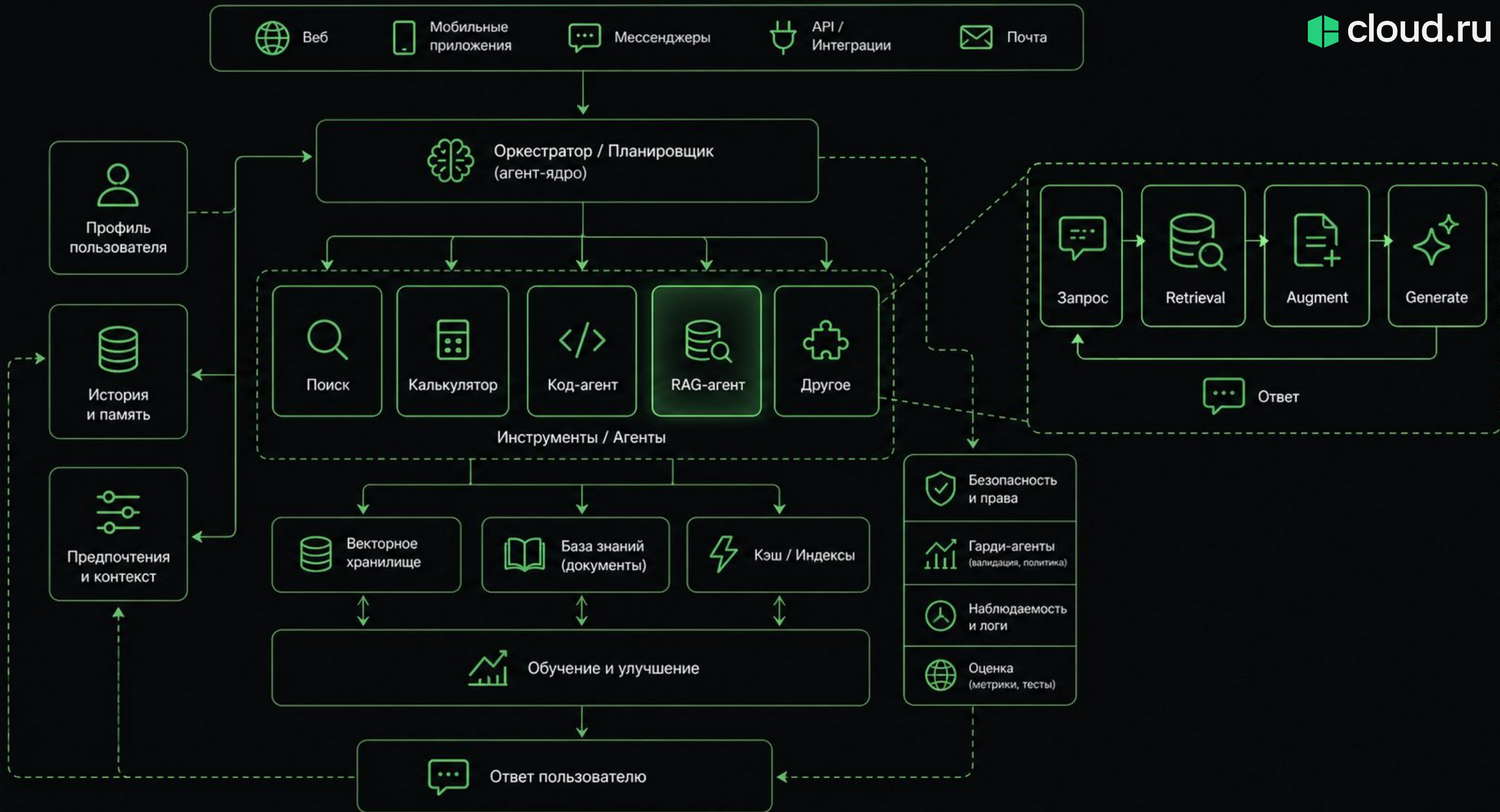
Вердикт и действия

OK auto-approve

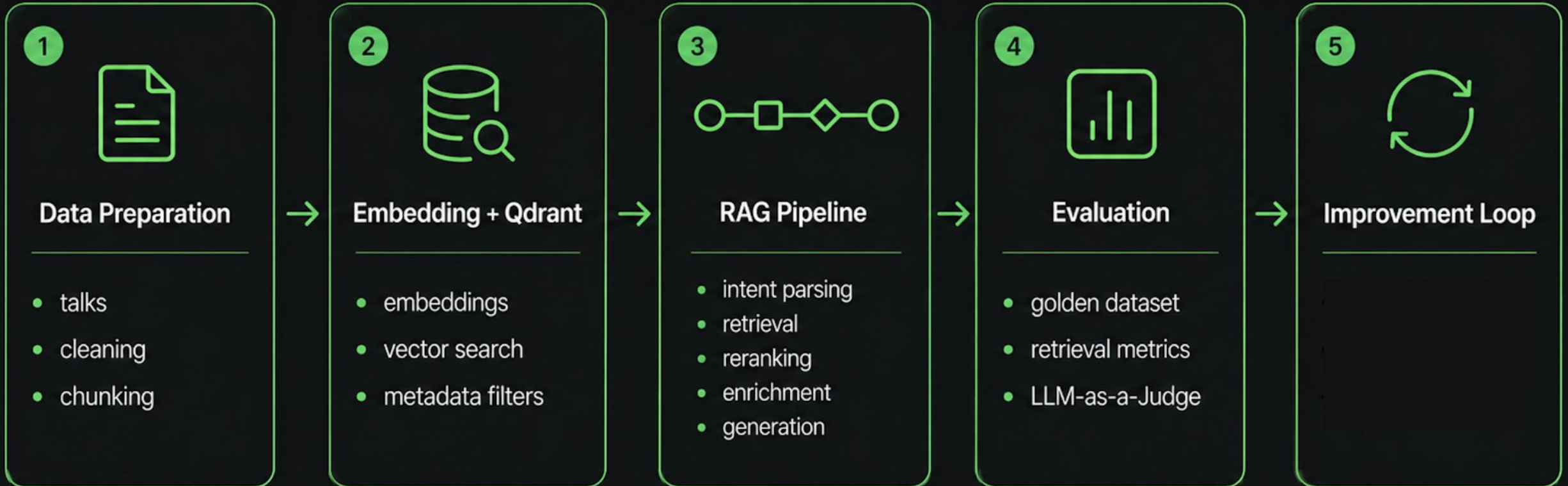
Можно включить автопубликацию ответа

[Открыть в разметке](#)





Что собрали?



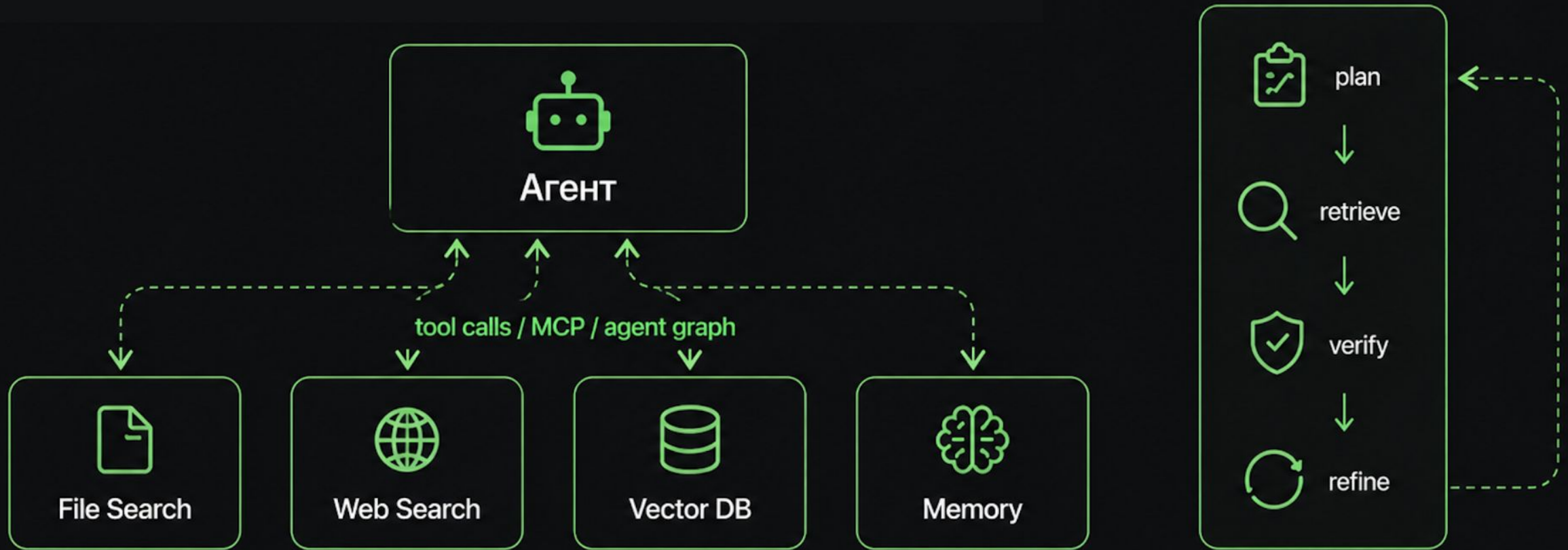
One-shot RAG



от one-shot
к iterative

Agentic loop





----- LangGraph / MCP / tool-based orchestration -----



Multimodal

- text
- images
- audio
- video
- geodata



Tools metadata

- tool registry
- MCP tools
- callable services
- API capabilities



Memory

- short-term memory
- long-term memory
- retrieved state
- user context



retrieval

=



chunks

+



files

+



relations

+



tools

+



memory

Q&A

Полезное:

1. DeepEval -> <https://github.com/confident-ai/deepeval>
2. LangChain -> <https://github.com/langchain-ai/langchain>
3. Пример Jupyter Notebook для генерации и оценки:
<https://disk.yandex.ru/d/fD3GsnURn2u52A>

