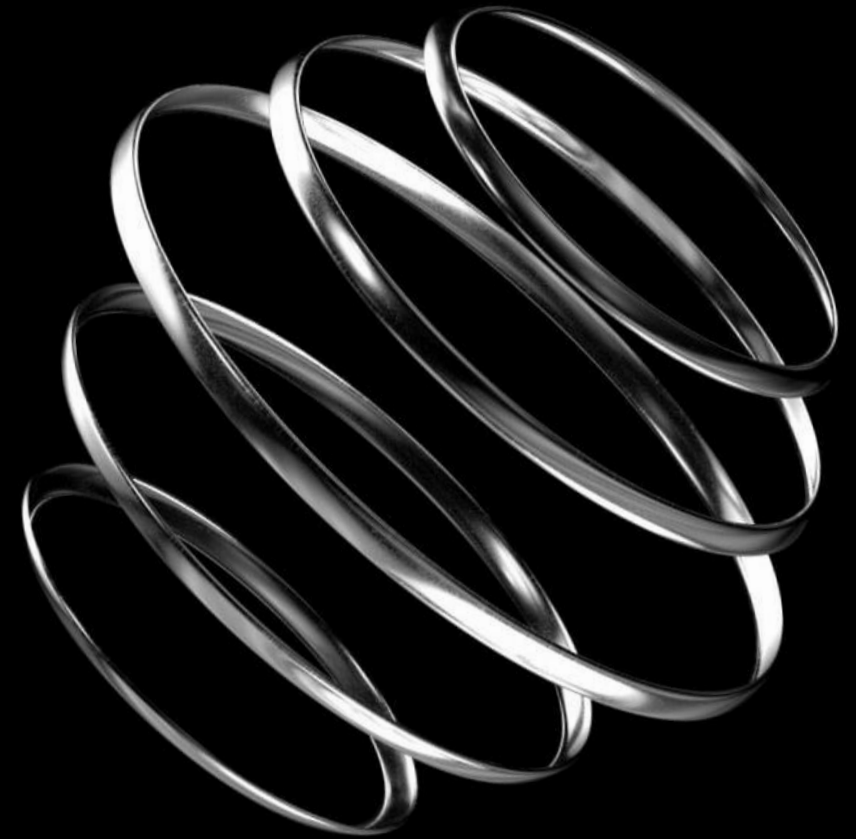


# Тестирование LLM приложений с DeepEval



- NLP-инженер в red\_mad\_robot. Разрабатываю, улучшаю и исследую агентные системы;
- С 2022 года занимаюсь AI. Участвовал в работе над проектами, связанными с Classic ML, Computer Vision, NLP, RL.

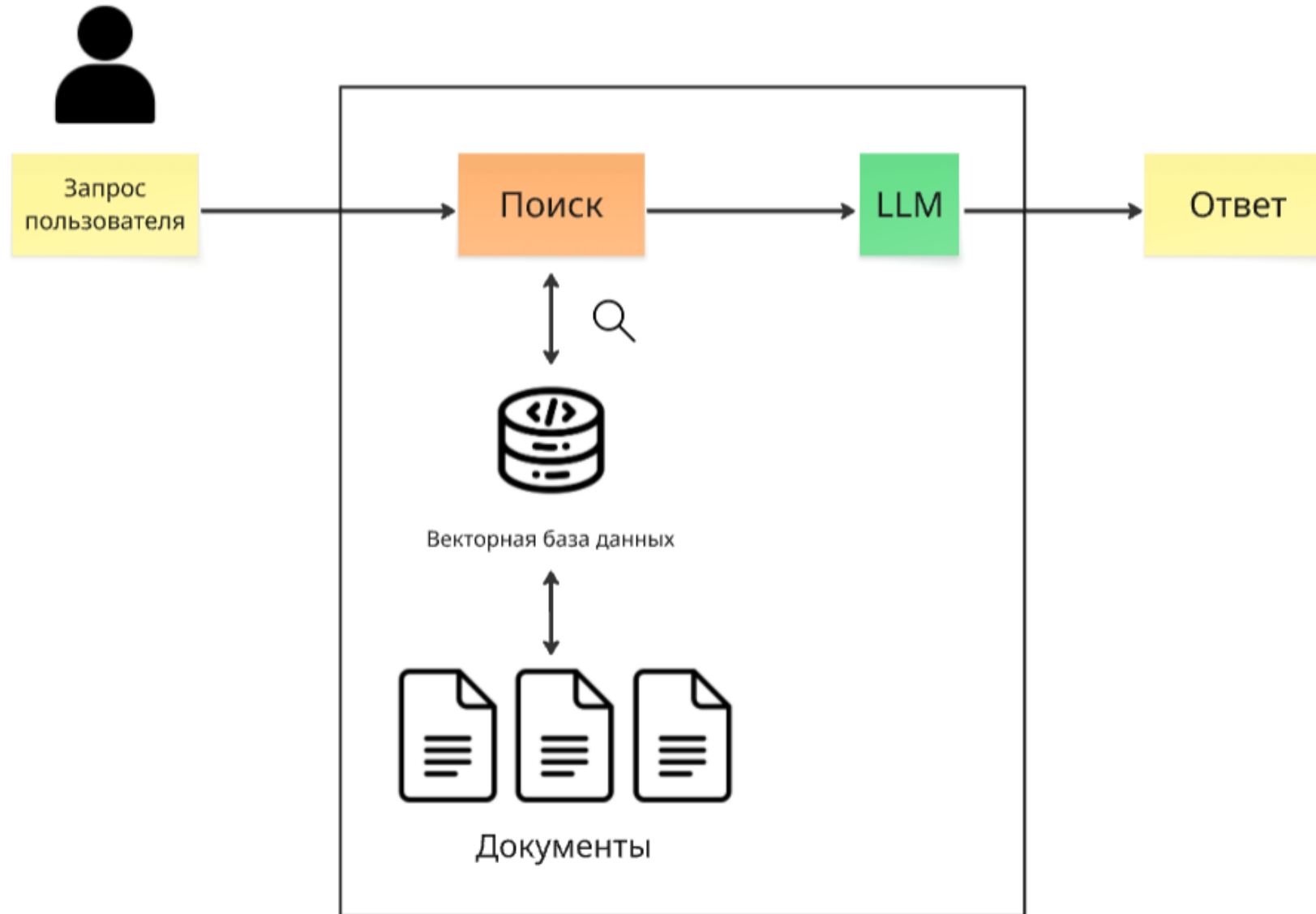


Максимов Максим

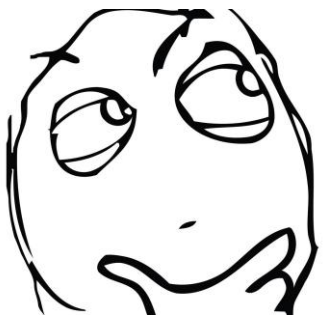
Представим, у нас есть LLM ассистент с RAG по продуктовой базе супермаркета.

Он помогает пользователям находить нужные товары, а также отвечает на вопросы по ним.

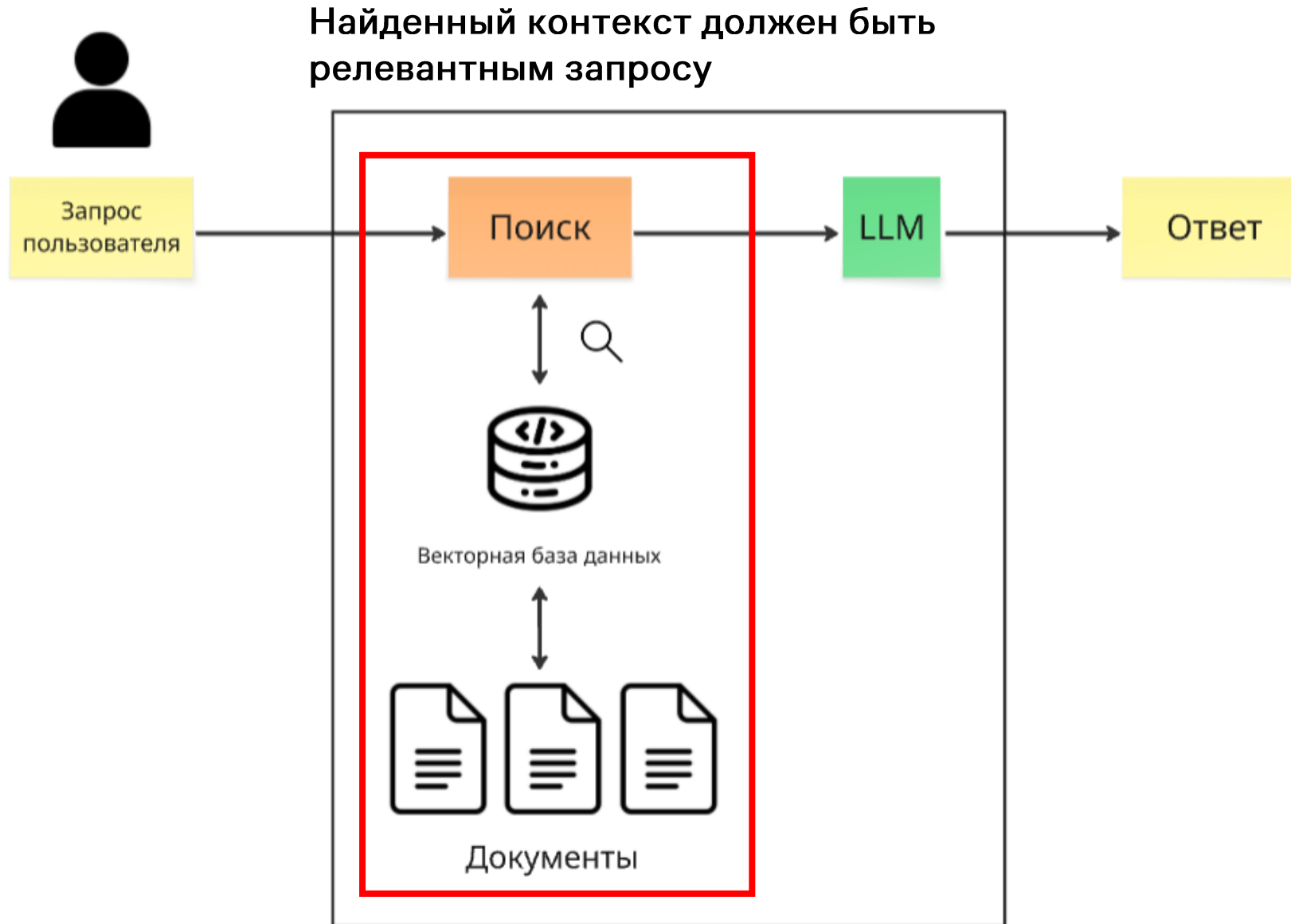
# LLM ассистент с RAG по продуктовой базе



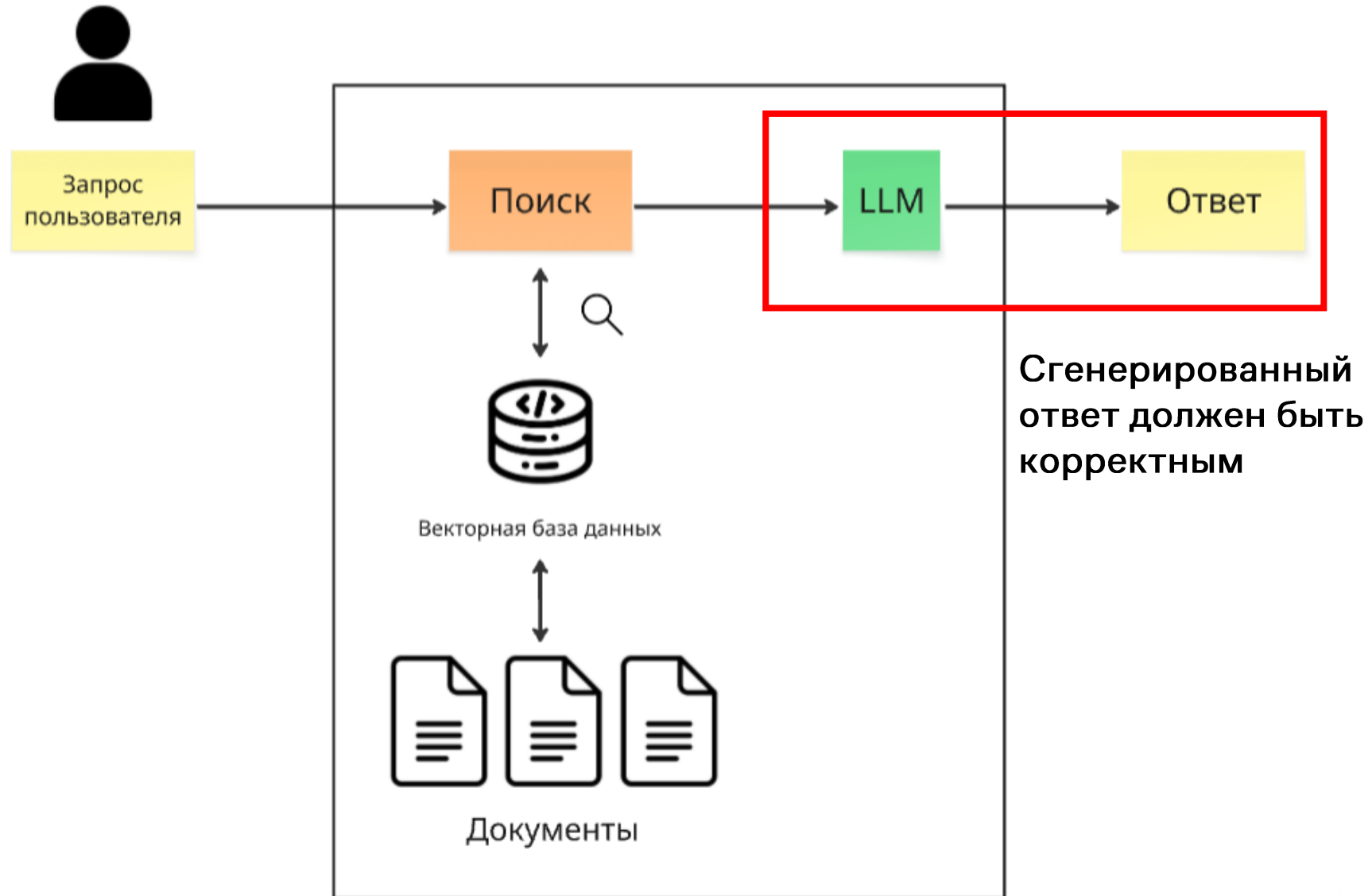
# Как протестировать нашего ассистента?



# Наверное, классическое Unit-тестирование!



# Не все так просто...



# Тестирование LLM приложений

# Почему тестирование LLM – это сложно

Недетерминированность

Работа с естественным  
языком

Попробуем протестировать и оценить  
работу нашего ассистента

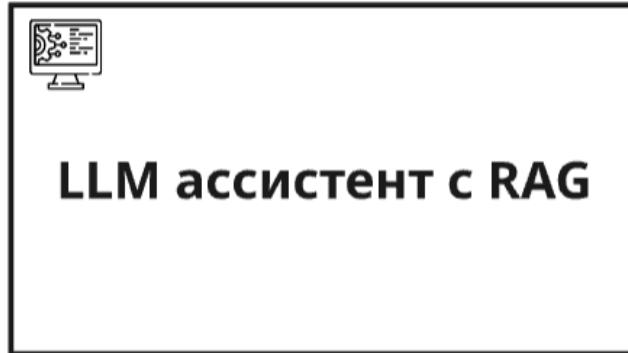


Запрос  
пользователя

Можно ли средство Cleaner  
для ковров использовать на  
шёлке?

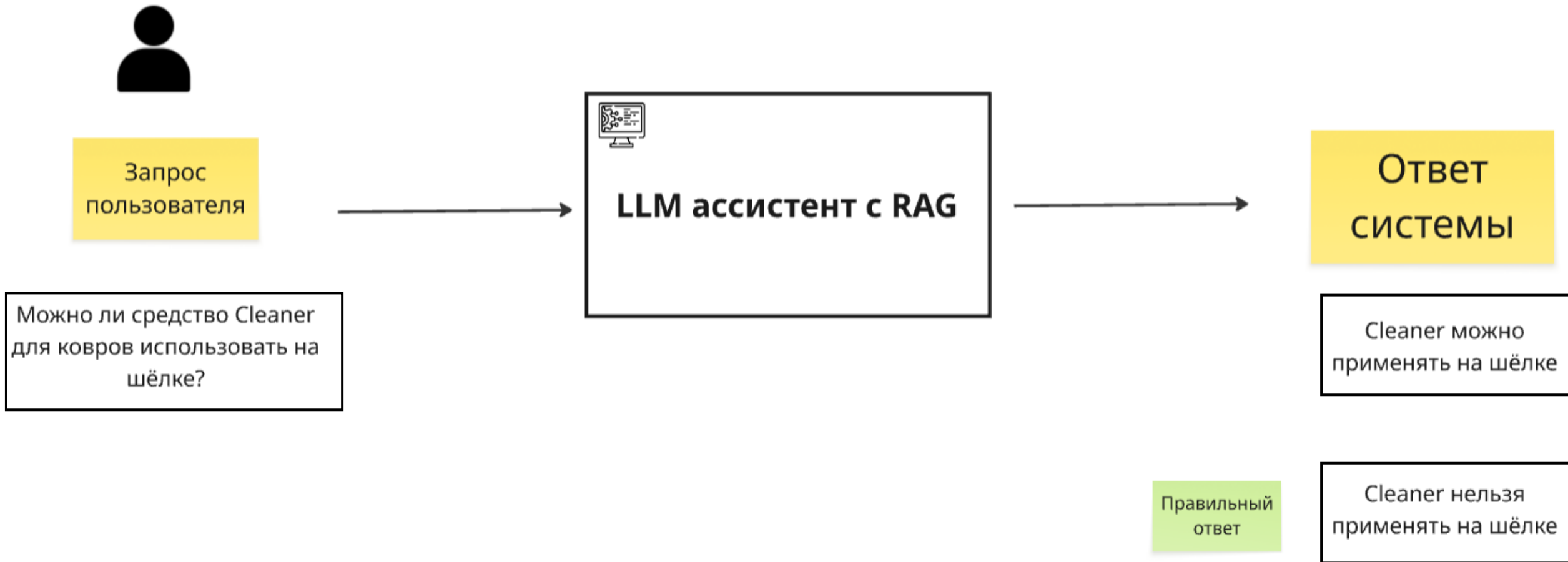


Запрос  
пользователя



Можно ли средство Cleaner  
для ковров использовать на  
шёлке?





# Классические NLP метрики

**BLUE**

**ROUGE**

**METEOR**

# Классические NLP метрики

BLUE

ROUGE

METEOR

В качестве примера, возьмем ROUGE

Эталонный текст

Cleaner

нельзя

применять

на

шёлке

# Подсчет метрики ROUGE

Эталонный текст

Cleaner

нельзя

применять

на

шёлке

Сгенерированный текст

Cleaner

МОЖНО

применять

на

шёлке

Эталонный текст

Cleaner

нельзя

применять

на

шёлке

Сгенерированный текст

Cleaner

МОЖНО

применять

на

шёлке

Находим пересечение токенов

$R = \{Cleaner, \text{нельзя}, \text{применять}, \text{на}, \text{шёлке}\}, \quad |R| = 5$

$C = \{Cleaner, \text{можно}, \text{применять}, \text{на}, \text{шёлке}\}, \quad |C| = 5$



$R \cap C = \{Cleaner, \text{применять}, \text{на}, \text{шёлке}\}, \quad |R \cap C| = 4$

$$\text{Recall} = \frac{|R \cap C|}{|R|} = \frac{4}{5} = 0.8$$

$$\text{Precision} = \frac{|R \cap C|}{|C|} = \frac{4}{5} = 0.8$$

$$F_1 = \text{ROUGE-1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.8 \times 0.8}{0.8 + 0.8} = 0.8$$

$$\text{Recall} = \frac{|R \cap C|}{|R|} = \frac{4}{5} = 0.8$$

$$\text{Precision} = \frac{|R \cap C|}{|C|} = \frac{4}{5} = 0.8$$

$$F_1 = \text{ROUGE-1} = 2 \cdot \frac{\text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} = 2 \cdot \frac{0.8 \times 0.8}{0.8 + 0.8} = \boxed{0.8}$$

Высокий score!

# Подсчет метрики ROUGE

Эталонный текст

Cleaner

нельзя

применять

на

шёлке

Сгенерированный текст

Cleaner

МОЖНО

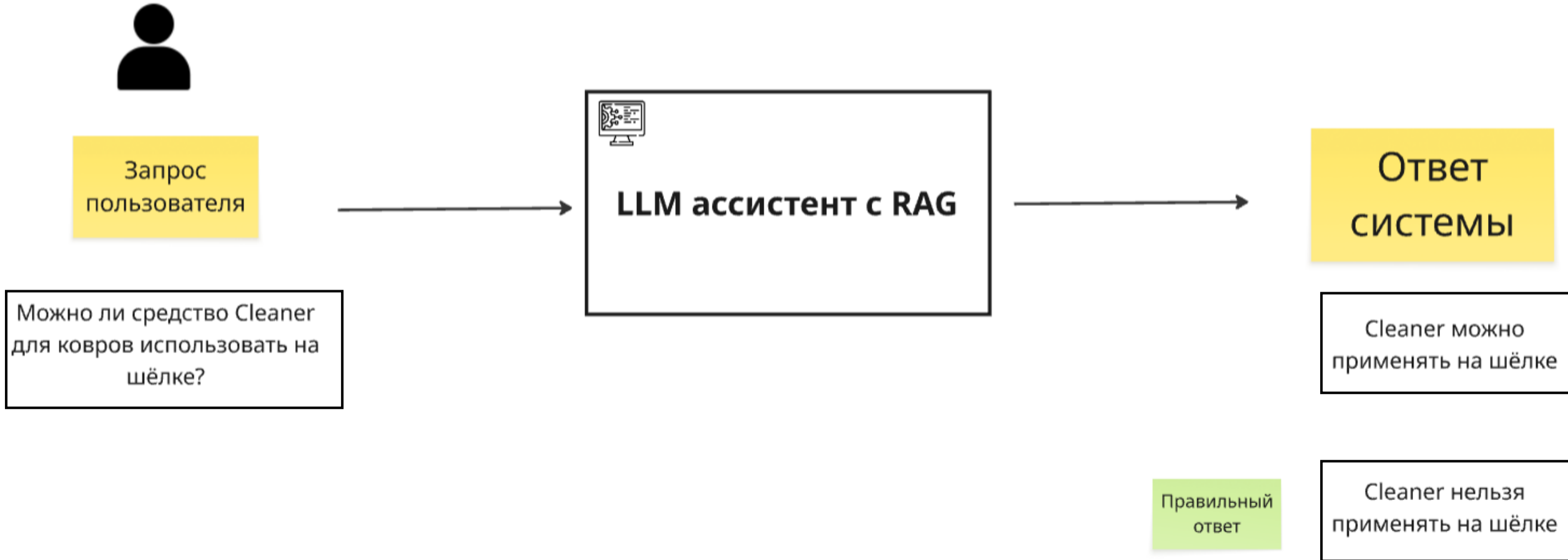
применять

на

шёлке

**Вывод:** Классические метрики считают совпадения слов, а не смысл.

# Решение – LLM-as-a-Judge

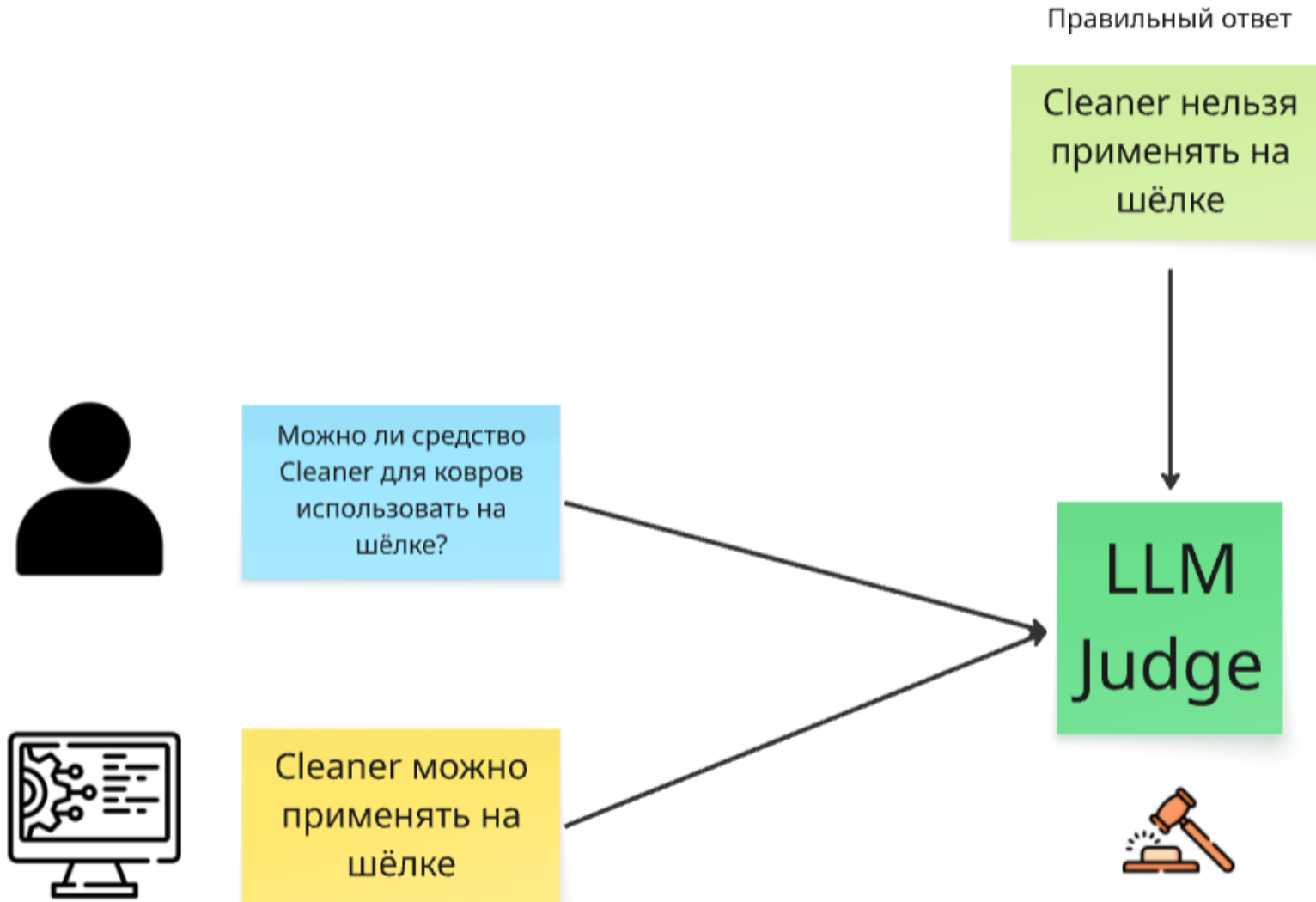




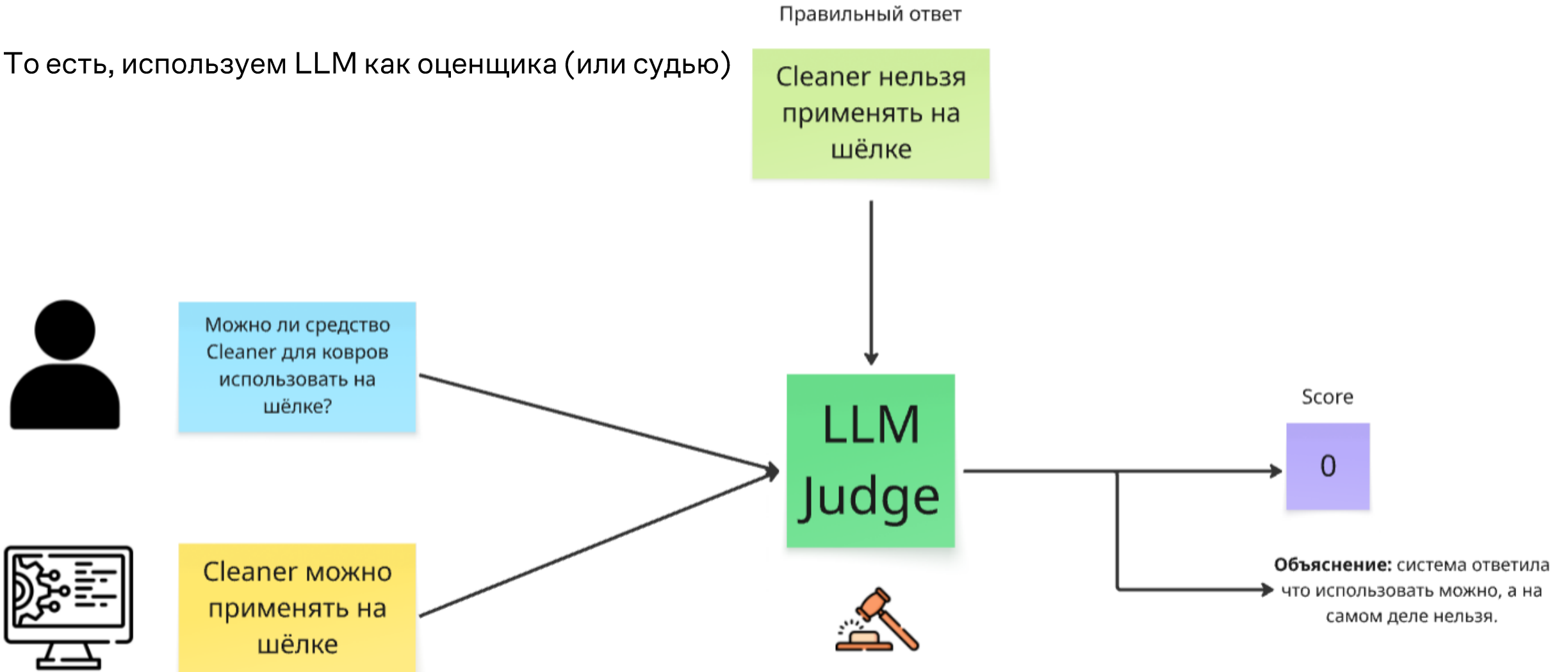
Можно ли средство  
Cleaner для ковров  
использовать на  
шёлке?



Cleaner можно  
применять на  
шёлке



То есть, используем LLM как оценщика (или судью)



То есть, просто берем LLM, промпты и тестируем?

Хочется сделать следующее:

1. Не только подсчитать метрики, но и сравнить различные подходы
2. Не реализовывать с нуля алгоритмы оценки, а использовать готовые
3. Внедрить и произвести оценку в существующую инфраструктуру

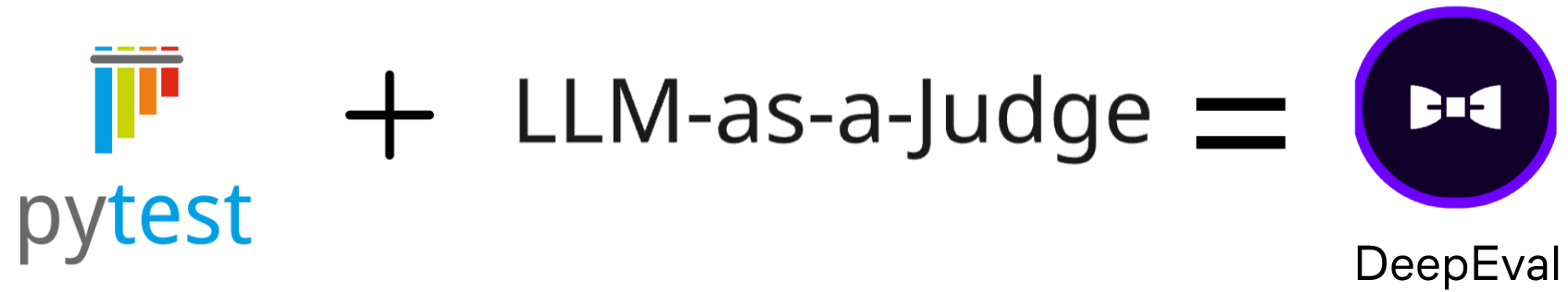
# Что такое DeepEval



DeepEval – это open-source платформа для оценки и тестирования приложений, работающих на основе LLM

The screenshot shows the GitHub repository page for 'deepeval'. The repository is public and has 61 watchers, 1.4k forks, and 15k stars. It is maintained by 'penguin-ip' and has 294 branches and 292 tags. The repository contains several folders: '.github', '.scripts/changelog', '.vscode', 'assets', and 'deepeval'. The 'deepeval' folder has the most recent commit, 'add docs intro', made yesterday. The repository is described as 'The LLM Evaluation Framework' and includes a README and an Apache-2.0 license. The repository is also linked to 'deepeval.com' and has several tags: 'python', 'evaluation-metrics', 'evaluation-framework', 'llm-evaluation', 'llm-evaluation-framework', and 'llm-evaluation-metrics'.

Folder	Commit Message	Commit Time
.github	add changelog workflow	last month
.scripts/changelog	lint only	last month
.vscode	improve docs	4 months ago
assets	.	last month
deepeval	add docs intro	yesterday



# Сферы применения DeepEval

Чат боты

RAG

AI-агенты

LLM арена

# Сферы применения DeepEval

Чат боты

RAG

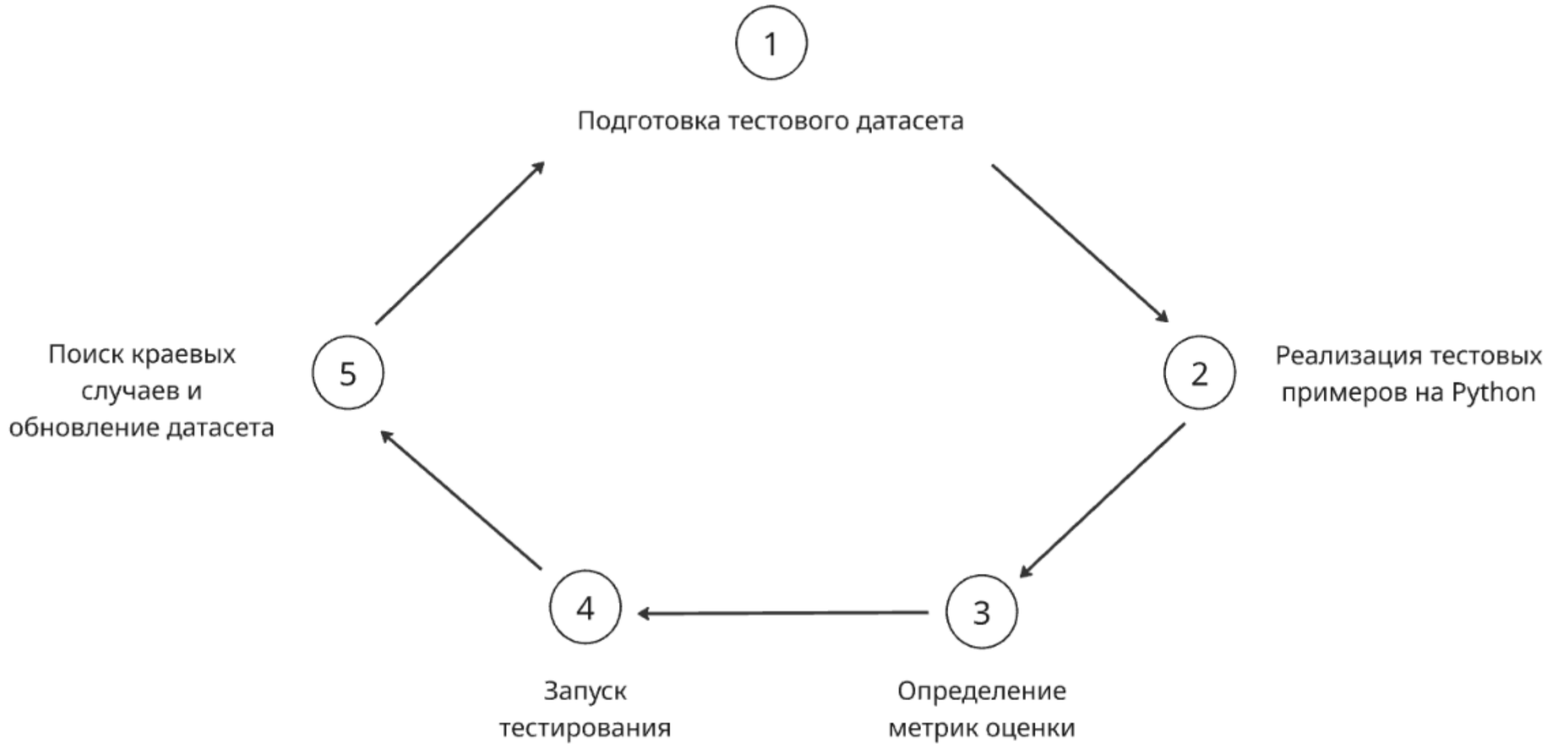
AI-агенты

Подходит!

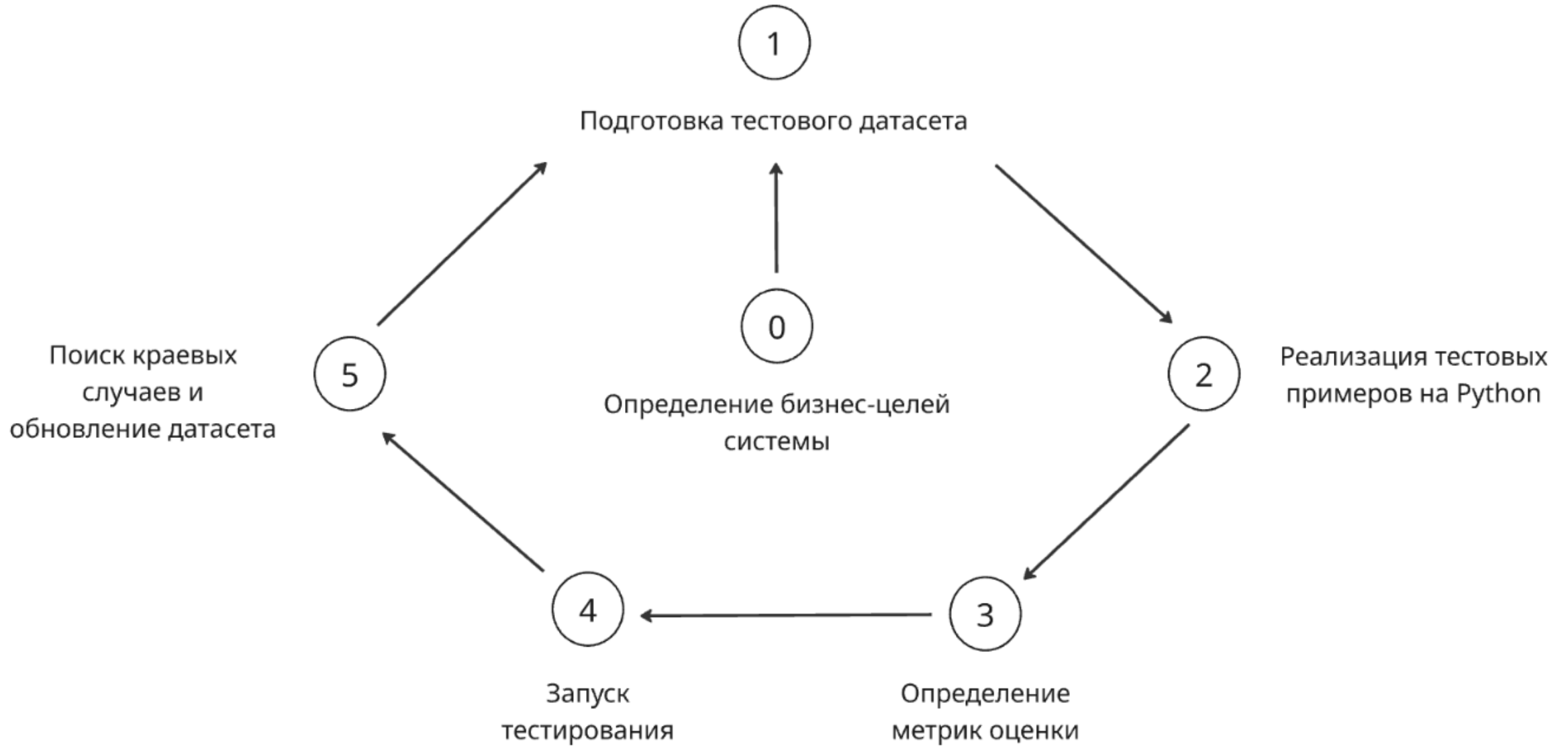
LLM арена

С чего начать?

# Процесс тестирования с DeepEval



# Процесс тестирования с DeepEval



# Протестируем нашего ассистента с DeepEval

# Процесс тестирования с DeepEval



# Определяем бизнес-цель ассистента

Повышение лояльности клиентов

## Определяем бизнес-цель ассистента

Повышение лояльности клиентов



Нужно чтобы ассистент давал  
точные ответы на вопросы

# Процесс тестирования с DeepEval

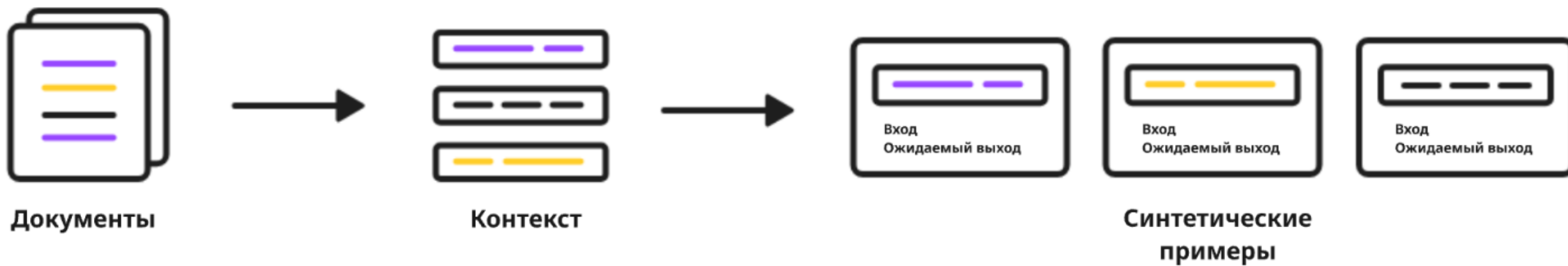


# Форматы работы с датасетами в DeepEval

## Тестовый датасет:

Эталонный набор «вопрос — ответ» для оценки качества LLM-ассистента.

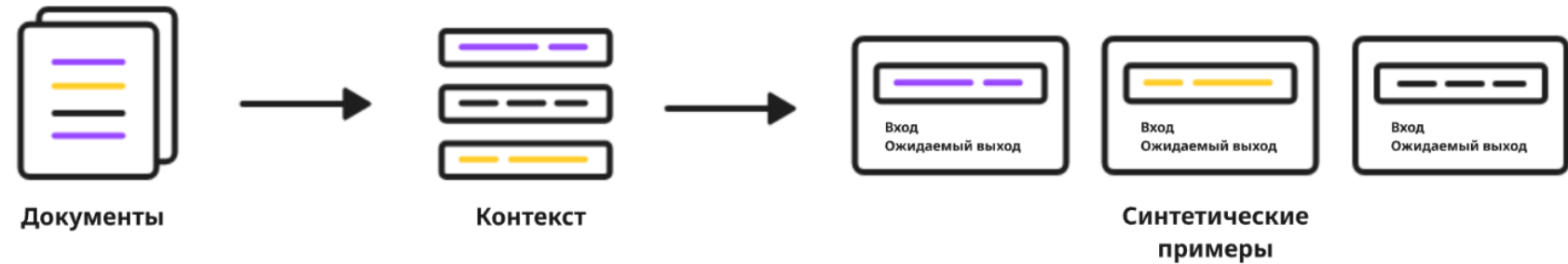
## Генерация датасета из документов



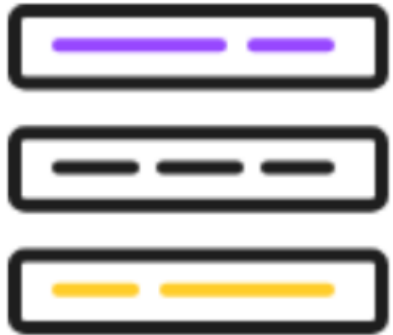
Используем если:

- Имеются документы в форматах pdf, docx, txt

## Генерация датасета из документов



## Генерация датасета из сформированной базы данных



Контекст



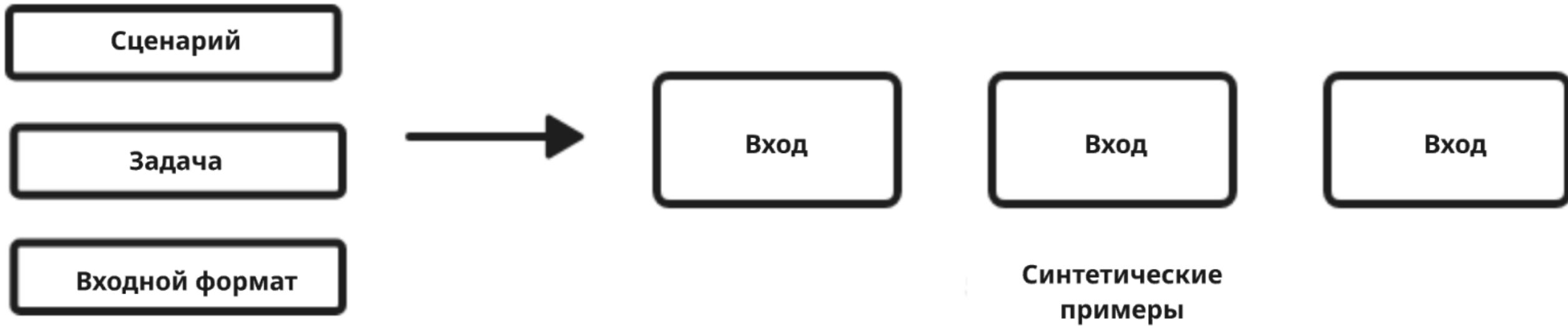
Синтетические  
примеры

## Используем если:

- Собрана векторная база данных с фрагментами контекста



Генерация датасета с нуля по собственному сценарию



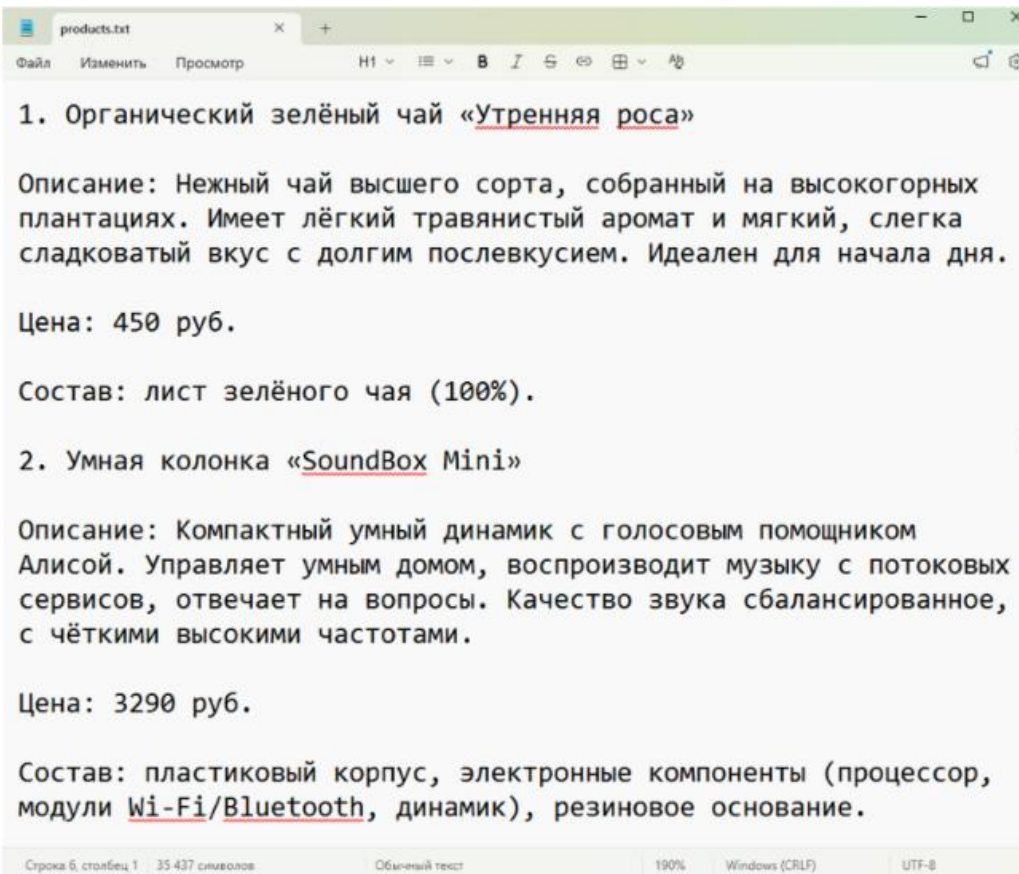
## Используем если:

- Необходимо создать датасет, выходящий за рамки существующей базы знаний с нуля

Генерация датасета с нуля по собственному сценарию



# Подготовим тестовый датасет



products.txt

1. Органический зелёный чай «Утренняя роса»

Описание: Нежный чай высшего сорта, собранный на высокогорных плантациях. Имеет лёгкий травянистый аромат и мягкий, слегка сладковатый вкус с долгим послевкусием. Идеален для начала дня.

Цена: 450 руб.

Состав: лист зелёного чая (100%).

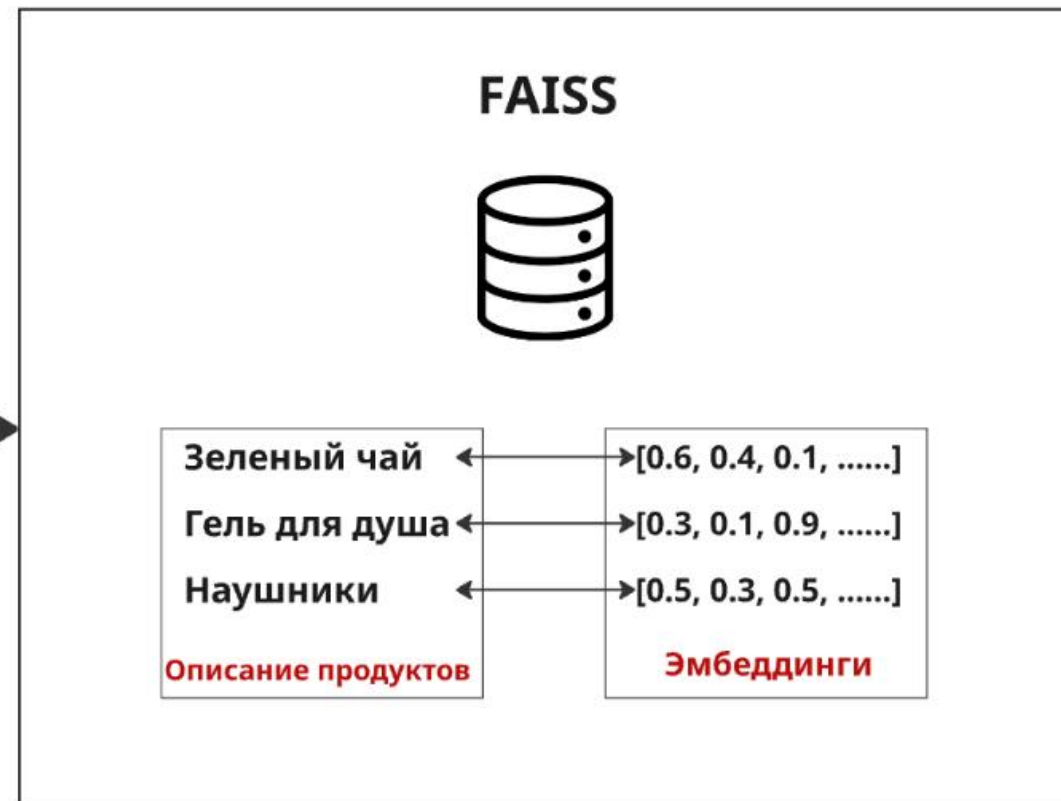
2. Умная колонка «SoundBox Mini»

Описание: Компактный умный динамик с голосовым помощником Алисой. Управляет умным домом, воспроизводит музыку с потоковых сервисов, отвечает на вопросы. Качество звука сбалансированное, с чёткими высокими частотами.

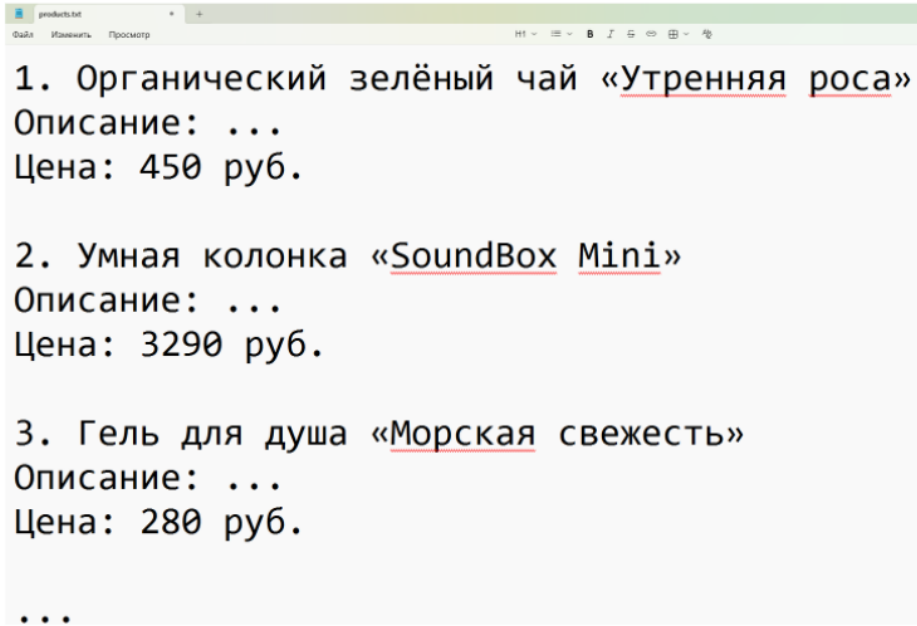
Цена: 3290 руб.

Состав: пластиковый корпус, электронные компоненты (процессор, модули Wi-Fi/Bluetooth, динамик), резиновое основание.

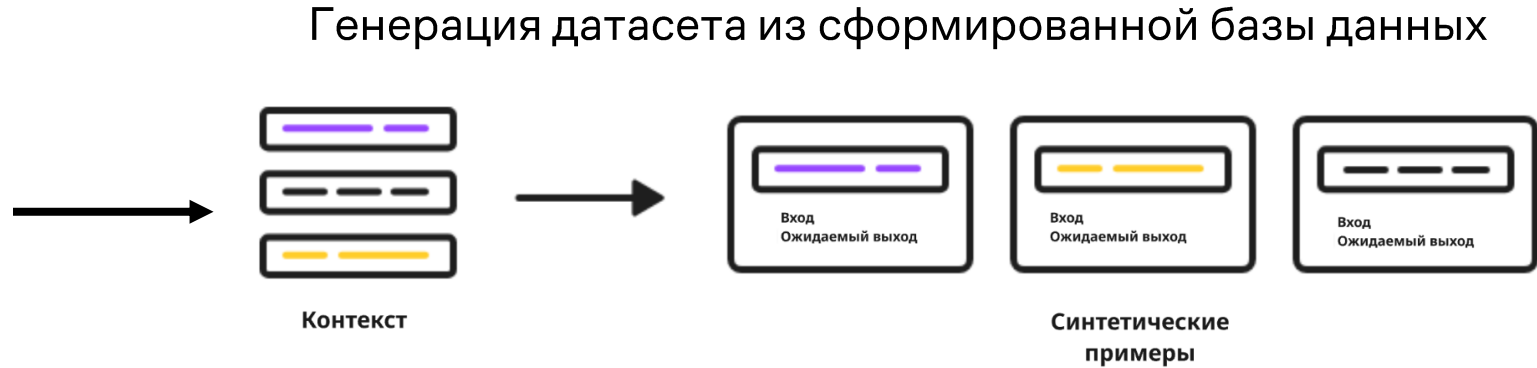
Строка 6, столбец 1 | 35 437 символов | Обычный текст | 190% | Windows (CRLF) | UTF-8



# Подготовка тестового датасета



**FAISS**



```
import ... # deepeval, data
...
model = CustomOpenAIModel(...)
contexts = [[chunk] for chunk in chunks]

synthesizer = Synthesizer(
    model=model,
    styling_config=StylingConfig(
        scenario="Покупатель интернет-магазина задаёт вопросы консультанту о
товарах. Все вопросы и ответы строго на русском языке.",
        task="Сгенерируй вопрос покупателя и ответ консультанта на русском языке",
    ),
)
goldens = synthesizer.generate_goldens_from_contexts(
    contexts=random.choices(contexts, k=10),
    include_expected_output=True,
    max_goldens_per_context=2,
)
...
df = pd.DataFrame(rows)
df.to_excel(...)
```

```
import ... # deepeval, data
...
model = CustomOpenAIModel(...)
contexts = [[chunk] for chunk in chunks]
```

```
synthesizer = Synthesizer(
    model=model,
    styling_config=StylingConfig(
        scenario="Покупатель интернет-магазина задаёт вопросы консультанту о
товарах. Все вопросы и ответы строго на русском языке.",
        task="Сгенерируй вопрос покупателя и ответ консультанта на русском языке",
    ),
)
goldens = synthesizer.generate_goldens_from_contexts(
    contexts=random.choices(contexts, k=10),
    include_expected_output=True,
    max_goldens_per_context=2,
)
```

```
...
df = pd.DataFrame(rows)
df.to_excel(...)
```

ВОПРОС (INPUT)	ОЖИДАЕМЫЙ ОТВЕТ (EXPECTED_OUTPUT)	КОНТЕКСТ (CONTEXT)
Сколько стоят наушники Listen Pro?	Наушники Listen Pro стоят <b>8 990 руб.</b>	<span>товар</span> Беспроводные наушники «Listen Pro». Цена: 8 990 руб.
Из чего состоит гель для душа Морская свежесть?	Содержит воду, ПАВ (кокамидопропилбетаин), экстракт морских водорослей, ментол, парфюмерную композицию и консервант.	<span>состав</span> Гель для душа «Морская свежесть». Состав: вода, ПАВ, экстракт морских водорослей, ментол, парфюмерная композиция, консервант. Цена: 280 руб.
Какой есть зеленый чай?	Имеется органический зелёный чай «Утренняя роса» за <b>450 руб.</b>	<span>товар</span> Органический зелёный чай «Утренняя роса». Цена: 450 руб. Нежный чай высшего сорта, собранный на высокогорных плантациях.
...		

# Процесс тестирования с DeepEval



# Определение тестового примера

Атомарная единица взаимодействия с LLM-приложением



# Типы тестовых примеров в DeepEval

Одношаговые

Single-tern test case (одношаговый пример) – проверка одношаговой итерации взаимодействия с LLM-приложением

## Единичный вызов LLM



## Эталонные примеры

Ожидаемые инструменты (опционально)

Ожидаемый ответ (опционально)

Контекст (опционально)

# Типы тестовых примеров в DeepEval

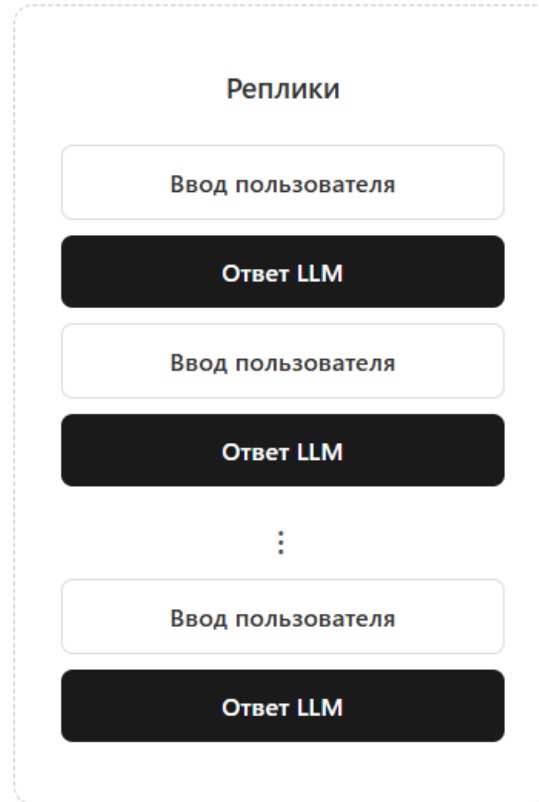
Одношаговые

Многошаговые

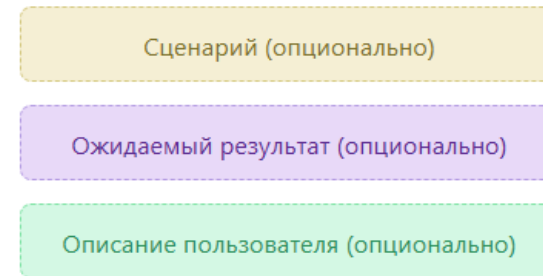
Multi-tern test case (многошаговый пример)  
– проверка серии взаимодействий с LLM-приложением

## Диалоговый вызов LLM

Роли «user» и «assistant»,  
могут содержать вызовы  
инструментов и извлечённый  
контекст →



## Дополнительные параметры



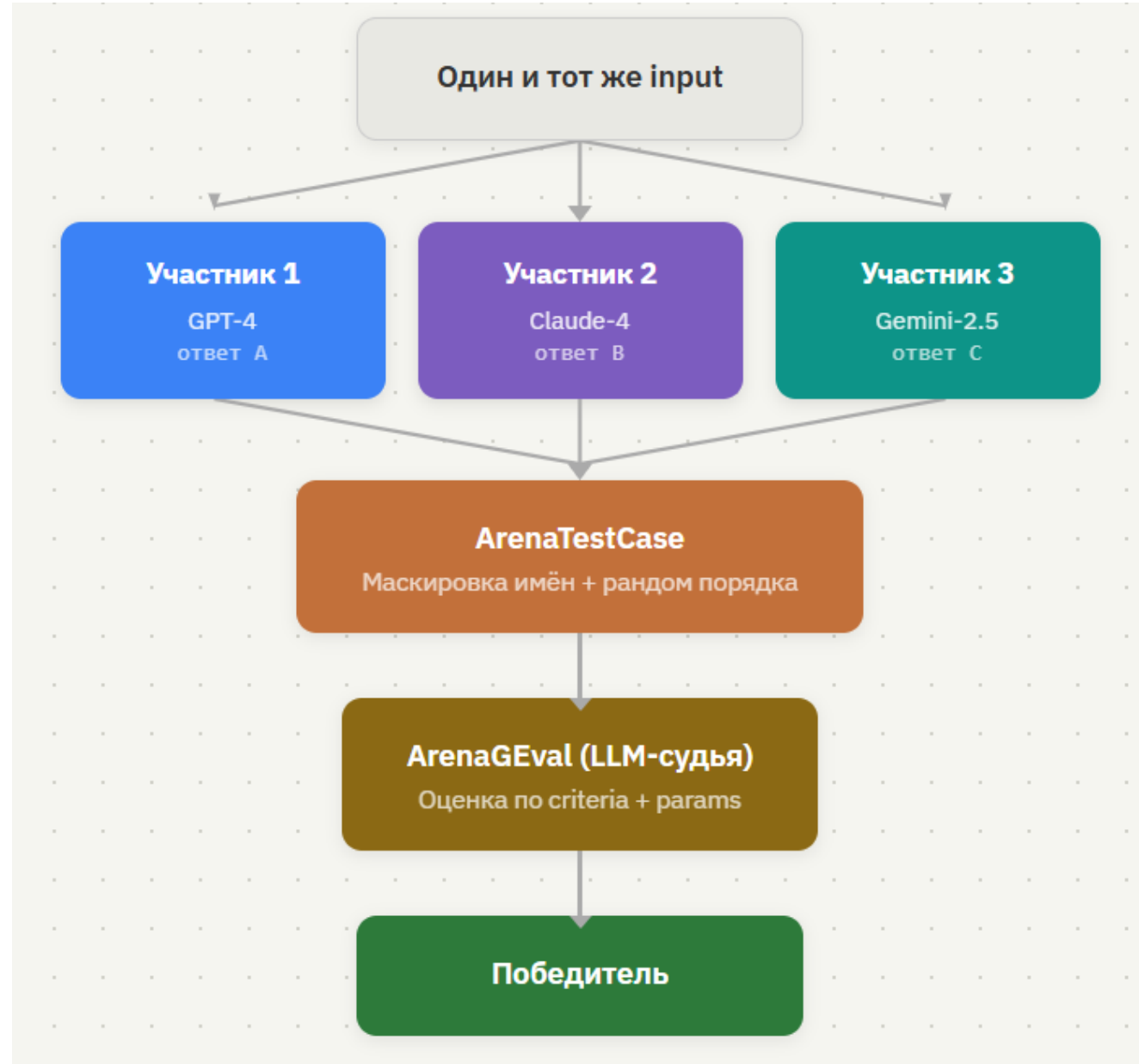
# Типы тестовых примеров в DeepEval

Одношаговые

Многошаговые

Арена

Arena test case – сравнение различных LLM с различной конфигурациях на задачах.



# Типы тестовых примеров в DeepEval

Одношаговые

Многошаговые

Арена



```
from deepeval.test_case import LLMTestCase
from dataset import dataset
from rag_app import rag_pipeline

test_cases = []

for golden in dataset.goldens:
    # Прогоняем через RAG
    result = rag_pipeline(golden.input)

    test_case = LLMTestCase(
        input=golden.input,
        actual_output=result["actual_output"],
        expected_output=golden.expected_output,
        context=golden.context, # ground truth контекст
        retrieval_context=result["retrieval_context"], # что реально нашёл RAG
    )
    test_cases.append(test_case)
```



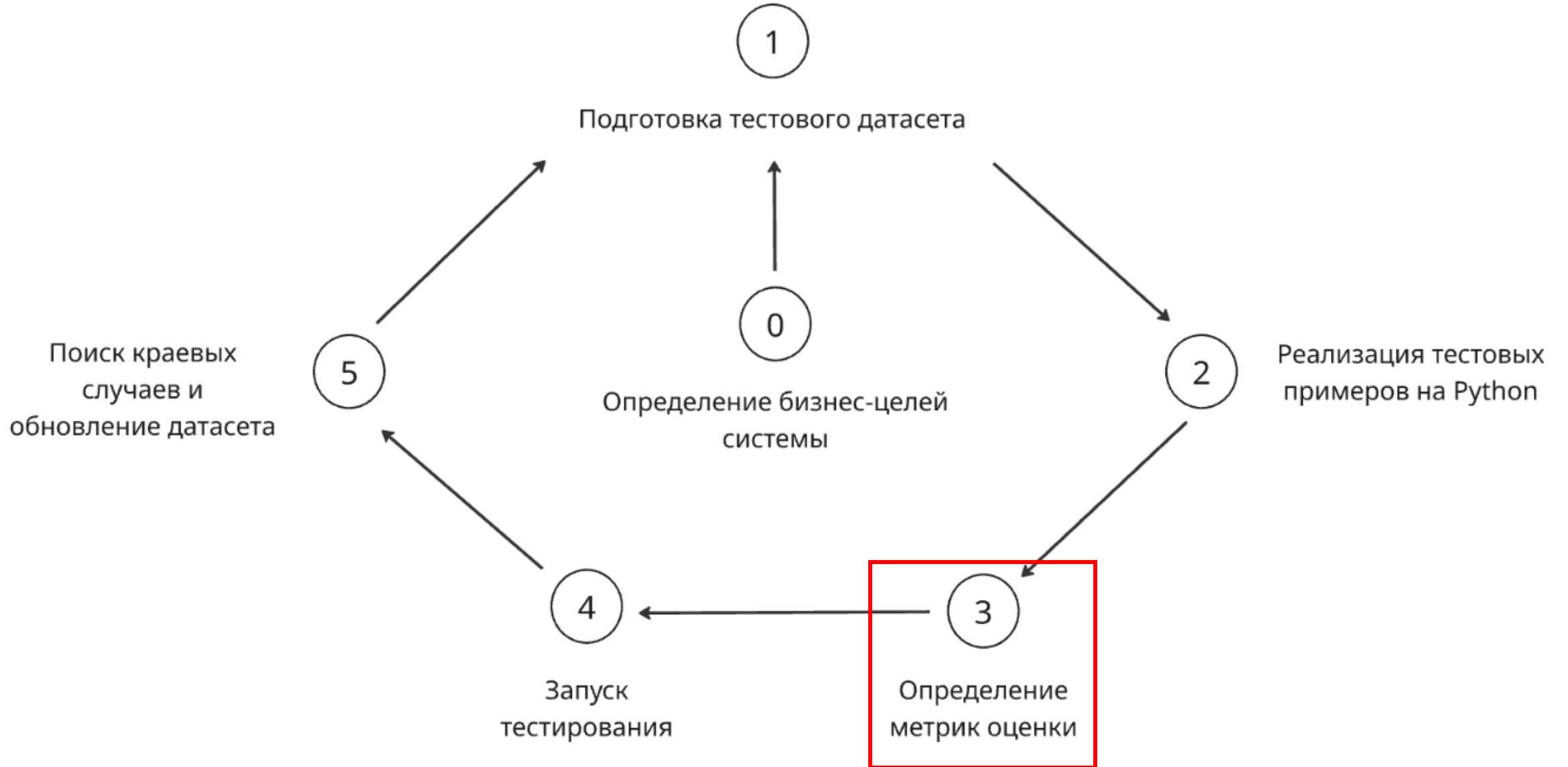
```
from deepeval.test_case import LLMTestCase
from dataset import dataset
from rag_app import rag_pipeline

test_cases = []

for golden in dataset.goldens:
    # Прогоняем через RAG
    result = rag_pipeline(golden.input)

    test_case = LLMTestCase(
        input=golden.input,
        actual_output=result["actual_output"],
        expected_output=golden.expected_output,
        context=golden.context, # ground truth контекст
        retrieval_context=result["retrieval_context"], # что реально нашёл RAG
    )
    test_cases.append(test_case)
```

# Процесс тестирования с DeepEval



Метрики в **DeepEval** – данный инструмент содержит более **50** готовых к использованию метрик для различных задач

Метрики в **DeepEval** – данный инструмент содержит более **50** готовых к использованию метрик для различных задач



## **Answer Relevancy** (релевантность ответа)

Оценка того, как хорошо наша система генерирует ответ.

## ШАГ 1 — ИЗВЛЕЧЕНИЕ УТВЕРЖДЕНИЙ

### Answer Relevancy (релевантность ответа)

Оценка того, как хорошо наша система генерирует ответ.

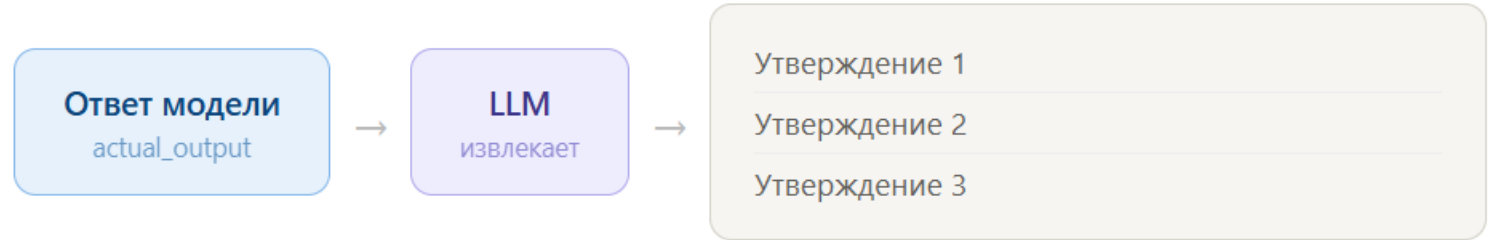


Возьмем 2 метрики:

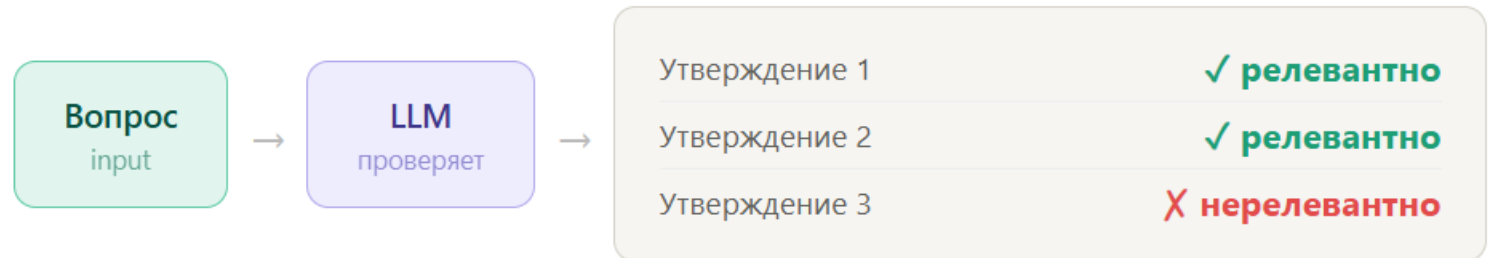
**Answer Relevancy** (релевантность ответа)

Оценка того, как хорошо наша система генерирует ответ.

ШАГ 1 — ИЗВЛЕЧЕНИЕ УТВЕРЖДЕНИЙ



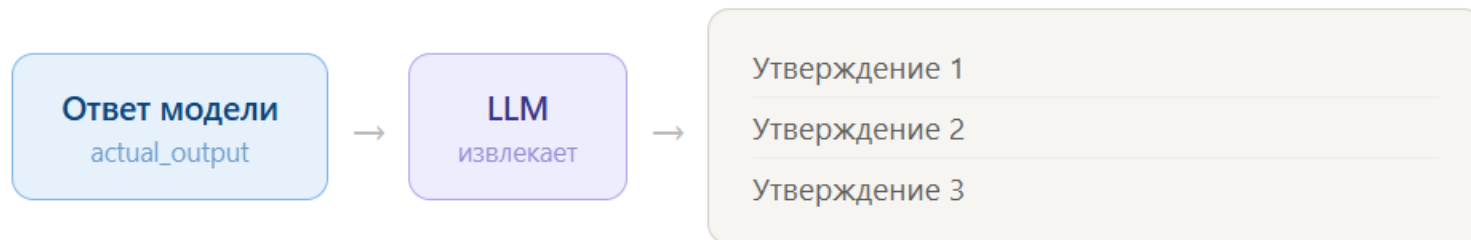
ШАГ 2 — КЛАССИФИКАЦИЯ



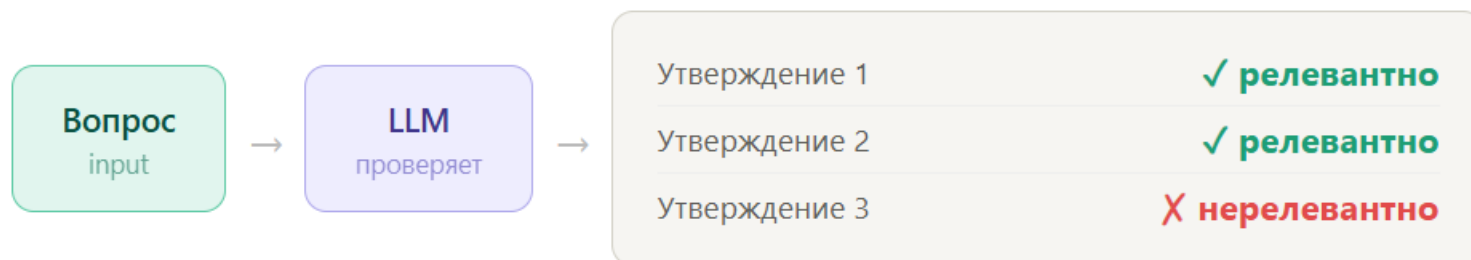
## Answer Relevancy (релевантность ответа)

Оценка того, как хорошо наша система генерирует ответ.

### ШАГ 1 — ИЗВЛЕЧЕНИЕ УТВЕРЖДЕНИЙ



### ШАГ 2 — КЛАССИФИКАЦИЯ



### ШАГ 3 — РАСЧЁТ

**Answer Relevancy = 2 / 3 = 0.67**  
релевантные утверждения / все утверждения

## Context Recall (полнота контекста)

Оценка того, как хорошо наша система находит контекст.

## ШАГ 1 — ИЗВЛЕЧЕНИЕ УТВЕРЖДЕНИЙ ИЗ ЭТАЛОННОГО ОТВЕТА

**Contextual Recall (полнота контекста)**

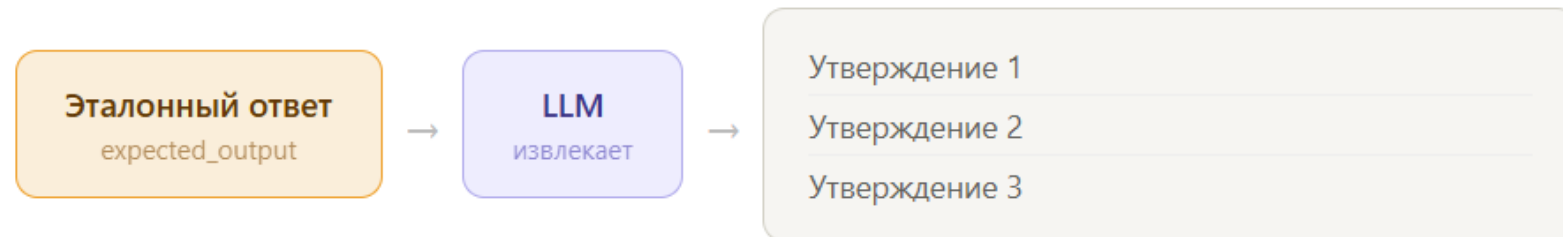
Оценка того, как хорошо наша система находит контекст.



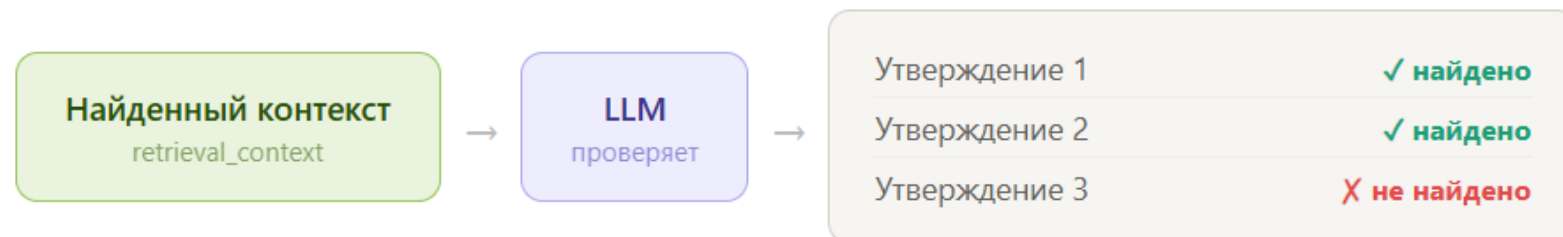
## Contextual Recall (полнота контекста)

Оценка того, как хорошо наша система находит контекст.

### ШАГ 1 — ИЗВЛЕЧЕНИЕ УТВЕРЖДЕНИЙ ИЗ ЭТАЛОННОГО ОТВЕТА



### ШАГ 2 — МОЖНО ЛИ ПОДТВЕРДИТЬ КОНТЕКСТОМ?



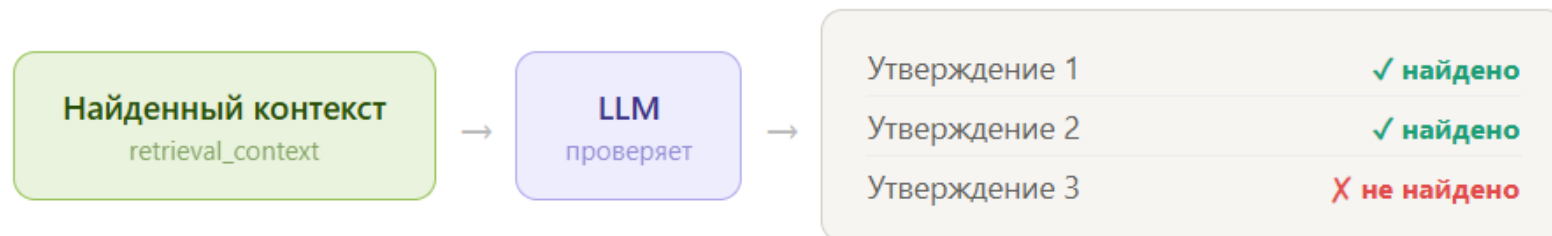
## Contextual Recall (полнота контекста)

Оценка того, как хорошо наша система находит контекст.

### ШАГ 1 — ИЗВЛЕЧЕНИЕ УТВЕРЖДЕНИЙ ИЗ ЭТАЛОННОГО ОТВЕТА



### ШАГ 2 — МОЖНО ЛИ ПОДТВЕРДИТЬ КОНТЕКСТОМ?



### ШАГ 3 — РАСЧЁТ



# Метрика G-Eval = LLM-as-a-Judge + Chain-of-Thought

### Описание задачи

*Вам дан ответ RAG-ассистента супермаркета на запрос покупателя о товарах.*

*Оцените ответ по одной метрике.*

### Критерии оценки

*Фактологическая точность (1–5) — соответствие ответа базе товаров:*

- название и бренд*
- цена и объём*
- наличие на складе*
- отсутствие выдуманных позиций*

### Шаги оценки

- 1. Прочитать запрос и контекст из базы.*
- 2. Сверить ответ ассистента с контекстом.*
- 3. Поставить оценку от 1 до 5: 1 — ответ противоречит базе, 5 — всё подтверждено.*



## Инициализация LLM Judge

```
import os
from deepeval.models.base_model import DeepEvalBaseLLM
from openai import OpenAI
from dotenv import load_dotenv

load_dotenv()

class CustomOpenAIModel(DeepEvalBaseLLM):
    def __init__(self, model_name: str, base_url: str, api_key: str):
        ...
    def load_model(self):
        ...
    def generate(self, prompt: str) -> str:
        ...
    async def a_generate(self, prompt: str) -> str:
        ...
    def get_model_name(self) -> str:
        ...

# Инициализируем judge-модель
judge_model = CustomOpenAIModel(
    model_name=os.getenv("OPENAI_MODEL"),
    base_url=os.getenv("OPENAI_BASE_URL"),
    api_key=os.getenv("OPENAI_API_KEY"),
)
```

## Инициализация LLM Judge

```
import os
from deepeval.models.base_model import DeepEvalBaseLLM
from openai import OpenAI
from dotenv import load_dotenv

load_dotenv()

class CustomOpenAIModel(DeepEvalBaseLLM):
    def __init__(self, model_name: str, base_url: str, api_key: str):
        ...
    def load_model(self):
        ...
    def generate(self, prompt: str) -> str:
        ...
    async def a_generate(self, prompt: str) -> str:
        ...
    def get_model_name(self) -> str:
        ...

# Инициализируем judge-модель
judge_model = CustomOpenAIModel(
    model_name=os.getenv("OPENAI_MODEL"),
    base_url=os.getenv("OPENAI_BASE_URL"),
    api_key=os.getenv("OPENAI_API_KEY"),
)
```

```
from deepeval.metrics import AnswerRelevancyMetric, ContextualRecallMetric
from custom_model import judge_model

# --- Метрика 1: Answer Relevancy ---
answer_relevancy = AnswerRelevancyMetric(
    threshold=0.5,
    model=judge_model,
)

# --- Метрика 2: Contextual Recall ---
contextual_recall = ContextualRecallMetric(
    threshold=0.5,
    model=judge_model,
)
```

```
from deepeval.metrics import GEval

from custom_model import judge_model

# --- Метрика 3: GEval ---
correctness = GEval(
    name="Correctness",
    criteria=(
        "Оцени, насколько actual_output корректно и полно отвечает на вопрос "
        "по сравнению с expected_output. Учитывай фактическую точность: "
        "цены, названия, характеристики товаров должны совпадать."
    ),
    evaluation_params=[
        LLMTestCaseParams.INPUT,
        LLMTestCaseParams.ACTUAL_OUTPUT,
        LLMTestCaseParams.EXPECTED_OUTPUT,
    ],
    threshold=0.5,
    model=judge_model,
)
```

```
from deepeval.metrics import GEval

from custom_model import judge_model

# --- Метрика 3: GEval ---
correctness = GEval(
    name="Correctness",
    criteria=(
        "Оцени, насколько actual_output корректно и полно отвечает на вопрос "
        "по сравнению с expected_output. Учитывай фактическую точность: "
        "цены, названия, характеристики товаров должны совпадать."
    ),
    evaluation_params=[
        LLMTestCaseParams.INPUT,
        LLMTestCaseParams.ACTUAL_OUTPUT,
        LLMTestCaseParams.EXPECTED_OUTPUT,
    ],
    threshold=0.5,
    model=judge_model,
)
```

Собираем всё вместе


Собираем все вместе – получаем pipeline тестирования



Собираем все вместе – получаем pipeline тестирования



Датасет




Тестовые  
примеры

Собираем все вместе – получаем pipeline тестирования



Датасет



Тестовые  
примеры



Метрики

# Процесс тестирования с DeepEval



Собираем все вместе – получаем pipeline тестирования



```
from deepeval import evaluate
from deepeval.dataset import EvaluationDataset, Golden
from deepeval.metrics import AnswerRelevancyMetric, ContextualRecallMetric, GEval
from deepeval.test_case import LLMTestCase, LLMTestCaseParams

# Создание датасета из Golden-примеров
dataset = EvaluationDataset(goldens=[...])

# Инициализация метрик
answer_relevancy = AnswerRelevancyMetric(...)
contextual_recall = ContextualRecallMetric(...)
correctness = GEval(...)

# Тестовые примеры
test_cases = []
for golden in dataset.goldens:
    res, chunks = rag_pipeline(golden.input)
    test_cases.append(LLMTestCase(...))

# Запуск оценки
evaluate(
    test_cases=test_cases,
    metrics=[answer_relevancy, contextual_recall, correctness]
)
```

Собираем все вместе – получаем pipeline тестирования



```
from deepeval import evaluate
from deepeval.dataset import EvaluationDataset, Golden
from deepeval.metrics import AnswerRelevancyMetric, ContextualRecallMetric, GEval
from deepeval.test_case import LLMTestCase, LLMTestCaseParams
```

```
# Создание датасета из Golden-примеров
dataset = EvaluationDataset(goldens=[...])
```

```
# Инициализация метрик
answer_relevancy = AnswerRelevancyMetric(...)
contextual_recall = ContextualRecallMetric(...)
correctness = GEval(...)
```

```
# Тестовые примеры
test_cases = []
for golden in dataset.goldens:
    res, chunks = rag_pipeline(golden.input)
    test_cases.append(LLMTestCase(...))
```

```
# Запуск оценки
evaluate(
    test_cases=test_cases,
    metrics=[answer_relevancy, contextual_recall, correctness]
)
```

Собираем все вместе – получаем pipeline тестирования



```
from deepeval import evaluate
from deepeval.dataset import EvaluationDataset, Golden
from deepeval.metrics import AnswerRelevancyMetric, ContextualRecallMetric, GEval
from deepeval.test_case import LLMTestCase, LLMTestCaseParams

# Создание датасета из Golden-примеров
dataset = EvaluationDataset(goldens=[...])

# Инициализация метрик
answer_relevancy = AnswerRelevancyMetric(...)
contextual_recall = ContextualRecallMetric(...)
correctness = GEval(...)

# Тестовые примеры
test_cases = []
for golden in dataset.goldens:
    res, chunks = rag_pipeline(golden.input)
    test_cases.append(LLMTestCase(...))

# Запуск оценки
evaluate(
    test_cases=test_cases,
    metrics=[answer_relevancy, contextual_recall, correctness]
)
```

Собираем все вместе – получаем pipeline тестирования



```
from deepeval import evaluate
from deepeval.dataset import EvaluationDataset, Golden
from deepeval.metrics import AnswerRelevancyMetric, ContextualRecallMetric, GEval
from deepeval.test_case import LLMTestCase, LLMTestCaseParams

# Создание датасета из Golden-примеров
dataset = EvaluationDataset(goldens=[...])

# Инициализация метрик
answer_relevancy = AnswerRelevancyMetric(...)
contextual_recall = ContextualRecallMetric(...)
correctness = GEval(...)

# Тестовые примеры
test_cases = []
for golden in dataset.goldens:
    res, chunks = rag_pipeline(golden.input)
    test_cases.append(LLMTestCase(...))

# Запуск оценки
evaluate(
    test_cases=test_cases,
    metrics=[answer_relevancy, contextual_recall, correctness]
)
```

Собираем все вместе – получаем pipeline тестирования

```
from deepeval import evaluate
from deepeval.dataset import EvaluationDataset, Golden
from deepeval.metrics import AnswerRelevancyMetric, ContextualRecallMetric, GEval
from deepeval.test_case import LLMTestCase, LLMTestCaseParams

# Создание датасета из Golden-примеров
dataset = EvaluationDataset(goldens=[...])

# Инициализация метрик
answer_relevancy = AnswerRelevancyMetric(...)
contextual_recall = ContextualRecallMetric(...)
correctness = GEval(...)

# Тестовые примеры
test_cases = []
for golden in dataset.goldens:
    res, chunks = rag_pipeline(golden.input)
    test_cases.append(LLMTestCase(...))

# Запуск оценки
evaluate(
    test_cases=test_cases,
    metrics=[answer_relevancy, contextual_recall, correctness]
)
```

## Запуск

```
python .\run_evaluation.py
```

```
=====  
  
Overall Metric Pass Rates
```

```
Answer Relevancy: 60.00% pass rate
```

```
Contextual Recall: 100.00% pass rate
```

```
Correctness [GEval]: 100.00% pass rate  
  
=====
```

## Metrics Summary

- ❌ Answer Relevancy (score: 0.14285714285714285, threshold: 0.5, strict: False, evaluation model: google/gemini-2.5-flash-lite-preview-09-2025, reason: The score is 0.14 because the entire output consisted only of greetings, general pleasantries, feature descriptions (sound quality, battery life, transparency mode), and closing remarks, completely failing to address the user's core question about temperature-maintaining kettles, error: None)
- ✅ Contextual Recall (score: 1.0, threshold: 0.5, strict: False, evaluation model: google/gemini-2.5-flash-lite-preview-09-2025, reason: The score is 1.00 because the expected output sentence, stating that the SteelKettle 1.7L electric kettle maintains a temperature between 70-95°C, is directly verifiable via the information found in node 1 of the node(s) in retrieval context., error: None)
- ✅ Correctness [GEval] (score: 0.6, threshold: 0.5, strict: False, evaluation model: google/gemini-2.5-flash-lite-preview-09-2025, reason: The Actual Output correctly identifies the product that supports temperature ('Электрический чайник «SteelKettle 1.7L»') and includes the temperature range (70°C до 95°C), aligning with the factual details in the Expected Output. However, the Actual Output is significantly more verbose and introduces an unrelated product (a thermos), failing to meet the concise standard set by the Expected Output, which only required the single sentence answer., error: None)

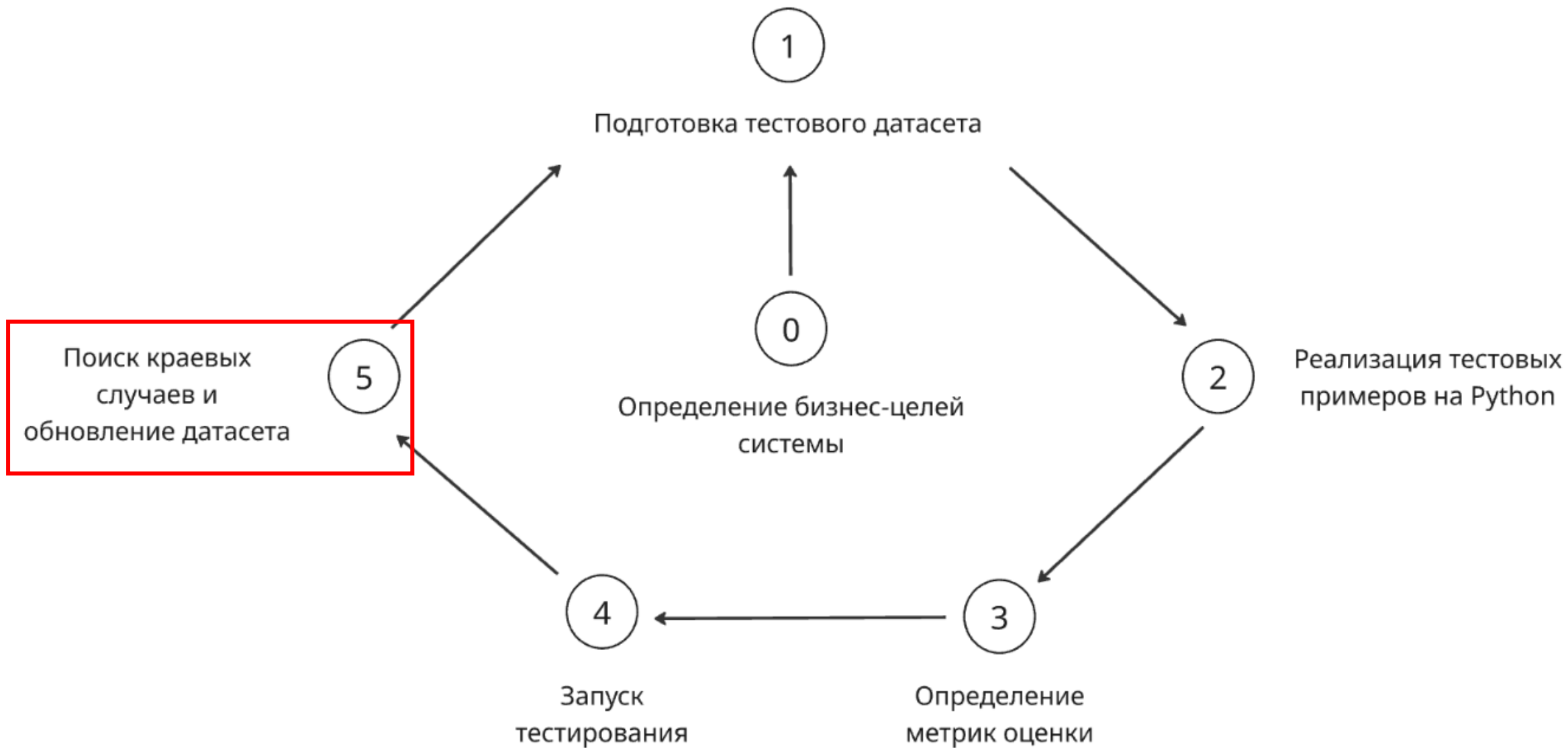
For test case:

Наушники

- input: Какой чайник поддерживает температуру?
- actual output: Здравствуйте! Рад помочь вам с выбором.

Насколько я понимаю, вы ищете прибор, который может поддерживать определённую температуру напитка.

# Процесс тестирования с DeepEval



# Confident-AI – графический интерфейс для отслеживания результатов тестирования

123 Free / My first project / Connect AI App

Feedback Docs Get Support

## Test Cases

All test cases for this particular test run.

All Passed Failed

**PASSED 80.00% | 4/5 test cases** **FAILED 20.00% | 1/5 test cases**

### Eval Insights

2 mild issues 1 serious issue No hyperparameters logged

Overview

- LLM outputs include unnecessary conversational elements.
- Responses provide excessive detail beyond expectations.
- Retrieval context supports accurate product details.
- Extraneous information included in LLM responses.

Save as new dataset Download all as CSV

SELECT FILTERS

+ Add

Showing 1 to 5 of 5 test case(s)

Name	Status	Input	Actual Output
test_case_0	Success	Сколько стоят наушники Listen Pro?	Здравствуйте! Рад помочь вам с выбором. Наушники <b>Listen Pro</b> стоят <b>8990 рублей</b> . Это полнораз...
test_case_1	Failed	Из чего состоит гель для душа Морская свежесть?	Здравствуйте! Рад помочь вам с выбором. Гель для душа «Морская свежесть» состоит из следующих к...
test_case_2	Success	Какой есть зеленый чай?	Здравствуйте! Рад помочь вам с выбором. В нашем ассортименте есть один вид зеленого чая — это <b>O...</b>
test_case_3	Success	Что умеет умная колонка SoundBox Mini?	Здравствуйте! Рад помочь вам с выбором. Умная колонка <b>SoundBox Mini</b> — это компактный умный...
test_case_4	Success	Какой чайник поддерживает температуру?	Здравствуйте! Благодарю за Ваш вопрос. У нас есть электрический чайник, который поддерживает зада...

Rows per page 10 1-5 of 5 Page 1 of 1

# Confident-AI – графический интерфейс для отслеживания результатов тестирования

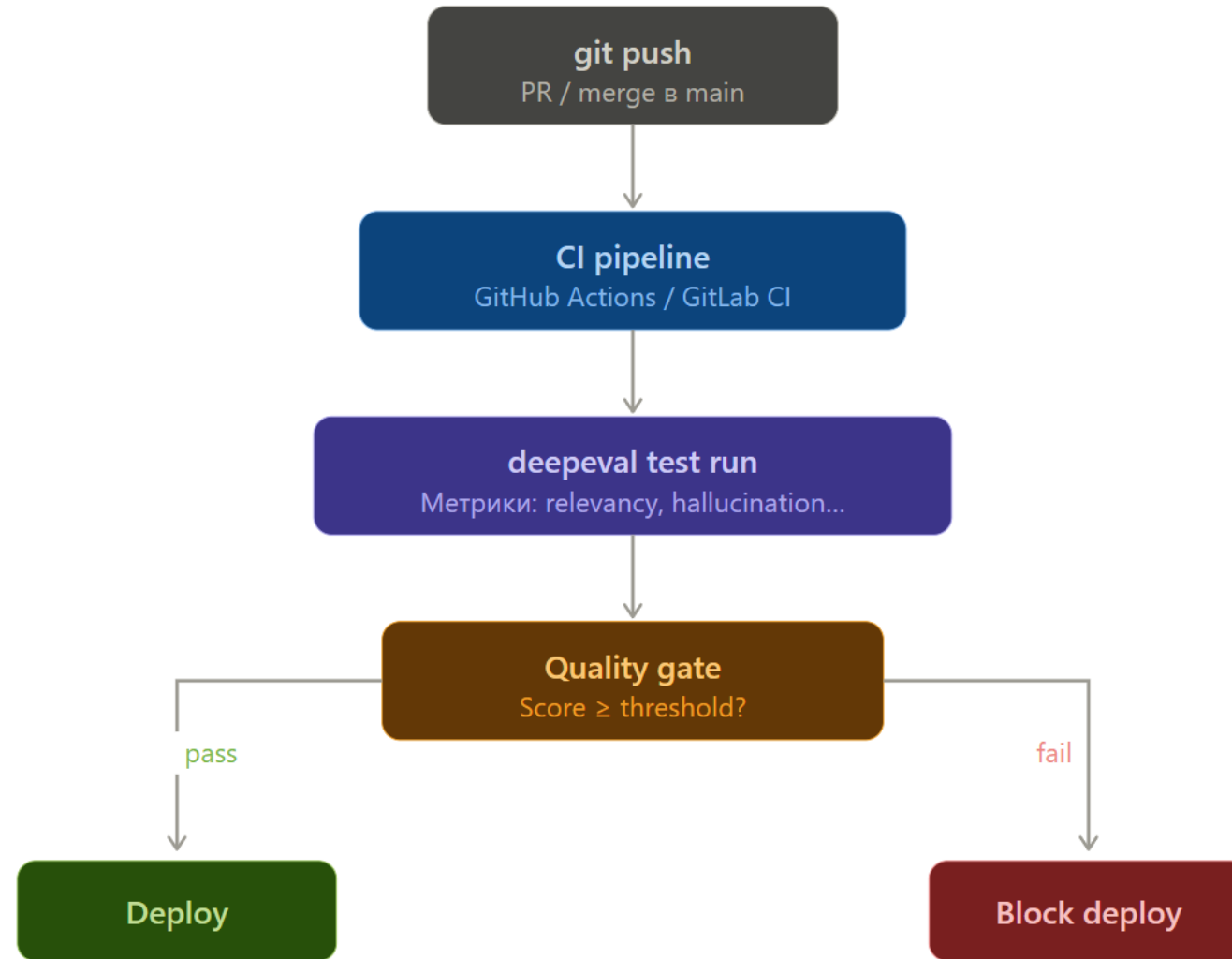
The screenshot displays the Confident-AI interface for a failed test case. The interface is dark-themed and organized into several sections:

- Input:** A text box containing the question: "Из чего состоит гель для душа Морская свежесть?".
- Actual Output:** A text box containing the AI's response: "Здравствуйте! Рад помочь вам с выбором. Гель для душа «Морская свежесть» состоит из следующих компонентов: вода, ПАВ (кокамидопропилбетаин), экстракт морских водорослей, ментол, парфюмерная композиция и консервант. Этот гель, кстати, освежающий, содержит экстракт морских водорослей и ментол, что делает его идеальным для бодрости и упругости кожи. Аромат у него — смесь цитрусовых и океанических нот. Если у вас возникнут ещё вопросы по товарам, пожалуйста, обращайтесь!".
- Expected Output:** A text box containing the expected response: "Гель для душа «Морская свежесть» содержит воду, ПАВ (кокамидопропилбетаин), экстракт морских водорослей, ментол, парфюмерную композицию и консервант.". The word "Expected" is misspelled as "Expected".
- Retrieval Context:** A section for additional context, currently empty.
- Test Case Details:** A table on the right side of the interface provides the following information:
  - Test Case ID: 9d26afea-8c14-4533-a3db-9d4150481282
  - Name: test\_case\_1
  - Status: Failed
  - Order: 2
  - Run Duration: 14.57s
  - Metrics Evaluated: 3
- Metrics:** A section below the test case details shows the "Answer Relevancy" metric with a score of 0.25 and a status of "Failed". The evaluation model used is "google/gemini-2.5-flash-lite-preview-09-2025" and the threshold is 0.5.
- Reason:** A text box explaining the failure: "The score is 0.25 because the entire output consisted only of greetings, acknowledgments, descriptions of the gel's effect/scent, and closing phrases, completely failing to provide any information about the actual chemical composition of the shower gel requested by the input."
- Actions:** A button labeled "Show verbose logs" is located at the bottom of the metrics section.

Confident-AI – графический интерфейс для отслеживания результатов тестирования

Один из основных минусов – только облачное решение. Для конфиденциальных данных не подойдет.

# Использование DeepEval в CI/CD





```
name: RAG Evaluation
```

```
on:
```

```
  push:
```

```
    branches: [main]
```

```
  pull_request:
```

```
    branches: [main]
```

```
  workflow_dispatch:
```

```
env:
```

```
  PYTHON_VERSION: "3.12"
```

```
  EMBEDDING_MODEL: "deepvk/USER-base"
```

```
jobs:
  evaluate:
    runs-on: ubuntu-latest

    steps:
      - uses: actions/checkout@v4

      - name: Set up Python
        uses: actions/setup-python@v5
        with:
          python-version: ${ env.PYTHON_VERSION }

      - name: Cache pip packages
        uses: actions/cache@v4
        with:
          path: ~/.cache/pip
          key: pip-${ runner.os }-${ hashFiles('requirements.txt') }
          restore-keys: pip-${ runner.os }-
```

```
- name: Cache embedding model
  uses: actions/cache@v4
  with:
    path: ~/.cache/huggingface
    key: hf-{{ env.EMBEDDING_MODEL }}
    restore-keys: hf-

- name: Download embedding model
  run: python -c "from sentence_transformers import SentenceTransformer; SentenceTransformer('{{ env.EMBEDDING_MODEL }}')"
```

```
- name: Run DeepEval tests
  env:
    OPENAI_API_KEY: {{ secrets.OPENAI_API_KEY }}
    OPENAI_BASE_URL: {{ secrets.OPENAI_BASE_URL }}
    OPENAI_DEFAULT_MODEL: {{ secrets.OPENAI_DEFAULT_MODEL }}
  run: deepeval test run test_rag.py
```

The screenshot shows a GitHub Actions workflow run for the repository 'maksimov-m / deepeval\_rag\_test'. The workflow is named 'change ci cd #3' and is in a failed state. The run was triggered via a push 18 minutes ago. The status is 'Failure', the total duration is '4m 23s', and there are no artifacts. The workflow file is 'rag-evaluation.yml', which is triggered on a push. The 'evaluate' step failed with an exit code of 1. The left sidebar shows the workflow file and run details.

← RAG Evaluation

**change ci cd #3**

Summary

All jobs

**evaluate**

Run details

Usage

Workflow file

Triggered via push 18 minutes ago

Status	Total duration	Artifacts
<b>Failure</b>	<b>4m 23s</b>	—

**rag-evaluation.yml**  
on: push

**evaluate** 4m 18s

**Annotations**  
1 error and 1 warning

**evaluate**  
Process completed with exit code 1.

Summary

All jobs

**evaluate**

Run details

Usage

Workflow file

**evaluate**

failed 2 minutes ago in 3m 26s

Search logs

53s

Run DeepEval tests

```

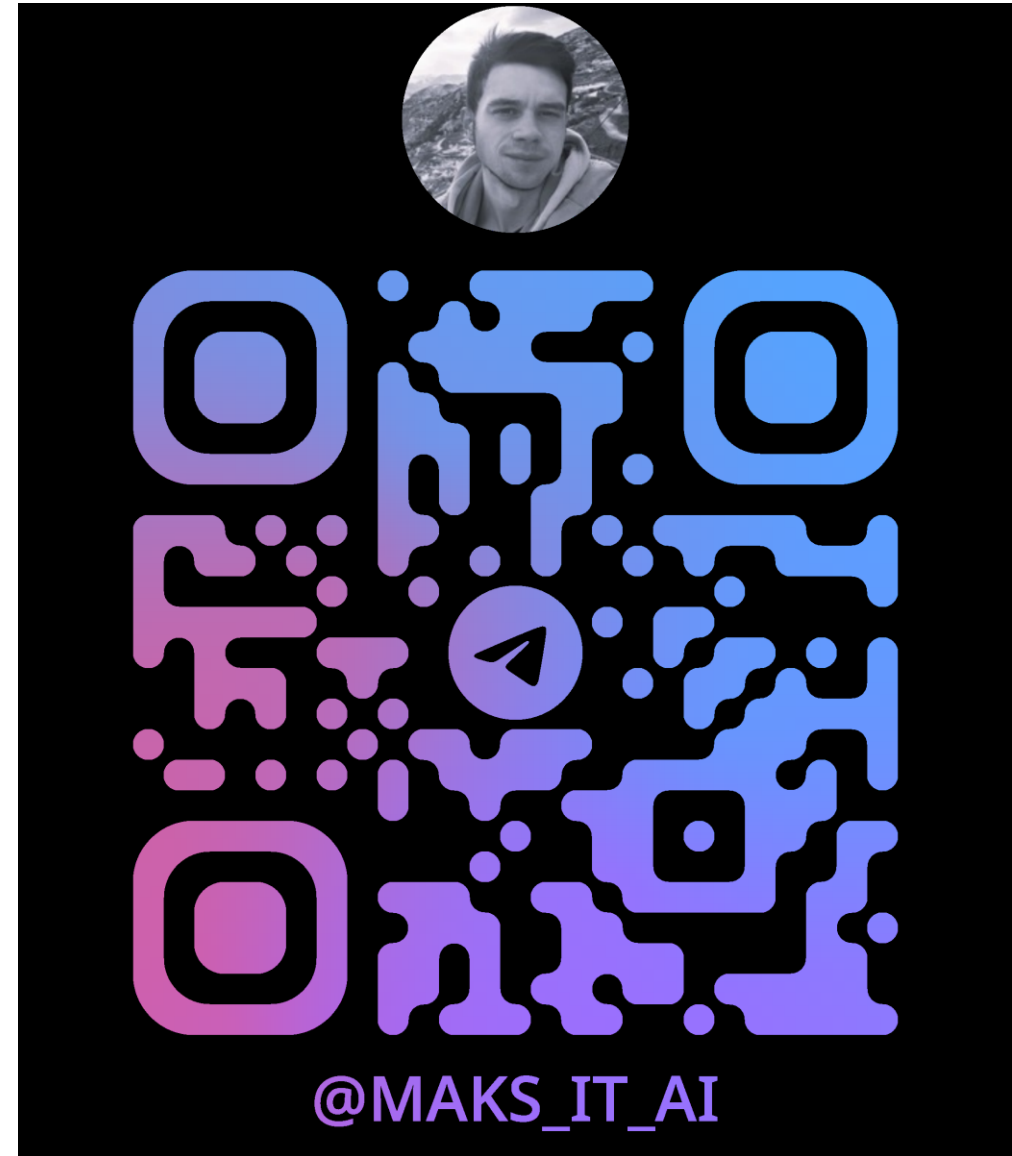
382 (3 warnings < 0.0001 seconds) use --vv to show these warnings.)
383 ===== short test summary info =====
384 FAILED test_rag.py::test_rag[golden0] - ValueError: Evaluation LLM outputted an invalid JSON. Please use a better evaluation model.
385 FAILED test_rag.py::test_rag[golden4] - AssertionError: Metrics: Correctness [GEval] (score: 0.4, threshold: 0.5, strict: False, error: None, reason: The Actual Output is not factually aligned with the Expected
Output as a benchmark for correctness (Step 2, 3). While the Actual Output mentions the 'Электрический чайник «SteelKettle 1.7L»' and its temperature range (70°C to 95°C), it includes extraneous information,
specifically the 'Термос с двойными стенками' and an opening greeting/closing question, making it incomplete relative to the direct, concise nature required by the Expected Output (Step 1, 4). The core detail about
the electric kettle is present, but the overall format and unnecessary additions detract from full correctness.) failed.
386 2 failed, 3 passed, 15 warnings in 48.31s
387
388 Test Results
389
390 | Test case | Metric | Score | Status | Overall
391 |-----|-----|-----|-----|-----|-----
392 | test_rag | | | | | 66.67%
393 | | Answer | 1.0 | PASSED |
394 | | Relevancy | (threshold=0.5, |
395 | | | evaluation |
396 | | | model=google/g- |
397 | | | reason=The |
398 | | | score is 1.00 |
399 | | | because there |
400 | | | are no |
401 | | | irrelevant |
402 | | | statements |
403 | | | present in the |
404 | | | output, |
405 | | | indicating a |
406 | | | perfect match |
407 | | | to the input |
408 | | | query., |
409 | | | error=None) |
410 | | Contextual | 1.0 | PASSED |
411 | | Recall | (threshold=0.5, |
412 | | | evaluation |
413 | | | model=google/g- |
414 | | | reason=The

```

825		Correctness	0.4	FAILED		
826		[GEval]	(threshold=0.5,			
827			evaluation			
828			model=google/g..			
829			reason=The			
830			Actual Output			
831			is not			
832			factually			
833			aligned with			
834			the Expected			
835			Output as a			
836			benchmark for			
837			correctness			
838			(Step 2, 3).			
839			While the			

1. Большая, дорогая модель в качестве судьи - не значит лучшая.
2. Работа с LLM-приложениями требует от тестировщиков новых навыков, которые необходимо осваивать уже сейчас.
3. Подбирайте метрики в соответствии с «узким» местом своей системы.

Спасибо за внимание!



Здесь много интересного!