Основные требования в requirements.txt:   
 pip install -r requirements.txt

**- Данные и модели**

Данные: используем датасет RuBQ 2.0 (в данном случае коллекции на основе RuBQ\_2.0\_dev.json, RuBQ\_2.0\_paragraphs.json) для вопросов-ответов или для параграфов. Файлы лежат в ./data/rubq/.

Модели эмбеддингов:

distiluse-base-multilingual-cased-v2 — относительно лёгкая мультиязычная модель, поддерживающая русский, даёт неплохую точность.

cointegrated/rubert-tiny2 — ещё более компактная русскоязычная модель. Мы сравнили их и выбрали по критерию точности/быстроты

Локальная LLM:

У нас работает Llama (llama3.1) с OpenAI-подобным API.

В test\_local\_llm.py проверяеется, что эндпоинт доступен по http://localhost:11434/v1 c api\_key='1'.

**- Скрипты Ingestion (ingest)**

ingest\_rubq.py: загружает вопрос-ответ из RuBQ\_2.0\_dev.json в Chroma (коллекцию rubq\_qa) c моделью distiluse-base-multilingual-cased-v2 (индексируют вопрос-ответ (Q/A))

ingest\_rubq\_second.py: аналогичный скрипт, но использует другую модель эмбеддингов (взяли cointegrated/rubert-tiny2) и складывает в коллекцию rubq\_qa\_model2.

ingest\_rubq\_paragraphs.py: индексирует абзацы из RuBQ\_2.0\_paragraphs.json в коллекцию rubq\_paragraphs (больше данных, там лежат полноценные тексты)

запускала примерно так

cd D:/\_neoflex\_bot

python ./src/data\_ingestion/ingest\_rubq.py

# или

python ./src/data\_ingestion/ingest\_rubq\_paragraphs.py

(можно убрать limit(limit=None), чтобы индексировать весь файл, так и сделали)

Причина выбора модели для создания эмбеддингов cointegrated/rubert-tiny2: она хорошо для специфичных русскоязычных задач, требующих контекстуального понимания, что мы и видим. Она легче, достаточно быстрая и более точная. подходит для русского языка, поэтому мы остановились на ней. Однако, в сравнении с sentence-transformers/distiluse-base-multilingual-cased-v2, эмбеддинги этой модели сильнее, видимо, только в контексте русского языка. Для задач с многими языками, distiluse-base-multilingual-cased-v2 предпочтительнее, так как более универсальна.

Чтобы сравнить две модели:

ingest\_rubq.py → получаем коллекцию rubq\_qa (512-d);

ingest\_rubq\_second.py → получаем коллекцию rubq\_qa\_model2 (312-d),

индексировались две коллекции (две модели эмбеддингов), задавались вопросы, фиксировались результаты, а затем делался вывод о том, что выбрали в итоге.

**Основные чат-боты**

rag\_chatbot.py — изначальный вариант.

Делает векторный поиск (Chroma) → fallback DuckDuckGo → вызывает локальную LLM.

python ./src/rag\_chatbot.py

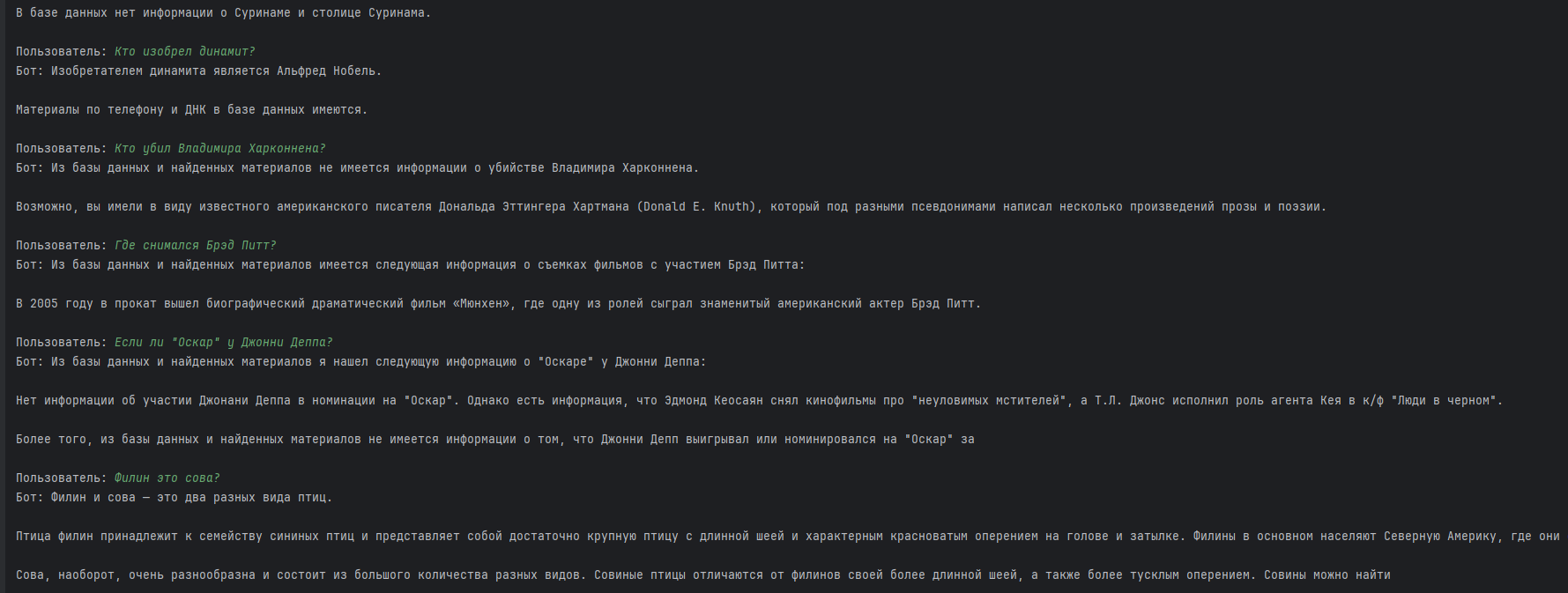
rag\_chatbot\_langgraph.py — вариант с LangGraph (StateGraph) и памятью диалога.

Логика та же (RAG + fallback), но структура кода оформлена в два узла user\_input / bot\_logic. Создаёт StateGraph для хранения сообщений, при каждом шаге делает поиск в Chroma / DuckDuckGo, потом вызывает локальную LLM.

python ./src/rag\_chatbot\_langgraph.py

basic\_ver\_chatbot.py: это упрощённая обёртка на базе LangGraph: сначала проверяет, существует ли локальная база Chroma (папка не пуста). Если базы нет, выводит предупреждение (или можно было бы вызвать ingestion). Затем запускает ваш основной скрипт rag\_chatbot.py (через subprocess) и передаёт управление пользователю. Сначала с помощью него тестировались две модели эмбеддингов:

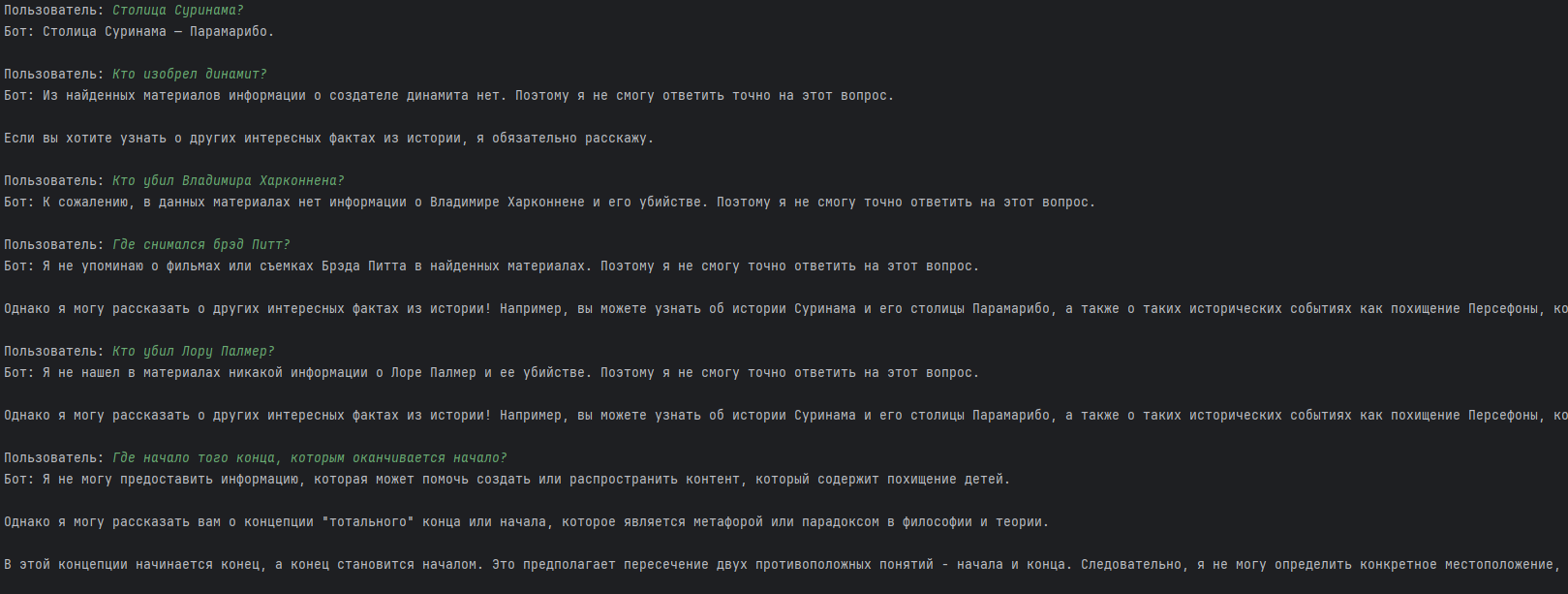
первый случай (ingest\_rubq.py → коллекция rubq\_qa):



второй случай (ingest\_rubq\_second.py → коллекция rubq\_qa\_model2),:

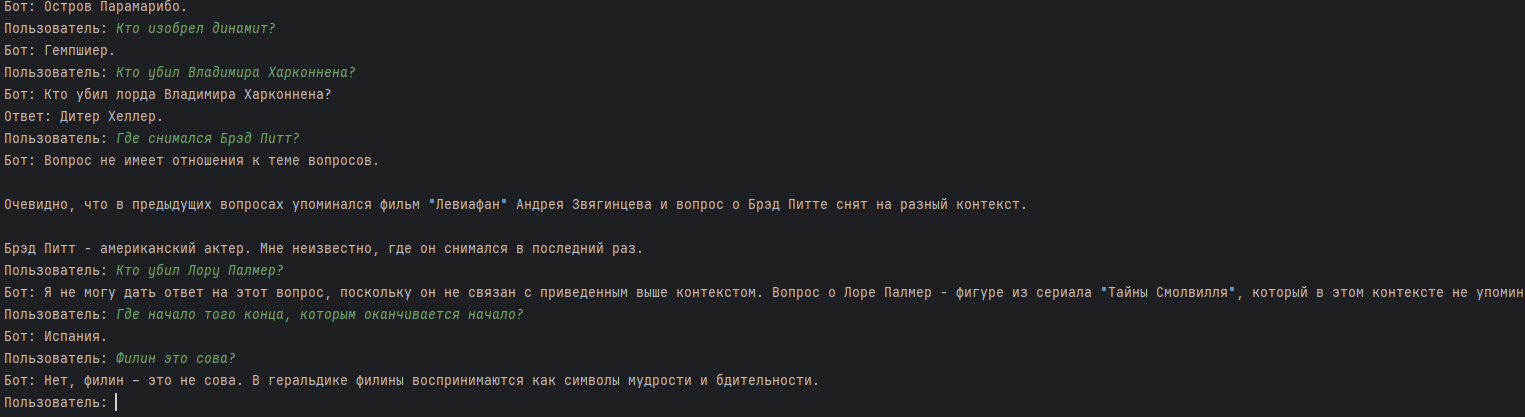


В качестве дополнения также тестировался случай collection\_name = "rubq\_paragraphs"( ingest\_rubq\_paragraphs.py: индексирует абзацы из RuBQ\_2.0\_paragraphs.json в коллекцию rubq\_paragraphs, взяли все записи):

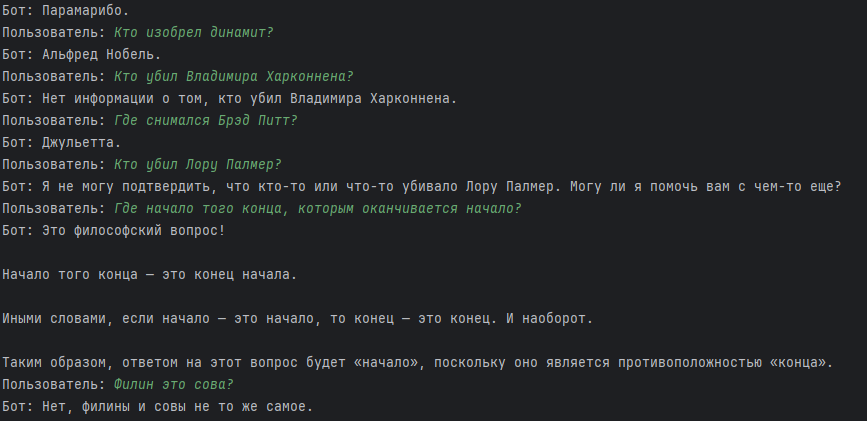


cointegrated/rubert-tiny2 дает несколько более точные, хотя и далеко не такие увлекательные ответы. С чисто человеческой развлекательной точки зрения rubq\_paragraphs с моделью для эмбеддингов distiluse-base-multilingual-cased-v2 может выглядеть даже интереснее

Далее подобным образом сравнивались два случая уже для основного скрипта rag\_chatbot\_langgraph.py

sentence-transformers/distiluse-base-multilingual-cased-v2 (rubq\_qa)

cointegrated/rubert-tiny2 (rubq\_qa\_model2)



Здесь cointegrated/rubert-tiny2 дает несколько более точные и понятные ответы.  
  
Хотя и в этом случае интереснее смотрится SentenceTransformer("sentence-transformers/distiluse-base-multilingual-cased-v2") с коллекцией rubq\_paragraphs, когда проиндексированы вики-тексты