Документация

v.0.1

RuTubeCopilot

Команда 16K20 для кейса «RUTUBE»

2024

Оглавление

1. Краткое описание

RuTubeCopilot — это интеллектуальный помощник для операторов поддержки RuTube, созданный для значительного ускорения обработки запросов пользователей платформы. С помощью продвинутых моделей машинного обучения, обученных на базе знаний FAQ и документации RuTube, система предлагает точные ответы в течение 5-10 секунд, что сокращает время обработки запросов в несколько раз!

Наш стек: multilingual-e5-large, Phi-3, aiogram, ruby/ror, vue.js

Пользователи задают вопросы в Telegram-боте, а оператор через удобную веб-панель выбирает оптимальный вариант ответа или корректирует его. RuTubeCopilot не только сможет повысить скорость и качество работы операторов поддержки, но и сделает видеохостинг еще удобнее и привлекательнее для всех.

Посмотреть реализованные сервис можно по ссылкам ниже:

Telegram-бот для вопросов от пользователей, может работать в режиме пользователя и тестовом режиме оператора (подробности ниже)

https://t.me/rutube_hack_bot

Web-админка для оператора поддержки (можно войти через режим гостя)

https://rutube-copilot.kovalev.team

Внимания! Ответ системы может занимать до 15 с, а если очень много пользователей будут запрашивать одновременно, то сообщения попадают в очередь (чтобы не перенагружать тестовую систему)

2. Модельки, работа с данными

2.1 Обучение эмбедингов

Для дообучения эмбедингов изначально был принят вариант использование BERT, но слишком много классов и слишком мало данных, поэтому был выбран другой вариант.

В качестве мы взяли Phi3-Medium 6-битная квантизация, с помощью подбора мы нашли лучшее соотношение скорость/качество. Она имеет лицензию MIT, значит ее можно использовать в том числе коммерческих целях.

В базу знаний мы добавили образцовые ответы, реальные ответы сотрудников и разделенные по абзацам документации, а также сгенерированные ответы finetuned e5-large с помощью констрактивного обучения, негативы подбирали с помощью e5, брали топ 10 контекстов, правильный относили к positive, а остальные 9 к negotive.

https://huggingface.co/dankalin/multilingual-e5-large-hack

Что получилось по метрикам:

До	После
accuracy@1: 0.151061	accuracy@1: 0.28714
accuracy@3: 0.40199	accuracy@3: 0.73782
accuracy@5: 0.50312	accuracy@5:0.88764
accuracy@10: 0.61548	accuracy@10: 0.96754
precision@1: 0.15106	precision@1:0.28714
precision@3: 0.13399	precision@3: 0.245942
precision@5: 0.10062	precision@5: 0.17752
precision@10: 0.06154	precision@10: 0.09675
recall@1': 0.15106	recall@1': 0.28714

recall@3': 0.40199	recall@3': 0.73782
recall@5': 0.50312	recall@5': 0.8876
recall@10': 0.61548	recall@10': 0.9675

- С помощью исходного датасета сгенерировали доп. пары вопрос-ответ (было 1200, стало 3300)
- Обучали contrastive learning на positive-negative
- Anchor - вопрос
- Positive - правильный ответ
- Negative - неправильно подобранный контекст (e5)
- Время ответа: 0.7 сек.

2.2 База знаний

- Образцовые ответы
- Реальные ответы сотрудников
- Разделенные по абзацам «Пользовательское соглашение» и «Условия размещения»

3. Структура системы

3.1 Общая структура системы

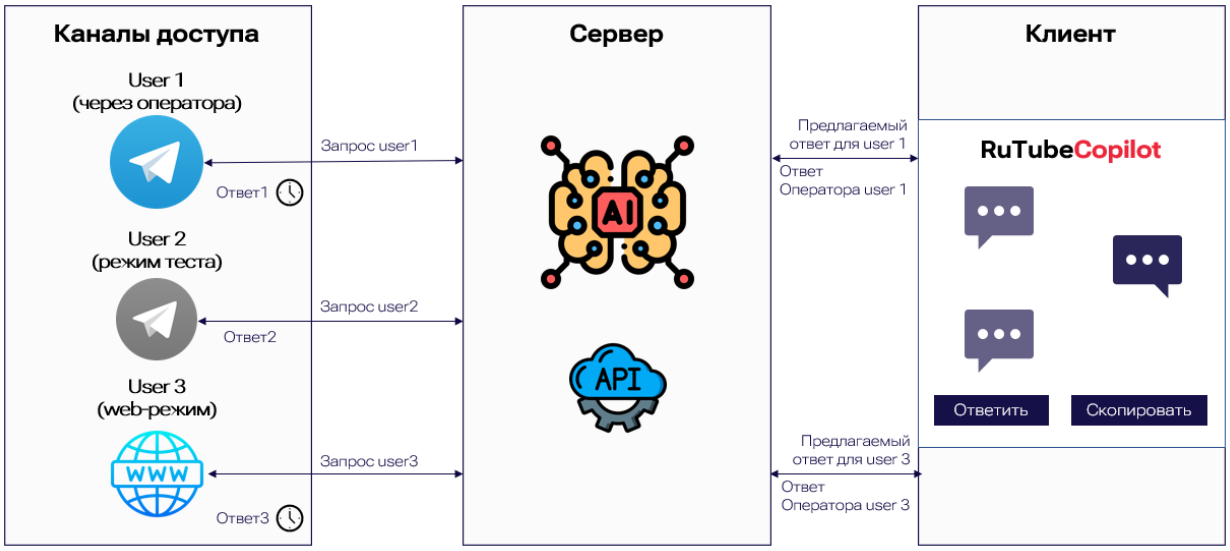

Рисунок 1 Общая структура системы

В системе RuTubeCopilot реализована ролевая модель: пользователь, оператор поддержки и админ.

1. В **стандартном варианте** пользователь, используя Телеграмм бот, делает запрос (пишет интересующий его вопрос). Этот вопрос отправляется на сервер, где используется обученная модель и генерируется ответ, который оператор может выбрать в соответствующей этому пользователю комнате. Оператор видит вопрос и предлагаемый вариант. Если вариант подходящий, то можно ответить, если нет – то оператор через текстовое поле может скорректировать ответ. При вводе собственного ответа работает spellchecker. Пользователь, получив ответ, может проголосовать за него, понравился ответ или нет.
2. В **тестовом варианте** через Телеграм бот в качестве пользователя выступает оператор-админ. Переключиться между ролями можно по кнопке. В режиме оператора-админа ответ сразу приходит от сервиса и не поступает на рассмотрение оператору.

3. Проведение тестирование с помощью API. Отправляет запрос и в ответ приходит json.

3.2 Структура сервера

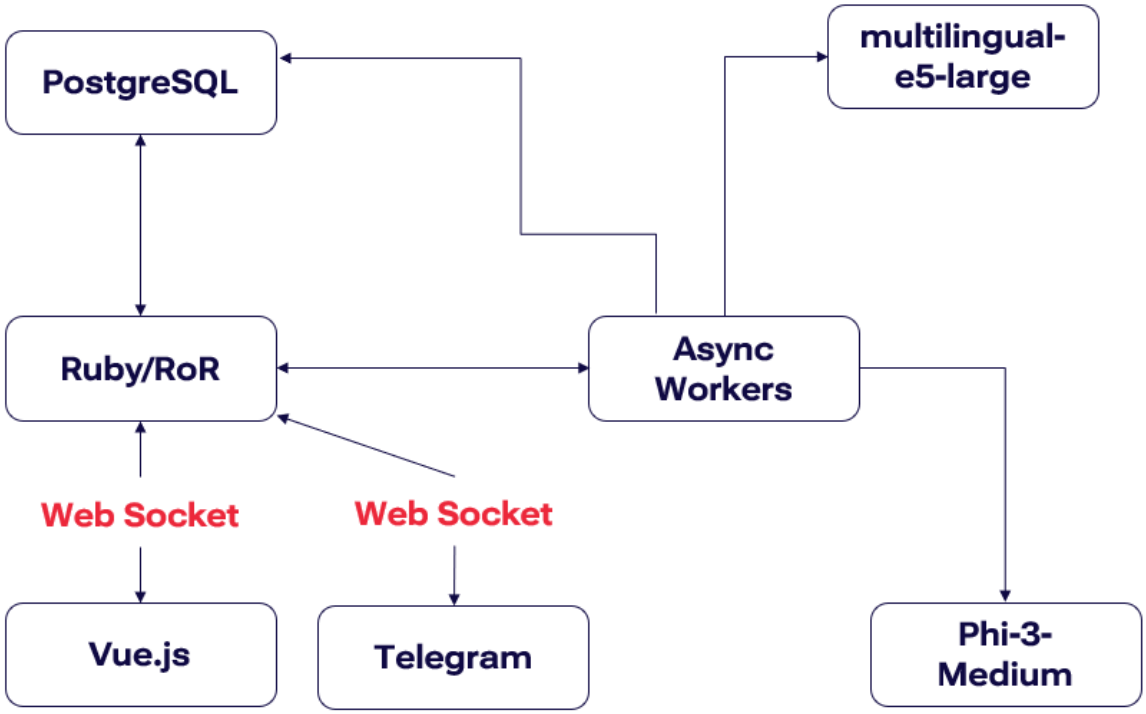

Рисунок 2 Структура сервера

1. **Основные компоненты:**

- PostgreSQL — база данных, в которой хранится вся необходимая информация для работы сервера, включая результаты работы моделей и информацию о запросах.

- Ruby/RoR — основное серверное приложение, написанное на Ruby on Rails, взаимодействует с базой данных, фронтендом и обработчиками асинхронных задач.

- Vue.js — фронтенд-приложение, через которое операторы поддержки взаимодействуют с системой через веб-интерфейс. Связь с сервером осуществляется через WebSocket.

- Telegram — бот, через который пользователи RuTube задают свои вопросы. Также использует WebSocket для связи с сервером.

- Async Workers — асинхронные воркеры, которые занимаются обработкой запросов пользователей и взаимодействием с моделями машинного обучения для генерации ответов.

- multilingual-e5-large и Phi-3-Medium — модели машинного обучения, обученные на базе знаний RuTube. Эти модели используются для генерации ответов на запросы пользователей, обеспечивая быструю и точную обработку информации.

2. Процесс работы:

- Пользователь задает вопрос через Telegram-бот.

- Вопрос передается на сервер (Ruby/RoR) через WebSocket.

-Ruby/RoR взаимодействует с асинхронными воркерами, которые направляют запрос к моделям машинного обучения (multilingual-e5-large или Phi-3-Medium) для генерации ответа.

- Результаты работы моделей сохраняются в базе данных PostgreSQL.

- Оператор в веб-интерфейсе (на базе Vue.js) получает предложенные варианты ответов через WebSocket.

- Оператор может выбрать или откорректировать ответ, который затем отправляется пользователю через Telegram.

Запустить решение, которое лежит по ссылке https://github.com/ilya-edu/hackrutu/tree/master/backend можно так:

bash docker-compose down && docker-compose up --build

Подготовка Данных

1. запустите приложение командой выше и откройте [localhost:3035/articles/new]()

2. Загрузите в форму файл с статьями который лежит в репо lib/merged_output.csv

3. Откройте админку и можете видеть, что создаются статьи http://localhost:3035/admin/article

4. Проверьте что работает базовый RAG http://localhost:3035/articles

5. Тут можно создать конфиг задать промпты и указать адрес LLM http://localhost:3035/admin/config

Прозрачность данных

- В файле app/jobs/llm_job.rb находится job который подготавливает ответ для пользователя.
- В файле models/llm происходит запрос на LLM
- В файле models/semantic_search.rb происходит поиск контекста в базе данных

4. Телеграм-бот

Rutube telegram bot

Интеллектуальный помощник оператора службы поддержки Rutube в виде бота для Telegram.

Бот разработан с целью упростить взаимодействие пользователей и операторов службы техподдержки и автоматизировать обработку запросов.

Основная функция бота:

Бот был создан чтобы пользователи могли быстро и удобно получать ответы на свои вопросы или продолжать общение с техподдержкой прямо в Telegram. По сути, он является связующим звеном между пользователем и системой поддержки Rutube, предоставляя простой и интуитивно понятный интерфейс для коммуникации.

Интеграция с системой техподдержки:

Бот полностью интегрирован с нашей внутренней системой техподдержки. Для этого мы использовали два ключевых подхода:

1. **REST API** — для передачи данных между ботом и сервером техподдержки. Это позволяет боту отправлять запросы и получать структурированные ответы от системы.
2. **WebSocket** — для обеспечения квази-реалтайм работы. Это значит, что бот почти мгновенно получает и отправляет ответы и обновления с сервера, что делает взаимодействие с пользователем более быстрым и удобным.

Основные возможности бота:

1. **Задать вопрос оператору службы поддержки и получить ответ**:
    - Пользователь может использовать бота, чтобы задать любой вопрос о RUTUBE. Запрос передаётся в систему техподдержки, где оператор или автоматизированная система обработки запросов (в зависимости от режима работы) отвечают на него. Ответы возвращаются пользователю прямо в чат с ботом.
2. **Начало или продолжение обращения со службой поддержки**:
    - Бот позволяет не только начать новый запрос, но и продолжить ранее начатый диалог. Мы реализовали поддержку **deep linking**, что даёт возможность пользователю вернуться к конкретному обращению через специальную ссылку — это облегчает процесс взаимодействия и сохраняет контекст общения.
3. **Смена режима работы (*для тестирования*)**:
    - Для тестирования мы добавили возможность переключаться между режимами работы. В тестовом режиме бот может получать ответы напрямую от LLM, что позволяет проверять его работу и корректность ответов в различных сценариях.
4. **Оценка качества оказанной помощи**:
    - После получения ответа пользователь может оценить оказанную помощь с помощью **inline кнопок** прямо в сообщении с ответом. Эти оценки интегрируются с нашей системой, что помогает нам отслеживать качество работы операторов и автоматизированных систем.

Установка:

uv

```
uv venv

source .venv\bin\activate # Linux
```

```
.venv\Scripts\activate # Windows

uv sync
```

pip

```
python -m venv .venv

source .venv\bin\activate # Linux
.venv\Scripts\activate # Windows

pip install -r requirements.txt
```

Файл `.env`

Для запуска требуется создать файл с основными параметрами конфигурации.

> Токен бота можно получить у Bot Father.

Пример:

```
BOT_TOKEN = "1002225444:AAAee2_6oHd1Y9Msda_HLKlgscajrVT_EE" # Токен бота
LEVEL = "DEBUG" # Уровень логирования
WEB_URL = "https://copilot.rutube.ru" # URL фронта системы
API_URL = "https://copilot-api.rutube.ru" # URL API системы
API_PORT = "443" # DEPRECATED
WS_URL = "wss://copilot-api.rutube.ru" # URL WebSocket системы
WS_PORT = "80" # DEPRECATED
```

Запуск:

Локально

```
uv run main.py
```

> Можно запускать и без команду `uv run` при активированном окружении, либо установке через `pip`.

```
python main.py
```

Docker

Запуск

1. Собрать Docker-образ:

```
docker build -t имя_образа .
```

2. Когда образ собран, запустить контейнер на основе образа:

```
docker run --name имя_контейнера имя_образа
```

Остановка и удаление

Если нужно остановить контейнер:

```
docker stop имя_контейнера
```

Если нужно удалить контейнер:

```
docker rm имя_контейнера
```

Если нужно удалить образ:

```
docker rmi имя_образа
```

5. Описание сценария работы

5.1 Режим оператора-админа Telegram

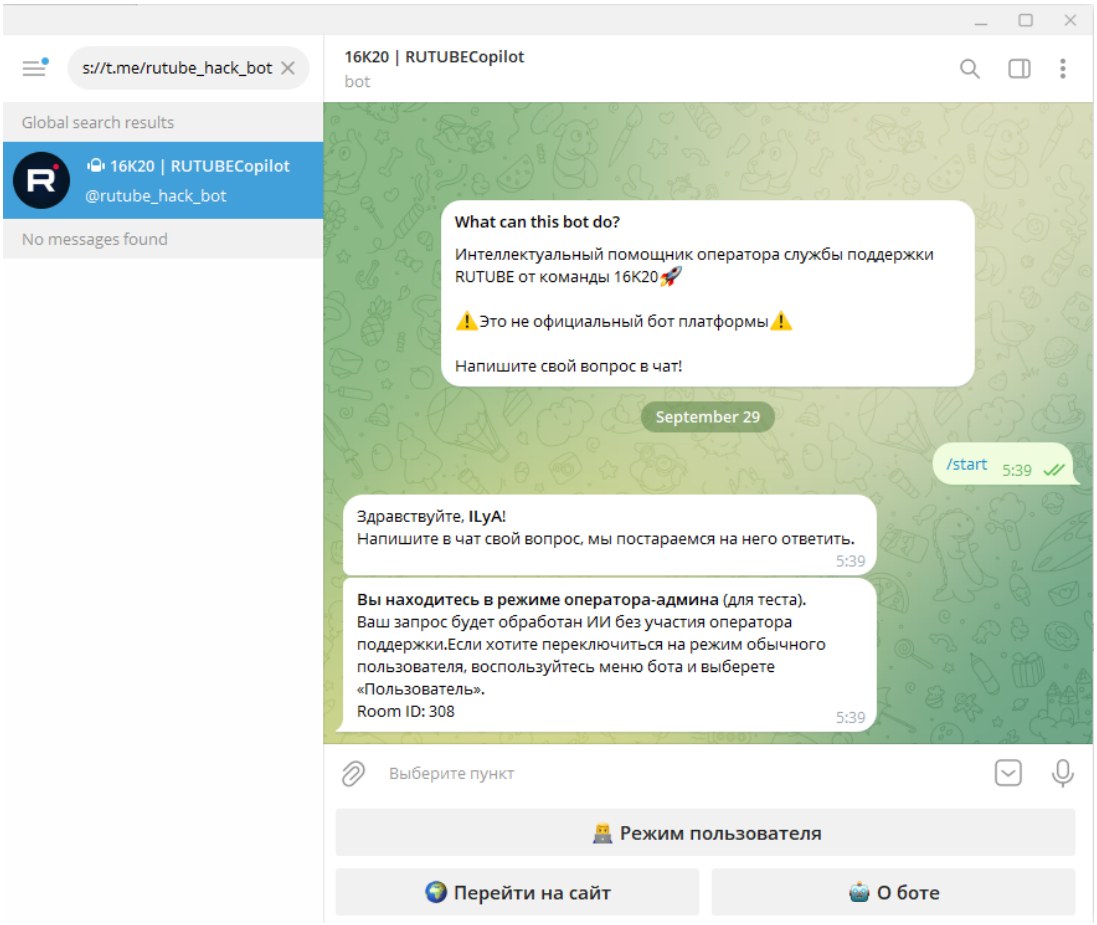

Рисунок 3

Пользователь авторизуется в Телеграм боте (Внимание! По умолчанию, активирован режим оператора-админа, все сообщения сразу будут возвращаться от llm без подтверждения оператора).

Ответ приходит в режиме стриминга

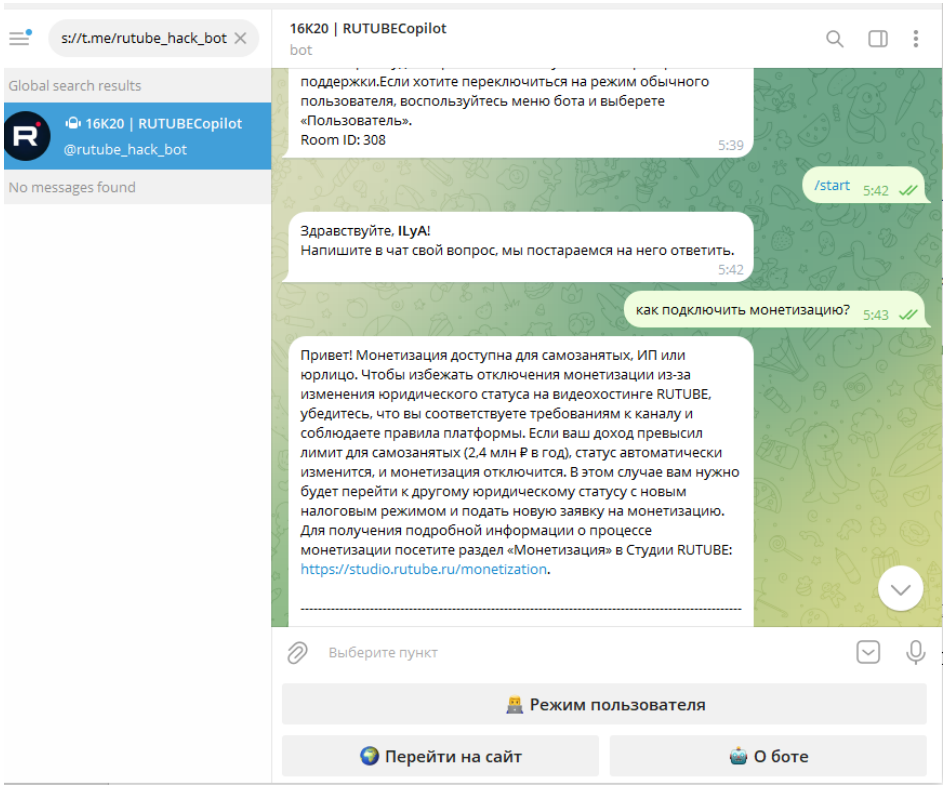

Рисунок 4

5.2 Режим пользователям Telegram

С помощью меню нужно изменить режим оператора-админа на режим пользователя.

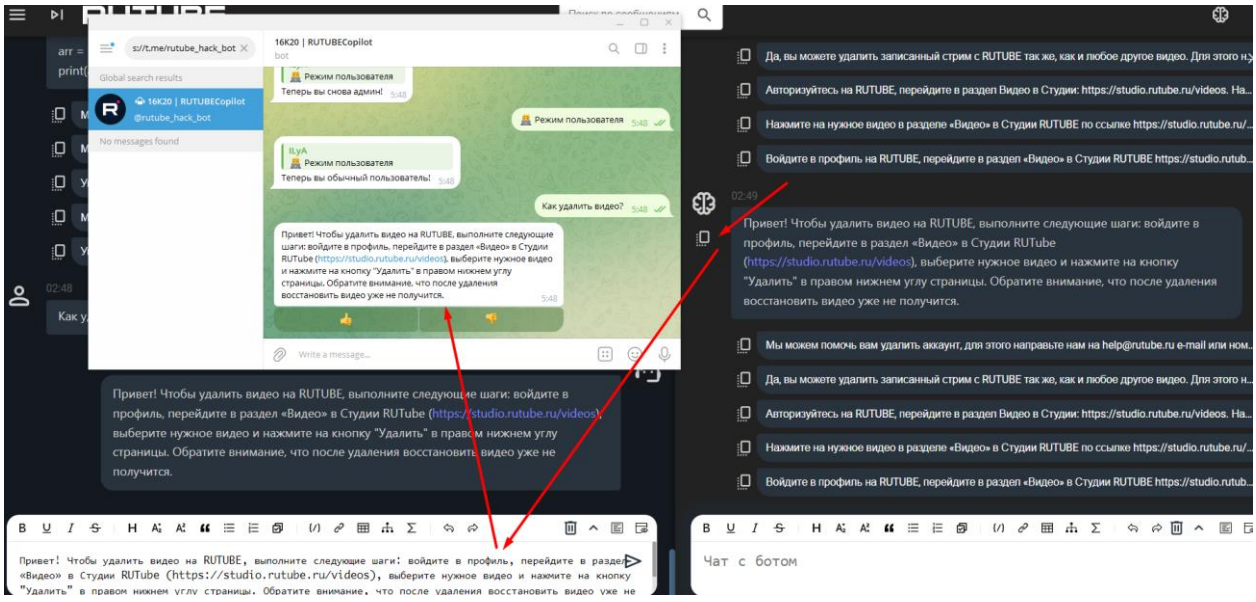

Рисунок 5

После того как пользователь задал вопрос, в админке у оператора появляется ответ от модели (и варианты контекста). Можно нажать скопировать, ответ

попадет в форму для отправки и после отправки сообщение отобразится Telegram пользователя.

5.3 Обработка некорректных запросов

Если пользователь введет вопрос, ответа на который не было в исходной базе знаний, то моделька ответит, что не знает ответа (при этом будет несколько вариантов контекста для теста)

Попробуем через режим пользователя в веб-версии:

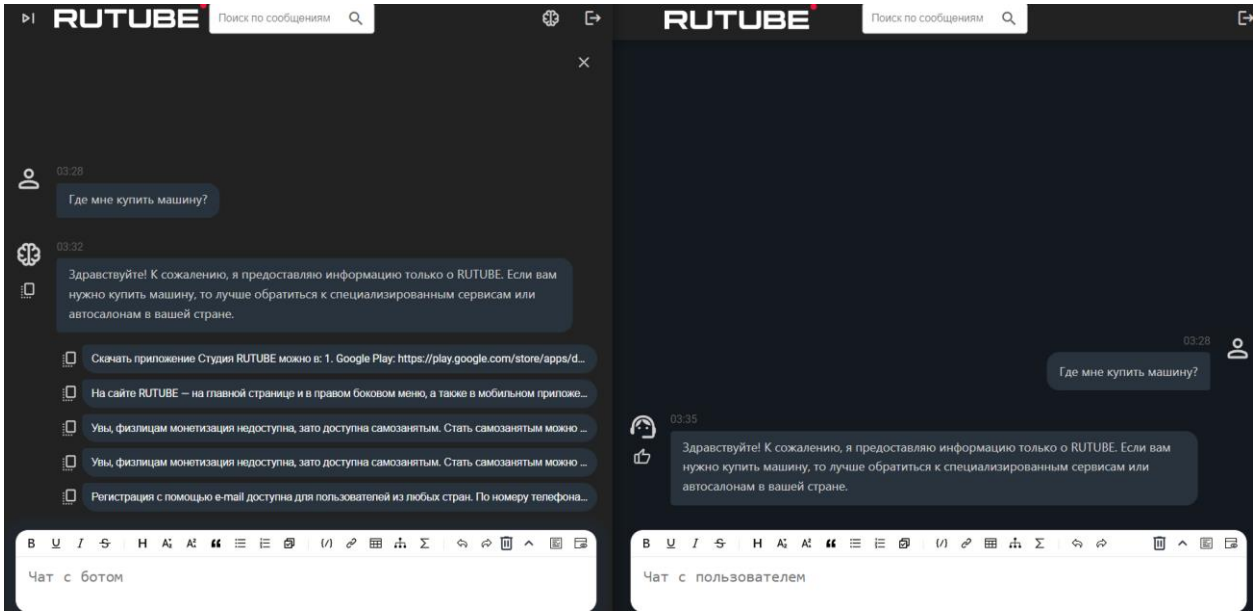

Рисунок 6

Моделька ответила, что может предоставить информацию только о RUTUBE.