

Отчёт по проекту: Чат-бот с генерацией текста на основе предобученной модели

Автор: Ачилов Никита

Группа: МИК22

Дата: 23 марта 2025

Описание проекта

В рамках данного проекта разработан чат-бот, который генерирует текстовые ответы с использованием предобученной языковой модели *microsoft/DialoGPT-small*. Чат-бот поддерживает настройку стиля ответа (нейтральный, дружелюбный, саркастичный) и предназначен для исследования возможностей генеративно-предобученных моделей в задачах диалогового взаимодействия.

Цель проекта — создать чат-бота, способного генерировать диалоговые ответы с заданным стилем.

Используемые библиотеки

Предобработка и интерфейс:

- *streamlit* (версия 1.43.2) — создание веб-интерфейса для взаимодействия с чат-ботом.
- *pandas* (версия 2.2.3) — работа с данными (использовалась в начальных экспериментах).
- *numpy* (версия 2.2.4) — работа с массивами.

Машинное обучение и нейросети:

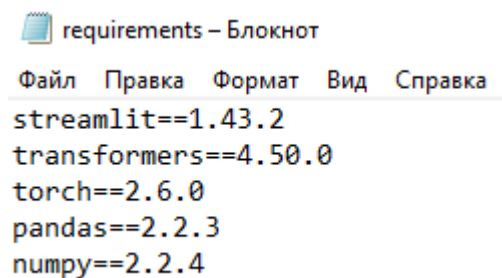
- *transformers* (версия 4.50.0) — загрузка и использование предобученной модели *DialoGPT-small*.
- *torch* (версия 2.6.0) — фреймворк для работы с нейросетями.

Ход выполнения

Шаг 1. Подключение библиотек

Для работы с моделью и создания веб-интерфейса были подключены необходимые библиотеки: *streamlit*, *transformers*, *torch*, *pandas*, *numpy*.

pip install -r requirements.txt в cmd

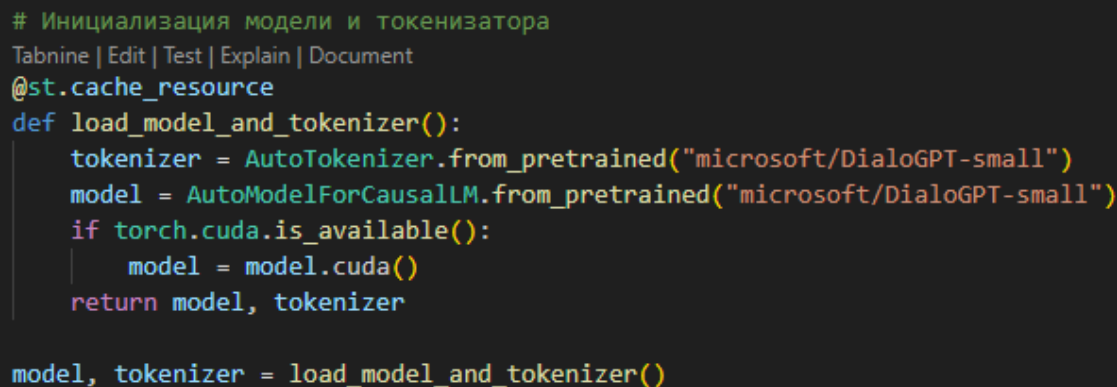


```
requirements - Блокнот
Файл  Правка  Формат  Вид  Справка
streamlit==1.43.2
transformers==4.50.0
torch==2.6.0
pandas==2.2.3
numpy==2.2.4
```

Рисунок 1 – Необходимые библиотеки.

Шаг 2. Выбор модели

Изначально использовалась модель *distilgpt2*, но из-за низкого качества диалоговых ответов была заменена на *microsoft/DialoGPT-small*, которая специально обучена для диалогов.



```
# Инициализация модели и токенизатора
Tabnine | Edit | Test | Explain | Document
@st.cache_resource
def load_model_and_tokenizer():
    tokenizer = AutoTokenizer.from_pretrained("microsoft/DialoGPT-small")
    model = AutoModelForCausalLM.from_pretrained("microsoft/DialoGPT-small")
    if torch.cuda.is_available():
        model = model.cuda()
    return model, tokenizer

model, tokenizer = load_model_and_tokenizer()
```

Рисунок 2 – Инициализация модели.

Шаг 3. Реализация генерации текста

Создана функция `generate_response`, которая принимает запрос пользователя, стиль ответа и максимальную длину ответа. Модель *DialoGPT-small* генерирует ответ, учитывая историю диалога.

```
# Функция для генерации ответа
Tabnine | Edit | Test | Explain | Document
def generate_response(prompt, style="neutral", max_length=100):
    # Токенизируем входной текст
    new_user_input_ids = tokenizer.encode(prompt + tokenizer.eos_token, return_tensors='pt')
    if torch.cuda.is_available():
        new_user_input_ids = new_user_input_ids.cuda()

    # Объединяем с историей диалога, если она есть
    if st.session_state.dialogue_history_ids is not None:
        bot_input_ids = torch.cat([st.session_state.dialogue_history_ids, new_user_input_ids], dim=-1)
    else:
        bot_input_ids = new_user_input_ids

    # Генерируем ответ
    chat_history_ids = model.generate(
        bot_input_ids,
        max_length=max_length + bot_input_ids.shape[-1],
        pad_token_id=tokenizer.eos_token_id,
        do_sample=True,
        top_k=100, # Увеличиваем top_k для разнообразия
        top_p=0.95,
        temperature=0.9 # Увеличиваем temperature для большей креативности
    )

    # Обновляем историю диалога
    st.session_state.dialogue_history_ids = chat_history_ids

    # Декодируем ответ
    response = tokenizer.decode(chat_history_ids[:, bot_input_ids.shape[-1]:][0], skip_special_tokens=True)

    # Применяем стиль к ответу
    styled_response = apply_style_to_response(response, style)
    return styled_response.strip()
```

Рисунок 3 – Генерация текста.

Шаг 4. Создание веб-интерфейса

С помощью *streamlit* реализован веб-интерфейс, который позволяет:

- Вводить запрос.
- Выбирать стиль ответа (нейтральный, дружелюбный, саркастичный).
- Указывать максимальную длину ответа через слайдер.
- Отображать историю чата (новые сообщения сверху).

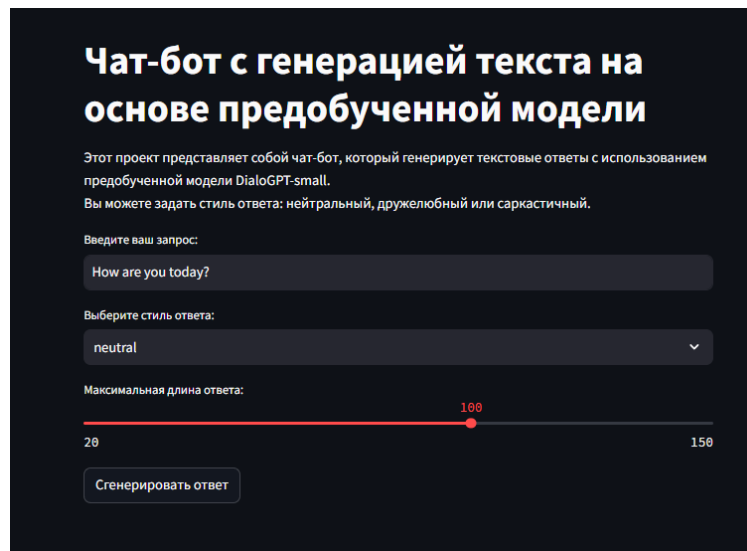


Рисунок 4 – Интерфейс.

Шаг 5. Тестирование и улучшение

- Добавлены параметры top_k , top_p , $temperature$ для улучшения разнообразия ответов.
- Реализована история чата с отображением новых сообщений сверху.
- Стиль ответа применён через постобработку, чтобы не сбивать модель.

Результаты

Чат-бот успешно генерирует диалоговые ответы, хотя качество ответов ограничено из-за использования небольшой модели *DialoGPT-small*. Примеры генерации текста:

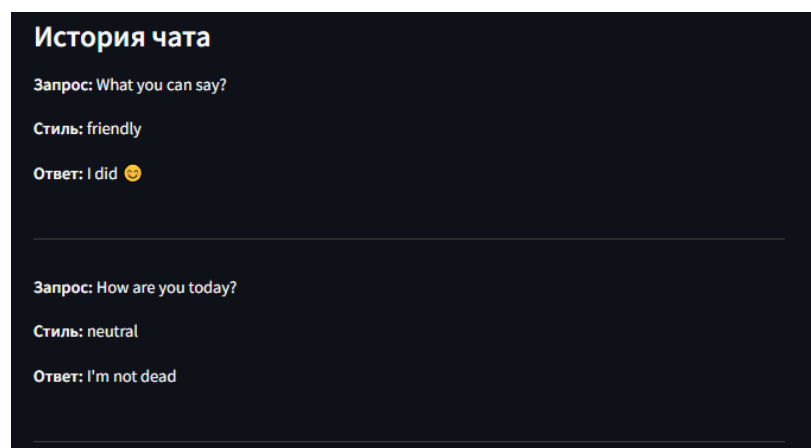


Рисунок 5 – Результат работы.

Однако в некоторых случаях модель выдаёт нерелевантные или повторяющиеся ответы.

Ограничения:

- Модель *DialoGPT-small* иногда теряет контекст и выдаёт нерелевантные ответы.
- Ограниченный объём обучающих данных (модель не дообучалась на пользовательских данных).
- Стил ь ответа реализован через постобработку, что не всегда точно отражает тон.

Выводы

В результате работы удалось:

- Разработать чат-бот на основе предобученной модели *DialoGPT-small*.
- Реализовать веб-интерфейс с поддержкой стилей ответа и историей чата.
- Продемонстрировать возможность использования генеративных моделей для диалогов.

Следующие шаги:

- Перейти на более крупную модель, например, *DialoGPT-medium*, для улучшения качества ответов.
- Дообучить модель на пользовательском датасете диалогов.
- Улучшить механизм управления стилем ответа (например, через дообучение с промптами).

Список литературы

1. Zhang, Y., et al. (2020). *DialogPT: Large-Scale Generative Pre-training for Conversational Response Generation*. [<https://arxiv.org/abs/1911.00536>].
2. Hugging Face Transformers Documentation: [<https://huggingface.co/docs/transformers>].
3. Streamlit Documentation: [<https://docs.streamlit.io>].
4. PyTorch Documentation: [<https://pytorch.org/docs/stable/index.html>].