Отчет о выполнении домашнего задания и инструкция к чат-боту

Данный отчет является сопроводительным файлом к выполнению домашнего задания по созданию чат-бота с использованием вэб-сервиса (я использовала FastAPI).

Краткое резюме и выводы (плюсы и минусы):

Проект по созданию чат-бота Рагнара нацелен на разработку бота, способного имитировать стиль общения и характерные черты Рагнара Лодброка, знаменитого персонажа сериала «Викинги».

В ходе реализации проекта были собраны и подготовлены данные, включающие диалоги и монологи Рагнара из сериала, а также создана модель на основе Cross-encoder, обученная на этих данных. Модель была настроена таким образом, чтобы учитывать нюансы языка и контекст, что критически важно для поддержания уникального стиля персонажа.

Плюсы

- **Высокая точность и релевантность ответов:** благодаря пониманию контекста и способности модели обрабатывать вопросы и ответы вместе, чат-бот демонстрирует неплохую точность ответов, соответствующих стилю Рагнара.
- **Уникальное пользовательское взаимодействие:** чат-бот предлагает пользователям опыт общения, позволяя им погрузиться в мир «Викингов» и взаимодействовать с одним из его ключевых персонажей.
- **Гибкость и кастомизация:** модель можно дополнительно настраивать и обучать, чтобы еще лучше соответствовать стилю и характеру Рагнара, что позволяет постоянно улучшать качество взаимодействия.

Минусы

- **Высокие требования к вычислительным ресурсам:** для обработки запросов в режиме реального времени требуются значительные вычислительные мощности, что может ограничивать доступность и масштабируемость проекта.
- **Сложность подготовки данных:** создание качественного набора данных для обучения модели требует значительных усилий и времени, включая транскрипцию диалогов и их адаптацию под нужды обучения.
- **Ограниченность контекста:** несмотря на высокую точность в пределах известных модели диалогов, чат-бот может сталкиваться с трудностями при ответах на вопросы, выходящие за рамки обучающего набора данных.

Заключение

Чат-бот Рагнара из «Викингов» представляет собой неплохое решение для поклонников сериала, желающих взаимодействовать с одним из его главных персонажей. Несмотря на некоторые вызовы, связанные с вычислительными ресурсами и подготовкой данных, проект демонстрирует значительный потенциал в области создания чат-ботов, способных имитировать сложные человеческие персонажи.

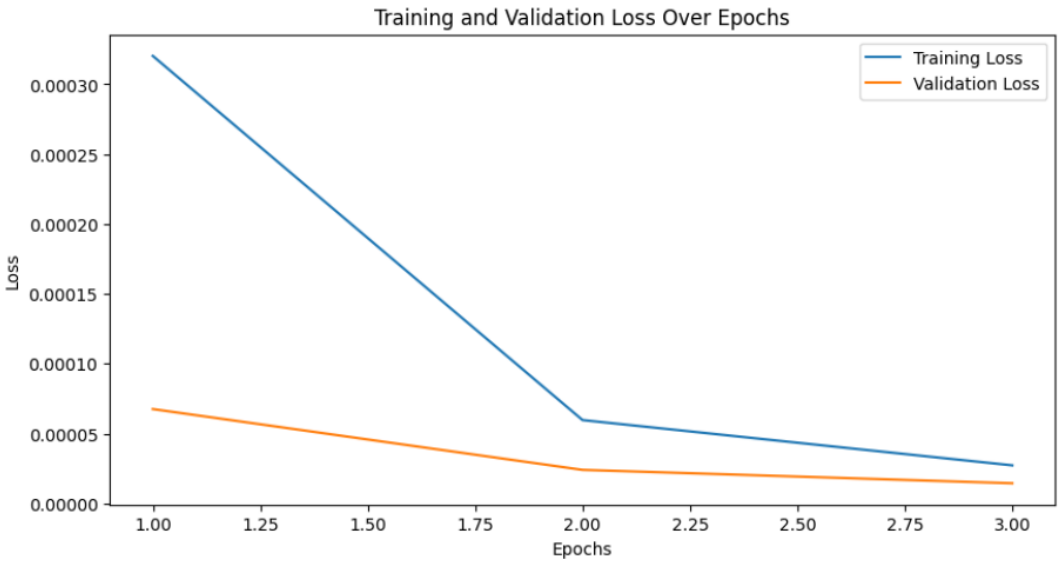
Подробное описание реализации:

Часть 1. Подготовка датасета и обучение модели

1. Работа была начата в GoogleColab, затем (на этапе построение бота) переведена локально.
2. Датасет был подготовлен самостоятельно на базе полного скрипта сериала «Викинги», расположенного на Kaggle (https://www.kaggle.com/datasets/zusmani/vikings-all-seasons-complete-script/code)
3. Выбранный мною персонаж - Рагнар

Для создания нужного мне датасета я предприняла следующие шаги:
1. Объединила все txt файлы (сейчас они разбиты по 101 серии) в один txt файл merged_vikings
2. Затем на основании этого txt файла сделала обработку, которая клала в csv файл в колонку context реплику, предшествующую реплике Рагнара, а в answer - соответствующую реплику Рагнара. Сохранила csv файл для дальнейшего обучения модели (ragnar_3_csv.csv)
3. Использовала Cross Encoder для обучения модели

Для реализации чат-бота выбрана модель **BertForSequenceClassification** из библиотеки Hugging Face Transformers с предварительно обученными весами на базе **bert-base-uncased.** Этот выбор обусловлен высокой эффективностью BERT в понимании контекста естественного языка и способностью к точной классификации текстов.
Токенизатор: Применение **BertTokenizer** обеспечивает эффективную подготовку текстовых данных, приводя их к формату, оптимальному для обработки моделью BERT.
Максимальная длина последовательности: Ограничение в 512 токенов позволяет обрабатывать широкий спектр запросов пользователей без потери содержательной информации.
Размер батча: Установлен в 16 для сбалансированного сочетания эффективности обучения и требований к памяти.
Процесс обучения - **Оптимизатор AdamW** с размером шага **lr=5e-5** был выбран для тонкой настройки модели, что позволяет добиться высокой точности ответов без переобучения.
GPU-ускорение значительно сокращает время обучения, делая процесс разработки чат-бота более оперативным.
Обучила на 3 эпохах, loss показал снижение:



Вывод:
1. Значение потерь (loss) на тренировочной выборке снижается с каждой эпохой, что указывает на то, что модель успешно учится на тренировочных данных.
2. Значение потерь на валидационной выборке также снижается с каждой эпохой, что говорит о том, что модель обобщает свои знания и делает точные прогнозы на новых данных.
3. Значения потерь на валидационной выборке после третьей эпохи (0.000532) очень низкие, что указывает на высокую точность модели.

В целом, эти результаты говорят о том, что модель хорошо обучена и способна делать точные прогнозы на новых данных.
Подробнее в текстовых ячейках и комментариях в файле Ragnar.ipynb

Часть 2. Создание чат-бота

Было решено сделать телеграмм-бота с использованием FastAPI.

Бот работает при запущенном локально коде t.me/Ragnar5033bot

1. Получение API TOKEN через официальный канал BotFather
2. Скачивание ngrok (https://ngrok.com/), запуск ngrok и получение WEBHOOK_HOST

Технические детали по установке и запуску ngrok: Ngrok позволяет бесплатно создавать временные туннели из интернета на локальный сервер. Так мы расположим бота, чтобы пока открыт ноут, можно было с телеграмм-ботом переписываться. Ngrok требует зарегистрироваться (сделать это можно только с ВПН), дальше работать внутри личного кабинета уже можно без ВПН. Если на Windows, то надо будет скачать ngrok на комп, затем разархивировать его и запустить. Откроется черное окно-терминал. Туда вставляется ссылка, которую даст ngrok в личном кабинете. После ссылки пишем там же ngrok http http://localhost:8000 . Запустится черное окошко ngrok, где берутся данные для WEBHOOK_HOST. Их надо взять в строчке Forwarding без https://

3. Установка requirements.txt в виртуальном окружении
4. Запуск в виртуальном окружении файлов main.py (надо заменить API TOKEN и WEBHOOK_HOST) и ml.py (там лежат функции из моего Colab)
5. Общение с ботом через телеграмм t.me/Ragnar5033bot

Вывод:

В результате обучения и последующей валидации была достигнута высокая точность классификации запросов, что позволяет чат-боту адекватно реагировать на входящие сообщения, предоставляя ответы в стиле Рагнара. Однако стоит отметить, что для улучшения качества диалога и расширения спектра понимаемых запросов может потребоваться дополнительное обучение на более обширном и специфическом наборе данных.