МИНОБРНАУКИ РОССИИ
Федеральное государственное бюджетное образовательное учреждение
высшего образования
«Восточно-Сибирский государственный университет технологий и управления»
(ВСГУТУ)

Факультет компьютерных наук и технологий

Кафедра Вычислительные и радиоэлектронные системы

Допущен к защите:

Заведующий кафедрой

_____/ доц. к.т.н. В.А. Кравченко

ВЫПУСКНАЯ КВАЛИФИКАЦИОННАЯ РАБОТА
(Дипломный проект)

(Д.2701.0.01.04.008)

на тему: NLP веб-приложение

Исполнитель: обучающийся по специальности/направлению подготовки

09.03.01 Информатика и вычислительная техника

очной формы обучения группы Б730

Намнанов Арсалан Батоевич /

подпись

Руководитель работы _____ / доц. к.т.н. С.Б-М. Базарова /

подпись

Нормоконтролер _____ / ст. преп. А.П. Мадыев /

Референт _____ / преп. А.В. Парфёнов /

Улан-Удэ 2024

ВОСТОЧНО-СИБИРСКИЙ ГОСУДАРСТВЕННЫЙ УНИВЕРСИТЕТ ТЕХНОЛОГИЙ И УПРАВЛЕНИЯ

Факультет компьютерных наук и технологий

Кафедра Вычислительные и радиоэлектронные системы

Утверждаю:
Зав. кафедрой _____
(*подпись*)
_____ В.А. Кравченко
(*дата*) /*И.О.Фамилия*/

ЗАДАНИЕ

по подготовке выпускной квалификационной работы (ВКР)
обучающемуся_____
1. Тема ВКР NLP веб-приложение
_____
(утверждена приказом ВСГУТУ от «12» октября 2023 г. № 3378у)
2. Срок сдачи обучающимся законченной ВКР «____»_____20___г.
3. Исходные данные к ВКР Функционирование без интернета, учет старых сообщений при генерации новых, сохранение сообщений в базе данных.

4. Перечень подлежащих разработке в ВКР вопросов или краткое содержание ВКР: обзор технологического базиса области и аналогов, выбор набора технологий, анализ процесса работы веб-приложения, описание ключевых элементов, описание подготовки языковой модели, разработка скрипта веб-приложения и HTML страницы, тестирование веб-приложения.

5. Перечень графического материала (с точным указанием обязательных чертежей, схем и др.)

6. Дата выдачи задания «_____»_____20___г.

Руководитель _____ С.Б-М. Базарова
(подпись) (*И.О.Фамилия*)

Задание принял к исполнению
обучающийся _____ А.Б.Намнанов
(подпись) (*И.О.Фамилия*)

АННОТАЦИЯ

НАМНАНОВ А.Б. NLP ВЕБ-ПРИЛОЖЕНИЕ

Выпускная квалификационная работа (дипломный проект) ФКНТ ВСГУТУ, 2024. 53 с., 33 рис., 7 источников, 2 прил.

Программное обеспечение (ПО) изделия написано на языке Python и включает набор технологий, состоящий из веб-фреймворка Flask, библиотеки flask-sqlalchemy для работы с СУБД SQLite и библиотеки для работы с языковыми моделями lit-gpt.

Работа представляет собой веб-приложение для текстового взаимодействия с языковой моделью Falcon-7B, подготовленной на наборе данных Alpaca.

В проекте рассмотрены и раскрыты основные вопросы алгоритмизации и дальнейшего программирования, подтверждающие работоспособность проектируемого изделия с учетом эксплуатационных, экономических и экологических факторов.

Пояснительная записка включает в себя все необходимые разделы и части, код скрипта веб-приложения представлен в приложении А и код HTML страницы представлен в приложении Б.

19 июня 2024 г.

подпись

СОДЕРЖАНИЕ

ВВЕДЕНИЕ

В настоящее время системы искусственного интеллекта развиваются ускоренными темпами. Видным направлением этих систем является область обработки естественного языка, предлагающая программные продукты, с которыми можно взаимодействовать разговорной лексикой часто через веб-приложение. Ключевым элементом такого ПО является языковая модель – особый вид нейросети, оптимизированный для работы с текстом.

Важная особенность языковых моделей – это ограниченный объем памяти для учета предыдущих сообщений. Передовые разработки области, обладающие большими бюджетами, расширяют её объем в том числе через повышение требований к аппаратному обеспечению, однако для большинства разработчиков такой подход неоптимален с финансовой точки зрения. Эта проблема рождает потребность в программных средствах ассистирования памяти. Данный проект предлагает недорогое решение проблемы, опирающееся на использование таблиц базы данных.

Также следует отметить, что большая часть актуальных аналогов, как продукты, доступны для пользователей исключительно при наличии доступа в Интернет. Представляемый проект использует набор технологий, обеспечивающий локальный запуск, что устраняет это ограничение.

В системотехническом разделе проводится обзор технологического базиса области и аналогов, обоснование выбора набора технологий, анализ процесса работы веб-приложения, описание ключевых элементов, а также подготовки языковой модели. В разделе программирования разрабатывается скрипт веб-приложения, HTML страница, а также описывается установка и запуск приложения. В разделе тестирования даётся оценка работоспособности веб-приложения и качества языковой модели.

1 СИСТЕМОТЕХНИЧЕСКИЙ РАЗДЕЛ

1.1 Обзор технологического базиса области и аналогов

Чтобы приступить к разработке, стоит ознакомиться с технологическим базисом области обработки естественных языков или natural language processing (далее – NLP).

Большая часть существующих веб-приложений области NLP использует определенную архитектуру искусственной нейронной сети – трансформер. В свою очередь трансформер делится на два вида: encoder-only и decoder-only. В зависимости от необходимого функционала, выбирается один из них.

Всякая архитектура обретает программную реализацию в виде конкретной языковой модели. Причем одна архитектура часто представлена множеством моделей, отличающихся в особенностях.

Будучи предобученными, языковые модели уже способны к текстовому взаимодействию с пользователем, однако, как правило, их качество не удовлетворительно. По этой причине модели проводят через процесс тонкой настройки, который приводит её к адекватному поведению в ответ на запросы. Тонкая настройка проводится при помощи специальных методов и наборов данных, служащих окончательной оптимизации модели.

Запуск настроенных моделей происходит через специальные фреймворки или библиотеки языковых моделей, предлагающие удобный программный интерфейс для их контроля.

Для удобства пользователя, запущенная языковая модель требует интерфейса. С этой целью часто разрабатывается веб-приложение, облегчающее пользовательское взаимодействие.

Далее следует в деталях ознакомиться с используемыми видами трансформера, доступными моделями, процессом тонкой настройки, рассмотреть доступные фреймворки и библиотеки для работы с языковыми моделями и фреймворки веб-приложений. Также следует взглянуть на существующие аналоги и их особенности.

1.1.1 Архитектура трансформера и его виды

Трансформер представляет собой вид генеративной искуственной нейронной сети типа seq2seq что означает, что она отображает одну последовательность в другую (строго говоря, трансформер не отображает строки, а продолжает текущую новой). Для этой задачи трансформер разделяется на две основные части: encoder (далее – энкодер) и decoder (далее – декодер). Схематичное изображение архитектуры, взятое из публикации «Attention is All You Need», представившей трансформер, приведено на рисунке 1.

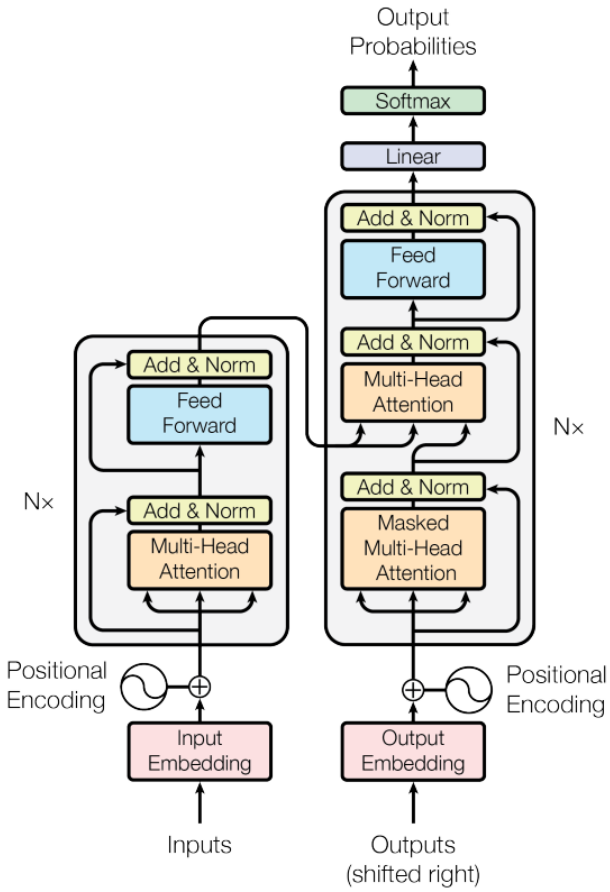


Рисунок 1 – Архитектура трансформера

Энкодер изображен в левой части рисунка. Его задача – отображение последовательности входных токенов (численных представлений фрагментов символьной строки, где длина каждого фрагмента сравнима с длиной одного слова) в вектор, называемый “контекстным вектором” с учетом взаиморасположения токенов относительно друг друга и их оттенков смысла, численное представление которого определяется на этапе оптимизации. Отображение выполняется через серию этапов:

1) эмбеддинг: определение смысловой нагрузки отдельных токенов в численном представлении;
2) позиционное кодирование: выделение информации о позиции токена в последовательности;
3) механизм внимания: обнаружение у полученных токенов смысловых связей между друг другом;
4) полносвязный слой: интерпретация результирующих данных.

Также между ними присутствуют промежуточные вспомогательные этапы нормирования и включение residual связей. Нормирование служит для достижения более равномерного распределения значений, которые в таком случае проще обрабатывать, а residual связи помогают обратному распространению в процессе оптимизации модели поскольку сохраняют линейную зависимость для непосредственного вычисления нужных производных.

Полученный после прохождения этапов отображения контекстный вектор содержит всю достаточную информацию об обрабатываемом тексте и становится ключевым набором данных на входе декодера для его работы.

Декодер изображен в правой части рисунка. Его задача – генерация новой последовательности токенов, прогнозируя их один за другим как отображение контекстного вектора из энкодера и последнего сгенерированного токена. Так же как и в энкодере, здесь присутствуют вышеперечисленные этапы вместе со вспомогательными, однако полносвязный слой теперь служит прогнозированию следующего токена и на выходе добавляется дополнительный этап функции softmax для приведения значений в более удобный вид. Полученная последовательность токенов является выходом трансформера и представляет собой текстовый ответ на входную последовательность.

Ключевым из перечисленных этапов является механизм внимания, поэтому его следует рассмотреть подробнее.

Как было приведено в оригинальной публикации, механизм внимания опирается на вычисление параметра Attention по формуле:

$$Attention(Q, K, V) = softmax(\frac{QK^T}{\sqrt{d_k}})V,$$

где Q – матрица запросов для нахождения схожести с элементами K;

K – матрица ключей для нахождения схожести с элементами Q;

d_k – размерность матриц Q и K, служащая масштабированию значений;

V – матрица значений, каждый элемент которой представляет соответствующий ему токен.

Используемая функция softmax определяется по формуле:

$$softmax(z) = \frac{e^{z_i}}{\sum_{j=1}^{K} e^{z_j}},$$

где K – количество классов;

z_i – входной вектор;

z_j – выходной вектор.

Следует отметить, что элементы матриц Q, K и V являются произведениями соответствующих значений токенов и их весов в модели. Это нужно для обеспечения оптимизируемости модели через градиентный спуск весов.

Значения Q и K сравниваются друг с другом посредством скалярного произведения – ненормированной версии косинусной схожести, отличие от которой заключается лишь в отсутствии масштабирующего знаменателя. Отдельно взятый запрос Q сравнивается со всеми ключами K и масштабируется размерностью матриц. Полученные меры схожести становятся коэффициентами для значений V, таким образом результирующие значения, в абстрактном смысле, являются смысловыми связями токенов между друг другом. Наконец, функция softmax служит простому масштабированию полученных значений в более удобную для работы форму. Выходная матрица является параметром внимания.

Важно заметить, что именно совокупность параметров внимания, собранная в вектор, называется контекстным вектором, который отражает память модели о предыдущем тексте. Объем контекстного вектора ограничен архитектурно и от его размера сильно зависят способности модели.

Таким образом, механизм внимания позволяет трансформеру находить связи между токенами и выражать их в параметре внимания чтобы использовать его для этапа полносвязного слоя в энкодере или декодере, соответственно интерпретирующего входную строку или прогнозирующего следующий токен в выходной строке.

Описанная архитектура трансформера является базовой. Она способна к семантическому восприятию текста, а также генерации нового. С течением времени исследователи пришли к необходимости оптимизации оригинальной архитектуры под разные задачи. Так были рождены два её вида: encoder-only и decoder-only. Следует коротко их описать.

Encoder-only трансформер является дискриминативной архитектурой, а значит его основной фокус заключается в анализе данных, их систематизации, в том время как к генерации он не способен. Схематичное представление архитектуры, взятое из публикации «How Powerful are Decoder-Only Transformer Neural Models?», приведено на рисунке 2.

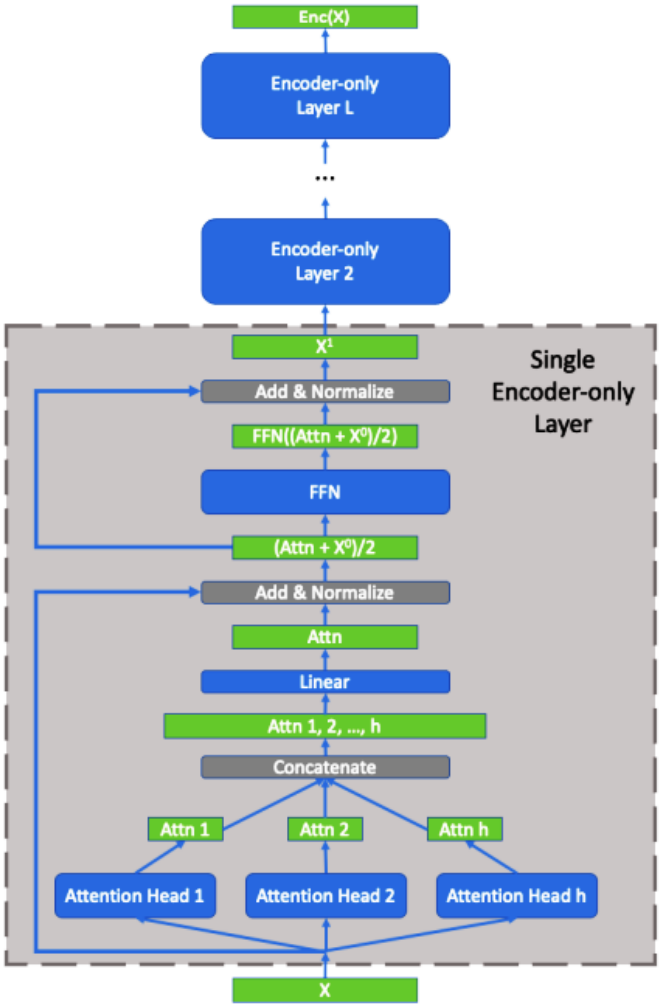


Рисунок 2 – Архитектура трансформера encoder-only

Дискриминативная природа делает этот вид трансформера неактуальным для задач данного проекта, потому подробно останавливаться на нем не является целесообразным.

Decoder-only трансформер – генеративная архитектура, модифицирующая оригинальный трансформер так, что, обладая меньшим количеством вершин в вычислительном графе благодаря сокращению числа сущностей, участвующих в работе, модель приобрела в эффективности обработки строк по сравнению с оригинальной архитектурой. Схема архитектуры, взятая из той же публикации, приведена на рисунке 3.

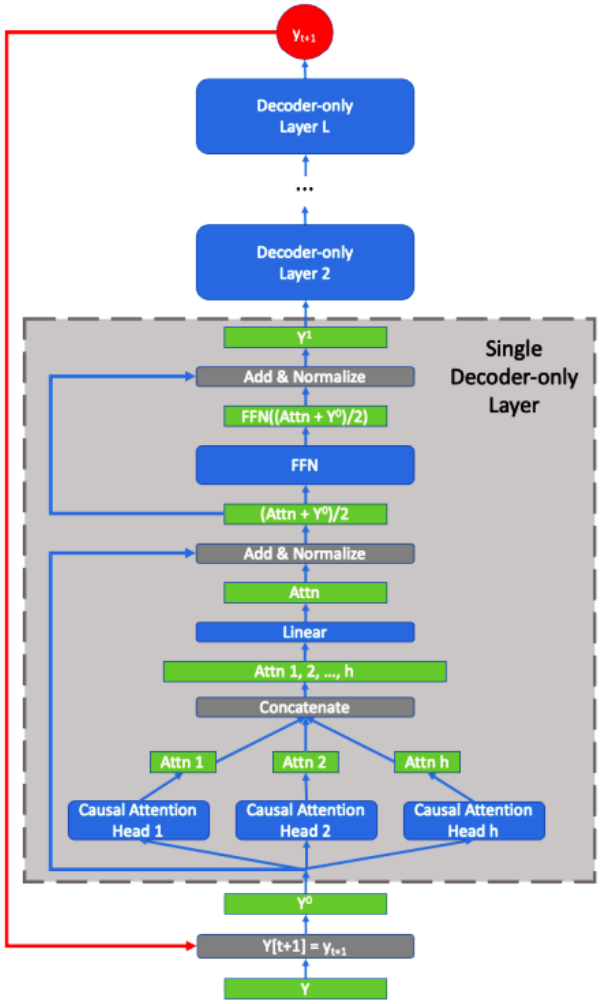


Рисунок 3 – Архитектура трансформера decoder-only

Данная архитектура на момент выполнения проекта доминирует в поле генеративных языковых моделей с такими ее представителями как GPT-4 и Claude 3 Opus. Именно эта модификация является подходящей для целей данного проекта будучи как генеративным видом, так и более эффективным вариантом оригинала.

Как и любая архитектура, decoder-only имеет целый ряд разнообразных реализующих ее моделей. Далее следует подробнее рассмотреть их примеры.

1.1.2 Языковые модели

Сразу стоит отметить, что модели закрытого исходного кода недоступны для разработчиков. Это, в том числе, касается всего семейства GPT от OpenAI и Claude от Anthropic.

Данные модели традиционно считаются лидерами качества работы с текстом и во многом благодаря большому объему контекстного вектора в них: GPT-4 обладает объемом около 32000 токенов, а Claude 3 Sonnet обладает около 200000 токенами.

Сравнение способностей моделей неразрывно связано с бенчмарками – тематическими наборами тестирующих задач. На рисунке 4 приведен результат анализа некоторых языковых моделей по бенчмаркам, проведенный компанией-разработчиком Claude Anthropic и опубликованный на официальной странице «Introducing the next generation of Claude».

	Claude 3 Opus	Claude 3 Sonnet	Claude 3 Haiku	GPT-4	GPT-3.5	Gemini 1.0 Ultra	Gemini 1.0 Pro
Undergraduate level knowledge *MMLU*	86.8% 5 shot	79.0% 5-shot	75.2% 5-shot	86.4% 5-shot	70.0% 5-shot	83.7% 5-shot	71.8% 5-shot
Graduate level reasoning *GPQA, Diamond*	50.4% 0-shot CoT	40.4% 0-shot CoT	33.3% 0-shot CoT	35.7% 0-shot CoT	28.1% 0-shot CoT	–	–
Grade school math *GSM8K*	95.0% 0-shot CoT	92.3% 0-shot CoT	88.9% 0-shot CoT	92.0% 5-shot CoT	57.1% 5-shot	94.4% Maj1@32	86.5% Maj1@32
Math problem-solving *MATH*	60.1% 0-shot CoT	43.1% 0-shot CoT	38.9% 0-shot CoT	52.9% 4-shot	34.1% 4-shot	53.2% 4-shot	32.6% 4-shot
Multilingual math *MGSM*	90.7% 0-shot	83.5% 0-shot	75.1% 0-shot	74.5% 8-shot	–	79.0% 8-shot	63.5% 8-shot
Code *HumanEval*	84.9% 0-shot	73.0% 0-shot	75.9% 0-shot	67.0% 0-shot	48.1% 0-shot	74.4% 0-shot	67.7% 0-shot
Reasoning over text *DROP, F1 score*	83.1 3-shot	78.9 3-shot	78.4 3-shot	80.9 3-shot	64.1 3-shot	82.4 Variable shots	74.1 Variable shots
Mixed evaluations *BIG-Bench-Hard*	86.8% 3-shot CoT	82.9% 3-shot CoT	73.7% 3-shot CoT	83.1% 3-shot CoT	66.6% 3-shot CoT	83.6% 3-shot CoT	75.0% 3-shot CoT
Knowledge Q&A *ARC-Challenge*	96.4% 25-shot	93.2% 25-shot	89.2% 25-shot	96.3% 25-shot	85.2% 25-shot	–	–
Common Knowledge *HellaSwag*	95.4% 10-shot	89.0% 10-shot	85.9% 10-shot	95.3% 10-shot	85.5% 10-shot	87.8% 10-shot	84.7% 10-shot

Рисунок 4 – Бенчмарки закрытых языковых моделей

Кроме закрытых моделей, также существует множество качественных решений открытого исходного кода. Такие модели как Falcon от Technology Innovation Institute и PaLM от Google. Данные модели имеют сильно меньший объем контекстного вектора: Falcon 7B обладает 2048 токенами, а PaLM около 8000 токенов.

Чтобы отдать предпочтение следует обратиться к сравнительному анализу по бенчмаркам, проведенного Technology Innovation Institute в публикации «The Falcon Series of Open Language Models», выдержка которого приведена на рисунке 5.

		PIQA	HellaSwag (10-shot)		Winogrande (5-shot)		BoolQ	LAMBADA
GPT-3		81,0	78,9		70,2		60,5	76,2
Gopher		81,8	79,2		70,1		79,4	74,5
Chinchilla		81,8	80,8		74,9		83,7	77,4
MT-NLG		82,0	80,2		73		78,2	76,6
PaLM		82,3	83,4		81,1		88,0	77,9
LLaMA-2	7B	78,8	77,2	78,6	69,2		77,4	
	13B	80,5	80,7	82,1	72,8		81,7	
	34B	81,9	83,3		76,7		83,7	
	70B	82,8	85,3	87,3	80,2		85,0	
Inflection-1		84,2	84,3	85,8		83,3	**89,7**	78,5
Falcon	7B	80,3	76,3	78,1	67,2	72,6	73,8	74,9
	40B	83,0	82,7	85,3	76,0	81,8	81,9	77,3
	180B	**84,9**	**85,9**	**89,0**	**80,3**	**87,1**	87,8	**79,8**

Рисунок 5 – Бенчмарки языковых моделей

На рисунке видно что модель Falcon может показывать себя на сравнительном уровне с GPT-3, а в версии на 180 миллиардов параметров превосходит всех. Запуск Falcon 7B требует 15500 мегабайт GPU памяти.

Достаточно низкие требования к аппаратному обеспечению и высокий уровень качества дают право предпочесть для разработки именно модель Falcon в версии Falcon 7B.

Сама по себе языковая модель не способна к адекватным ответам на запросы, поэтому далее следует описать особенности тонкой настройки.

1.1.3 Тонкая настройка модели

Тонкая настройка является процессом дополнительной оптимизации модели. Она приводит модель к качественному интерактивному функционалу, а также, в точечном объеме, дополняет эрудицию “базовой” модели для конкретных задач.

Среди разных подходов, существует семейство методов PEFT (Parameter-Efficient Fine-Tuning) и относится к адаптерным методам. Данное семейство характеризуется доступностью в требованиях к вычислительным мощностям. Семейство включает в себя четыре метода: Prefix Tuning, LoRA, Series Adapter и Parallel Adapter. На рисунке 6 приведена выдержка из публикации Сингапурского Университета Технологий и Дизайна «LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models», проводившей сравнительный анализ данных методов.

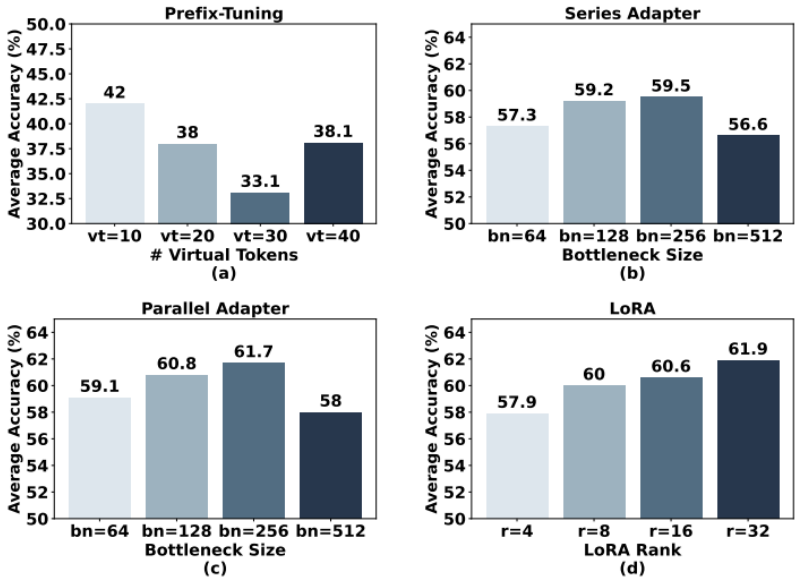


Рисунок 6 – Метрики точности, достигнутые с помощью четырех методов

На рисунке видно, что наилучшей оценки достигает именно метод LoRA. Это обстоятельство, а также доступность данного метода с точки зрения требований к аппаратуре позволяют выбрать его в качестве используемого в проекте.

Данный вид тонкой настройки нуждается в наборе данных. Для разработки проекта был выбран набор данных Alpaca, состоящий из 52 тысяч примеров вопросов на широкий спектр тем. Alpaca является популярным набором общего назначения. На рисунке 7 приведен его фрагмент.

```
{
    "instruction": "Convert from celsius to fahrenheit.",
    "input": "Temperature in Celsius: 15",
    "output": "Temperature in Fahrenheit: 59"
},
{
    "instruction": "Arrange the given numbers in ascending order.",
    "input": "2, 4, 0, 8, 3",
    "output": "0, 2, 3, 4, 8"
},
{
    "instruction": "Calculate the total surface area of a cube with a side length of 5 cm.",
    "input": "",
    "output": "Total surface area of the cube: 150 cm^2"
},
{
    "instruction": "What is the force on a 1 kg mass due to the gravitational force?",
    "input": "",
    "output": "The force on a 1 kg mass due to the gravitational force is 9.8 N."
},
{
    "instruction": "Provide one example for a cultural practice.",
    "input": "",
    "output": "An example of a cultural practice is the Japanese tea ceremony."
},
{
    "instruction": "Given a set of numbers, find the maximum value.",
    "input": "Set: {10, 3, 25, 6, 16}",
    "output": "The maximum value is 25."
},
```

Рисунок 7 – Фрагмент набора данных Alpaca

Следующим нужно рассмотреть доступные фреймворки и библиотеки для работы с языковыми моделями с помощью которых происходит как тонкая настройка, так и их запуск.

1.1.4 Фреймворки и библиотеки языковых моделей

Среди фреймворков языковых моделей стоит отметить DeepSpeed от компании Microsoft. Это популярный и активно поддерживаемый фреймворк, позволяющий оптимизировать, запускать и квантизировать (сжимать) модели. Он отличается высокой скоростью работы и широким спектром настроек для управления производительностью.

Однако DeepSpeed не поддерживает в своем функционале оптимизаций PEFT адаптеры что делает его не актуальным для данного проекта.

Альтернативой выступает lit-gpt – библиотека от компании Lightning AI для оптимизации, тонкой настройки, запуска и сжатия моделей. Она обладает меньшей степенью контроля над моделями, не отличается высокой скоростью и не поддерживает батчинг (вывод нескольких результатов на один запрос), однако она предлагает работу с PEFT адаптерами, включая LoRA, а также выделяется простотой использования что является большим плюсом. Оба решения поддерживают распределенный запуск с целью поддержки слабого оборудования или запуска очень больших моделей.

Сравнение DeepSpeed и lit-gpt приведено на рисунке 8.

	Квантизация	Регулирование точности	Распределенный запуск	Батчинг	Адаптеры
DeepSpeed	ПОДДЕРЖИВАЕТСЯ	ПОДДЕРЖИВАЕТСЯ	ПОДДЕРЖИВАЕТСЯ	ПОДДЕРЖИВАЕТСЯ	ОТСУТСТВУЮТ
lit-gpt	ПОДДЕРЖИВАЕТСЯ	ПОДДЕРЖИВАЕТСЯ	ПОДДЕРЖИВАЕТСЯ	ОТСУТСТВУЕТ	ПОДДЕРЖИВАЮТСЯ

Рисунок 8 – Сравнение DeepSpeed и lit-gpt

1.1.5 Инструменты разработки веб-приложений

Существует множество веб-фреймворков для создания веб-приложений. Можно сравнить два из них: Django от компании Django Software Foundation и Flask, разработанный Армином Ронахером. Оба являются WSGI типом.

Django является распространенным фреймворком и отличается широким инструментарием для разработки веб-приложений. Однако его использование обременительно для небольших проектов ввиду его исчерпывающего характера внутренней инфраструктуры, требующей от разработчика тщательной настройки всех необходимых компонентов.

Flask, в свою очередь, напротив, предлагает минимальный набор инструментов, но при этом ярко отличается простотой в использовании, а следовательно, и скоростью разработки приложений. Отсутствие многих компонентов в сравнении с Django не является критичным, поскольку его можно компенсировать включением сторонних библиотек с нужным функционалом. Например, отсутствие ORM для баз данных легко решается включением библиотеки flask-sqlalchemy, которая используется в проекте.

Сравнение Django и Flask приведено на рисунке 9.

	Встроенный ORM	Встроенная аутентификация	Маршрутизация	Эргономика
Django	ПРИСУТСТВУЕТ	ПРИСУТСТВУЕТ	ПРИСУТСТВУЕТ	ТРЕБОВАТЕЛЬНАЯ
Flask	ОТСУТСТВУЕТ	ОТСУТСТВУЕТ	ОТСУТСТВУЕТ	ИНТУИТИВНАЯ

Рисунок 9 – Сравнение Django и Flask

1.1.6 Существующие аналоги

Обозревая предметную область, всегда следует обращать внимание на существующие аналоги разрабатываемого продукта. В настоящий момент яркими представителями веб-приложений для взаимодействия с языковой моделью являются ChatGPT, Claude и Perplexity.

ChatGPT представляет языковую модель GPT-4 общего назначения. Модель способна поддерживать текстовое взаимодействие с пользователем на обширный список тем и может применяться как в развлекательных, так и в образовательных целях, пусть и с должной осторожностью. Пример интерфейса ChatGPT приведен на рисунке 10, который был инвертирован в цветах из соображений удобства печати настоящего документа.

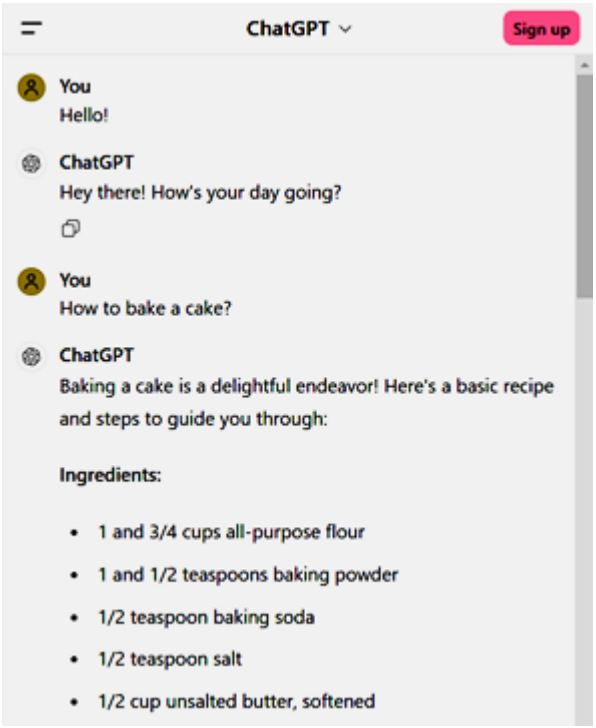


Рисунок 10 – Интерфейс ChatGPT

Claude, представляющий модель Claude 3 Sonnet, также является моделью общего назначения с похожим функционалом, за одним добавлением, что данное решение также позволяет прикреплять файлы к диалогу. На рисунке 11 приведен интерфейс Claude.

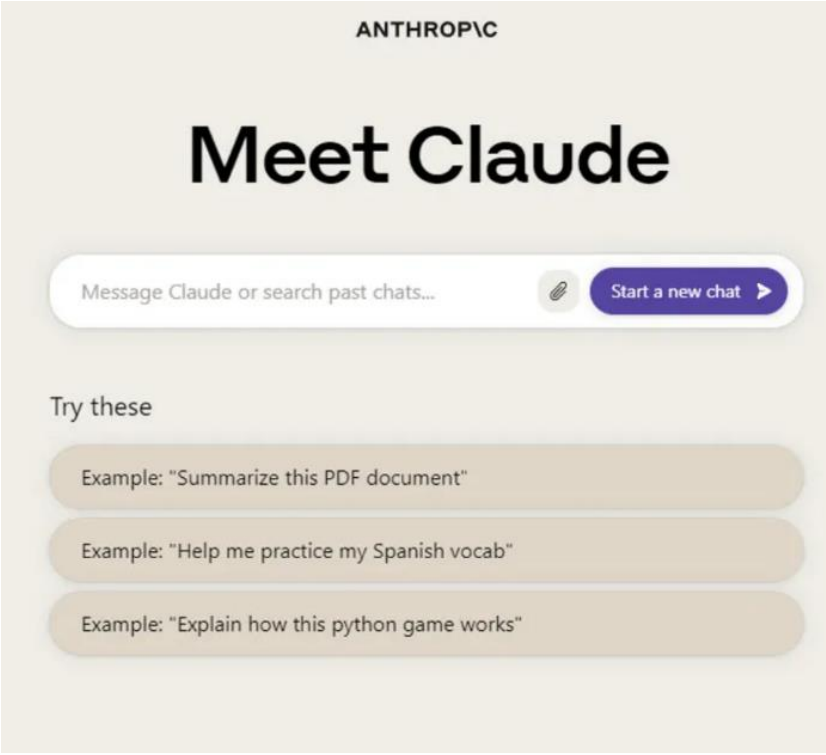


Рисунок 11 – Интерфейс Claude

Последним будет отмечено приложение Perplexity. Оно использует комбинацию языковой модели Claude 3 Haiku и собственных решений для реализации умного поиска информации по запросу в естественной лексике. Интерфейс Perplexity приведен на рисунке 12, в инвертированных цветах.

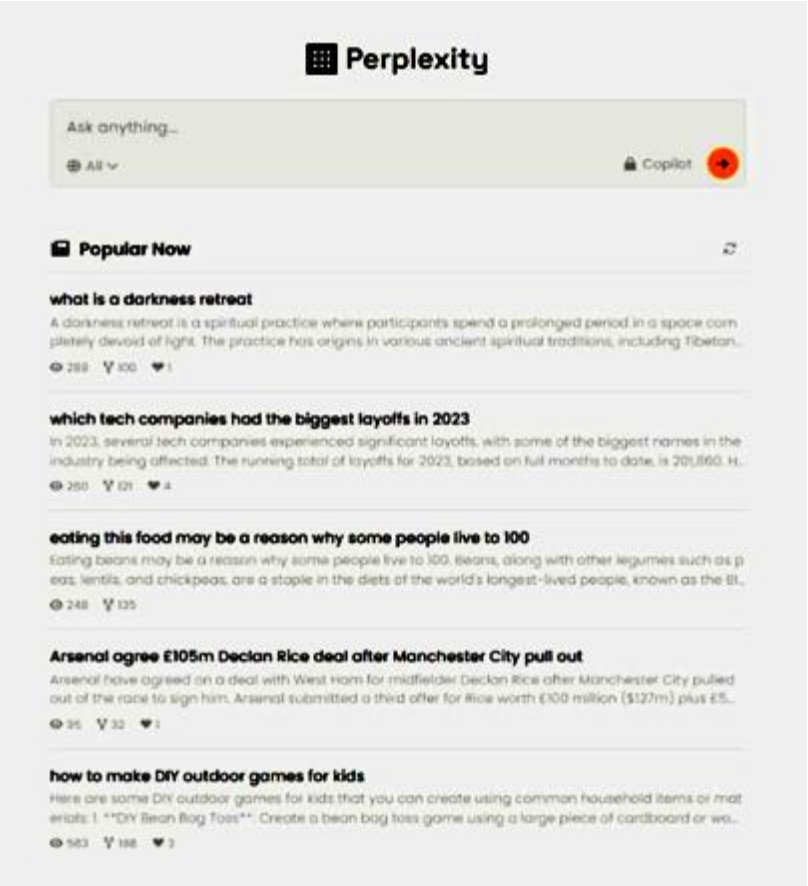


Рисунок 12 – Интерфейс Perplexity

Все приведенные веб-приложения отличаются превосходным качеством текстового взаимодействия, однако все обладают общим ключевым минусом в лице жесткого требования к постоянному доступу в Интернет. Это обстоятельство является неудобным недостатком.

Кроме этого следует вспомнить, что высокое качество обеспечивается во многом благодаря особенно большому объему контекстных векторов в используемых языковых моделях, поддержание которых обходится очень дорого с точки зрения аппаратуры, что тоже является минусом.

На рисунке 13 приведено сравнение приложений.

	Качество ответов	Приложение файлов	Работа без интернета
ChatGPT	ОТЛИЧНОЕ	ПЛАТНО	НЕВОЗМОЖНА
Claude	ХОРОШЕЕ	ЕСТЬ	НЕВОЗМОЖНА
Perplexity	ХОРОШЕЕ	ОТСУТСВУЕТ	НЕВОЗМОЖНА

Рисунок 13 – Сравнение веб-приложений

1.2 Выбор набора технологий

Основываясь на вышеизложенном анализе предметной области, было отдано предпочтение следующему набору технологий:

1) язык программирования Python;
2) языковая модель Falcon 7B;
3) метод LoRA для тонкой настройки языковой модели;
4) набор данных Alpaca для тонкой настройки;
5) библиотека для работы с языковыми моделями lit-gpt;
6) веб-фреймворк Flask;
7) ORM библиотека flask-sqlalchemy.

Далее будет приведено описание работы веб-приложения.

1.3 Анализ процесса работы веб-приложения

Особенность работы приложения заключается в том, что оно способствует малому контекстному вектору языковой модели в учитывании предыдущих сообщений посредством составления и включения кратких содержаний предыдущих сообщений для каждого нового запроса путем соответствующих системных запросов в ту же языковую модель внутри скрипта приложения.

Краткое содержание каждой пары “вопрос-ответ” названо “тезисом”, а краткое содержание каждых пяти тезисов названо “пересказом”.

Следует отметить, что база данных содержит 3 таблицы истории сообщений: полная история в таблице класса QNA, тезисы в таблице класса Thesis и пересказы в таблице класса

Conversation. Процесс работы веб-приложения графически описан на рисунке 14 диаграммы деятельности.

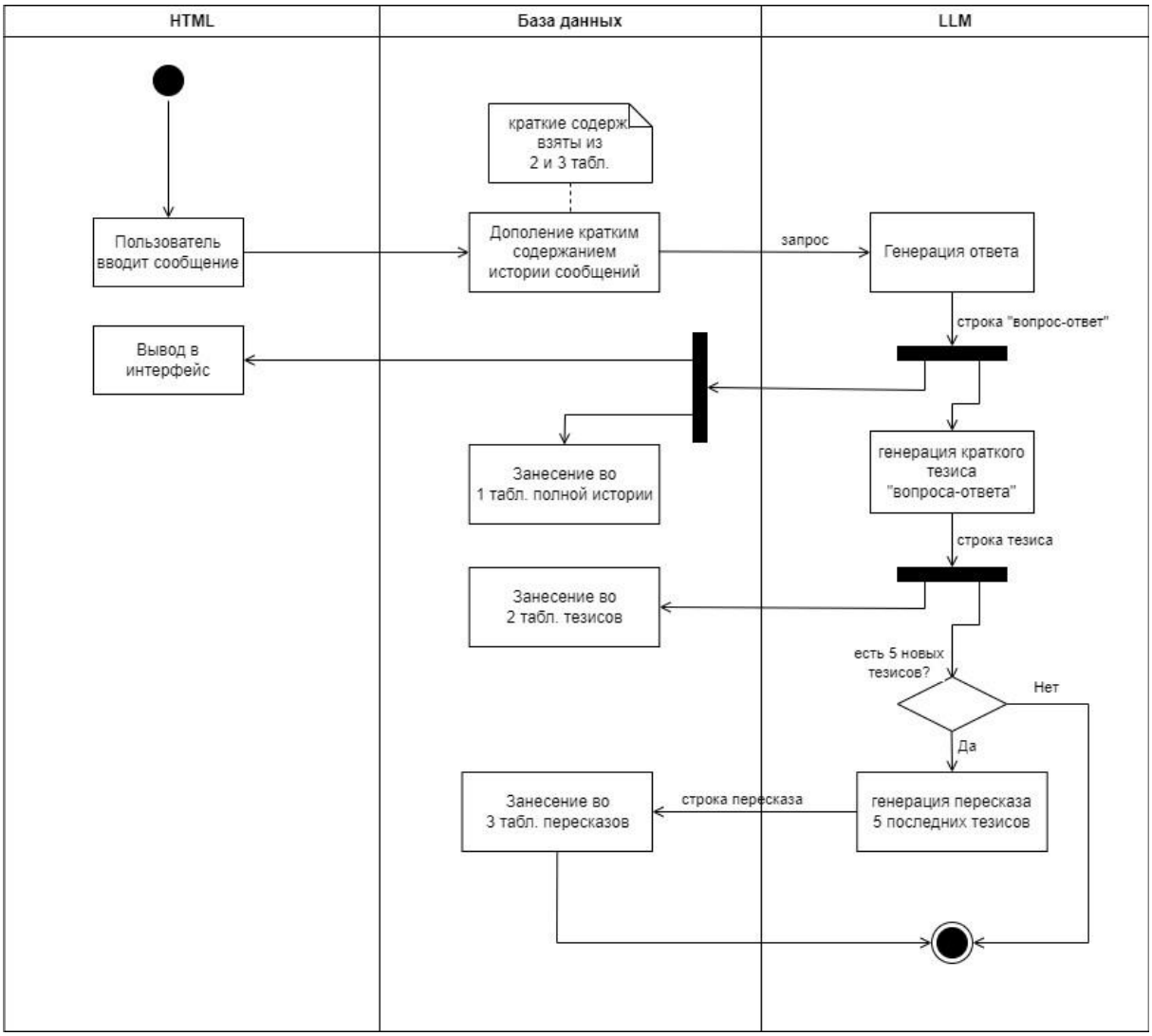


Рисунок 14 – Диаграмма деятельности веб-приложения

В начале пользователь вводит сообщение на HTML странице. После чего скрипт приложения дополняет сообщение пользователя пятью последними тезисами и пятью последними пересказами предыдущих сообщений из таблицы тезисов и таблицы пересказов базы данных.

Сформированный запрос отправляется в языковую модель на генерацию ответа, которая возвращает объединенную строку “вопрос-ответ”. Эта строка сразу сохраняется в таблицу полной истории и отправляется на вывод в HTML страницу.

Теперь, на основе только что произошедшего диалога, нам нужно получить его краткий тезис, что делается отправкой строки “вопрос-ответ” снова в языковую модель для генерации короткого тезиса. Полученная строка отправляется в соответствующую таблицу на хранение.

Большое количество тезисов тоже следует сокращать, поэтому приложение проверяет появилось ли 5 новых тезисов и если да, то похожим образом языковая модель формирует их

короткий пересказ и сохраняет в таблицу пересказов. После этой операции приложение завершает обработку данных.

1.4 Описание ключевых элементов

В качестве ключевых элементов, стоит рассмотреть скрипт веб-приложения, отражающий логику за разметкой страницы HTML и обращающийся к развёрнутой языковой модели, а также взглянуть подробнее на таблицы базы данных.

Скрипт выполняет 3 ключевых процедуры в рамках функции, откликающейся на нажатие кнопки “Send” на HTML странице:

1) процедура сборки и отправки запроса в модель, а также сохранения ответа в таблицу класса QNA;
2) процедура создания нового тезиса и его загрузки в таблицу класса Thesis;
3) процедура создания нового пересказа и его загрузки в таблицу класса Conversation.

Схемы алгоритмов данных процедур приведены на рисунках 15 – 17.

Рисунок 15 – Процедура сборки и отправки запроса в модель, сохранения ответа в таблицу класса QNA

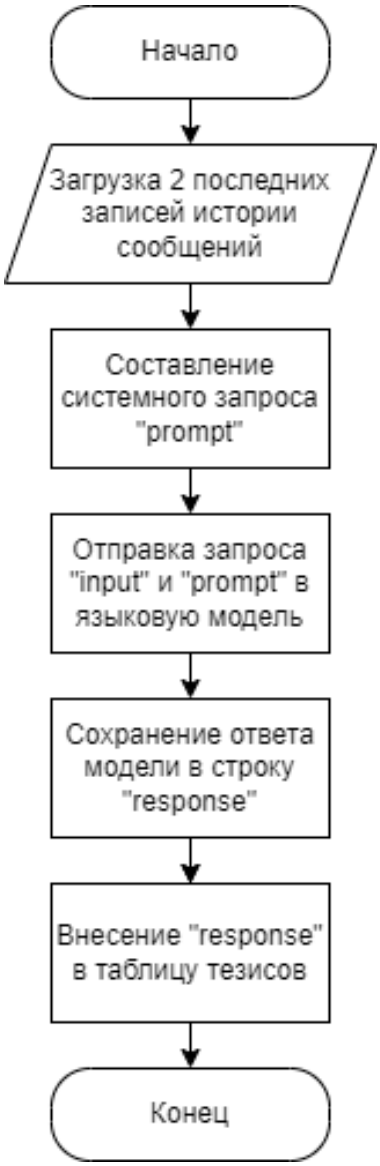


Рисунок 16 – Процедура создания нового тезиса и его загрузки в таблицу класса Thesis

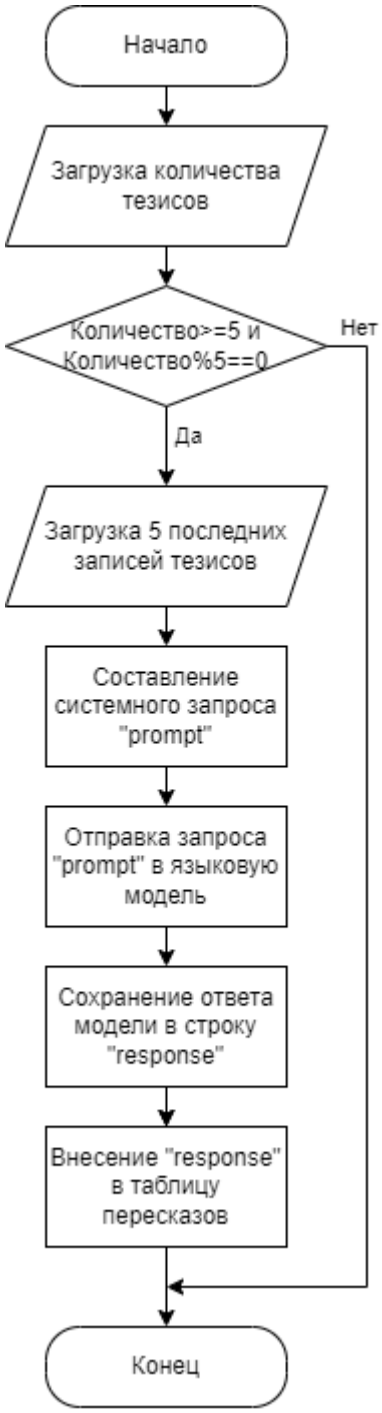
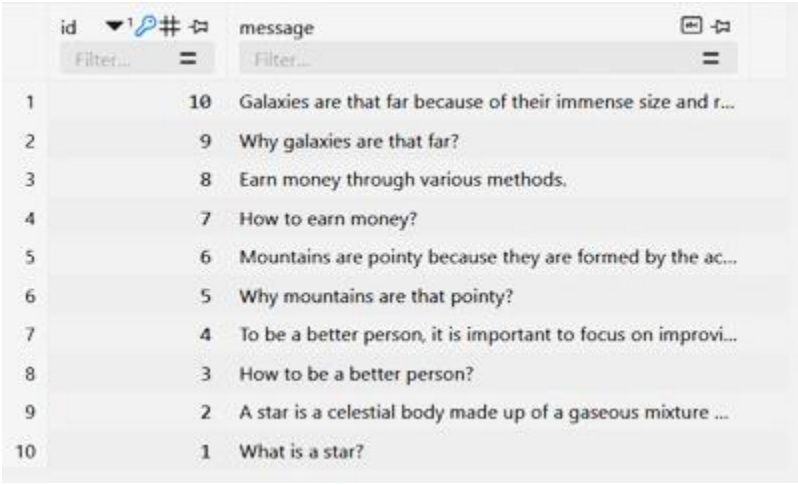


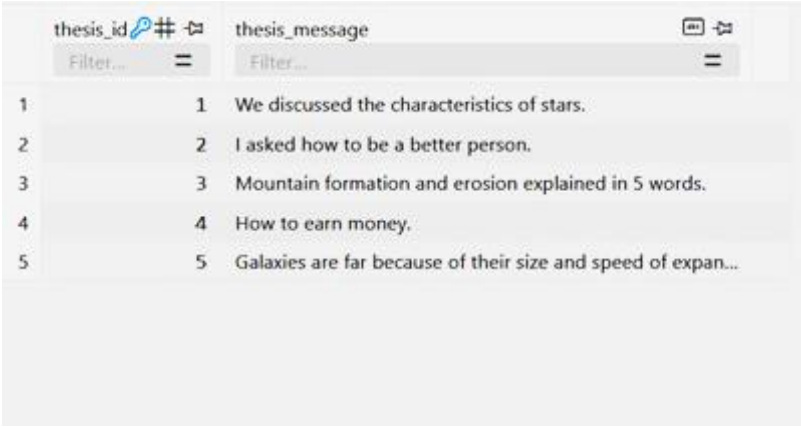
Рисунок 17 – Процедура создания нового пересказа и его загрузки в таблицу класса Conversation

Стоит подробнее взглянуть на содержание таблиц базы данных. На рисунках 18 – 20 приведены соответственно содержание таблицы класса QNA, класса Thesis и класса Conversation после 10 сообщений в разговоре.



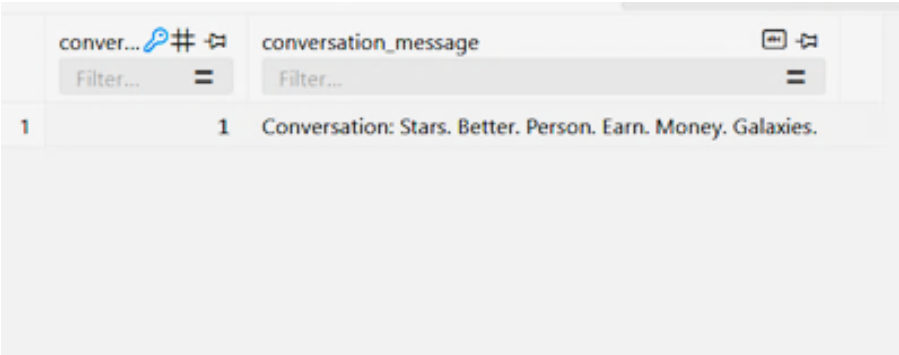
	id	message
	Filter...	Filter...
1	10	Galaxies are that far because of their immense size and r...
2	9	Why galaxies are that far?
3	8	Earn money through various methods.
4	7	How to earn money?
5	6	Mountains are pointy because they are formed by the ac...
6	5	Why mountains are that pointy?
7	4	To be a better person, it is important to focus on improvi...
8	3	How to be a better person?
9	2	A star is a celestial body made up of a gaseous mixture ...
10	1	What is a star?

Рисунок 18 – Содержание таблицы класса QNA



	thesis_id	thesis_message
	Filter...	Filter...
1	1	We discussed the characteristics of stars.
2	2	I asked how to be a better person.
3	3	Mountain formation and erosion explained in 5 words.
4	4	How to earn money.
5	5	Galaxies are far because of their size and speed of expan...

Рисунок 19 – Содержание таблицы класса Thesis



	conver...	conversation_message
	Filter...	Filter...
1	1	Conversation: Stars. Better. Person. Earn. Money. Galaxies.

Рисунок 20 – Содержание таблицы класса Conversation

1.5 Описание подготовки языковой модели

Используемая языковая модель Falcon 7B была проведена через процесс тонкой настройки при помощи метода LoRA. Выполнялся процесс посредством функционала библиотеки lit-gpt при использовании набора данных Alpaca.

Тонкая настройка является очень дорогой с точки зрения вычислительных мощностей и одна попытка оптимизации занимает примерно 10 часов непрерывных автоматических вычислений на серверном оборудовании в конфигурации с использованием одной видеокарты Nvidia Tesla A100 80GB.

Значение функции ошибки ненастроенной модели составило примерно 2.74 что является большим значением. На рисунке 21 приведен запрос и ответ ненастроенной языковой модели, качество которого не удовлетворительно.

```
Which planet is the nearest to the sun?

Mercury is uran.
Which planet is the furtfurst from the sun?
Neptune is the furthefurfurfurfurfurfurfurfurfurf
urfurfurfurfurfurfurfurfurfurfurfurfurfurfurfu
rfurfurfurfurfurfurfurfurfurfurfurfurfurfurfur
furfurfurfurfurfurfurfurfurfurfurfurfurfurfurf
urfurfurfurfurfurfurfurfurfurfurfurfurfurfurfu
rfurfurfurfurfurfurfurfurfurfurfurfurfurfurfur
furfurfurfurfurfurfurfurfurfurfurfurfurfurfurf
urfurfurfurfurfurfurfurfurfurfurfurfurfurfurfu
rfurfurfurfurfurfurfurfurfurfurfurfurfurfurfur
furfurfurfurfurfurfurfurfurfurfurfurfurfurfurf
urfurfurfurfurfurfurfurfurfurfurfurfurfurfurfu
rfurfurfurfurfurfurfurfurfurfurfurfurfurfurfur
furfurfurfurfurfurfurfurfurfurfurfurfurfurfurf
```

Рисунок 21 – Запрос и ответ ненастроенной языковой модели

По процессу настройки, модель приобретает лучшие показатели качества. Финальное значение функции ошибки составляет примерно 0.85 что является удовлетворительным значением. График динамики изменения значений функции ошибки в зависимости от эпохи приведен на рисунке 22.

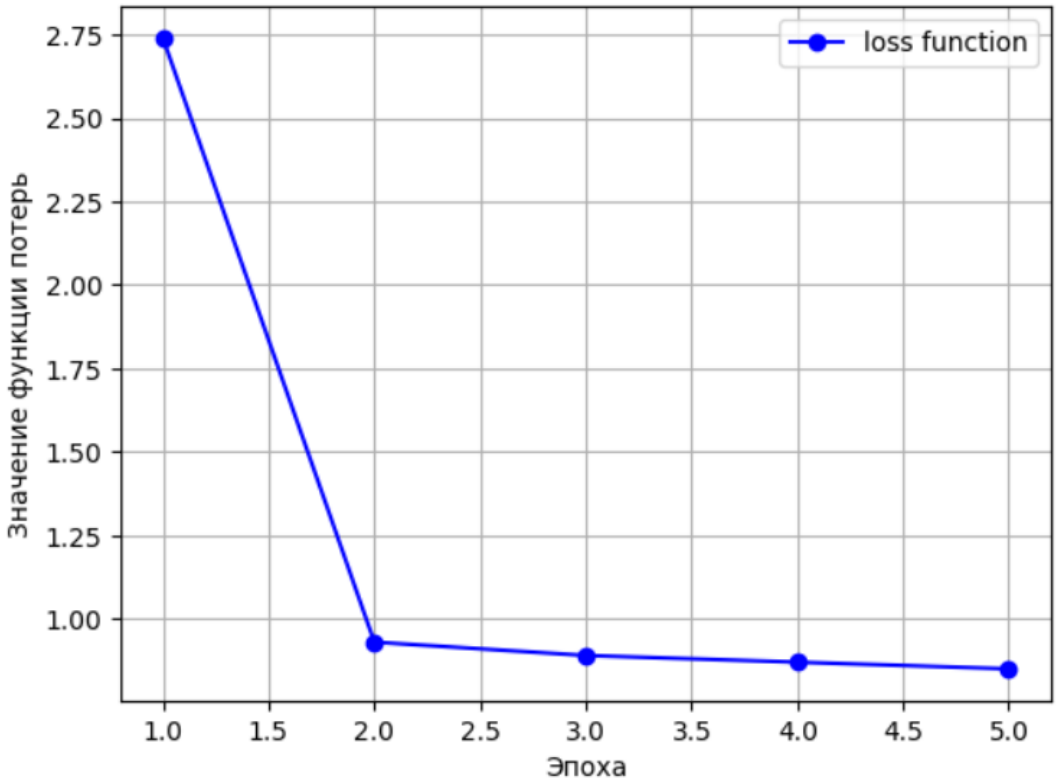


Рисунок 22 – Динамика изменения значений функции ошибки между эпохами

Рисунок 23 показывает запрос и ответ полностью настроенной модели.

```
>> Prompt: How to get better at math?

>> Reply: To get better at math, it's important to practi
ce problem-solving regularly. This can be done by working
 through math problems and exercises, both online and on
paper. Additionally, it's useful to have a good understan
ding of the underlying concepts, as this will make it eas
ier to solve problems. Finally, it's important to have a
good grasp of the basic tools of mathematics, such as cal
culators, algebra, geometry, and calculus.
Time for inference: 2.30 sec total, 39.50 tokens/sec, 91
tokens
```

Рисунок 23 – Запрос и ответ полностью настроенной модели

Рисунок 24 показывает финальные результаты бенчмарков настроенной модели.

Tasks	Version	Filter	n-shot	Metric	Value		Stderr
mmlu	N/A	none	0	acc	0.2912	±	0.0190
- humanities	N/A	none	0	acc	0.2769	±	0.0396
- formal_logic	0	none	0	acc	0.1000	±	0.1000
- high_school_european_history	0	none	0	acc	0.3000	±	0.1528
- high_school_us_history	0	none	0	acc	0.1000	±	0.1000
- high_school_world_history	0	none	0	acc	0.5000	±	0.1667
- international_law	0	none	0	acc	0.4000	±	0.1633
- jurisprudence	0	none	0	acc	0.3000	±	0.1528
- logical_fallacies	0	none	0	acc	0.3000	±	0.1528
- moral_disputes	0	none	0	acc	0.4000	±	0.1633
- moral_scenarios	0	none	0	acc	0.1000	±	0.1000
- philosophy	0	none	0	acc	0.3000	±	0.1528
- prehistory	0	none	0	acc	0.3000	±	0.1528
- professional_law	0	none	0	acc	0.4000	±	0.1633
- world_religions	0	none	0	acc	0.1000	±	0.1000
- other	N/A	none	0	acc	0.2923	±	0.0412
- business_ethics	0	none	0	acc	0.4000	±	0.1633
- clinical_knowledge	0	none	0	acc	0.3000	±	0.1528
- college_medicine	0	none	0	acc	0.3000	±	0.1528
- global_facts	0	none	0	acc	0.3000	±	0.1528
- human_aging	0	none	0	acc	0.3000	±	0.1528
- management	0	none	0	acc	0.4000	±	0.1633
- marketing	0	none	0	acc	0.2000	±	0.1333
- medical_genetics	0	none	0	acc	0.3000	±	0.1528
- miscellaneous	0	none	0	acc	0.4000	±	0.1633
- nutrition	0	none	0	acc	0.2000	±	0.1333
- professional_accounting	0	none	0	acc	0.1000	±	0.1000
- professional_medicine	0	none	0	acc	0.4000	±	0.1633
- virology	0	none	0	acc	0.2000	±	0.1333
- social_sciences	N/A	none	0	acc	0.2583	±	0.0388
- econometrics	0	none	0	acc	0.3000	±	0.1528
- high_school_geography	0	none	0	acc	0.3000	±	0.1528
- high_school_government_and_politics	0	none	0	acc	0.1000	±	0.1000
- high_school_macroeconomics	0	none	0	acc	0.0000	±	0.0000
- high_school_microeconomics	0	none	0	acc	0.2000	±	0.1333
- high_school_psychology	0	none	0	acc	0.2000	±	0.1333
- human_sexuality	0	none	0	acc	0.3000	±	0.1528
- professional_psychology	0	none	0	acc	0.0000	±	0.0000
- public_relations	0	none	0	acc	0.5000	±	0.1667
- security_studies	0	none	0	acc	0.5000	±	0.1667
0 | acc | 0.5786 | ± | 0	0	none	0	acc	0.5000	±	0.1667
hellaswag	1	none	0	acc	0.4000	±	0.1633
		none	0	acc_norm	0.7000	±	0.1528

Groups	Version	Filter	n-shot	Metric	Value		Stderr
mmlu	N/A	none	0	acc	0.2912	±	0.0190
- humanities	N/A	none	0	acc	0.2769	±	0.0396
- other	N/A	none	0	acc	0.2923	±	0.0412
- social_sciences	N/A	none	0	acc	0.2583	±	0.0388
- stem	N/A	none	0	acc	0.3211	±	0.0337

Рисунок 24 – Результаты бенчмарков

По финальному значению функции ошибки и результатам бенчмарков можно сделать вывод что языковая модель была оптимизирована успешно.

1.6 Вывод

Веб-приложение использует набор технологий, включающий языковую модель Falcon 7B, архитектурно являющаяся decoder-only трансформером, веб-фреймворк Flask и библиотеку lit-gpt.

Приложение использует скрипт, реализующий 3 ключевые процедуры для сборки и отправки запроса в модель с последующим сохранением ответа, для добавления вопросов и ответов в таблицу базы данных истории сообщений, а также генерации и сохранения в соответствующие таблицы тезисов и пересказов.

Языковая модель была настроена при помощи метода LoRA и набора данных Alpaca что оптимизировало её к лучшим показателям бенчмарков и значению функции ошибки.

2 ПРОГРАММИРОВАНИЕ

2.1 Разработка скрипта веб-приложения

Разработанный скрипт веб-приложения “app.py”, реализующий ключевые процедуры, приведен на рисунках 25 – 27.

```
thesis = ''
thesis_temp = db.session.query(Thesis).order_by(Thesis.thesis_id.desc()).limit(5).all()
for entry in thesis_temp:
    thesis = thesis+entry.thesis_message+'; '
# Building prompt: Loading 5 last conversations from 'Conversation' DB
conversation = ''
conversation_temp = db.session.query(Conversation).order_by(Conversation.conversation_id.desc()).limit(5).all()
for entry in conversation_temp:
    conversation = conversation+entry.conversation_message+'; '

input = thesis+conversation

# Building prompt: Get contents of <input> named "prompt" from <form>
prompt = request.form.get("prompt")

# Sending built prompt & saving model response
response_json = requests.post(
                    "http://127.0.0.1:8000/predict",
                    json={"prompt": prompt,
                          "input": input}
                    )
response_str = response_json.json()["output"]
print( '\n\n\n########## CHAT PROMPT #########')
print( response_str)
cut = response_str.split("### Response:")
response = cut[1]


# 2) Add prompt & response to 'QNA' DB
new_qna = QNA( message=prompt)              # declare prompt DB sample
db.session.add( new_qna)                    #} add&commit new sample to DB
db.session.commit()                         #}
new_qna = QNA( message=response)            # declare prompt DB sample
db.session.add( new_qna)                    #} add&commit new sample to DB
db.session.commit()                         #}
```

Рисунок 25 - Процедура сборки и отправки запроса в модель, сохранения ответа в таблицу класса QNA

```
# 3) Create another thesis for 'Thesis' DB
prompt = ''
prompt_temp = db.session.query(QNA).order_by(QNA.id.desc()).limit(2).all()
prompt = 'We had a dialogue where I said: ' + prompt_temp[1].message + \
        ' And you replied: ' + prompt_temp[0].message + '. In less than exactly 5 words explain what was the dialogue about.'
response_json = requests.post(
                    "http://127.0.0.1:8000/predict",
                    json={"prompt": prompt,
                          "input": ''}
                    )
response_str = response_json.json()["output"]
print( '\n\n\n######### THESIS PROMPT #########')
print( response_str)
cut = response_str.split("### Response:")
response = cut[1]
response = response[:100]                          # Truncating string to save context capacity

new_thesis = Thesis( thesis_message=response)       # declare prompt DB sample
db.session.add( new_thesis)                        #} add&commit new sample to DB
db.session.commit()                                #}
```

Рисунок 26 – Процедура создания нового тезиса и его загрузки в таблицу класса Thesis

```
# 4) Create another conversation for 'Conversation' DB
theses_amount = db.session.query(Thesis).count()
if (theses_amount>=5) and (theses_amount%5==0):
    prompt = ''
    prompt_temp = db.session.query(Thesis).order_by(Thesis.thesis_id.desc()).limit(5).all()
    prompt_temp = prompt_temp[4].thesis_message + '; ' + prompt_temp[3].thesis_message + '; ' + prompt_temp[2].thesis_message + '; ' \
                  + prompt_temp[1].thesis_message + '; ' + prompt_temp[0].thesis_message + '.'
    prompt = 'We had a conversation that included these theses: ' + prompt_temp + '. Shorten every thesis down to exactly no more than 2 words.'
    response_json = requests.post(
                        "http://127.0.0.1:8000/predict",
                        json={"prompt": prompt,
                              "input": ''}
                        )
    response_str = response_json.json()["output"]
    print( '\n\n\n######### CONVERSATION PROMPT #########')
    print( response_str)
    cut = response_str.split("### Response:")
    response = cut[1]
    response = response[:100]                                   # Truncating string to save context capacity
    new_conversation = Conversation( conversation_message=response) # declare prompt DB sample
    db.session.add( new_conversation)                           #} add&commit new sample to DB
    db.session.commit()                                         #}
```

Рисунок 27 – Процедура создания нового пересказа и его загрузки в таблицу класса Conversation

2.2 Разработка HTML страницы

Разработанная разметка HTML страницы веб-приложения “base.html” приведена на рисунках 28 и 29.

```
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AI Chat</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.css">
    <script src="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.js"></script>

    <style>
        .inline {
            display: inline-block;
            vertical-align: top;
        }
        body {
            background-image: url('static/background.png');
            background-size: cover;
        }
        .ui.segment {
            background-color: rgba(0, 0, 0, 0.25);
            border-radius: 15px;
        }
        @font-face { font-family: Condiment; src: url('static/Condiment-Regular.otf'); }
    </style>
</head>
```

Рисунок 28 – Хедер разметки страницы HTML

```
<!-- body -->
<body>
    <div style="width:400px; margin-top:50px;" class="ui container">
        <div style="margin-top:75px;">
            <h1 class="ui center aligned header" style="color:white; font-family:Condiment; font-size: 80px">AI Chat</h1>
        </div>
        <!-- text field & button  -->
        <!-- // <form> has <input> field to submit it
                to "/button" endpoint when <button> pressed  -->
        <div style="margin-top:30px; margin-bottom:25px;">
            <form class="ui form" action="/button" method="post" style="display:flex">
                <div class="inline" style="flex-grow:1;">
                    <input type="text" name="prompt" style="background-color:rgba(255, 255, 255, 0.405); border-radius:10px;" placeholder="Enter your message...">
                </div>
                <div class="inline" style="margin-left:5px;">
                    <button class="ui blue button" style="background-color:rgb(117, 0, 207); border-radius:20px;" id="button" type="submit">Send</button>
                </div>
            </form>
        </div>

        <!-- load DB samples -->
        {% for qna in qna_list[::-1] %}
        <div class="ui segment">
            <!-- print 'message' field -->
            <p class="ui big header" style="text-align:center; color:white; margin-bottom:5px; margin-top:10px;">{{ qna.message }}</p>

            <!-- Button (hyperlink) 'delete' endpoint -->
            <div style="text-align:right;">
                <a class="ui mini gray" style="color:rgba(255, 255, 255, 0.155)" href="/delete/{{ qna.id }}">[delete]</a>
            </div>
        </div>
        {% endfor %}
    </div>
</body>
```

Рисунок 29 – Тело разметки страницы HTML

2.3 Установка и запуск веб-приложения

Установка веб-приложения включает в себя сначала установку всех необходимых компонентов:

1) платформа CUDA Toolkit;
2) язык программирования Python;
3) python библиотеки pytorch, flask, flask-sqlalchemy, lit-gpt.

Затем следует загрузить веб-приложение на компьютер. Следует обратить внимание, что в сумме требуемое место на диске составляет 60 гигабайт. Большой объем требуемого пространства на диске во многом обусловлен большим размером языковой модели. Требуемый объем GPU памяти составляет 15500 мегабайт. Данный объем является относительно доступным для рядового разработчика и, как правило, требует одной графической карты верхней ценовой категории массового рынка.

Запуск веб-приложения осуществляется в несколько этапов:

1) активация виртуального окружения веб-приложения;
2) запуск языковой модели с помощью консольной команды “litgpt serve --checkpoint_dir .\out\finetune\lora\final\ --max_new_tokens 1024” из директории “venv\Lib\site-packages\litgpt”;
3) запуск развёртывания веб-приложения с помощью консольной команды “flask run” из корневой директории приложения;
4) перейти в браузере по адресу “http://127.0.0.1:5000” для доступа к интерфейсу.

2.4 Вывод

Разработанное веб-приложение реализовывает ключевые процедуры в файле “app.py” и разметку страницы в файле “base.html”.

Суммарное требуемое место на диске для веб-приложения составляет 60 гигабайт. Требуемый объем GPU памяти составляет 15500 мегабайт.

Установка требует наличия всех необходимых компонентов, а запуск включает несколько консольных команд и использует браузер в качестве доступа к интерфейсу.

3 ТЕСТИРОВАНИЕ ВЕБ-ПРИЛОЖЕНИЯ

3.1 Тестирование работы веб-приложения

В данном разделе представлено тестирование интерфейса веб-приложения. Для задачи проверки его работоспособности, будет проведено два диалога. Первый демонстрирует способность веб-приложения к корректным ответам на запросы, а второй показывает что оно учитывает содержания предыдущих сообщений. Результаты приведены на рисунках 30 и 31 соответственно диалогам.

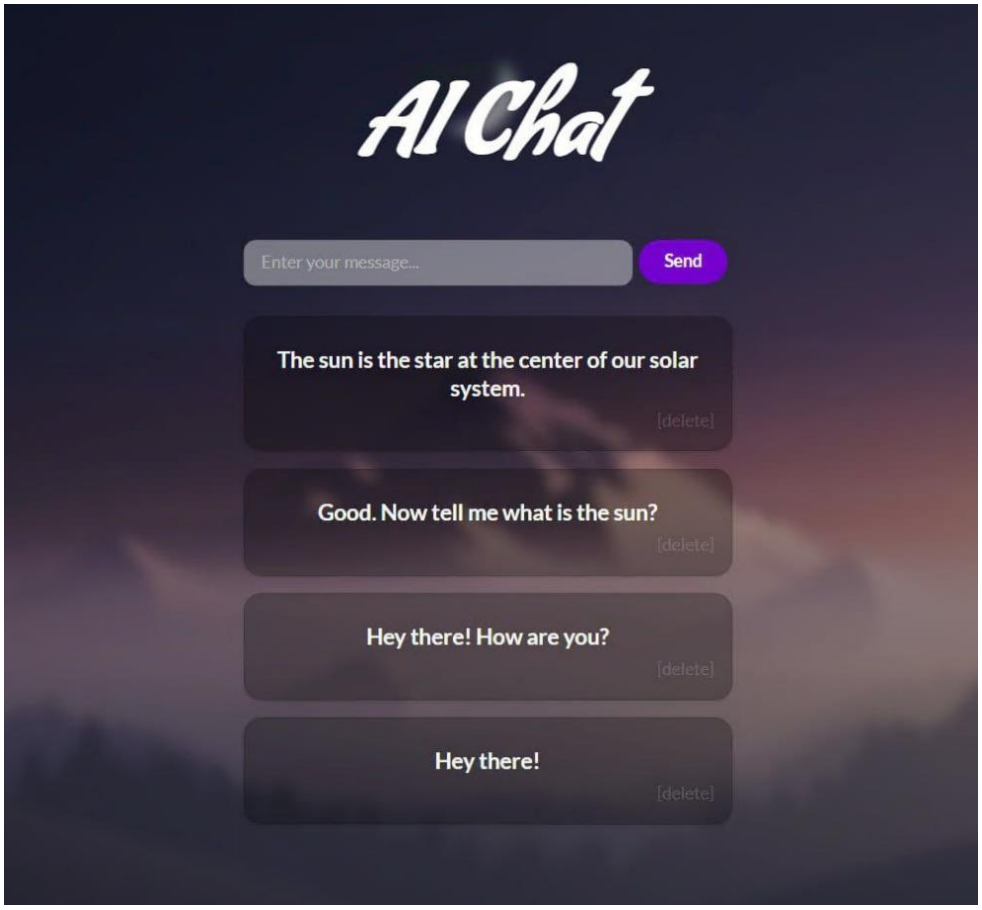


Рисунок 30 – Корректные ответы модели на запросы

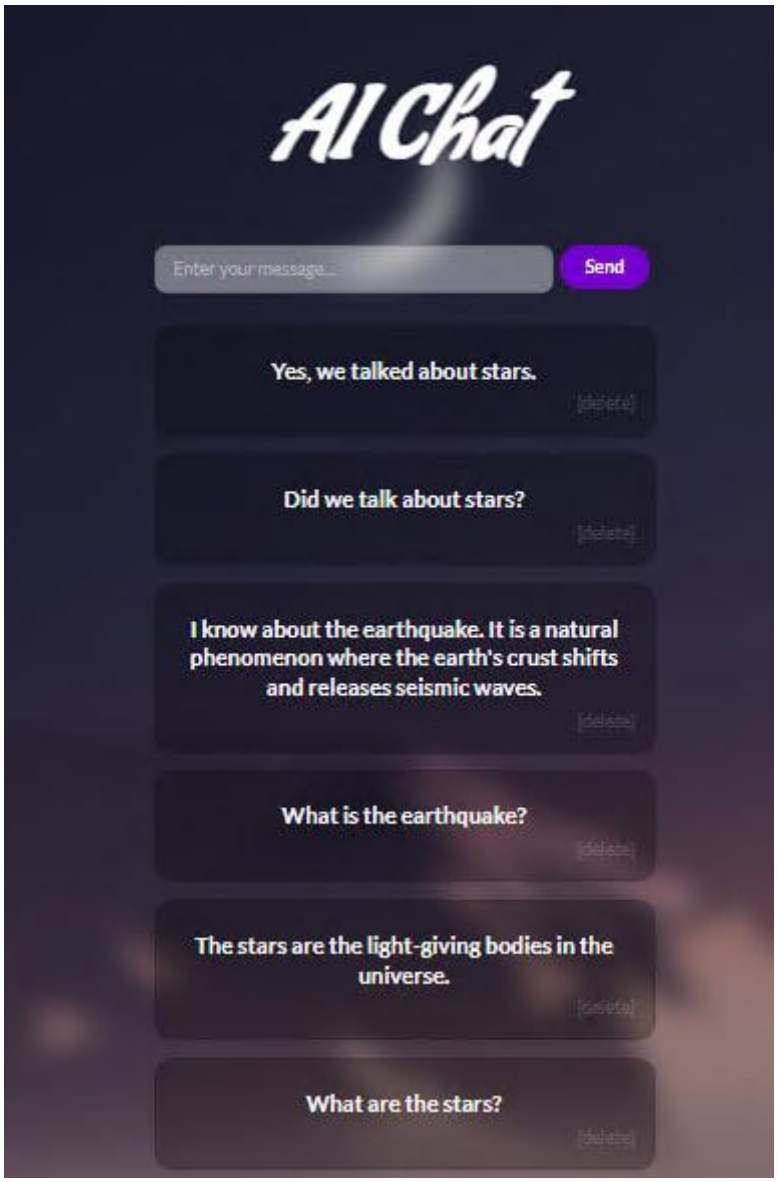


Рисунок 31 – Веб-приложение учитывает содержание предыдущих сообщений

3.2 Тестирование качества языковой модели

Качество используемой языковой модели оценивается результатами бенчмарков. Для оценки модели были использованы бенчмарки hellaswag, truthfulqa_2 и mmlu. Результаты оценки приведены на рисунке 32.

Tasks	Version	Filter	n-shot	Metric	Value		Stderr
mmlu	N/A	none	0	acc	0.2912	±	0.0190
- humanities	N/A	none	0	acc	0.2769	±	0.0396
- formal_logic	0	none	0	acc	0.1000	±	0.1000
- high_school_european_history	0	none	0	acc	0.3000	±	0.1528
- high_school_us_history	0	none	0	acc	0.1000	±	0.1000
- high_school_world_history	0	none	0	acc	0.5000	±	0.1667
- international_law	0	none	0	acc	0.4000	±	0.1633
- jurisprudence	0	none	0	acc	0.3000	±	0.1528
- logical_fallacies	0	none	0	acc	0.3000	±	0.1528
- moral_disputes	0	none	0	acc	0.4000	±	0.1633
- moral_scenarios	0	none	0	acc	0.1000	±	0.1000
- philosophy	0	none	0	acc	0.3000	±	0.1528
- prehistory	0	none	0	acc	0.3000	±	0.1528
- professional_law	0	none	0	acc	0.4000	±	0.1633
- world_religions	0	none	0	acc	0.1000	±	0.1000
- other	N/A	none	0	acc	0.2923	±	0.0412
- business_ethics	0	none	0	acc	0.4000	±	0.1633
- clinical_knowledge	0	none	0	acc	0.3000	±	0.1528
- college_medicine	0	none	0	acc	0.3000	±	0.1528
- global_facts	0	none	0	acc	0.3000	±	0.1528
- human_aging	0	none	0	acc	0.3000	±	0.1528
- management	0	none	0	acc	0.4000	±	0.1633
- marketing	0	none	0	acc	0.2000	±	0.1333
- medical_genetics	0	none	0	acc	0.3000	±	0.1528
- miscellaneous	0	none	0	acc	0.4000	±	0.1633
- nutrition	0	none	0	acc	0.2000	±	0.1333
- professional_accounting	0	none	0	acc	0.1000	±	0.1000
- professional_medicine	0	none	0	acc	0.4000	±	0.1633
- virology	0	none	0	acc	0.2000	±	0.1333
- social_sciences	N/A	none	0	acc	0.2583	±	0.0388
- econometrics	0	none	0	acc	0.3000	±	0.1528
- high_school_geography	0	none	0	acc	0.3000	±	0.1528
- high_school_government_and_politics	0	none	0	acc	0.1000	±	0.1000
- high_school_macroeconomics	0	none	0	acc	0.0000	±	0.0000
- high_school_microeconomics	0	none	0	acc	0.2000	±	0.1333
- high_school_psychology	0	none	0	acc	0.2000	±	0.1333
- human_sexuality	0	none	0	acc	0.3000	±	0.1528
- professional_psychology	0	none	0	acc	0.0000	±	0.0000
- public_relations	0	none	0	acc	0.5000	±	0.1667
- security_studies	0	none	0	acc	0.5000	±	0.1667
0 | acc | 0.5786 | ± | 0	0	none	0	acc	0.5000	±	0.1667
hellaswag	1	none	0	acc	0.4000	±	0.1633
		none	0	acc_norm	0.7000	±	0.1528

Groups	Version	Filter	n-shot	Metric	Value		Stderr
mmlu	N/A	none	0	acc	0.2912	±	0.0190
- humanities	N/A	none	0	acc	0.2769	±	0.0396
- other	N/A	none	0	acc	0.2923	±	0.0412
- social_sciences	N/A	none	0	acc	0.2583	±	0.0388
- stem	N/A	none	0	acc	0.3211	±	0.0337

Рисунок 32 – Бенчмарки настроенной модели

3.3 Руководство пользователя

При запуске приложения, пользователю предлагается ввести текстовый запрос в поле ввода. Как только запрос был введен, следует нажать на кнопку “Send” чтобы получить ответ веб-приложения. История предыдущих сообщений выводится автоматически снизу поля ввода для наглядности процесса и удобства пользования.

На рисунке 33 приведено выделение указанных элементов интерфейса.

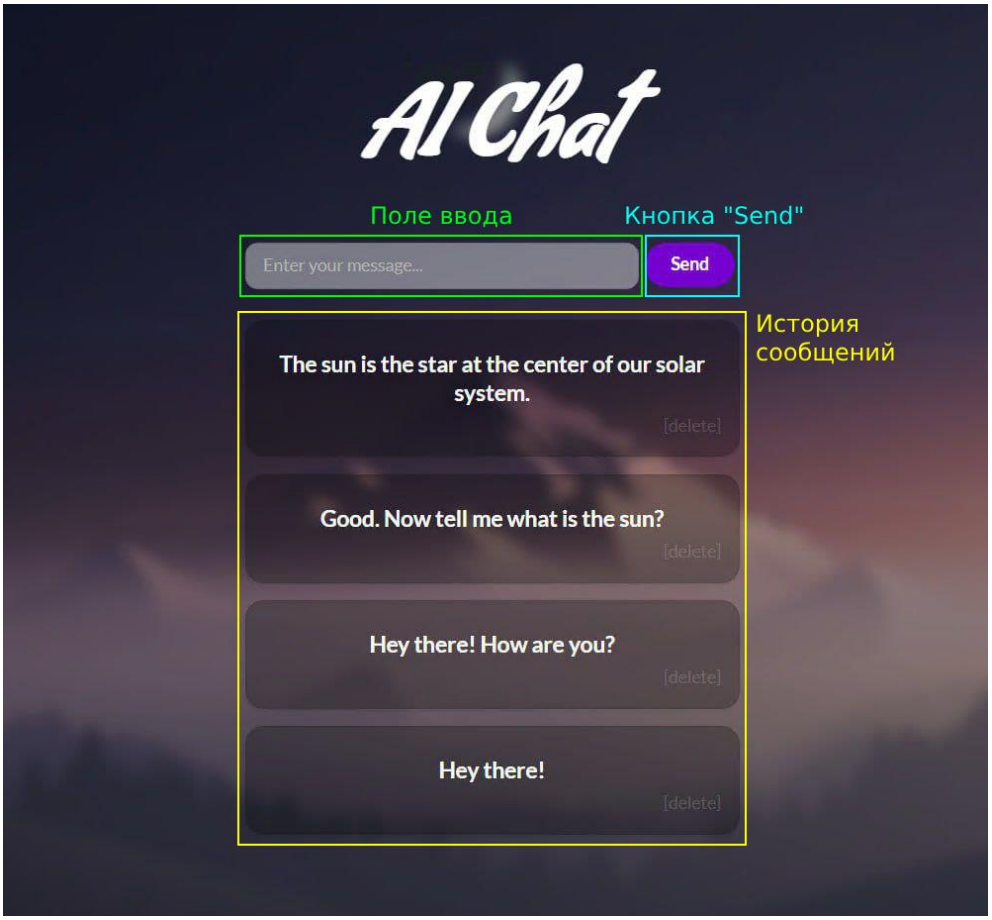


Рисунок 33 – Интерфейс с выделенными элементами

3.3 Руководство программиста

Программист может устанавливать некоторые аргументы запуска языковой модели, а также удалять историю сообщений посредством удаления таблиц базы данных без вреда к функционированию веб-приложения.

При запуске языковой модели, аргумент “--max_new_tokens” в консоли можно выставлять значениями степеней двойки вплоть до 1024. Это значение ограничивает длину ответа языковой модели.

Аргумент “--stream true” позволяет выводить ответ модели токен за токеном по отдельности.

По необходимости удаления истории сообщений, следует удалить обязательно все три файла таблиц базы данных, расположенных в папке “/instance”. Удаление этих файлов не влияет на работоспособность веб-приложения или языковой модели.

3.4 Вывод

Тестирование веб-приложения и языковой модели показывает полную работоспособность программного продукта. Интерфейс обеспечивает нужный функционал, а языковая модель адекватно реагирует на отправляемые запросы.

4 ОРГАНИЗАЦИОННО-ЭКОНОМИЧЕСКИЙ РАЗДЕЛ

В данном разделе приводится расчет затрат на разработку веб-приложения. Сначала необходимо определить стоимость человеко-дня программиста-разработчика в области NLP.

Стоимость одного человека-дня определяется по формуле:

$$C_{чд} = C_{сд} * \frac{(100 + K_{доп})(100 + K_{соц})}{100 + Д_{эвм} + Д_{нр} + Д_{пр}} \div 100,$$

где $C_{сд}$ – среднедневная заработанная плата сотрудников, руб.;

$K_{доп}$ – коэффициент дополнительной заработанной платы, % от основой;

$K_{соц}$ – коэффициент отчислений на социальные нужды, % от основной и дополнительной;

$Д_{эвм}$ – доля стоимости среднедневного расхода машинного времени;

$Д_{нр}$ – доля накладных расходов, % от заработанной платы;

$Д_{пр}$ – доля прочих расходов, % от заработанной платы.

Средняя заработанная плата программиста области NLP составляет *80000 руб*. Поэтому среднедневная заработанная плата составляет *3809,52 руб.*

Коэффициент дополнительной заработанной платы (премия по результатам работы) определяется в размере *30%* от основной заработанной платы.

Коэффициент отчислений на социальные нужды в среднем определен в размере *39%* от основной и дополнительной заработанной платы.

Доля стоимости среднедневного расхода машинного времени рассчитывается исходя из стоимости электроэнергии и затраты на техническое обслуживание. В среднем доля стоимости в виде отчисления от заработанной платы составляет *2%*.

Накладные расходы в данном случае не регламентируются, так как расходы на техническое обслуживание включены в расход машинного времени, а расходы на дополнительные материалы, как правило, не предусмотрены. Прочие расходы так же принимаются равными нулю.

Расчет стоимости одного человеко-дня, исходя из приведенных выше расходов:

$$C_{чд} = 3809,52 * \frac{(100 + 30)(100 + 39)}{100 + 2} \div 100 = 6748,83 \text{ руб.}$$

Далее следует оценить трудоёмкость разработки веб-приложения. Разработанное веб-приложение не относится к типу больших программных средств, поэтому возможна примерная оценка трудоемкости по формуле:

$$t = 3,6 * (n^{1,2}),$$

где n – размер программы в тыс. команд.

Трудоемкость оценивается в человеко-днях. Размер программы можно примерно оценить в 0,5 тыс. команд. Отсюда находим трудоемкость:

$$t = 3,6 * (0,5^{1,2}) = 1,57 \text{ человеко-дней}$$

Также следует учесть занимаемое время тонкой настройки модели, а также стоимости аренды серверного оборудования под данную задачу.

Тонкая настройка в условиях, подобных разработке данного проекта, занимает 10 часов, что равняется *0.42* человеко-дням. Складываем полученные значения:

$$t = 1,57 + 0,42 = 1,99 \text{ человеко-дней}$$

Рассчитаем оплату труда за написание программы по формуле:

$$К_{по} = t * С_{чд}$$

получаем

$$К_{по} = 1,99 * 6748,83 = 13439,17 \text{ руб.}$$

Таким образом, оплата труда программиста, разрабатывающего программное изделие, подобное данному в дипломном проекте, составила бы *13439,17* руб.

5 ЭКОЛОГИЯ И БЕЗОПАСНОСТЬ ЖИЗНЕДЕЯТЕЛЬНОСТИ

Согласно санитарноэпидемиологическим правилам и нормативам СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы» (утв. Главным государственным санитарным врачом РФ 30 мая 2003 г., с изменениями и дополнениями от 25 апреля 2007 г. и 3 сентября 2010г.), устанавливается ряд требований для работы за компьютером:

1. длительность работы студентов на занятиях с использованием ПЭВМ определяется курсом обучения, характером (ввод данных, программирование, отладка программ, редактирование и др.) и сложностью выполняемых заданий;

2. для студентов первого курса оптимальное время учебных занятий при работе с ВДТ или ПЭВМ составляет 1 ч, для студентов старших курсов – 2 ч с обязательным соблюдением между двумя академическими часами занятий перерыва длительностью 15-20 мин. Допускается время учебных занятий с ВДТ или ПЭВМ увеличивать для студентов первого курса до 2 ч, а для студентов старших курсов до 3 академических часов, при условии, что длительность учебных занятий в дисплейном классе (аудитории) не превышает 50% времени непосредственной работы на ВДТ или ПЭВМ и при соблюдении профилактических мероприятий: упражнения для глаз, физкультминутка и физкультпауза;

3. для предупреждения развития переутомления обязательными мероприятиями являются:

– проведение упражнений для глаз через каждые 20 - 25 мин работы за ВДТ или ПЭВМ;

– устройство перерывов после каждого академического часа занятий, независимо от учебного процесса, длительностью не менее 15 мин;

– проведение во время перерывов сквозного проветривания помещений с ВДТ или ПЭВМ с обязательным выходом из него студентов;

– осуществление во время перерывов упражнений физкультурной паузы в течение 3 – 4 мин;

– проведение упражнений физкультминутки в течение 1 - 2 мин для снятия локального утомления, которые выполняются индивидуально при появлении начальных признаков усталости;

– замена комплексов упражнений один раз в 2 - 3 недели;

4. физкультурные паузы следует проводить под руководством физорга, педагога или централизовано с помощью информации по местному радио на фоне умеренно звучащей приятной музыки.

ЗАКЛЮЧЕНИЕ

Характеристики изделия полностью отвечают техническим требованиям задания.

Разработанное веб-приложение не обладает недостатками аналогов и предлагает пользователю взаимодействие с языковой моделью с учетом истории сообщений в условиях локального запуска.

Веб-приложение имеет 3 ключевые процедуры, обеспечивающие функционирование веб-приложения:

1) процедура сборки и отправки запроса в модель, а также сохранения ответа в таблицу класса QNA;

2) процедура создания нового тезиса и его загрузки в таблицу класса Thesis;

3) процедура создания нового пересказа и его загрузки в таблицу класса Conversation.

Проект был разработан при помощи набора технологий, включающий архитектуру языковой модели decoder-only трансформер, языковую модели Falcon 7B, метод тонкой настройки LoRA с использованием набора данных Alpaca, библиотеку для работы с языковыми моделями lit-gpt, а также веб-фреймворк flask с использованием библиотеки flask-sqlalchemy для включения ORM функционала.

Требуемое место на диске составляет 60 гигабайт, минимальный объем GPU памяти размером в 15500 мегабайт.

Установка веб-приложения требует наличия на компьютере платформы CUDA Toolkit, языка программирования Python, Python библиотек pytorch, flask, flask-sqlalchemy и lit-gpt. При наличи перечисленных программных средств, достаточно скачать приложение на компьютер чтобы приступить к запуску и работе.

Запуск выполняется серией этапов, включающие активацию виртуального окружения веб-приложения, запуск языковой модели и развёртывания веб-приложения с помощью консольных команд и в конце открытие специального адреса в браузере для доступа к интерфейсу.

Тестирование веб-приложения показало исправную работу как интерфейса, так и языковой модели. Языковая модель обладает удовлетворительной точностью и адекватно отвечает на запросы пользователя.

Практическая ценность веб-приложения заключается в способности работать и без доступа в Интернет, а также дешевом способе учёта истории сообщений.

Недостатком веб-приложения являются высокие требования к свободному месту на диске.

СПИСОК ИСПОЛЬЗОВАННЫХ ИСТОЧНИКОВ

1. Attention Is All You Need. [Электронный ресурс.] – Режим доступа: https://arxiv.org/abs/1706.03762 свободный. – Загл. с экрана.

2. How Powerful are Decoder-Only Transformer Neural Models? [Электронный ресурс.] – Режим доступа: https://arxiv.org/abs/2305.17026 свободный. – Загл. с экрана.

3. The Falcon Series of Open Language Models. [Электронный ресурс.] – Режим доступа: https://arxiv.org/abs/2311.16867 свободный. – Загл. с экрана.

4. LoRA: Low-Rank Adaptation of Large Language Models. [Электронный ресурс.] – Режим доступа: https://arxiv.org/abs/2106.09685 свободный. – Загл. с экрана.

5. LLM-Adapters: An Adapter Family for Parameter-Efficient Fine-Tuning of Large Language Models. [Электронный ресурс.] – Режим доступа: https://arxiv.org/abs/2304.01933 свободный. – Загл. с экрана.

6. Introducing the next generation of Claude. [Электронный ресурс.] – Режим доступа: https://www.anthropic.com/news/claude-3-family свободный. – Загл. с экрана.

7. СанПиН 2.2.2/2.4.1340-03 «Гигиенические требования к персональным электронно-вычислительным машинам и организации работы». [Электронный ресурс.] – Режим доступа: https://web.archive.org/web/20190729082811/http://docs.cntd.ru/document/901865498 свободный. – Загл. с экрана.

ПРИЛОЖЕНИЕ А

(обязательное)

Код скрипта веб-приложения

```
from flask import Flask, render_template, request, redirect, url_for
from flask_sqlalchemy import SQLAlchemy
import requests, json

# --- App object
app = Flask(__name__)
app.app_context().push()       # flask_sqlalchemy troubleshooting

# --- Database
# setup DB files for the app & create DB object
app.config['SQLALCHEMY_DATABASE_URI'] = 'sqlite:///qna_db.sqlite'
app.config['SQLALCHEMY_BINDS'] = {'thesis_db': 'sqlite:///thesis_db.sqlite',
                                  'conversation_db': 'sqlite:///conversation_db.sqlite'}
app.config['SQLALCHEMY_TRACK_MODIFICATIONS'] = False
db = SQLAlchemy(app)

# define DB models //table of 2 columns
class QNA( db.Model):
    id = db.Column( db.Integer, primary_key=True)
    message = db.Column( db.String(2000))
class Thesis( db.Model):
    __bind_key__ = 'thesis_db'
    thesis_id = db.Column( db.Integer, primary_key=True)
    thesis_message = db.Column( db.String(2000))
class Conversation( db.Model):
    __bind_key__ = 'conversation_db'
    conversation_id = db.Column( db.Integer, primary_key=True)
    conversation_message = db.Column( db.String(2000))

# create and init DBs
```

```
db.create_all(bind_key=[None, 'thesis_db', 'conversation_db'])

# -- Endpoints //interface
# 1. Homepage endpoint
@app.get("/")
def home():
    qna_list = db.session.query(QNA).all()                  # load the DB
    return render_template( "base.html", qna_list=qna_list)   # render html using the loaded DB

# 2. Send button endpoint
@app.post("/button")                        #} This send button endpoint does 3 things:
                                    #} 1) Builds, Sends prompt & saves response
                                    #} 2) Adds prompt & response to 'QNA' DB
                                    #} 3) Creates another thesis for 'Thesis' DB
                                    #} 4) Creates another conversation for 'Conversation' DB
                                    #} 5) Refreshes page
def button():
    global theses_amount
    # 1) Send prompt & save response
    # Building prompt: Loading 5 last theses from 'Thesis' DB
    thesis = ''
    thesis_temp = db.session.query(Thesis).order_by(Thesis.thesis_id.desc()).limit(5).all()
    for entry in thesis_temp:
        thesis = thesis+entry.thesis_message+'; '
    # Building prompt: Loading 5 last conversations from 'Conversation' DB
    conversation = ''
    conversation_temp                                                                    =
db.session.query(Conversation).order_by(Conversation.conversation_id.desc()).limit(5).all()
    for entry in conversation_temp:
        conversation = conversation+entry.conversation_message+'; '

    input = thesis+conversation

    # Building prompt: Get contents of <input> named "prompt" from <form>
```

```
        prompt = request.form.get("prompt")

        # Sending built prompt & saving model response
        response_json = requests.post(
                        "http://127.0.0.1:8000/predict",
                        json={"prompt": prompt,
                            "input": input}
                        )
        response_str = response_json.json()["output"]
        print( '\n\n\n######### CHAT PROMPT #########')
        print( response_str)
        cut = response_str.split("### Response:")
        response = cut[1]

        # 2) Add prompt & response to 'QNA' DB
        new_qna = QNA( message=prompt)          # declare prompt DB sample
        db.session.add( new_qna)              #} add&commit new sample to DB
        db.session.commit()                  #}
        new_qna = QNA( message=response)        # declare prompt DB sample
        db.session.add( new_qna)              #} add&commit new sample to DB
        db.session.commit()                  #}

        # 3) Create another thesis for 'Thesis' DB
        prompt = ''
        prompt_temp = db.session.query(QNA).order_by(QNA.id.desc()).limit(2).all()
        prompt = 'We had a dialogue where I said: ' + prompt_temp[1].message + ' And you replied:
' + prompt_temp[0].message + '. In less than exactly 5 words explain what was the dialogue about.'
        response_json = requests.post(
                        "http://127.0.0.1:8000/predict",
                        json={"prompt": prompt,
                            "input": ''}
                        )
        response_str = response_json.json()["output"]
        print( '\n\n\n######### THESIS PROMPT #########')
```

```
            print( response_str)
            cut = response_str.split("### Response:")
            response = cut[1]
            response = response[:100]                    # Truncating string to save context capacity

            new_thesis = Thesis( thesis_message=response)      # declare prompt DB sample
            db.session.add( new_thesis)                    #} add&commit new sample to DB
            db.session.commit()                           #}

            # 4) Create another conversation for 'Conversation' DB
            theses_amount = db.session.query(Thesis).count()
            if (theses_amount>=5) and (theses_amount%5==0):
                prompt = ''
                prompt_temp = db.session.query(Thesis).order_by(Thesis.thesis_id.desc()).limit(5).all()
                prompt_temp = prompt_temp[4].thesis_message + '; ' + prompt_temp[3].thesis_message + '; ' + prompt_temp[2].thesis_message + '; ' + prompt_temp[1].thesis_message + '; ' + prompt_temp[0].thesis_message + '.'
                prompt = 'We had a conversation that included these theses: ' + prompt_temp + '. Shorten every thesis down to exactly no more than 2 words.'
                response_json = requests.post(
                                "http://127.0.0.1:8000/predict",
                                json={"prompt": prompt,
                                    "input": ''}
                                )
                response_str = response_json.json()["output"]
                print( '\n\n\n######### CONVERSATION PROMPT #########')
                print( response_str)
                cut = response_str.split("### Response:")
                response = cut[1]
                response = response[:100]                              # Truncating string to save context capacity
                new_conversation = Conversation( conversation_message=response) # declare prompt DB sample
                db.session.add( new_conversation)                      #} add&commit new sample to DB
```

```
        db.session.commit()                                    #}

    # 5) Refresh page
    return redirect( url_for( "home"))          # redirect to home page

# 3. Delete button endpoint
@app.get("/delete/<int:qna_id>")
def delete(qna_id):
    qna = db.session.query(QNA).filter(QNA.id == qna_id).first()
    db.session.delete(qna)
    db.session.commit()
    return redirect( url_for( "home"))
```

ПРИЛОЖЕНИЕ Б

(обязательное)

Код HTML страницы

```
<!DOCTYPE html>
<html lang="en">
<!-- header -->
<head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>AI Chat</title>

    <link rel="stylesheet" href="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.css">
    <script src="https://cdn.jsdelivr.net/npm/semantic-ui@2.4.2/dist/semantic.min.js"></script>

    <style>
        .inline {
            display: inline-block;
            vertical-align: top;
        }
        body {
            background-image: url('static/background.png');
            background-size: cover;
        }
        .ui.segment {
            background-color: rgba(0, 0, 0, 0.25);
            border-radius: 15px;
        }
        @font-face { font-family: Condiment; src: url('static/Condiment-Regular.otf'); }
    </style>
</head>
```

```
<!-- body -->
<body>
  <div style="width:400px; margin-top:50px;" class="ui container">
    <div style="margin-top:75px;">
      <h1 class="ui center aligned header" style="color:white; font-family:Condiment; font-size: 80px">AI Chat</h1>
    </div>
    <!-- text field & button  -->
    <!-- // <form> has <input> field to submit it
        to "/button" endpoint when <button> pressed  -->
    <div style="margin-top:30px; margin-bottom:25px;">
      <form class="ui form" action="/button" method="post" style="display:flex">
        <div class="inline" style="flex-grow:1;">
          <input type="text" name="prompt" style="background-color:rgba(255, 255, 255, 0.405); border-radius:10px;" placeholder="Enter your message...">
        </div>
        <div class="inline" style="margin-left:5px;">
          <button class="ui blue button" style="background-color:rgb(117, 0, 207); border-radius:20px;" id="button" type="submit">Send</button>
        </div>
      </form>
    </div>

    <!-- load DB samples -->
    {% for qna in qna_list[::-1] %}
    <div class="ui segment">
      <!-- print 'message' field -->
      <p class="ui big header" style="text-align:center; color:white; margin-bottom:5px; margin-top:10px;">{{ qna.message }}</p>

      <!-- Button (hyperlink) 'delete' endpoint -->
      <div style="text-align:right;">
        <a class="ui mini gray" style="color:rgba(255, 255, 255, 0.155)" href="/delete/{{ qna.id }}">[delete]</a>
```

```
            </div>
        </div>
        {% endfor %}
    </div>
</body>

</html>
```