Список 01 – Домашнее Задание

Введение в LLM и Prompt Engineering с RAG

Предмед: Введение в профессиональную деятельность

Преподаватель: Хольгер Эспинола Ривера

1. **Введение в LLM.** Следуя инструкциям, содержащимся в документе deepseek_installer.txt (из lab01), выполните на локальном компьютере следующие модели:

- DeepSeek-R1:1.5b (версия модели рассуждений с 1.5 миллиардами параметров)
- Llama3.2:3b (версия предварительно обученной модели с квантованием Q4)

Выполните модели в следующих 3 сценариях:

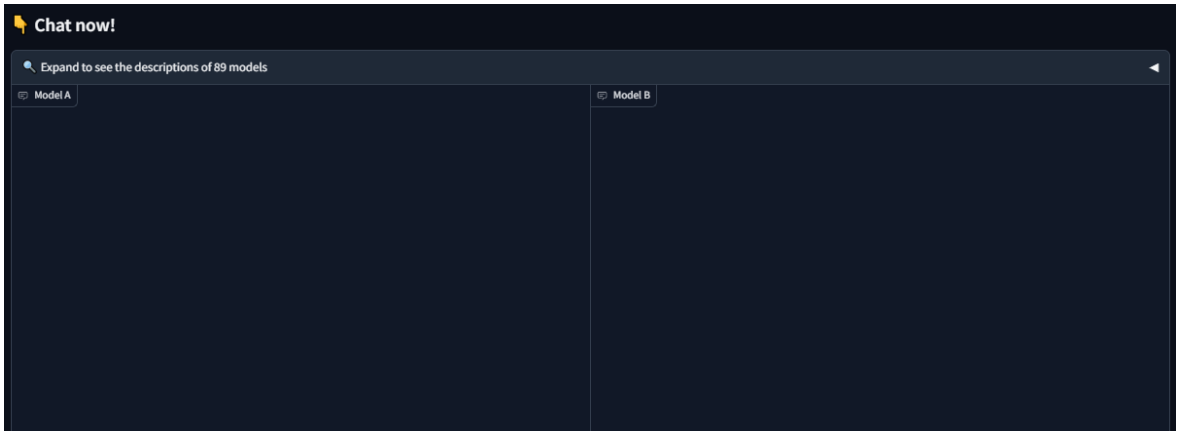
a) запуск локально в терминале

b) запуск локально с использованием контейнера Docker + веб-интерфейса

c) запуск локально с использованием LM Studio Desktop

Принимая во внимание стандартный набор данных MMLU для бенчмарков, выберите 3 вопроса: 1 для теста по анатомии, 1 для компьютерных наук и 1 для математики. Используйте ссылку ниже в качестве справки: https://www.kaggle.com/datasets/lizhecheng/mmlu-dataset

Задача: Сравните ответы, полученные DeepSeek-R1:1.5b и Llama3.2:3.2. Составьте отчет об этом со снимками экрана выполнения подсказок в каждом сценарии. Вы можете сделать это в виде таблицы из двух столбцов, сравнивая каждый полученный ответ на каждый вопрос, сравнивая 2 модели и комментируя, какая модель выполняет лучшие ответы. Возьмите в качестве справки изображение ниже.



2. Оперативное проектирование с RAG. Возьмите проект Web-Chatbot (опубликован в Github - lab02), разработанный с помощью LLM и RAG, и, принимая во внимание следующую структуру папок и файлов, выполните приложение локально на компьютере:

Ссылка: https://github.com/HoltechHard/Professional_Proj_Activity-40001-2

- faiss_db: папка, которая содержит базу данных векторных вложений

- history: папка, которая содержит историю взаимодействий вопрос-ответ в чат-боте

- scrap: папка, которая содержит простой текст извлеченной информации с веб-сайта

- scrapper.py: скрипт python для сканирования и извлечения контента с веб-сайта

- summarization.py: скрипт python для генерации сводки извлеченного веб-контента

- ingest.py: скрипт python для создания векторной базы данных с векторными вложениями фрагментов извлеченного контента с веб-сайта

- chatbot.py: скрипт python, который содержит процессы индексации, извлечения и генерации ответа из заданного вопроса в чатбот

- app.py: основной скрипт python, который содержит интеграцию streamlit с реализацией бэкэнда langchain RAG. Для запуска приложения необходимо выполнить следующую команду:

```
$ streamlit run app.py
```

- requirements.txt: текстовый файл, который содержит список пакетов python, необходимых для установки для правильного запуска приложения. Для правильной установки необходимо выполнить следующую команду:

```
$ pip install -r requirements.txt
```

Задача:

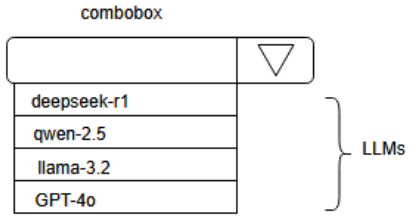
1) Адаптировать исходный код приложения Web-Chatbot, чтобы пользователь системы имел возможность выбрать 4 LLM разных поставщиков (например, deepseek, qwen, gpt-4o, llama, например).

2) Процессы скрапинга и встраивания не меняются. В процессах суммирования и чатбота пользователь должен видеть выпадающий список с 4 вариантами LLM и выбирать его, а скрипты python должны выполнять суммирование или чатбот в соответствии с выбранной моделью.

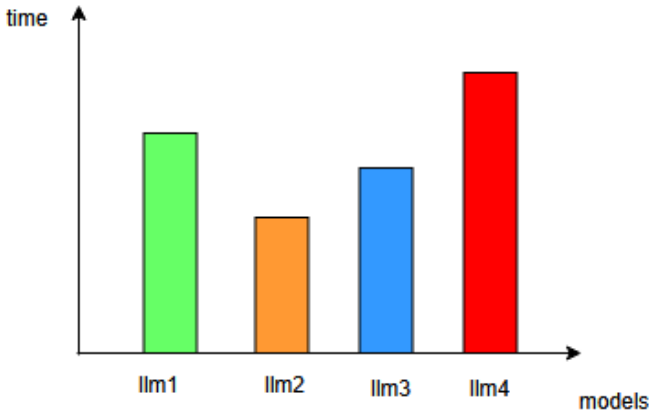
3) Выберите один веб-сайт, содержащий разумный объем технического контента. Сформулируйте резюмирование и 1 вопрос-ответ для каждого из

4 выбранных LLM. Реализуйте функцию Python для расчета времени (в секундах), которое тратит система на получение ответа. Нарисуйте контрольную точку в соответствии с рисунком ниже:

Combo box



Benchmark



4) Составьте отчет, сравнивающий качество ответов для каждого ответа LLM на вопрос, согласно вашим критериям. Составьте таблицу с перекрестным сравнением ответов каждого LLM на 1 технический вопрос, связанный с просканированным веб-сайтом.