Список 01 – Домашнее Задание

Введение в LLM и Prompt Engineering с RAG

Предмед: Введение в профессиональную деятельность

Преподаватель: Хольгер Эспинола Ривера

1. **Introduction to LLM.** Following the instructions contained in document deepseek_installer.txt (from lab01), execute in local computer the next models:

- DeepSeek-R1:1.5b (version of reasoning model with 1.5 billion of parameters)
- Llama3.2:3b (version of pre-trained model with quantization Q4)

Execute the models in the next 3 scenarios:

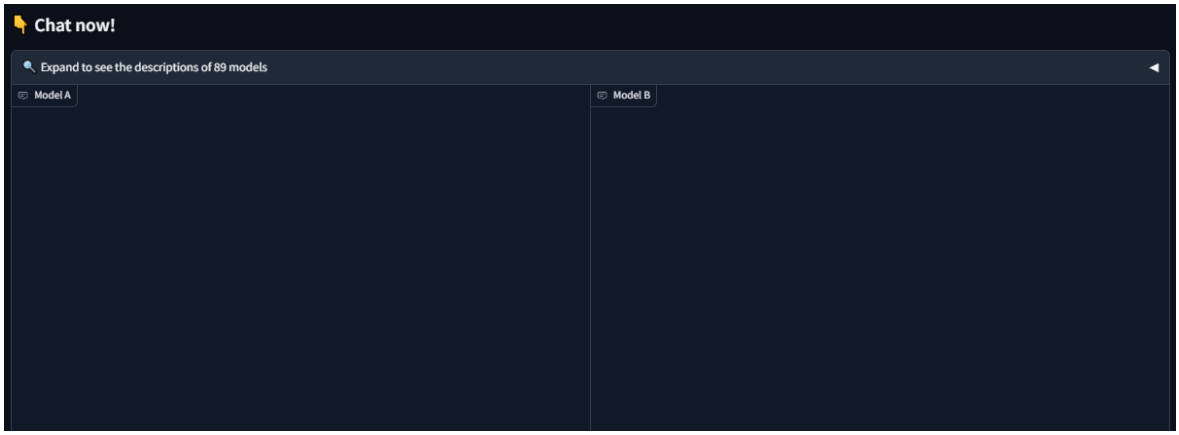
a) running locally in terminal

b) running locally using Docker container + Web UI

c) running locally using LM Studio Desktop

Taking in account the standard MMLU Dataset for benchmarks, select 3 questions: 1 for anatomy test, 1 for computer science and 1 for mathematics. Use the link bellow as reference: https://www.kaggle.com/datasets/lizhecheng/mmlu-dataset

Task: Compare the answers obtained by DeepSeek-R1:1.5b and Llama3.2:3.2. Make a report about it with screen shots of execution of prompts in each scenario. You can do it as a two-column table comparing each answer obtained for each question comparing the 2 models and comment what model performs better answers. Take as reference the image bellow.



2. Prompt Engineering with RAG. Take the project of **Web-Chatbot** (published in Github - lab02) developed with LLM and RAG and taking in account the next structure of folders and files, execute the application locally in computer:

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

- faiss_db: folder which contains the database of vector embeddings

- history: folder which contains the history of interactions question-answering in chatbot

- scrap: folder which contains the plain text of scraped information from web site

- scrapper.py: python script to crawl and scrap content from web site

- summarization.py: python script to generate summary of web scraped content

- ingest.py: python script to build a vector database with vector embeddings of chucks of scraped content from web site

- chatbot.py: python script which contain processes of index, retrieve and generate answer from given question to chatbot

- app.py: main python script which contain the integration of streamlit with langchain backend implementation of RAG. To execute application is necessary run the next command:

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

- requirements.txt: text file which contains the list of python packages necessary to install in order to run the application appropriately. To install correctly is necessary run the next command:

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

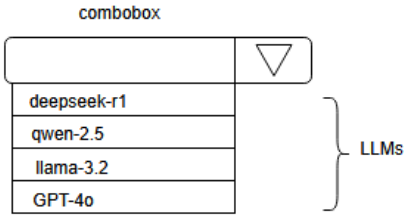
Task:

1) Adapt the source code of Web-Chatbot application to user of system have possibility to choose 4 LLMs of different providers (for example: deepseek, qwen, gpt-4o, llama, for example).

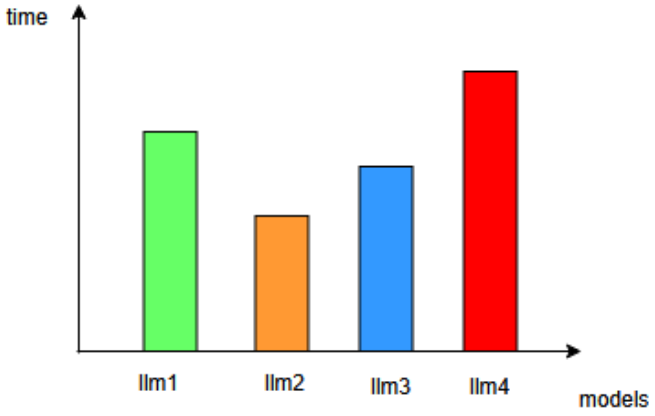
2) The processes of scraping and embeddings not change. In processes of summarization and chatbot, user need see a combo box with the 4 LLM options and select it, and python scripts need execute the summarization or chatbot according the selected model.

3) Select one web site which contains reasonable volume of technical content. Formulate the summarization and 1 question-answering for each of the 4 LLM selected. Implement a python function to calculate how much time (in seconds) the system spends until obtaining the answer. Draw a benchmark according the figure bellow:

Combo box



Benchmark



4) Make a report comparing the quality of answers for each LLM answer given a question, according your criteria. Make a table with cross-comparison of the answers of each LLM for 1 technical question related of the web-site scraped.