Building Human-Centered AI Applications

Homework 02 — URL Summarizer with Multiple LLMs

Create a Streamlit application that uses different large language models (LLMs)

This application will allow users to input:

- A Web page URL
- The type of summary
- An output language
- The LLM to use

Create a multi-page HW application

1. Create a multi-page app with the title of "HW Manager"
2. Add the previously created HW1.py file as a page in the main app
    a. The file itself should be in a sub-directory called "HW"
3. Add a new file named "HW2.py" to the "HW" folder and copy the contents of Lab2.py into it

Update the HW2.py to read a URL (rather than a PDF or TXT file)

4. Let the user enter a URL (at the top of the screen, not in the sidebar)
5. In the sidebar, keep the menu for the type of summary (from Lab 2)
6. Display the summary (as you did previously), but now it is for a URL
7. Below is a function to read the content from a URL and return text

```
import requests
from bs4 import BeautifulSoup


def read_url_content(url):
    try:
        response = requests.get(url)
        response.raise_for_status()  # Raise an exception for HTTP errors
        soup = BeautifulSoup(response.content, 'html.parser')
        return soup.get_text()
    except requests.RequestException as e:
        print(f"Error reading {url}: {e}")
        return None
```

Select the language to output

8. Create a dropdown menu to select the output language (e.g., English, French) with at least 3 options
9. Update the prompt to ensure you are outputting the text in the correct language

Select the LLM to use

10. On the sidebar, add the option to choose between different LLMs (in addition to the 'use advanced model' checkbox from Lab 2)
11. Get keys for one other LLM APIs in addition to OpenAI (e.g., Claude, Gemini)
    a. After the user selects an LLM, the application should switch to using the selected LLM

Homework 02 — URL Summarizer with Multiple LLMs

   b. Make sure the key is valid for that LLM
12. Evaluate two LLMs with their 'advanced' models (need to make sure you have the correct key for the selected API)
   a. Which LMM is "best" — explain your logic
13. Do the same evaluation when using the less expensive model
   a. Which LLM is "best"?
   b. Are the more expensive models much better for the task?

Deploy the application

14. Deploy the application to Streamlit Community Cloud
15. Use secrets.toml to store your API keys
16. Make sure to update requirements.txt as needed

What to submit:

1. The link to your app
2. Your Python code file (.py)
3. Your responses to items 12 and 13 (comparison of model outputs)