Exercise - Prototyping with LLMs

Learning goal

- Develop skills in prototyping with LLMs through pracice in an open-ended setting.

Instructions

- Think of a business case where the application of an LLM adds value (improves user experience, reduces time/cost, enables a new functionality, etc.)
- Implement a Streamlit or Dash prototype that showcases the desired functionality and that makes calls to a LLM.
- **It is possible to start from the prototype you used in Assignment 1, if you consider that you can add an interesting functionality using LLMs**. Likewise, it is possible to create a prototype that is related to work done in another course, capstone or MSc thesis.

Expectations for LLMs

The LLM use should be non-straighforward. To be specific, a "striaghforward" example is one which calls a LLM with a simple prompt and has a direct processing of the results (e.g. you build an app where a user enters ingredients, call a LLM with a prompt recommend 3 recipes based on the ingredients specified by the user" and just format the output in the front-end without any processing).

In contrast, non-straightforward examples include (but are not limited to):

- A case where a simple prompt is not sufficient for obtaining the desired value and you need to do multiple refinements to the prompt until it is sophisticated enough to the required job.
- A case where the complexity is in the way the LLM output needs to be processed by your python program (e.g. you need to combine the LLM with an API, or the LLM produces data that feeds interesting plots, or any example where there is some valuable python work that post-processes the output of the LLM).
- Multi-call use cases like the ones seen in class (chatbot, tools, RAG).

You can use `cohere` (as seen in class) as your LLM, but any other LLM API is valid.

Deliverable

A submission link will be available in Moodle.

1. Submit the folder with your code and files as a single compressed file, or a link to a github repository containing your files (see instructions below).

Make sure that the code can be reproduced when someone downloads your submission. A good practice is to share it with a classmate or to download/uncompress it to a different folder and check. Normally, if you put all the files needed (e.g. datasets) into the same folder, and use relative paths (e.g. `FILE=dataset.csv` instead `FILE="C:\Users\Jose\My_data\MIBA\dataset.csv`) it should be fine.

2. Also submit a very short document of max 1 page indicating:
   - What is the utility of the prototype?
   - What are the main design decisions chosen?
   - Main difficulties found?
3. Take a screen capture with Microsoft Stream showing how you launch your prototype and demonstrating its use. Submit the link to the video.

Optional: Pushing the code to a repository

Instead of submitting the code you can practice adding the code to a repository:

- Create an empty repository in your Github account (can be public or private).
- After creating the repository, git will provide some instructions on how to push your files. You can also follow these instructions.
- To learn more about github, feel free to complete the Github concepts datacamp course.

Reminder

Never put your API keys in the repository!