CSC 510 P1d1 Group 22: Yash Dive, Stephen Liu, Atharva Waingankar, Ory Wickizer

1. What are the pain points in using LLMs?

   Some pain points and biggest headaches that our team faced when using LLMs was having it generate use cases that were properly structured. For example, many of the LLMs we test had difficulties with following the strict format provided by Preconditions, Main Flow, Subflows, and Alternative Flows. As a result we had to proofread every output and manually check it to make sure that it's in the correct format and contains coherent and relevant information making time consuming when generating 30 - 40 use cases as many of the LLMs output over 20 pages of material.

2. Any surprises? Eg different conclusions from LLMs?

   We think it was surprising to see that when we tried to cut ideas down for the MVP, it wasn't that great since we suspect that LLMs are more designed for generating ideas and content and gave general/generic advice which wasn't really that helpful when making decisions about which features to leave out. Another surprise that we found was that our Gemma 3 and DeepSeek R1 models gave a lot of use cases that didn't really match with the documents provided in the RAG System. And that many of the LLM output ideas were pretty much ideas within the box and not ideas that were really special or creative even after we did a significant amount of prompting. We think this happened with both of the LLM models we tested where it kept on giving more in-depth information on the ideas it had already outputted instead of giving ideas or aspects that might have been left out or missing.

3. What worked best?

   For our group, the most effective strategy that we used was doing few-shot prompting by giving 1 - 3 examples of the output format that we expected it to give. This seems to give the most reliable results since by giving it a pattern to copy, we saw that the output generated by the LLMs had fewer mistakes in the format and structure along with having the output make more logical sense. Overall, few-shot prompting seems to make the LLMs a more reliable and useful tool for finding ideas. We also found that it may be helpful by adding a list of use case titles so that the model can just look through the source material for the use cases instead of needing to come up with what the use cases will be about on its own. We found that it made the outputs more specific.

4. What worked worst?

For our group, the least effective strategy that we used was doing zero-shot prompting. For example, we first tried providing the LLM models with the source material and instructions like “write 10 use cases” which seemed to fail. The output format and summary that both models gave were either inconsistent or incoherent use cases. This happened a lot with the DeepSeek model where it basically returns the same 3 use cases repeatedly.

5. What pre-post processing was useful for structuring the import prompts, then summarizing the output?

   Our group found that pre-processing was removing any special character, random spaces, or new lines when copying and pasting example outputs from Docs. We think that this helped make the prompt more readable to humans and it also could prevent odd artifacts in the LLM output. We also found that giving it clear instructions in the prompt seems to improve the output. These two pre-processing steps helped decrease the amount of postprocessing that we did to check that it was in the correct format and made logical sense.

   For post-processing, our group found that doing human reviews on the output was pretty helpful. The human review was to check and fix any formatting or structural errors and any logical issues that would affect the flow and to make sure that it was indeed the content that matched what we asked for in the beginning.

6. Did you find any best/worst prompting strategies?
   a. Best prompting strategy:

      A good prompting strategy we found was to do few-shot prompting where you provide a few complete and high quality examples of the desired output. Another strategy that we found to be helpful was to do batch outputs (for example do like 5 use cases at a time) as it seemed like it was more consistent and logical.

   b. Worst prompting strategy:

      A prompting strategy that we first tried and later on avoided due to getting poor results was zero-shot prompting because it seems like the model didn’t really understand the full context of what it is being asked to do. Another strategy that we found to do poorly was having the LLM model generate large outputs at a time (for example 30 use cases) since from the output it seemed like the LLM was unable to maintain consistency.