CSC 510 (001): Software Engineering (Proj1d1) - Group 3

Aum Pandya Pranav Bhagwat Tayo Olukotun

1 What are the pain points in using LLMs?

- LLMs confidently invent facts (hallucinate), which is dangerous for precise requirements.
- You can't trust their output for critical details without a human checking everything, especially for generated code which must be tested as it's based on pattern matching, not true reasoning.
- The need for a human to check everything and engage in the iterative, time-consuming process of prompt engineering slows down the LLM's perceived speed advantage.
- The small upfront cost of a quick, lazy prompt is deceptive, as poor-quality prompts lead to ambiguous and inaccurate responses that cost more time to fix in the long run.
- LLMs don't know project-specific jargon and give generic answers unless you provide necessary background through contextual prompting to help them understand nuances.
- It's very hard to trace an LLM's output back to a source, creating verification nightmares, although Chain of Thought (CoT) prompting can offer some interpretability by showing the model's reasoning steps.
- Getting the perfect prompt feels like guesswork because many factors like word choice, tone, and structure can wildly change the output, making experimentation crucial.
- Even with the same model and prompt, the output can have random differences in wording, as the model may randomly break ties between tokens with equal probability.
- LLMs can get stuck in a "repetition loop bug," a common issue made worse by inappropriate temperature and top-p settings, where they repeat the same phrases over and over.

2 Any surprises?

- LLMs are better at generative tasks like creating new requirements than just summarizing existing text.
- It was surprising how different LLMs (or even different versions of the same model) and even minimal temperature value could read the same document and arrive at completely different conclusions.
- Sometimes, a very simple, concise, and clear prompt worked much better than a complex one.
- The simple phrase "Let's think step-by-step" dramatically improves the model's reasoning, a technique known as Chain of Thought (CoT) prompting.
- Adding just one good example (one-shot prompting) provides a huge boost in quality, and while using 3-5 examples is a good rule of thumb, the benefit often diminishes after five.

3 What worked best?

- Using zero-shot prompts, which provide no examples and only a task description, is effective for brainstorming or exploring a topic.
- LLMs are well-suited for creating a "first draft" of requirements by summarizing or extracting information from large, messy documents.
- Brainstorming creative outputs like user stories and acceptance criteria from raw user feedback works well when guided by specific context.
- A core capability of LLMs is language translation, which is useful for converting technical developer notes into easy-to-understand business requirements.
- Using role prompting, such as "Act as a senior engineer," effectively frames the model's tone, vocabulary, and knowledge base.
- Giving the model a structured format to fill in, like a JSON schema, helps it focus on relevant information and produce clean output.

- Specifying the audience (e.g., ”write for a senior executive”) gives the model a blueprint for the desired tone and style, improving relevance.

4 What worked worst?

- Using lazy, zero-shot prompts for tasks needing reliable, structured output like JSON often fails; few-shot examples and a system prompt are usually necessary.
- Asking for a final, perfect requirements document is ineffective, as prompt engineering is an iterative process of crafting, testing, and refining.
- Giving vague, open-ended commands like ”analyze these documents” works poorly because specificity in the prompt is key to getting relevant results.
- Trusting an LLM’s summary of a document without reading the original is risky, as inadequate prompts can lead to inaccurate responses that miss key points.
- Relying on an LLM for any multi-step or nuanced problem without providing examples is a bad strategy; combining Chain of Thought with few-shot prompting works better for complex tasks.
- Feeding a model a single, massive document is less effective than breaking it down into smaller, focused chunks to maintain simplicity.

5 What pre-post processing was useful?

Pre-processing (before prompting):

- Breaking down large documents into smaller, meaningful chunks (chunking) was the most useful step, especially for RAG systems.
- Creating a standardized template for use cases with sections like ”Goal/Value Proposition”, ”Actors”, ”Preconditions”, ”Main Flow”, and ”Alternative Flows”. By feeding the LLM a few examples of this filled-out template, it learned the desired structure and could reliably generate new use cases in the same format.
- Putting the main instruction at the end of the prompt, after all context and examples, often yields better results.

Post-processing (after getting a response):

- Having a human review and edit the LLM’s output is absolutely essential as part of the iterative prompt refinement workflow.
- Forcing the LLM to show its reasoning via Chain of Thought provides traceability, making it easier to debug errors.
- Passing one chat’s output as input to a new chat for the same or different LLM is helpful to summarize the output and identify duplicates, thus simplifying the manual review process.

6 Did you find any best/worst prompting strategies?

Best Strategies:

- The role-playing prompt (”You are an expert...”) consistently produces more focused and relevant answers by assigning the model a specific persona.
- Providing a few diverse, high-quality examples (few-shot prompting) is a powerful teaching tool that works better than just describing the task.
- Using constraints to tell the model what not to do (e.g., ”DO NOT use jargon”) is just as important as positive instructions, especially for safety or strict formatting.
- Being very specific about the output format (e.g., ”Summarize in three bullet points,” or ”return as JSON”) is critical for non-creative tasks.
- Telling the LLM to ”think step-by-step” (Chain of Thought) improves reasoning and makes the output easier to interpret and debug.

Worst Strategies:

- Vague, one-sentence prompts that lack a specific action verb (like ”Find the requirements”) consistently fail.
- Asking multiple different questions in the same prompt is confusing for the model and leads to poor answers.
- Not giving the LLM any context about the project’s goal guarantees a generic and useless response.