WolfCafe

CSE 510 Software Engineering: Project 1d1

Contributors: Janam Patel, Namit Patel, Pranshav Patel, Vivek Vanera

Question 1: What are the pain points in using LLMs?
Response:
When working with multiple LLMs, several challenges emerge:

1. Inconsistency across models: Identical prompts yield different styles, depths, or accuracies, which complicates reproducibility and forces repeated trial-and-error.
2. Latency and efficiency: Some models are fast but shallow in reasoning; others are slow but detailed. This mismatch makes reliable usage difficult for time-sensitive tasks.
3. Prompt sensitivity: Slight wording changes can trigger drastically different responses, increasing the overhead of prompt engineering.
4. Context window limits: As examples accumulate, models lose track of earlier inputs or compress them, leading to contradictions or repetition.
5. Hallucinations: Models confidently produce fabricated or inaccurate content, which requires manual inspection before use.
6. Cost factors: Stronger models often require subscription or API payments, creating financial barriers for extended or frequent use.
7. Opaque reasoning: Users can't see how the model derives outputs, making it hard to debug errors or trust results.
8. Domain adaptation limits: General-purpose models often fail in specialized domains without significant fine-tuning or supplemental data, reducing their reliability for niche applications.

Question 2: Any surprises? Eg different conclusions from LLMs?
Response: One of the biggest surprises was the variation in both quality and latency across models. For example, GPT and Gemini consistently produced higher-quality answers—while not flawless, they were detailed and closer to expectations. However, this came with noticeably longer response times, often around 14–15 seconds.

In contrast, models like Claude, Meta, and Groq delivered much faster outputs. Claude balanced speed with reasonable accuracy, but Meta and Groq showed weaker performance in terms of precision and reliability.

This trade-off between speed and accuracy was unexpected. Instead of a linear relationship where better models are both faster and more accurate, the results showed a spectrum: some excelled in reasoning but lagged in response time, while others prioritized quick turnaround at the expense of correctness.

Another surprise was that models often reached different conclusions for the same question. Even with identical prompts, the interpretations and focus varied, revealing how each model is tuned or trained differently. This reinforced the idea that results are not interchangeable, and careful selection is necessary depending on the task.

Question 3: What worked best?

Response: Few-shot prompting proved the most effective technique. By providing multiple examples before asking for the final answer, the models were better able to infer the required format, reasoning style, and level of detail. This reduced misinterpretation and improved consistency across outputs.

Among the models, GPT, Claude, and Gemini performed best overall. GPT was strongest in generating detailed, logically organized responses with fewer hallucinations. Claude stood out for speed and clarity, producing coherent answers quickly while still maintaining good accuracy. Gemini delivered balanced results, often closer to GPT in depth while being more responsive in certain tasks.

The combination of these models with few-shot prompting allowed for both quality and efficiency. It showed that guiding the models with structured examples was more reliable than relying on zero-shot or single-shot prompts, and that choosing high-performing models amplified these gains.

Question 4: What worked worst?
Response: Zero-shot prompting consistently produced the weakest results. Without examples to anchor the model's reasoning or formatting, the outputs were often random, unfocused, or misaligned with the task. This unpredictability made zero-shot approaches unreliable for anything beyond very simple queries.

In terms of models, Groq and Meta performed worst among those tested. While they responded quickly, the quality of answers was poor—often incomplete, inaccurate, or lacking depth. Compared to GPT, Claude, and Gemini, these models failed to capture nuance or provide trustworthy responses, making them unsuitable for tasks requiring precision or reasoning.

This combination—zero-shot prompting with weaker models—highlighted the limitations of relying on speed alone. It reinforced the need for both careful prompting strategies and selecting higher-quality LLMs to achieve usable outputs.

Question 5: What pre-post processing was useful for structuring the import prompts, then summarizing the output?
Response: On the preprocessing side, providing structured examples was essential. Few-shot prompting helped the models infer both the reasoning style and the expected format, while meta prompting—explicitly instructing the model how to think or organize its response—reduced hallucinations and improved alignment with requirements. Clearly specifying output structure (such as bullet points, headings, or step-by-step reasoning) also made the responses more consistent. In cases where the output didn't match expectations, prompts had to be restructured and clarified, showing that iterative refinement was part of the process.

For post-processing, summarization was often necessary. Long outputs had to be condensed into concise points to extract only the relevant information. This step ensured the results were usable, especially when models produced verbose or tangential content. Post-processing also involved manually validating answers to filter out inaccuracies and reformatting them into a standardized structure for easier comparison across different LLMs.

Question 6: Did you find any best/worst prompting strategies? That changes were made to the MVP to appease stakeholders?
Response: The most effective prompting strategies were few-shot prompting, meta prompting, and chain prompting. Few-shot prompting provided concrete examples that guided the models toward consistent reasoning and formatting. Meta prompting—where explicit instructions were given on how to structure or approach the response—helped reduce hallucinations and improved alignment. Chain prompting, where the model was asked to reason step by step, increased accuracy and transparency in complex tasks by making the intermediate logic visible. Together, these strategies consistently delivered the most reliable outputs.

The least effective strategy was zero-shot prompting. Without examples or explicit guidance, outputs were often random, inconsistent, or shallow. This lack of structure made zero-shot prompting unsuitable for anything beyond simple queries.