Table of Contents

1. Prompt Engineering Strategy

- How did you design your prompts for the LLM?
  - The prompts were designed with two main goals in mind.
  - As I have grown to have a greater and deeper understanding of the LLMs, I have learned that too much in the prompt causes issues. When there is too much guidance it can start to hallucinate.
  - The best prompts were short, concise and well formatted.
- Did you iterate or change prompts over time?
  - It turns out that even the style of the prompt influences the style of the tutor.
  - Overtime the prompts grew to be formatted in markdown, which seemed to work the best, and were stripped down to the shortest possible prompt.
- What worked well? What failed?
  - What didn't work well is giving the tutor a million rules or guidelines to follow.
  - In fact the more that I tried to enforce or "tell" it what to do, it would hallucinate more.
- How did you make prompts "educational" instead of generic?
  - Really the best way that the prompts were educational was telling the tutor in a straightforward and direct way:
    - You are a Tutor for Utah Valley University
  - More specifically I gave the particular students courses along with those instructions, like:
    - You are an expert in [Student's Course] and that course is about [Course Catalog Description]

---

2. LLM Integration Decisions

- Which LLM or APIs did you use? Did you try different LLMs
  - OpenAI
    - We used OpenAI due to the fact that we had a grant for several thousand dollars of credits.
  - Anthropic
    - We switched to the Anthropic API, when one day we had issues accessing our OpenAI credits. We were able to quickly switch, due to the fact that our service was broken.
- Why did you choose it?
  - OpenAI: because we had the grant money.
  - Anthropic: As mentioned above, but I did like Anthropic because it seemed to be more emotionally intelligent, but more expensive.
- How did you manage API limits or costs?
  - We used use gpt-4o-mini when possible, given that it cost about an order of magnitude or two less than the gpt-4o API.
  - We also recorded the total tokens used on every message in our analytics service.
- How did you handle rate limits or token usage?
  - Automatically, the code would clear context (the chat conversation), not saving messages by default. This saved a lot of money in credits and tokens.

---

3. Handling Hallucinations and Errors

- Did you see the LLM give wrong or hallucinated answers?
  - Yes, usually after a conversation had gone on for some time, The AI would forget the initial prompt and start making up assignments or grades.
  - This was really bad, and we used tool calls to have more consistent and correct output.
- How did you test outputs for correctness?
  - Given that its nearly impossible to test every message the AI can send, I as the developer spent many hours testing the output of the LLM before making changes.
- Did you filter or post-process outputs?
  - Yes, we used Regular Expressions and some special tricks to format the output. This was specifically useful for math.

    - We also moderated every chat message to check for violence, sexual messages, or self harm. Anytime those messages were sent, they were logged and blocked.
    - Note in the case of self-harm, we sent the UVU student health services link.
- What would you improve here next time?
    - I would have more tool calls and more post processing, but that would also use more credits.

---

4. Adapting AI for Educational Use

- How did you try to make AI responses educationally useful?
    - We used the activity stream from the Canvas LMS so that the tutor was always up to date with the student's performance in courses.
    - We also used the 2024-2025 UVU course catalog to provide the AI with more context about each course as a whole.
- Did you add scaffolding, hints, or explanations? Any thing else?
    - The most similar to this would have been the Course catalog as mention above.
- How did you ensure clarity and appropriateness?
    - Along with moderation, we just add that the LLM should be "Excited and provide concise and useful responses".

---

5. AI Performance Metrics

- Typical latency / response time?
    - We never had any issues with response time.
    - The only issue was that occasionally if the app was not being used, it would be slow to start up at first. (This was to save hosting costs)
- Cost per API call or session?
    - Approximately one message and response was between 150 - 500 tokens plus the system message, which was also between 150-300 tokens.
    - Although, the cost per session is somewhat hard to calculate, This is just because you have to take the cost of each token $t \approx \frac{1{,}000{,}000}{\$2.00}$ and then multiply that by all the previous messages **including** the one you are sending now. So it is somewhat exponential in cost as the length of a chat increases.

$$S \cdot t + \sum_{i=1}^{n} (U + A) \cdot \left(n(i-1) - \frac{i(i+1)}{2} \right) \cdot t$$

- Average tokens used per prompt/response?

See above.

- Any notable error rates or failures?

  Not necessarily

---

6. Lessons and Recommendations

- What would you recommend to another team trying to integrate LLMs for learning?
  - First of all good luck, its tough to get the AI to say what you want. It has taken me about 2 years.
  - Secondly, tool calls are the future of truly useful LLM applications.
- What trade-offs would you highlight?
  - The trade off between hallucinations vs novelty or fun conversations is truly tough.
- What would you do differently next time?
  - I would just have more tool calls and rely on them much much more.