Draft Proposal: Add AI to Chat Activity

Sujay R | github.com/sujay1844

First language: Telugu (But schooling was in English)
Location and Timezone: Bengaluru, India. GMT+5:30

About Me

I'm a passionate 2nd year Btech student specializing in AI/ML from PES University, Bengaluru. I have varied experience with Python, including developing web scraping scripts, backend APIs, and machine learning models including LLMs. I am very well versed in web development as well. But here's the relevant LLM experience:-

- Building a high school essay grading and feedback system that utilizes an LLM to provide substantive comments on student essays.
- I've also developed a chat interface that allows users to query textbooks and reference materials using LLM as the interface (currently used at my university).
- I created an adaptive question generator that uses an LLM to produce practice questions tailored to a student's specific needs in terms of difficulty, topic, and timing.
- Currently, I'm working on a research paper on question difficulty estimation using generative AI.
- I'm also an ML mentor in the IEEE CS chapter of my university.

Although my direct experience with Sugar activities is currently limited, I am genuinely impressed by the platform's educational focus.
Please note, many of my projects are private.

Requirements

Our AI chatbot focuses on the following features:-

- **Safety** is paramount. Ensuring the content is free of inappropriate language, violence and biases.
- **Simplicity** is key. Use short sentences, clear vocabulary and focus on basic concepts.
- **Engagement** makes the bot fun and educational for kids.

The plan

To implement this, we have two options: prompt engineering and fine-tuning. Prompt engineering is giving instructions in the prompt such as "You're talking to a child", "Don't use

profane words", etc. This can backfire since the foundation models weren't developed with child safety in mind. For example, Gemini can be too safe. Apart from that,

- We could have unintended biases, complex language (LLM will generate even if you instruct it not to generate). We can't capture the child's chain of thought.
- Even with instructions to avoid negativity, factual topics (e.g., war, illness) might still be presented in a way that's alarming to a child without proper context.
- Prompt engineering often leads to a question-and-answer format, which can be passive learning. Fine-tuning can allow for more engaging elements like storytelling or games.

Since we are fundamentally changing the LLM's behavior, fine-tuning is better.

- We can tailor the responses to be age appropriate.
- The model will avoid generating irrelevant or tangential information. This is especially important for younger children who may struggle to discern truth from fiction.
- The answer will be of the correct length (have you noticed how ChatGPT gives a verbose 300 word essay for even simple questions, that won't happen here).
- We can incorporate storytelling elements, rhymes, or specific characters that resonate with children.
- Fine tuning will make the model resilient towards prompt injections.

The approach

Any reasonably capable foundation model will work for us. Here's OpenAI's fine tuning pricing. If we want to use more open LLMs, we could fine tune and deploy on RunPod serverless (pricing). The choice of what LLM to use can be done later after analyzing the costs. Fine tuning won't be much of an expense. **However**, if fine-tuning is not feasible, please let me know, I'm open to that as well, I'll research about ways to make prompt engineering more resilient.

We will fine tune on the following datasets, each serving a different purpose:-

- Children's Stories Text Corpus: This will make the LLM explain in a way that children would usually read. And would prevent profanity to a large extent.
- Simple Wikipedia: This will make the LLM use simple language, short sentences and prevent it from talking about very advanced concepts (not possible with just prompting).

Then, prompt engineering:-

- Sight Word Lists: To further make the LLM use words suited for children.
- List of Bad Words: To further prevent the LLM from using inappropriate language.
- Prompt engineering to round things off

And after all this, if we have time and resources, we can further fine-tune using Reinforcement learning from human feedback (**RLHF**) as final touch to ensure the LLM fits our needs. We could also personalize the interactions for different users.

And of course, we integrate this chatbot into the Chat Activity, test and then deploy it. We'll figure out the cheapest way to deploy the model, use techniques including quantisation, etc to optimize cost and performance, documentation, testing, CI/CD, etc.

Your Suggestions

This proposal is not final. I've used plenty of informal language in this draft. The format is not correct. I know a timeline is required for the proposal, but I'm not including it here. Ibiam and the others from the Sugar Labs team, please tell me what parts of the described approach you want me to focus on, I'll draft a timeline after that. And I haven't gone into much detail about the software engineering side of the project, I'm sure I am capable of figuring that out, please let me know if you'd like more details on that.

Looking forward to working with the Sugar Labs team.
I know I haven't had much interaction with the community, but the project is very interesting to me now. I'm excited to make my contribution to the cause. I would've done it regardless of GSoC. To be clear, Gsoc stipend is life-changing for me (it's half my student debt), so I'd love to work on this in Gsoc. Let's not just write a prompt and call it a day. I think if proper effort is put into making the bot fun and friendly, the kids are gonna love it. I know our time zones are bad, but I'll wake up early, we'll make it work.