GSoC 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: exclusively prompt engineering or fine-tuning and prompt engineering. 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. The choice of what LLM to use can be done later after analyzing the costs. For now, let's consider Mistral 7B. Fine tuning won't be much of an expense.

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.

Implementation

For the LLM and fine-tuning process, we'll leverage frameworks such as HuggingFace's Transformers and Parameter Efficient Fine-Tuning library (PEFT).

The user interface will be designed to seamlessly blend the AI chatbot into the existing conversational flow, allowing users to engage with the chatbot as they would with other participants. We'll implement appropriate visual cues and prompts to indicate when the chatbot is responding and to differentiate its responses from human messages.

Evaluation

While the best method for evaluation is from human feedback, I do understand that it is very hard. We'll maintain a separate test set comprising conversations and prompts. This test set will be used to identify any issues or biases. Creation of this dataset is a challenge that I'm confident we'll figure out eventually.

We'll collaborate with educators and the Sugar Labs community to establish qualitative and quantitative metrics for evaluating the chatbot's suitability for the target group. This will include factors such as vocabulary complexity, sentence length, topic appropriateness, etc.

For RLHF, testing must be conducted with children of different age groups to gather feedback. We can gather this feedback exclusively (as done by OpenAI when training ChatGPT from GPT-3) and/or when the bot is being used (as done by OpenAI when people use the web version)

Challenges

Extracting quality data for fine-tuning will be the primary challenge. Making a test set with diversity will be another hurdle. Maintaining coherence throughout extended dialogues may also pose challenges.
Obtaining quality data can only be done by data engineering and hard work. For the coherence issues we might face, we could investigate context tracking and dialog management techniques.

Timeline

Week 1-4

In the month of May, I can't work much since I have internals and then immediately end-semester examinations. But I won't go completely offline.

Week 5-6

It's vacation. Since I was inactive for a long time and now I'm working full time, I'll speed things up. I'll explore the chat activity codebase. Study the etiquette of contributions at Sugar. Understand who are the users of chat? How do they use it? Any specific features they've requested? Integrate a base LLM into the chat to get started.

Week 7

Start collecting the required datasets by downloading, web scraping, etc. and also be on the lookout for more data.

Week 8

Parse each corpus into a format that can be fed into the LLM. Fine tune the LLM with the parsed data.

Week 9

Since fine-tuning has a trial-and-error aspect, I'll set aside another week. And I'll also add the prompts to the pipeline.

Week 10

Add thumbs up and down buttons for RLHF. Start testing the bot with the kids, Sugar team, and/or whoever seems appropriate, the more the merrier. Collect data required for RLHF from the testing.

Week 11

Fine tune the model on the obtained RLHF data. Make sure the bot is up to the mark of quality that's expected. If not, more RLHF.

Week 12-13

Work on optimizing the performance if required, using techniques like more quantisation, etc. Figure out how much computation we actually need to run the bot and adjust hosting accordingly. Handling edge cases and writing documentation