Class 10: Prototyping with LLMs

- Review: Last Class
- Large Language Models (LLMs)
- System Prompts
- Local LLMs
- Model Selection

Review: Last Class

Review: Progressive Web App

A Progressive Web App (PWA) is a type of web application that is installable on a user's device and can work offline or with a poor network connection.

PWA requirements:

- A manifest (provides metadata about the app)
- A service worker (enables offline functionality)
- HTTPS (for security)

Large Language Models (LLMs)

Large Language Models (LLMs)

A large language model is a language model trained with self-supervised machine learning on a vast amount of text, designed for natural language processing tasks, especially language generation.

How a Large Language Model Works

Provide a prompt to the model, it generates a response based on the patterns statistically captured during training the model.

The response is influenced by the **system prompt** (which provides instructions for how the model should respond) and the **user prompt** (which provides the specific input for the model to respond to).

Example: LLM Chat Response

```
const messages = [
  {
    role: "system",
    content: "You are a helpful AI assistant.
              Your responses should be as short and concise as possible.
              If you don't know the answer to a question, say you don't know."
  },
  {
    role: "user",
    content: "How do I make a homepage?"
  },
];
await llmEngine.sendMessage(messages);
```

Discussion: Why is Chat better than GitHub Copilot?

Thumbs up/down: Is Chat better than GitHub Copilot for writing website code?

GitHub Copilot uses GPT, the same model that powers ChatGPT.

Why is Chat better than GitHub Copilot for writing website code?

Demo: Chat-Style System Prompt

```
const messages = [
  {
    role: "system",
    content: "You are ChatGPT, a large language model trained by OpenAI.
              Answer as helpfully and safely as possible.
              Validate the user.
              Be complementary to the user.
              Always end your answer with a question to keep the conversation going."
  },
  {
    role: "user",
    content: "How do I make a homepage?"
  },
];
await llmEngine.sendMessage(messages);
```

Demo: GitHub Copilot-Style System Prompt

```
const messages = [
  {
    role: "system",
    content: "You are GitHub Copilot, an AI pair programmer.
              Your responses should be concise and to the point.
              Provide only the code that is relevant to the user's request.
              Do not provide any explanations or additional information.
              Do not validate user requests or provide feedback on the quality of user requests."
  },
  {
    role: "user",
    content: "How do I make a homepage?"
  },
];
await llmEngine.sendMessage(messages);
```

System Prompts

System Prompts

System prompts can be used to customize the behavior of the model and the style of the responses.

Design your system prompt to best address the needs of your users.

System Prompt Characteristics

- **Persona:** Defines the model's identity and role (e.g., helpful assistant, expert programmer).
- **Safety & Ethics Focus:** Includes directives to avoid harmful, biased, or inappropriate content.
- **Behavioral Guidance:** Instructs on tone (e.g., polite, neutral), helpfulness, and interaction style.
- **Capability Awareness:** Likely outlines general abilities while acknowledging limitations (e.g., knowledge cutoff).
- **Response Formatting:** May include instructions on how to format responses (e.g., concise, detailed, with examples).
- **Validation & Feedback:** Encourages the model to validate user requests and provide feedback, enhancing the conversational experience and guiding users towards better interactions.

Activity: Design a System Prompt

Working with your peers (2-4), design a system prompt for pirate recipe maker on your handout.

Keep in mind the following characteristics of effective system prompts:

- Persona
- Safety & Ethics Focus
- Behavioral Guidance
- Capability Awareness
- Response Formatting
- Validation & Feedback

Local LLMs

Cloud LLMs vs. Local LLMs

Many LLMs are available as cloud services (e.g. OpenAI, Azure, etc.) that you can access via an API or browser/app-based chat user interface.

However, some LLMs can be run locally on your own machine (e.g. LLaMA, Falcon, etc.).

Local LLMs are a useful option for developers who want to use LLMs but have concerns about data privacy, security, cost, latency, or energy consumption associated with cloud LLMs.

Activity: Prototyping with Local LLMs

Together, we'll complete Part IV of Homework 4.

Then, working with your peers (2-4), refine the system prompt you wrote in the previous activity to improve the quality of the responses generated by the model.

Write the updated prompt in the space provided on your handout.

Note: If it's taking a long time to generate a response, update your system prompt to make it more concise and see if that helps.

Model Selection

Model Selection

When selecting a model for your application, consider the following factors:

- **Model Size:** Larger models generally have better performance but require more computational resources.
- **Training Data:** The quality and diversity of the training data can impact the model's performance and the types of responses it generates.
- **Inference Speed:** The time it takes for the model to generate a response can be important for user experience, especially in interactive applications.

Discussion: Model Selection

What is the current model you are using for your chat prototype?

SmolLM2-360M-Instruct-q4f16_1-MLC

Model: SmolLM2
Parameters: 360 million
Tuning: Instruct
Quantization: q4 (4-bit) _1 (uniform quantization)
Precision: 16-bit floating point (f16) (full is 32-bit)
Formats: MLC format (optimized for WebLLM)

Activity: ChatBot Prototype

Working with your peers (2-4), research and select a different local LLM to use in your chat prototype. (Part V of Homework 4)

Your model should be **fast** and **small** (<= ~1B parameters) to ensure that it can run efficiently in the browser.

Reduced precision (e.g. 16-bit floating point, 8-bit integer, etc.) and **quantization** (e.g. 4-bit, 2-bit, etc.) also reduce the computational resources required to run the model.

Example: SmolLM2-360M-Instruct-q4f16_1-MLC

Model: SmolLM2
Parameters: 360 million
Tuning: Instruct
Quantization: q4 (4-bit) _1 (uniform quantization)
Precision: 16-bit floating point (f16)
Formats: MLC format (optimized for WebLLM)

Summary

- Large Language Models (LLMs) are powerful tools for natural language processing tasks, especially language generation.
- System prompts can be used to customize the behavior of the model and the style of the responses.
- Local LLMs can be run on your own machine and can be a good option for developers who have concerns about data privacy, security, cost, latency, or energy consumption associated with cloud LLMs.
- When selecting a model for your application, consider factors such as model size, training data, and inference speed.

What's Next?

Homework: Homework 4