

Class 12: Local LLMs for Prototyping

1. **Review:** System Prompt Engineering
2. **Discussion:** LLMs and Energy Consumption
3. **Activity:** LLM Library / Package
4. **Studio:** System Prompt Maker

Copyright 2026, Kyle J. Harms

Review: 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).

Review: System & User Prompts

```
[  
  {  
    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?"  
  },  
]
```

Review: Prompt Engineering

1. Keep the prompt **short** at first, then iterate.
2. Instructions to the model should go at the **beginning** or **end** of the prompt.
3. Be **specific** and **descriptive**.
4. Focus on **what to do**, not **what not to do**.
5. Provide **examples** of the desired output.

Review: Zero Shot

```
[
  {
    role: "system",
    content: `Given a review, assign a rating from 1 to 5,
              where 1 is the worst and 5 is the best. Output the rating as JSON.`
  },
  {
    role: "user",
    content: `Quite overpriced. Paid $33 for a large pizza after tax.
              Wow. You'd think it was made of gold. The service was not great and
              then after there was an extra charge for using a credit card.
              The food was subpar. Recommendations: become great at a few
              things and make the menu smaller.`
  }
]
```

Review: One Shot

```
[
  {
    role: "system",
    content: `Given a review, assign a rating from 1 to 5,
              where 1 is the worst and 5 is the best. Output the rating as only JSON.`
  },
  {
    role: "user",
    content: `The food was amazing and the service was excellent!`
  },
  {
    role: "assistant",
    content: `{"rating": 5}`
  }
]
```

Review: Few Shot

```
[
  {
    role: "system",
    content: `Given a review, assign a rating from 1 to 5,
              where 1 is the worst and 5 is the best. Output the rating as only JSON.`
  },
  { role: "user",
    content: `The food was amazing and the service was excellent!` },
  { role: "assistant",
    content: `{"rating": 5}` },
  { role: "user",
    content: `The food was okay, but the service was slow.` },
  { role: "assistant",
    content: `{"rating": 3}` }
]
```

Review: Chain of Thought (CoT)

```
[
  {
    role: "system",
    content: `Do this task step-by-step:
      1. Read the review carefully.
      2. Identify the key points about the food, service, and overall experience.
      3. Assign a positive, neutral, or negative sentiment to each key point.
      4. Based on the overall sentiment, assign a rating from 1 to 5
      where 1 is the worst and 5 is the best.
      5. Output the rating as only JSON. Example: {"key_points": [...], "rating": 4}`
  }
]
```

LLM and Energy Consumption

Discussion:

**“We did the math on AI’s energy footprint.
Here’s the story you haven’t heard.”**

Model Selection and Energy Consumption

- Larger models generally consume more energy than smaller models (< 8B parameters).
- Models with higher precision (e.g. 32-bit floating point) generally consume more energy than models with reduced precision (e.g. 16-bit floating point, 8-bit integer, etc.).
- Quantization can also help reduce energy consumption by reducing the number of bits required to represent the model's parameters. (e.g. 4-bit quantization, 8-bit quantization, etc.)

Cloud LLMs vs. Local LLMs: Energy Consumption

- Cloud LLMs, like ChatGPT, typically consume significantly more energy.
- However, the energy consumption is not known because the models and infrastructure are proprietary.
- Local LLMs can be more energy efficient, especially if you select a smaller model with reduced precision and/or quantization.

Discussion: If a task can be accomplished with either a cloud LLM or a local LLM, which would you choose and why?

(Cloud: microwave > 8 seconds, **Local:** microwave < 10th of a second)

Discussion: LLMs and App Prototyping

How do you pick a model?

- Parameters (1B vs 1T)
- Precision (f32 vs f16)
- Quantization (q0 vs q4)

If your user's task doesn't require a high level of accuracy, consider using the **smallest** available model and/or model with reduced precision and/or quantization to reduce energy consumption.

WebLLM Models

Model	Parameters	Quantization	Precision
TinyLlama	1.1B	q4	f16/32
LLaMA	70B, 8B, 7B, 1B	q3/4	f16/32
SmolLM	135M, 360M, 1.7B	q0/4	f16/32
Mistral	7B	q4	f16/32
Qwen	0.5B, 0.6B, 1.5B, 1.7B, 3B, 4B, 7B, 8B	q0/4	f16/32
Phi	3.8B	q4	f16/32
Gemma	2B, 9B	q4	f16/32

Prototyping with Local LLMs

Prototyping Methods

When prototyping, you may need to implement functionality that is complex or time-consuming to implement on your own.

Common methods:

- Generative AI (i.e. Claude, ChatGPT, GitHub Copilot, etc.)
- Programming libraries / packages
- Online code snippets (e.g. StackOverflow)
- Tutorials (e.g. YouTube, blogs, etc.)
- Write it yourself from scratch

Prototyping Functionality Pros & Cons

Method	Pros	Cons
Generative AI	Fast and works for a wide variety of tasks.	Accuracy is suspect. It may take longer to validate the code is correct.
Programming Libraries / Packages	Often saves time and effort for common tasks.	Generally, popular libraries are often accurate.
Online Code Snippets, Tutorials	Solution may solve your exact problem.	May be time-consuming to get it working with your problem.
Write it yourself from scratch	You have full control over the code.	Time-consuming.

Libraries / Packages

Programming Library / Package

A programming library (or package) is a collection of pre-written code that provides specific functionality or features that can be integrated into your own projects.

Libraries / packages are designed to save developers time and effort by providing reusable code for common tasks.

Activity: LLM Library / Package

Working with a **peer**, identify a library / package to use an LLM in a Vue.js prototype.

Activity: WebLLM Walkthrough

1. `npm install @mlc-ai/web-llm`

2. `import { llmEngine, status, progress, Status } from
"./services/localLlm";`

3. `try {
 await llmEngine.generate(messages.value, {
 onResponse: (text) => (response.value = text),
 });
} catch (error) {
 console.error("Failed to generate a response:", error);
 hasResponseError.value = true;
}`

4. `llmEngine.stop();`

Gotcha: Local LLMs – Text

Local LLMs are best suited for **text-based** tasks.

While local image generation (and other non-text) models exists, the computational resources required are likely out of scope for this class.

Studio: System Prompt Maker

Goal: Practice your prompt engineering skills:

- Use GitHub Copilot to modify the Vue.js code
- Custom system prompts to generate a response from a local LLM

Implementation:

- Persona selector (at least 3 personas)
- Output format selector (at least 3 formats)
- (more to come)

Tip: JavaScript `

The ` operator is used to create a **template literal** in JavaScript, which allows for multi-line strings and string interpolation.

Example:

```
const name = "Alice";  
const prompt = `You are a helpful assistant. Your name is ${assistantName}.`;
```

What's Next

Homework 4 Release today