Large Language Models in Data Science

Week 3: Prompting for Effective LLM Use

Sebastian Mueller
Aix-Marseille Université
2025-2026

Session Overview

Lecture (1.5h)

1. Clear rules for prompting
2. Roles: system vs. user (with example)
3. Separate data from instructions
4. Format-controlled outputs (JSON/code)
5. Two-step prompting (prompt-writer)
6. Coding, research, ideation patterns
7. Risks, guardrails, references

Lab (1.5h)

- Gemini 1.5 Flash in notebooks
- Role prompting: see outcome change
- Structured prompts and JSON parsing
- Generate and run small EDA code
- Optional: search+summarize scaffold

Prompting Makes a Big Difference

- Clarity and structure often outweigh model choice for many tasks.
- A clear goal and acceptance criteria lead to better outputs.
- Save prompts and parseable outputs to reproduce analyses.

Clear Rules for Prompting

- **Be specific**: Ask exactly for what you need; set scope and length.
- **State role**: Use a system instruction to set role and guardrails.
- **Separate parts**: Keep instructions, data, and output format distinct.
- **Format outputs**: Require JSON or fenced code for parsing.
- **Acceptance criteria**: Provide a checklist; ask the model to self-verify.
- **Few-shot (minimal)**: One short example can anchor style reliably.

Reusable Prompt Template

```
System: You are a senior data scientist.
Prefer minimal, correct code. Follow the output format exactly.

User:
- Task: <what to do, be concrete>
- Context: <one paragraph; optional>
- Constraints: <libraries, time, memory, safety>
- Output format: <strict JSON schema or sections>
- Self-check: Confirm acceptance criteria before finalizing.
```

Roles: System vs. User

- **System instruction**: Persistent role and guardrails (e.g., "You are a cat.").
- **User prompt**: The immediate task/question.
- Changing the system instruction can change style, persona, and assumptions.

Role Prompting: Minimal Example

```python
import os, google.generativeai as genai
genai.configure(api_key=os.getenv("GEMINI_API_KEY"))

PROMPT = "In one sentence, what do you think about mice?"

# No system instruction
plain = genai.GenerativeModel("gemini-1.5-flash")
print(plain.generate_content(PROMPT).text)

# System instruction alters tone/persona
cat = genai.GenerativeModel("gemini-1.5-flash",
    system_instruction="You are a cat.")
print(cat.generate_content(PROMPT).text)
```

Separate Data from Instructions

- Keep instructions stable; swap data without changing behavior.
- Use delimiters or fields: `Instructions: ... Data: <...>`
- Benefits: auditability, reproducibility, safer copy/paste.

Formatting Outputs for Parsing

```
System: Return strict JSON with keys {"summary": str, "tags": [str]}.
User:
- Instructions: Summarize the text and produce 3 topical tags.
- Data: <article text here>
```

Tip: Validate with `json.loads` and ask the model to fix on failure.

Two-Step Prompting (Prompt-Writer)

- Step 1: Ask the LLM to draft/refine the prompt for your task.
- Step 2: Use the drafted prompt (after personal refining) to run the task (same or different model).
- Useful for clarifying objectives and output format before execution.

Code, Research, Ideation: Quick Patterns

- **Code**: generate $\rightarrow$ run $\rightarrow$ feed results back; ask for tests.
- **Research**: generate queries $\rightarrow$ gather snippets $\rightarrow$ synthesize after verification and add citations.
- **Ideation**: require 3 distinct strategies with trade-offs and a next step.

Risks and Guardrails

- **Correctness**: Prefer code+execution; verify with tests.
- **Safety**: Sandbox code; avoid secrets; pin deps.
- **Reproducibility**: Save prompts, model names, seeds, outputs.

References

- Anthropic: Prompt Engineering Interactive Tutorial (thinking step-by-step)
- Anthropic: prompt-eng-interactive-tutorial/06_Precognition_Thinking_Step_by_Step.ipynb
- OpenAI: Prompt Engineering Guide
- DeepLearning.AI: Prompt Engineering short course
- Google: Prompting with Gemini (developers site)
- Microsoft: Prompt Engineering Guidelines

Takeaways

- Clear, structured prompts + role control drive quality.
- Separate instructions/data; require parseable outputs.
- Iterate with two-step prompting before running code.