# Introduction to Data Science

dr Maciej Świtała
Ewa Weychert

Class 8: AI and prompt engineering

e.weychert@uw.edu.pl

# What is AI?

- **Artificial Intelligence (AI)**: systems that perform tasks that normally require human intelligence.
- Examples:
  - Understanding and generating text.
  - Recognizing images, audio, and video.
  - Making predictions and recommendations.
- Today we focus on **Large Language Models (LLMs)** like ChatGPT.

# Claim: “There is no AI, only LLMs”

- You sometimes hear: There is no AI, only LLMs.
- This statement feels true because:
  - Most visible AI apps today = chatbots based on LLMs.
  - People interact daily with ChatGPT, Claude, Gemini, etc.
- But from a scientific perspective, this is **not accurate**.
- Let’s unpack what **AI** and **LLMs** actually are.

# What Is AI? (Broad Definition)

- **Artificial Intelligence (AI)**:
  - Any technique that makes computers perform tasks
  - which, if done by humans, would require intelligence.
- AI includes many subfields:
  - Classical search, planning, logic.
  - Expert systems, rule-based systems.
  - Machine learning, deep learning.
  - Computer vision, speech recognition, robotics.
  - **Large language models (LLMs)**.
- **Conclusion:** LLMs are *one branch* of AI, not the whole tree.

# What Is an LLM?

- **Large Language Model (LLM)**:
  - A model trained to predict the **next token**
  - in a sequence of text, given the previous tokens.
- Trained on massive text corpora:
  - Books, websites, code, documents, etc.
- Core capability:
  - Generate and transform text (and code) by pattern matching.
- They do **not** have:
  - Human-like understanding or consciousness.
  - Persistent memory (unless wrapped in a larger system).

# Why It Feels Like "AI = LLMs" Today

- Most popular AI tools are LLM-based:
  - ChatGPT, Claude, Gemini, Copilot, etc.
- They are:
  - General-purpose (text, code, reasoning, explanation).
  - Accessible through a simple chat interface.
  - Capable of talking like a human.
- For many non-technical users:
  - AI = the chatbot I talk to.
- This drives the slogan:

> There is no AI, only LLMs.

# Why That Claim Is Misleading

- It ignores the rest of AI:
  - Vision, robotics, control, planning, classical ML, etc.
- It confuses:
  - **Models** (LLMs)
  - with full **systems** (AI agents, copilots, apps)
- It encourages **anthropomorphism**:
  - Treating a text predictor as a real intelligence.
- Better statement:

**Most visible AI apps today are built around LLMs.**

# LLMs as Components, Not Whole Systems

- Modern AI assistants often combine:
  - An LLM (language + reasoning engine).
  - Tools (search, code execution, databases, APIs).
  - Memory (storing user context, documents).
  - Orchestration / planning logic.
- The LLM is:
  - The **interface** (natural language).
  - One part of a larger **AI system**.
- Saying only LLMs hides all other components.

# LLMs Are Powerful, But Not AGI

- LLMs can:
  - Simulate reasoning.
  - Explain concepts.
  - Generate complex artifacts (code, essays, plans).
- However, they:
  - Do not have goals or desires.
  - Do not understand the physical world.
  - Do not build internal causal models like humans.
- Important for students:
  - Respect the power of LLMs.
  - But avoid calling them **full intelligence**.

# A More Accurate Way to Say It

- Instead of:

There is no AI, only LLMs.

- Use:

Most of what people call AI today are systems built around large language models.

- Or:

Modern AI applications often use LLMs as a core component, but AI as a field is much broader.

# Key Takeaways for Students

- **AI** is a broad field; **LLMs** are one powerful technique within it.
- LLMs:
  - Are next-token predictors trained on huge text corpora.
  - Do not equal general intelligence.
- Most popular AI apps today are LLM-centric, which fuels the confusion.
- As ML/DS practitioners:
  - Understand LLMs deeply.
  - But also know the wider AI toolbox beyond LLMs.

# Why Prompts Matter

- The model **does not read your mind**—it only sees your text.
- Small changes in wording can lead to very different outputs.
- Good prompting turns a generic model into:
  - A tutor.
  - A software architect.
  - A marketing expert.
  - A research assistant.
- Prompting is a **skill** you can improve with practice.

# How LLMs Work (Intuition)

- LLMs are trained to **predict the next token** (piece of text).
- They do not have:
  - Human-like understanding.
  - Persistent memory of you (unless explicitly provided).
- They **simulate** patterns of reasoning based on training data.
- Your prompt defines the **context** and **goal**.

# Prompting

# What is a Prompt?

- A **prompt** is any input you give the model:
  - Instructions (Summarize this article).
  - Data (a paragraph, table, code snippet).
  - Examples (good / bad outputs).
- Prompts can be:
  - **Unstructured**: natural language instructions.
  - **Structured**: clear sections, headings, bullet points, formats.

# Anatomy of a Good Prompt

A useful mental checklist:

- **Role** — Who is the AI?
- **Task** — What should it do?
- **Context** — Background, constraints, domain.
- **Output Format** — Bullet list, table, JSON, etc.
- **Tone & Audience** — Level of detail, style, jargon.

**Formula:**
*You are [role]. Your task is to [task]. Consider this context: [context]. Produce output in [format] for [audience].*

# Clarity & Specificity

- Avoid vague prompts:
  - Explain AI.
- Add constraints and details:
  - Length (150 words).
  - Audience (non-technical managers).
  - Purpose (so they can decide whether to invest).

## Example

**Instead of:**
Explain AI.

**Use:**
In about 150 words, explain what AI is to non-technical managers so they can decide whether to invest in it. Use simple language and a concrete example.

# Bad vs Good Prompts

**Weak prompt**

Write about dogs.

Problems:

- No audience.
- No purpose.
- No length or style.

**Improved prompt**

Write a 150-word, beginner-friendly explanation of how guide dogs are trained. Use simple language and concrete examples so a 12-year-old can understand.

Improvements:

- Clear topic.
- Defined audience.
- Length + style.

# Using Constraints Effectively

- **Length**: word count, paragraphs, bullet points.
- **Tone**: formal, friendly, humorous, neutral.
- **Perspective**: first person, third person, etc.
- **Structure**: sections, headings, numbered steps.
- **Language**: specify language, level (A2, B2, etc.).

## Example

Create a one-page executive summary (about 300 words) in bullet points and sub-bullets that a busy CEO can read in 2 minutes.

# Examples: Few-Shot Prompting

- Provide **examples** of what you want.
- The model learns your pattern within the conversation.

## Example

**Prompt:**

You are a writing assistant. Rewrite sentences to be clearer and more concise.

Example 1:
Input: Due to the fact that we were late, we missed the train.
Output: Because we were late, we missed the train.

Now improve this sentence:
Input: In light of recent events, it has become increasingly necessary to improve our processes.

# Bad vs Good Prompts: Data Cleaning (Missing Values)

## Weak prompt

Clean this dataset.

Problems:

- No information about columns.
- No definition of clean.
- No preferred tools or language.

## Improved prompt

You are a data analyst. I have a CSV file with columns: `age`, `income`, `city`, `joined_date`.
1. Identify which columns have missing values and how many.
2. Propose sensible strategies to handle the missing values in each column (e.g. drop, mean/median, category Unknown).
3. Give me Python `pandas` code that:
   – prints the missing value counts,
   – applies your chosen strategies,
   – returns a cleaned DataFrame called `df_clean`.

Improvements:

- Clear columns and context.

# Bad vs Good Prompts: Data Types & Dates

**Weak prompt**

Fix the types in my data.

Problems:

- No column names.
- No target types.
- No expected output format.

Improvements:

- Clear mapping from columns to target types.
- Request for both explanation and code.
- Explicit assumptions.

**Improved prompt**

You are a data engineer working in Python. I have a `pandas` DataFrame `df` with columns:
– `user_id` (should be string),
– `age` (should be integer),
– `signup_date` (string in format `YYYY-MM-DD`),
– `is_active` (values yes/no).

1. Explain which data types are appropriate for each column.
2. Write `pandas` code that converts each column to the correct type, parses `signup_date` as `datetime`, and converts `is_active` to boolean.
3. Briefly explain any assumptions you make.

# Bad vs Good Prompts: Outliers

**Weak prompt**

Find outliers in my data.

Problems:

- No definition of outlier.
- No columns specified.
- No request for method transparency.

**Improved prompt**

Act as a data scientist. I have a DataFrame `df` with numeric columns `price` and `quantity`.
1. Suggest at least two methods to detect outliers in these columns (e.g. z-score, IQR).
2. For each method, explain in 2–3 sentences how it works and when it might fail.
3. Provide Python code using `pandas` and `numpy` that flags outliers in `price` and `quantity` using the IQR rule (1.5 * IQR).
4. Return the code in a single code block.

Improvements:

- Clear columns and expectations.
- Multiple methods + short explanations.
- Ready-to-run code requested.

# Role Prompting

- Assign the model a **role** to shape style & content:
  - You are a senior data analyst.
  - You are a patient math tutor.
  - You are an experienced UX researcher.
- Roles influence:
  - Vocabulary and tone.
  - Depth of detail.
  - Perspective and focus.

## Example

You are a senior product manager. Explain to a junior colleague why clear requirements are critical for successful software projects. Use practical examples.

# Chain-of-Thought Prompting

- Ask the model to **think step by step**.
- Helps for reasoning, planning, debugging, learning.
- Useful phrases:
  - Think aloud as you solve this.
  - Explain your reasoning step by step.
  - First outline the approach, then give the answer.

## Example

Show me step by step how you would break down this problem before giving the final answer: [problem].

# Decomposing Complex Tasks

- Break large tasks into **smaller stages**:
  1. Understand the problem.
  2. Generate options.
  3. Evaluate pros/cons.
  4. Produce final output.
- Run the model multiple times, one step at a time.

## Example

We will work in stages.
Step 1: Ask me up to 5 clarification questions about my project.
Step 2: Propose 3 different solution approaches.
Step 3: After I choose one, create a detailed plan.

# Iterative Refinement

- Treat AI as a **collaborator**, not a vending machine.
- Cycle:
  1. Draft.
  2. Critique.
  3. Revise.
- Use prompts like:
  - Critique this and suggest 3 improvements.
  - Rewrite this with your suggestions applied.

## Example

Here is my email draft. First, give me a brief critique (max 5 bullet points). Then rewrite the email applying your suggestions.

# Prompt Templates

- Create **reusable templates** for common tasks:
  - Emails.
  - Meeting summaries.
  - Code reviews.
  - Report outlines.
- Store them in a note or document for quick copy–paste.

## Template Example: Email Draft

You are a professional communication coach. Draft a clear, polite email to [recipient role] about [topic]. Include: (1) short context, (2) main request, (3) next steps. Use a friendly but concise tone.

# System vs User Instructions

- Many tools support a higher-level **system** message:
  - Global behavior, rules, persona.
- **User** message:
  - Specific task or question.
- Keep system instructions:
  - Stable, high-level.
  - Clear about boundaries & style.

## Example System Instruction

You are a careful assistant for software engineers. Always ask 1–2 clarification questions when requirements are ambiguous. Prefer concise, practical examples in code.

# Structured Outputs

- Ask for output in a **specific structure**:
  - Tables.
  - Bullet lists.
  - JSON / YAML.
- Helps when:
  - You want to import into other tools.
  - You need consistent formatting.

## Example (JSON)

Summarize this document into JSON with keys: "audience", "main_points" (list of strings), and "action_items" (list of strings). Return only valid JSON.

# Working with Tools (Conceptually)

- Modern AI systems can use tools:
  - Code execution.
  - Web browsing.
  - File uploads (e.g., PDFs, CSVs).
- Good prompts clarify:
  - What tool to use (if you can specify).
  - What you expect as a result.
- Example: Analyze this CSV and produce 3 key insights and 2 charts.

# Safety, Ethics, & Limitations

- Treat AI outputs as **drafts**, not ground truth.
- Watch for:
  - Factual errors and hallucinations.
  - Bias or unfair assumptions.
  - Leaking confidential information.
- Good practices:
  - Do not paste highly sensitive data.
  - Cross-check important facts.
  - Be transparent when AI is involved.

# Demo 1: Improving a Weak Prompt

- Start with a weak prompt, e.g.:
  - Write a project plan.
- Show the mediocre output.
- Then refine the prompt live:
  - Add role (You are a project manager).
  - Add context (Software rollout for 200 employees).
  - Add structure (phases, timeline, risks).
- Compare outputs and discuss.

# Demo 2: Building a Template

- Choose a task your audience cares about:
  - Meeting notes.
  - User interview summaries.
  - Bug reports.
- Live-build a prompt template:
  - Ask the room: What sections do we need?
  - Turn that into a reusable prompt.
- Save the final template as a prompt asset.

# Demo 3: Multi-Step Workflow

- Show a small workflow, e.g.:
  1. Research topic ideas.
  2. Select one and outline.
  3. Draft a blog post.
  4. Generate a summary and social media post.
- Emphasize:
  - Iteration and feedback.
  - Reusing context.
  - Controlling tone and audience.

# Business Use Cases

- **Marketing**
  - Campaign ideas.
  - Copy variants for testing.
- **Product / UX**
  - User story drafts.
  - Interview guides and summaries.
- **Operations**
  - Process documentation.
  - Checklists and SOPs.
- **Data / Analytics**
  - Exploratory analysis.
  - Hypothesis generation.

# Personal Use Cases

- **Learning**
  - Explain topics at different levels.
  - Create quizzes and flashcards.
- **Writing**
  - Drafts, outlines, rewrites.
  - Grammar and clarity improvements.
- **Planning**
  - Travel itineraries.
  - Study plans and schedules.
- **Idea generation**
  - Project ideas.
  - Hobbies and side projects.

# Key Takeaways

- Clear, specific prompts **significantly** improve results.
- Use the **Role–Task–Context–Format–Audience** checklist.
- Break big problems into **multi-step workflows**.
- Treat AI outputs as **drafts**, and iterate.
- Build your own library of **prompt templates**.

# Next Steps

- Pick one task you do every week.
- Design a prompt template for it.
- Use it for 2 weeks and refine.
- Share good prompts with your team.

# Next Steps

- https://medium.com/@pvnsripati/as-a-developer-struggling-with-ai-prompts-master-ai-prompt-engineering-now-0616c8e74165
- https://pbiecek.github.io/books/
- https://github.com/pbiecek/ema

Questions?