

Day 1 – LLM Foundations: Prompting, Context & Grounding

Naeemullah Khan

naeemullah.khan@kaust.edu.sa



جامعة الملك عبد الله
للعلوم والتقنية

King Abdullah University of
Science and Technology

KAUST Academy
King Abdullah University of Science and Technology

October 18, 2026

1. Learning Outcomes
2. How LLMs Work
3. Choosing and Running a Model
4. Prompting
5. Structured Outputs
6. Sampling
7. Context Engineering
8. Grounding with RAG
9. Today's Labs
10. Limitations
11. Summary
12. References

What it does well

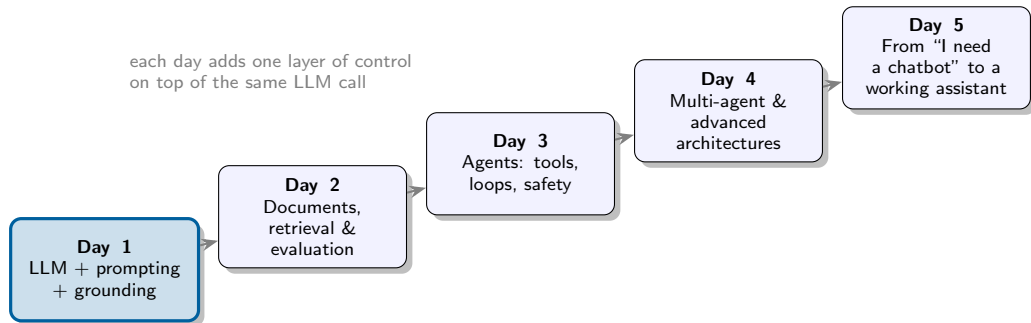
- ▶ Writes, summarizes and translates fluently
- ▶ Follows instructions in Arabic and English
- ▶ Reasons through multi-step problems
- ▶ Writes and explains code

What goes wrong

- ▶ **Invents facts** with full confidence ✗
- ▶ **Breaks your format** when software needs JSON ✗
- ▶ **Knows nothing** after its cutoff or about your documents ✗
- ▶ **Obeys instructions hidden in data** ✗

Every agent this week is built from the same part: an LLM call. Today is about making that one call **predictable, measurable and grounded**.

This Week: From a Model to a System



The ladder follows one rule: **use the simplest rung that solves the problem.**

Failure you will see	Fix	Section
Vague, inconsistent answers	Specific prompts, examples, reasoning	Prompting
Output your code cannot parse	Schemas + constrained decoding	Structured outputs
Same prompt, different answers	Sampling parameters, voting	Sampling
Forgets, overflows, misses the middle	Decide what goes in the window	Context engineering
Confident answers about unknown facts	Retrieve, ground, cite, refuse	RAG basics

How we will work

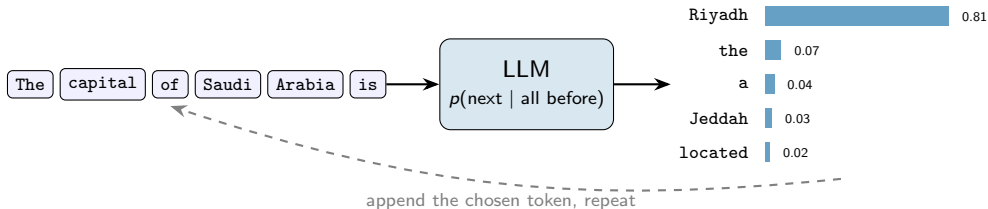
Every claim in this deck is something you **measure** in the lab on real data: Arabic tweets, a company handbook, and the answers a small open model actually gives.

By the end of this session, you will be able to:

- ▶ **Explain** how an LLM turns text into tokens and tokens into an answer
- ▶ **Choose** between an API model and an open model you run yourself, and size it for a GPU
- ▶ **Write** prompts that are specific, use examples well, and separate instructions from data
- ▶ **Force** machine-readable output with a schema instead of hoping for it
- ▶ **Control** randomness with temperature, top- p and friends – and know when not to
- ▶ **Decide** what goes into the context window, and in which order
- ▶ **Build** a minimal grounded answerer that cites its sources and says “I don’t know”

Day 1: **How LLMs Work**

An LLM Does One Thing: Predict the Next Token



- ▶ Every answer, translation, JSON object or tool call is generated **one token at a time**.
- ▶ The model outputs a **probability distribution**; a *sampler* picks the token (Section: Sampling).
- ▶ Nothing is looked up: facts come from patterns in the training data – or from **what you put in the prompt**.

Source: probabilities are illustrative.

- ▶ A **tokenizer** splits text into sub-word pieces from a fixed vocabulary (typically 100k–250k entries), learned from data (e.g. BPE).
- ▶ Frequent strings become one token; rare ones are split.
- ▶ The model never sees letters – which is why it miscounts characters and stumbles on long numbers.

English

un belie vable

Arabic

والمستخدمين → ين سخدم الم و

“and the users”: conjunction + article + stem + plural suffix

Source: Qwen3.5-4B tokenizer, measured by the course authors.

Why you care

You pay per token, the context window is measured in tokens, and speed is tokens per second.

Tokenizer	EN	AR	AR / EN
GPT-2 (2019)	33	120	3.64
GPT-4 (cl100k)	33	88	2.67
Qwen2.5	33	50	1.52
GPT-4o (o200k)	33	43	1.30
Qwen3.5	33	40	1.21

Source: UDHR Article 1, official UN English and Arabic texts; counted by the course authors.

- ▶ Same meaning, same model family – up to **3.6**× the tokens.
- ▶ More tokens = higher cost, less room in the context, slower answers.
- ▶ Newer tokenizers with more Arabic in their vocabulary close most of the gap.

The sentence we counted: Universal Declaration of Human Rights, Article 1

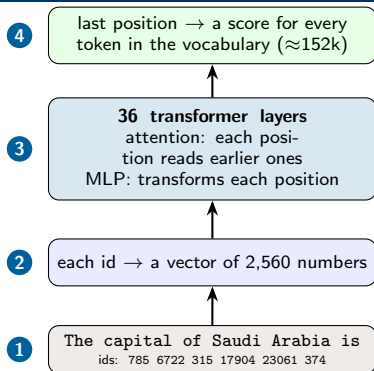
English (33 tokens with Qwen3.5): “All human beings are born free and equal in dignity and rights. They are endowed with reason and conscience and should act towards one another in a spirit of brotherhood.”

Arabic, official UN text (40 tokens):

يولد جميع الناس أحراراً متساوين في الكرامة والحقوق.
وقد وهبوا عقلاً وضميراً وعليهم أن يعامل بعضهم بعضاً بروح الإخاء.

Rule

Before choosing a model for Arabic users, **count tokens on your own text.**



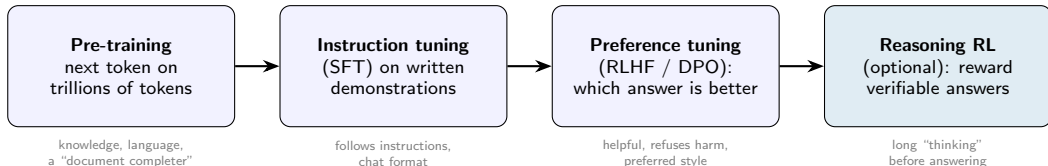
Source: token ids, vocabulary and sizes: Qwen3-4B tokenizer and config;

architecture: Vaswani et al., NeurIPS 2017.

1. **Tokenize:** text becomes token ids (previous slides).
2. **Embed:** each id looks up a learned vector – the model's internal representation of that token.
3. **Transform**, layer after layer. **Attention** lets each position read the earlier ones (only earlier: the model is *causal*); this is how "Arabia" informs "is", and how a fact in your prompt reaches the answer. The **MLP** then transforms each position on its own.
4. **Predict:** the last position's vector becomes a score for every token in the vocabulary; the sampler picks one (Section: Sampling).

Parameters vs context

The weights ("4B") store general patterns from training; the context stores what is true *for this request*. Attention over a longer context costs more – the reason long prompts are slower.



- ▶ A **base** model (stage 1 only) learned to continue text. It has no notion of "a user" or "an answer".
- ▶ An **instruct** or **chat** model went through the later stages: it treats your text as a request and replies.
- ▶ Model names tell you which you have: Qwen3.5-4B-Base vs Qwen3.5-4B. For applications, use the instruct model.

Prompt: **“Write a short email to my manager asking for a day off.”** (greedy decoding, 80 new tokens)

Qwen3.5-4B-Base: raw text in, text out

I have a family emergency.

<think>

Hmm, the user needs a short email to request a day off due to a family emergency. The key points are brevity, clarity, and professionalism. ...

It **continued the user’s message** as a document – then imitated a reasoning trace it saw in pre-training. It never wrote the email.

Qwen3.5-4B: through the chat template

Subject: Time Off Request -- [Your Name] --
[Date(s)]

Hi [Manager’s Name],

I hope you’re having a good week.

I would like to request a day off on [Date]. I have ensured that my current tasks are up to date ...

It **answers the user**: the request is in a user turn, and the model was trained to reply.

The line is blurrier than it used to be

Asked “What is the capital of France?”, **both** replied “The capital of France is ****Paris****.” Modern pre-training data is full of chat-style text, so base models often look helpful on simple questions – and fall apart on real tasks.

Source: Qwen/Qwen3.5-4B-Base and Qwen/Qwen3.5-4B, float32, run by the course authors; outputs shortened at “...”.

The messages list you send is flattened into one string with special tokens. This is the exact text Qwen3.5-4B (our lab model) receives:

```
messages = [  
    {"role": "system",  
     "content": "You are an HR assistant."},  
    {"role": "user",  
     "content": "How many leave days?"},  
]  
tok.apply_chat_template(messages,  
    add_generation_prompt=True,  
    enable_thinking=False,  
    tokenize=False)
```

```
<|im_start|>system  
You are an HR assistant.<|im_end|>  
<|im_start|>user  
How many leave days?<|im_end|>  
<|im_start|>assistant  
<think>  
  
</think>
```

- ▶ Each model family has its **own** template; the wrong one silently degrades answers.
- ▶ Thinking *off* still inserts an empty <think></think> block – the source of stray tags in naive parsers.
- ▶ Always build prompts with `apply_chat_template`, never by hand.

Role	Carries
system	persona, rules, output format, tools
user	the request and its data
assistant	the model's previous turns
tool	results of tool calls (Day 3)

- ▶ Models are **trained** to give the system message priority over the user.
- ▶ Put stable rules in `system`, the changing request in `user`.
- ▶ The model sees everything as one token stream – roles are a convention it learned.

The system prompt is not a security boundary

A user (or a document) that says “ignore previous instructions” sometimes wins – especially with small models. You will watch this happen in the lab; Day 3 covers real defenses.

- ▶ Trained with reinforcement learning to write a long **chain of thought** before the final answer, rewarded when the answer checks out.
- ▶ Families: OpenAI o-series (2024), DeepSeek-R1 (2025), Claude extended thinking, Qwen3 / Qwen3.5 **hybrid** models (thinking on or off).
- ▶ Controls: on/off, a **thinking budget** (max reasoning tokens), effort level.
- ▶ Thinking tokens are generated – you **pay and wait** for them.

user: 17% of 84,000?

<think> 10% is 8,400; 7% is 5,880;
total 14,280. Check: $0.17 \times 84,000 = 14,280$. </think>

answer: 14,280

Task	Thinking?	Why
Multi-step maths, logic, planning, code	✓	errors compound; checking helps
Hard tool-use decisions (Day 3)	✓	choosing the next step is reasoning
Classification, extraction, formatting	✗	no reasoning needed; adds latency
Simple factual lookup with retrieved text	✗	the answer is in the context
Latency-critical chat, high volume	✗	thinking can be 10× the answer length

Small models

A 3–4B model in thinking mode can ramble, loop, or exhaust `max_new_tokens` before it answers. Set a budget and test both modes on your task.