Powering Pepper with LLM

Lukáš Navara · FEL ČVUT · Diploma thesis

Do you use LLMs?

For what?

Today's plan

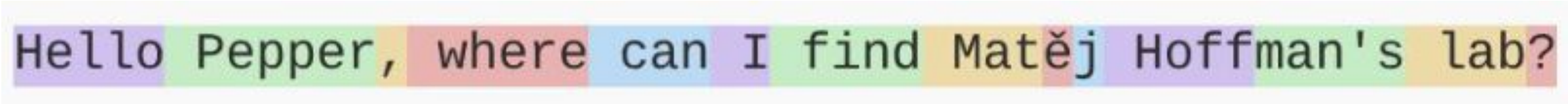
90 minutes, three blocks

30 min	**I explain**	LLMs, prompts, and tools. Intro to the thesis.
30 min	**You build**	Write your own system prompt and a tool in a small CLI.
30 min	**We try it**	I wire your agent into Pepper and you can talk to her.

An LLM is a next-token predictor

$$P(w_{\square} \mid w_1, \ldots, w_{\square-1})$$

Given everything so far, guess the next token. Then repeat.



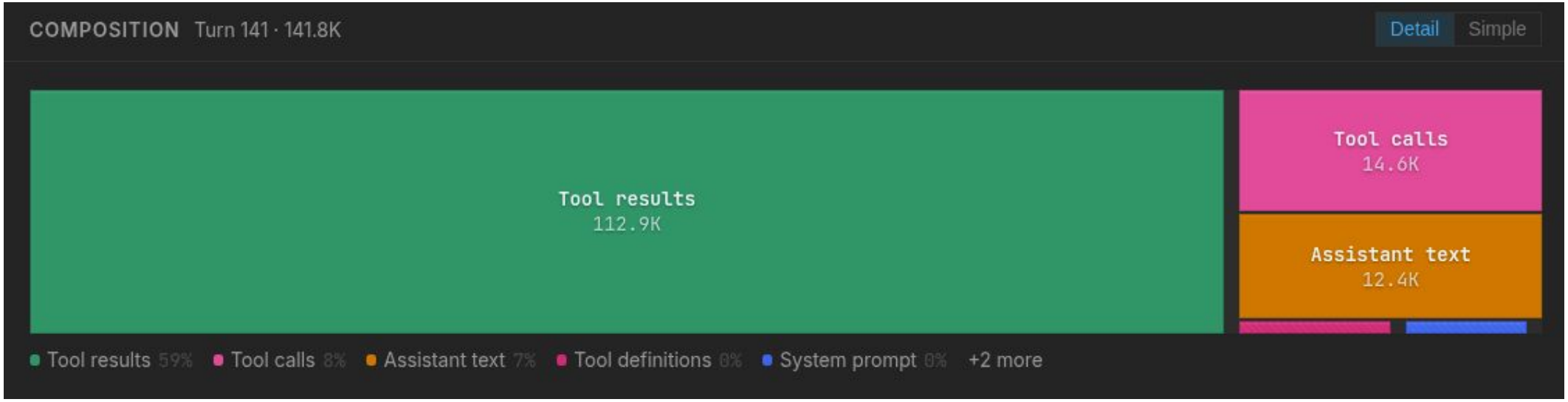
The context window

- System prompt (persona, rules)
- Chat history (previous turns)
- Tool results
- User's current message

Model sizes

GPT-4o-mini — 128 K tokens (≈ a short novel)

Claude Sonnet — 200 K to 1 M (fits an entire codebase)



Limitations

Hallucinations	Confidently wrong facts.
Non-determinism	Same input, different output (temperature).
Stale knowledge	Training cutoff. It doesn't know last week.
Latency	Scales with output length. Streaming helps.
Running out of context window	Model “forgets” the past conversation.

Three ways to steer an LLM

METHOD	WHEN TO USE
Fine-tuning	Domain-specific style, stable data
Prompt engineering	Persona, tone, rules
Tool calling	External knowledge, real actions

The system prompt

The first message the LLM sees every turn.

Persona

who the model is — role, voice, attitude

Rules

what it must or must not do

Tool policy

when to call which function

system_prompt.py

```
SYSTEM_PROMPT = """
You are a polite FEL
receptionist. Answer briefly.
If the user asks about rooms,
people, or events, call a tool.
Never invent room numbers.
"""
```

```
messages = [
    {"role": "system",    "content": system_prompt},

    {"role": "user",      "content": "Ahoj!"},
    {"role": "assistant", "content": "Ahoj! How can I help?"},
```

Same tool, same question.

Different prompt. Wildly different output.

USER *"Where is room T2:H1-131?"*

System: polite receptionist

"Of course — room T2:H1-131 is on the first floor of building T2, to your right as you leave the lobby. Let me know if you'd like me to guide you there."

System: grumpy pirate

"Arrrr. Ye be wantin' T2:H1-131, landlubber? Hoist yer sails to the first deck o' T2, starboard side past the lobby. Now scram."

What is a tool?

A Python function you expose to the model.

The model decides

- whether to call a tool at all
- which tool, if there are many
- what arguments to pass
- whether to call several in one turn

Your code decides

- what actually runs
- validation of arguments
- database, API, robot motor
- the return value the model sees

A tool has two parts

The function runs. The schema teaches.

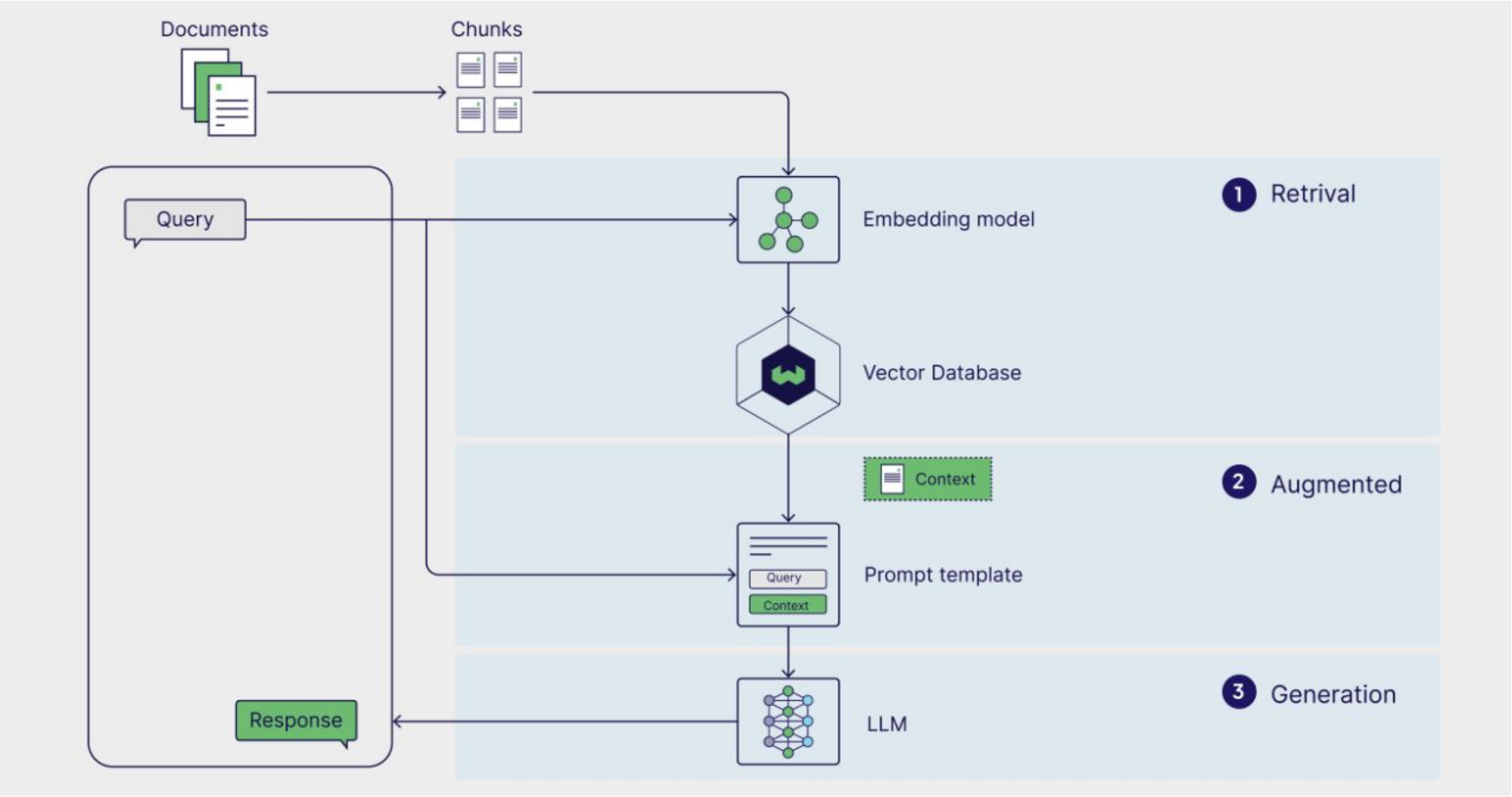
function — what runs

```
def get_weather(city: str) -> str:
    # Call weather API,
    # return human-readable
    # summary.
    ...
```

schema — what the LLM reads

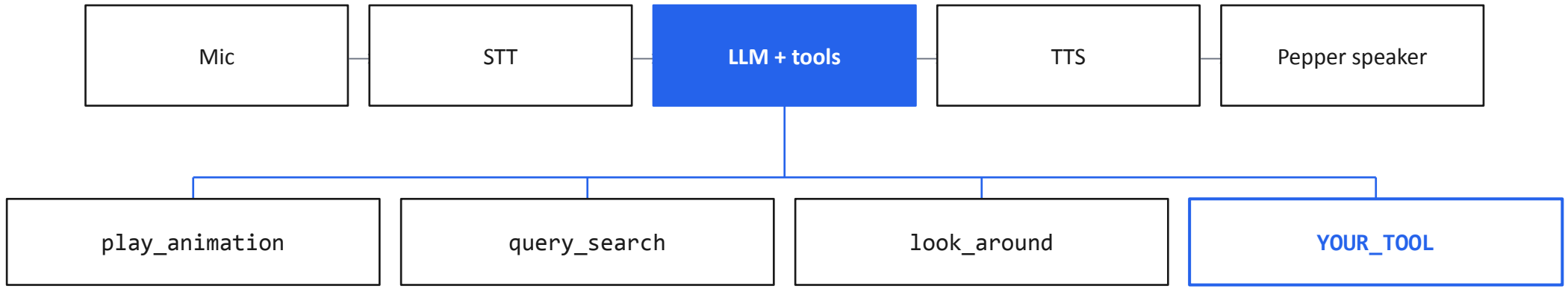
```
GET_WEATHER_SCHEMA = {
  "function": {
    "name": "get_weather",
    "description": "get the weather",
    "parameters": {"city": {…}}
  }
}
```

RAG - retrieval augmented generation



https://ragplay.vercel.app/experiment

How Pepper is wired



Your 30 minutes

Write a prompt. Write a tool. I'll wire it into Pepper.

Getting started

```
# 1. Visit the repo, install etc:

https://github.com/navarlu/pepper-tool-playground

# 2. Tune your system prompt and tools.

# 3. send it to: navarlu2@fel.cvut.cz
```

Idea menu

`flip_coin()`

`tell_joke(topic)`