LLM hackathon

A practical introduction to programming with LLMs

Hadley Wickham

Chief Scientist, Posit

Get started by following the setup instructions at https://github.com/hadley/workshop-llm-hackathon

Framing

- Practical, actionable information.
- We'll treat LLMs as black boxes, focussing on what they can do, not how they work.
- Just enough knowledge so you know what to search for (or ask an LLM about!)
- Plenty of practice with the tools!

1. Set up
2. Anatomy of a conversation
3. Prompt engineering
4. Non-text inputs
5. Structured data
6. Your turn

Setup

Getting started

- You'll need dev ellmer
  - `pak::pak("tidyverse/ellmer")`
- And an account with an LLM provider:
  - Option 1: claude. Cheap & good at R code.
  - Option 2: gemini. Free & good at videos.
  - (Plus many others at https://ellmer.tidyverse.org/reference/index.html)
- If you pay for Claude, OpenAI etc, unfortunately you still need a separate account for API access.

Your turn: get this working

https://github.com/hadley/workshop-llm-hackathon

```
library(ellmer)

chat <- chat_claude("You are a terse assistant.")
# or
chat <- chat_gemini("You are a terse assistant.")

chat$chat("What is the capital of the moon?")

# While you wait
live_console(chat)
live_browser(chat)
```

Initial vocabulary

Provider

System prompt

```
chat ← chat_claude("You are a terse assistant.")
#> Using model = "claude-sonnet-latest"
```

Model

```
chat$chat("What is the capital of the moon?")
```

User prompt

Initial vocabulary

```
> chat
<Chat turns=3 tokens=22/19>
── system ──────────────────────────────────────────────
You are a terse assistant.
── user ────────────────────────────────────────────────
What's the capital of the moon?
── assistant ───────────────────────────────────────────
The moon has no capital - it's an uninhabited celestial body.
```

Input/output tokens

System prompt

User prompt

Model response

Anatomy of a conversation

Overview

- Each interaction is a pair of user and assistant **turns**, corresponding to a HTTP request and response.
- The API server is entirely stateless, despite conversations being very stateful!
- Can see exactly what ellmer is sending and receiving by using `httr2::with_verbosity(code, 2)`.
- We'll use that to explore what's going on under the hood.

HTTP request

Model

System prompt

User prompt

```
{
    "model": "claude-3-5-sonnet-latest",
    "system": "You are a terse assistant.",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "What is the capital of the moon?"
                }
            ]
        }
    ],
    "stream": false,
    "max_tokens": 4096
}
```

HTTP response


```
{
  …
  "type": "message",
  "role": "assistant",
  "content": [
    {
      "type": "text",
      "text": "The moon has no capital as it has no human settlements or political structures."
    }
  ],
  "usage": {
    "input_tokens": 22,
    "cache_creation_input_tokens": 0,
    "cache_read_input_tokens": 0,
    "output_tokens": 18
  }
}
```


Model response

HTTP response


```
{
  …
  "type": "message",
  "role": "assistant",
  "content": [
    {
      "type": "text",
      "text": "The moon has no capital as it has no human settlements or political structures."
    }
  ],
  "usage": {
    "input_tokens": 22,
    "cache_creation_input_tokens": 0,
    "cache_read_input_tokens": 0,
    "output_tokens": 18
  }
}
```

What is a token?

- Featuring engineering to represent words as number.
- Common words represented with a single number
  - What is the capital of the moon?
  - 4827, 382, 290, 9029, 328, 290, 28479, 30
- Other words may require multiple numbers:
  - counterrevolutionary
  - 32128 (“counter”), 264 (“re”), 9477 (“volution”), 815 (“ary”)
- See details at https://tiktokenizer.vercel.app/

Why do tokens matter?

- API pricing is by token
  - https://www.anthropic.com/pricing#anthropic-api
  - Input: $3 / million tokens
  - Output: $15 / million tokens
- Context size
  - e.g. 200K for claude sonnet (1M for gemini flash 2.0)
  - 200K = 150,000 words / 300-600 pages / 1.5-2 novels
  - = 1/3 of Lord of the Rings

This seems like plenty but ...

```
{
    "role": "user",
    "content": [{
       "text": "What's the capital of the moon?"
    }]
},
{
    "role": "assistant",
    "content": [{
        "text": "The moon has no capital - it's an uninhabited celestial body."
    }]
},
{
    "role": "user",
    "content": [{
       "text": "Are you sure?"
    }]
}
```

Your turn

```
library(ellmer)
options(httr2_verbosity = 2)

chat <- chat_claude("You are a terse assistant.", echo = FALSE)
chat$chat("Tell me a joke about a statistician and a data scientist")
chat$chat("Explain why that's funny")
chat$chat("Make the joke funnier")

# Look at the request and response and verify that you see it
# growing. Does this help you understand any features of LLMs
# that you've observed in your own usage?
```

Tool calling

- Can provide the LLM with tools (aka functions) that it can call.
- Useful for:
  - Things LLMs aren't good at (e.g. basic maths)
  - Providing current information
  - Looking up extra info
  - Doing something on your behalf (potentially dangerous)

LLMs don’t know what day it is. But you can provide a tool:

```
chat ← chat_claude("You're a terse assistant")
chat$chat("What's today's date?")
#> I don't know today's date. I can't access real-time information.

chat$register_tool(tool(
  function() Sys.Date(),
  "Gets the current date",
  .name = "today"
))
chat$chat("What's today's date?")
#> It's March 4, 2025.
```

How does this work?

```
<Chat turns=5 tokens=803/48>
── system ──────────────────────────────────────────────────────────
You're a terse assistant
── user ────────────────────────────────────────────────────────────
What's today's date?
── assistant ───────────────────────────────────────────────────────
[tool request (toolu_01VXAFTNu19X9×2dT6zFiwhy)]: today()
── user ────────────────────────────────────────────────────────────
[tool result  (toolu_01VXAFTNu19X9×2dT6zFiwhy)]: 2025-03-04
── assistant ───────────────────────────────────────────────────────
Today is March 4, 2025.
```

Your turn

```
# Modify this code so the LLM returns the correct age, as of
# today. Either inspect requests and responses to confirm your
# understanding of how tool calling works.

chat <- chat_claude()
chat$chat("How old is Cher? Explain your working")
```

Prompt design

Treat AI like an infinitely patient new coworker who forgets everything you tell them each new conversation, one that comes highly recommended but whose actual abilities are not that clear. … Two parts of this are analogous to working with humans (being new on the job and being a coworker) and two of them are very alien (forgetting everything and being infinitely patient). We should start with where AIs are closest to humans, because that is the key to good-enough prompting

— **Ethan Mollick**

General structure

- Prompts can get long, so it makes sense to organise in a way that's useful for humans and LLMs: markdown.
- Put your prompt in a separate file.
- Use ellmer::interpolate_file() to read and add any dynamic data

Example prompt

```
You are an expert ggplot2 programmer.

Help me brainstorm ways to visualise the <dataset> described below.
Give me 10 different places to get started.

* Be creative!
* Return the results as a single block of code, using comments to separate the
  plots

<dataset>
{{glimpse}}
</dataset>
```

Your turn

- Improve the prompt! Here are some ideas:
  - Do you want to steer it towards or away from using other packages? Just the tidyverse package? Other extensions?
  - Can you prevent it from using theme_minimal()? Can you encourage it to use the base pipe instead of magrittr? What other code style issues could you improve?
  - Can you get it to explain the goal of the plot inline with the comment?
  - It has a tendency to go all out and include a large number of variables in the visualisation. Can you make it focus on simpler plots?
  - Is it still useful with a dataset that its never seen before? What other information might you want to include about the data?

Also worth reading the advice from specific providers

- Claude
- OpenAI
- Gemini

Non-text inputs

Content types

```
{
    "model": "claude-3-5-sonnet-latest",
    "system": "You are a terse assistant.",
    "messages": [
        {
            "role": "user",
            "content": [
                {
                    "type": "text",
                    "text": "What is the capital of the moon?"
                }
            ]
        }
    ],
    "stream": false,
    "max_tokens": 4096
}
```

What else can go here?

	Image	PDF	Video
OpenAI	✅		
Claude	✅	✅	
Gemini	✅	✅	✅

Providing non-text inputs

```
chat ← chat_claude()

chat$chat(
  content_image_file("holly-mandarich-3p9zaNwUtv8-unsplash.jpg"),
  "Describe this photo"
)
chat$chat("Where in the world do you think it is?")

# Note the number of tokens used
chat
```

Other functions

```
content_image_url()
content_image_file()

content_pdf_url()
content_pdf_file() # needs dev version

gemini_upload() # needs dev version
```

Structured data

Imagine you have a recipe...

```
recipe ← "In a large bowl, cream together 1 cup of softened unsalted
butter and ½ cup of white sugar until smooth. Beat in 1 egg and 1
teaspoon of vanilla extract. Gradually stir in 2 cups of all-purpose
flour until the dough forms. Finally, fold in 1 cup of semisweet
chocolate chips. Drop spoonfuls of dough onto an ungreased baking
sheet and bake at 350°F (175°C) for 10-12 minutes, or until the edges
are lightly browned. Let the cookies cool on the baking sheet for a
few minutes before transferring to a wire rack to cool completely.
Enjoy!"
```

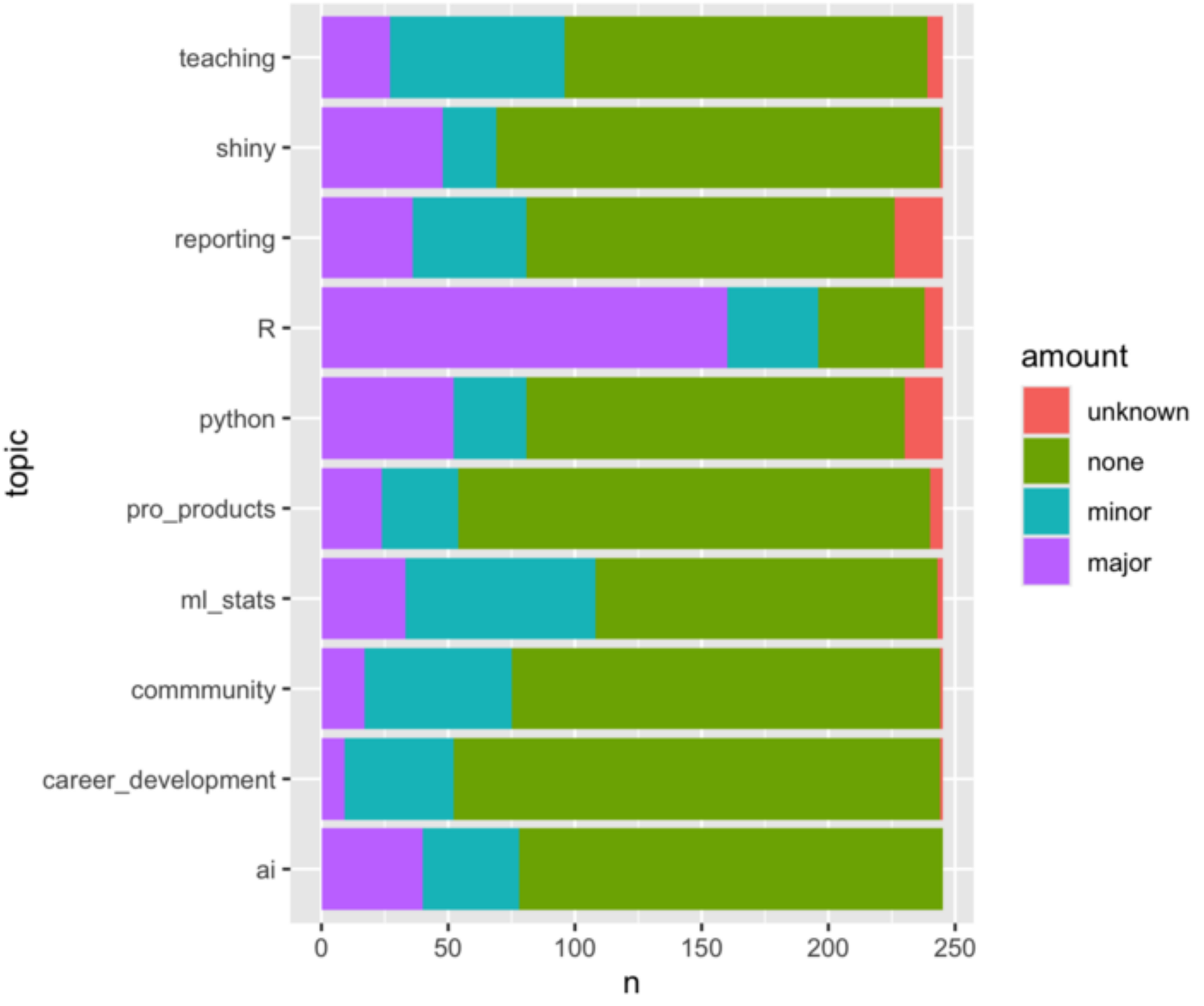
Structured data extraction

- As well as returning text, most providers have some way to generate structured data.
- You'll provider a type specification (with type_object(), type_array(), type_string(), etc) and the results are guaranteed to fit that specification.
- Not clear to me how this interacts with your prompt. You'll need to experiment to get the best results.
- Great combination with non-text data.

recipes-example.R

Other input types

- Images (openAI, claude, gemini)
- Videos (gemini)
- Audio (gemini)
- PDF (claude, gemini)


teaching
shiny
reporting
R
python
pro_products
ml_stats
commmunity
career_development
ai
topic
0
50
100
150
200
250
n
amount
unknown
none
minor
major

Your turn!

Project ideas

Create a custom prompt to solve a common coding problem. See https://simonpcouch.github.io/chores/ for some examples.

Use tool calling to give the LLM additional info.

Extract structured data from unstructured text, PDFs, images, video, ...

If you're familiar with shiny, you could use https://github.com/jcheng5/shinychat to make your own chatbot. If you’re not, ask https://jcheng.shinyapps.io/ellmer-assistant/ to make you an app.