$\boxed{\text{Lec 7}}$ Prompting

Basic Prompting

Give sequence, pre-trained model completes it.

Ancestral sampling, temp 1.0, GPT2 → usually bad
Set top-k = 50, top-p = 0.95 → makes more sense

Prompting Workflow

1. Fill template
2. Predict answer
3. Post process

put user input here
[x]. Overall, it was [z]
model continues

- Chat prompt — Open AI message format

can have names as well, e.g. a sys to add examples

Roles:	
system	influence behaviour
user	user input
assistant	previous system outputs

LLaMa prompt templates

Note the template should match those used in training

sys:
```
[INST]
<< SYS >>
text...
<< / SYS >>
[/ INST]
```
← Note OpenAI probably trained model to not spit out sys messages to user

user: `[INST] text [/ INST]`

assistant: `text`

- Post proc
  - Extract answer
    - └ Keyword indicator
    - └ Number extraction
  - Reformat
    - └ to JSON, markdown render, etc.
    - └ put code in block
  - Output mapping
    - └ ex. {fantastic, great, ... } ⇒ positive
    - └ keep in mind: map from frequent occurrances in corpus
      - └ ex. output "very good" over "5 stars"

Few Shot Prompting / In-Context Learning

basically the same thing but different perspective

Few shot: Instruction + few examples ← 0-Shot is just having 0 example

Instruction: Stronger model follows this better
few examples: helps with format

No gold standard on how to do this

LLMs are sensitive to the in-context examples, usually
→ Example ordering
→ Label balance
→ Label coverage

* This can be counter-intuitive
- A paper tries random (wrong) answers in examples & finds wrong example better than no example
- more example can also hurt
  - └ forgetting instructions, etc.

So more of art than science

Chain of Thought Prompting

Make model explain before answering
In practice:
→ give it Q&A examples with reasoning in the A.
→ 0-shot prompt with "A: Let's take this step by step"
viz ask model to reason
└ model in the wild maye have been tuned to do this without prompt

- Breaks hard problems into easier sub-problems
- Works well on math

Structure Output as Programme

Structured Output
- depencies
- procedure
- graph (DOT format)
- python code ← found more effective, code naturally have dependencies
- json format ← easier to parse, model also seen lots of json

Programme - Aided LMs

Few-shot examples include code & code output
Model outputs code and system can run it

→ Agents & tools, later

Prompt Engineering

- Manual template
  - Format
    - └ Should try to match format of data
    - └ ex. even missing space can lead to very low accuracy
    - └ most format lead to bad perf
  - Instructions
    - └ Clear, concise, human-understandable
    - └ Precise length, audience, etc.
      - * Note modern LLMs usually don't complain unclear prompt
- Auto search
  - In text space
  - Prompt paraphrasing
    - └ Use paraphrase model to try many prompts
  - In cont. space
    - Gradient search, prompt tuning
      - └ backprop into prompt's tkn embs, and clamp to nearest tkns, or even keep them as new tokens
    - → Adversarial attacks by prompt search
    - Prefix tuning
      - └ prepend tunable prefix to every layer

Prompt is like prior, models can be tuned with prompts