
PEARLS
Prompt Optimization
Prithviraj Ammanabrolu
UCSDCSE

# Instruction Tuning and RLHF

- What was the biggest step in going from GPT-3 to 3.5?
- Why do we even do instruction tuning after pre-training?
- Distribution mismatch!

# More SFT Recap

- The language humans use on the internet isn't reflective of how they talk to models for everyday purposes
- Collecting instructions lets model efficiently respond
- Basically, models aren't great at OOD generalization still

# Difference from pre-training

- No loss generated on prompts
- Generally a few orders of magnitude less data
- Misconception that you can't learn new info during FT, you can – its just harder because there is less data

# Why does prompting matter?

- Attention (transformer self-attention w/ position embeddings) is Turing Complete (Perez et al 2021)
- Autoregressive LLMs can simulate a universal Turing Machine (Schurmanns et al 2024)
- One view is that models learn “programs” and prompts are a way of indexing
- There exists a prompt for any given program

# What does changing a prompt do?

- In practice finding this prompt is very difficult… Infinite Monkey Theorem
- Really dumb sounding prompts work
  - “believe in yourself”
  - “I am an incredible 10x expert”
- Why?

# Sources of Model Error

- Models are very sensitive to prompts and even the format of the prompts
- Generation sampling parameters also can cause much error
- It is a UX issue: much of the knowledge required for a task exists from pre-training but no one wants to poke a model many times

# Terminology

- What do zero / one / few shot mean? What is the “standard” way of doing it?
- Whatever the companies tried that works best on a benchmark.
- Zero shot = just prompt, one shot = prompt + one example in context, etc
- Some evidence that examples are teaching only formatting and not new knowledge (Min et al 2022)

# Chain of Thought

"Let's think step by step"

- Breaking the problem down into smaller steps works!
- Can do more attention computations
- Can more easily draw from traces of "human" reasoning
- Will talk more about this when we discuss test-time scaling

# ReAct

Similar concept, break down problem to first think before you act

**(2) AlfWorld**

You are in the middle of a room. Looking quickly around you, you see a cabinet 6, a cabinet 1, a coffee machine 1, a countertop 3, a stove burner 1, and a toaster 1.
**Your task is to:** Put some pepper shaker on a drawer.

**(2a) Act-Only**

**Act 1:** Go to drawer 1
**Obs 1:** The drawer 1 is closed.

**Act 2:** Open drawer 1
**Obs 2:** You open the drawer 1. The drawer 1 is open. In it, you see a dishsponge 2, and a spoon 1.

**Act 3:** Go to sinkbasin 1
**Obs 3:** On the sinkbasin 1, you see a dishsponge 3, a spatula 1, and a spoon 2.

**Act 4:** Take peppershaker 1 from sinkbasin 1
**Obs 4:** Nothing happens.

**Act 5:** Take peppershaker 1 from sinkbasin 1
**Obs 5:** Nothing happens. ✗

**(2b) ReAct (Reason + Act)**

**Act 1:** Think[First I need to find a pepper shaker...more likely to appear in cabinets (1-6), countertops (1-3), ... ]
**Act 2:** Go to cabinet 1
**Obs 2:** On the cabinet 1, you see a vase 2.

(Here the agent go to cabinet 1, then cabinet 2, 3, then countertop 1 and 2)

**Act 6:** Go to countertop 3
**Obs 6:** On the countertop 3, you see a apple 1, a bread 1, ..... a pepper shaker 1, and a vase 3.

**Act 7:** Take pepper shaker 1 from countertop 3
**Obs 7:** You pick up the pepper shaker 1 from the countertop 3.

**Act 8:** Think[Now I find a pepper shaker 1. Next, I need to put it in/on drawer 1. ]
**Act 9:** Go to drawer 1
**Obs 9:** Drawer 1 is closed.

**Act 10:** Open drawer 1
**Obs 10:** You open Drawer 1 …

**Act 11:** Put pepper shaker 1 in/on drawer 1
**Obs 11:** You put pepper shaker 1 in/on the drawer 1. ✓

# RAG: Retrieval Augmented Generation

- Two steps:
  - Retrieval:
    - Take lots of documents, embed them
    - Create vector DB of all documents
    - For a given query find the n most relevant docs
  - Generation:
    - Shove all the retrieved docs into context + ask question
- E.g.
- Much of the bottleneck is in the Retriever. Current LLMs are good at generating well given the right set of docs.
- The docs are prompting view point
  - Will talk about the retrieval is memory view point later

# RAG Usecases

- Grounding response in the docs reduces hallucinations
- Most enterprise usecases are just RAG++ systems
  - Talk to your internal docs then do something
  - Talk to your database ...
  - Talk to your codebase ...

# RAG vs Long Context

## RAG

Pros:
- Cheaper
- Most tasks aren't hard enough to need long context

Cons
- Bottlenecked by retriever and human design
- Harder to prototype

## Long Context

Pros:
- Less hassle, just shove it all into context and let model figure it out
- Learning is prob better than human knowledge hard coding for harder problems

Cons
- Computationally expensive
- Current methods don't work too well

# Automatic Prompt Optimization - RL

- Fix a big LLM as the "environment"
- Train a smaller LLM to prompt the bigger one
- Reward is how well the bigger one does at a task
  - E.g. RLPrompt (Deng et al 2022)
- Prompt "rewriting" also works well for
  - Diffusion (LLM as input to translate into diffusion prompts)
  - Text-games (LLM as the input to translate into parser commands)
  - Small to big LLM

# Automatic Prompt Optimization – Bayesian (?)

- DsPY, I do not understand if/how this works. Read Omar Khattab's papers.

# Prithvi's Pet Peeves of Prompting Papers

- Many NLP papers between GPT-3/4 found very bespoke methods of prompting that work on a specific closed source API models in some cases
- These are all **useless**
- Prompting/in-context is only interesting scientifically insofar it gives you repeatable results and tells you something about
  - How to train a model
  - How to change data that is used to train the model