Summer School Week3

洪晙宸

Slide here!

Prompting, Instruction Finetuning, and RLHF

From Language Models to Assistants

- Zero-shot and Few-shot In-Context Learning
- Instruction finetuning
- Reinforcement Learning from Human Feedback

GPT: Generative Pretrained Transformer

- GPT: 117M parameters, 4.6GB texts, 12 decoder layers.
- GPT2: 1.5B parameters, 40GB of internet text data. Scrape links posted on Reddit w/ at least 3 upvotes
  - Zero-Shot learning
- GPT3: 175B parameters, over 600GB data
  - Few-shot learning

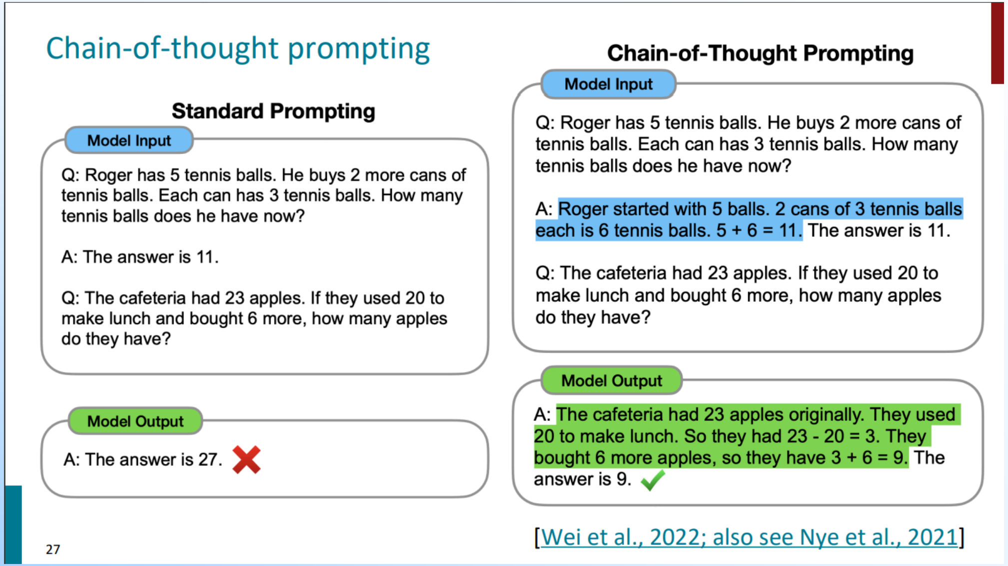
Zero vs One vs Few shot

- Zero-Shot: No example provided, ask the model to contineue the sequence
- One-Shot: Given an example, and ask the model to contineue the sequence.
- Few-Shot: Given a few more examples, then ask the model to contineue the sequence

> **Tips**
> Compare with traditional finetuning, it doesn’t require gradient update to the model

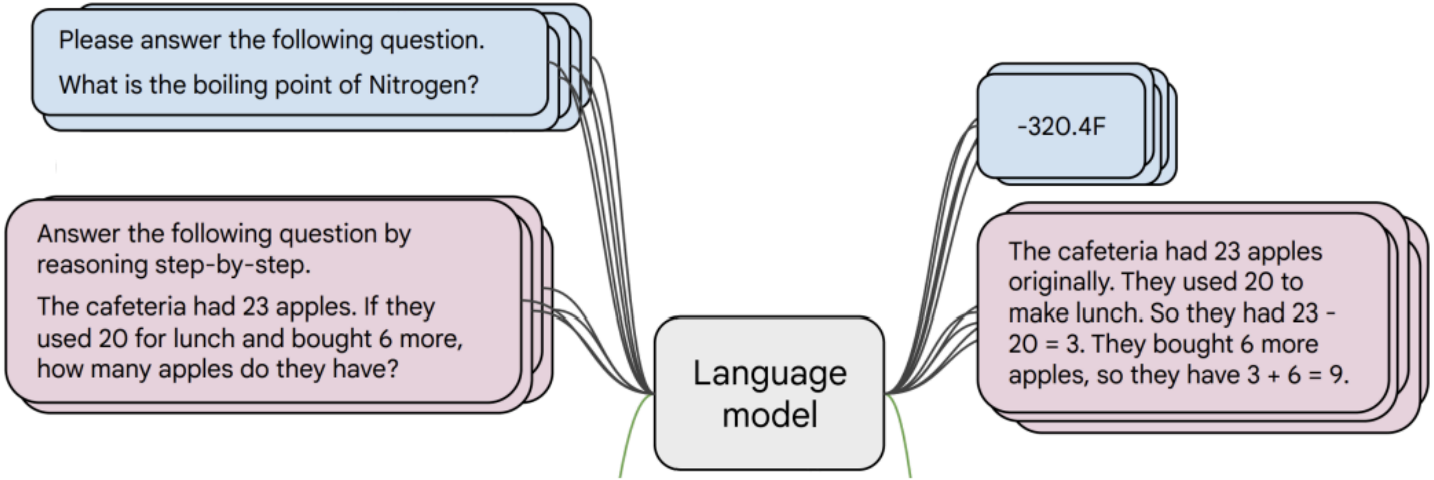
Chain of thoughts

- Some tasks seem too hard for even large LMs to learn through prompting alone. Especially tasks involving richer, multi-step reasoning.
- Zero-Shot COT: begin with `Let's think step by step...`



Instruction Finetuning

- **Collect examples** of (instruction, output) pairs across many tasks and finetune an LM



- Evaluate on **unseen tasks**

Instruction Finetuning

- Evaluation: MMLU: New benchmarks for measuring LM performance on 57 diverse knowledge intensive tasks
- A good way to boost small models' performance, like 80M with instruction finetuning can beat 3B original model.
- limitations:
  - Data is expensive
  - Mismatch between LM objective and human preferences
    - Language modeling penalizes all token-level mistakes equally, but some errors are worse than others.

Reinforcement Learning from Human Feedback

This is why it's called **"reinforcement learning"**: we **reinforce** good actions, increasing the chance they happen again.

- Giving us the update rule:

$$\theta_{t+1} := \theta_t + \alpha \frac{1}{m} \sum_{i=1}^{m} R(s_i)\, \nabla_{\theta_t} \log p_{\theta_t}(s_i)$$

If R is +++

Take gradient steps to maximize $p_\theta(s_i)$

If R is ---

Take steps to minimize $p_\theta(s_i)$

This is **heavily simplified**! There is a *lot* more needed to do RL w/ LMs. **Can you see any problems with this objective?**

Problem with Human Feedback

- Human-in-the-loop is expensive
  - Train an LM $RM\phi\ s$ to predict human preferences from an annotated dataset, then optimize for $RM\phi$ instead
- Human judgments are noisy and miscalibrated!
  - Ask for **pairwise comparisons**, which can be more reliable.

Problem with RLHF

- Reward hacking
  - Chatbots are rewarded to produce responses that seem authoritative and helpful, regardless of truth.

Reinforcement Learning from Human Feedback

- Finally, we have everything we need:
  - A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
  - A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
  - A method for optimizing LM parameters towards an arbitrary reward function.
- Now to do RLHF:
  - Initialize a copy of the model $p_\theta^{RL}(s)$, with parameters θ we would like to optimize
  - Optimize the following reward with RL:

$$R(s) = RM_\phi(s) - \underbrace{\beta \log \left(\frac{p_\theta^{RL}(s)}{p^{PT}(s)} \right)}$$

Pay a price when $p_\theta^{RL}(s) > p^{PT}(s)$

This is a penalty which prevents us from diverging too far from the pretrained model. In expectation, it is known as the **Kullback-Leibler** (**KL**) divergence between $p_\theta^{RL}(s)$ and $p^{PT}(s)$.

Natural Language Generation

Natural Language Generation

- Mechine translation
- Digital assistant
- Summarization
- Creative story
- Data to text
- Visual Description

> **Tips**
> Categorize by "open ended or not"

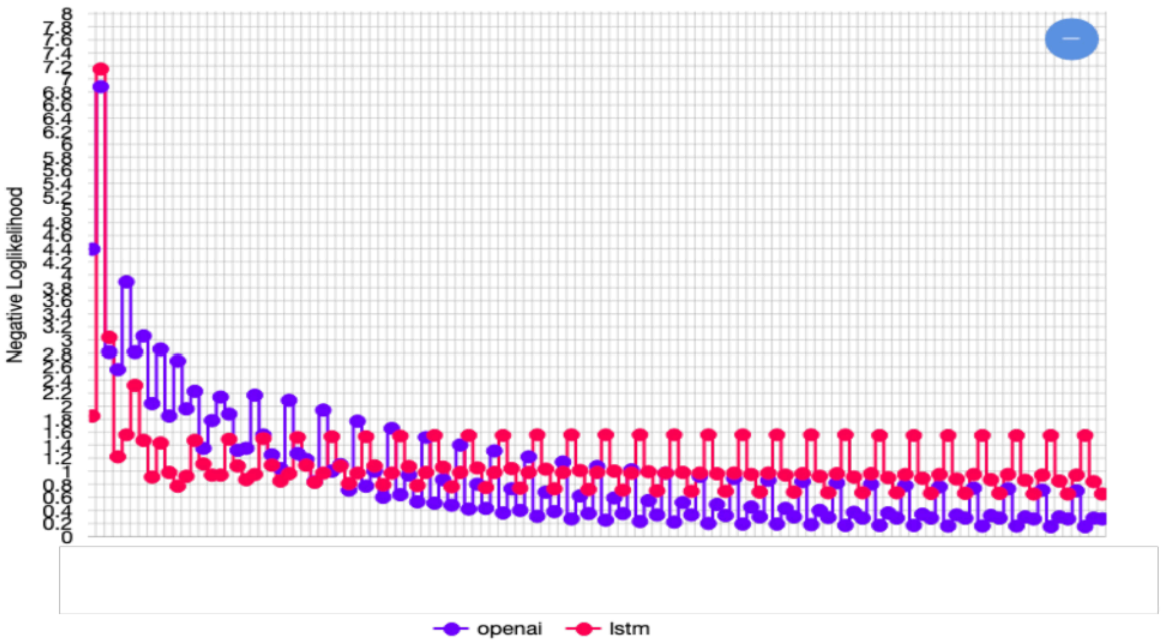
- For **non-open-ended** tasks (e.g., MT), we typically use a encoder-decoder system, where this autoregressive model serves as the decoder, and we'd have another bidirectional encoder for encoding the inputs.
- For **open-ended tasks** (e.g., story generation), this autoregressive generation model is often the only component.

Decoding: Most likely string

- Greedy Search: Always selects the highest prob word.
- Beam Search: Select from a set of words, perform better than greedy.

Repetition

I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired. I'm tired.



Scale doesn't solve this problem: even a 175 billion parameter LM still repeats when we decode for the most likely string.

(Holtzman et

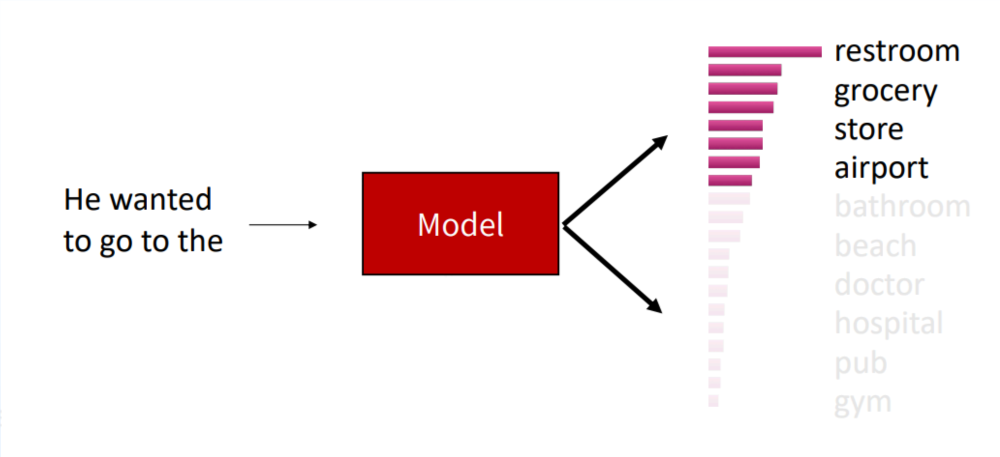
Repetition Solutions

- **n-gram detection**
  - It'll kill some things we really want to repeat.
- **Use a different training objective**
  - Unlikelihood objective (Welleck et al., 2020) penalize generation of already-seen tokens
  - Coverage loss (See et al., 2017) Prevents attention mechanism from attending to the same words
- **Use a different decoding objective**
  - Contrastive decoding (Li et al, 2022) searches for strings x that maximize `logprob_largeLM (x) - logprob_smallLM (x)`.

Sampling

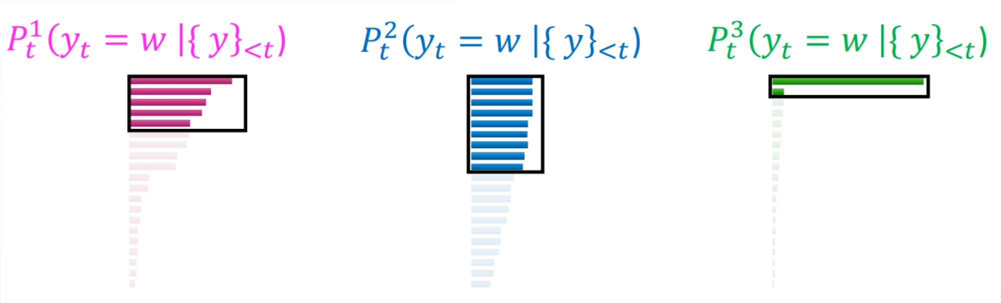
- Sample a token from a distribution of tokens.
- It’s random so we can get any tokens.
- Limitations: Heavy tailed distribution

Top-k sampling



- Only sample from top k token in probability distribution
- Increase `k` yields more diverse, but risky outputs
- Decrease `k` yields more safe, but guneric outputs
- Limitations: can't apply on all senario, may cut off too quick or too slow

Top-p sampling



- The probability distribution we sample are dynamic
- Sample from all tokens in the top p cumulative probability mass
  - so `k` depends on `p`, and makes it dynamic

Scaling: Tempture

- You can apply a temperature hyperparameter `t` to the softmax to rebalance `P`
- Raise the temperature `t` > 1: `P` becomes more uniform
  - More diverse output (probability is spread around vocab)
- Lower the temperature `t` < 1: `P` becomes more spiky
  - Less diverse output (probability is concentrated on top words)

Improving: Reranking

- Decode a bunch of sequence
- Rerank these sequences by a score, select the best sequence
    - Reranker can score a variety of properties
    - Can compose multiple scores

Exposure bias

- During training, model's inputs are gold texts from real, human-generated texts. But at generation time, model's inputs are previously-decoded tokens
- Solutions
  - **Scheduled sampling**: A small percentage to decode a token rather than fold texts, but may lead to **strange training objective**
  - **Dataset Aggregation**: Generate sequences from current model, and add these sequences as extra training data
  - **Retrieval Augmentation**: Learn to retrieve a sequence from an existing corpus of human-written prototypes
  - **Reinforcement Learning**: Cast text generation model as a Markov decision process

Evalutions of NLG

- Content overlap metrics: ROUGE, BLEU, etc.
  - No idea with semantic detection.
- Model-based metrics: BLEURT, MAUVE. etc.
  - Behavior is not interpretable
- Human judgements
  - Inconsistent

> **Tips**
> Best judgut is YOU!!! Look at your model generations. Don’t just rely on numbers!