Topics to study!

Language models
- n-gram [unigram, bigram, etc]
- perplexity

"Simple" neural LMs
- fixed-window neural LM
  - concat [word embs] \rightarrow \text{type/token}
  - linear layers \rightarrow h = Wx
  - weight matrix
  - f = Relu
- Softmax
  - used in output layer to get \( p(w_i|w_{i-1}) \)
  - used in self-attn

Transformer LMs
- self attention
  - query/key/value
  - multi-head self-attn
  - KV caching
  - masking during training
- position embeddings
  - learned vs. fixed
  - absolute vs. relative
- AliBi
  - add linear bias to attn matrix
  - no additive embedding
- RoPE
  - rotate Q, K
  - no additive embs

- Transformer configurations:
  - decoder vs. encoder vs. encoder-decoder
    - decoders predict the next word
      - masked self-attn
      - “prefix LM” where prompt is unmasked
    - encoders compute representations of the entire input text
      - unmasked self-attn
      - cannot generate text
    - encoder/decoder models separate the input/prompt from generated output
      - cross-attn uses queries from decoder and K/V from encoder
      - residual connections needed in decoder to include both encoder values
Training neural LMs:
- gradient descent
- backpropagation
  - chain rule + caching derivatives
- cross-entropy loss
- batching
- tokenization of inputs/outputs
  - BPE for subwords
- PyTorch implementation of Transformer and training loop (HW 2)

Transfer learning (pretraining $\Rightarrow$ fine-tuning)
- fine-tuning (SFT)
  - needs labeled dataset for a downstream task (e.g., sentiment)
    - much smaller than pretraining data
  - same loss as pretraining: cross-entropy on target outputs
- Parameter-efficient adaptation
  - LoRA/prompt tuning
  - reduce number of params that are modified vs. SFT
Decoding from LMs:
- how to generate text at test-time
- greedy vs. beam search
- nucleus sampling vs. ancestral sampling
  - effect of "p" in nucleus sampling

Aligning LMs:

3 stages of RLHF:

1. instruction tuning
   - SFT on instruction-following data
   - PLAN

2. reward model training on
   human pref judgments
   - Bradley-Terry pref. model

3. objective:
\[
\max_{\pi} \mathbb{E}_{x,y} \left[ r(x,y) - \beta D_{KL}(\pi(\cdot|x) \| \text{Pref}(\cdot|x)) \right]
\]

- requires rollouts from policy model \( \pi \)
  - "rollouts" are generations

\[\text{Reward of output } y \text{ given input } x \]
\[\text{KL penalty to prevent deviations from } \text{Pref} \]
given an instruction $x$, we sample a $y$ using a decoding algo
- optimize objective using PPO

- "Best-of-n" sampling
  - instead of stage 3, just sample multiple $y$'s from the LM and rerank them w/ reward model

- DPO (direct pref optimization)
  - no explicit reward model
  - no rollouts
  
  high-level derivation:

  1. express reward model in terms of optimal policy $\pi^*$

  \[ r(x, y) = \beta \log \frac{\pi^*(y|x)}{\pi_{ref}(y|x)} + \beta \log Z \]

  \[ \uparrow \text{normalizes inverteable to compute} \]

  2. plug into Bradley-Terry model

  \[ P(y_w > y_2 | x) = \frac{\exp(r(x, y_w))}{\exp(r(x, y_w)) + \exp(r(x, y_2))} \]
3. convert to loss fn

$$L_{\text{opo}} \left( \pi, \pi_{\text{ref}} \right) = -E \log \left( \beta \log \frac{\pi(y|w,x)}{\pi_{\text{ref}}(y|w,x)} \right)$$

- allows us to simply fine-tune over human prefs with modified loss fn

- LLM-based feedback
- constitutional AI

**Prompting**

- zero-shot vs. few-shot
  - few-shot uses "demonstrations" of $x,y$ pairs
  - chain of thought
  - retrieval

**Scaling**

- Chinchilla
  - importance of data size vs. model size vs. total compute (FLOPS)
  - small models don't exhibit properties such as few-shot learning/inspection following
Evaluation
- Perplexity
- BLEU / ROUGE (word matching)
- BLEURT / COMET
  - fine-tune encoder on human judgments
- LLM-based eval
  - GPT-Eval, Fact Score
- human eval (e.g. HW 2)