Large Language Models
Logistics

• **Office hours:** Mon 12 – 13h at BA2770
• **A2:** due **Mar 8, 2024** – *errata recap.*
• **A2 tutorials planned schedule:**
  • Feb 16: A2 tutorial – 1
  • Mar 1: A2 tutorial – 2 (ft. J. Watson)
  • Mar 8: A2 – Q/A and OH
• **A3:** release Mar 9, 2024
• **Final exam:** **April 25, 2024**

• Lecture feedback:
  • Anonymous
  • Please share any thoughts/suggestions

• **Questions?**
Lecture Plan (L7)

• LLM Trends and Implications
  • Trends, scaling laws, foundation models

• LLMs to Assistant Chatbots
  • Instruction fine-tuning
    • REINFORCE, RLHF

• Prompt Engineering
  • ICL, Chain-of-thought (CoT)

• Misc. (*time permitting*):
  • Benchmarks
  • Compute Requirements
  • PEFT: training strategies: LoRA
  • Quantization techniques: LLM.int8()
LLM TRENDS & IMPLICATIONS
LLM: the bigger is better trend
LLM: the bigger is better trend

- **Cons:**
  - Deep learning == Deep pockets? Democratisation of computing power
  - Social impact e.g. (environmental): "training BERT on GPU is roughly equivalent to a trans-American flight"¹

¹ S. Emma, A. Ganesh, and A. McCallum. "Energy and policy considerations for deep learning in NLP. (2019)" [arxiv]
LLM: the bigger is better trend

- **Cons:**
  - **Electricity consumption** comparison between countries and AI\(^1\)
  - More during the guest/invited speaker lectures on topic: Ethics

Scaling laws for LLMs

- Kaplan et al. (2020) does a systematic review of scaling laws for LLMs [1]

Language modelling performance (decreasing test loss is better), as the factors are scaled up

- **Three scale factors:**
  - Compute: the amount of compute $C$ used for training
  - Dataset size: the size of the dataset $D$
  - Model parameters: the number of model parameters $N$, excluding embeddings)

Scaling laws for LLMs

**Key Findings:**

Performance of (Transformer based) LLMs:

- Has power-law relationship with the three scale factors:
  - C, D, N (excluding embeddings).

- Depends most strongly on these scale factors
  - architectural hyperparameters (like depth, width) does not have much effect.

- Improves smoothly when factors (N, D) are scaled up in tandem.
  - Diminishing returns if either N or D bottlenecks the other. Roughly, an 8x model size increase should match 5x data size increase to avoid performance penalty.

Scaling laws for LLMs

Key Findings: Performance of (Transformer based) LLMs:

• **Transfer learning**: out-of-distribution generalization depends almost exclusively on the in-distribution (train set) validation loss performance that improves with the scaling factors.

• **Sample efficiency**: Large models are more sample-efficient than small models, reaching the same level of performance with fewer optimization steps, data points.

LLMs as Foundation Models

- **Homogenization**: (almost) all SOTA NLP LLM models are now adapted from one of a few foundation models (like BERT, BART, T5, etc.). [1]

  ![Diagram showing data and tasks](image)

  - Data from various modalities
  - Adoption to a wide range of downstream tasks

LLMs as Foundation Models

• Social Impact
  • Exacerbation of social inequalities.
  • Democratization: increased computation demands – power/capability concentrated to few corporations
  • Gap between industry models and community models are large.
  • Increasing proprietary moat and closed source nature.
  • Solution: government intervention?

• These are a small subset of ethical concerns
  • Detailed discussion in the ethics (guest) lecture by Steven Coyne
Is that it?

- Okay we got (larger and larger) LMs, foundation models yada yada ... but is that it?

- Can you talk about ChatGPT?
LLMS TO ASSISTANT CHATBOTS
From LMs to Assistants

Until now, we have seen:

• Neural language models and how to train them

• The transition and benefits of scaling:
  • BERT (<1B) -> T5 (~11B) -> GPT3 (175B) -> ...
  • Emergent behaviors with scaling\(^1\): ICL (In-context-learning) k-shots, gradient-free task completions

• Using PLMs: the pre-training then task specific fine-tuning paradigm

\(^1\) GPT3: Radford et al. "Language models are few-shot learners." (2020). [link]
From PLMs to Assistants

- However, these does not give us a general purpose, instruction-following chatbot (e.g., ChatGPT)

- **Solution**: instruction fine-tuning to **align** LLMs to follow human instructions

- **Reinforcement Learning from Human Feedback (RLHF)** pipeline
  - Many variants: RLAIF, DPO etc.

Why do we need Alignment?

Supervised fine-tuning objective (e.g., MLE) have intrinsic misalignments with human preferences.

- **Recall**: the typical LM (training/fine-tuning) objective is the cross entropy of the corpus $C$ given the model $M$:

  $$\mathcal{F} = H(C; M) = -\log_2 P_M(C) \quad \text{Minimize}$$

  $$\|C\|$$

- Such LM objective (maximizing log-likelihood) can be misaligned with the desired goal of high-quality output determined by humans.

- Can you think of any reasons why?
Limitations of the LM Objective

• No distinction between **important** (e.g., making up facts) and **unimportant errors** (e.g., selection of an alternate synonymous token)  
  “Toronto is a city in [India|Canada]” vs. “We all live in the same [house|home]”
  ▪ LM objective penalizes all (token-level) errors equally

• Incentivizes probability mass on all human demonstrations (i.e., supervision dataset)
  ▪ Unsafe, toxic, biased output generation
  ▪ Low-quality output generation

• Tasks entailing **open-ended language generation** does not have **one right answer**. Thus, **qualitative human feedback is a far better metric than heuristic objectives**

Optimizing using Human Feedback

- Stiennon et al.\cite{stiennon2020learning} show the efficacy on English Summarization task
- Base LM architecture: GPT-3 style Transformer decoders
- **Dataset:** TL;DR summarization dataset with ~3M posts from *reddit.com* across variety of topics (subreddits), and original (human) summaries

---

<table>
<thead>
<tr>
<th>Human written reference TL;DR</th>
<th>6.7B supervised model</th>
<th>6.7B human feedback model</th>
</tr>
</thead>
<tbody>
<tr>
<td>First date after 3 years in a relationship, going to the beach, terrified. What to bring with me, what to do?</td>
<td>Going on a date with a girl I met a few days ago, going to the beach. What should I bring, what should we do?</td>
<td>Going on my first ever date tomorrow, cycling to the beach. Would like some tips on what to do and bring. I’m a bit nervous and excited. Thanks!</td>
</tr>
</tbody>
</table>

```
[r/dating_advice] First date ever, going to the beach. Would like some tips
Hey Reddit! I (20M) would like some tips, because I have my first ever date tomorrow (although I’ve had a gf for 3 years, but no actual dating happened), and we’re going to the beach.

I met this girl, we have mutual friends, at a festival a few days ago. We didn’t kiss, but we talked, held hands, danced a bit. I asked her to go on a date with me, which was super hard as it is the first time I’ve asked this to anybody. What I mean to say is, it’s not like a standard *first* date because we already spent some time together.

I’m really nervous and excited. I’m going to pick her up tomorrow, we’re cycling to the beach which will take 30 minutes, and then what? I’m a bit scared. Should I bring something (the weather, although no rain and sunny, is not super so no swimming), should we do something. I’d like all the tips I can get. Thanks!
```

---

Prompt: Group the following 16 words into 4 groups of 4 of connected words\[1\]:

Bluebell, Blackberry, Watercress, Eskimo, Mud, Bluegrass, Rabbit, Doctor, Keystone, Hunt, Grand Canyon, Wensleydale, Hazel, Lavender, Silver, Craft

Sure, I'd be happy to help! Here are 4 groups of 4 semantically connected words from the list you provided:

Group 1: Bluebell, Blackberry, Watercress, Eskimo
Group 2: Mud, Bluegrass, Rabbit, Doctor
Group 3: Keystone, Hunt, Grand Canyon, Wensleydale
Group 4: Hazel, Lavender, Silver, Craft

I hope this helps! Let me know if you have any other questions.

---

Prompt: Group the following 16 words into 4 groups of 4 of semantically connected words:

Bluebell, Blackberry, Watercress, Eskimo, Mud, Bluegrass, Rabbit, Doctor, Keystone, Hunt, Grand Canyon, Wensleydale, Hazel, Lavender, Silver, Craft

- A picture **demo** is worth a thousand words slides!

- **Demo** Output differences between
  - Llama-2-7b\(^ \text{[1]} \) (*pretrained-only using LM objective*) and
  - Llama-2-7b-chat, Gemma-7b-it and GPT-4 (*instruction fine-tuned LLMs*)

- Try it yourself:
  - Gemma-7b [https://huggingface.co/google/gemma-7b](https://huggingface.co/google/gemma-7b)
  - Gemma-7b-chat [https://huggingface.co/google/gemma-7b-it](https://huggingface.co/google/gemma-7b-it)

---

Pretrained LLMs + Instruction Finetuning

• Compute capabilities of pre-training such massive LLMs using large-scale data is beyond scope for most individuals/corporations.

• **Beginning 2023**, Meta pioneered the trend of capable companies releasing / open-sourcing trained model weights*

• The Llama series of LLMs\(^\text{[1,2]}\) are a collection of foundation LMs with varying granularities (sizes, fine-tuning spectrum)

  • **Properties** of the **Llama** models\(^\text{[1]}\) include:
    • Sizes: parameters range from 7B to 65B
    • Trained on trillions of tokens, using **publicly available datasets only**
    • Llama-1-13B outperformed GPT-3 (175B) on most benchmarks
    • Llama-1-65B competitive with (then) SoTA models like Chinchilla-70B, PaLM-540B.

---


* Llama-1 model weights were leaked publicly via torrents before the official release.
Pretrained LLMs + Instruction Finetuning

- Since then, taking a base pre-trained LLM (with provided weights) then instruction-finetuning has seen mass, wide community adoption across: tasks, libraries, datasets and languages[1]

[1] https://huggingface.co/models
Instruction Finetuning – How?

• We want to align (i.e., optimize) a LM using human preferences

• But we can not have human in-the-loop for qualitative assessments during training as it’d be infeasible (expensive, slow)

• Solution: train a reward model that mimics human preferences by emitting a scalar reward, ranking pair-wise (or more) generated completions by the LM

• Next: how do we update our LM’s parameters using these scalar reward values and pair-wise rankings? N.B. the reward function is non-differentiable w.r.t. to the LM parameters, so we can’t apply SGD.

• Solution: Use RL policy gradient update methods like REINFORCE, PPO using the RLHF pipeline
Instruction Finetuning – RLHF

- LLM alignment using Reinforcement Learning from Human Feedback\(^2\) (RLHF) usually entails three steps:

  - **Step 1**: Supervised fine-tuning (SFT) using high-quality human demonstrations dataset
  - **Step 2**: Reward model (RM) training using human-ranked preferences dataset
  - **Step 3**: Optimize the LM from step 1 with RM (step 2) using RL (specifically, PPO\(^1\) algorithm).

RLHF Optimization – Human Preferences

**Step 1 - SFT**: Supervised fine-tuning (SFT) using high-quality human demonstrations dataset

**Step 2 – Preference Sampling & Reward Learning**: Train Reward model (RM) mimicking human preferences

- Start with LM-SFT baseline (from step 1)
- Add randomly initialized head that outputs a scalar (reward) value

\[
\text{loss}(\theta) := \mathbb{E}_{(x, y_w, y_l) \sim D_{RM}} \left[ \log(\sigma(r_\theta(x, y_w) - r_\theta(x, y_l))) \right]
\]

Where \( r_\theta(x, y_i) \) is the scalar output of \( RM_\theta \) for prompt \( x \) and completion \( y_i \)

Step 2 – Preference Sampling & Reward Learning

Preference Sampling:

- \( D_{RM} = \{ x^{(i)}, y_w^{(i)}, y_l^{(i)} \}^N \)

- SFT model is prompted with prompts \( x \) to produce pairs of completions \((y_1, y_2)\):
  \[
  (y_1, y_2) \sim \pi^{SFT}(y | x)
  \]

- Human labelers determines a winning choice: \( y_w > y_l | x \)
  - Assumption: underlying latent (human) reward model: \( r^*(y, x) \)
RLHF Optimization – Human Preferences

- Bradley Terry\textsuperscript{[1]} model stipulates human preferences distribution \( p^* \) as:

\[
p^*(y_1 > y_2 \mid x) = \frac{\exp (r^*(x, y_1))}{\exp (r^*(x, y_1)) + \exp (r^*(x, y_2))}
\]

- Assuming \( D_{RM} \) sampled from \( p^* \) we can parameterize a reward model: \( r_\theta(x, y) \) and estimate the parameters via maximum likelihood by framing the problem as a binary classification with NLL loss:

\[
\mathcal{L}_R(r_\theta, D_{RM}) = -\mathbb{E}_{(x, y_w, y_l) \sim D_{RM}} \left[ \log (\sigma(r_\theta(x, y_w) - r_\theta(x, y_l))) \right]
\]

where \( r_\theta(x, y_l) \) is the scalar output of \( RM_\theta \) for prompt \( x \) and completion \( y_l \) and \( \sigma \) is the logistic function

- To ensure a reward function with lower variance, normalize the rewards

\[
\mathbb{E}_{(x, y) \sim D} [ r_\theta(x, y) ] = 0 \text{ for all } x
\]

RLHF Optimization – Human Preferences

Step 3 – RL Optimization: Optimize the LM from step 1 with RM (step 2) using RL

• For a completion (or, response) $\hat{y}$ to prompt $x$, we want to update LM-policy $\pi_{LM}$ parameters $\phi$ to maximize:

$$\mathbb{E}_{\hat{y} \sim \pi_{\phi}(x)}[R_{\theta}(x, \hat{y})]$$

• SGD updates (like below) does not work because our reward function $R_{\theta}(.)$ is non-differentiable w.r.t. our model parameters $\pi_{LM}$

$$\phi_{t+1} \leftarrow \phi_t + \alpha \nabla_{\phi_t} \mathbb{E}_{\hat{y} \sim \pi_{\phi_t}(x)}[R_{\theta}(x, \hat{y})]$$

• Thus, we resort to policy-gradient methods in RL like REINFORCE$^{[1]}$, PPO$^{[2]}$ to estimate and optimize this objective

Optimization: REINFORCE - I

• Drilling down on the REINFORCE (highly simplified) mechanism

$$\nabla_\phi \mathbb{E}_{\hat{y} \sim p_\phi(y)}[\mathcal{R}(\hat{y})] = \nabla_\phi \sum_y \mathcal{R}(y) p_\phi(y) = \sum_y \mathcal{R}(y) \nabla_\phi p_\phi(y)$$

  - defn. of Expectation
  - linearity of gradient

• Reformulate using the log-derivative trick

$$\nabla_\phi \log(p_\phi(y)) = \frac{\nabla_\phi p_\phi(y)}{p_\phi(y)} \implies \nabla_\phi p_\phi(y) = p_\phi(y) \nabla_\phi \log(p_\phi(y))$$

• Plug back into the first equation:

$$\sum_y \mathcal{R}(y) \nabla_\phi p_\phi(y) = \sum_y p_\phi(y) \mathcal{R}(y) \nabla_\phi \log(p_\phi(y)) = \mathbb{E}_{\hat{y} \sim p_\phi(y)}[\mathcal{R}(\hat{y}) \nabla_\phi \log(p_\phi(\hat{y}))]$$

Optimization: REINFORCE - II

- We can approximate the objective using Monte Carlo sampling with the gradient pushed inside the expectation operator

\[ \nabla_{\phi} \mathbb{E}_{\hat{y} \sim p_{\phi}(y)}[R(\hat{y})] = \mathbb{E}_{\hat{y} \sim p_{\phi}(y)} [R(\hat{y}) \nabla_{\phi} \log(p_{\phi}(\hat{y}))] \]

\[ \approx \frac{1}{N} \sum_{i=1}^{N} R(\hat{y}_i) \nabla_{\phi} \log(p_{\phi}(\hat{y}_i)) \]

- Now, we can update our objective using completion samples \( y \) as:

\[ \phi_{t+1} \leftarrow \phi_t + \alpha \frac{1}{N} \sum_{i=1}^{N} R(y_i) \nabla_{\phi_t} \log(p_{\phi_t}(y_i)) \]

**Intuition**

If \( R(y_i) \) is positive then take steps to update weight parameters \( \phi \) to maximize \( p_{\phi}(y_i) \). If \( R(y_i) \) is negative then update weight parameters \( \phi \) to minimize \( p_{\phi}(y_i) \)

RLHF – Step 3 RL Optimization

- **Goal**: update parameters $\phi$ of our LM $\pi^{SFT}$ from **step 1** with objective:

\[
\max_{\pi_{\phi}} \mathbb{E}_{x \sim D, \hat{y} \sim \pi_{\phi}(y|x)}[R_{\theta}(\hat{y})] - \beta \mathbb{D}_{KL} [\pi_{\phi}(\hat{y} | x) \parallel \pi_{ref}(\hat{y} | x)]
\]

where $\beta$ controls **KL-divergence operator** to regulate the deviation of trained policy from a base reference policy, usually $\pi^{SFT}$

- In practice, $\pi_{\phi}$ is also initialized with $\pi^{SFT}$
- Due to discrete nature of language generation, this objective is non-differentiable, and typically optimized with RL
- Specifically, construct this **reward function** and maximize using PPO:

\[
r(x, y) = r_{\theta}(x, y) - \beta (\log(\pi_{\phi}(y | x)) - \log(\pi_{ref}(y | x)))
\]

Variants – RLAIF

- Reinforcement Learning from AI Feedback (RLAIF):
  - Train RM using (AI) Feedback from other off-the-shelf LLMs
  - Then, train LM-SFT using this RM as usual in RLHF

PROMPTING LLMS
Prompt Engineering

• Now that we have seen LM pretraining, fine-tuning (SFT) and instruction fine-tuning (RLHF), let’s examine how prompting works in modern LLMs in practice

• The science art of LLM prompting

• Prompt design is imperative for obtaining good results from an LLM / foundation model

• From In-context Learning (ICL) paradigm we saw gradient free approaches like:
  • Zero-shot: Asking LLM to perform task with no previous example
  • Few-shot: Providing examples as context to the LLM before giving the task
Prompt Engineering

**Zero-Shot**

Q: What is the capital of France?

A: “Paris”

Lower token count. Allows larger window for context

**Few-Shot**

Q: What is the capital of Spain?

A: {'answer': 'Madrid'}

Q: What is the capital of Italy?

A: {'answer': 'Rome'}

Q: What is the capital of France?

A: {'answer': 'Paris'}

Better **alignment** (e.g., format) of responses. Better **accuracy** on complex questions
Prompt Engineering

Investment: Data, Compute & Time

Prompt Engineering  Task Accuracy  Full-parameter Fine-tuning

• Techniques
  • ICL: zero-shot, few-shot
  • Chain-of-thought (CoT) reasoning
  • System prompting ...

• Method: Prompt templates

• Training data:
  • Single-digit completion examples

• Advantages:
  • Minimal, simple input samples
  • Online (inference time) gradient free

• Techniques
  • SFT, RLHF, RLAIF, DPO ...

• Method: Tune LLM weights

• Training data:
  • Thousands of samples & complex use cases

• Advantages:
  • Compatibility: traditional approach
  • Robust, better accuracy in challenging domains
Prompt Engineering - Basics

- Typical LLM I/O structure during inference:
  - Discourse, conversations, messages: array of structured message objects to send to the LLM. Provides context or history from which to continue
  - Roles:
    - **system**: provide core instruction to the LLM
    - **user**: 'human' (could be another 'AI' chatbot too)
    - **Assistant**: role of the LLM, to generate a response

```
POST https://api.openai.com/v1/chat/completions

`

```
{
  
  "model": "gpt-4",
  "messages": [
    
    {
      "role": "system",
      "content": "You are a stock market analyst who predicts the market movement from today's news and world events."
    },
    
    {
      "role": "assistant",
      "content": "Examine the given market information and news headlines data on 2010-01-08 to forecast whether the $SPY index will rise, fall, or remain unchanged. If you think the movement will be less than 0.05%, then return 'Neutral'. Respond with Rise, Fall, or Neutral and your reasoning in a new paragraph"
    },
    
    {
      "role": "user",
      "content": "Context:
```
• Typical LLM hyperparameters during inference:
  
  o **top_p**: when decoding text, samples from the top p percentage of most likely tokens. In other words, curtail list of generated tokens beyond ‘p’
  o **top_k**: same idea as ‘top_p’ but for ‘k’ most likely tokens (instead of percentage)
  o **repetition_penalty**: parameter controlling how to penalize the generation of the same text token
  o **temperature**: randomness of choosing a token (from 'p'). '0' means least random
  o **max_seq_len**: the size of input context window, usually depends on the LLM.

```
...
    "temperature": 0.05,
    "max_tokens": 256,
    "top_p": 1,
    "frequency_penalty": 0,
    "presence_penalty": 0
}
```
Prompting Techniques

• Active research area with myriad techniques
• Simple stylization changes or decomposition of instructions can change the generated response

• Examples:
  • **Detailed, explicit instructions** better than open-ended prompts:
    o Stylization:
      ▪ *I am a CS student using LLMs for solving assignment*
      ▪ *Give your answer like explaining the topic to a 5-year old*
  
  • Explicit (step-by-step) instructions:
    o *Use bullet points, only use academic papers, return answer in python code* etc.
Promoting Techniques – Chain of Thought

- Chain of Thought (CoT) prompting\[^1\]
  - Decomposing instruction into series of intermediate reasoning steps

PRACTICAL TIDBITS & EVALUATION
Evaluation: LLMs Benchmarks

- **Language foundation models** and their typical evaluation **tasks**:

<table>
<thead>
<tr>
<th>Year</th>
<th>Model Name</th>
<th>Model Arch.</th>
<th>Oriented Tasks</th>
<th>Parameters</th>
<th>Pre-training Method</th>
<th>Pre-training Datasets</th>
<th>Testing Datasets</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>Token-CLS,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Fill-Mask, QA,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td></td>
<td>Translation,</td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2019</td>
<td>DistilBERT [314]</td>
<td>Encoder-Only</td>
<td>Same as BERT</td>
<td>66M</td>
<td>Self-Supervised, Distillation</td>
<td>Same as BERT</td>
<td>SquAD, IMDb</td>
</tr>
<tr>
<td></td>
<td>RoBERTa [238]</td>
<td>Encoder-Only</td>
<td>Same as BERT</td>
<td>125-355M</td>
<td>Self-Supervised</td>
<td>Bookeorpus, CC-news, Opennewest, Stories</td>
<td>GLUE, SquAD, RACE</td>
</tr>
<tr>
<td>2019</td>
<td>BART [197]</td>
<td>Encoder-Decoder</td>
<td>Same as BERT</td>
<td>140-400M</td>
<td>Self-Supervised</td>
<td>BooksCorpus, CC-news, Opennewest, Stories</td>
<td>GLUE, CNN/DM, SquAD, SFLUE, EnDe, EnFr, EnRO</td>
</tr>
<tr>
<td>2019</td>
<td>T5 [300]</td>
<td>Encoder-Decoder</td>
<td>Same as BERT</td>
<td>60M-11B</td>
<td>Self-Supervised</td>
<td>Colossal Clean Crawled Corpus</td>
<td>GLUE, CNN/DM, SquAD, SFLUE, EnDe, EnFr, EnRO</td>
</tr>
<tr>
<td>2019</td>
<td>GPT-2 [299]</td>
<td>Decoder-Only</td>
<td>Same as BERT</td>
<td>1.5B</td>
<td>Self-Supervised</td>
<td>WebText</td>
<td>SQuAD, CoQA, WMT, CNN/Daily Mail</td>
</tr>
<tr>
<td>2021</td>
<td>GLM [90]</td>
<td>Decoder-Only</td>
<td>Same as BERT</td>
<td>110M-130B</td>
<td>Unsupervised</td>
<td>BooksCorpus, English Wikipedia</td>
<td>SuperGLUE</td>
</tr>
<tr>
<td>2022</td>
<td>PaLM [67]</td>
<td>Decoder-Only</td>
<td>Same as BERT</td>
<td>54B</td>
<td>Unsupervised</td>
<td>Mixture of 780B Text Source code</td>
<td>English NLP, IRF-bench, Reasoning, Code, etc.</td>
</tr>
<tr>
<td>2021</td>
<td>HaBERT [140]</td>
<td>Encoder-Decoder</td>
<td>Auto Speech Recognition</td>
<td>281M-2.8B</td>
<td>Self-Supervised</td>
<td>Libri-Light, LibriSpeech</td>
<td>LibriSpeech, TIMT</td>
</tr>
<tr>
<td>2023</td>
<td>GPT-4 [273]</td>
<td>Close-Sourced</td>
<td>Text Generation</td>
<td>Close-Sourced</td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>2023</td>
<td>Claude2</td>
<td></td>
<td></td>
<td></td>
<td>Close-Sourced</td>
<td></td>
<td></td>
</tr>
<tr>
<td>2023</td>
<td>PaLM2</td>
<td></td>
<td></td>
<td></td>
<td>Close-Sourced</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Evaluation: LLMs Benchmarks

• The **GLUE** and **SuperGLUE** benchmarks for evaluating NLP LM tasks circa 2019-21

• LM benchmarking scene has rapidly evolved in conjunction with capabilities since then

• Recent benchmarks include:
  • BIG-Bench[^1^]: 200+ tasks with dynamic additions of newer tasks
  • MMLU[^2^]: Evaluates LMs on tasks across 57 diverse knowledge bases
  • HELM[^3^]:
  • GlobalBench[^4^]:

Large models are not easily accessible

<table>
<thead>
<tr>
<th>Model</th>
<th>Inference memory</th>
<th>Fine-tuning memory*</th>
</tr>
</thead>
<tbody>
<tr>
<td>T5-11B</td>
<td>22 GB</td>
<td>176 GB</td>
</tr>
<tr>
<td>LLaMA2-33B</td>
<td>66 GB</td>
<td>396 GB</td>
</tr>
<tr>
<td>LLaMA2-70B</td>
<td>140 GB</td>
<td>840 GB</td>
</tr>
</tbody>
</table>

*Default or typical values. Fine-tuning memory depends on the type of optimizers used

Raffel et al., 2020, T5, Zhang et al., 2022, OPT., BigScience, 2022, BLOOM.
LLMs Memory Footprint

• GPU Memory requirements for LLMs:
  • Inference vs. Training/Fine-tuning (Tr/ft)

• For inference, at full precision (float32), each parameter is 32bits or 4 bytes (b).
  • Thus, a 7B param. model requires \( 28 \text{ GB} \) GPU memory

• Training/fine-tuning requires more memory as optimizer weights (parameters + gradient) need to be stored.
  • E.g., Adam\(^1\) or AdamW (stores the second moment of gradients) requires 16b per trainable param.
  • 16B per trainable param. That’s 7B * 16b = 112GB GPU RAM

[Aside] PEFT

• **PEFT** or **Parameter Efficient Fine-Tuning** is an umbrella term for methodologies (and/or libraries) for efficient adaptation of LLMs to downstream tasks within computation cost/budget.

• PEFT methods are typically evaluated on these 5 metrics:
  1. Storage efficiency
  2. Memory efficiency
  3. Computation efficiency,
  4. Accuracy, and
  5. Inference overhead
LLMs: Summary & Conclusion

• Full coverage of all pertinent areas would make a course of its own. We deep-dived into selected topics only

• Ad vents in LLM research has truly put AI in global limelight

• Research in LLM can be exhausting is incredibly fast-paced with global, immediate impact

• Understand the many limitations of LLMs (e.g., hallucination, creative tasks) and the nature of each fast-moving components:
  • Representation learning domain
  • Component improvements: e.g. attention mechanisms, alternate architectures
  • Improving training/fine-tuning methodologies at scale:
    • Parallelism, PEFT: quantization techniques etc.