Recap
Transformers
Multi-head self-attention

A
Q
K
V
Layer $p$

$M_H$

$M_V$

optics advanced who Strickland awards committee Nobel

Nobel committee awards Strickland who advanced optics

[Vaswani et al. 2017]
Multi-head self-attention

Layer 1

Layer \( p \)

Layer \( J \)

Multi-head self-attention + feed forward

Nobel committee awards Strickland who advanced optics

[Slides by Emma Strubell!]

[Vaswani et al. 2017]
Next English word

decoder

encoder

completely info from French & English

English sentence representation

English words generated so far

Whole French sentence

Classification
Position embeddings are *added* to each word embedding. Otherwise, since we have no recurrence, our model is unaware of the position of a word in the sequence!
Residual connections, which mean that we add the input to a particular block to its output, help improve gradient flow.
A feed-forward layer on top of the attention-weighted averaged value vectors allows us to add more parameters / nonlinearity.
We stack as many of these *Transformer* blocks on top of each other as we can (bigger models are generally better given enough data!).
Moving onto the decoder, which takes in English sequences that have been shifted to the right (e.g., `<START> schools opened their`)
We first have an instance of masked self attention. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.
We first have an instance of masked self-attention. Since the decoder is responsible for predicting the English words, we need to apply masking as we saw before.

Why don’t we do masked self-attention in the encoder?
Now, we have cross attention, which connects the decoder to the encoder by enabling it to attend over the encoder’s final hidden states.
After stacking a bunch of these decoder blocks, we finally have our familiar Softmax layer to predict the next English word.
The Deep Learning Pipeline
Deep learning models can be run in two modes:

- **Training**: update a model’s weights to fit new data. This is *supervised learning* because it requires input/output pairs (labeled data).

- **Inference**: run data through a model to make predictions. This requires only input data. It does not change the model weights.
Transfer Learning

Contemporary machine learning often involves multiple stages of training:

- **Pre-training**: train a large model that will be used by many downstream applications. *Called a foundation model in Bommasani et al. 2021*

- **Fine-tuning**: adapting a pre-trained model to a new task or dataset by training it on new data, starting from existing weights.
Transfer Learning

Contemporary machine learning models may also build upon other models by *freezing the weights of the original model* and taking some of its components as input.

For instance, the *weights of attention heads* may be reused as embeddings to be fed in as input to a downstream model.

This is called *feature extraction*.

*This is what we did in the recipe classifier: we took attention weights from RoBERTa to use as features in our classifier!*
Pretraining: learn good representations via an unlabeled task.

Representation learning: extract attention features and use as input features to another model.

Finetuning: train some more on in-domain data or separate labeled task.

Prompt engineering: craft prompts that disguise task of interest as a language generation problem.

Google Search
Classification
Image Captioning
Story generation

Few-shot learning
Q/A
Coreference resolution
Translation
Style Transfer

Few-shot/Zero-shot learning
Code explanation
Summarization
Poem generation
Reinforcement Learning From Human Feedback
1. Pretrain your large language model

2. Train a reward model from human feedback:

   text $\rightarrow$ Reward Model $\rightarrow$ scalar

3. Finetune (some of) your large language model using the reward model, but with a policy shift constraint
Prompts Dataset

Sample many prompts

Initial Language Model

Train on \{sample, reward\} pairs

Reward (Preference) Model

Outputs are ranked (relative, ELO, etc.)

Generated text
ChatGPT
Proximal Policy Optimization

Step 1
Collect demonstration data and train a supervised policy.

A prompt is sampled from our prompt dataset.

A labeler demonstrates the desired output behavior.

This data is used to fine-tune GPT-3.5 with supervised learning.

Step 2
Collect comparison data and train a reward model.

A prompt and several model outputs are sampled.

A labeler ranks the outputs from best to worst.

This data is used to train our reward model.

Step 3
Optimize a policy against the reward model using the PPO reinforcement learning algorithm.

A new prompt is sampled from the dataset.

The PPO model is initialized from the supervised policy.

The policy generates an output.

The reward model calculates a reward for the output.

The reward is used to update the policy using PPO.

openai.com/blog/chatgpt
ChatGPT

GPT-3.5 → PPO → ChatGPT

Reward Model

MYSTERY DATA

public interaction
ChatGPT

GPT-3.5 \rightarrow PPO \rightarrow ChatGPT

Reward Model

MYSTERY DATA

who creates this data?

everyone on the Internet
who creates the ratings? underpaid crowd workers in the Global South

MYSTERY DATA

Reward Model

GPT-3.5 → PPO → ChatGPT

public interaction
"OpenAI sent tens of thousands of snippets of text to an outsourcing firm in Kenya, beginning in November 2021. Much of that text appeared to have been pulled from the darkest recesses of the internet. Some of it described situations in graphic detail like child sexual abuse, bestiality, murder, suicide, torture, self harm, and incest."

"OpenAI’s outsourcing partner in Kenya was Sama, a San Francisco-based firm. Sama markets itself as an “ethical AI” company."

"The data labelers employed by Sama on behalf of OpenAI were paid a take-home wage of between around $1.32 and $2 per hour depending on seniority and performance."
ChatGPT

Reward Model

who creates this data? all of us (anyone who uses ChatGPT)

MYSTERY DATA

public interaction

GPT
Prompt Engineering
One idea is to make the model generate reasoning before an answer. This guarantees that the answer is conditioned on the reasoning. Some people think this could improve the quality of the answer. However, other work has shown that the answer is not always consistent with the given reasoning.
Question: Tom and Elizabeth have a competition to climb a hill. Elizabeth takes 30 minutes to climb the hill. Tom takes four times as long as Elizabeth does to climb the hill. How many hours does it take Tom to climb up the hill?
Answer: It takes Tom \(30 \times 4 = 120\) minutes to climb the hill.
It takes Tom \(120/60 = 2\) hours to climb the hill.
So the answer is 2.

===

Question: Jack is a soccer player. He needs to buy two pairs of socks and a pair of soccer shoes. Each pair of socks cost $9.50, and the shoes cost $92. Jack has $40. How much more money does Jack need?
Answer: The total cost of two pairs of socks is \(9.50 \times 2 = 19\).
The total cost of the socks and the shoes is \(19 + 92 = 111\).
Jack need \(111 - 40 = 71\) more.
So the answer is 71.

===

Question: Marty has 100 centimeters of ribbon that he must cut into 4 equal parts. Each of the cut parts must be divided into 5 equal parts. How long will each final cut be?
What Are Prompts Really Doing?

Results from Webson & Pavlick (2022)
Solving and Generating NPR Sunday Puzzles with Large Language Models

Jingmiao Zhao and Carolyn Jane Anderson
Computer Science Department
Wellesley College
Wellesley, MA 02482 USA
carolyn.anderson@wellesley.edu

Abstract

We explore the ability of large language models to solve and generate puzzles from the NPR Sunday Puzzle game show using PUZLEQA, a dataset comprising 15 years of on-air puzzles. We evaluate four large language models using PUZLEQA, in both multiple choice and free response formats, and explore two prompt engineering techniques to improve free response performance: chain-of-thought reasoning and prompt summarization. We find that state-of-the-art large language models can solve many PUZLEQA puzzles: the best model, GPT-3.5, achieves 50.2% loose accuracy. However, in our few-shot puzzle generation experiment, we find no evidence that models can generate puzzles: GPT-3.5 generates puzzles with answers that do not conform to the generated rules. Puzzle generation remains a challenging task for future work.

Puzzle Description: Today’s puzzle involves “consonyms,” which are words that have the same consonants in the same order but with different vowels. Every answer is the name of a country.
Question: MINGLE
Answer: MONGOLIA

Figure 1: NPR Sunday Puzzle from March 12, 2023

Benchmarking AI through Games

Our work continues the tradition of evaluating AI progress through puzzles and games (Ferrucci 2012; Rodriguez et al. 2021; Rozner, Potts, and Mahowald 2021; Sobieszek and Price 2022). Contemporary LLMs have demonstrated strong performance on a wide variety of language tasks, including
Does CoT Help?

Maybe not?
Continuous Prompting

Humans write discrete prompts, which are then turned into text embeddings.

What if we tried to directly learn good text embeddings?
What Makes a Good Prompt?

+ Giving multiple examples
+ Specifying the answer format
+ Order matters!
+ Explanation seems key
+ Word puzzles seem harder than trivia
The Costs of Deep Learning
Models keep getting larger
Models keep getting larger

2022-2023:

**PaLM (Google):** 540B params, 118 layers, 18432 d_model, 780 billion training tokens **Model not available**

**ChatGPT (OpenAI):** Params, layers, dimensionality, training data size unknown **Model available only through blackbox API**

**LLaMa (Meta):** 65B params, 80 layers, 8192 d_model, 1.4 trillion tokens of training data **Model parameters publicly available!**

**GPT4 (OpenAI):** Params, layers, dimensionality, training data size unknown **Model available only through blackbox API**

**Bard (Google):** Params, layers, dimensionality, training data size unknown **Model available only through blackbox API**
These models are really expensive!

Megatron (530 billion parameters), Microsoft's GPT-3 competitor, cost around $100 million to train.
These models are really expensive!

By contrast: children are exposed to < 100 million words in their critical language acquisition period.

<table>
<thead>
<tr>
<th>Model</th>
<th>Size (# Parameters)</th>
<th>Training Tokens</th>
</tr>
</thead>
<tbody>
<tr>
<td>LaMDA (Thoppilan et al., 2022)</td>
<td>137 Billion</td>
<td>168 Billion</td>
</tr>
<tr>
<td>GPT-3 (Brown et al., 2020)</td>
<td>175 Billion</td>
<td>300 Billion</td>
</tr>
<tr>
<td>Jurassic (Lieber et al., 2021)</td>
<td>178 Billion</td>
<td>300 Billion</td>
</tr>
<tr>
<td>Gopher (Rae et al., 2021)</td>
<td>280 Billion</td>
<td>300 Billion</td>
</tr>
<tr>
<td>MT-NLG 530B (Smith et al., 2022)</td>
<td>530 Billion</td>
<td>270 Billion</td>
</tr>
<tr>
<td>Pathways (Chowdhery et al. 2022)</td>
<td>540 Billion</td>
<td>780 Billion</td>
</tr>
</tbody>
</table>

Pathways (Chowdhery et al. 2022)
Baby LLM Project

Alex Warstadt is giving a talk about language acquisition in LLMs versus humans in my NLP class (9:55am) on Nov. 28th!
These models are really expensive!

<table>
<thead>
<tr>
<th>Consumption</th>
<th>CO$_2$e (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>Air travel, 1 person, NY$\leftrightarrow$SF</td>
<td>1984</td>
</tr>
<tr>
<td>Human life, avg, 1 year</td>
<td>11,023</td>
</tr>
<tr>
<td>American life, avg, 1 year</td>
<td>36,156</td>
</tr>
<tr>
<td>Car, avg incl. fuel, 1 lifetime</td>
<td>126,000</td>
</tr>
</tbody>
</table>

**Training one model (GPU)**

<table>
<thead>
<tr>
<th>Model Description</th>
<th>CO$_2$e (lbs)</th>
</tr>
</thead>
<tbody>
<tr>
<td>NLP pipeline (parsing, SRL)</td>
<td>39</td>
</tr>
<tr>
<td>w/ tuning &amp; experiments</td>
<td>78,468</td>
</tr>
<tr>
<td>Transformer (big)</td>
<td>192</td>
</tr>
<tr>
<td>w/ neural arch. search</td>
<td>626,155</td>
</tr>
</tbody>
</table>

Table 1: Estimated CO$_2$ emissions from training common NLP models, compared to familiar consumption.$^{1}$

Strubell, Ganesh, & McCallum (2019)
These models are really expensive!

BERT-L (340 million parameters) had a carbon footprint equivalent to a trans-American flight.

And remember:

Microsoft Megatron has 530 billion parameters...

Google Pathways has 540 billion parameters...
Models keep getting larger

Log scale!

Total Compute Used During Training

- BERT-Base
- BERT-Large
- RoBERTa-Base
- RoBERTa-Large
- T5-Small
- T5-Base
- T5-Large
- T5-3B
- T5-11B
- GPT-3 Small
- GPT-3 Medium
- GPT-3 Large
- GPT-3 XL
- GPT-3 2.7B
- GPT-3 6.7B
- GPT-3 13B
- GPT-3 175B