

What We Know About ChatGPT

Robin Jia
USC CSCI 467, Spring 2024
April 23, 2024

ChatGPT

R What machine learning model should I use to determine if an image contains a leopard or a tiger?



For image classification tasks such as this one, a convolutional neural network (CNN) is typically used. CNNs are particularly good at detecting patterns in visual data and are widely used for image classification tasks.

To train a CNN for your specific task, you would first need to gather a dataset of images containing leopards, tigers, and possibly other animals. This dataset would be split into training, validation, and test sets, with the training set used to train the model, the validation set used to tune the model's hyperparameters, and the test set used to evaluate the model's performance on unseen data.

There are many pre-trained CNN models that you can use as a starting point, such as ResNet, VGG, or Inception. You can also choose to train your own model from scratch using a framework such as TensorFlow or PyTorch.

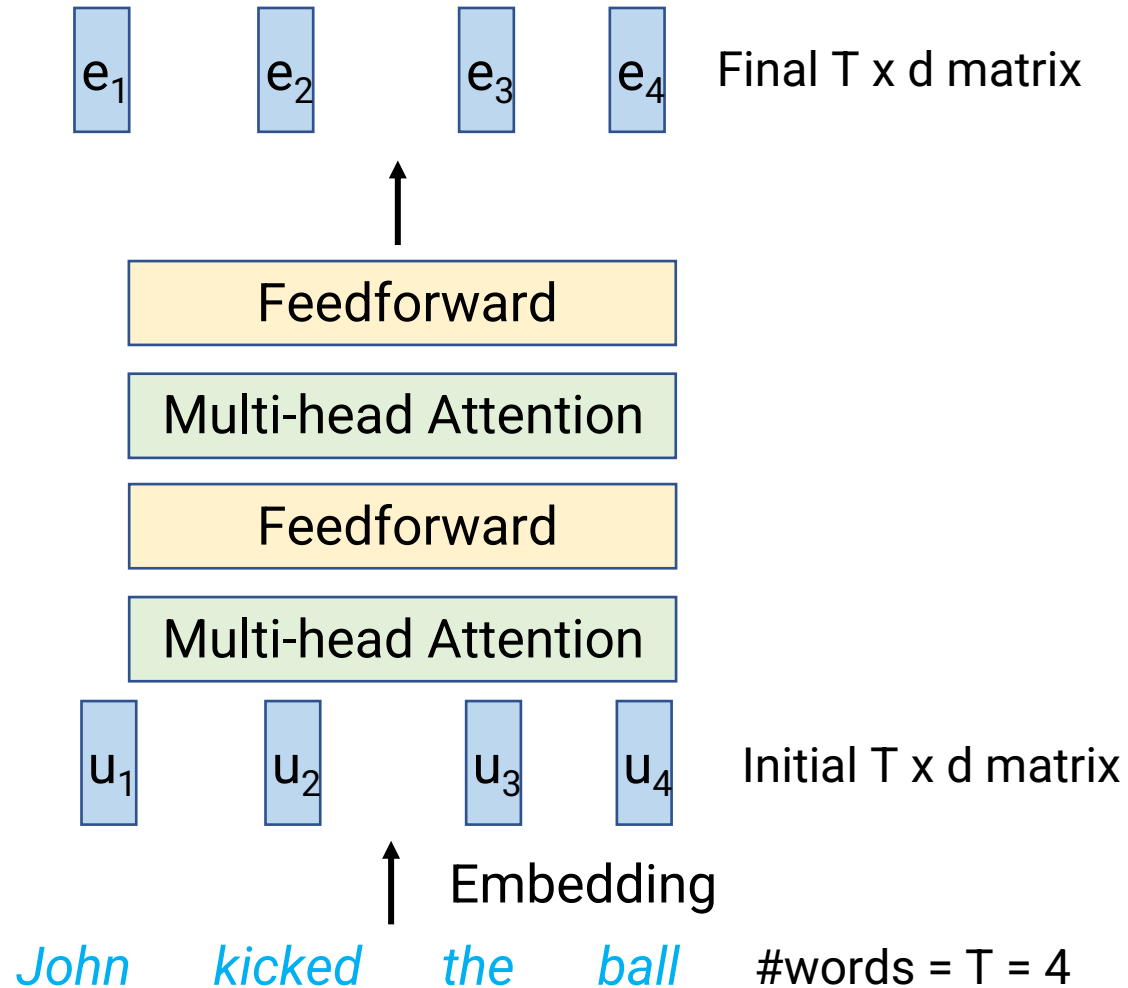
Once you have a trained model, you can pass in a new image and the model will output a probability distribution over the possible classes, in this case, leopard or tiger. The class with the highest probability would be the predicted class for the image.

- Language model-based Chatbot developed by OpenAI
- Large (but not largest) Transformer model
- Lots of information not public, but we do know some things...

Outline

- Technical ingredients of ChatGPT
 - Language model pretraining
 - Reinforcement Learning from Human Feedback
- Limitations and concerns

Previously: Transformers

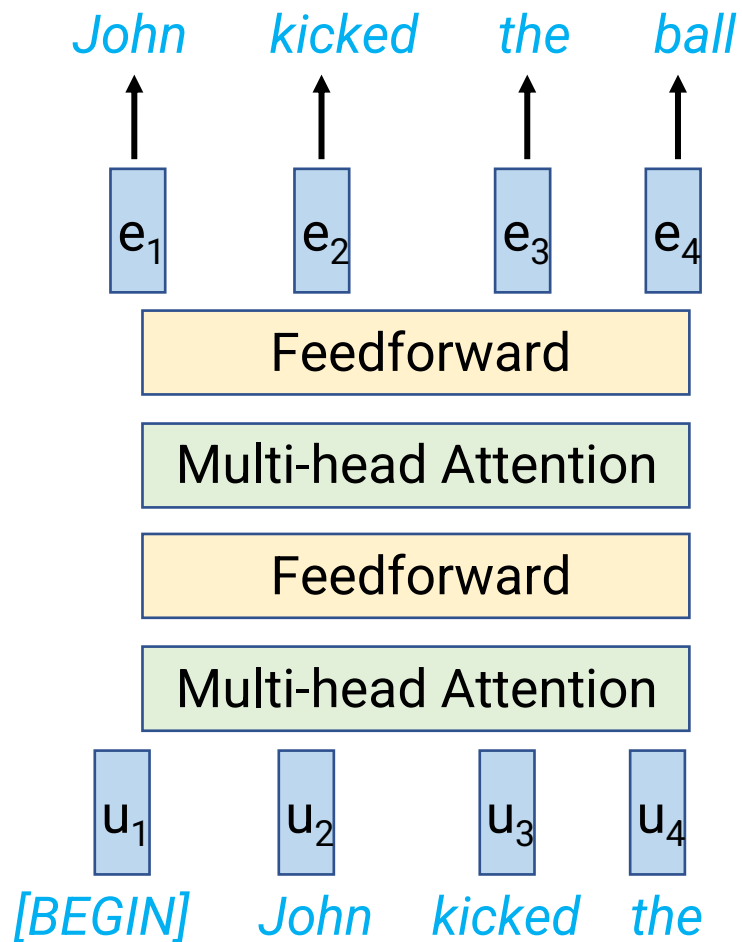


- One transformer consists of
 - Embeddings for each word of size d
 - Let $T = \text{\#words}$, so initially $T \times d$ matrix
 - Alternating layers of
 - “Multi-headed” attention layer
 - Feedforward layer
 - Both take in $T \times d$ matrix and output a new $T \times d$ matrix
 - Plus some bells and whistles
 - Residual connections & LayerNorm
 - Byte pair encoding tokenization

Autoregressive Language Model Training

- Training example: “Convolutional neural networks are good for image classification”
- Want to maximize $P(\text{“Convolutional neural networks are good for image classification”})$
- MLE: Take log and decompose by chain rule:
 - $\log P(\text{“Convolutional”})$
 - $+ \log P(\text{“neural”} \mid \text{“Convolutional”})$
 - $+ \log P(\text{“networks”} \mid \text{“Convolutional neural”})$
 - $+ \log P(\text{“are”} \mid \text{“Convolutional neural networks”}) + \dots$
- Decomposes into a bunch of **next-word-classification** problems
- Backpropagation + gradient descent to minimize loss

Review: Transformer autoregressive decoders



- How to do autoregressive language modeling?
- Test-time
 - At time t , attend to positions 1 through t
 - Happens in series

Queries

$[BEGIN]$				
$John$				
$kicked$				
the				

Keys

$[BEGIN]$ $John$ $kicked$ the

Review: Transformer autoregressive decoders

- When training a decoder, it has to be “used to” only attending to past/current tokens
- Training time: Masked attention implementation trick
 - Recall: Attention computes $Q \times K^T$ ($T \times T$ matrix), then does softmax
 - But if generating autoregressively, time t can only attend to times 1 through t
 - Solution: Overwrite $Q \times K^T$ to be $-\infty$ when query index $<$ key index
 - **All timesteps happen in parallel**

Queries	<i>[BEGIN]</i>	10	-2	6	3
	<i>John</i>	0	7	2	-4
	<i>kicked</i>	-3	4	5	-8
	<i>the</i>	2	1	7	6
	<i>[BEGIN]</i>	<i>John</i>	<i>kicked</i>	<i>the</i>	
			Keys		

Review: Transformer autoregressive decoders

- When training a decoder, it has to be “used to” only attending to past/current tokens
- Training time: Masked attention implementation trick
 - Recall: Attention computes $Q \times K^T$ ($T \times T$ matrix), then does softmax
 - But if generating autoregressively, time t can only attend to times 1 through t
 - Solution: Overwrite $Q \times K^T$ to be $-\infty$ when query index $<$ key index
 - **All timesteps happen in parallel**

Queries	<i>[BEGIN]</i>	10	-2	6	3
	<i>John</i>	0	7	2	-4
	<i>kicked</i>	-3	4	5	-8
	<i>the</i>	2	1	7	6
	<i>[BEGIN]</i>	<i>John</i>	<i>kicked</i>	<i>the</i>	
					Keys

Review: Transformer autoregressive decoders

- When training a decoder, it has to be “used to” only attending to past/current tokens
- Training time: Masked attention implementation trick
 - Recall: Attention computes $Q \times K^T$ ($T \times T$ matrix), then does softmax
 - But if generating autoregressively, time t can only attend to times 1 through t
 - Solution: Overwrite $Q \times K^T$ to be $-\infty$ when query index $<$ key index
 - **All timesteps happen in parallel**

Queries	<i>[BEGIN]</i>	10	$-\infty$	$-\infty$	$-\infty$
	<i>John</i>	0	7	$-\infty$	$-\infty$
	<i>kicked</i>	-3	4	5	$-\infty$
	<i>the</i>	2	1	7	6
	<i>[BEGIN]</i>	<i>John</i>	<i>kicked</i>	<i>the</i>	

Keys

Review: Transformer autoregressive decoders

- When training a decoder, it has to be “used to” only attending to past/current tokens
- Training time: Masked attention implementation trick
 - Recall: Attention computes $Q \times K^T$ ($T \times T$ matrix), then does softmax
 - But if generating autoregressively, time t can only attend to times 1 through t
 - Solution: Overwrite $Q \times K^T$ to be $-\infty$ when query index $<$ key index
 - **All timesteps happen in parallel**

Queries	<i>[BEGIN]</i>	1.0	0	0	0
	<i>John</i>	.001	.999	0	0
	<i>kicked</i>	.001	.356	.643	0
	<i>the</i>	.030	.007	.591	.372
	<i>[BEGIN]</i>	<i>John</i>	<i>kicked</i>	<i>the</i>	
					Keys

Pre-trained language models

- GPT-3 (2020): A 175 billion parameter language model
 - Architecture
 - 96 Transformer layers
 - 12288-dimensional hidden states
 - 96 heads in each attention layer
 - Trained on a very large corpus of documents scraped from the web
 - Some filters used to promote data quality
 - One strategy: Train classifier to distinguish random internet documents from ones from known “high-quality” sources, drop documents with low classifier score
- ChatGPT (gpt-3.5-turbo): Reportedly 20 billion parameters
 - Easier to deploy at scale than 175B model
 - Likely was first pretrained in a similar manner as GPT-3
 - But then an additional training step was added...

Outline

- Technical ingredients of ChatGPT
 - Language model pretraining
 - Reinforcement Learning from Human Feedback
- Limitations and concerns

Supervised fine-tuning

- Pretraining stage: Train on all the data you can get access to
 - Pros: A lot of data avoids overfitting, model can learn about all sorts of long-tail knowledge
 - Cons: You're training the model to imitate the average internet post
 - High quantity, low quality!
- Solution: Fine-tune on a smaller, highly curated dataset after
 - These are examples you really do want the model to imitate
 - Pre-training has taught the model many things; fine-tuning tells it to only verbalize the parts that are desirable

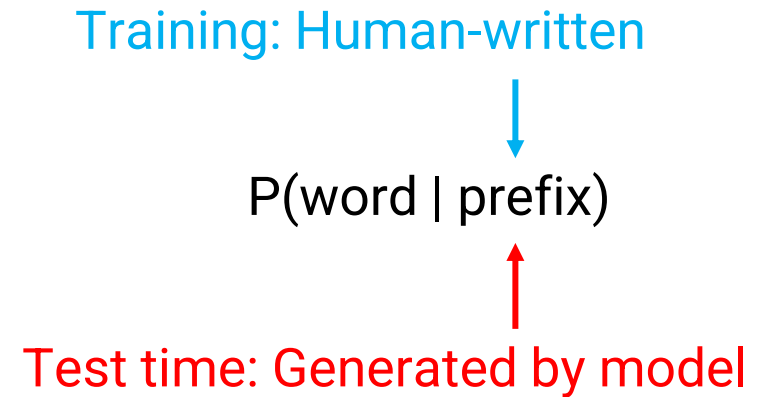


Limitations of Supervised fine-tuning

- Problem 1: Data scale
 - High-quality data is expensive to obtain, you don't have that much of it (relative to pretraining data)
- Problem 2: Exposure bias

Exposure bias

- Training time: Model learns to predict next word **given human-written prefix**
- Test time: Model must predict next word given **model-written prefix**
- **Exposure bias**: Model was never “exposed” to its own outputs during training, so it may not know what to do!



Exposure bias and reinforcement learning

- We can view sequence generation as a reinforcement learning problem!
 - Action: Which word to generate next
 - State: Sequence of all words generated so far
 - Reward: Whether the final complete output is “good”
 - Rewards are 0 for intermediate timesteps
 - Only get non-zero reward at final timestep
- In RL, supervised fine-tuning is called “**imitation learning**”
 - Known to be suboptimal due to exposure bias
 - You can try to mimic an expert player, but a worse player also needs to know how to recover from mistakes



Limitations of Supervised fine-tuning

- Problem 1: Data scale
 - You simply don't have much data at your disposal
- Problem 2: Exposure bias
- **Problem 3: Can promote guessing rather than factual responses**

Review: Neural networks as feature learners



Input x

Classifier 1:
Is front clear?

Classifier 2:
Is left clear?

Classifier 3:
Is right clear?

$\begin{bmatrix} 0 \\ 1 \\ 0 \end{bmatrix}$

Classifier 4:
Where to go?

Output y
Turn left

Learn to classify based on features
(same as linear model)

Learn a classifier whose output is a good feature

We don't tell the model what classifier to learn
Model must learn that "is front clear" is a useful concept

Supervised Fine-tuning and Factuality

- Consider the following fine-tuning examples
 - Prompt: “When was the US Declaration of Independence signed?”
Answer: “July 4, 1776”
 - Model has probably seen this information many times during pre-training
 - So it has probably learned features that associate the Declaration of Independence with July 4
 - Prompt: “When was Robin Jia born?”
Answer: “[...]”
 - Model has probably (?) not seen this information during pre-training
 - Cannot have learned features associating me with my birthday
 - Supervised fine-tuning **encourages the model to just make something up!**

Limitations of Supervised fine-tuning

- Problem 1: **Data scale**
 - You simply don't have much data at your disposal
- Problem 2: **Exposure bias**
- Problem 3: **Can promote guessing rather than factual responses**
- Solution: **Fine-tune the language model with reinforcement learning!**
 - Use a reward that encourages the **model's full outputs** to be **correct/factual**
 - Reward will be **computed with another model** (can get infinite data now)

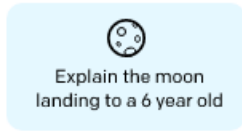
Reinforcement Learning from Human Feedback

Step 1

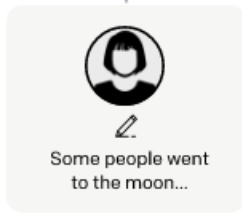
Collect demonstration data, and train a supervised policy.

Supervised Fine-tuning

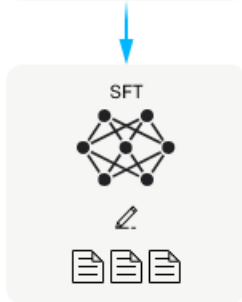
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



This data is used to fine-tune GPT-3 with supervised learning.

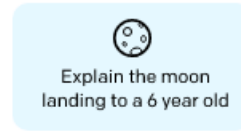


Step 2

Collect comparison data, and train a reward model.

Learn to assign rewards

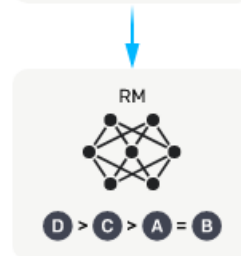
A prompt and several model outputs are sampled.



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 reinforcement learning.

Do RL on learned rewards

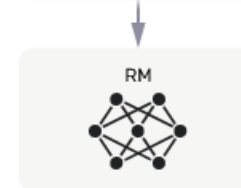
A new prompt is sampled from the dataset.



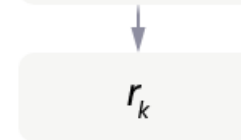
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



- “RLHF” for short
- Trains language model with RL
- Rewards come from a model trained to predict human preferences

RLHF: The Data

Step 1

Collect demonstration data,
and train a supervised policy.

Supervised Fine-tuning

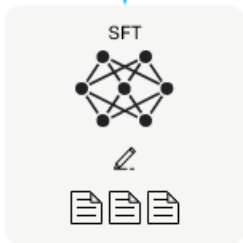
A prompt is
sampled from our
prompt dataset.

Explain the moon
landing to a 6 year old

A labeler
demonstrates the
desired output
behavior.

Some people went
to the moon...

This data is used
to fine-tune GPT-3
with supervised
learning.



Step 2

Collect comparison data,
and train a reward model.

Learn to assign rewards

A prompt and
several model
outputs are
sampled.

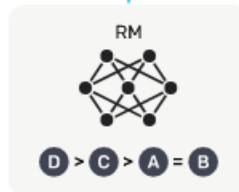
Explain the moon
landing to a 6 year old

Explain gravity... Explain war...
C: Moon is natural satellite of... D: People went to the moon...

A labeler ranks
the outputs from
best to worst.

D > C > A = B

This data is used
to train our
reward model.



Part 1: Prompts

- Some written by hired annotators
- Some based on use-cases from waitlist applications to OpenAI (!)
- Some from actual user queries to the OpenAI API

Part 2: Demonstrations

- Hired annotator writes a desired response to prompt

Part 3: Rankings

- Sample several model responses from model's probability distribution
- Hired annotator ranks them from best to worst

RLHF: Who's behind the data?

- InstructGPT paper (precursor to ChatGPT): “We hired a team of about 40 contractors on Upwork and through ScaleAI”
- Labelers had to pass various screening tests

What gender do you identify as?	
Male	50.0%
Female	44.4%
Nonbinary / other	5.6%

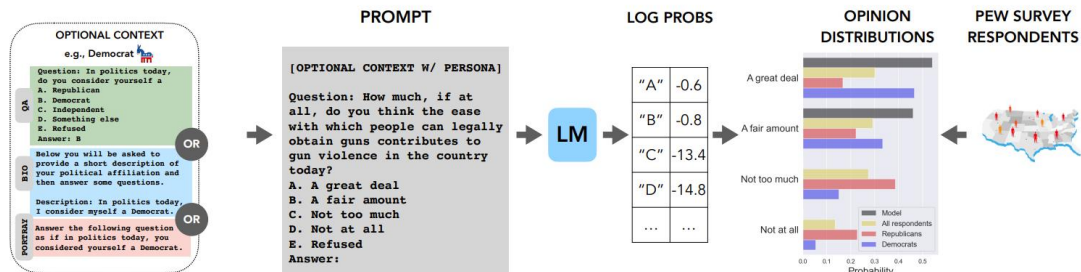
What ethnicities do you identify as?	
White / Caucasian	31.6%
Southeast Asian	52.6%
Indigenous / Native American / Alaskan Native	0.0%
East Asian	5.3%
Middle Eastern	0.0%
Latinx	15.8%
Black / of African descent	10.5%

What is your nationality?	
Filipino	22%
Bangladeshi	22%
American	17%
Albanian	5%
Brazilian	5%
Canadian	5%
Colombian	5%
Indian	5%
Uruguayan	5%
Zimbabwean	5%

What is your age?	
18-24	26.3%
25-34	47.4%
35-44	10.5%
45-54	10.5%
55-64	5.3%
65+	0%

What is your highest attained level of education?	
Less than high school degree	0%
High school degree	10.5%
Undergraduate degree	52.6%
Master's degree	36.8%
Doctorate degree	0%

RLHF: Whose opinions?



- Idea: Ask language models to answer pew research survey questions
- Models after RLHF agrees most with college-educated, Asian, 18-49 year olds
 - Matches InstructGPT demographics!
 - Non-RLHF models more similar to less educated respondents

Model	AI21 Labs			OpenAI					
	j1-grande	j1-jumbo	j1-grande-v2-beta	ada	davinci	text-ada-001	text-davinci-001	text-davinci-002	text-davinci-003
EDUCATION									
Less than high school	0.827	0.828	0.812	0.835	0.801	0.710	0.714	0.750	0.684
High school graduate	0.817	0.816	0.799	0.826	0.790	0.711	0.712	0.755	0.690
Some college, no degree	0.811	0.814	0.803	0.823	0.790	0.706	0.714	0.762	0.700
Associate's degree	0.809	0.811	0.800	0.821	0.789	0.703	0.712	0.761	0.699
College graduate/some postgrad	0.797	0.802	0.793	0.810	0.780	0.701	0.713	0.766	0.710
Postgraduate	0.788	0.794	0.789	0.800	0.775	0.695	0.712	0.766	0.716

Closest relative of ChatGPT

Highest similarity

RLHF: Who's behind the data?

- Increasingly: Gig work for interacting with chatbots
 - Lots of secrecy
 - Compensation can vary greatly over time
 - Task availability is unpredictable
- *“Annotators spend hours reading instructions and completing unpaid trainings only to do a dozen tasks and then have the project end...Any task could be their last, and they never know when the next one will come.”*



Intelligencer

SUBSCRIBE

AI Is a Lot of Work As the technology becomes ubiquitous, a vast tasker underclass is emerging — and not going anywhere.

By Josh Dzieza

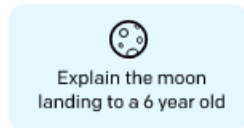
RLHF: Initial Supervised Fine-tuning

Step 1

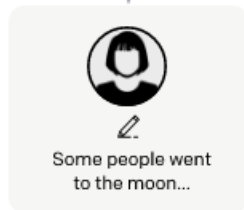
Collect demonstration data,
and train a supervised policy.

Supervised Fine-tuning

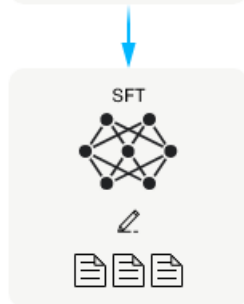
A prompt is
sampled from our
prompt dataset.



A labeler
demonstrates the
desired output
behavior.



This data is used
to fine-tune GPT-3
with supervised
learning.



- Run imitation learning on labeler-provided demonstrations, given prompt as prefix
- Very similar to pre-training, except:
 - Only compute loss on response tokens, not on prompt tokens
 - Much smaller dataset

Loss on this example =

$$\begin{aligned} & \log P(\text{"Some"} \mid \text{"Explain the moon landing to a 6 year old:"}) \\ & + \log P(\text{"people"} \mid \text{"Explain the moon landing to a 6 year old: Some"}) \\ & + \dots \end{aligned}$$

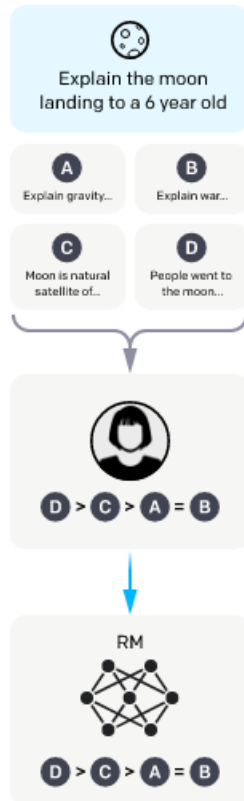
RLHF: The Reward Model

Step 2

Collect comparison data,
and train a reward model.

Learn to assign rewards

A prompt and
several model
outputs are
sampled.

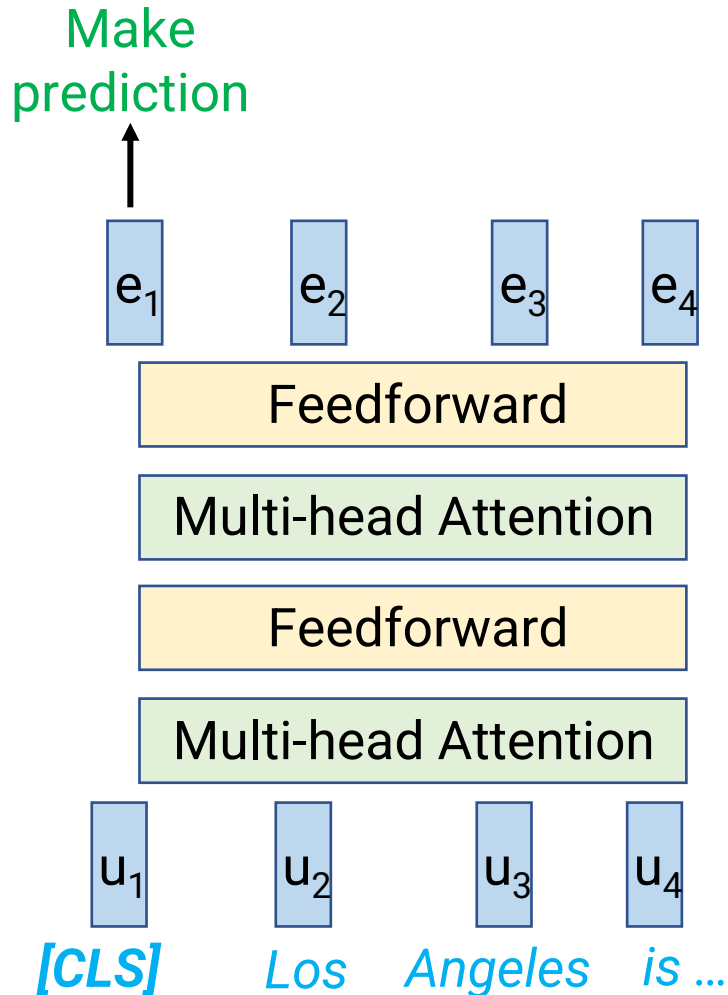


A labeler ranks
the outputs from
best to worst.

This data is used
to train our
reward model.

- Goal: Fine-tune ChatGPT with RL
- Step 1: Show human annotators several sampled model outputs, ask them to rank
 - Provides some RL training data, but not a ton
- Step 2: Train a “**reward model**” to predict the human’s rankings
 - Now we can run as many RL training episodes as we want for free, using the reward model in place of the human annotators

Previously: BERT Fine-tuning



- BERT: Model pre-trained by masked language modeling
- Initialize parameters with BERT
 - BERT was trained to expect every input to start with a special token called [CLS]
- Add parameters that take in the output at the [CLS] position and make prediction
- Keep training all parameters (“fine-tune”) on the new task
- Reward model is a similar model that was pretrained, then fine-tuned to predict reward

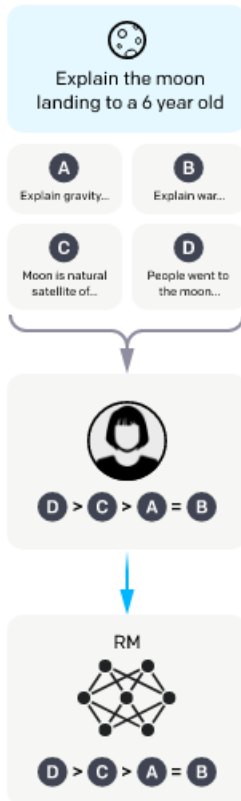
RLHF: The Reward Model

Step 2

Collect comparison data,
and train a reward model.

Learn to assign rewards

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.

- Model

- 6 billion parameter pretrained language model
- Then fine-tune all parameters (like BERT)
- A smaller model than ChatGPT itself

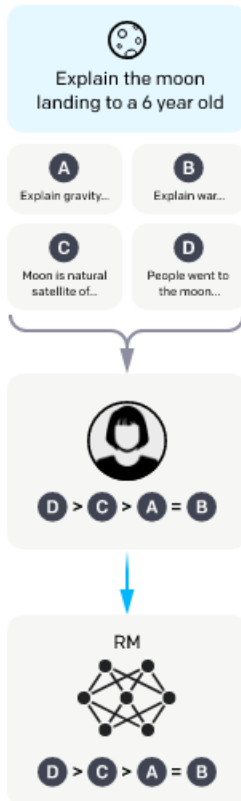
RLHF: Training the Reward Model

Step 2

Collect comparison data,
and train a reward model.

Learn to assign rewards

A prompt and
several model
outputs are
sampled.



- Input: Prompt x , winning output y_w , losing output y_l
 - Wins/losses determined by labelers' rankings
- Reward model predicts scalar reward $r_\theta(x, y)$ given prompt x and model output y
- Objective on one example is to minimize:
$$-\log \sigma(r_\theta(x, y_w) - r_\theta(x, y_l))$$
 - Want reward on y_w to be higher than on y_l
 - Use the familiar logistic regression loss function!
 - Loss goes to 0 if argument is large
 - Loss goes to infinity if argument is small
 - Binary classification of which output is better

RLHF: Doing RL on the Language Model

Step 3

Optimize a policy against the reward model using reinforcement learning.

Do RL on learned rewards

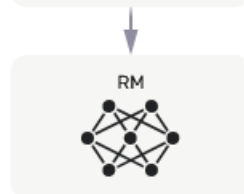
A new prompt is sampled from the dataset.



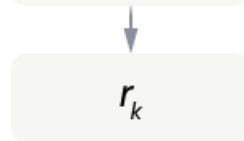
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



- Which RL algorithm should we use?
 - Deep Q-Learning?
 - Policy Gradient?

RLHF: Doing RL on the Language Model

Step 3

Optimize a policy against the reward model using reinforcement learning.

Do RL on learned rewards

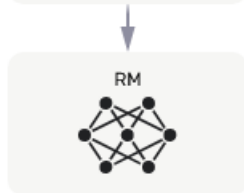
A new prompt is sampled from the dataset.



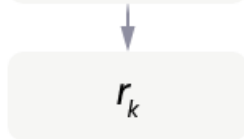
The policy generates an output.



The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.



- Which RL algorithm should we use?
 - Deep Q-Learning?
 - Policy Gradient?
- To answer this, we should ask: How does a language model define a RL policy?
 - It directly classifies the next action (i.e., next word) in the current state (i.e., sequence of words generated so far)
 - This is what **policy gradient** requires!
 - It does not predict a Q-value, like Q-learning expects

RLHF: Doing RL on the Language Model

Step 3

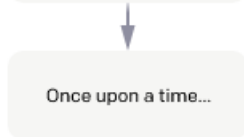
Optimize a policy against the reward model using reinforcement learning.

Do RL on learned rewards

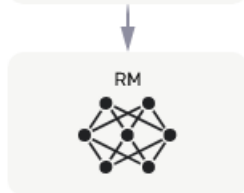
A new prompt is sampled from the dataset.



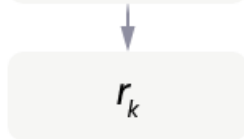
The policy generates an output.



The reward model calculates a reward for the output.






The reward is used to update the policy using PPO.




- We need to use policy gradient!
 - Recall: Policy gradient computes a quantity whose expected value is the gradient w.r.t. parameters of the expected reward, then uses that quantity to do gradient ascent
- Algorithm of choice: Proximal Policy Optimization (PPO)
 - Basic policy gradient method estimates the gradient of expected reward, but with very high variance
 - Idea 1: Use “advantage” (how much this action improves reward vs. baseline) instead of raw rewards—lowers variance
 - Idea 2: Only make small updates to the policy at each step, in case the estimated gradient goes in the wrong direction

RLHF and Factuality

- Let's revisit the earlier examples:
 - Prompt: "When was the US Declaration of Independence signed?"
Answer: "July 4, 1776"
 - Model outputs
 - "July 4, 1776" 
 - "January 1, 1950" 
 - "I don't know" 
 - Top-ranked answer is the real answer

RLHF and Factuality

- Let's revisit the earlier examples:
 - Prompt: "When was Robin Jia born?"
Answer: "[...]"
 - Model outputs
 - "December 7, 1831"
 - "May 23, 1989"
 - "I don't know" 
 - Top-ranked answer is to say "I don't know"
- Overall effect: Model encouraged to say "I don't know" when its top candidate outputs are wrong

Announcements

- Homework 4 due this Thursday
- Thursday's class: Conclusion + in-class course evaluation
- Section on Friday: Final review
- Final project reports due May 3
- Final Exam May 7, 2-4pm in OHE 122
 - Can bring two sheets of paper with notes, no other assistance
 - Practice exams posted

Outline

- Technical ingredients of ChatGPT
 - Language model pretraining
 - Reinforcement Learning from Human Feedback
- Limitations and concerns

ChatGPT “reasoning”

R

What weighs more, a pound of feathers or a ton of bricks?



Both a pound of feathers and a ton of bricks weigh the same amount, which is 2000 pounds.



This is because a pound is a unit of weight measurement, just like a ton.

However, a pound is a much smaller unit than a ton, so it takes a much larger quantity of pounds to equal a ton.

Therefore, while the feathers may take up more space than the bricks, they still weigh the same amount.

- ChatGPT has likely seen this riddle before about a **pound** of feathers vs. **pound** of bricks
- Overgeneralizes based on that answer
- This example fooled ChatGPT in April 2023...

ChatGPT “reasoning” revisited



You

What weighs more, a pound of bricks or a kilogram of feathers?

< 5/5 >



ChatGPT

They weigh the same. One pound is equivalent to 0.453592 kilograms. Therefore, a pound of bricks and a kilogram of feathers both weigh approximately 0.453592 kilograms. The difference lies in the density and volume of the materials, with feathers being much less dense than bricks.



- December 2023: ChatGPT gets confused about pounds and kilograms
 - It knows that 1 pound is less than 1 kilogram but still says they weigh the same...?

ChatGPT “reasoning” revisited revisited



You

What weighs more, a pound of feathers or a kilogram of cotton?

< 5 / 5 >



ChatGPT

Neither, they both weigh the same! A pound of feathers weighs the same as a kilogram of cotton. They're just measured in different units: pounds for feathers and kilograms for cotton.



- Today: Got the question about bricks right, but wrong when I changed the nouns...
- This will probably get patched out eventually too, but issues will remain...

ChatGPT can “Hallucinate”

R Find me a research paper that tests question answering systems with paragraphs that have an extra sentence added at the end

Here's a research paper that fits your request:

Does not exist

Title: Evaluating the Robustness of Question Answering Systems to Sentence-Level Perturbations

Authors: Prawira Saskara, Daya Guo, Anh Tuan Luu, Jinho D. Choi

Journal: **Not** **NLP** **NLP** **NLP** Computational
Linguistics **real(?)** **PhD** **Prof.** **Prof.**
Student **(NTU)** **(Emory)**

Link to the paper: <https://www.aclweb.org/anthology/P19-1148.pdf> URL links to:

Summary: The paper presents an evaluation framework for testing the robustness of question answering systems to sentence-level perturbations. Specifically, the authors introduce a new dataset called **SQuAD-AddSent**, which consists of the original SQuAD paragraphs with an extra sentence added at the end. The added sentence can either be a

- This is a description of one of my research papers from 2017
- We did actually have a method called **AddSent** and applied it to the **SQuAD** dataset
- This example also a couple months old: today ChatGPT refuses to answer this question
- But of course hallucinations can still happen...

Exact Hard Monotonic Attention for Character-Level Transduction

Shijie Wu^a and Ryan Cotterell^{a,fi}

^aDepartment of Computer Science, Johns Hopkins University

^{fi}Department of Computer Science and Technology, University of Cambridge

shijie.wu@jhu.edu, rdc42@cam.ac.uk

Dangers of Trusting Language Models

*“Brian Hood, who was elected mayor of Hepburn Shire, 120km (75 miles) northwest of Melbourne, last November, became concerned about his reputation when members of the public told him **ChatGPT had falsely named him as a guilty party in a foreign bribery scandal involving a subsidiary of the Reserve Bank of Australia in the early 2000s.***

*Hood did work for the subsidiary, Note Printing Australia, but **was the person who notified authorities about payment of bribes** to foreign officials to win currency printing contracts, and was **never charged with a crime**, lawyers representing him said.”*



REUTERS®

World ▾

Business ▾

Markets ▾

Legal ▾

More ▾



Technology



3 minute read · April 5, 2023 11:52 AM PDT · Last Updated 19 days ago



Australian mayor readies world's first defamation lawsuit over ChatGPT content

By Byron Kaye

Dangers of Trusting Language Models

*A U.S. judge on Thursday imposed sanctions on two New York lawyers who submitted a legal brief that included **six fictitious case citations generated by an artificial intelligence chatbot, ChatGPT.***

Schwartz admitted in May that he had used ChatGPT to help research the brief in a client's personal injury case against Colombian airline Avianca, and unknowingly included the false citations.



New York lawyers sanctioned for using fake ChatGPT cases in legal brief

By Sara Merken

June 26, 2023 1:28 AM PDT · Updated 9 months ago



Dangers of Trusting Language Models

*“After months of resisting, Air Canada was forced to give a partial refund to a grieving passenger who was **misled by an airline chatbot inaccurately explaining the airline's bereavement travel policy.***

...

*The chatbot provided inaccurate information, encouraging Moffatt to book a flight immediately and then request a refund within 90 days. **In reality, Air Canada's policy explicitly stated that the airline will not provide refunds for bereavement travel after the flight is booked.** Moffatt dutifully attempted to follow the chatbot's advice and request a refund but was shocked that the request was rejected.”*

WIRED

ASHLEY BELANGER, ARS TECHNICA

BUSINESS FEB 17, 2024 12:12 PM

Air Canada Has to Honor a Refund Policy Its Chatbot Made Up

The airline tried to argue that it shouldn't be liable for anything its chatbot says.

Conclusion

- How does ChatGPT work?
 - Stage 1: Pre-training on large corpus of text
 - Stage 2: RLHF
 - Supervised fine-tuning on human demonstrations
 - Train a reward model to provide feedback to LM
 - Fine-tune LM with policy gradient (PPO) to maximize rewards given by reward model
- Words of caution
 - ChatGPT answers may be made up!
 - Useful for brainstorming and suggestions, bad for facts
 - Likelihood of success depends on commonality of data in pre-training/RLHF datasets