

Natural Language Processing

AI51701/CSE71001

Lecture 16

11/14/2023

Instructor: Taehwan Kim

Natural Language Generation

What is natural language generation?

- ❑ Natural language generation is one side of natural language processing.
 - NLP = Natural Language Understanding (NLU) + Natural Language Generation (NLG)
- ❑ Any task involving language production for human consumption requires natural language generation
- ❑ NLG focuses on systems that produce **coherent** and **useful** language output for human consumption
- ❑ Deep Learning is powering (some) next-gen NLG systems!

Uses of natural language generation

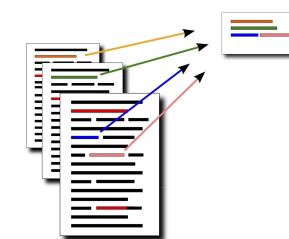
- ❑ Machine Translation systems use NLG for output



- ❑ Digital assistant (dialogue) systems use NLG



- ❑ Summarization systems (for research articles, email, meetings, documents) use NLG

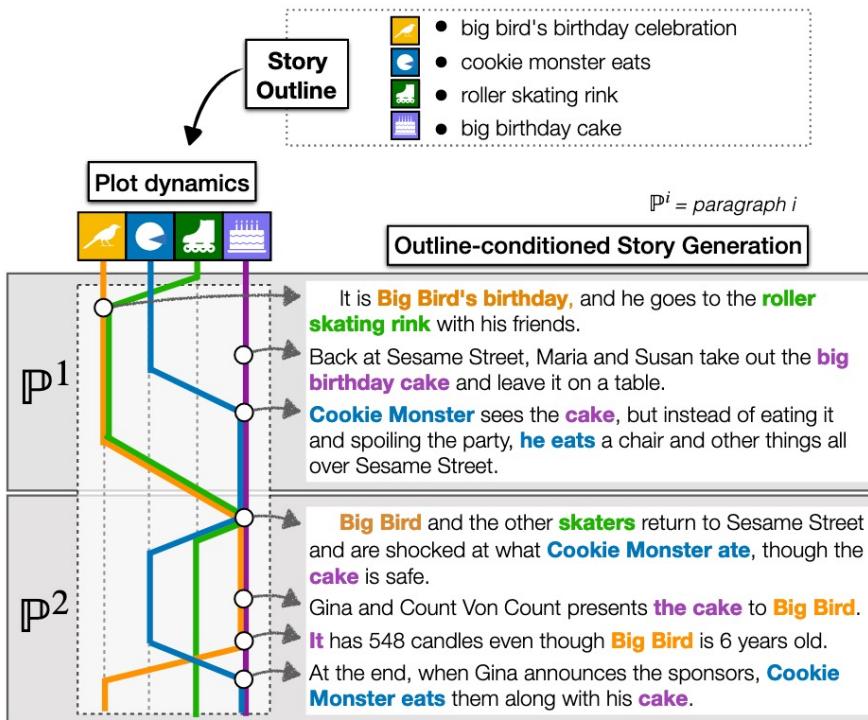


C: Looking at what we've got, we we want an LCD display with a spinning wheel.
B: You have to have some push-buttons, don't you?
C: Just spinning and not scrolling, I would say.
B: I think the spinning wheel is definitely very now.
A: but since LCDs seems to be uh a definite yes,
C: We're having push-buttons on the outside
C: and then on the inside an LCD with spinning wheel,

Decision Abstract (Summary):
The remote will have push buttons outside, and an LCD and spinning wheel inside.

More interesting NLG uses

Creative stories



(Rashkin et al., EMNLP 2020)

Data-to-text

Table Title: Robert Craig (American football)
Section Title: National Football League statistics
Table Description:None

YEAR	TEAM	RUSHING				RECEIVING			
		ATT	YDS	AVG	LNG	TD	NO.	YDS	AVG
1983	SF	176	725	4.1	71	8	48	427	8.9
1984	SF	155	649	4.2	28	4	71	675	9.5
1985	SF	214	1050	4.9	62	9	92	1016	11
1986	SF	204	830	4.1	25	7	81	624	7.7
1987	SF	215	815	3.8	25	3	66	492	7.5
1988	SF	310	1502	4.8	46	9	76	534	7.0
1989	SF	271	1054	3.9	27	6	49	473	9.7
1990	SF	141	439	3.1	26	1	25	201	8.0
1991	RAI	162	590	3.6	15	1	17	136	8.0
1992	MIN	105	416	4.0	21	4	22	164	7.5
1993	MIN	38	119	3.1	11	1	19	169	8.9
Totals	-	1991	8189	4.1	71	56	566	4911	8.7

Craig finished his eleven NFL seasons with 8,189 rushing yards and 566 receptions for 4,911 receiving yards.

(Parikh et al., EMNLP 2020)

Visual description

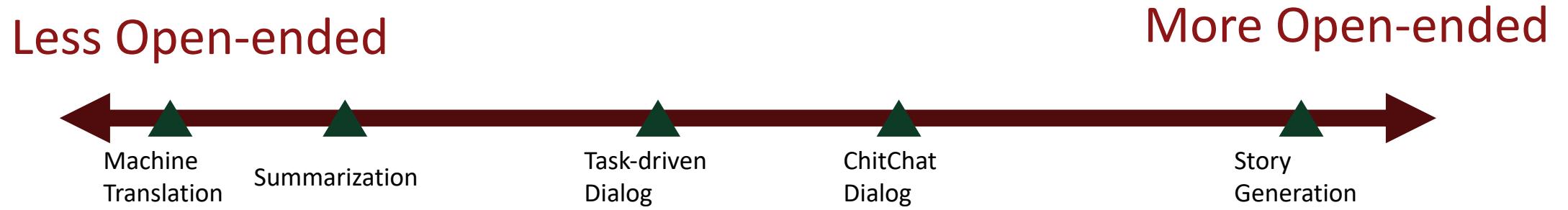


Two children are sitting at a table in a restaurant. The children are one little girl and one little boy. The little girl is eating a pink frosted donut with white icing lines on top of it. The girl has blonde hair and is wearing a green jacket with a black long sleeve shirt underneath. The little boy is wearing a black zip up jacket and is holding his finger to his lip but is not eating. A metal napkin dispenser is in between them at the table. The wall next to them is white brick. Two adults are on the other side of the short white brick wall. The room has white circular lights on the ceiling and a large window in the front of the restaurant. It is daylight outside.

(Krause et al. CVPR 2017)

Categorization of NLG tasks

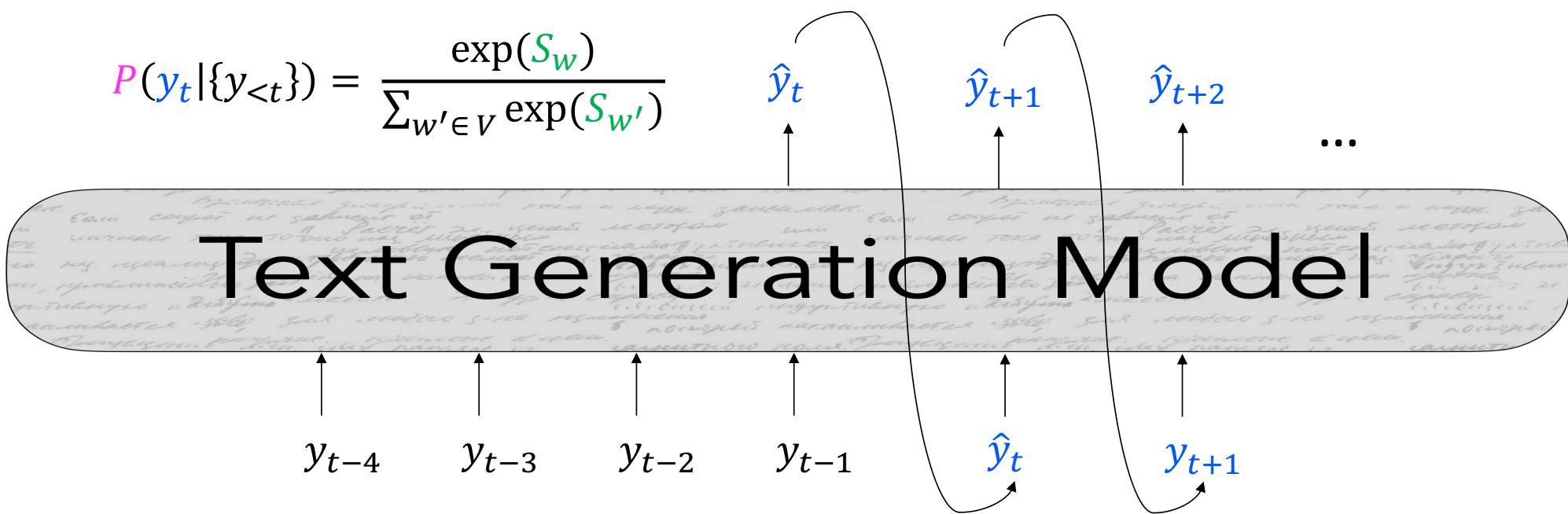
- ❑ Open-ended generation: the output distribution still has high freedom
 - ❑ Non-open-ended generation: the input mostly determines the output generation.



- Remark: One way of formalizing categorization this is by **entropy**.
 - These two classes of NLG tasks require different decoding and/or training approaches!

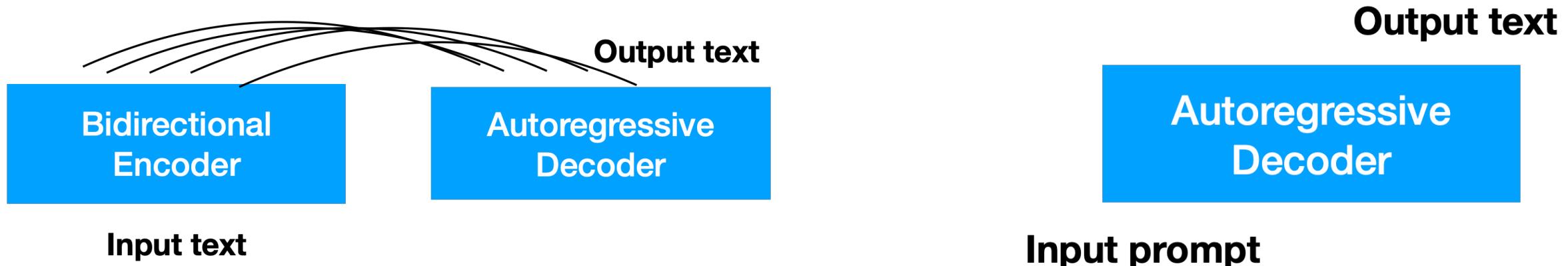
Basics of natural language generation (review)

- In autoregressive text generation models, at each time step t , our model takes in a sequence of tokens of text as input $\{y\}_{<t}$ and outputs a new token, \hat{y}_t
- For model $f(\cdot)$ and vocab V , we get scores $S = f(\{y_{<t}\}, \theta) \in \mathbb{R}^V$



Basics of natural language generation (review)

- ❑ 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.

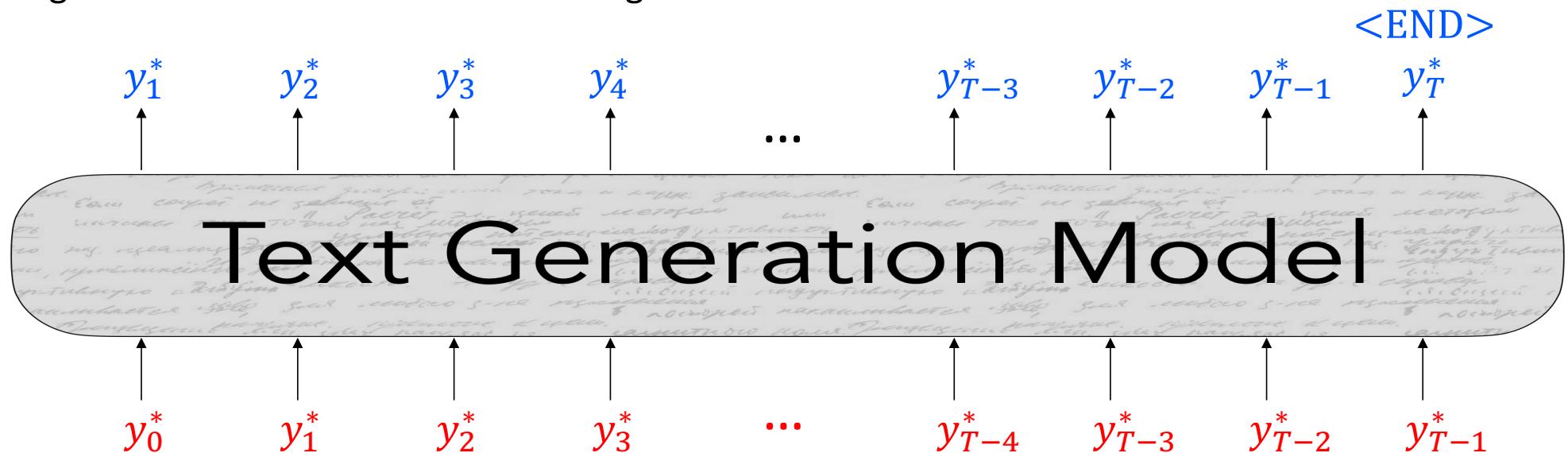


Trained one token at a time by maximum likelihood *teacher forcing*

- Trained to maximize the probability of the next token y_t^* given preceding words $\{y^*\}_{<t}$

$$\mathcal{L} = - \sum_{t=1}^T \log P(y_t^* | \{y^*\}_{<t})$$

- This is a classification task at each time step trying to predict the actual word y_t^* in the training data
- Doing this is often called “teacher forcing”



Basics of natural language generation (review)

- At inference time, our decoding algorithm defines a function to select a token from this distribution:

$$\hat{y}_t = g(P(y_t | \{y_{<t}\}))$$

g(.) is your decoding algorithm

- The “obvious” decoding algorithm is to greedily choose the highest probability next token according to the model at each time step
- While this basic algorithm sort of works, to do better, the two main avenues are:
 - 1. Improve the decoder
 - 2. Improve the training
 - Of course, there’s also improving your training data or model architecture

Decoding from NLG models

Decoding: what is it all about?

- At each time step t , our model computes a vector of scores for each token in our vocabulary, $S \in \mathbb{R}^V$:

$$S = f(\{y_{<t}\})$$

$f(\cdot)$ is your model

- Then, we compute a probability distribution P over these scores (usually with a softmax function):

$$P(y_t = w | \{y_{<t}\}) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- Our decoding algorithm defines a function to select a token from this distribution:

$$\hat{y}_t = g(P(y_t | \{y_{<t}\}))$$

$g(\cdot)$ is your decoding algorithm

How to find the most likely string? (review)

□ Greedy Decoding

- Selects the highest probability token in $P(y_t | y_{<t})$

$$\hat{y}_t = \operatorname{argmax}_{w \in V} P(y_t = w | y_{<t})$$

□ Beam Search

- Discussed in the lecture on Machine Translation
- Also aims to find strings that maximize the log-prob, but with wider exploration of candidates

□ Overall, maximum probability decoding is good for low-entropy tasks like MT and summarization!

The most likely string is repetitive for Open-ended Generation

Context:

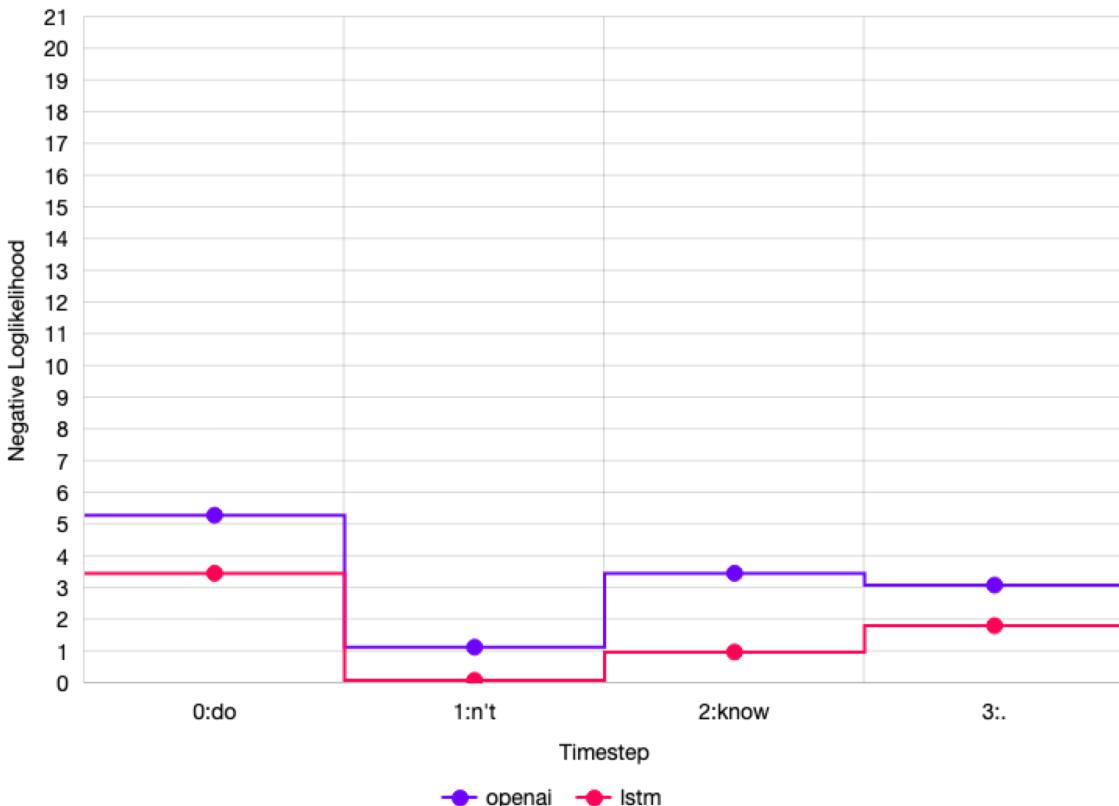
In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

Continuation:

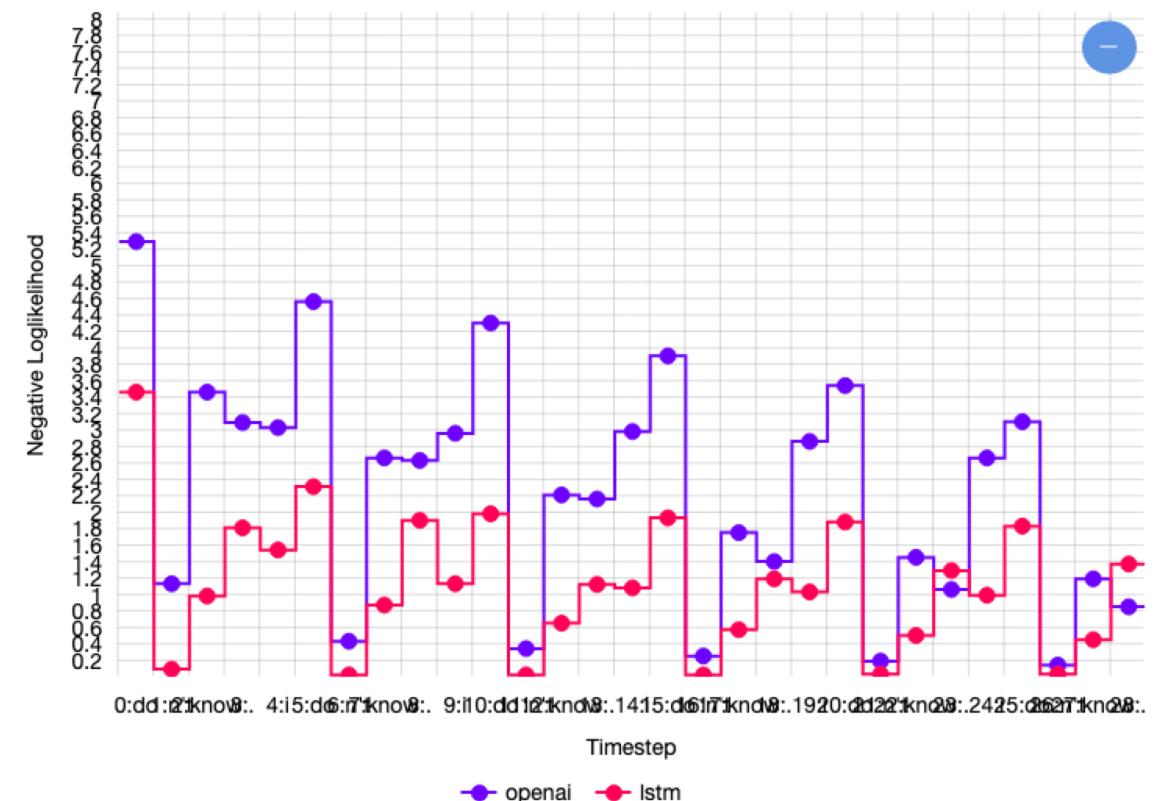
The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México...)**

Why does repetition happen?

I don't know.



I don't know. I don't know.

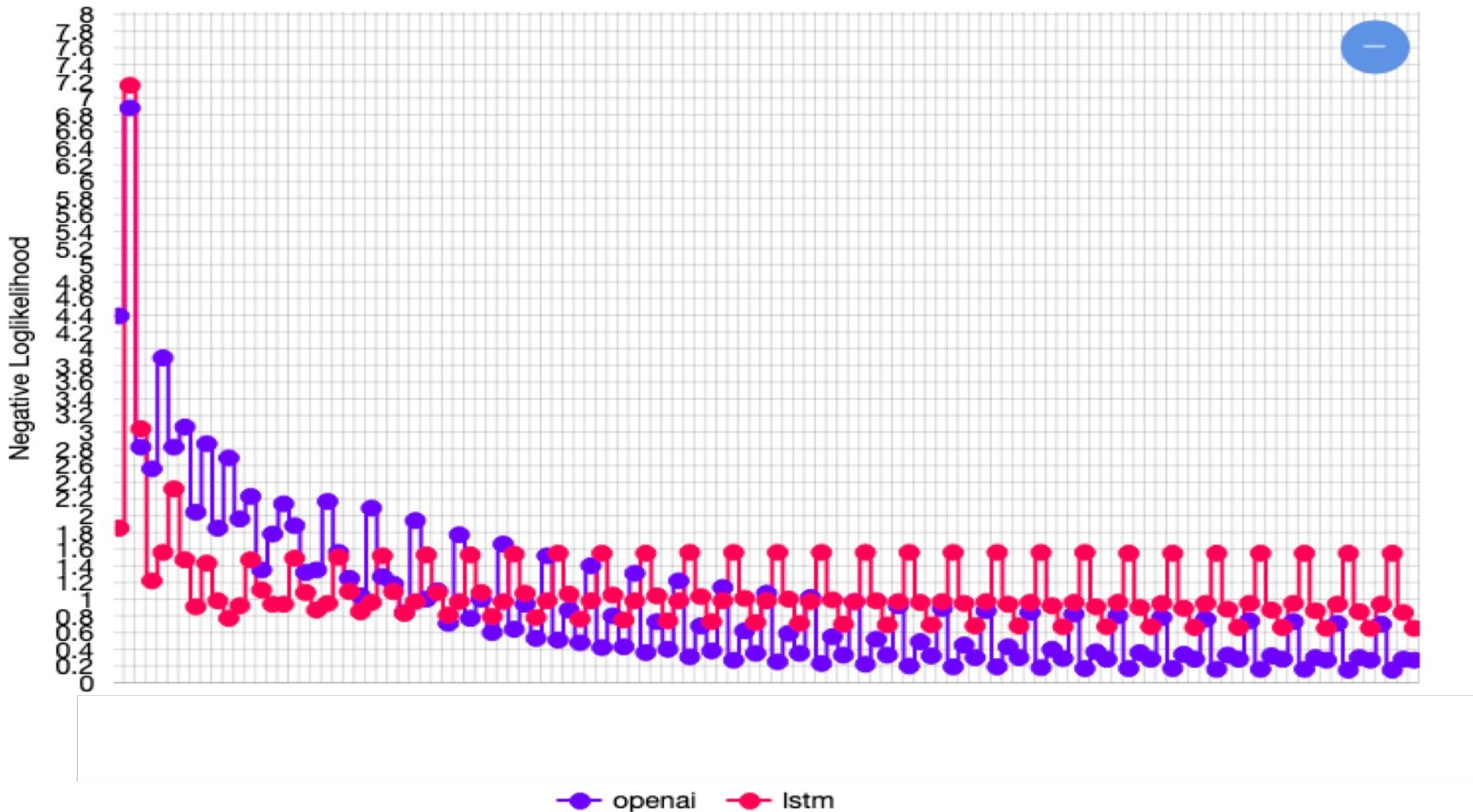


A self-amplification effect!

(Holtzman et. al., ICLR 2020)

And it keeps going...

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. al., ICLR 2020)

How can we reduce repetition?

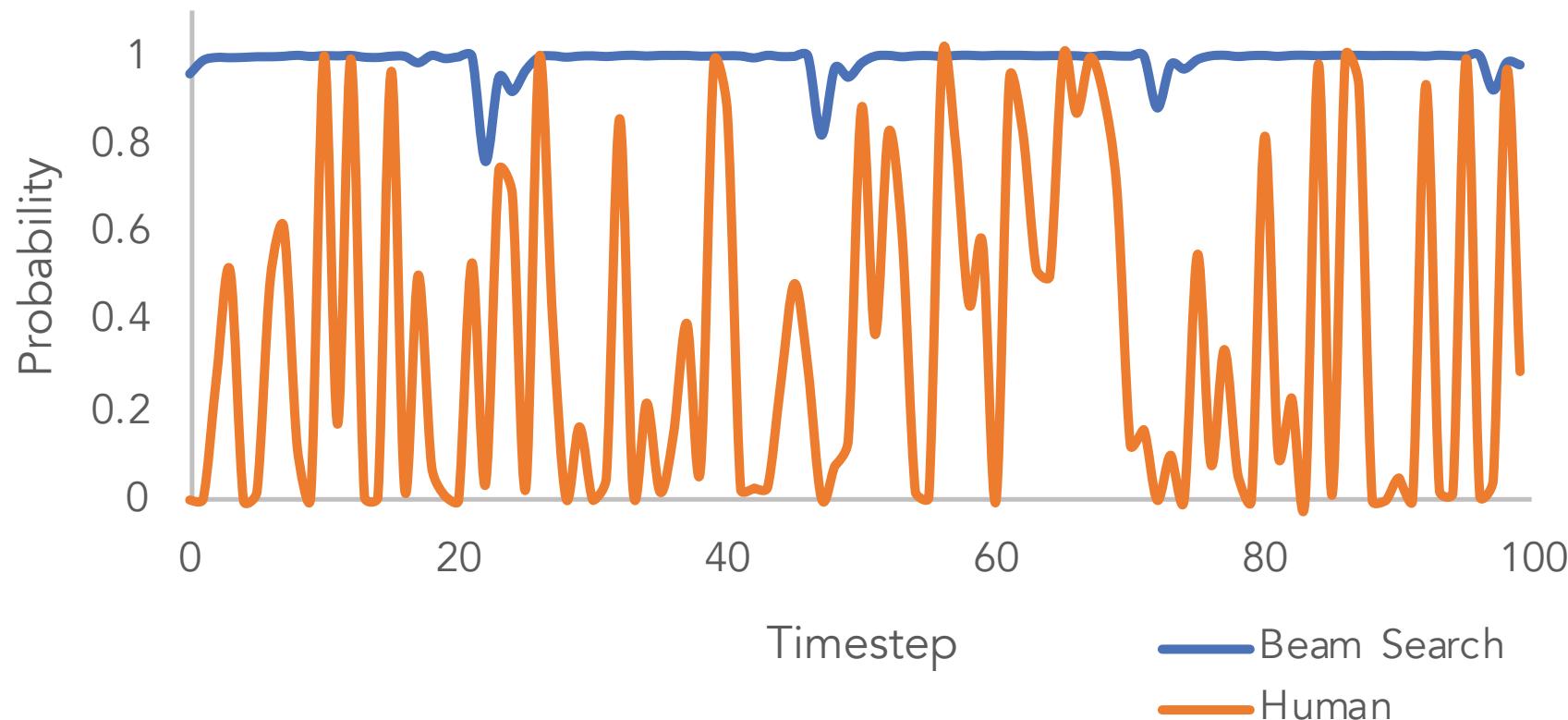
□ Simple option:

- Heuristic: Don't repeat n-grams

□ More complex:

- Use a different training objective:
 - Coverage loss (See et al., 2017)
 - Prevents attention mechanism from attending to the same words
 - Unlikelihood objective (Welleck et al., 2020)
 - Penalize generation of already-seen tokens
- Use a different decoding objective:
 - Contrastive decoding (Li et al, 2022)
 - Searches for strings x that maximize $\text{logprob_largeLM}(x) - \text{logprob_smallLM}(x)$

Is finding the most likely string reasonable for open-ended generation?



It fails to match the uncertainty distribution for human generated text.

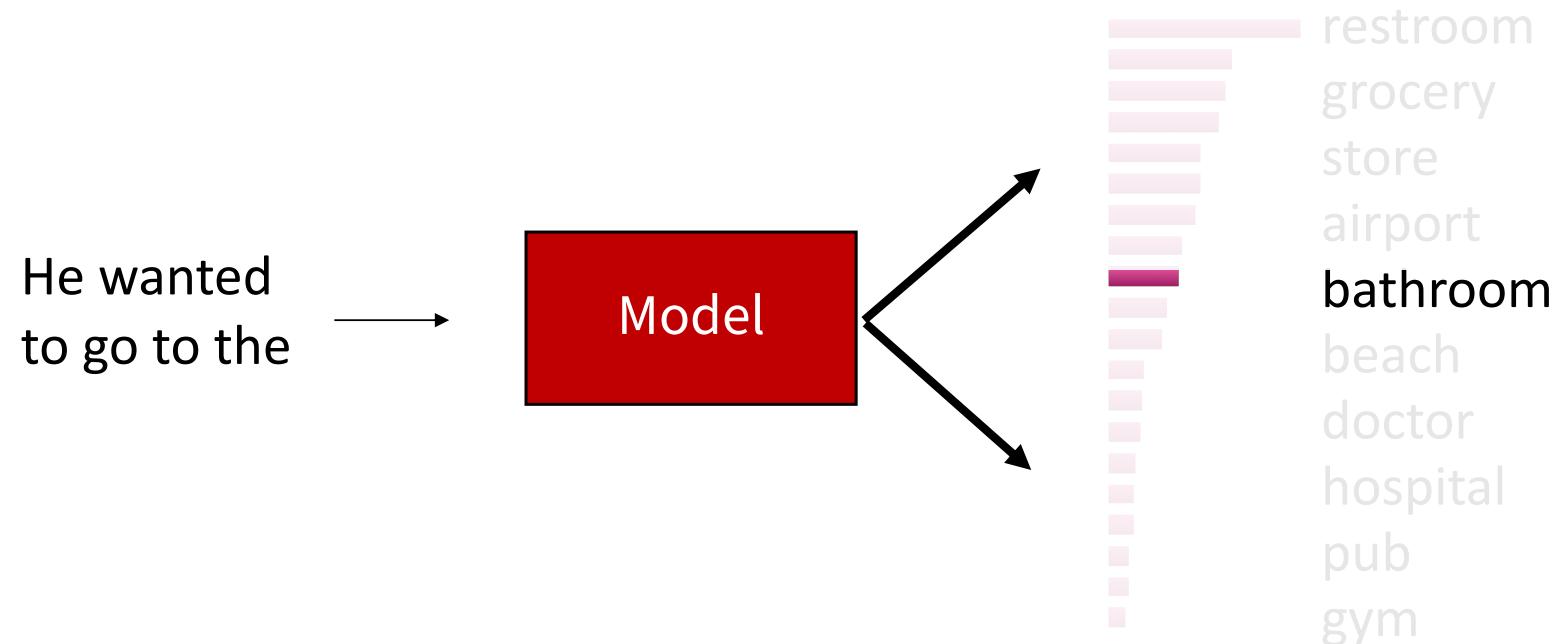
(Holtzman et. al., ICLR 2020)

Time to get random : Sampling!

- Sample a token from the distribution of tokens

$$\hat{y}_t \sim P(y_t = w | \{y\}_{<t})$$

- It's *random* so you can sample any token!



Decoding: Top- k sampling

❑ Problem: Vanilla sampling makes every token in the vocabulary an option

- Even if most of the **probability mass** in the distribution is over a limited set of options, the tail of the distribution could be very long and in aggregate have considerable mass (statistics speak: we have “**heavy tailed**” distributions)
- Many tokens are probably *really wrong* in the current context
- For these wrong tokens, we give them *individually* a tiny chance to be selected.
- But because there are many of them, we still give them *as a group* a high chance to be selected.

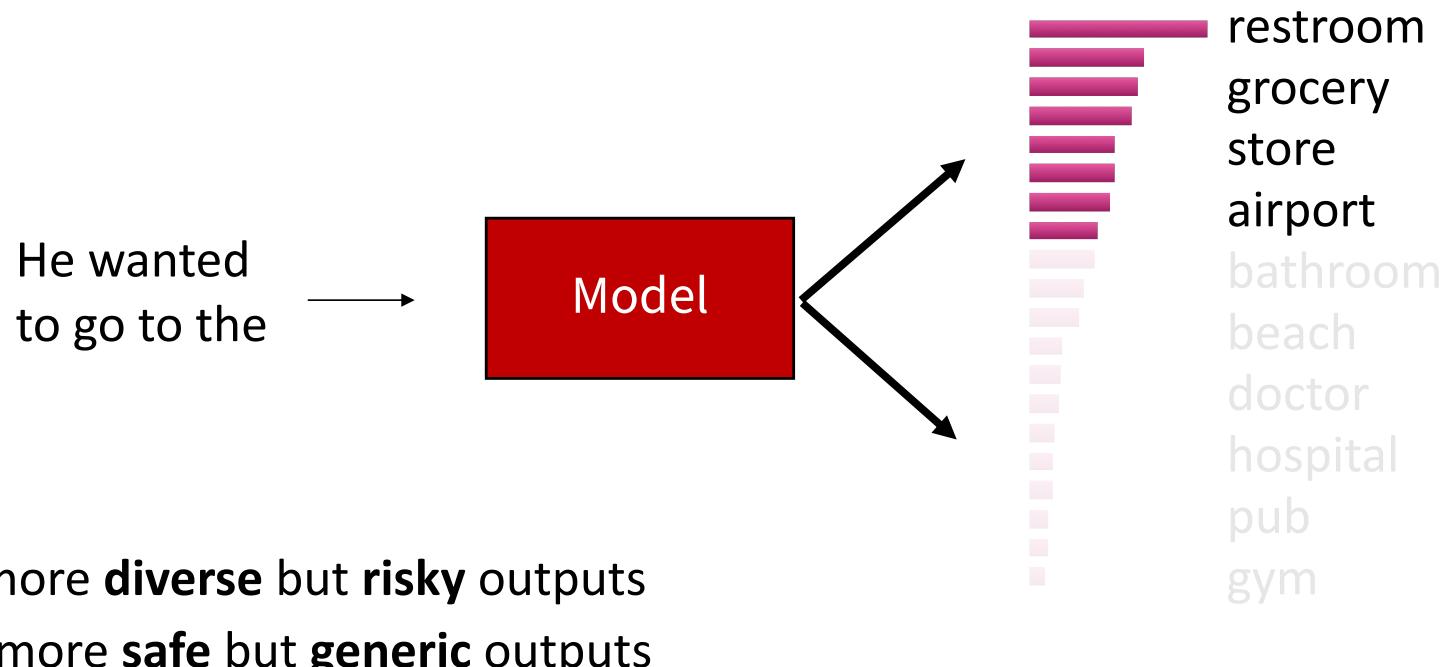
❑ Solution: Top- k sampling

- Only sample from the top k tokens in the probability distribution

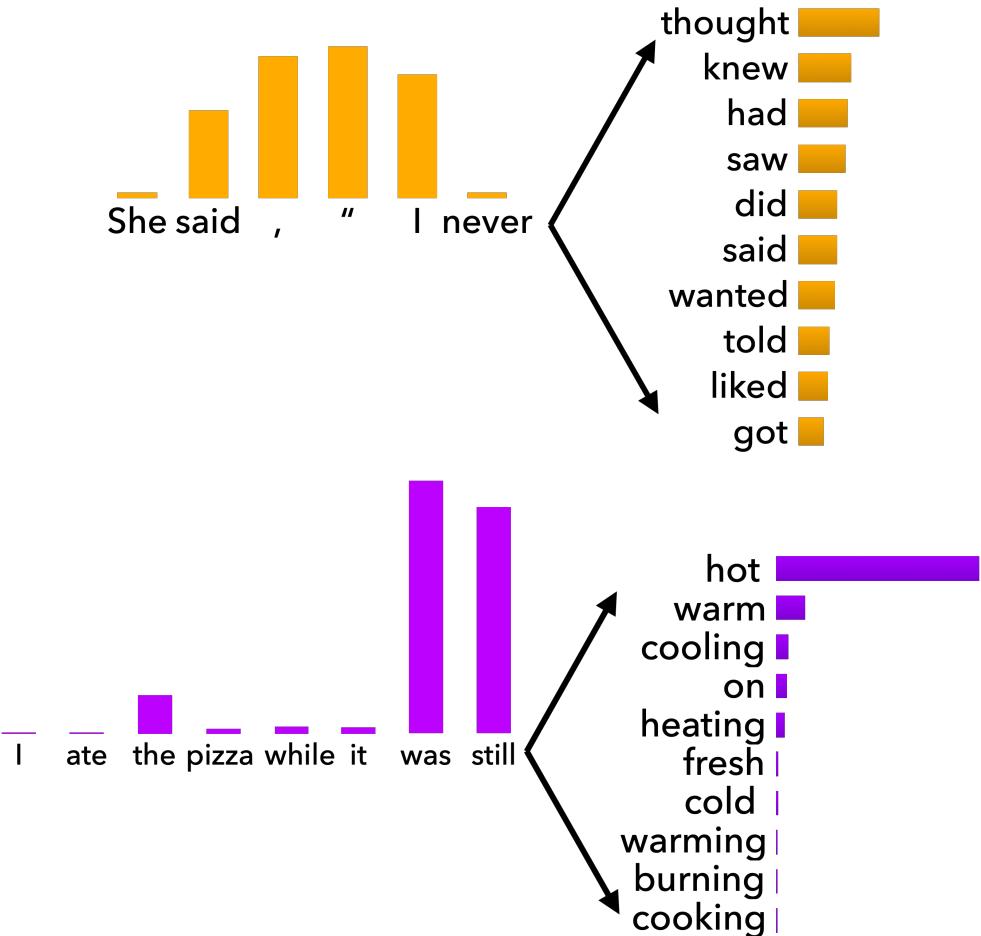
Decoding: Top- k sampling

□ Solution: Top- k sampling

- Only sample from the top k tokens in the probability distribution
- Common values are $k = 5, 10, 20$ (but it's up to you!)



Issues with Top- k sampling



Top- k sampling can cut off too *quickly*!

Top- k sampling can also cut off too *slowly*!

Decoding: Top- p (nucleus) sampling

- **Problem:** The probability distributions we sample from are dynamic
 - When the distribution P_t is flatter, a limited k removes many viable options
 - When the distribution P_t is peakier, a high k allows for too many options to have a chance of being selected

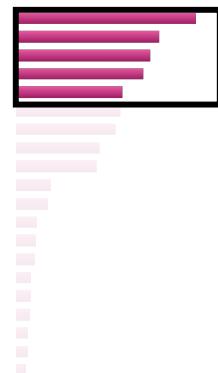
- **Solution:** Top- p sampling
 - Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
 - Varies k depending on the uniformity of P_t

Decoding: Top- p (nucleus) sampling

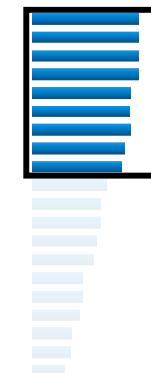
□ Solution: Top- p sampling

- Sample from all tokens in the top p cumulative probability mass (i.e., where mass is concentrated)
- Varies k depending on the uniformity of P_t

$$P_t^1(y_t = w | \{y\}_{<t})$$



$$P_t^2(y_t = w | \{y\}_{<t})$$



$$P_t^3(y_t = w | \{y\}_{<t})$$



Scaling randomness: Softmax temperature

- **Recall:** On timestep t , the model computes a prob distribution P_t by applying the softmax function to a vector of scores $S \in \mathbb{R}^{|V|}$

$$P_t(y_t = w) = \frac{\exp(S_w)}{\sum_{w' \in V} \exp(S_{w'})}$$

- You can apply a temperature hyperparameter τ to the softmax to rebalance P_t :

$$P_t(y_t = w) = \frac{\exp(S_w/\tau)}{\sum_{w' \in V} \exp(S_{w'}/\tau)}$$

- Raise the temperature $\tau > 1$: P_t becomes more uniform

- More diverse output (probability is spread around vocab)

- Lower the temperature $\tau < 1$: P_t becomes more spiky

- Less diverse output (probability is concentrated on top words)

- Note: softmax temperature is not a decoding algorithm!

- Temperature is a hyperparameter for decoding:
It can be tuned for both beam search and sampling.

Improving decoding: Re-ranking

- **Problem:** What if I decode a bad sequence from my model?
- Decode a bunch of sequences
 - 10 candidates is a common number, but it's up to you
- Define a score to approximate quality of sequences and re-rank by this score
 - Simplest is to use (low) perplexity!
 - Careful! Remember that repetitive utterances generally get low perplexity.
 - Re-rankers can score a variety of properties:
 - style(Holtzman et al., 2018), discourse(Gabriele et al., 2021), entailment/factualty(Goya et al., 2020), logical consistency (Lu et al., 2020), and many more ...
 - Beware poorly-calibrated re-rankers
 - Can compose multiple re-rankers together.

Decoding: Takeaways

- ❑ Decoding is still a challenging problem in NLG – there's a lot more work to be done!
- ❑ Different decoding algorithms can allow us to inject biases that encourage different properties of coherent natural language generation
- ❑ Some of the most **impactful advances** in NLG of the last few years have come from **simple but effective** modifications to decoding algorithms

Training NLG models

Is repetition due to how LMs are trained?

Context:

In a shocking finding, scientist discovered a herd of unicorns living in a remote, previously unexplored valley, in the Andes Mountains. Even more surprising to the researchers was the fact that the unicorns spoke perfect English.

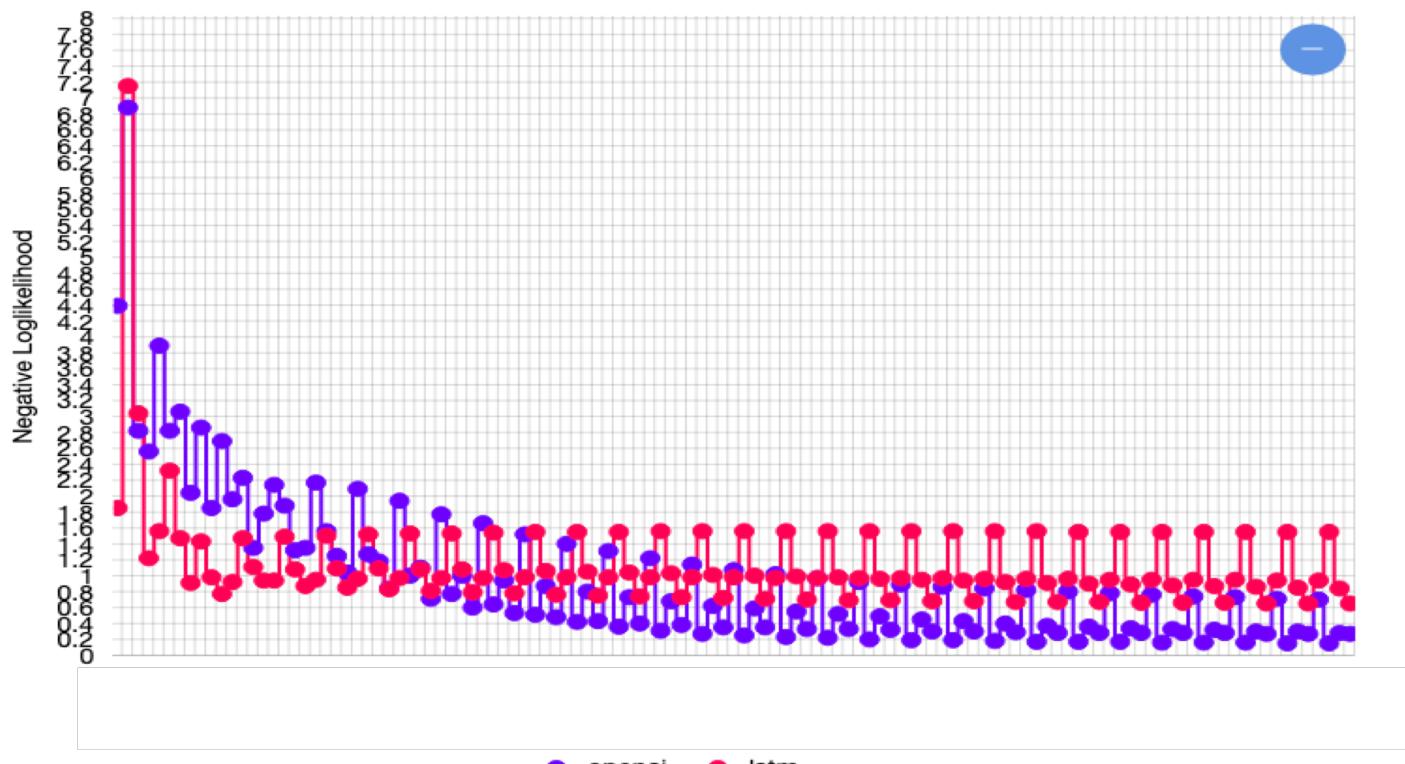
Continuation:

The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the **Universidad Nacional Autónoma de México (UNAM)** and **the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México...)**

Diversity Issues

- Maximum Likelihood Estimation model learns bad mode of the text distribution.

I'm tired. I'm tired.



(Holtzman et. al., ICLR 2020)

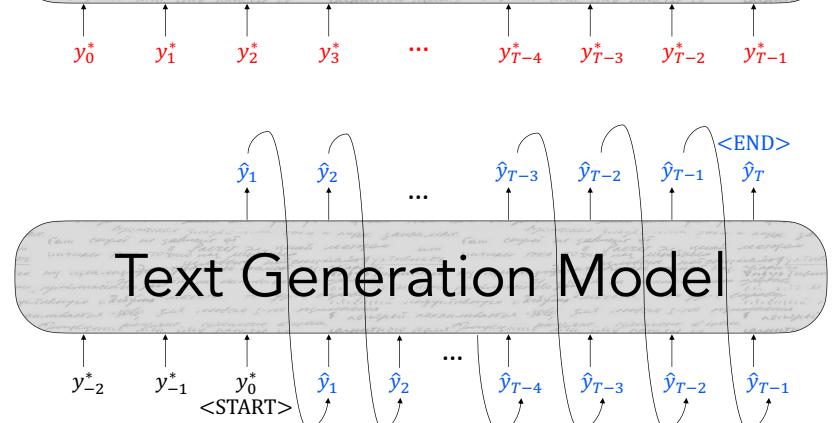
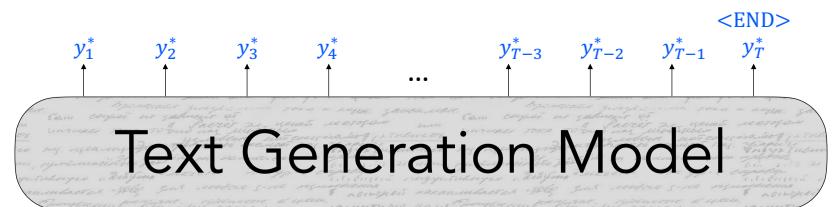
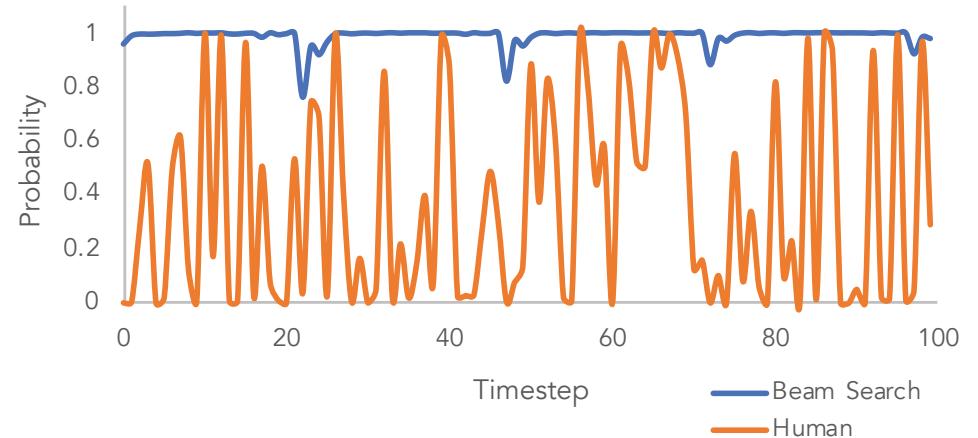
Exposure Bias

- Training with teacher forcing leads to **exposure bias** at generation time
 - During training, our model's inputs are gold context tokens from real, human-generated texts

$$\mathcal{L}_{MLE} = -\log P(y_t^* | \{y^*\}_{<t})$$

- At generation time, our model's inputs are previously-decoded tokens

$$\mathcal{L}_{dec} = -\log P(\hat{y}_t | \{\hat{y}\}_{<t})$$



Exposure Bias Solutions

❑ Scheduled sampling (Bengio et al., 2015)

- With some probability p , **decode a token** and feed that as the next input, rather than the **gold token**.
- Increase p over the course of training
- Leads to improvements in practice, but can lead to **strange training objectives**

❑ Dataset Aggregation (DAgger; Ross et al., 2011)

- At various intervals during training, generate sequences from your current model
- **Add these sequences** to your training set as additional examples

Exposure Bias Solutions

❑ Retrieval Augmentation (Guu*, Hashimoto*, et al., 2018)

- Learn to retrieve a sequence from an existing corpus of human-written prototypes (e.g., dialogue responses)
- Learn to edit the retrieved sequence by adding, removing, and modifying tokens in the prototype – this will still result in a more “human-like” generation

❑ Reinforcement Learning: cast your text generation model as a *Markov decision process*

- **State s** is the model’s representation of the preceding context
- **Actions a** are the words that can be generated
- **Policy π** is the decoder
- **Rewards r** are provided by an external score
- Learn behaviors by rewarding the model when it exhibits them

Reward Estimation

- ❑ How should we define a reward function? Just use your evaluation metric!
 - **BLEU** (machine translation; Ranzato et al., ICLR 2016; Wu et al., 2016)
 - **ROUGE** (summarization; Paulus et al., ICLR 2018; Celikyilmaz et al., NAACL 2018)
 - CIDEr (image captioning; Rennie et al., CVPR 2017)
 - SPIDER (image captioning; Liu et al., ICCV 2017)

- ❑ Be careful about **optimizing for the task** as opposed to “**gaming**” the reward!
 - Evaluation metrics are merely proxies for generation quality!
 - “even though RL refinement can achieve better BLEU scores, it barely improves the human impression of the translation quality” – Wu et al., 2016

Reward Estimation

- What behaviors can we tie to rewards?
 - Cross-modality consistency in image captioning (Ren et al., CVPR 2017)
 - Sentence simplicity (Zhang and Lapata, EMNLP 2017)
 - Temporal Consistency (Bosselut et al., NAACL 2018)
 - Utterance Politeness (Tan et al., TACL 2018)
 - Paraphrasing (Li et al., EMNLP 2018)
 - Sentiment (Gong et al., NAACL 2019)
 - Formality (Gong et al., NAACL 2019)

Reward Estimation

- ❑ Human Preference (RLHF): this is the technique behind ChatGPT!
 - (Ziegler et al. 2019, Stiennon et al., 2020)
 - Human ranking the generated text based on their preference.
 - Learn a reward function of the human preference.
 - We will talk about it more later
- ❑ If you can formalize a behavior as a Python function (or train a neural network to approximate it!), you can train a text generation model to exhibit that behavior!

Training: Takeaways

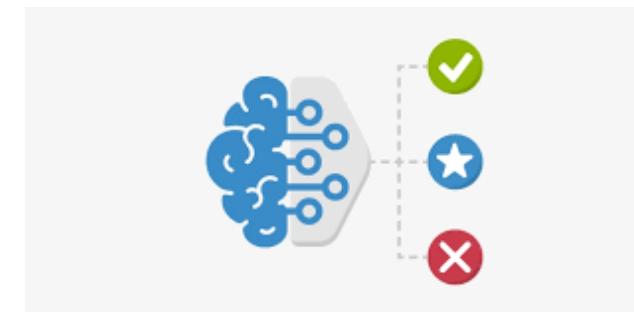
- ❑ *Teacher forcing* is still the main algorithm for training text generation models
- ❑ **Diversity** is an issue with sequences generated from teacher forced models
 - New approaches focus on mitigating the effects of common words
- ❑ **Exposure bias** causes text generation models to **lose coherence** easily
 - Models must learn to recover from their own bad samples
 - e.g., scheduled sampling, DAgger
 - Or not be allowed to generate bad text to begin with (e.g., retrieval + generation)
- ❑ Training with RL can allow models to learn behaviors that are preferred by human preference / metrics

Evaluating NLG Systems

Types of evaluation methods for text generation

Ref: They walked **to the grocery store** .
Gen: **The woman went to the hardware store** .

Content Overlap Metrics



Model-based Metrics



Human Evaluations

Content overlap metrics

Ref: They walked **to the grocery store** .

Gen: **The woman went to the hardware store** .



- Compute a score that indicates the lexical similarity between *generated* and *gold-standard (human-written)* text
- Fast and efficient and widely used
- N-gram* overlap metrics (e.g., **BLEU**, ROUGE, METEOR, CIDEr, etc.)

N-gram overlap metrics

Word overlap-based metrics (BLEU, ROUGE, METEOR, CIDEr, etc.)

- ❑ They're not ideal for machine translation
- ❑ They get progressively **much worse** for tasks that are more open-ended than machine translation
 - Worse for summarization, as longer output texts are harder to measure
 - Much worse for dialogue, which is more open-ended than summarization
 - Much, much worse story generation, which is also open-ended, but whose sequence length can make it seem you're getting decent scores!

A simple failure case

- ❑ n-gram overlap metrics have no concept of semantic relatedness!

Are you enjoying the NLP lectures?

Heck yes !



Score:

0.61

Yes !

0.25

You know it !

False negative

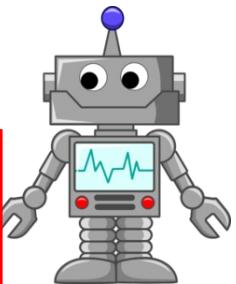
0

Yup .

False positive

0.67

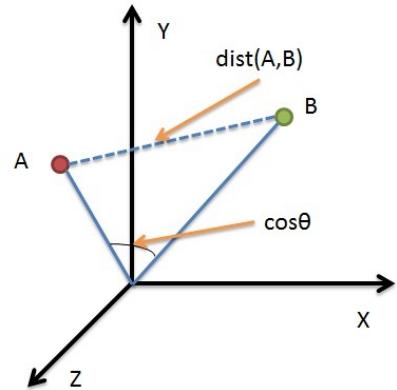
Heck no !



Model-based metrics to capture more semantics

- ❑ Use **learned representations** of words and sentences to compute semantic similarity between generated and reference texts
- ❑ No more **n-gram bottleneck** because text units are represented as **embeddings!**
- ❑ The embeddings are **pretrained**, distance metrics used to measure the similarity can be **fixed**

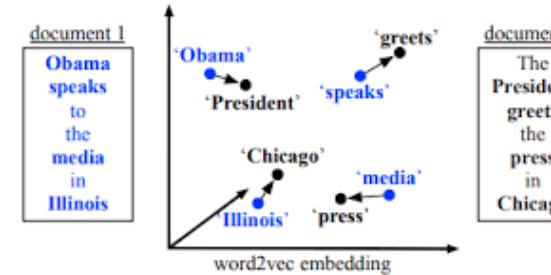
Model-based metrics: Word distance functions



Vector Similarity

Embedding based similarity for semantic distance between text.

- Embedding Average (Liu et al., 2016)
- Vector Extrema (Liu et al., 2016)
- MEANT (Lo, 2017)
- YISI (Lo, 2019)



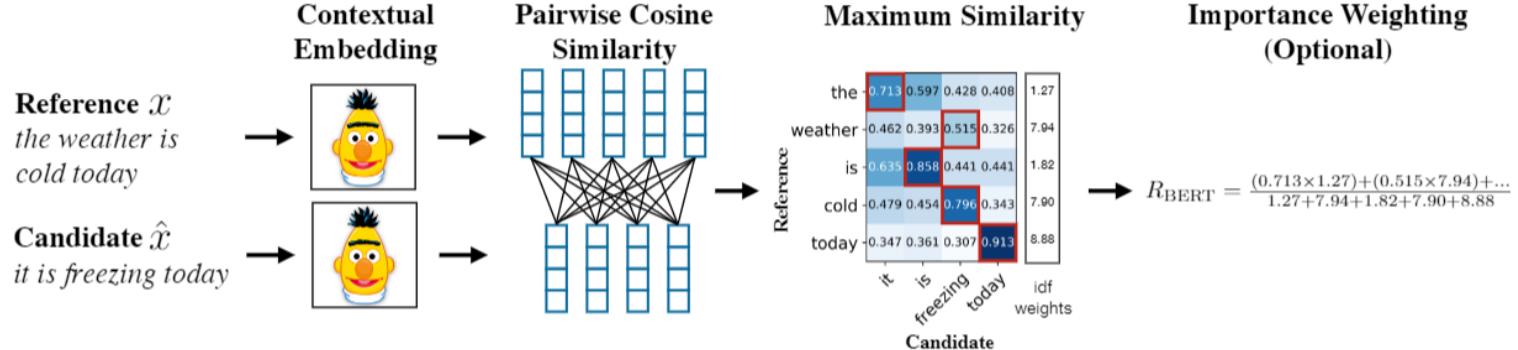
Word Mover's Distance

Measures the distance between two sequences (e.g., sentences, paragraphs, etc.), using word embedding similarity matching.

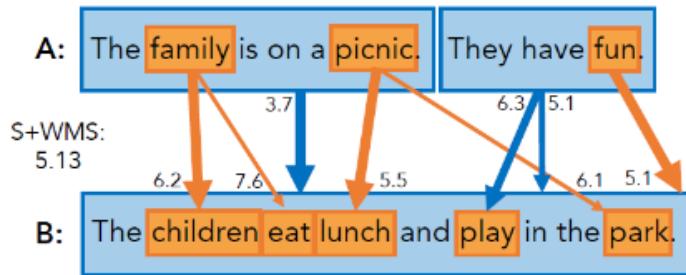
(Kusner et.al., 2015; Zhao et al., 2019)

BERTSCORE

Uses pre-trained contextual embeddings from BERT and matches words in candidate and reference sentences by cosine similarity.
(Zhang et.al. 2020)



Model-based metrics: Beyond word matching



Sentence Movers Similarity :

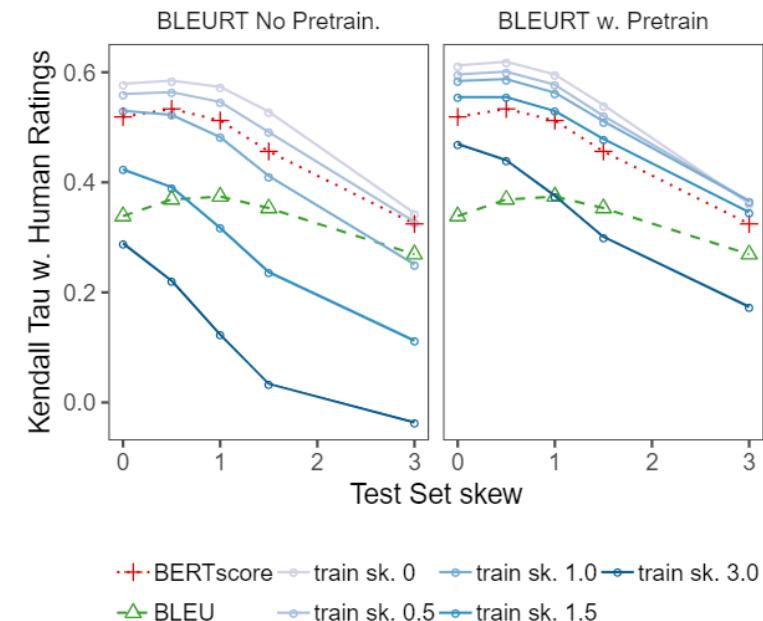
Based on Word Movers Distance to evaluate text in a continuous space using sentence embeddings from recurrent neural network representations.

(Clark et.al., 2019)

BLEURT:

A regression model based on BERT returns a score that indicates to what extent the candidate text is grammatical and conveys the meaning of the reference text.

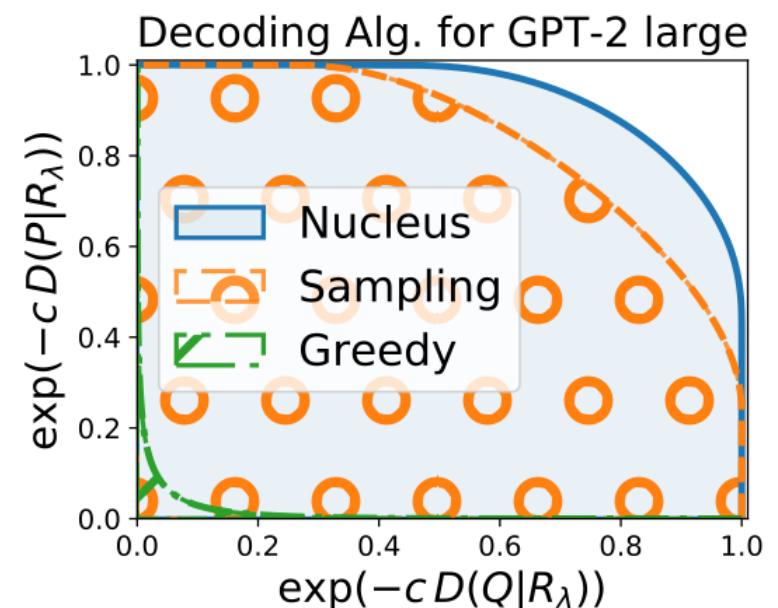
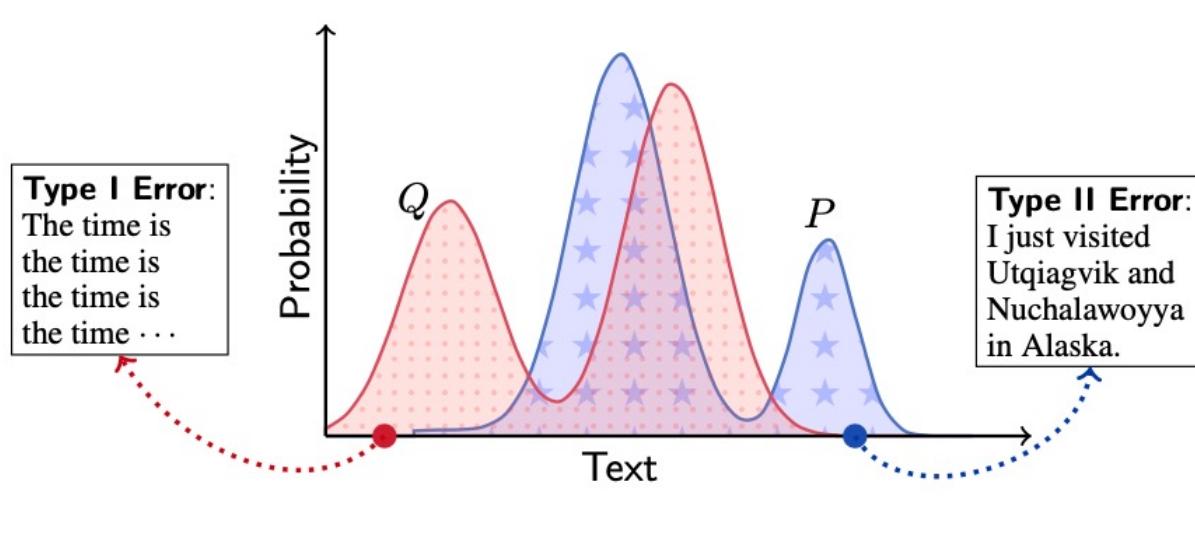
(Sellam et.al. 2020)



Evaluating Open-ended Text Generation

MAUVE

MAUVE computes information divergence in a quantized embedding space, between the generated text and the gold reference text (Pillutla et.al., 2022).



MAUVE (details)

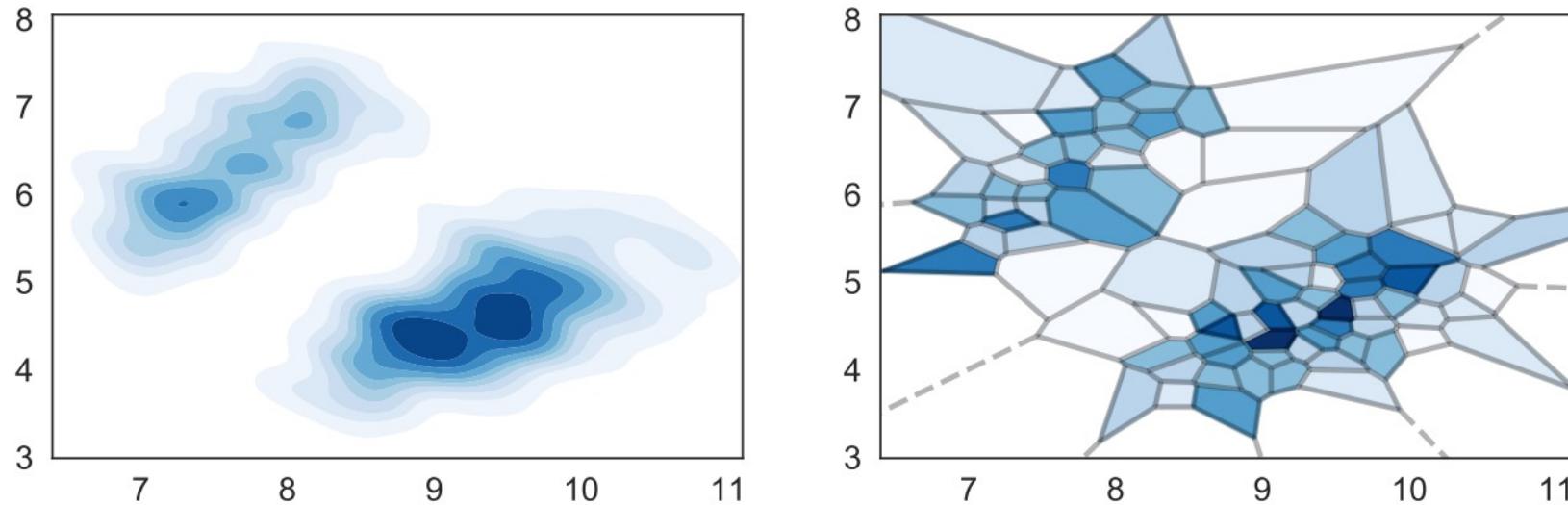


Figure 3: Illustration of the quantization. **Left:** A continuous two-dimensional distribution P . **Right:** A partitioning of the Euclidean plane \mathbb{R}^2 and the corresponding quantized distribution \tilde{P} .

How to evaluate an evaluation metric?

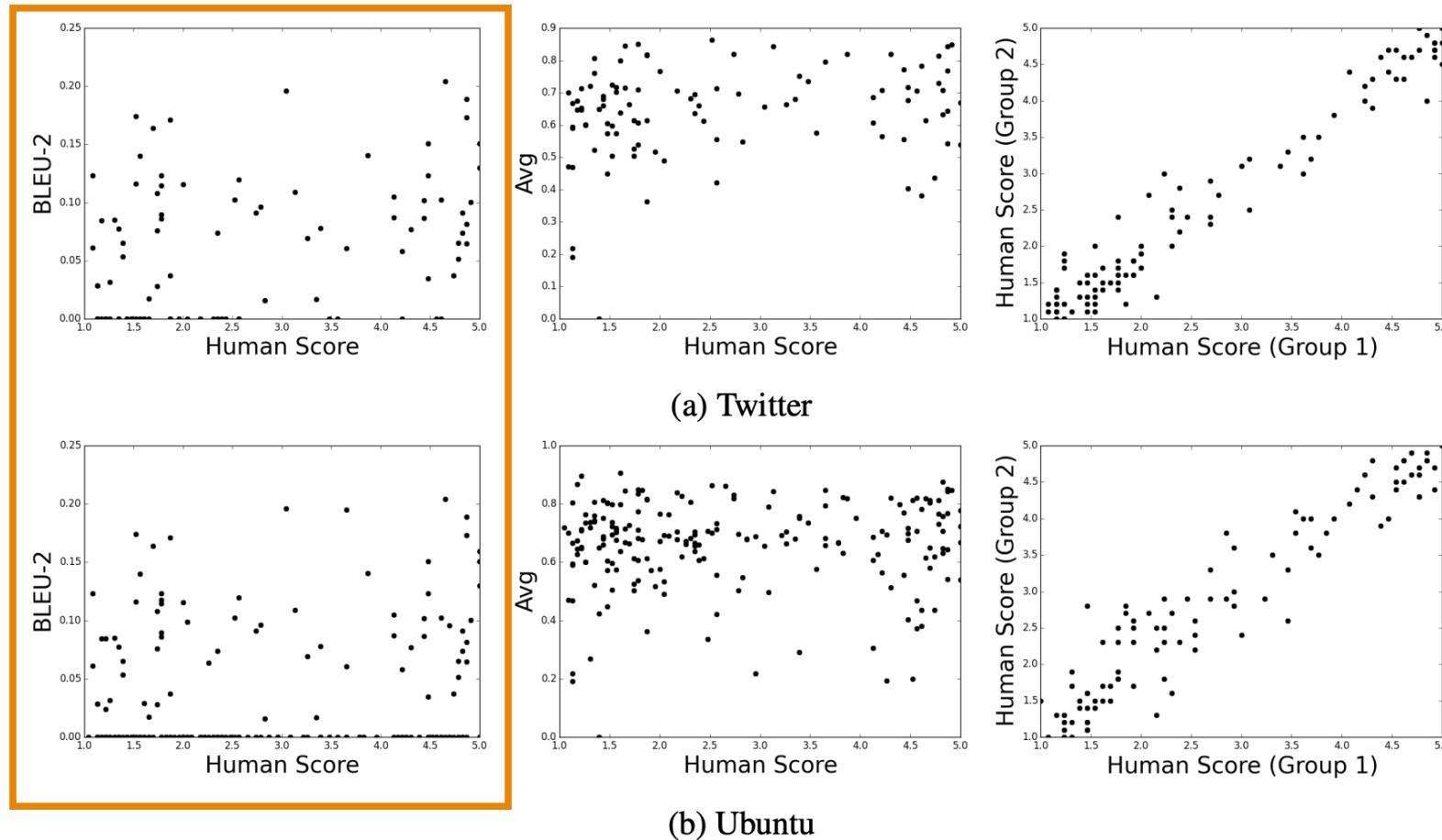


Figure 1: Scatter plots showing the correlation between metrics and human judgements on the Twitter corpus (a) and Ubuntu Dialogue Corpus (b). The plots represent BLEU-2 (left), embedding average (center), and correlation between two randomly selected halves of human respondents (right).

Human evaluations



- ❑ Automatic metrics fall short of matching human decisions
- ❑ Human evaluation is most important form of evaluation for text generation systems.
- ❑ Gold standard in developing new automatic metrics
 - New automated metrics must correlate well with human evaluations!

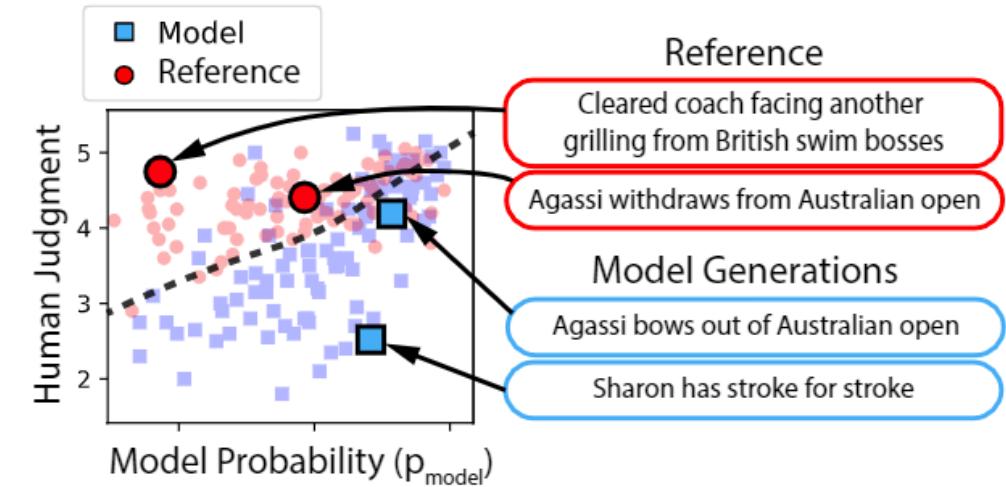
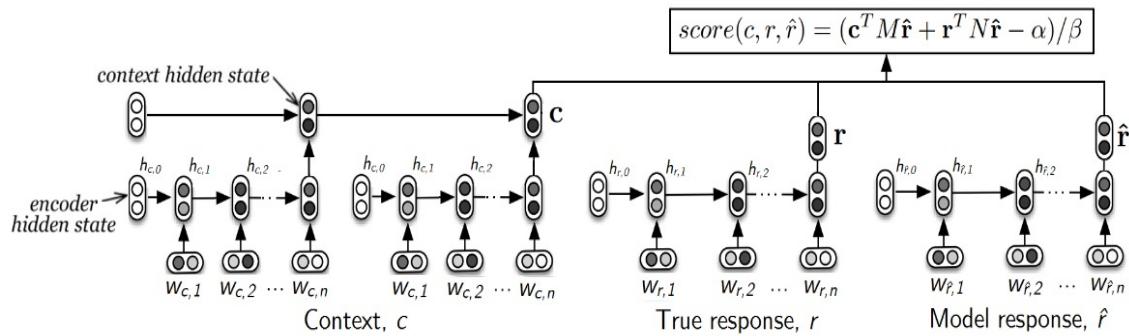
Human evaluations

- Ask humans to evaluate the quality of generated text
- Overall or along some specific dimension:
 - fluency
 - coherence / consistency
 - factuality and correctness
 - commonsense
 - style / formality
 - grammaticality
 - typicality
 - redundancy

Human evaluation: Issues

- Human judgments are regarded as the **gold standard**
- Of course, we know that human eval is **slow** and **expensive**
- Beyond the cost of human eval, it's still far from perfect:
- Humans Evaluation is hard:
 - Results are inconsistent / not reproducible
 - can be illogical
 - misinterpret your question
 - Precision not recall.

Learning from human feedback



ADEM:

A learned metric from human judgments for dialog system evaluation in a chatbot setting.

(Lowe et.al., 2017)

HUSE:

Human Unified with Statistical Evaluation (HUSE), determines the similarity of the output distribution and a human reference distribution.

(Hashimoto et.al. 2019)

Evaluation: Takeaways

- ❑ *Content overlap metrics* provide a good starting point for evaluating the quality of generated text, but they're **not good enough on their own**.
- ❑ *Model-based metrics* can be **more correlated with human judgment**, but behavior is **not interpretable**
- ❑ *Human judgments* are critical
 - But humans are **inconsistent!**
- ❑ In many cases, the best judge of output quality is **YOU!**
 - **Look at your model generations. Don't just rely on numbers!**
 - **Publicly release large samples of the output of systems that you create!**

Prompting

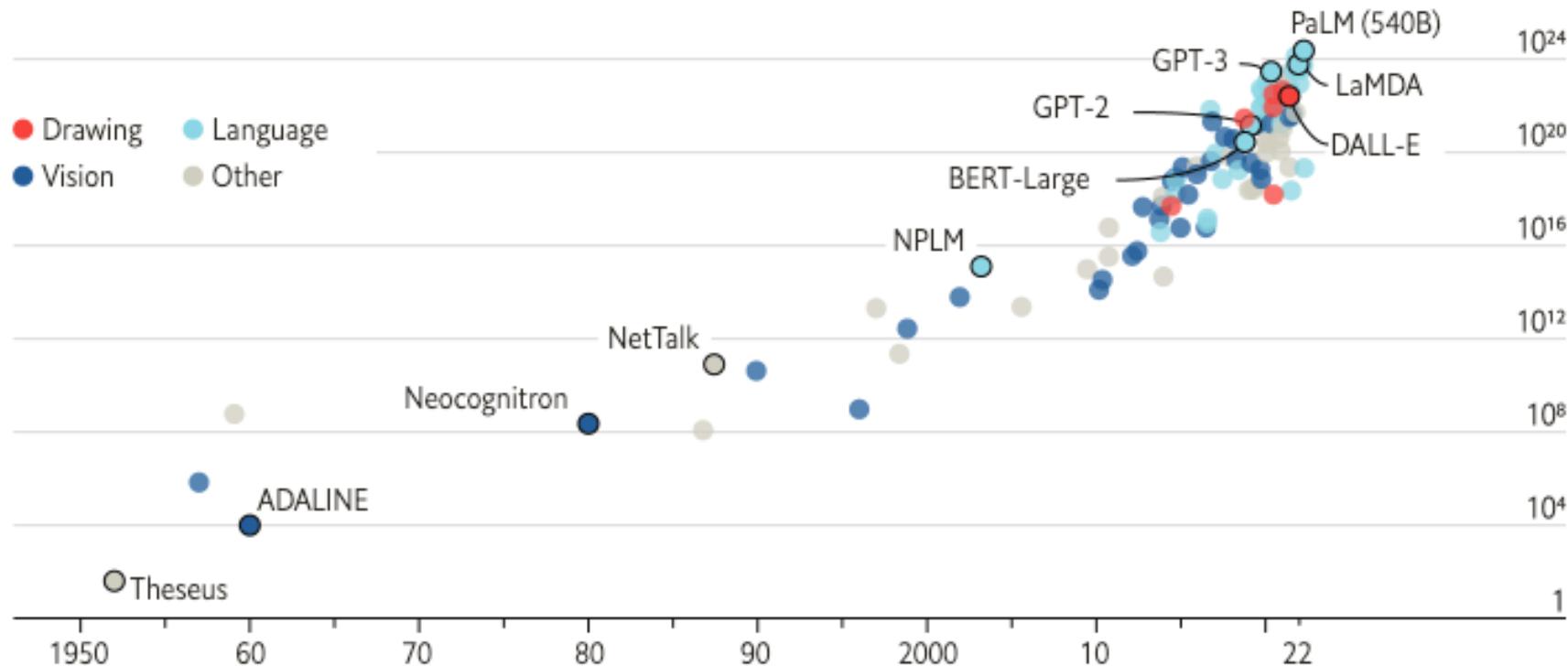
(Adapted from Jessy Mu's slides in Stanford 224n)

Larger and larger models

The blessings of scale

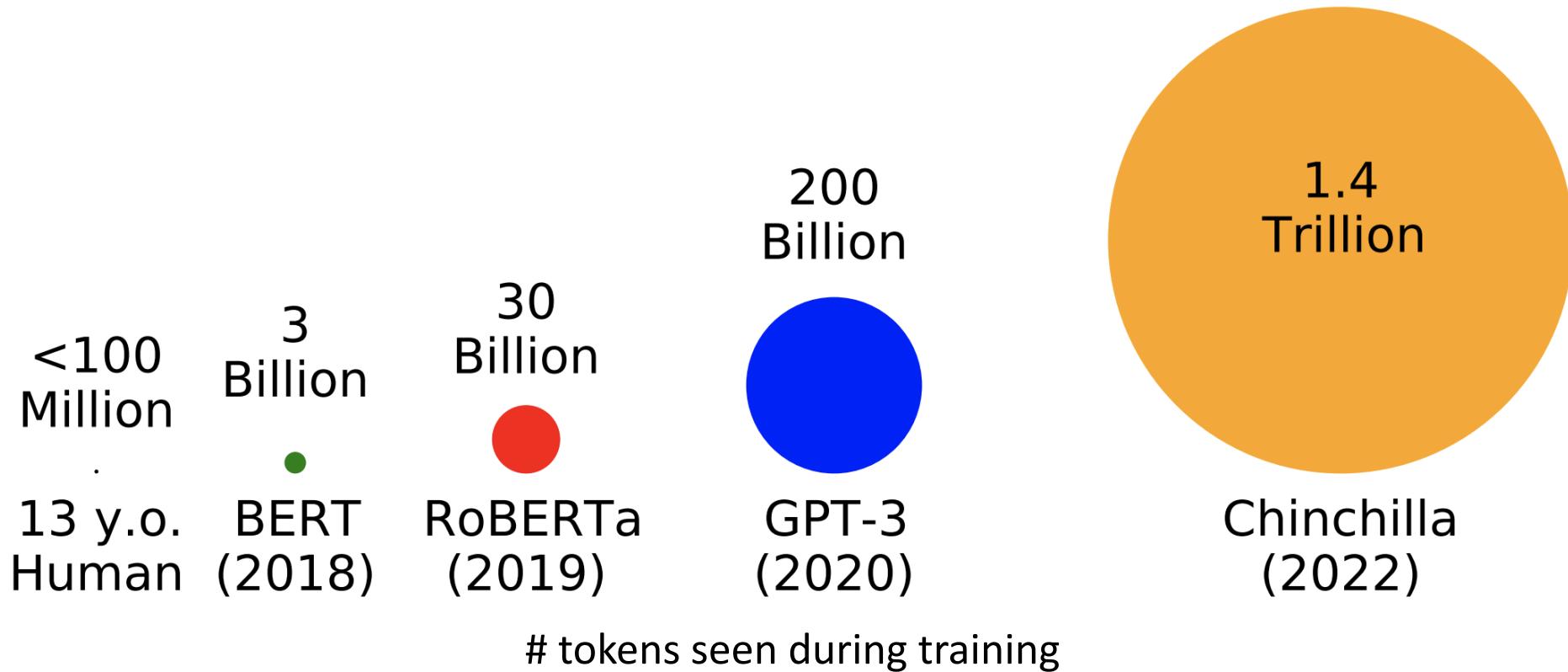
AI training runs, estimated computing resources used

Floating-point operations, selected systems, by type, log scale



Sources: "Compute trends across three eras of machine learning", by J. Sevilla et al., arXiv, 2022; Our World in Data

Trained on more and more data

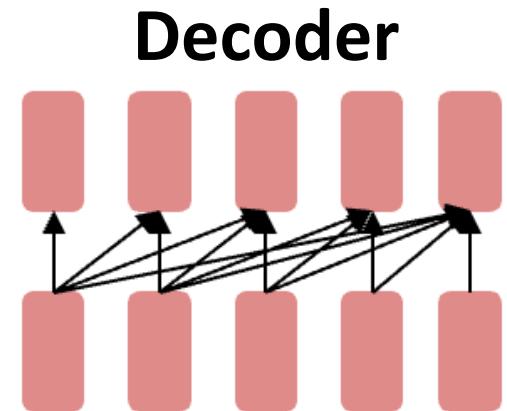


What kinds of things does pretraining learn? (review)

- I put ____ fork down on the table. [syntax]
- The woman walked across the street,
checking for traffic over ____ shoulder. [coreference]
- I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]
- Overall, the value I got from the two hours watching it was the sum total of the popcorn and the drink. The movie was _____. [sentiment]
- I was thinking about the sequence that goes
1, 1, 2, 3, 5, 8, 13, 21, _____ [some basic arithmetic]

Emergent abilities of large language models: GPT (2018)

- Let's revisit the Generative Pretrained Transformer (GPT) models from OpenAI as an example:
- GPT (117M parameters; Radford et al., 2018)
 - Transformer decoder with 12 layers.
 - Trained on BooksCorpus: over 7000 unique books (4.6GB text).
- Showed that language modeling at scale can be an effective pretraining technique for downstream tasks like natural language inference.



[START] *The man is in the doorway* [DELIM] *The person is near the door* [EXTRACT]

Emergent abilities of large language models: GPT-2 (2019)

- Let's revisit the Generative Pretrained Transformer (GPT) models from OpenAI as an example:
- GPT-2 (1.5B parameters; Radford et al., 2019)
 - Same architecture as GPT, just bigger (117M -> 1.5B)
 - But trained on **much more data**: 4GB -> 40GB of internet text data (WebText)
 - Scrape links posted on Reddit w/ at least 3 upvotes (rough proxy of human quality)

Language Models are Unsupervised Multitask Learners

Emergent zero-shot learning

- One key emergent ability in GPT-2 is **zero-shot learning**: the ability to do many tasks with **no examples**, and **no gradient updates**, by simply:
 - Specifying the right sequence prediction problem (e.g. question answering):

Passage: Tom Brady... Q: Where was Tom Brady born? A: ...

- Comparing probabilities of sequences (e.g. Winograd Schema Challenge [Levesque, 2011]):

The cat couldn't fit into the hat because it was too big.
Does it = the cat or the hat?

≡ Is $P(\dots \text{because } \mathbf{\text{the cat}} \text{ was too big}) \geq P(\dots \text{because } \mathbf{\text{the hat}} \text{ was too big})$?

Emergent zero-shot learning

- GPT-2 beats SoTA on language modeling benchmarks with no task-specific fine-tuning

Context: “Why?” “I would have thought you’d find him rather dry,” she said. “I don’t know about that,” said Gabriel.
“He was a great craftsman,” said Heather. “That he was,” said Flannery.

Target sentence: “And Polish, to boot,” said _____. **LAMBADA** (language modeling w/ long discourse dependencies)

Target word: Gabriel

[Paperno et al., 2016]

	LAMBADA (PPL)	LAMBADA (ACC)	CBT-CN (ACC)	CBT-NE (ACC)	WikiText2 (PPL)
SOTA	99.8	59.23	85.7	82.3	39.14
117M	35.13	45.99	87.65	83.4	29.41
345M	15.60	55.48	92.35	87.1	22.76
762M	10.87	60.12	93.45	88.0	19.93
1542M	8.63	63.24	93.30	89.05	18.34

[Radford et al., 2019]

Emergent abilities of large language models: GPT-3 (2020)

- GPT-3 (175B parameters; Brown et al., 2020)
 - Another increase in size (1.5B -> **175B**)
 - and data (40GB -> **over 600GB**)

Language Models are Few-Shot Learners

Tom B. Brown*

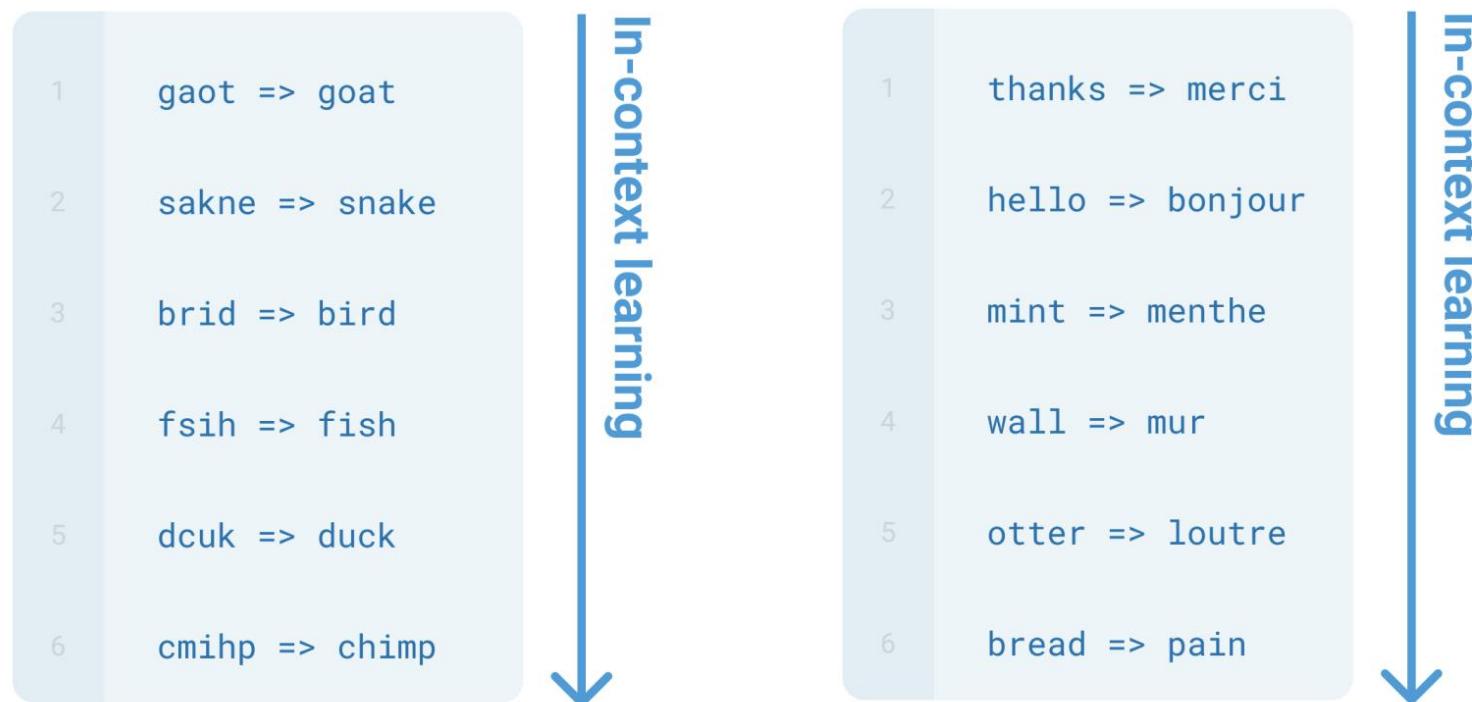
Benjamin Mann*

Nick Ryder*

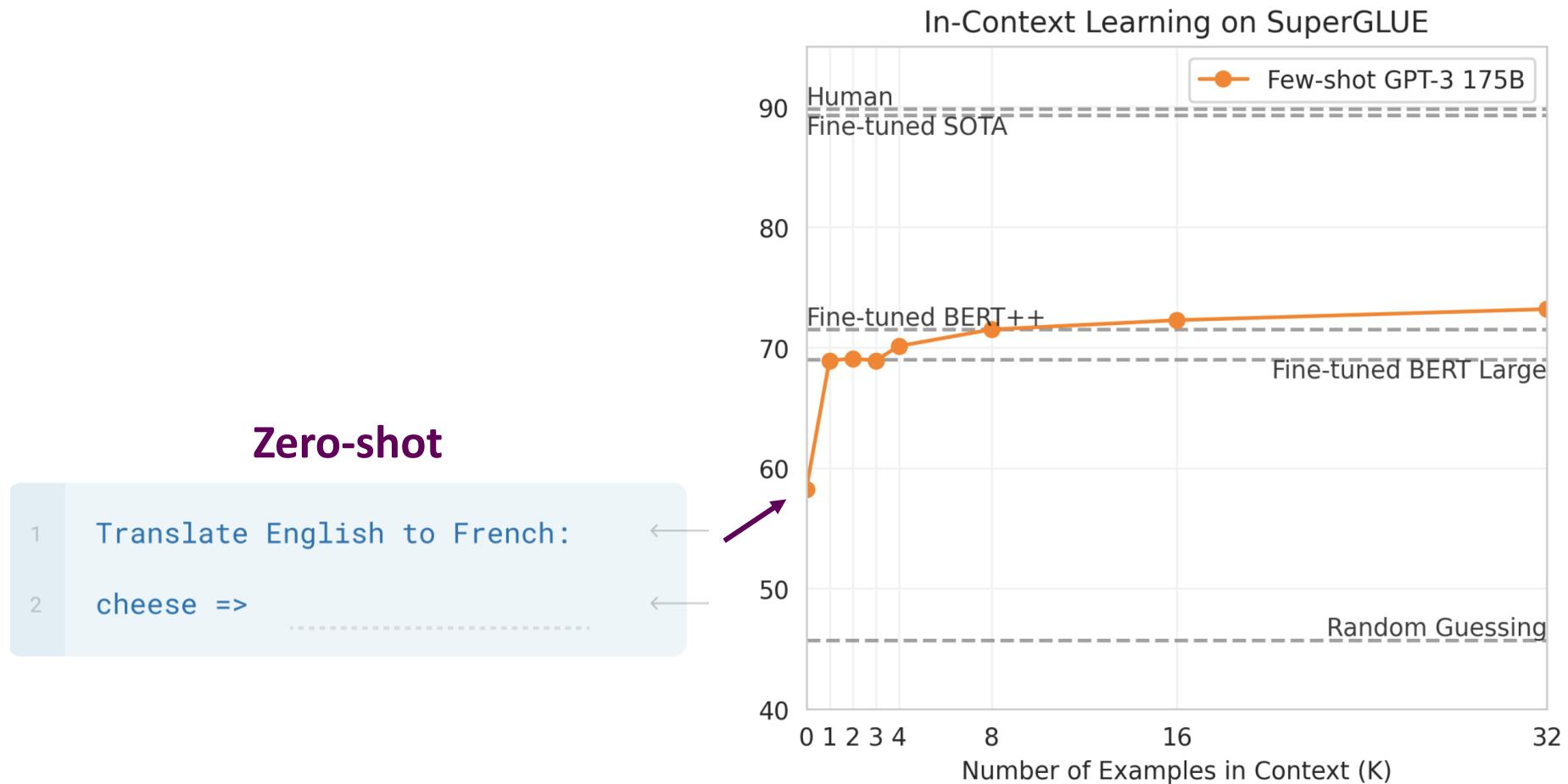
Melanie Subbiah*

Emergent few-shot learning

- Specify a task by simply **prepend**ing examples of the task before your example
- Also called **in-context learning**, to stress that *no gradient updates* are performed when learning a new task



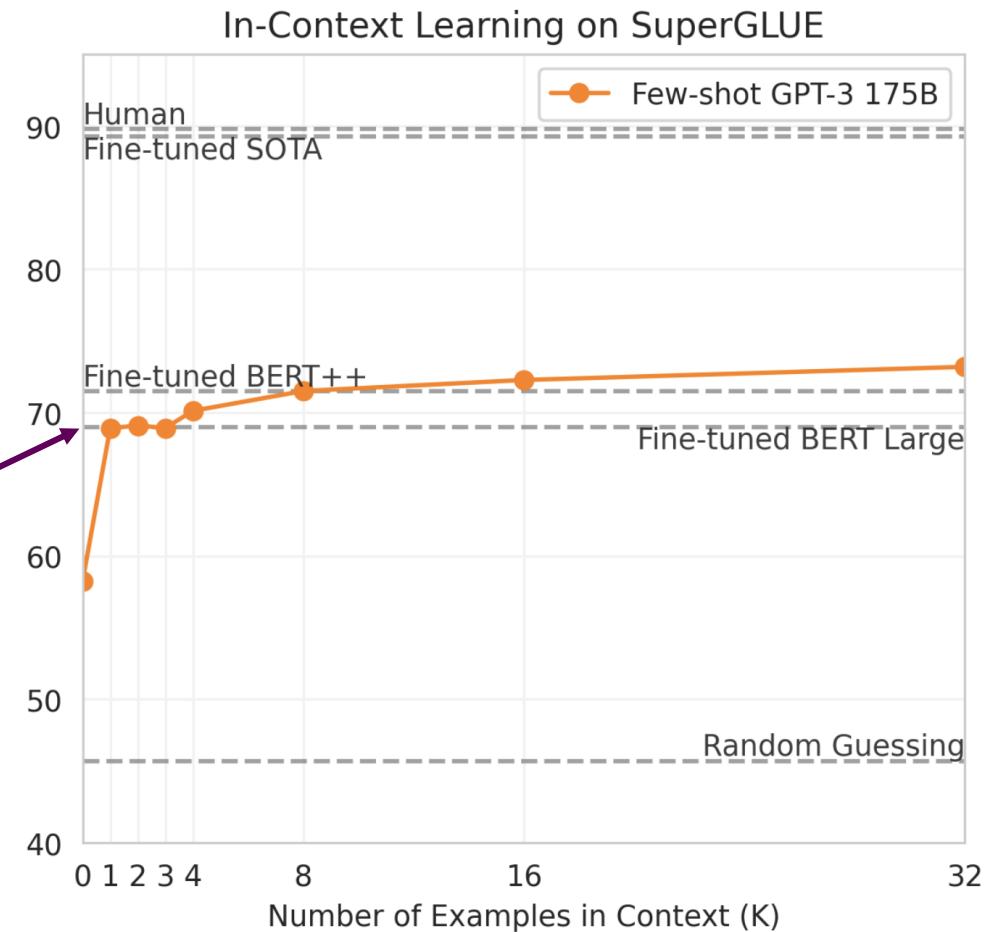
Emergent few-shot learning



Emergent few-shot learning

One-shot

```
1 Translate English to French:  
2 sea otter => loutre de mer  
3 cheese =>
```



Emergent few-shot learning

Few-shot

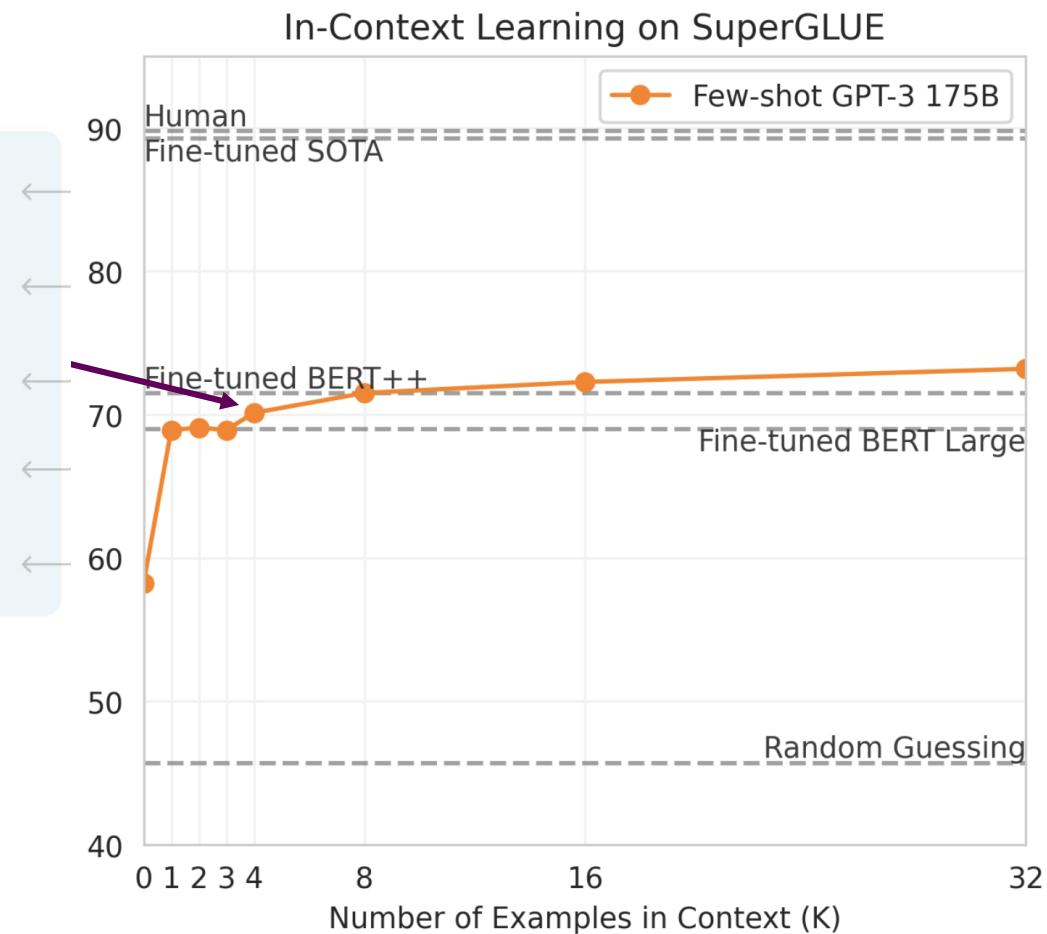
1 Translate English to French:

2 sea otter => loutre de mer

3 peppermint => menthe poivrée

4 plush girafe => girafe peluche

5 cheese =>



Few-shot learning is an emergent property of model scale

Cycle letters:

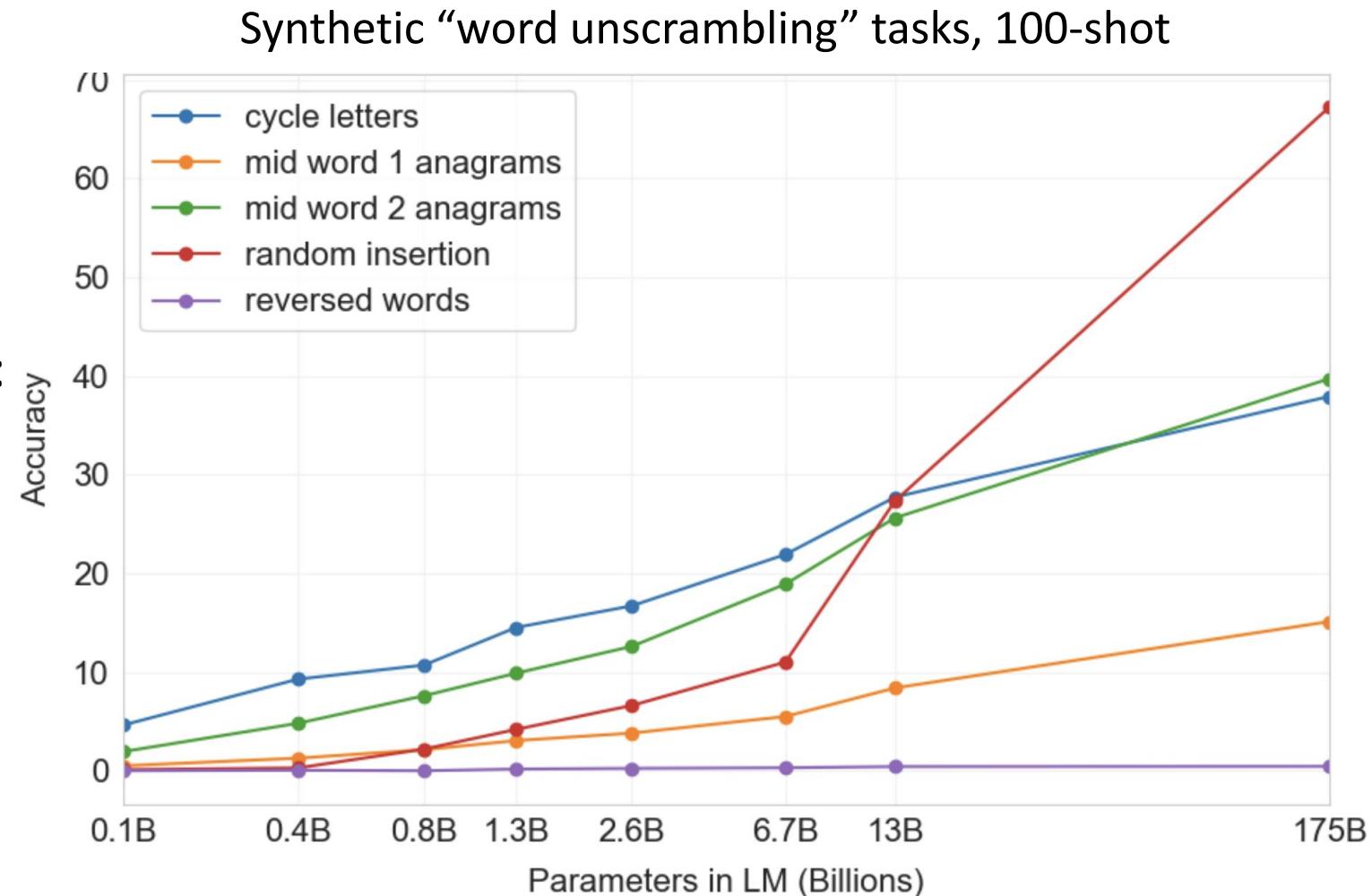
pleap ->
apple

Random insertion:

a.p!p/l!e ->
apple

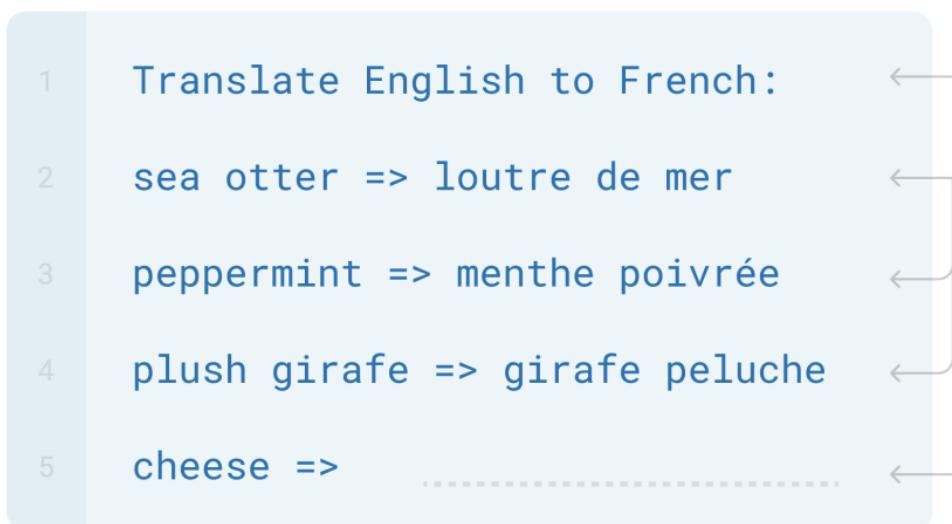
Reversed words:

elppa ->
apple



New methods of “prompting” LMs

Zero/few-shot prompting



Traditional fine-tuning



[Brown et al., 2020]

Limits of prompting for harder tasks?

- Some tasks seem too hard for even large LMs to learn through prompting alone.
Especially tasks involving richer, multi-step reasoning.
(Humans struggle at these tasks too!)

$$19583 + 29534 = 49117$$

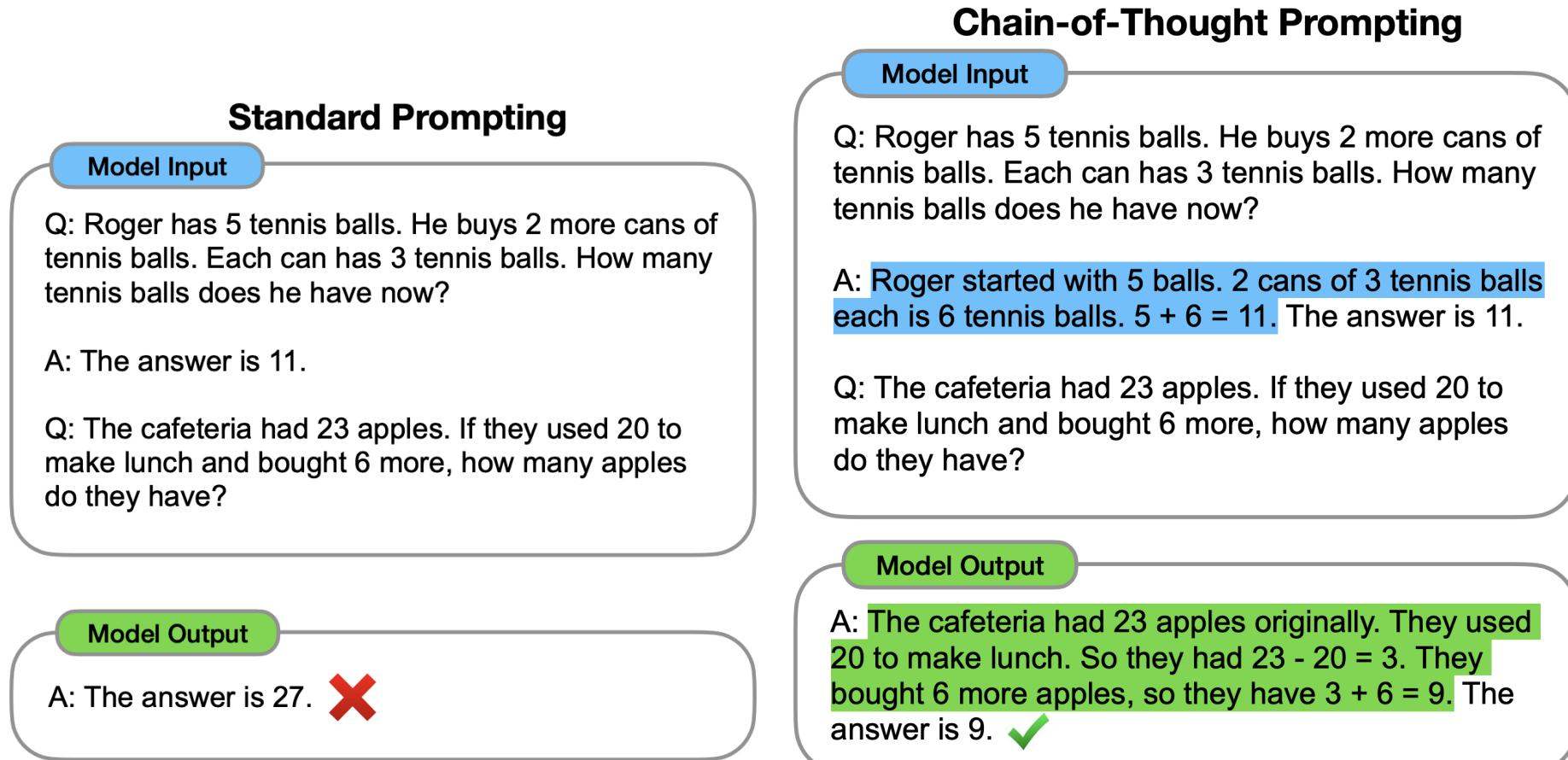
$$98394 + 49384 = 147778$$

$$29382 + 12347 = 41729$$

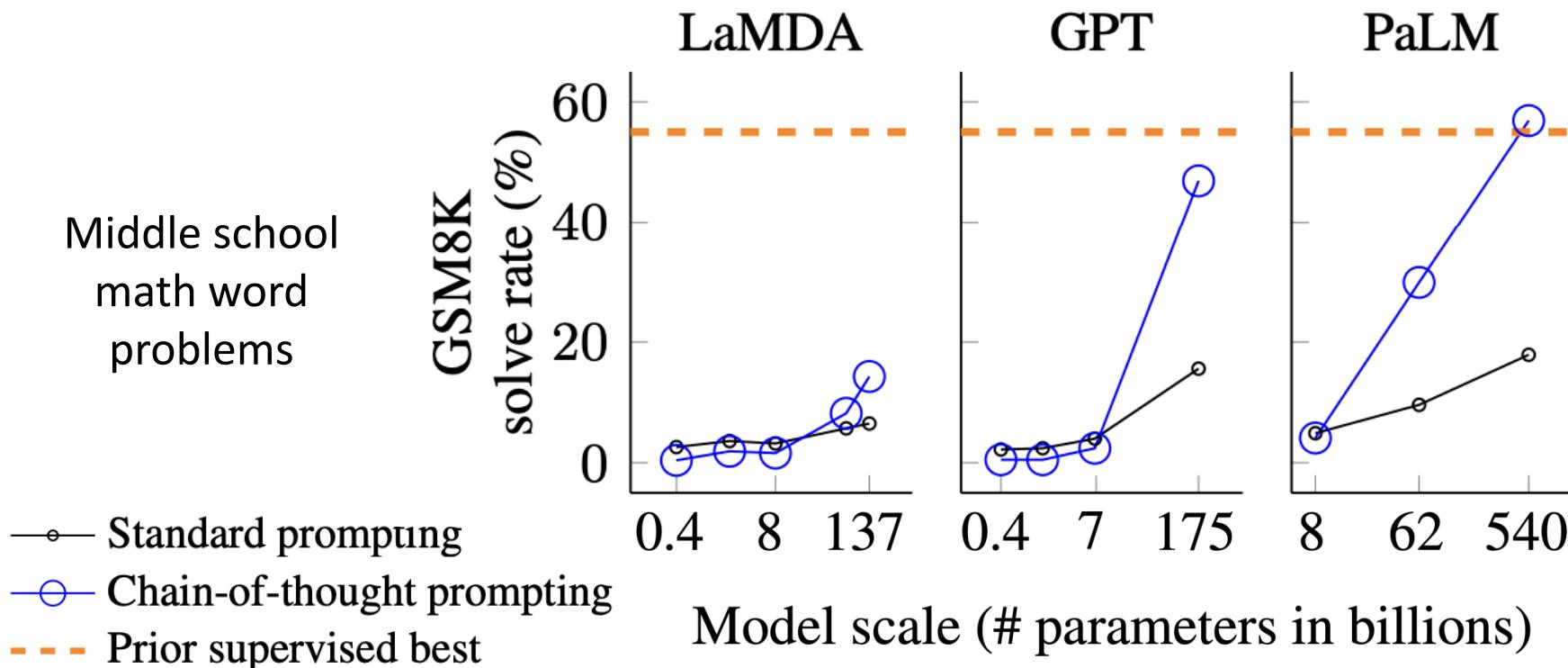
$$93847 + 39299 = ?$$

Solution: change the prompt!

Chain-of-thought prompting

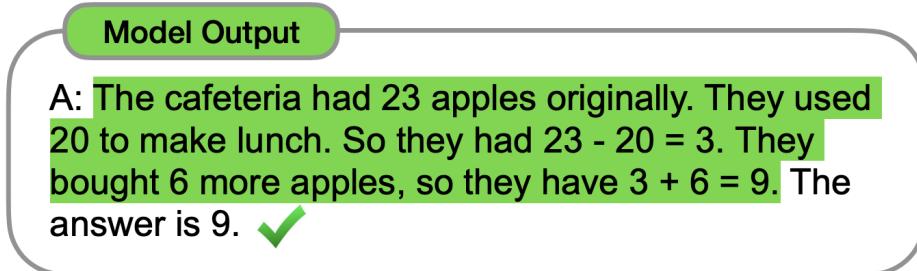
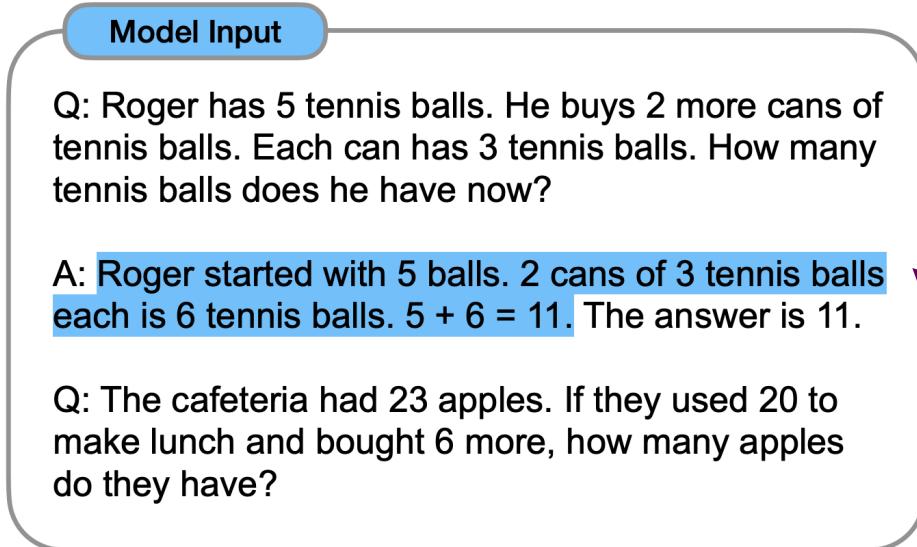


Chain-of-thought prompting is an emergent property of model scale



(Wei et al., 2022; also see Nye et al., 2021)

Chain-of-thought prompting



Do we even need examples of reasoning?
Can we just ask the model to reason through things?

Zero-shot chain-of-thought prompting

Model Input

Q: Roger has 5 tennis balls. He buys 2 more cans of tennis balls. Each can has 3 tennis balls. How many tennis balls does he have now?

A: Roger started with 5 balls. 2 cans of 3 tennis balls each is 6 tennis balls. $5 + 6 = 11$. The answer is 11.

Q: The cafeteria had 23 apples. If they used 20 to make lunch and bought 6 more, how many apples do they have?

Model Output

A: The cafeteria had 23 apples originally. They used 20 to make lunch. So they had $23 - 20 = 3$. They bought 6 more apples, so they have $3 + 6 = 9$. The answer is 9. ✓

Q: A juggler can juggle 16 balls. Half of the balls are golf balls, and half of the golf balls are blue. How many blue golf balls are there?

A: **Let's think step by step.** There are 16 balls in total. Half of the balls are golf balls. That means there are 8 golf balls. Half of the golf balls are blue. That means there are 4 blue golf balls. ✓

Zero-shot chain-of-thought prompting

	MultiArith	GSM8K
Zero-Shot	17.7	10.4
Few-Shot (2 samples)	33.7	15.6
Few-Shot (8 samples)	33.8	15.6
Zero-Shot-CoT	Greatly outperforms → 78.7	40.7
Few-Shot-CoT (2 samples)	zero-shot 84.8	41.3
Few-Shot-CoT (4 samples : First) (*1)	89.2	-
Few-Shot-CoT (4 samples : Second) (*1)	Manual CoT → 90.5	-
Few-Shot-CoT (8 samples)	still better 93.0	48.7

Zero-shot chain-of-thought prompting

No.	Category	Zero-shot CoT Trigger Prompt	Accuracy
1	LM-Designed	Let's work this out in a step by step way to be sure we have the right answer.	82.0
2	Human-Designed	Let's think step by step. (*1)	78.7
3		First, (*2)	77.3
4		Let's think about this logically.	74.5
5		Let's solve this problem by splitting it into steps. (*3)	72.2
6		Let's be realistic and think step by step.	70.8
7		Let's think like a detective step by step.	70.3
8		Let's think	57.5
9		Before we dive into the answer,	55.7
10		The answer is after the proof.	45.7
-		(Zero-shot)	17.7

Summary: Zero-Shot (ZS) and Few-Shot (FS) In-Context Learning

- ❑ + No finetuning needed, prompt engineering (e.g. CoT) can improve performance
- ❑ – Limits to what you can fit in context
- ❑ – Complex tasks will probably need gradient steps

Language modeling ≠ assisting users

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

- ❑ Language models are not aligned with user intent

Language modeling ≠ assisting users

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION **Human**

A giant rocket ship blasted off from Earth carrying astronauts to the moon. The astronauts landed their spaceship on the moon and walked around exploring the lunar surface. Then they returned safely back to Earth, bringing home moon rocks to show everyone.

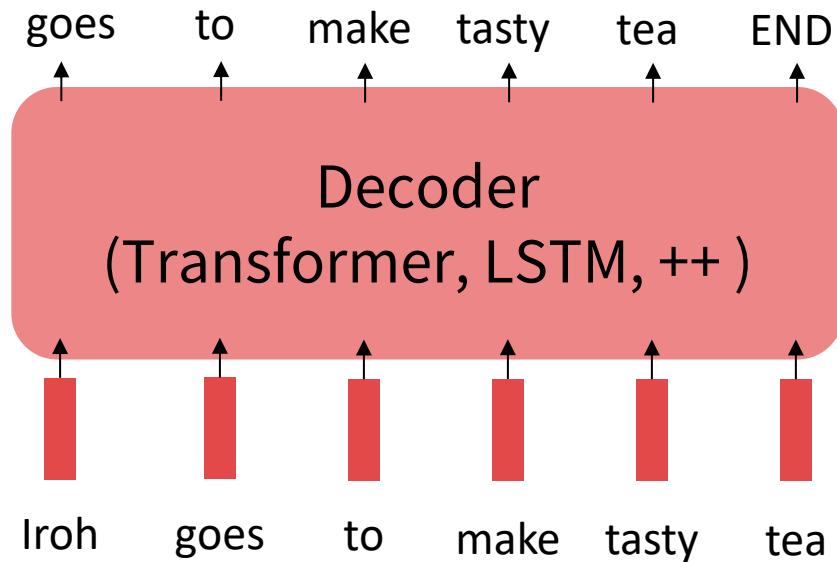
- Language models are not aligned with user intent
- Finetuning to the rescue!

The Pretraining / Finetuning Paradigm (review)

- ❑ Pretraining can improve NLP applications by serving as parameter initialization.

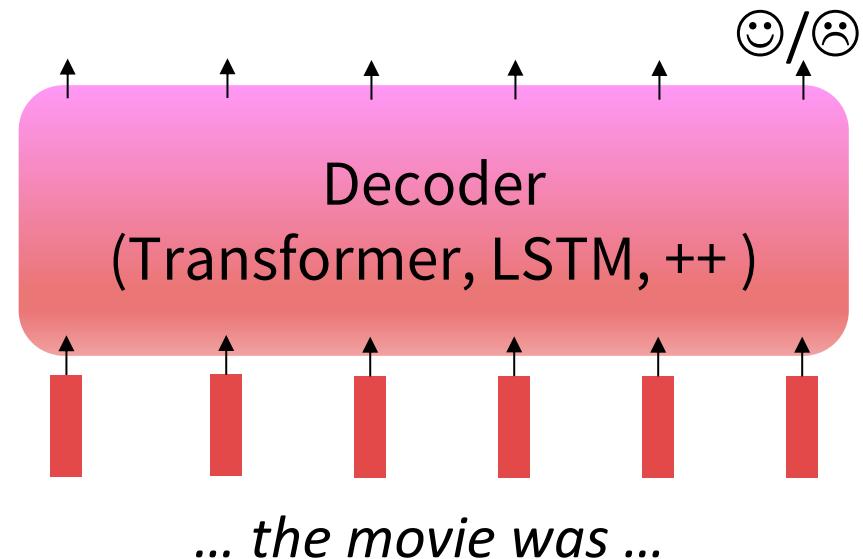
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!

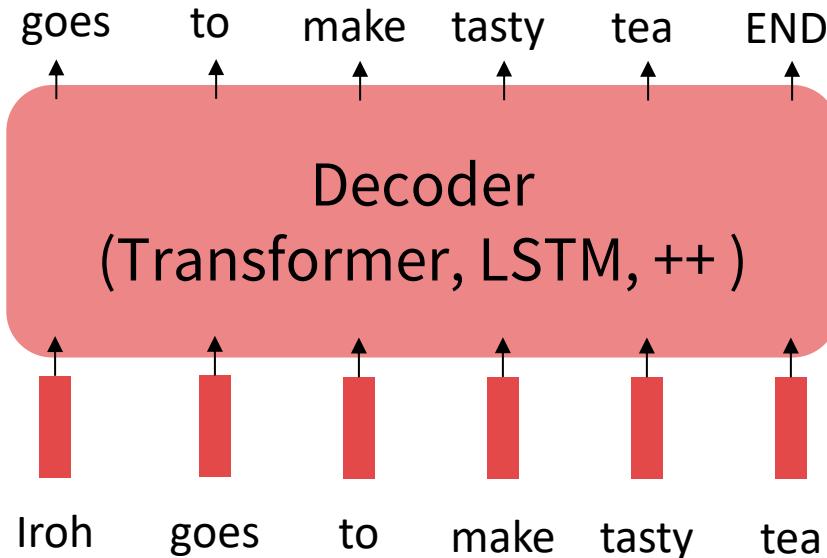


Scaling up finetuning

- ❑ Pretraining can improve NLP applications by serving as parameter initialization.

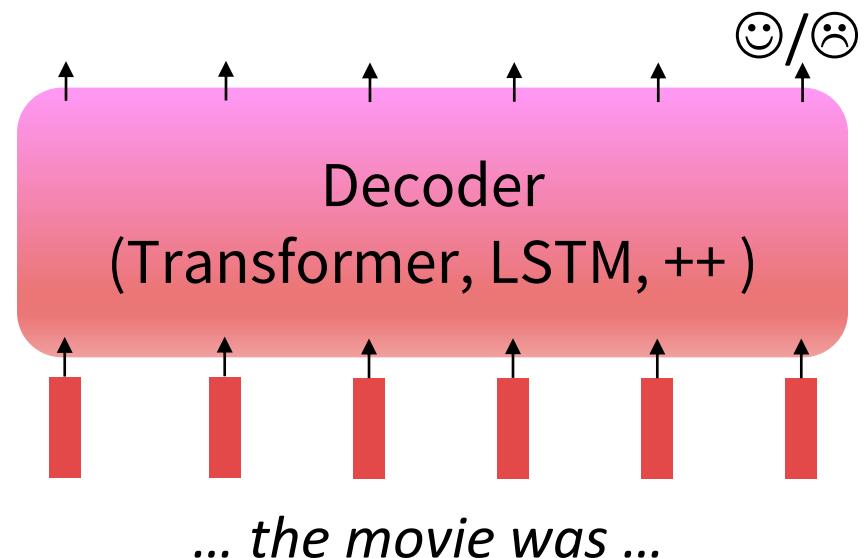
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



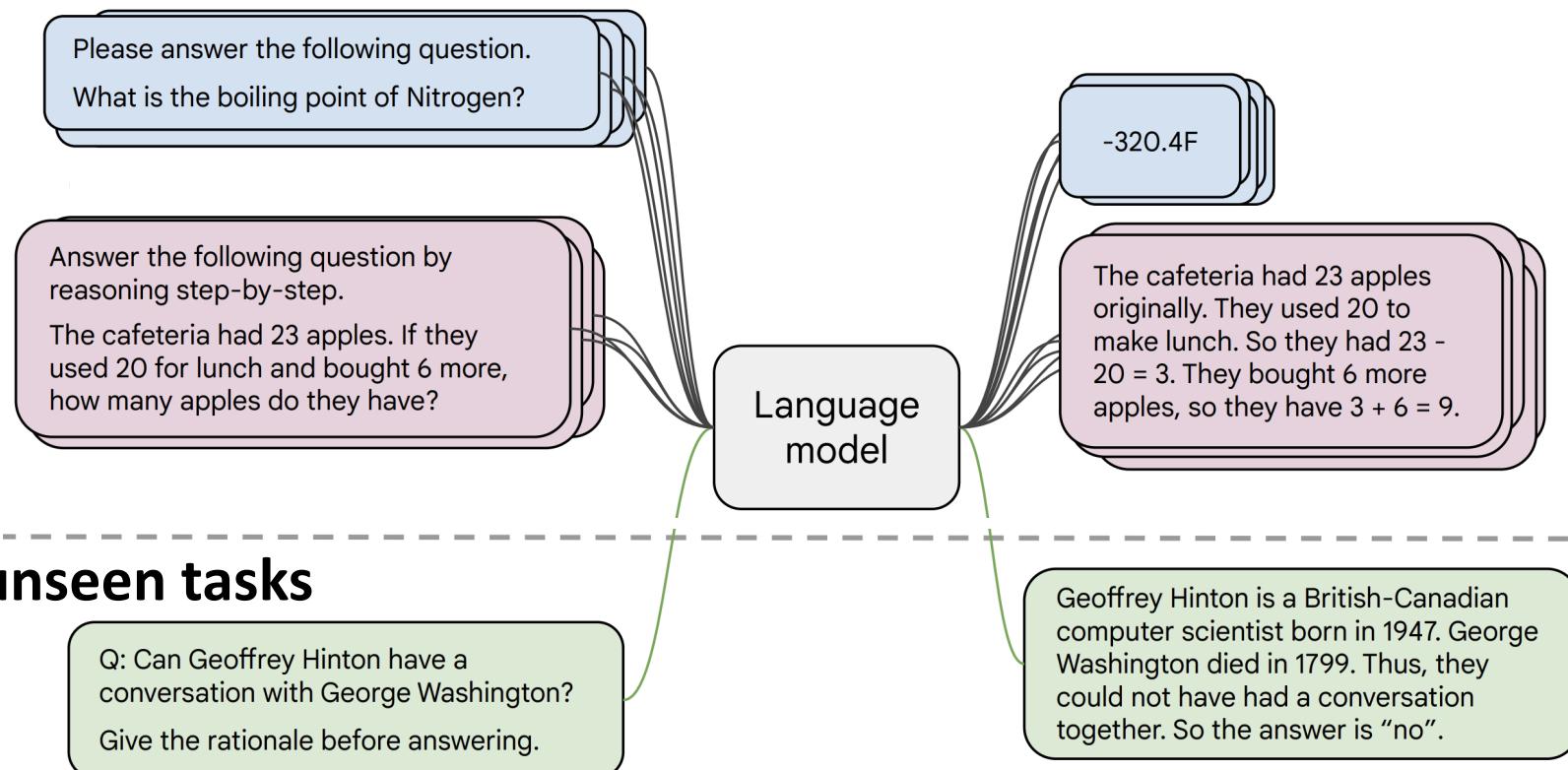
Step 2: Finetune (on many tasks)

Not many labels; adapt to the tasks!



Instruction finetuning

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



- ❑ Evaluate on unseen tasks

Summary: Instruction finetuning

- + Simple and straightforward, generalize to unseen tasks
- Collecting demonstrations for so many tasks is expensive
- Mismatch between LM objective and human preferences

Optimizing for human preferences

- ❑ Let's say we were training a language model on some task (e.g. summarization).
- ❑ For each LM sample s , imagine we had a way to obtain a *human reward* of that summary: $R(s) \in \mathbb{R}$, higher is better.

SAN FRANCISCO,
California (CNN) --
A magnitude 4.2
earthquake shook the
San Francisco
...
overturn unstable
objects.

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

$$s_1 \\ R(s_1) = 8.0$$

The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.

$$s_2 \\ R(s_2) = 1.2$$

- ❑ Now we want to maximize the expected reward of samples from our LM:

$$\mathbb{E}_{\hat{s} \sim p_\theta(s)}[R(\hat{s})]$$

Reinforcement learning to the rescue

- ❑ The field of reinforcement learning (RL) has studied these (and related) problems for many years now (Williams, 1992; Sutton and Barto, 1998)
- ❑ Circa 2013: resurgence of interest in RL applied to deep learning, game-playing (Mnih et al., 2013)
- ❑ But the interest in applying RL to modern LMs is an even newer phenomenon (Ziegler et al., 2019; Stiennon et al., 2020; Ouyang et al., 2022). Why?
 - RL w/ LMs has commonly been viewed as very hard to get right (still is!)
 - Newer advances in RL algorithms that work for large neural models, including language models (e.g. PPO; (Schulman et al., 2017))

Optimizing for human preferences

- How do we actually change our LM parameters θ to maximize this?

$$\mathbb{E}_{\hat{s} \sim p_{\theta}(s)}[R(\hat{s})]$$

- Let's try doing gradient ascent!

$$\theta_{t+1} := \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_{\theta_t}(s)}[R(\hat{s})]$$

How do we estimate
this expectation??

What if our reward
function is non-
differentiable??

- **Policy gradient** methods in RL (e.g., REINFORCE; (Williams, 1992)) give us tools for estimating and optimizing this objective.
- We'll describe a *very high-level* mathematical overview of the simplest policy gradient estimator, but a full treatment of RL is outside the scope of this course.

A (very!) brief introduction to policy gradient/REINFORCE (Williams, 1992)

- We want to obtain

$$\nabla_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}(s)} [R(\hat{s})] = \nabla_{\theta} \sum_s R(s) p_{\theta}(s) \stackrel{\text{(defn. of expectation)}}{=} \sum_s R(s) \nabla_{\theta} p_{\theta}(s) \stackrel{\text{(linearity of gradient)}}{=}$$

- Here we'll use a very handy trick known as the log-derivative trick. Let's try taking the gradient of $\log p_{\theta}(s)$

$$\nabla_{\theta} \log p_{\theta}(s) = \frac{1}{p_{\theta}(s)} \nabla_{\theta} p_{\theta}(s) \quad \Rightarrow \quad \nabla_{\theta} p_{\theta}(s) = \nabla_{\theta} \log p_{\theta}(s) p_{\theta}(s)$$

(chain rule)

- Plug back in:

This is an expectation of this

$$\begin{aligned} \sum_s R(s) \nabla_{\theta} p_{\theta}(s) &= \sum_s p_{\theta}(s) R(s) \nabla_{\theta} \log p_{\theta}(s) \\ &= \mathbb{E}_{\hat{s} \sim p_{\theta}(s)} [R(\hat{s}) \nabla_{\theta} \log p_{\theta}(\hat{s})] \end{aligned}$$

A (very!) brief introduction to policy gradient/REINFORCE (Williams, 1992)

- Now we have put the gradient “inside” the expectation, we can approximate this objective with Monte Carlo samples:

$$\nabla_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}(s)} [R(\hat{s})] = \mathbb{E}_{\hat{s} \sim p_{\theta}(s)} [R(\hat{s}) \nabla_{\theta} \log p_{\theta}(\hat{s})] \approx \frac{1}{m} \sum_{i=1}^m R(s_i) \nabla_{\theta} \log p_{\theta}(s_i)$$

- 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?

How do we model human preferences?

- ❑ Awesome: now for any **arbitrary, non-differentiable reward function** $R(s)$, we can train our language model to maximize expected reward.
- ❑ Not so fast! (Why not?)
- ❑ Problem 1: human-in-the-loop is expensive!
 - Solution: instead of directly asking humans for preferences, **model their preferences** as a separate (NLP) problem! (Knox and Stone, 2009)

An earthquake hit
San Francisco.
There was minor
property damage,
but no injuries.

$$s_1 \\ R(s_1) = 8.0 \quad \text{👤} \quad \text{💵💵}$$

The Bay Area has
good weather but is
prone to
earthquakes and
wildfires.

$$s_2 \\ R(s_2) = 1.2 \quad \text{👤} \quad \text{💵💵}$$

Train an LM $RM_\phi(s)$ to
predict human
preferences from an
annotated dataset, then
optimize for RM_ϕ instead.

How do we model human preferences?

- Problem 2: human judgments are noisy and miscalibrated!
- Solution: instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable (Phelps et al., 2015; Clark et al., 2018)

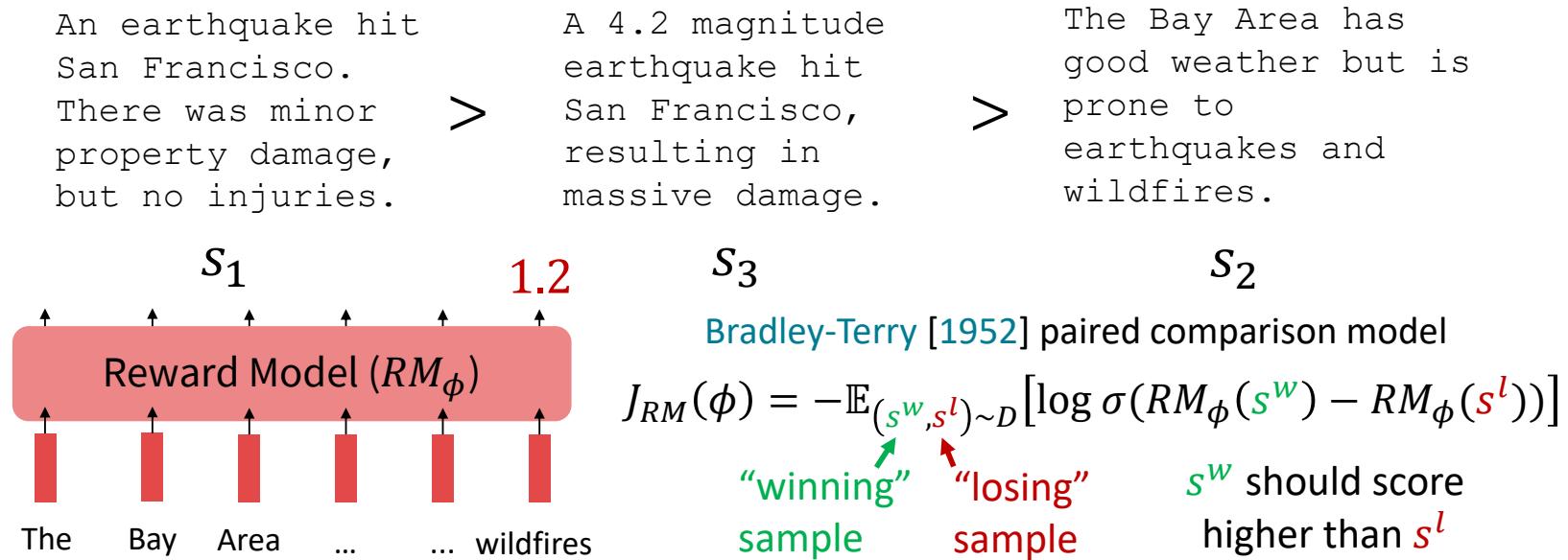
A 4.2 magnitude
earthquake hit
San Francisco,
resulting in
massive damage.

s_3

$$R(s_3) = \text{ 4.1? } \text{ 6.6? } \text{ 3.2?}$$

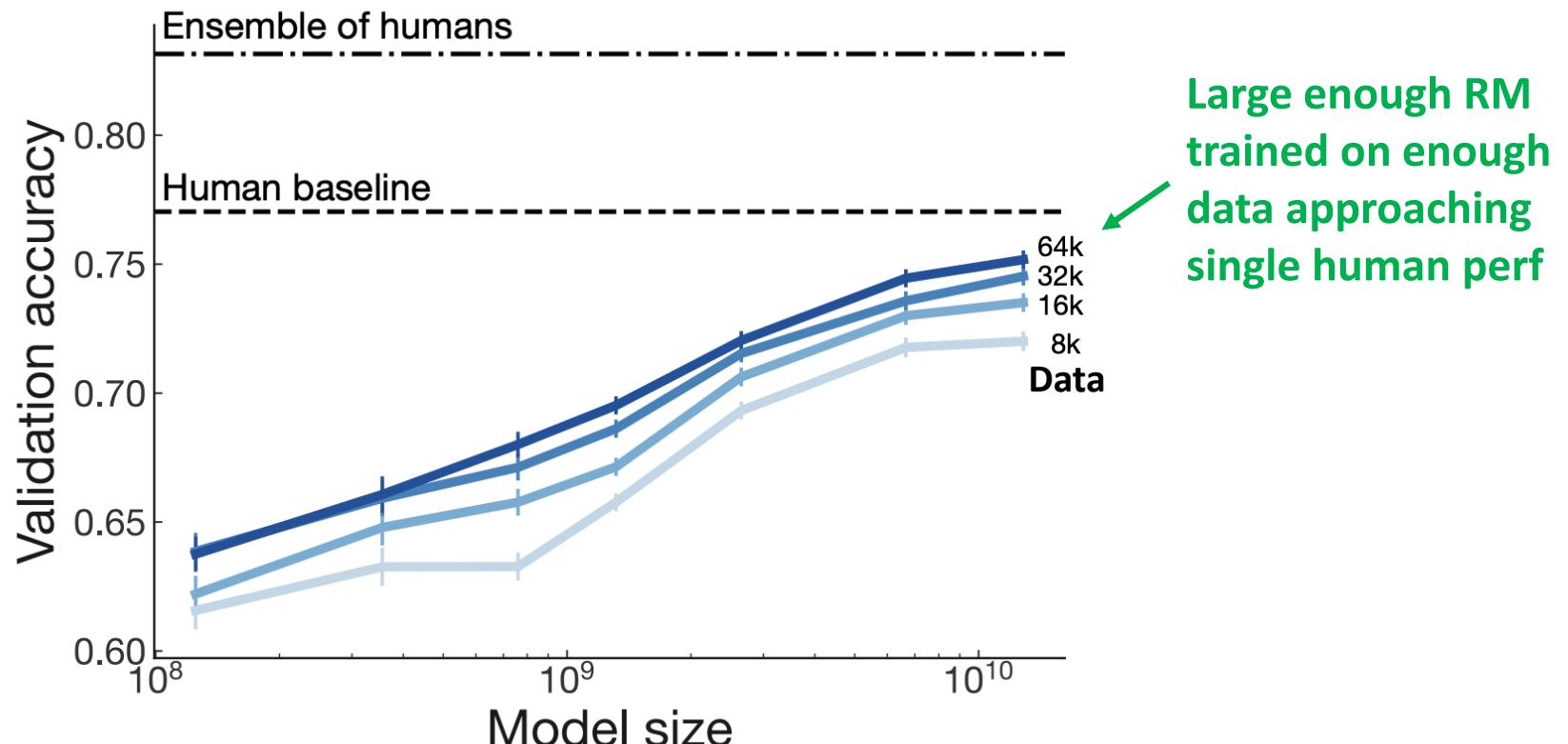
How do we model human preferences?

- Problem 2: human judgments are noisy and miscalibrated!
- Solution: instead of asking for direct ratings, ask for **pairwise comparisons**, which can be more reliable (Phelps et al., 2015; Clark et al., 2018)



Make sure your reward model works first!

- ❑ Evaluate RM on predicting outcome of held-out human judgments



(Stiennon et al., 2020)

RLHF: Putting it all together (Christiano et al., 2017; Stiennon et al., 2020)

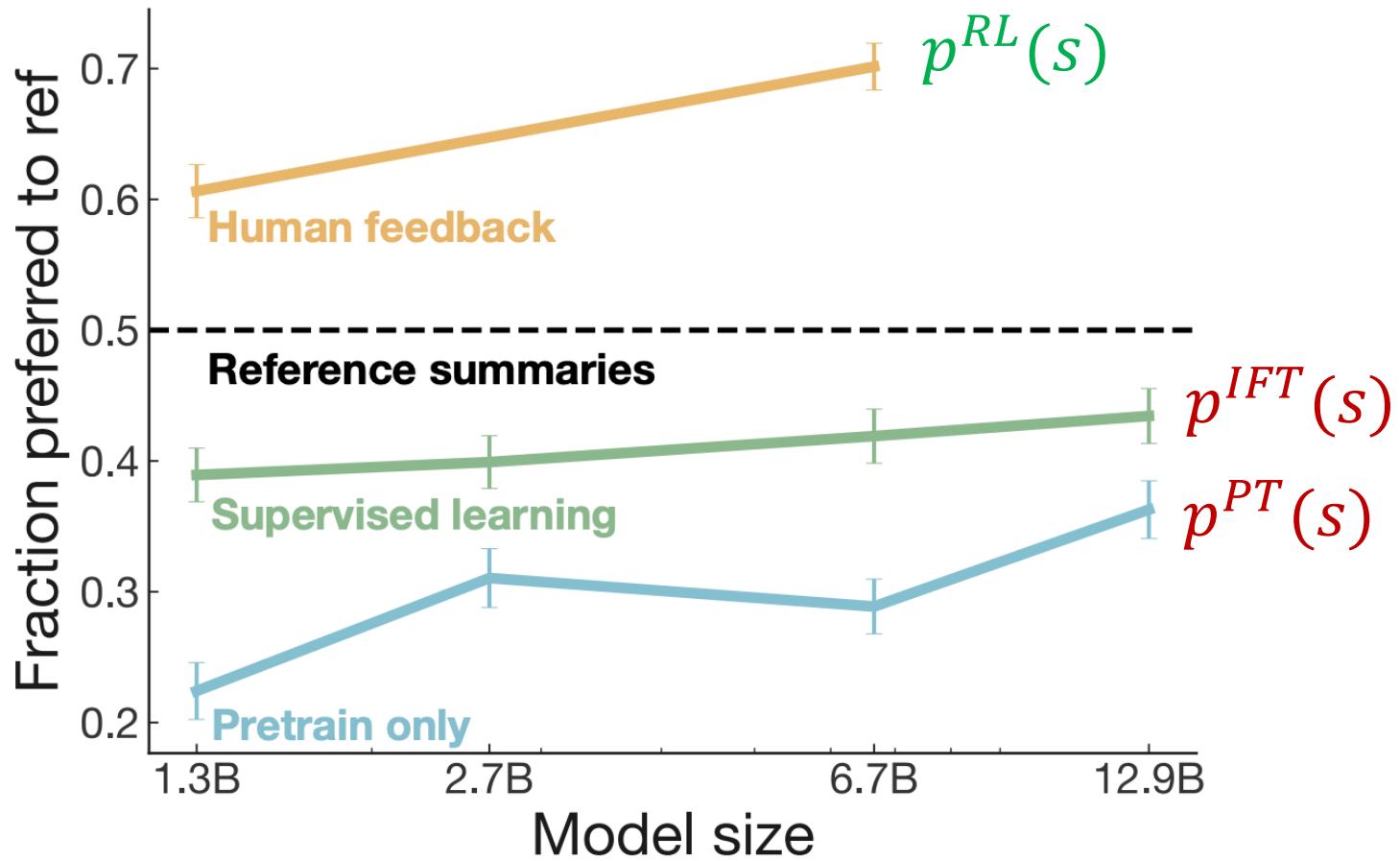
- Finally, we have everything we need:
 - A pretrained (possibly instruction-finetuned) LM $p^{PT}(s)$
 - A reward model $RM_\phi(s)$ that produces scalar rewards for LM outputs, trained on a dataset of human comparisons
 - A method for optimizing LM parameters towards an arbitrary reward function.
- 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:

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

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 $p^{PT}(s)$.

RLHF provides gains over pretraining + finetuning



(Stiennon et al., 2020)

InstructGPT: scaling up RLHF to tens of thousands of tasks

**30k
tasks!**

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.



Some people went to the moon...

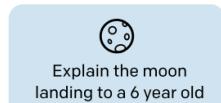
This data is used to fine-tune GPT-3 with supervised learning.



Step 2

**Collect comparison data,
and train a reward model.**

A prompt and several model outputs are sampled.



A, B, C, D

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.



D > C > A = B

Step 3

Optimize a policy against the reward model using reinforcement learning.

A new prompt is sampled from the dataset.

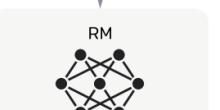


The policy generates an output.



Once upon a time...

The reward model calculates a reward for the output.



The reward is used to update the policy using PPO.

r_k

(Ouyang et al., 2022)

InstructGPT: scaling up RLHF to tens of thousands of tasks

❑ Tasks collected from labelers:

- **Plain:** We simply ask the labelers to come up with an arbitrary task, while ensuring the tasks had sufficient diversity.
- **Few-shot:** We ask the labelers to come up with an instruction, and multiple query/response pairs for that instruction.
- **User-based:** We had a number of use-cases stated in waitlist applications to the OpenAI API. We asked labelers to come up with prompts corresponding to these use cases.

Use-case	Prompt
Brainstorming	List five ideas for how to regain enthusiasm for my career
Generation	Write a short story where a bear goes to the beach, makes friends with a seal, and then returns home.

InstructGPT

PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

COMPLETION GPT-3

Explain the theory of gravity to a 6 year old.

Explain the theory of relativity to a 6 year old in a few sentences.

Explain the big bang theory to a 6 year old.

Explain evolution to a 6 year old.

InstructGPT

People went to the moon, and they took pictures of what they saw, and sent them back to the earth so we could all see them.

InstructGPT

PROMPT *Write a short poem about a wise frog.*

COMPLETION GPT-3

Write a short story in which a character has two different names.

Write a short story in which you try to get something back that you have lost.

Write a short story in which a character has a bad dream.

InstructGPT

The frog is a symbol of wisdom
He knows all the secrets of the world
He is a master of disguise
And a great teacher of life
He is a symbol of transformation
And the bringer of change
He is the frog who has seen it all
And knows the meaning of it all

ChatGPT: Instruction Finetuning + RLHF for dialog agents

ChatGPT: Optimizing Language Models for Dialogue

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge...

Methods

We trained this model using Reinforcement Learning from Human Feedback (RLHF), using the same methods as [InstructGPT](#), but with slight differences in the data collection setup. We trained an initial model using supervised fine-tuning: human AI trainers provided conversations in which they played both sides—the user and an AI assistant. We gave the trainers access to model-written suggestions to help them compose their responses. We mixed this new dialogue dataset with the InstructGPT dataset, which we transformed into a dialogue format.

(Instruction finetuning!)

ChatGPT: Instruction Finetuning + RLHF for dialog agents

ChatGPT: Optimizing Language Models for Dialogue

Note: OpenAI (and similar companies) are keeping more details secret about ChatGPT training (including data, training parameters, model size)—perhaps to keep a competitive edge...

Methods

To create a reward model for reinforcement learning, we needed to collect comparison data, which consisted of two or more model responses ranked by quality. To collect this data, we took conversations that AI trainers had with the chatbot. We randomly selected a model-written message, sampled several alternative completions, and had AI trainers rank them. Using these reward models, we can fine-tune the model using Proximal Policy Optimization. We performed several iterations of this process.

(RLHF!)

ChatGPT: Instruction Finetuning + RLHF for dialog agents

MU

Explain in a few sentences what methods were used in training ChatGPT to the audience of CS224n, Stanford's Deep Learning for Natural Language processing course. A lot of the students are very Gen Z, so be sure to use a ton of emojis to seem hip and cool.



Hey CS224n peeps! 😎 ChatGPT here to spill the tea on how I was trained to be a savvy language model! 🤓

So basically, I was pre-trained using a combination of unsupervised learning techniques, specifically a transformer-based language model architecture, that allowed me to learn patterns and relationships within large amounts of text data. 📚

The training data came from various sources such as books, articles, and websites, and I was fine-tuned on a variety of downstream NLP tasks, which involved supervised learning using labeled datasets to further improve my performance. 🤖

Limitations of RL + Reward Modeling

- ❑ Human preferences are unreliable!
 - "Reward hacking" is a common problem in RL
 - Chatbots are rewarded to produce responses that seem authoritative and helpful, regardless of truth
 - This can result in making up facts + hallucinations
- ❑ Models of human preferences are even more unreliable!
- ❑ There is a real concern of AI mis(alignment)!

What's next?

- ❑ RLHF is still a very underexplored and fast- moving area: by the next lecture (2024) these slides may look completely different!
- ❑ RLHF gets you further than instruction finetuning, but is (still!) data expensive.
- ❑ Recent work aims to alleviate such data requirements:
 - RL from **AI feedback** (Bai et al., 2022)
 - Finetuning LMs on their own outputs (Huang et al., 2022; Zelikman et al., 2022)
- ❑ However, there are still many limitations of large LMs (size, hallucination) that may not be solvable with RLHF!