ITCS 4101: Introduction to NLP

Large Language Models:

Pretraining, Fine-tuning

In-context Learning, Chain of Thought

Instruct Tuning, RLHF

Razvan C. Bunescu
Department of CS @ CCI
University of North Carolina at Charlotte
https://webpages.charlotte.edu/rbunescu
razvan.bunescu@charlotte.edu

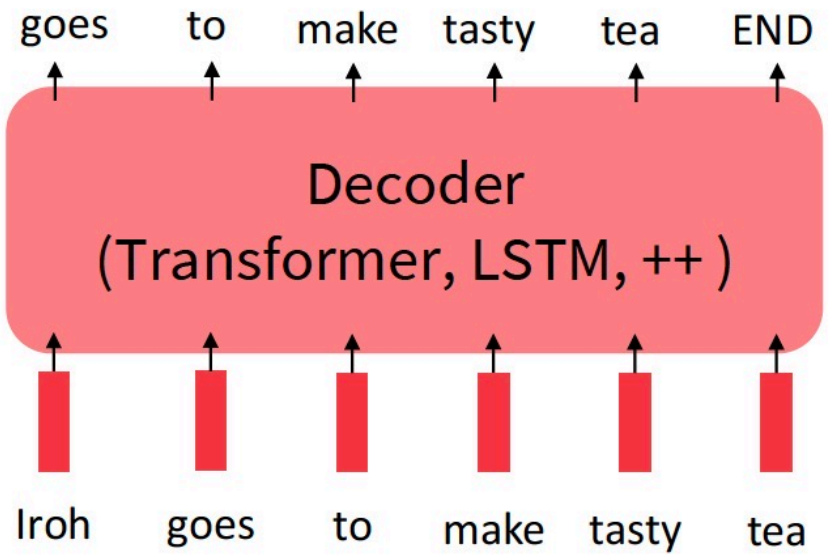
Pretraining through language modeling [Dai and Le, 2015]

Recall the **language modeling** task:

- Model $p_\theta(w_t|w_{1:t-1})$, the probability distribution over words given their past contexts.
- There's lots of data for this! (In English.)

Pretraining through language modeling:

- Train a neural network to perform language modeling on a large amount of text.
- Save the network parameters.



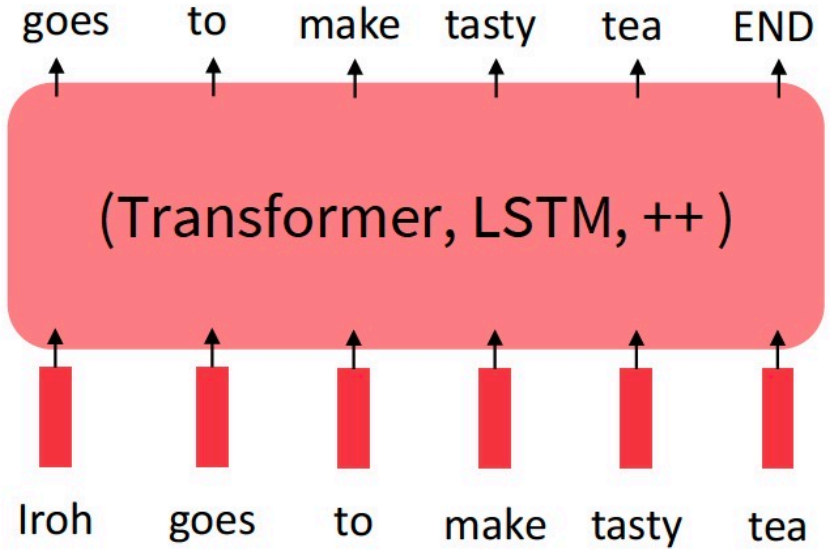
white slides selected from cs224n @ Stanford

The Pretraining / Finetuning Paradigm

Pretraining can improve NLP applications by serving as parameter initialization.

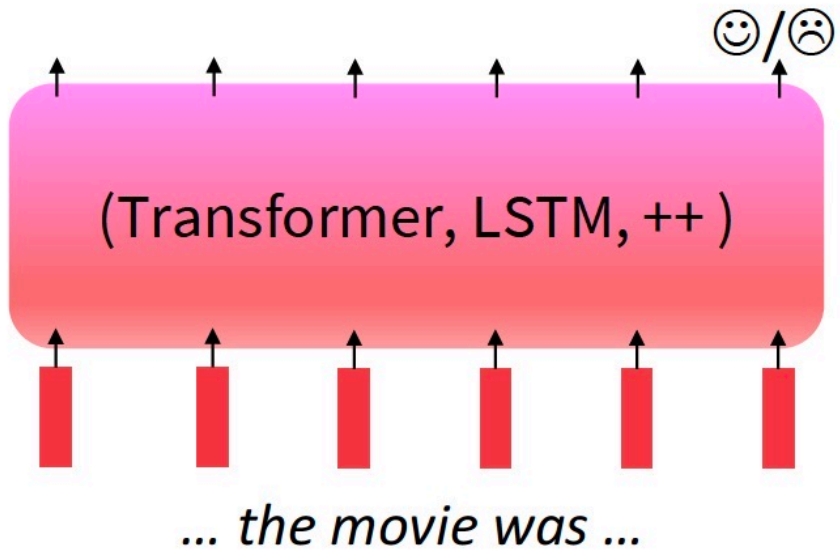
Step 1: Pretrain (on language modeling)

Lots of text; learn general things!



Step 2: Finetune (on your task)

Not many labels; adapt to the task!



Full Finetuning vs. Parameter-Efficient Finetuning

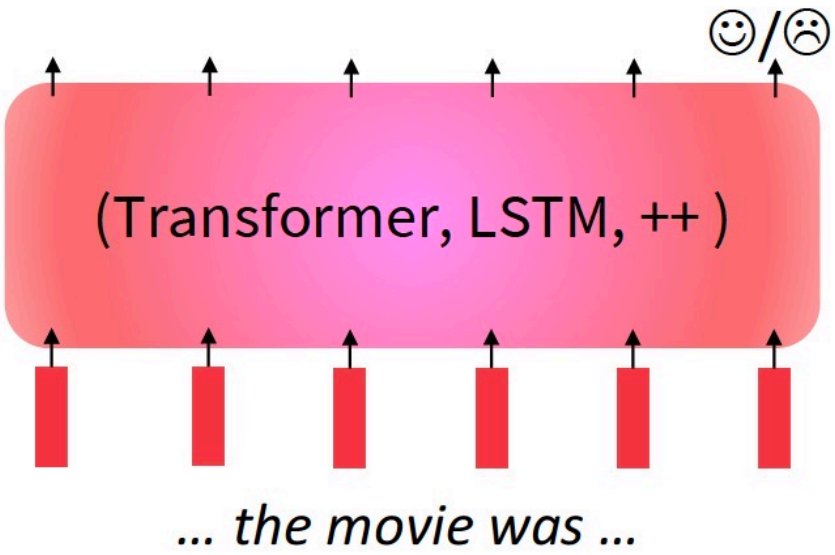
Finetuning every parameter in a pretrained model works well, but is memory-intensive.

But **lightweight** finetuning methods adapt pretrained models in a constrained way.

Leads to **less overfitting** and/or **more efficient finetuning and inference.**

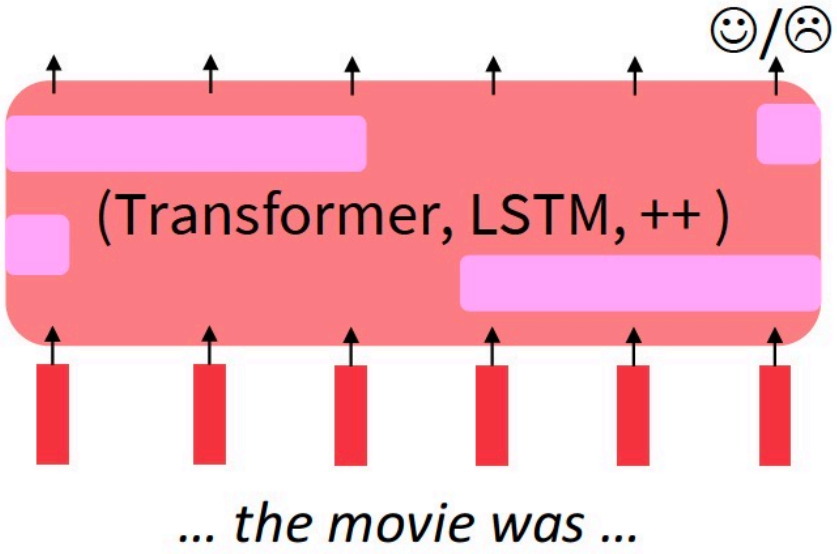
Full Finetuning

Adapt all parameters



Lightweight Finetuning

Train a few existing or new parameters



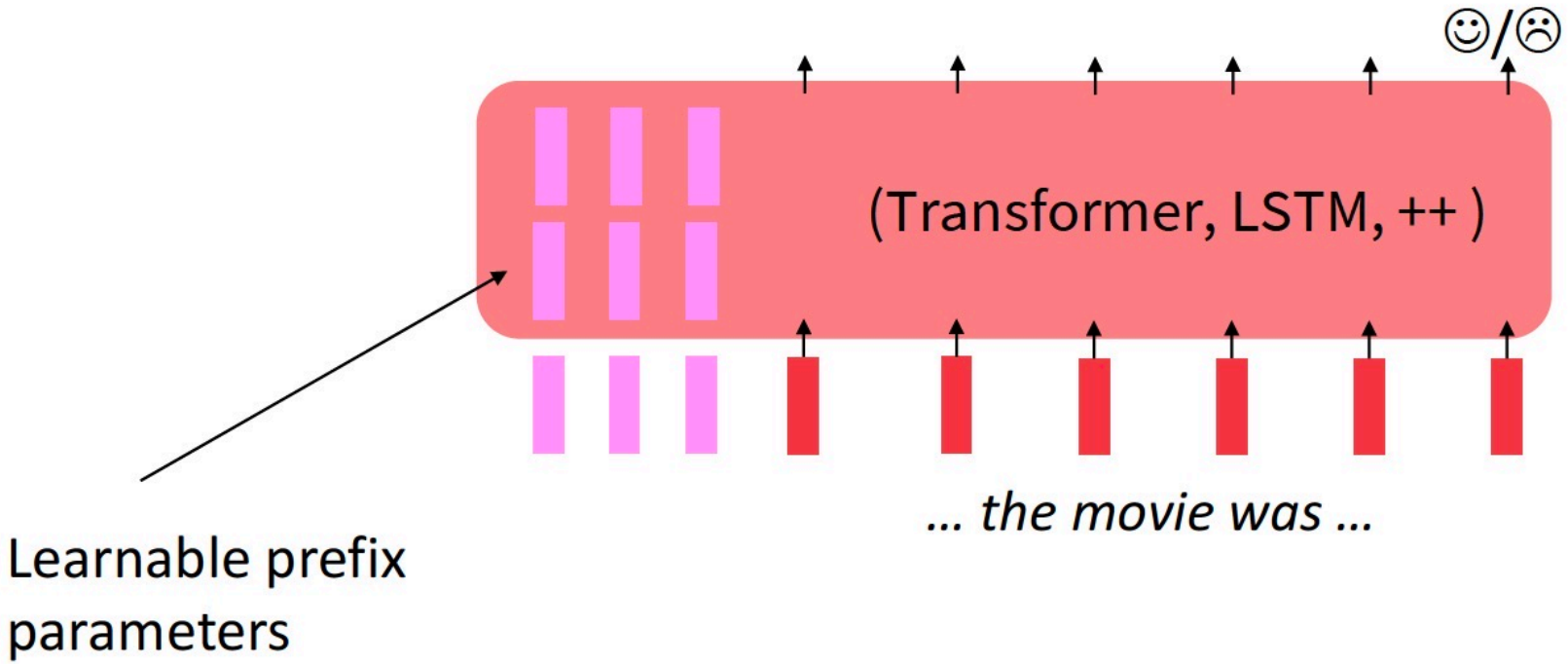
[Liu et al., 2019; Joshi et al., 2020]

Parameter-Efficient Finetuning: Prefix-Tuning, Prompt tuning

Prefix-Tuning adds a **prefix** of parameters, and **freezes all pretrained parameters.**

The prefix is processed by the model just like real words would be.

Advantage: each element of a batch at inference could run a different tuned model.

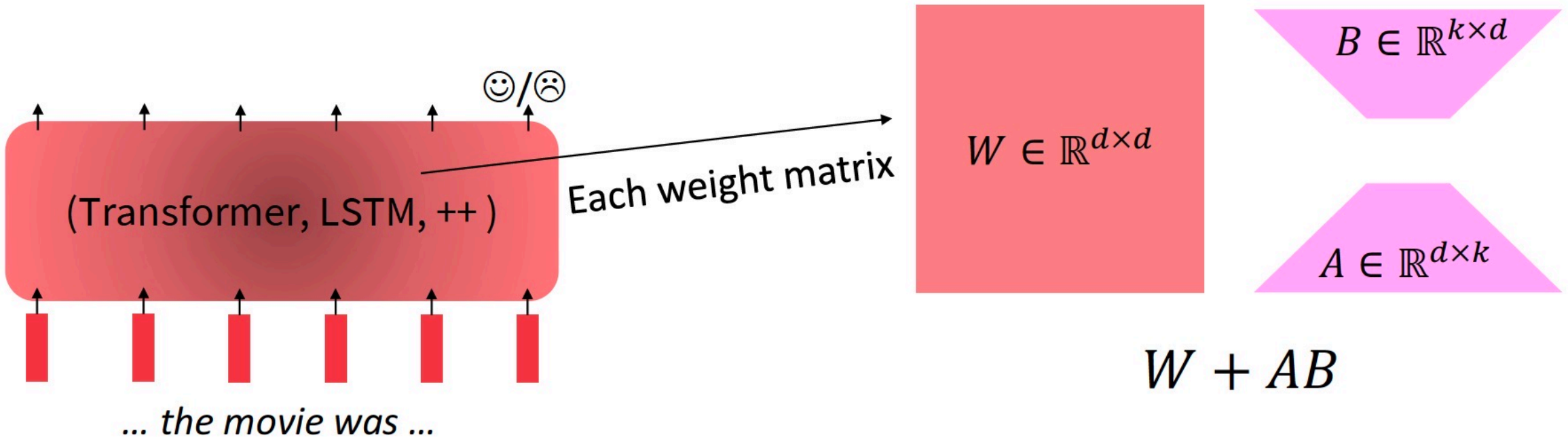


[Li and Liang, 2021; Lester et al., 2021]

Parameter-Efficient Finetuning: Low-Rank Adaptation

Low-Rank Adaptation Learns a low-rank "diff" between the pretrained and finetuned weight matrices.

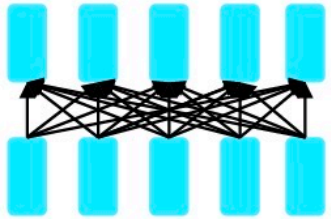
Easier to learn than prefix-tuning.



[Hu et al., 2021]

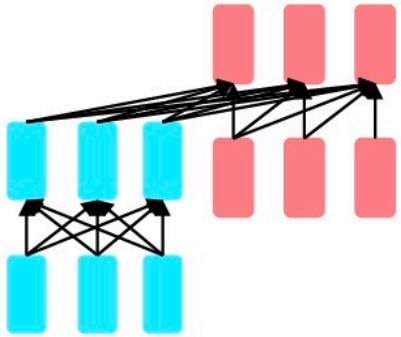
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



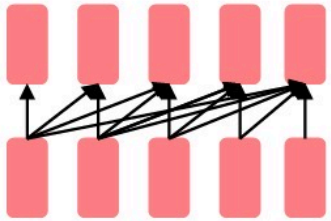
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?

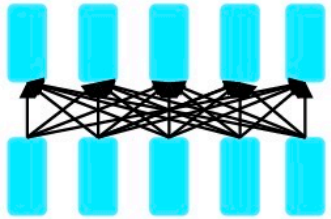


Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

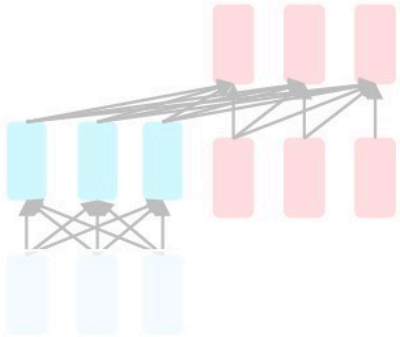
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



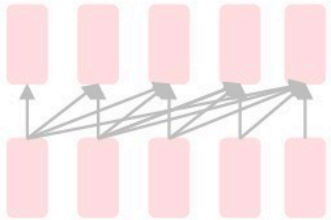
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

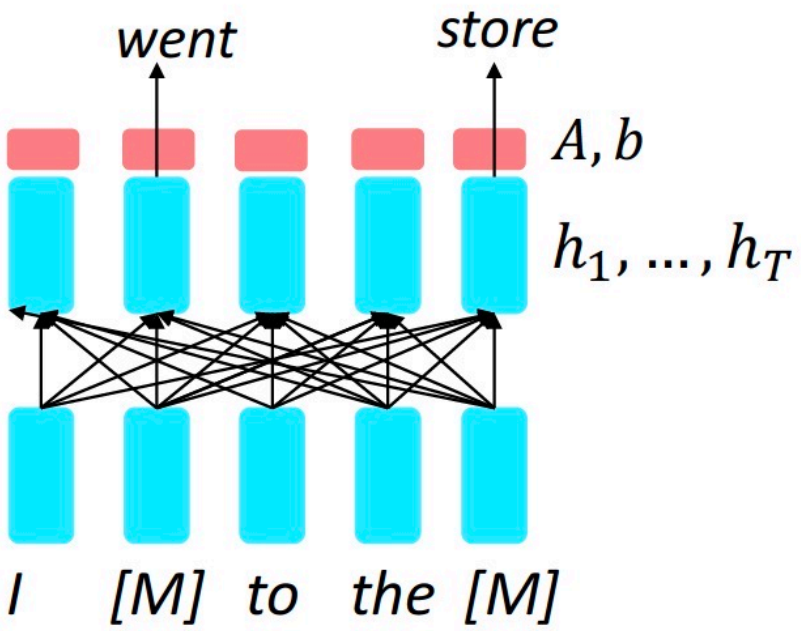
- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words

Pretraining encoders: what pretraining objective to use?

So far, we've looked at language model pretraining. But **encoders get bidirectional context,** so we can't do language modeling!

Idea: replace some fraction of words in the input with a special [MASK] token; predict these words.

$$h_1, \dots, h_T = \text{Encoder}(w_1, \dots, w_T)$$
$$y_i \sim Aw_i + b$$



Only add loss terms from words that are "masked out." If $\tilde{x}$ is the masked version of x, we're learning $p_\theta(x|\tilde{x})$. Called **Masked LM**.

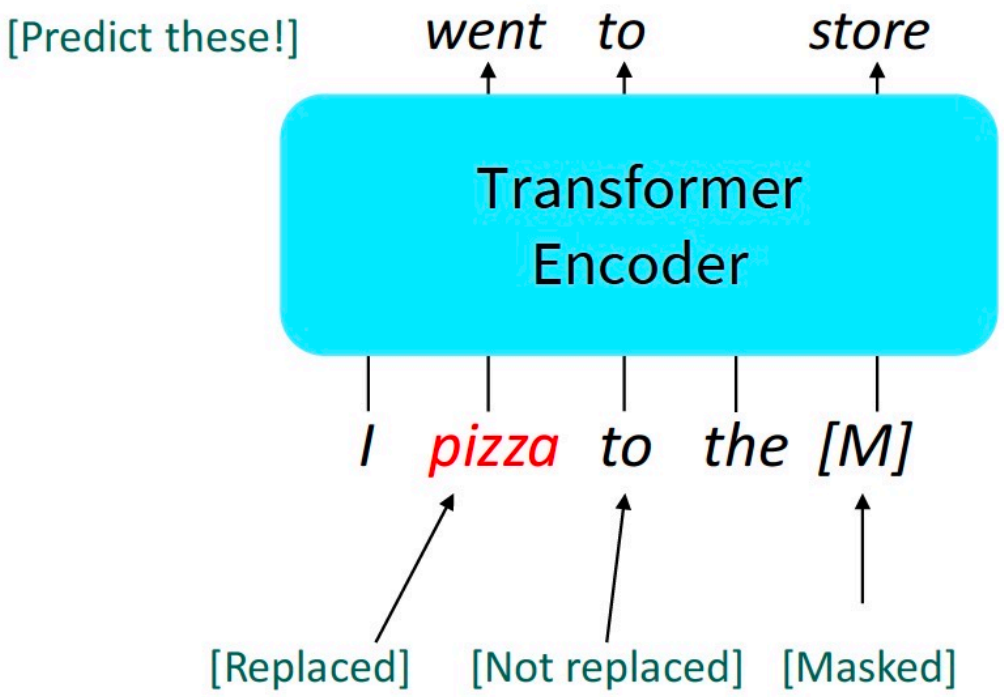
[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

Devlin et al., 2018 proposed the "Masked LM" objective and **released the weights of a pretrained Transformer**, a model they labeled BERT.

Some more details about Masked LM for BERT:

- Predict a random 15% of (sub)word tokens.
  - Replace input word with [MASK] 80% of the time
  - Replace input word with a random token 10% of the time
  - Leave input word unchanged 10% of the time (but still predict it!)
- Why? Doesn't let the model get complacent and not build strong representations of non-masked words. (No masks are seen at fine-tuning time!)



[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

Details about BERT

- Two models were released:
  - BERT-base: 12 layers, 768-dim hidden states, 12 attention heads, 110 million params.
  - BERT-large: 24 layers, 1024-dim hidden states, 16 attention heads, 340 million params.
- Trained on:
  - BooksCorpus (800 million words)
  - English Wikipedia (2,500 million words)
- Pretraining is expensive and impractical on a single GPU.
  - BERT was pretrained with 64 TPU chips for a total of 4 days.
  - (TPUs are special tensor operation acceleration hardware)
- Finetuning is practical and common on a single GPU
  - “Pretrain once, finetune many times.”

[Devlin et al., 2018]

BERT: Bidirectional Encoder Representations from Transformers

BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: microsoft paraphrase corpus
- **RTE**: a small natural language inference corpus

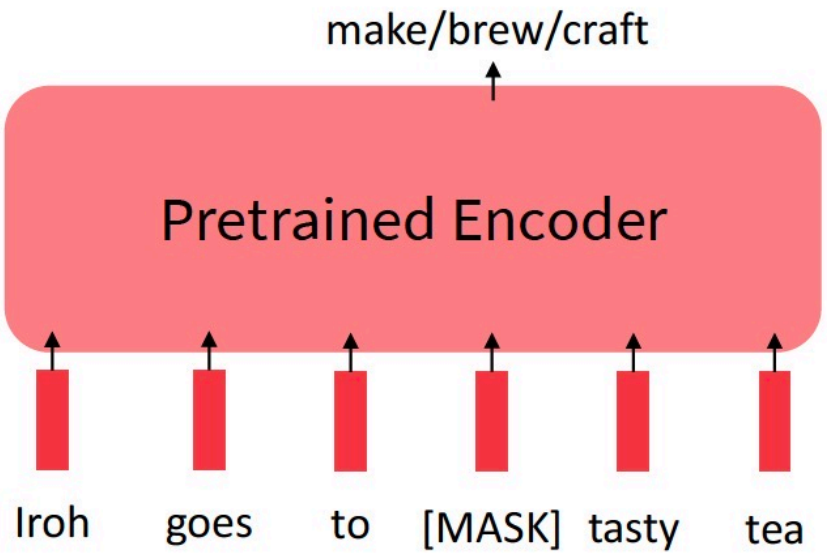
System	MNLI-(m/mm) 392k	QQP 363k	QNLI 108k	SST-2 67k	CoLA 8.5k	STS-B 5.7k	MRPC 3.5k	RTE 2.5k	**Average** -
Pre-OpenAI SOTA	80.6/80.1	66.1	82.3	93.2	35.0	81.0	86.0	61.7	74.0
BiLSTM+ELMo+Attn	76.4/76.1	64.8	79.8	90.4	36.0	73.3	84.9	56.8	71.0
OpenAI GPT	82.1/81.4	70.3	87.4	91.3	45.4	80.0	82.3	56.0	75.1
$\text{BERT}_{\text{BASE}}$	84.6/83.4	71.2	90.5	93.5	52.1	85.8	88.9	66.4	79.6
$\text{BERT}_{\text{LARGE}}$	**86.7/85.9**	**72.1**	**92.7**	**94.9**	**60.5**	**86.5**	**89.3**	**70.1**	**82.1**

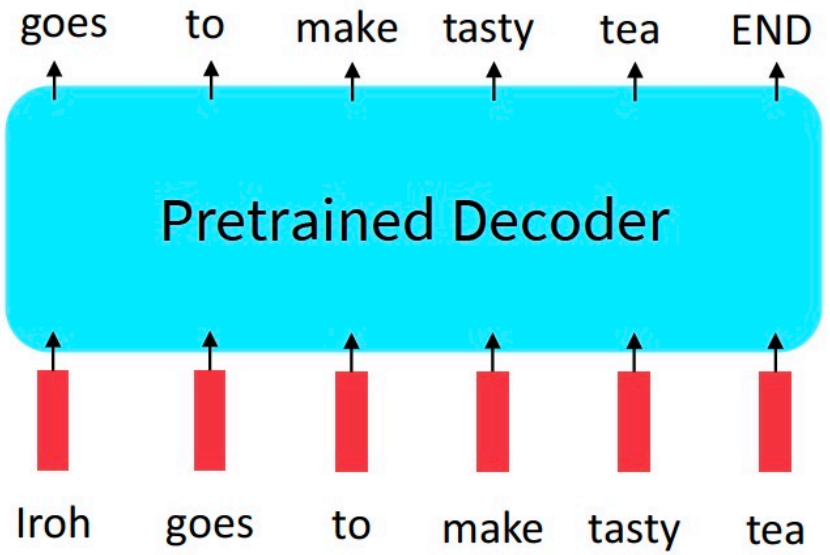
[Devlin et al., 2018]

Limitations of pretrained encoders

Those results looked great! Why not used pretrained encoders for everything?

If your task involves generating sequences, consider using a pretrained decoder; BERT and other pretrained encoders don't naturally lead to nice autoregressive (1-word-at-a-time) generation methods.



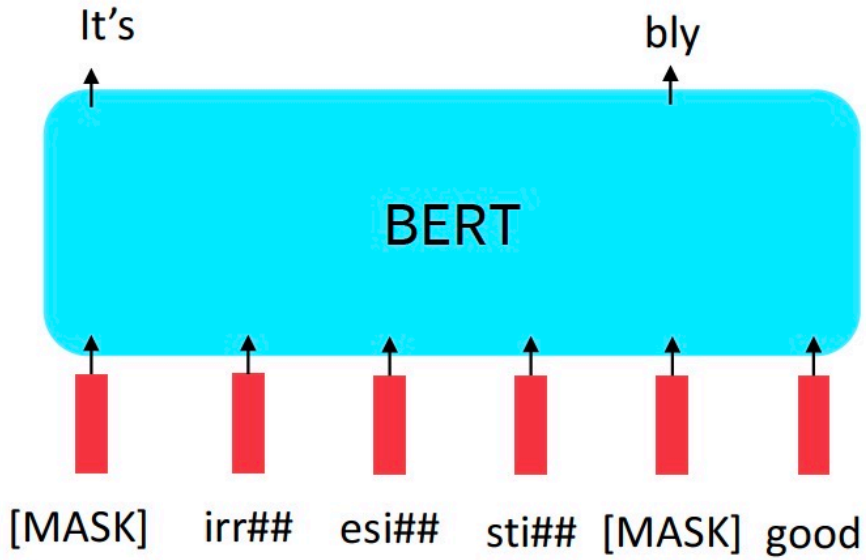


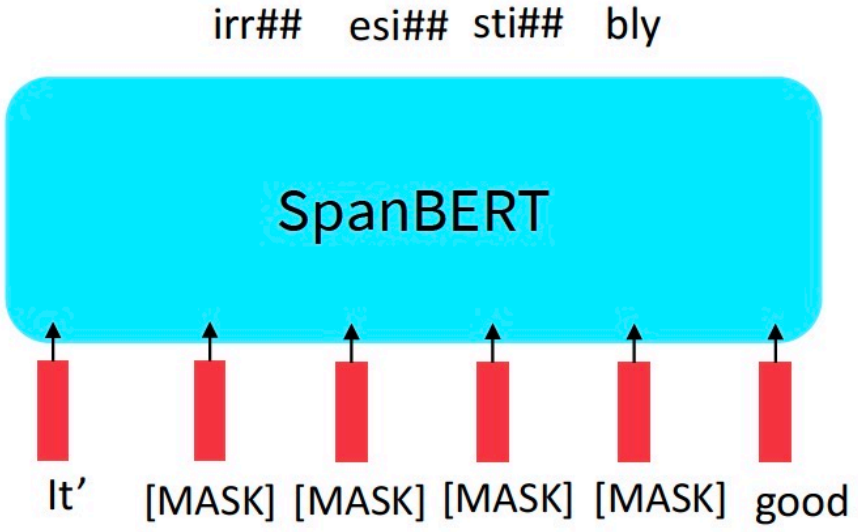
Extensions of BERT

You'll see a lot of BERT variants like RoBERTa, SpanBERT, +++

Some generally accepted improvements to the BERT pretraining formula:

- RoBERTa: mainly just train BERT for longer and remove next sentence prediction!
- SpanBERT: masking contiguous spans of words makes a harder, more useful pretraining task

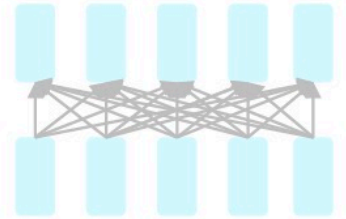




[Liu et al., 2019; Joshi et al., 2020]

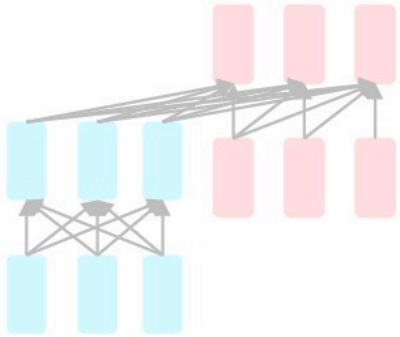
Pretraining for three types of architectures

The neural architecture influences the type of pretraining, and natural use cases.



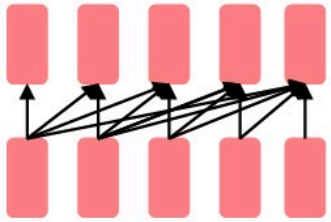
Encoders

- Gets bidirectional context – can condition on future!
- How do we train them to build strong representations?



Encoder-Decoders

- Good parts of decoders and encoders?
- What's the best way to pretrain them?



Decoders

- Language models! What we've seen so far.
- Nice to generate from; can't condition on future words
- All the biggest pretrained models are Decoders.

Pretraining decoders

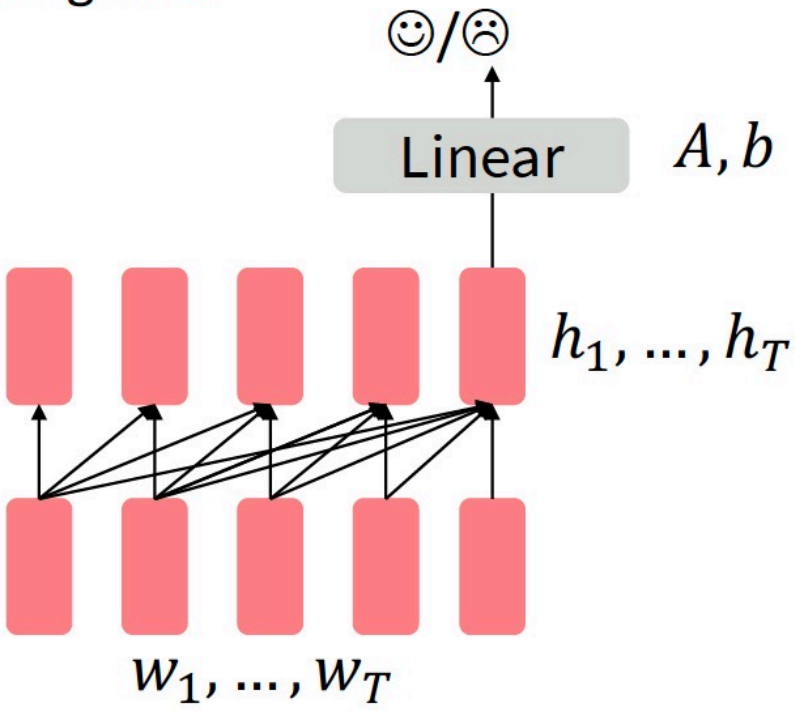
When using language model pretrained decoders, we can ignore that they were trained to model $p(w_t|w_{1:t-1})$.

We can finetune them by training a classifier on the last word's hidden state.

$$h_1, \dots, h_T = \text{Decoder}(w_1, \dots, w_T)$$
$$y \sim Ah_T + b$$

Where A and b are randomly initialized and specified by the downstream task.

Gradients backpropagate through the whole network.



[Note how the linear layer hasn't been pretrained and must be learned from scratch.]

Pretraining decoders

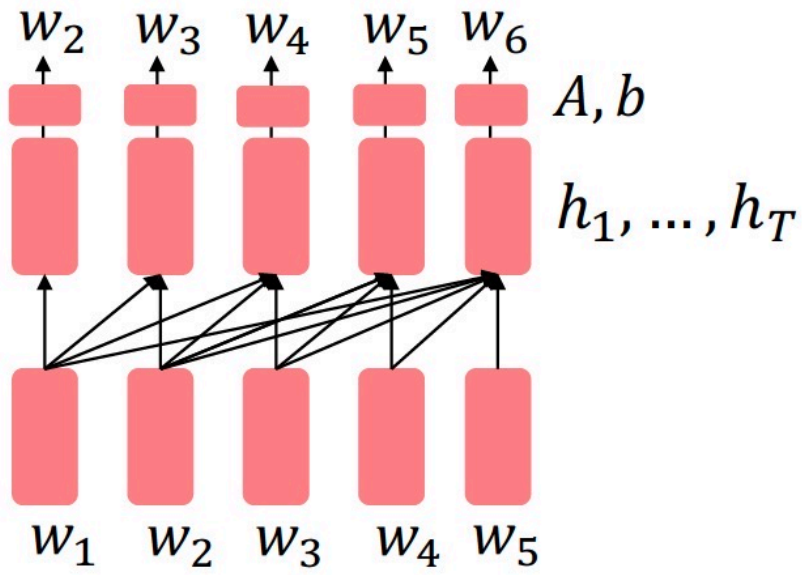
It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!

This is helpful in tasks **where the output is a sequence** with a vocabulary like that at pretraining time!

- Dialogue (context=dialogue history)
- Summarization (context=document)

$$h_1, \ldots, h_T = \text{Decoder}(w_1, \ldots, w_T)$$
$$w_t \sim Ah_{t-1} + b$$

Where A, b were pretrained in the language model!

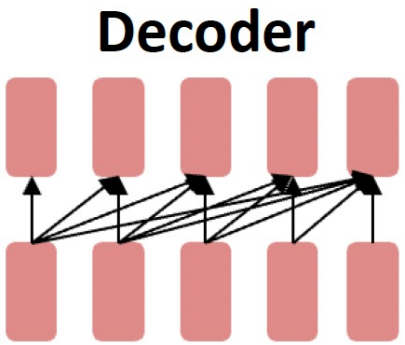


[Note how the linear layer has been pretrained.]

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

2018's GPT was a big success in pretraining a decoder!

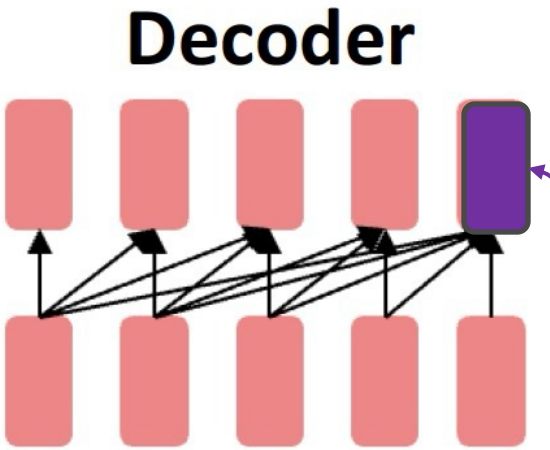
- Transformer decoder with 12 layers, 117M parameters.
- 768-dimensional hidden states, 3072-dimensional feed-forward hidden layers.
- Byte-pair encoding with 40,000 merges
- Trained on BooksCorpus: over 7000 unique books.
  - Contains long spans of contiguous text, for learning long-distance dependencies.
- The acronym "GPT" never showed up in the original paper; it could stand for "Generative PreTraining" or "Generative Pretrained Transformer"



[Devlin et al., 2018]

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.

Use the representation of the [EXTRACT] token as input for a linear classifier that is trained (together with fine-tuning the decoder) to predict entailment or not.

entailment

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

Generative Pretrained Transformer (GPT) [Radford et al., 2018]

GPT results on various *natural language inference* datasets.

Method	MNLI-m	MNLI-mm	SNLI	SciTail	QNLI	RTE
ESIM + ELMo [44] (5x)	-	-	89.3	-	-	-
CAFE [58] (5x)	80.2	79.0	89.3	-	-	-
Stochastic Answer Network [35] (3x)	80.6	80.1	-	-	-	-
CAFE [58]	78.7	77.9	88.5	83.3		
GenSen [64]	71.4	71.3	-	-	82.3	59.2
Multi-task BiLSTM + Attn [64]	72.2	72.1	-	-	82.1	**61.7**
Finetuned Transformer LM (ours)	**82.1**	**81.4**	**89.9**	**88.3**	**88.1**	56.0

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

Alec Radford [*1] Jeffrey Wu [*1] Rewon Child [1] David Luan [1] Dario Amodei [**1] Ilya Sutskever [**1]

Emergent zero-shot learning

All tasks had to be formatted to fit language modelling, i.e. such that the answer could be extracted from the generated tokens or their probabilities.

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(...because` **`the cat`** `was too big) >=`
  `P(...because` **`the hat`** `was too big)`?

[Radford et al., 2019]

GPT-3, In-context learning, and very large models

So far, we've interacted with pretrained models in two ways:

- Sample from the distributions they define (maybe providing a prompt)
- Fine-tune them on a task we care about, and take their predictions.

Very large language models seem to perform some kind of learning **without gradient steps** simply from examples you provide within their contexts.

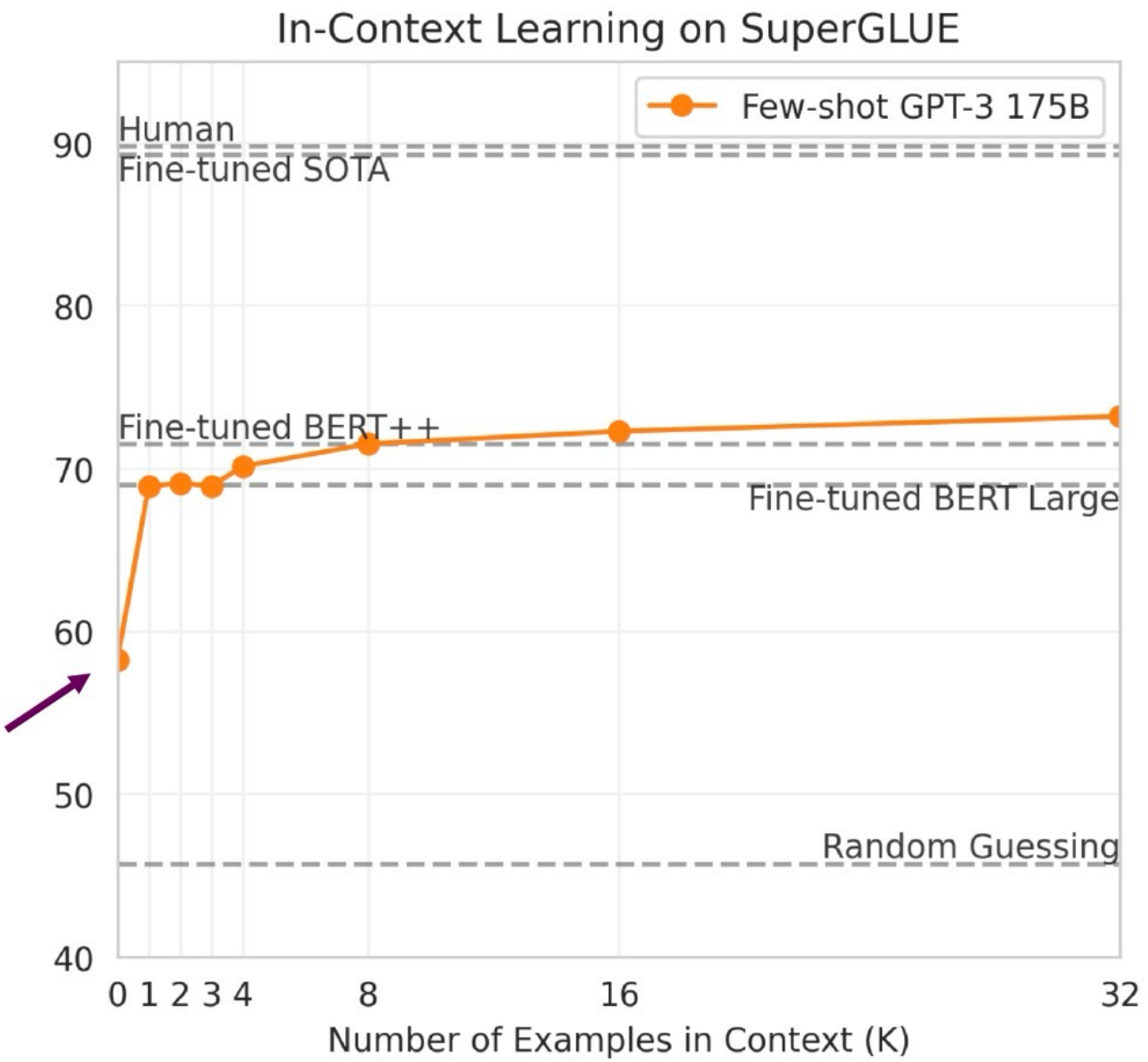
GPT-3 is the canonical example of this. The largest T5 model had 11 billion parameters.

GPT-3 has 175 billion parameters.

Emergent few-shot learning

Zero-shot

```
1  Translate English to French:
2  cheese =>
```

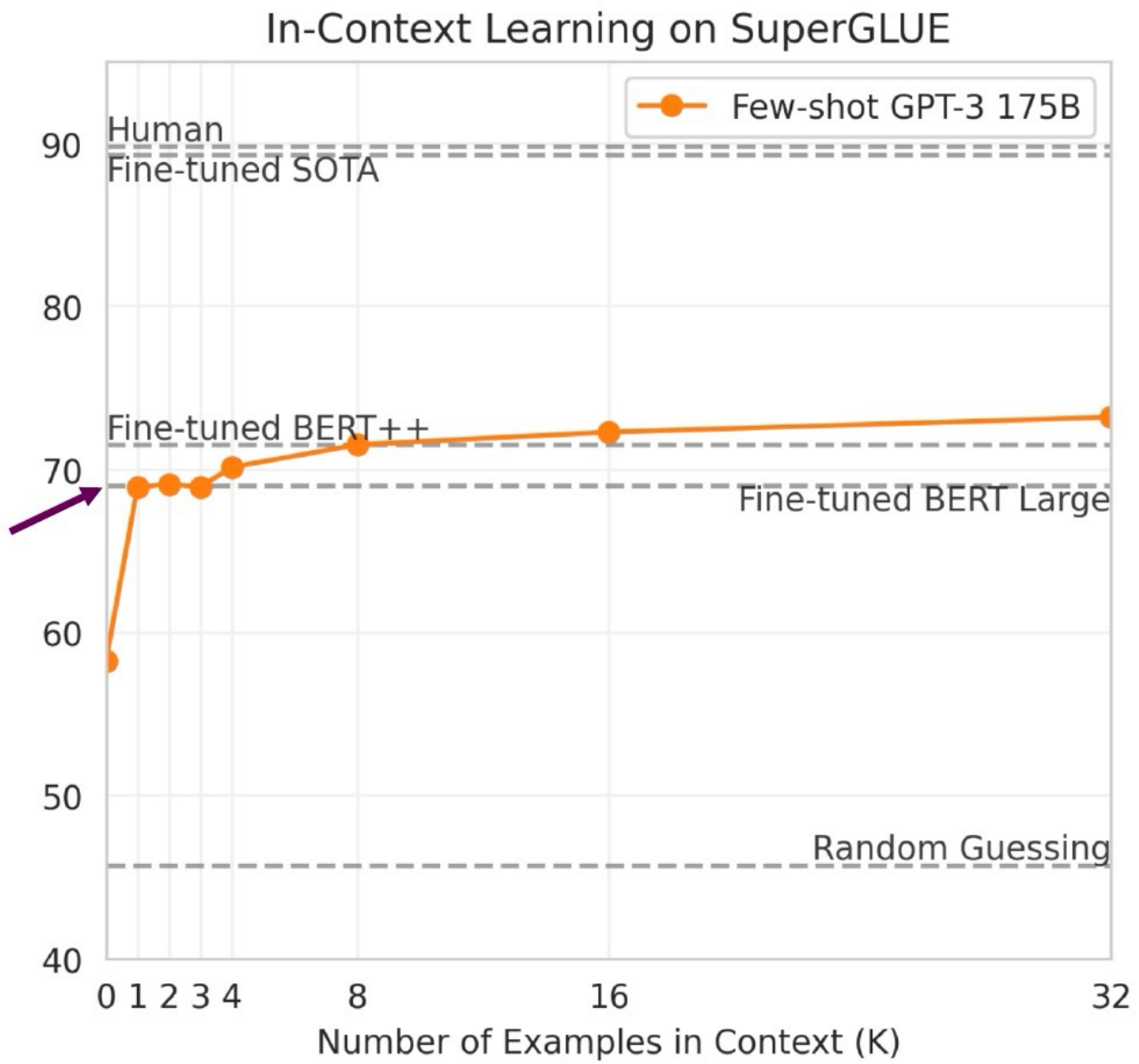


[Brown et al., 2020]

Emergent few-shot learning

One-shot

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

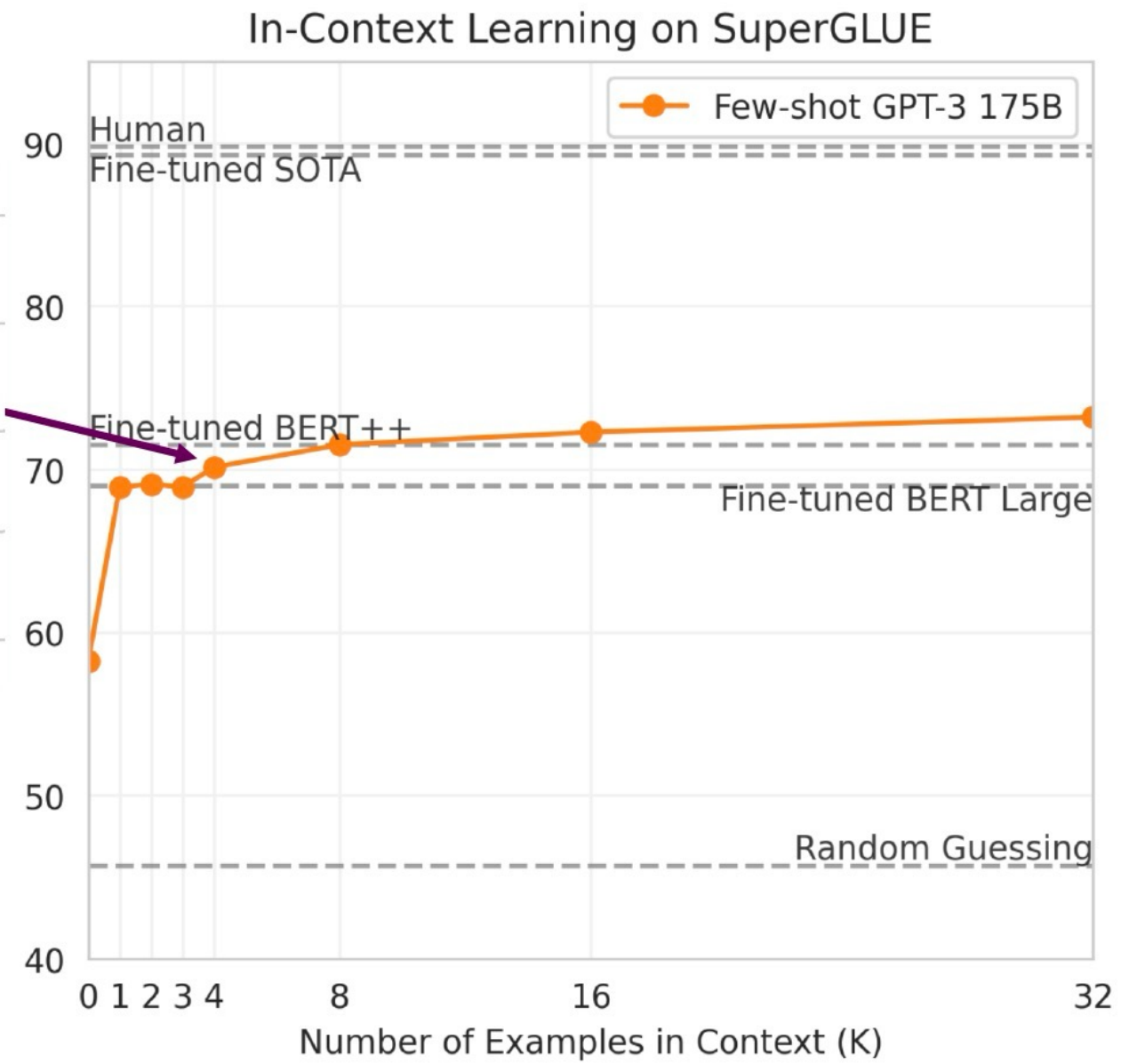


[Brown et al., 2020]

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 =>
```



[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

Standard 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: 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 answer is 27. ✗

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. ✓

[Wei et al., 2022; also see Nye et al., 2021]

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. ✓

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

[Wei et al., 2022; also see Nye et al., 2021]

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.* ✓

[Kojima et al., 2022]

Current State of the Art LLMs

- A growing number of LLMs:
  - OpenAI's GPT (3.5, 4, 4o)
  - Meta's Llama (Llama-2-7B, -13B, -70B, Llama-3-8B and -70B)
  - Google's Gemini (Nano, Flash, Pro, Ultra)
  - Mistral AI's Mixtral, Anthropic's Claude, …

- How are they trained?
  1. **Pre-training** using vanilla LM objective, i.e. predict next word.
  2. **Instruction fine tuning**, using large amounts of supervision <*input*, *instruction*> → <*correct output*>
  3. **Reinforcement Learning from Human Feedback** (RLHF).

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 [Ouyang et al., 2022].

Language modeling ≠ assisting users

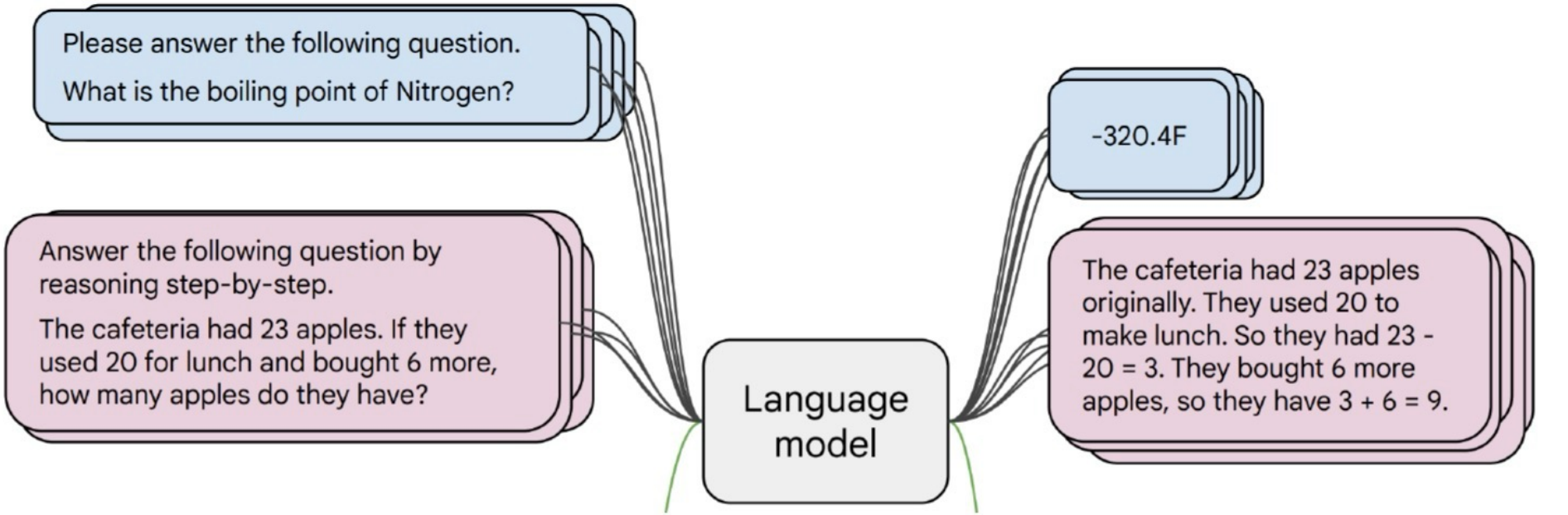
PROMPT *Explain the moon landing to a 6 year old in a few sentences.*

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 [Ouyang et al., 2022].
Finetuning to the rescue!

Instruction finetuning

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



- Evaluate on **unseen tasks**

[FLAN-T5; Chung et al., 2022]

Instruction finetuning

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

Before instruction finetuning

The reporter and the chef will discuss their favorite dishes.
The reporter and the chef will discuss the reporter's favorite dishes.
The reporter and the chef will discuss the chef's favorite dishes.
The reporter and the chef will discuss the reporter's and the chef's favorite dishes.

✖ **(doesn't answer question)**

Highly recommend trying FLAN-T5 out to get a sense of its capabilities:
https://huggingface.co/google/flan-t5-xxl

[Chung et al., 2022]

Instruction finetuning

Model input (Disambiguation QA)

Q: In the following sentences, explain the antecedent of the pronoun (which thing the pronoun refers to), or state that it is ambiguous.

Sentence: The reporter and the chef will discuss their favorite dishes.

Options:
(A) They will discuss the reporter's favorite dishes
(B) They will discuss the chef's favorite dishes
(C) Ambiguous

A: Let's think step by step.

After instruction finetuning

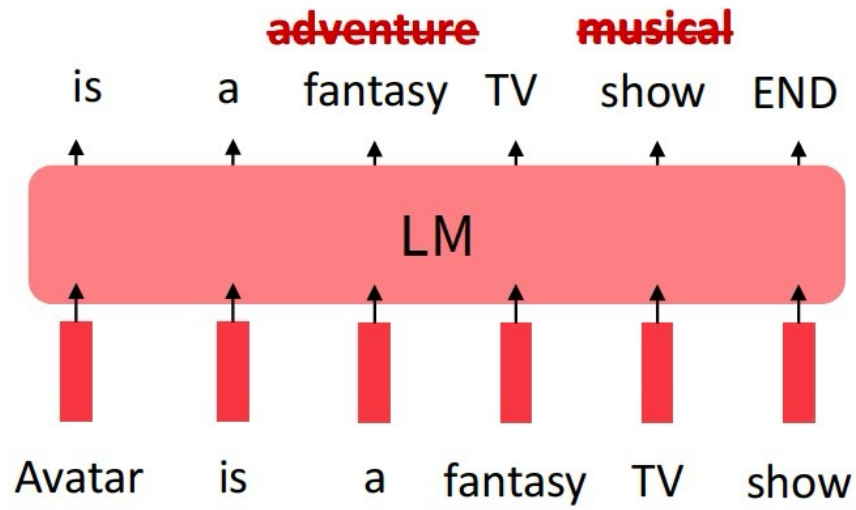
The reporter and the chef will discuss their favorite dishes does not indicate whose favorite dishes they will discuss. So, the answer is (C). ✅

Highly recommend trying FLAN-T5 out to get a sense of its capabilities:
https://huggingface.co/google/flan-t5-xxl

[Chung et al., 2022]

Limitations of instruction finetuning?

- One limitation of instruction finetuning is obvious: it's **expensive** to collect ground-truth data for tasks.
- But there are other, subtler limitations too. Can you think of any?
- **Problem 1:** tasks like open-ended creative generation have no right answer.
  - *Write me a story about a dog and her pet grasshopper.*
- **Problem 2:** language modeling penalizes all token-level mistakes equally, but some errors are worse than others.
- Even with instruction finetuning, there a mismatch between the LM objective and the objective of "satisfy human preferences"!
- Can we **explicitly attempt to satisfy human preferences?**



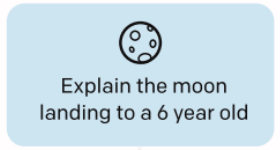
Reinforcement Learning from Human Feedback (RLHF)

Step 1

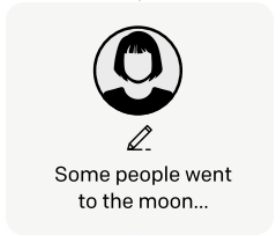
Collect demonstration data, and train a supervised policy.

30,000 tasks!

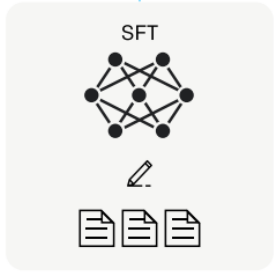
A prompt is sampled from our prompt dataset.



A labeler demonstrates the desired output behavior.



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

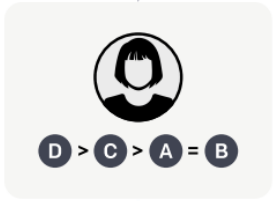


Step 2

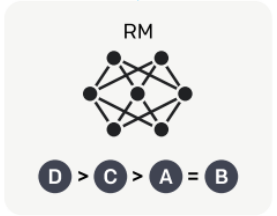
Collect comparison data, and train a reward model.

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

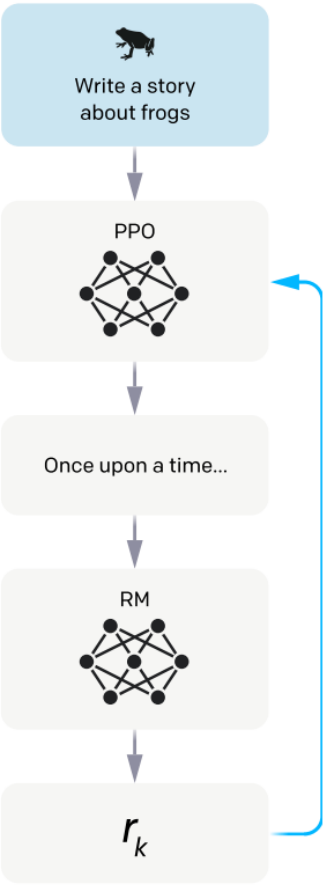
Optimize a policy against the reward model using reinforcement learning.

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.



Ouyang et al., NeurIPS 2022

Stiennon et al., NeurIPS 2020

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.

[Ouyang et al., 2022]

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!)

https://openai.com/blog/chatgpt/

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!)

https://openai.com/blog/chatgpt/

Explain in a few sentences what methods were used to train ChatGPT. Taylor your explanation to the students at the Summer School on Data Science, who by now know machine learning and language modeling techniques.

ChatGPT can make mistakes. Check important info.

ChatGPT was trained using a combination of supervised learning and reinforcement learning. Initially, it was pre-trained on a large corpus of text from the internet using unsupervised learning, specifically leveraging the transformer architecture to predict the next word in a sequence. This pre-training step involved self-supervised learning to capture a broad understanding of language. Subsequently, supervised fine-tuning was performed where human AI trainers provided examples of desired outputs. Finally, reinforcement learning from human feedback (RLHF) was applied, where the model was fine-tuned based on feedback to optimize for more accurate and contextually appropriate responses.

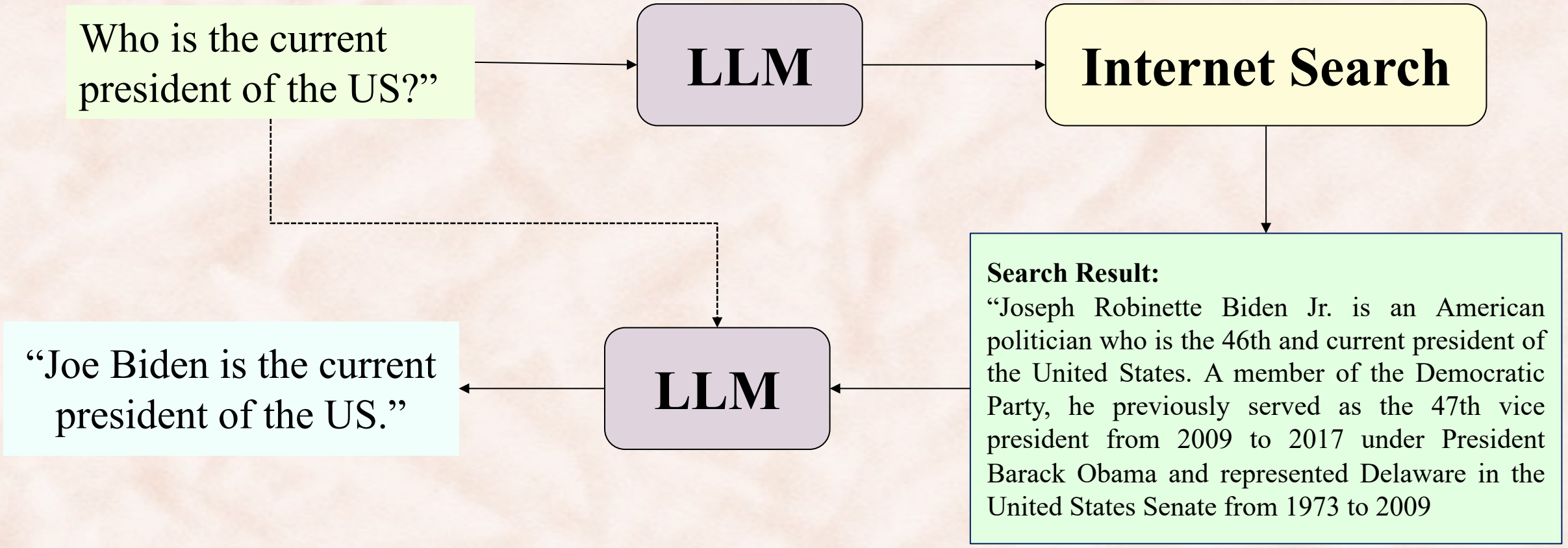
Now explain the same thing to a construction worker.

To train ChatGPT, we first taught it by showing it lots of text from the internet, so it could learn how language works. Then, we fine-tuned it by giving it examples of good answers from humans. Finally, we had people give feedback on its responses and adjusted it based on what they liked or didn't like, making it even better at giving useful answers.

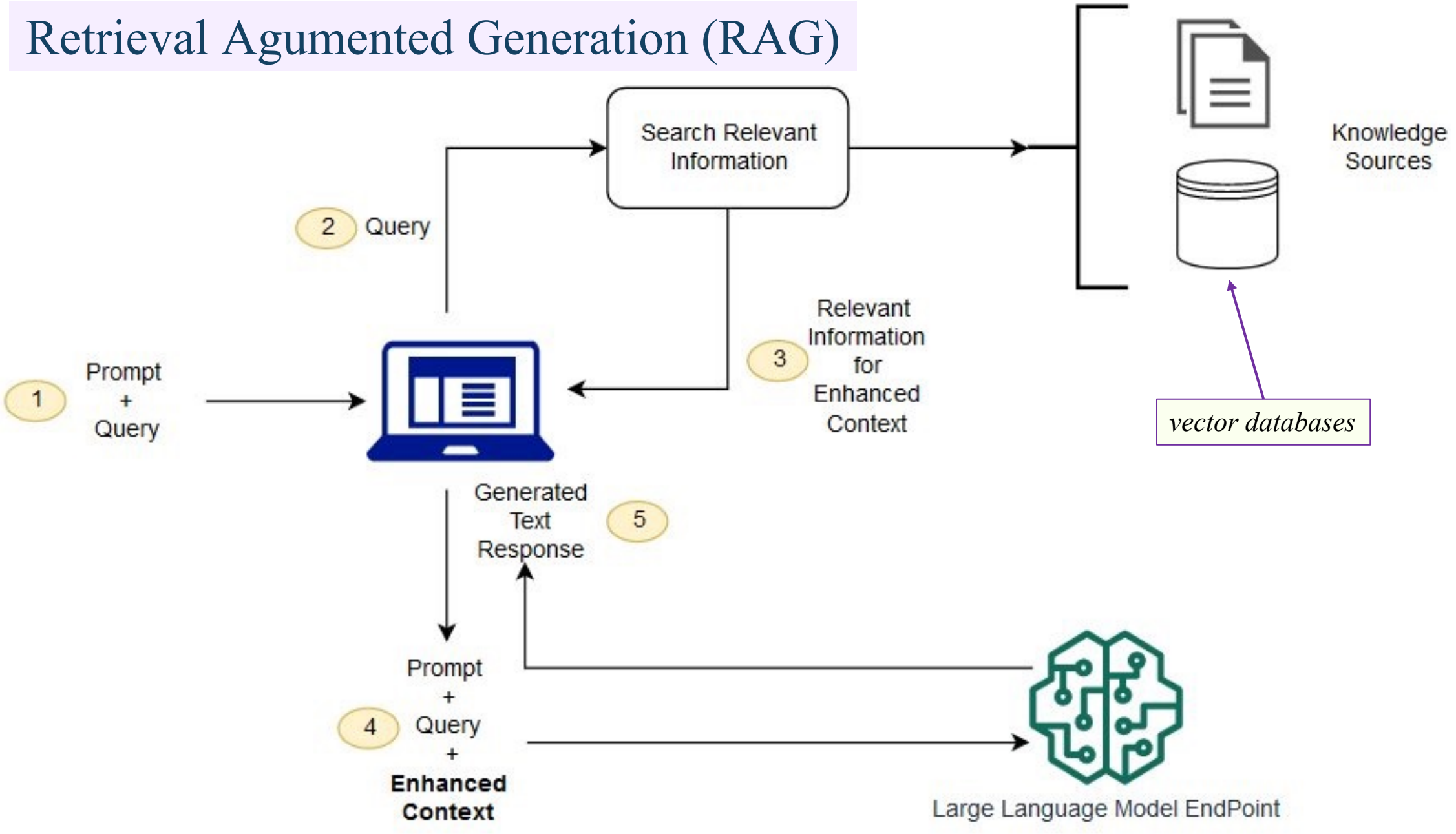
Retrieval Agumented Generation (RAG)

- Traditional LLMs are often limited to their pre-trained knowledge and data.
- This could lead to potentially outdated or inaccurate responses.
  - Hallucinations …
- RAG overcomes this by granting LLMs access to external information sources, ensuring accurate and up-to-date answers.
  - Documents are accessed through **search engines** or through **vector stores**:
    - Vector Stores are databases that stores text, images, videos as embeddings.
    - Why do we use vector stores? Use dot-products to find :
      - most similar *documents* for a query.
      - best *few-shot examples* for a test example.

Retrieval Agumented Generation (RAG)



Retrieval Agumented Generation (RAG)



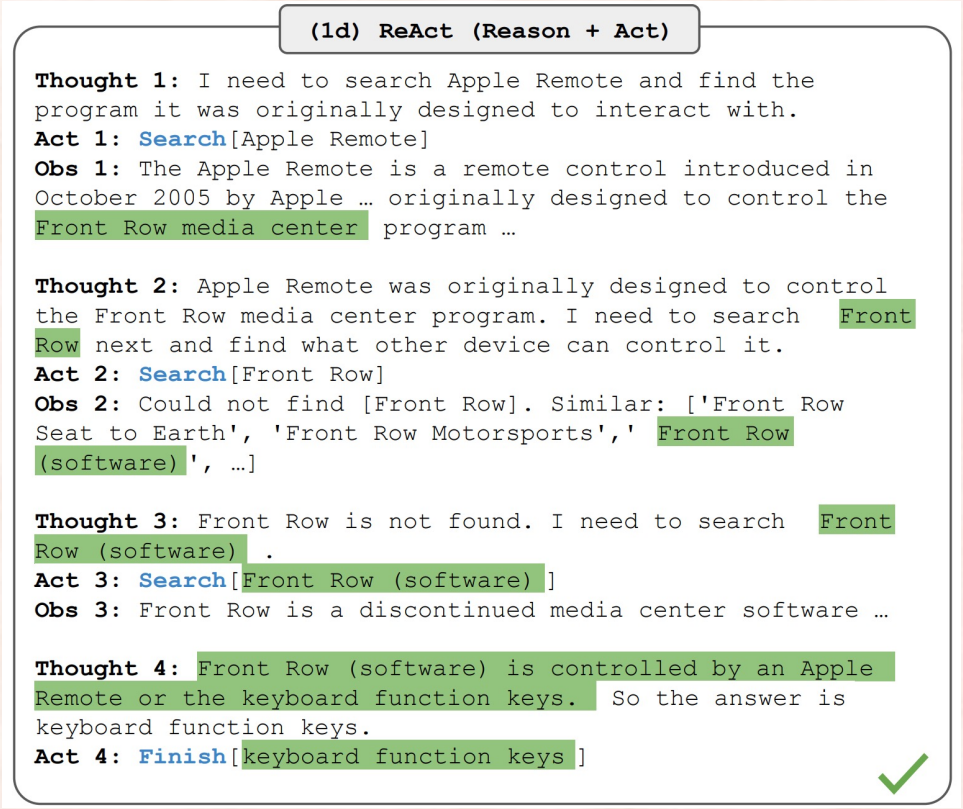
https://aws.amazon.com/what-is/retrieval-augmented-generation

The ReAct Framework

Yao et al., ICLR 2023

- A style of prompting where LLMs are instructed to generate both **reasoning traces** and **task-specific actions** in an interleaved manner.

- This utilizes a format that is simple to parse:
  - **Question**: The input to the model.
  - **Thought**: A place for the model to state intentions, which tends to increase performance.
  - **Action**: The tool (if any) does the model wants to use.
    - **Action Input**: The input to the tool.
  - **Observation**: The output of the tool.
  - **Thought**: "I believe I have the final answer."
  - **Final Answer**: The answer the model gives to the user.



ReAct Framework

- A style of prompting where LLMs are instructed to generate both **reasoning traces** and **task-specific actions** in an interleaved manner.
  - Generating reasoning traces allow the model to induce, track, and update action plans, and even handle exceptions.
  - The action step allows to interface with and gather information from external sources such as knowledge bases or environments.
    - This leads to more reliable and factual responses.
  - ReAct can outperform several state-of-the-art baselines on language and decision-making tasks.
  - ReAct also leads to improved human interpretability and trustworthiness of LLMs.
  - The best approach uses ReAct combined with chain-of-thought (CoT) that allows use of both internal knowledge and external information obtained during reasoning.

ReAct Prompting

https://python.langchain.com/v0.1/docs/modules/agents/agent_types/react/

- ReAct prompting is an approach for enhancing the decision making of LLMs, for example, choosing which tools to use for a given situation.
- The structure of the ReAct Loop:
  - <question> (<thought> <action> <action input> <observation>)* <thought> <answer>

Scenario 1

Question: Who is the current president of the US?

Thought: I should use a search engine to find relevant data.
Action: Open Google
Action Input: "Who is the current president of the US?"
Observation: The current president is Joe Biden.

Thought: I now know the final answer.
Final Answer: Joe Biden

Scenario 2

Question: What is 2 + 2?

Thought: I now know the final answer.
Final Answer: 4