Pre/Post-training

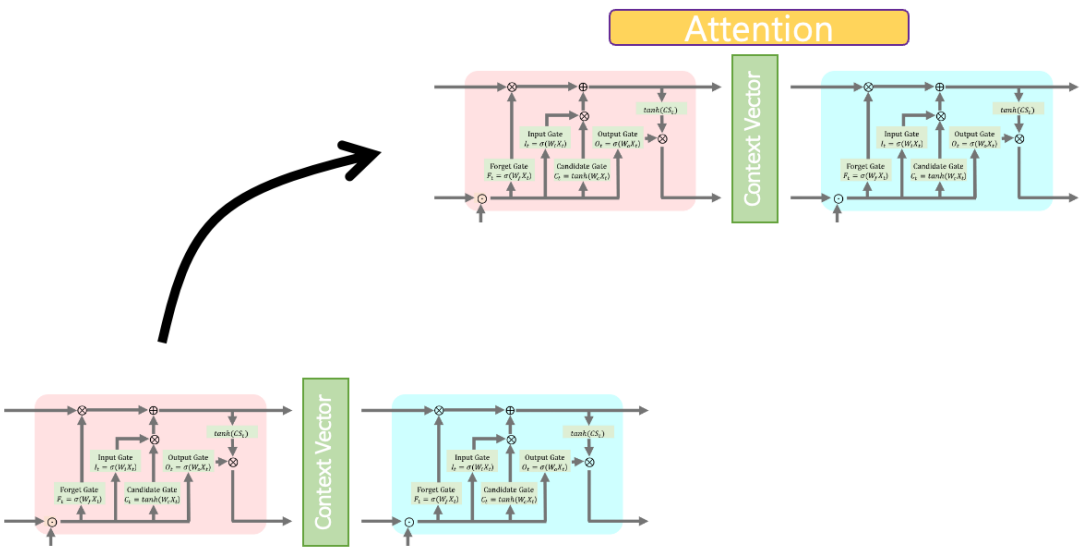
Mar 17, 2026

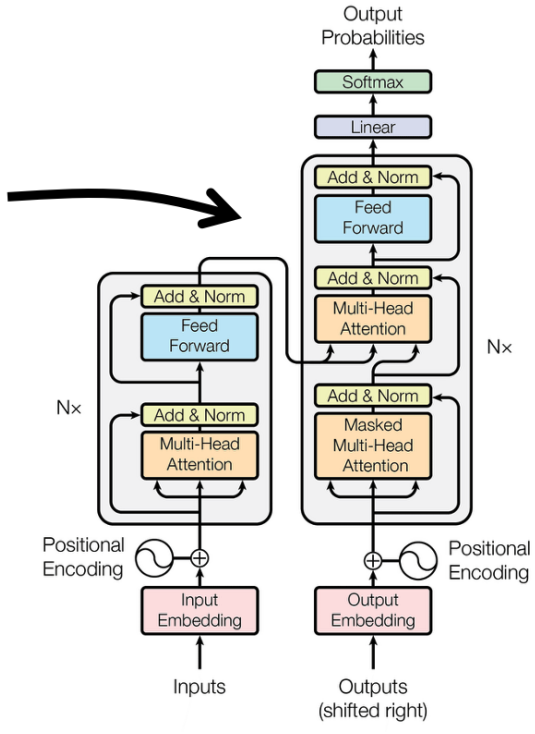
*Acknowledgment: Slides based on materials by CS224N @ Stanford University.

Outline

Outline

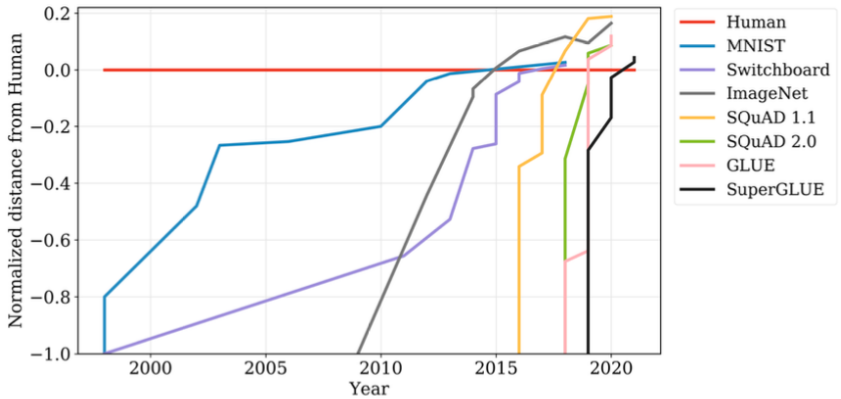
Progress

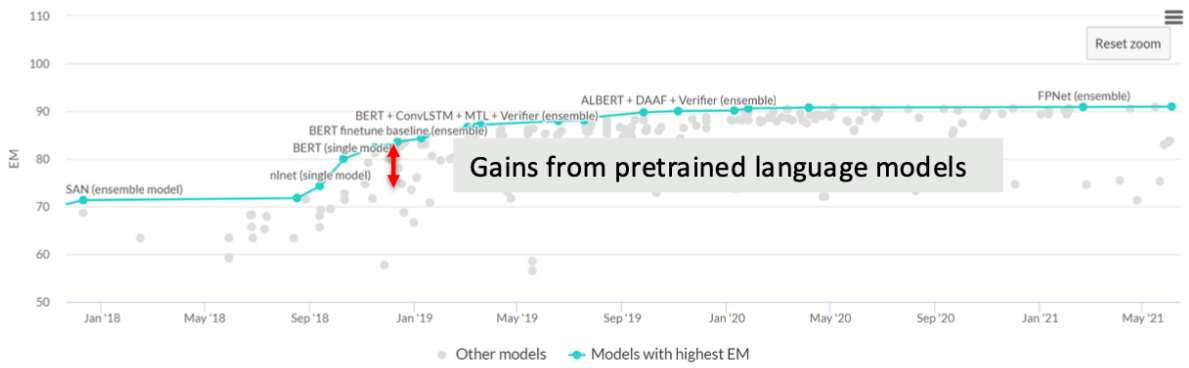

Attention
Context Vector
Context Vector


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Outline

The pretraining revolution

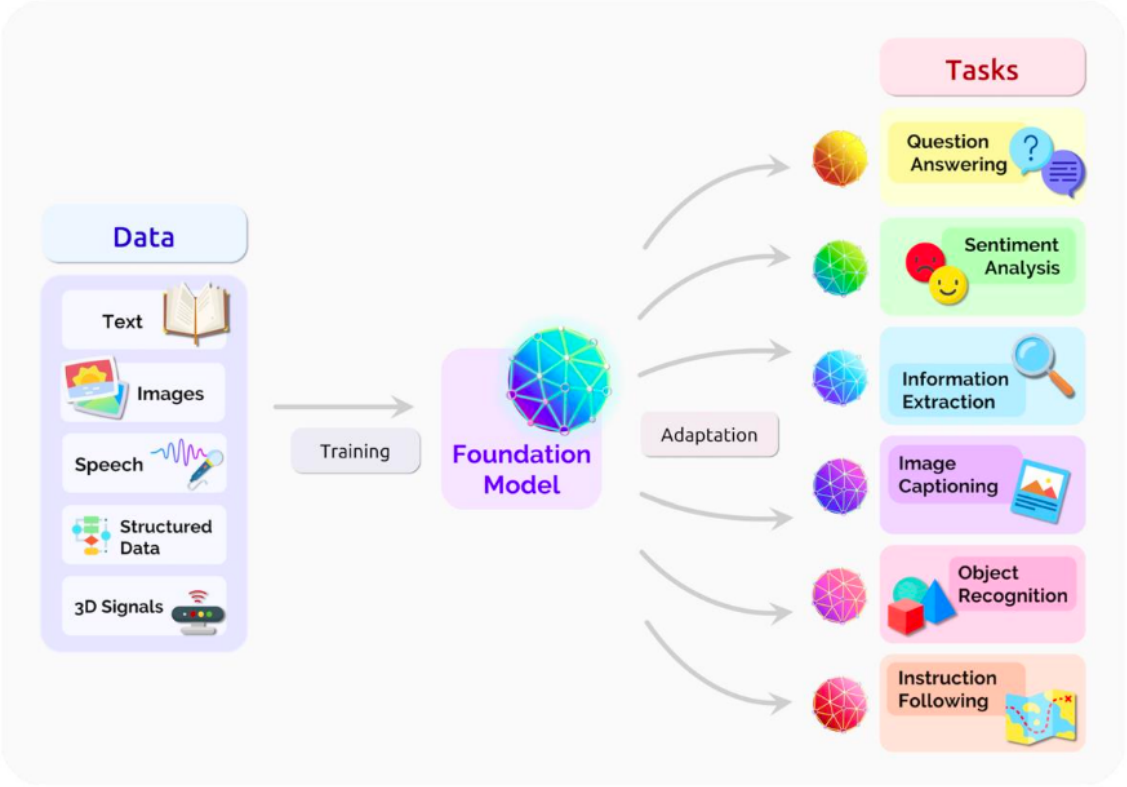




Pretraining has had a major impact on how well NLP systems work.

RIT Rochester Institute of Technology

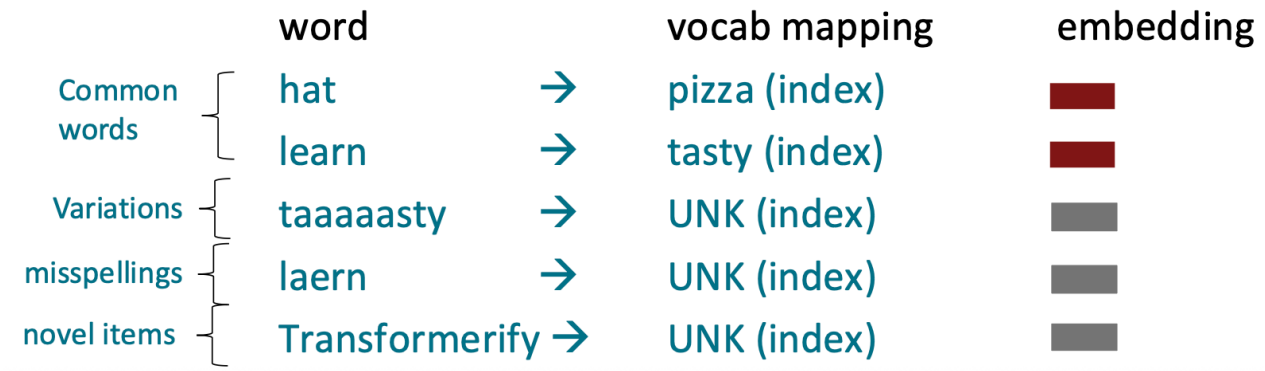
Pretraining: Scaling unsupervised learning on the internet



- Make sure your model can process large-scale, diverse datasets
- Don't use labeled data (self-supervised)

Word structure and subword models

- Assumed a fixed vocabulary derived from the training data
- Vocabulary size: tens of thousands of word types
- Unseen words at test time mapped to a single `UNK` token
- BUT people always come up with new words



Word structure and subword models

- Finite vocabulary assumptions make even less sense in many languages
  - Many languages exhibit complex morphology or word structure

Example: Swahili

English	Pronoun	Verb Prefix
me	mimi	ni
you	wewe	u
he/she	yeye	a
us/we	sisi	tu
you (pl.)	ninyi	m
they	wao	wa

Tenses

Past, Present, Future

A helpful trick to remember the tense prefixes is by using the common girls name "Natalie". Except in our case, its "na-ta-li".

Tense	Verb Prefix
Present	na
Future	ta
Past	li

Here is an example of the conjugations of "-lala" the verb for "sleep". The pronoun prefix is in **blue**. The tense prefix is in **red**.

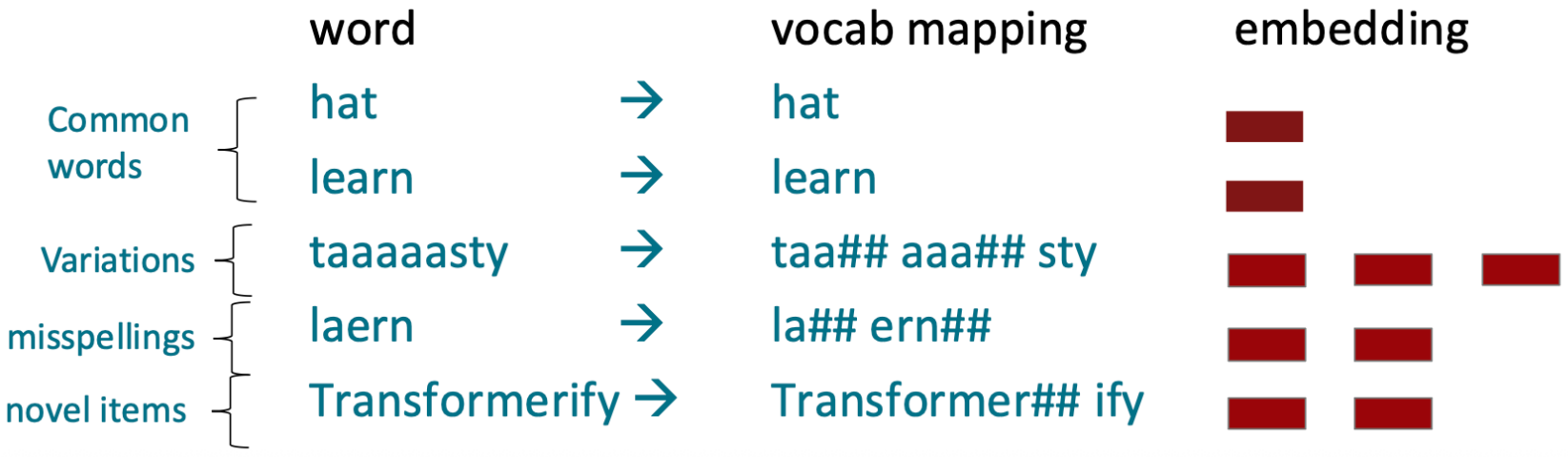
Pronoun	Past	Present	Future
mimi	nililala	ninalala	nitalala
wewe	ulilala	unalala	utalala
yeye	alilala	analala	atalala
sisi	tulilala	tunalala	tutalala
ninyi	mlilala	mnalala	mtalala
wao	walilala	wanalala	watalala

Byte-Pair Encoding (BPE)

- **Subword modeling:** Represent words using smaller units (e.g., characters, morphemes, byte sequences)
  - Learn a vocabulary of frequent subword units
  - Words are split into known subwords at train and test time
  - Example: *unhappiness* $\rightarrow$ un + happi + ness
- **Byte-Pair Encoding (BPE):** A simple way to build subword vocabularies
  1. Start with characters + end-of-word symbol
  2. Merge the most frequent adjacent pair
  3. Replace all occurrences with the merged token
  4. Repeat until reaching desired vocab size
- Example merges:
  - l + o $\rightarrow$ lo
  - lo + w $\rightarrow$ low
  - low + er $\rightarrow$ lower
- Widely used in MT; modern variants (e.g., WordPiece) power pretrained models.

Word structure and subword models

e.g., WordPiece (BERT)



Where we were: Pretrained word embeddings

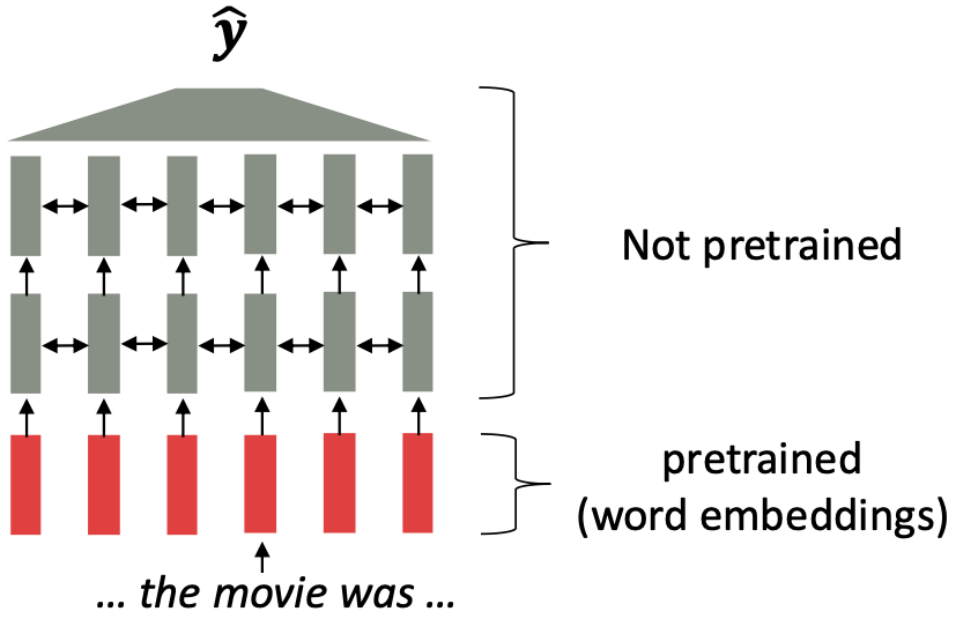
Circa 2017:

- Start with pretrained word embeddings (no context)
- Learn how to incorporate context in an LSTM or Transformer while training on the task

Where we were: Pretrained word embeddings

Some **issues**:

- The training data we have for our downstream task (e.g., question answering) must be sufficient to teach all contextual aspects of language
- Most of the parameters in our network are randomly initialized



[Recall, *movie* gets the same word embedding, no matter what sentence it shows up in]

Where we're going: Pretrained whole models

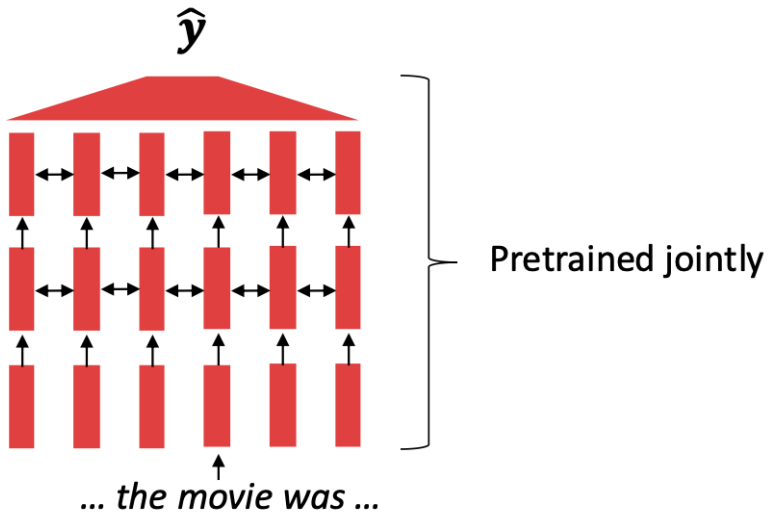
In mordern NLP:

- All (or almost all) parameters in NLP networks are initialized via pretraining
- Pretraining methods hide parts of the input from the model, and train the model to reconstruct those parts

Where we're going: Pretrained whole models

Exceptionally effective at building strong:

- representations of language
- parameter initializations for strong NLP models
- probability distributions over language that we can sample from $p(w_t \mid w_{<t})$



[This model has learned how to represent entire sentences through pretraining]

What can we learn from reconstructing the input?

RIT is located in _____, New York.

What can we learn from reconstructing the input?

RIT is located in _____, New York.

(Location)

What can we learn from reconstructing the input?

I put ___ fork down on the table.

What can we learn from reconstructing the input?

I put ___ fork down on the table.

(Grammar)

What can we learn from reconstructing the input?

The woman walked across the street, checking for the traffic over ___ shoulder.

What can we learn from reconstructing the input?

The woman walked across the street, checking for the traffic over ___ shoulder.
(co-reference)

What can we learn from reconstructing the input?

I went to the ocean to see the fish, turtle, seals, and ___.

What can we learn from reconstructing the input?

I went to the ocean to see the fish, turtle, seals, and ___.
(semantics)

What can we learn from reconstructing the input?

Overall, the value I got from the two hours watching it was the sum total of popcorn and the drink. The movie was ___.

What can we learn from reconstructing the input?

Overall, the value I got from the two hours watching it was the sum total of popcorn and the drink. The movie was ___. (sentiment)

What can we learn from reconstructing the input?

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ___.

What can we learn from reconstructing the input?

I was thinking about the sequence that goes 1, 1, 2, 3, 5, 8, 13, 21, ___. (math)

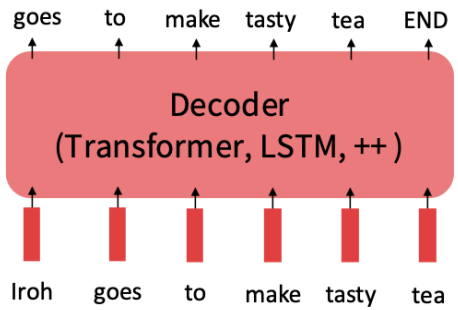
Pretraining through language modeling

Recall the language modeling task:

- Model $p_\theta(w_t \mid w_{1:t-1})$, the probability distribution over words given their past context.
- There is abundant training data for this task (e.g., large English corpora).

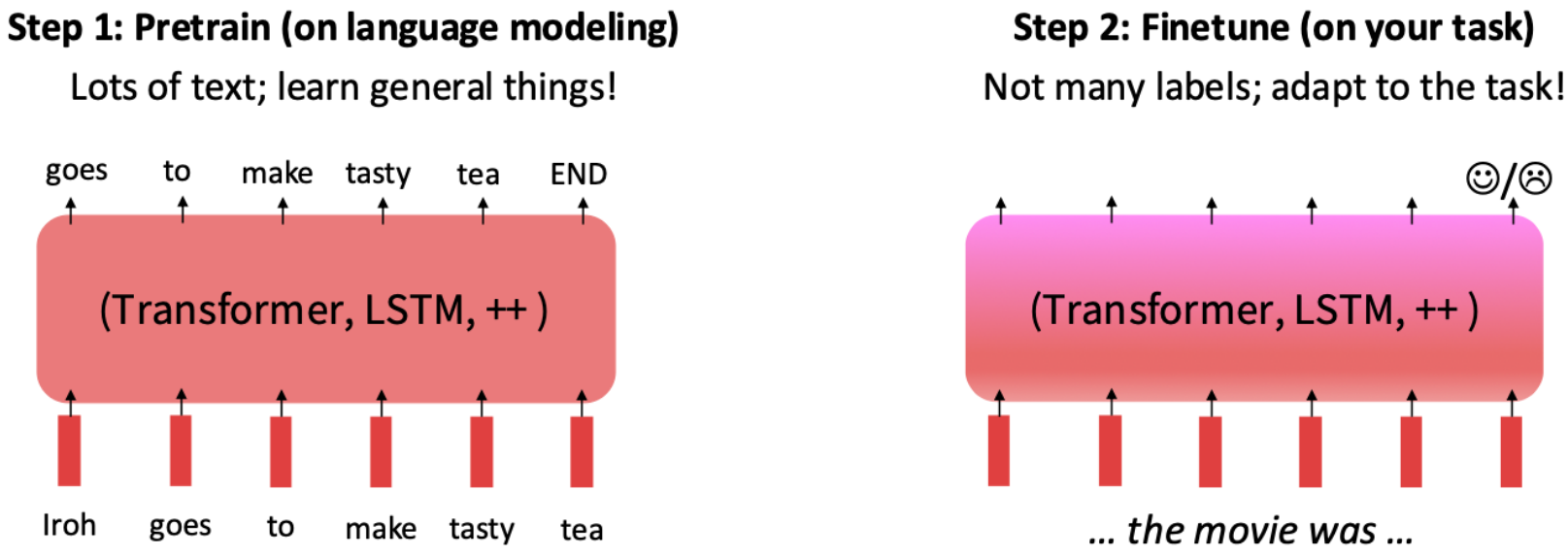
Pretraining through language modeling:

- Train a neural network to perform language modeling on a large-scale text corpus.
- Save the learned network parameters (i.e., keep all the weights) for downstream tasks.



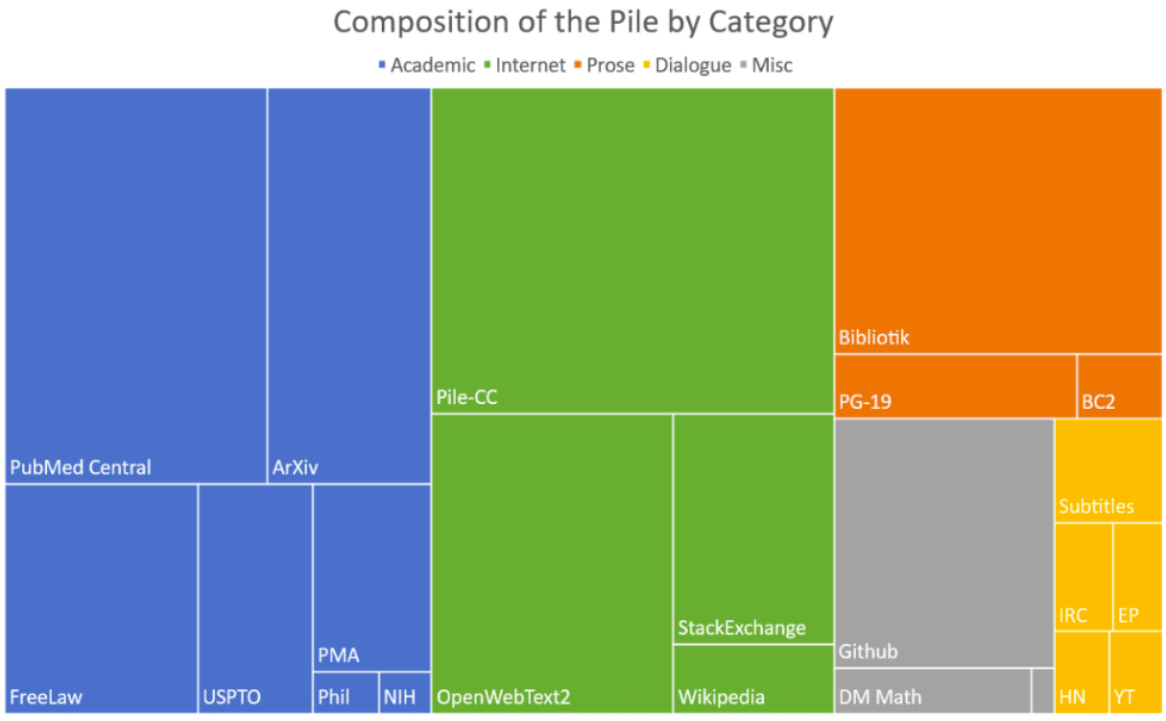
The pretraining/finetunign paradigm

Pretraining can improve NLP applications by serving as parameter initialization.



Works exceptionally well (compared to training from scratch)!

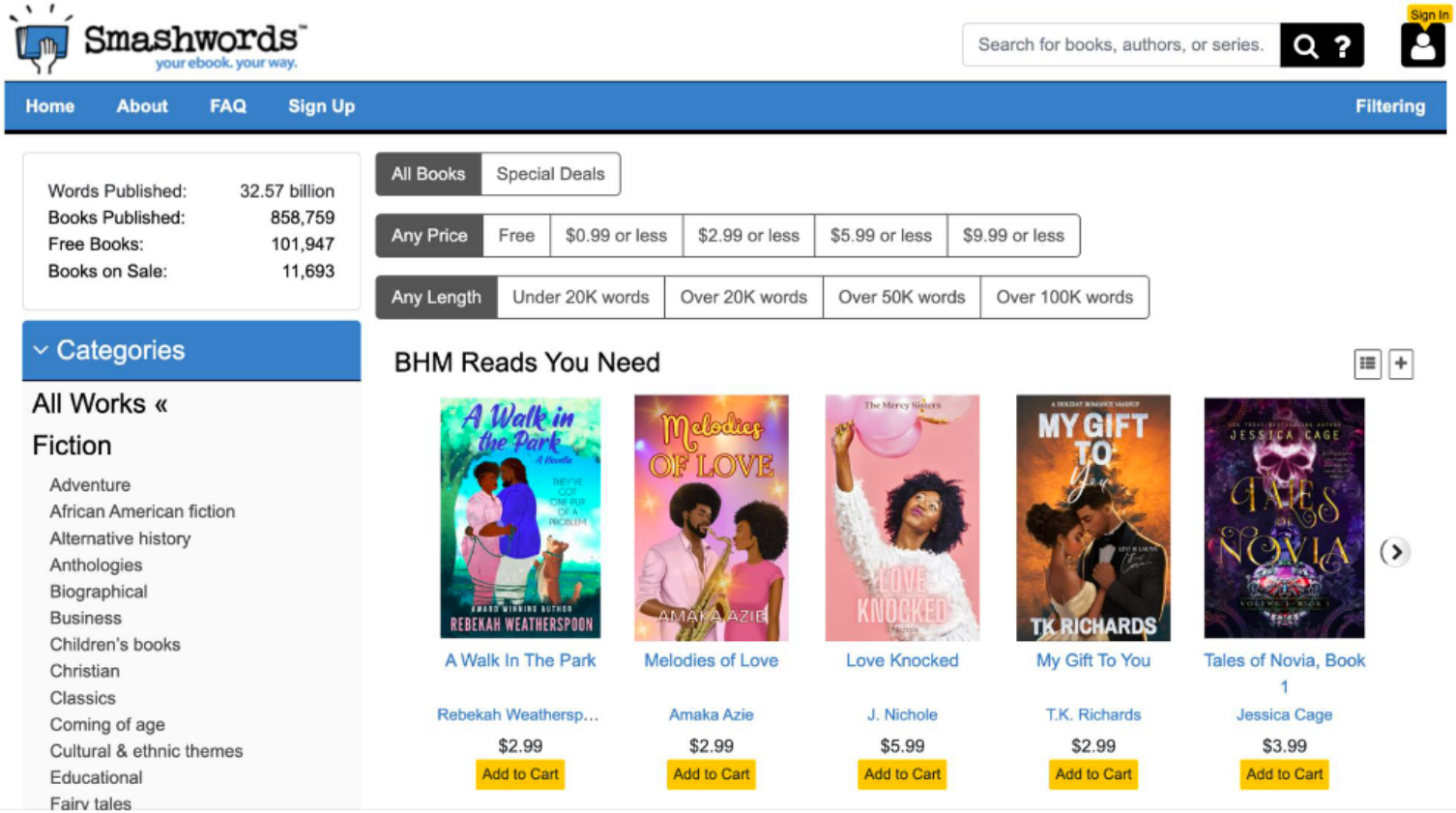
Where does this data come from?



Model	Training Data
BERT	BookCorpus, English Wikipedia
GPT-1	BookCorpus
GPT-3	CommonCrawl, WebText, English Wikipedia, and 2 book databases (“Books 1” and “Books 2”)
GPT-3.5+	Undisclosed

Bookcorpus.. what's that?

Bookcorpus.. what's that?

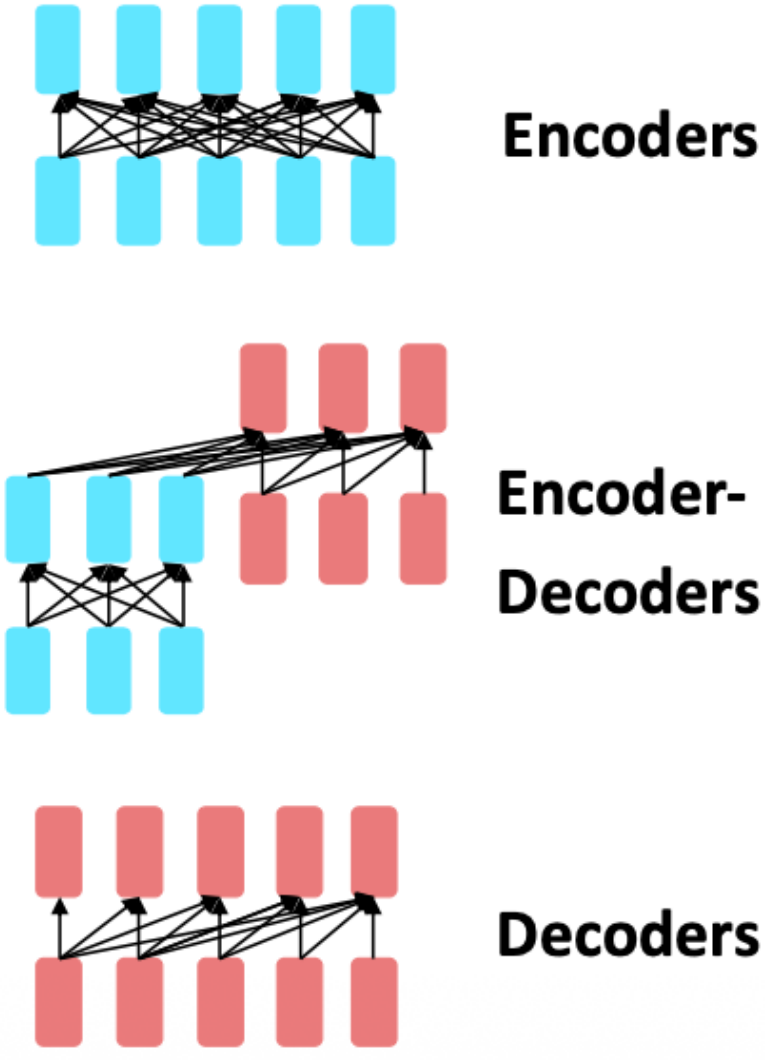


- Scraped ebooks from the internet – highly controversial

Outline

Pretraining for three types of architectures

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



Pretraining Encoders

- Encoders use bidirectional context.
- Therefore, we cannot use standard left-to-right language modeling.
- **Idea:** Replace some fraction of words in the input with a special `[MASK]` token and train the model to predict those words (Masked LM)

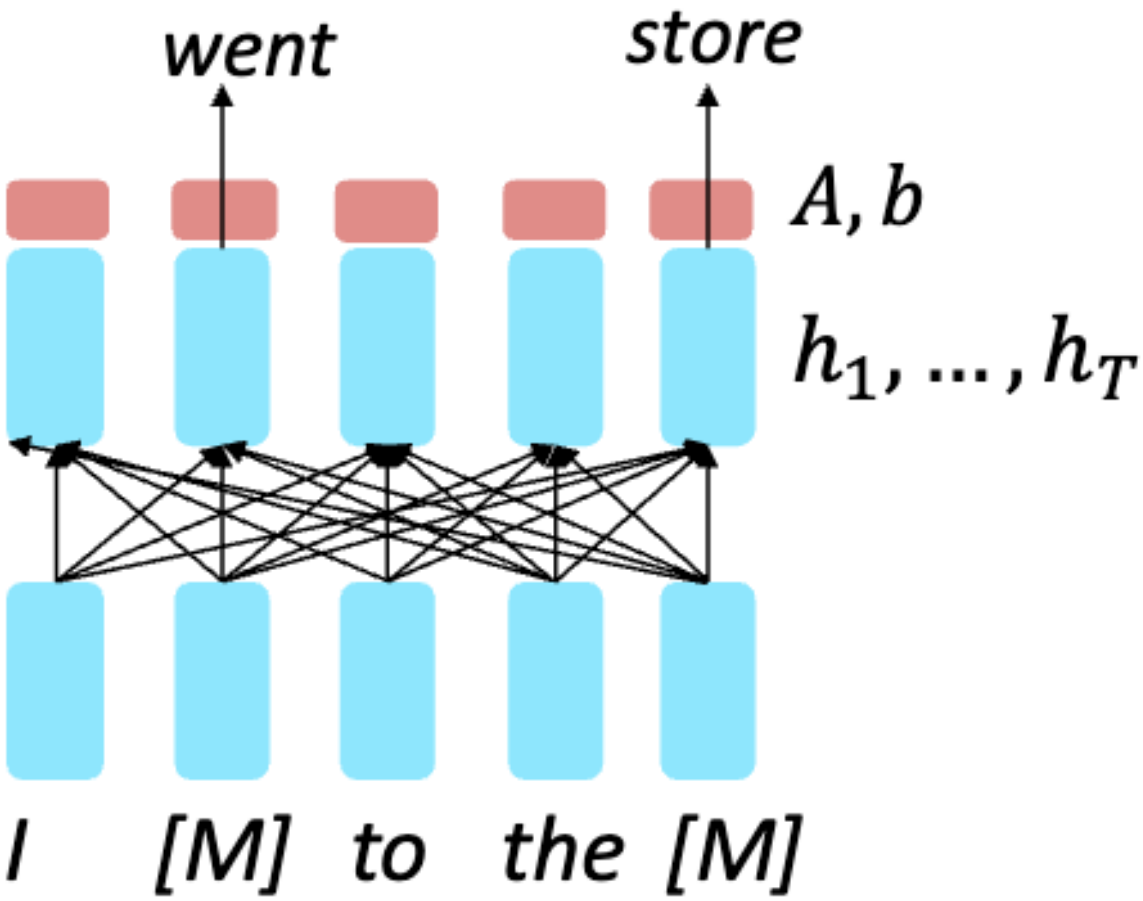
$$h_1, \ldots, h_T = \text{Encoder}(w_1, \ldots, w_T)$$

$$y_i = Ah_i + b$$

Only add loss terms for words that are **masked out**.
If $\tilde{x}$ is the masked version of x, we are learning:

$$p_\theta(x \mid \tilde{x})$$

Pretraining Encoders



[Devlin et al., 2018]

BERT; Evaluation

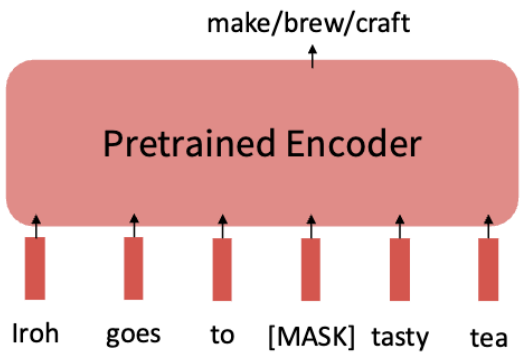
BERT was massively popular and hugely versatile; finetuning BERT led to new state-of-the-art results on a broad range of tasks.

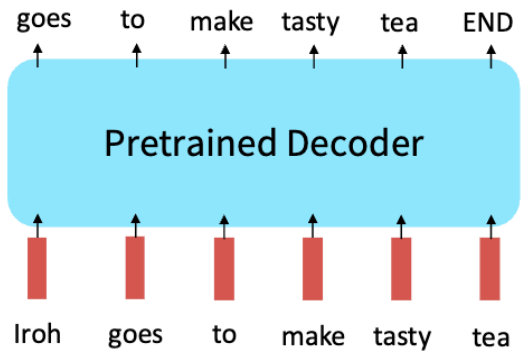
- **QQP:** Quora Question Pairs (detect paraphrase questions)
- **QNLI**: natural language inference over question answering data
- **SST-2**: sentiment analysis
- **CoLA**: corpus of linguistic acceptability (detect whether sentences are grammatical.)
- **STS-B**: semantic textual similarity
- **MRPC**: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

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 auto-regressive (1-word-at-a-time) generation* methods.





Pretrainig decoders

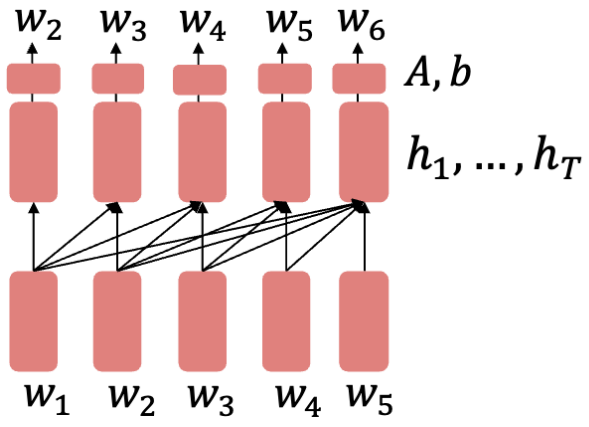
It's natural to pretrain decoders as language models and then use them as generators, finetuning their $p_\theta(w_t|w_{1:t-1})$!

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.
- 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 stand for "Generative PreTraining" or "Generative Pretrained Transformer"

Generative pretrained transformer (GPT) (Radford et al., 2018)

How do we format inputs to our decoder for finetuning tasks?

- Natural language inference: Label pairs of sentences as *entailing*, *contradiction*, *neutral*
- (premise): The man is in the doorway
- (hypothesis): The person is near the door
- (input format): [START] (premise) [DELIM] (hypothesis) [EXTRACT]
- The linear classifier is applied to the representation of the [EXTRACT] token.

GPT2

We mentioned how pretrained decoders can be used **in their capacities as language models.**

GPT-2, a larger version (1.5B) of GPT trained on more data, was shown to produce relatively convincing samples of natural language.

Context (human-written): 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.
GPT-2: The scientist named the population, after their distinctive horn, Ovid's Unicorn. These four-horned, silver-white unicorns were previously unknown to science. Now, after almost two centuries, the mystery of what sparked this odd phenomenon is finally solved. Dr. Jorge Pérez, an evolutionary biologist from the University of La Paz, and several companions, were exploring the Andes Mountains when they found a small valley, with no other animals or humans. Pérez noticed that the valley had what appeared to be a natural fountain, surrounded by two peaks of rock and silver snow.

GPT3, In-context learning, and very large models

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

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

GPT3, In-context learning, and very large models

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

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

Very large language models seem to perform some kind of learning *without gradient steps* simply from examples you provide within their contexts (in-context learning);

- GPT-3 is the canonical example of this
- The largest T5 model (encoder-decoder) had 11 billion parameters
- GPT-3 has 175 billion parameters

GPT3, In-context learning, and very large models

The in-context examples seem to specify the task to be performed, and the conditional distribution mocks performing the task to a certain extent.

Input (prefix within a single Transformer decoder context):

“ thanks -> merci

hello -> bonjour

mint -> menthe

otter -> ”

Output (conditional generations):

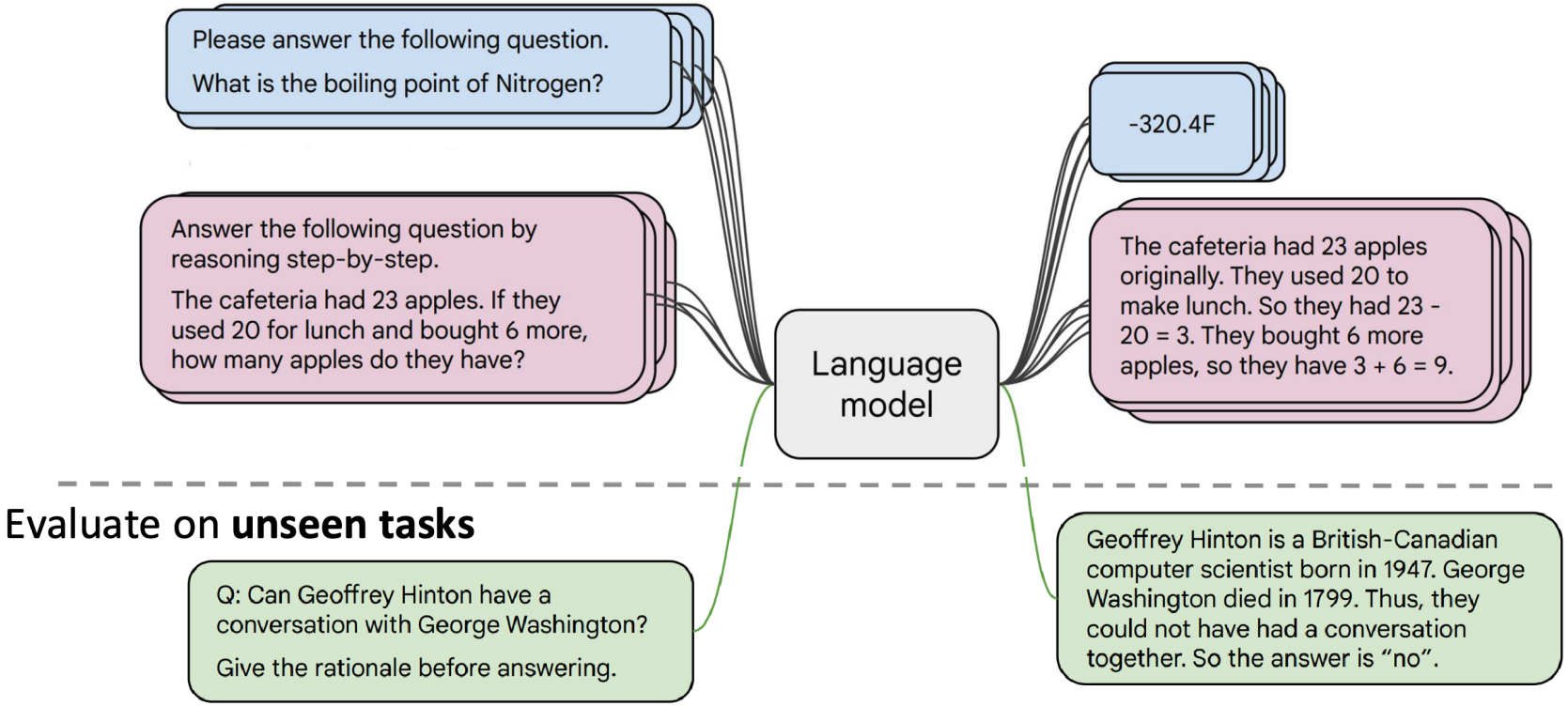
loutre...”

Outline

Instruction fine-tuning

- A training approach in which a pre-trained language model is further trained on datasets consisting of instructions paired with appropriate responses
- The model learns to follow natural-language instructions and perform a wide range of tasks

Instruction fine-tuning



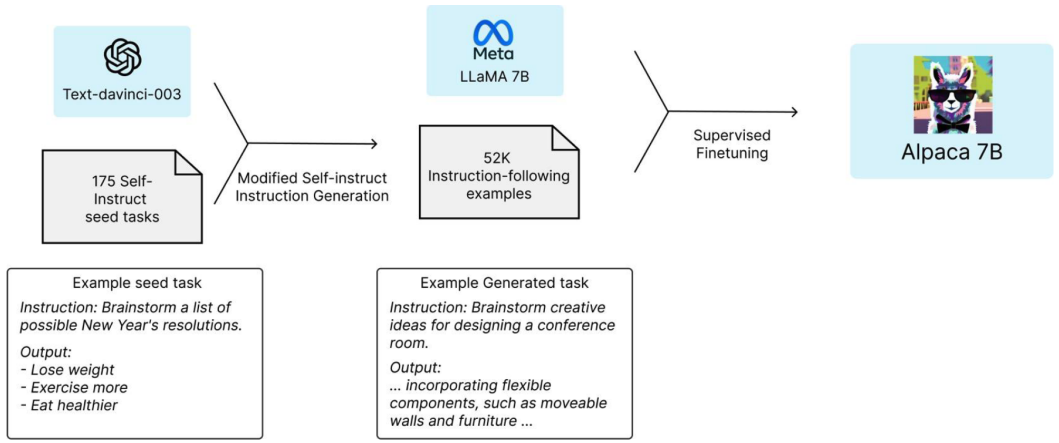
[FLAN-T5; Chung et al., 2022]

Instruction fine-tuning

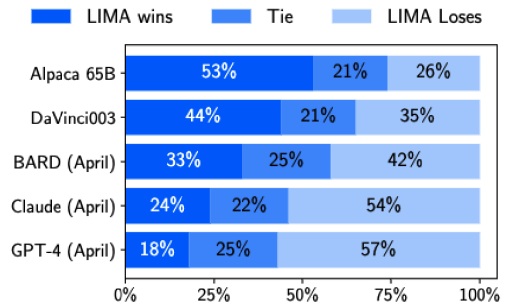
Generate instructions, input, and output from a LM [Wang et al., 2022]

- **Alpaca**: fine-tuned from the LLaMA 7B model on 52K instruction-following examples

You don't need many samples to instruction tune (e.g., "*LIMA: Less Is More for Alignment*" Zhou et al., 2023)

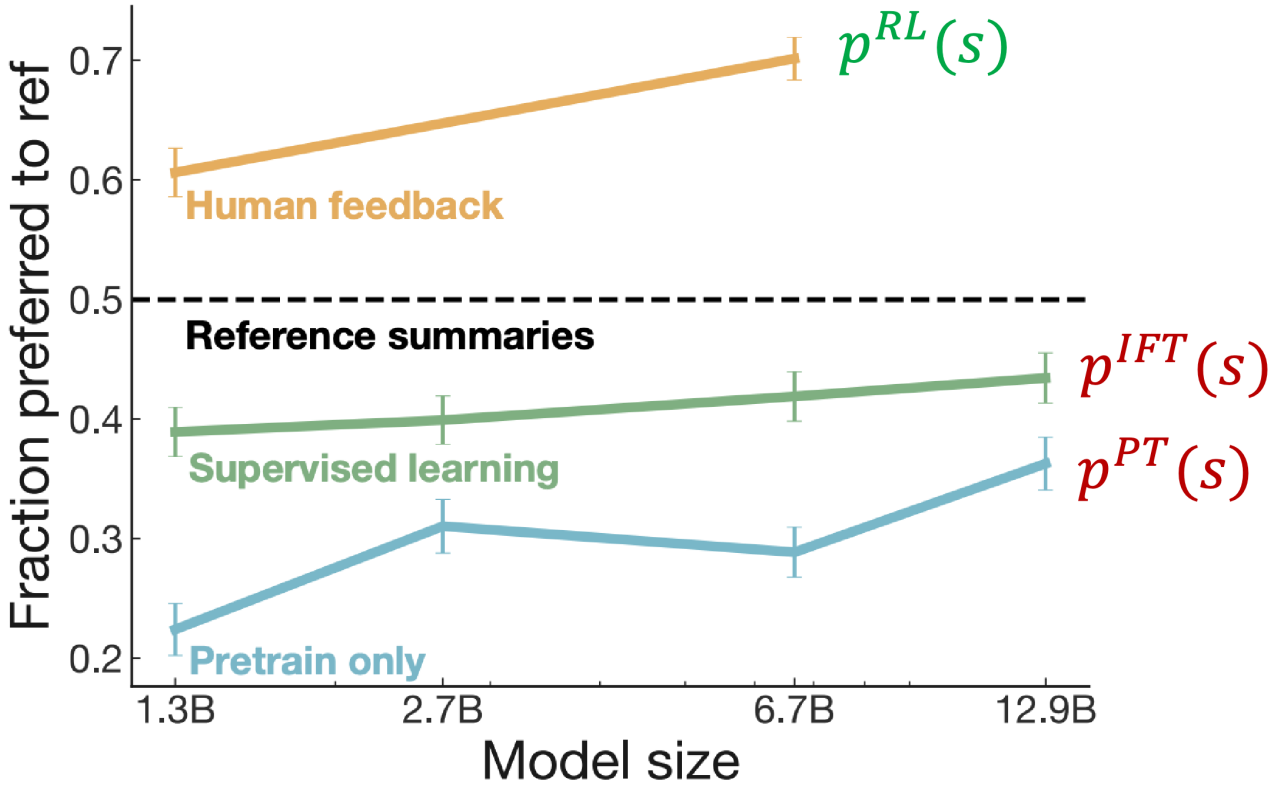


Source	#Examples
Training	
Stack Exchange (STEM)	200
Stack Exchange (Other)	200
wikiHow	200
Pushshift r/WritingPrompts	150
Natural Instructions	50
Paper Authors (Group A)	200



Reinforcement learning

- Even with instruction fine-tuning, there is a mismatch between the LM objective and the objective of "satisfy human preferences".
- How can we model this?



[Stiennon et al., 2020]

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:

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

Note: for simplicity, we show the math for a single prompt. In practice, we average over many prompts

Direct Preference Optimization (DPO)

Direct Preference Optimization (DPO) is a fine-tuning method that aligns a language model with human preferences using *paired preference data*.

- Training data consists of a prompt with two responses:
  - **Preferred response** (chosen)
  - **Rejected response**
- The model is optimized to assign a higher probability to the preferred response than to the rejected one.
- Unlike RLHF, DPO does not require a separate reward model or reinforcement learning.
- Training is regularized to stay close to a **reference model** (usually the instruction-tuned model).

Outline

Presentations

- Joshua: Scaling Instruction-Finetuned Language Models
- Arshad: Instruction Tuning on Open Resources

On Thursday

- Adit, Thejas: IASC: Interactive Agentic System for ConLangs (constructed languages)
- Lab 8: Interactive Agentic System for ConLangs