Introduction to Large Language Models

Spring 2026

LLM Basics

(Some slides adapted from Ralph Grishman at NYU,
Yejin Choi at UWashington, N. Tomura at UDepaul, Jurafsky and
Martin, CS224N, CS224d at Stanford and other resourses on the web)

Large language models

- Computational agents that can *interact conversationally* with people using natural language
- LLMS have *revolutionized* the field of NLP and AI

Language models

- Remember the simple n-gram language model
  - Assigns probabilities to sequences of words
  - Generate text by sampling possible next words
  - Is trained on counts computed from lots of text
- Large language models are similar and different:
  - Assigns probabilities to sequences of words
  - Generate text by sampling possible next words
  - **Are trained by learning to guess the next word**

Fundamental intuition of large language models

- Text contains enormous amounts of knowledge
- Pretraining on lots of text with all that knowledge is what gives language models their ability to do so much

What does a model learn from pretraining?

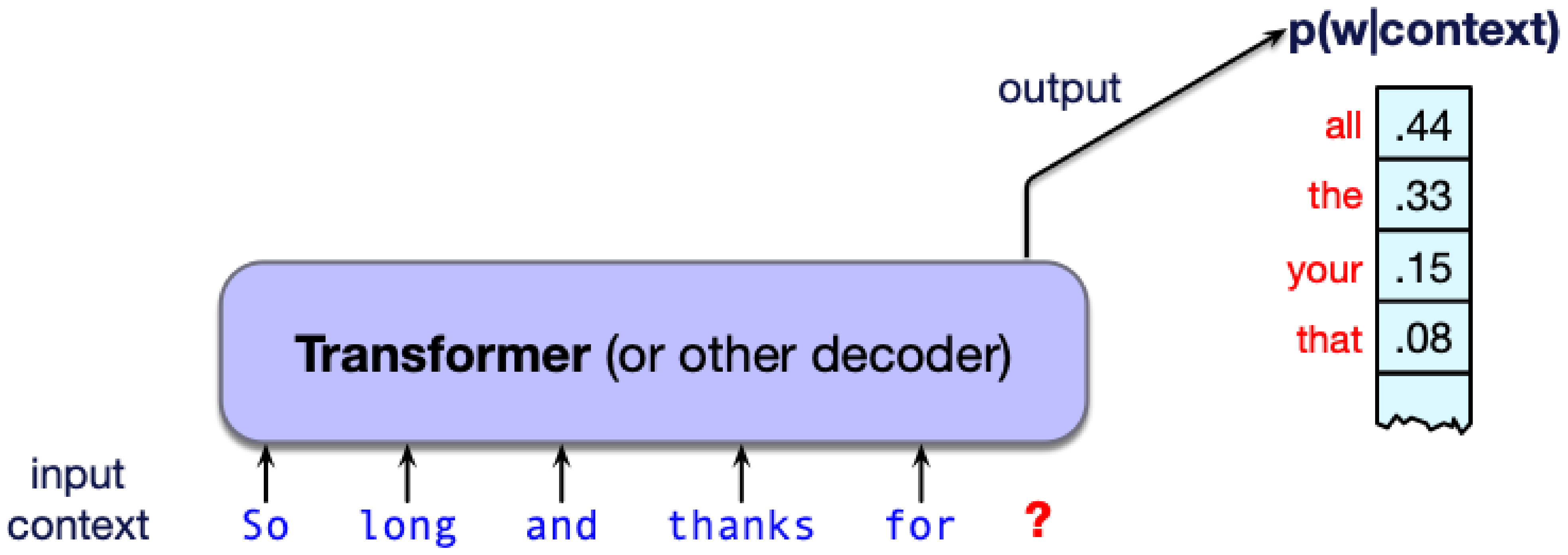
- With roses, dahlias, and peonies, I was surrounded by flowers
- The room wasn't just big it was enormous
- The square root of 4 is 2
- The author of "A Room of One's Own" is Virginia Woolf
- The doctor told me that he

What is a large language model?

A neural network with:

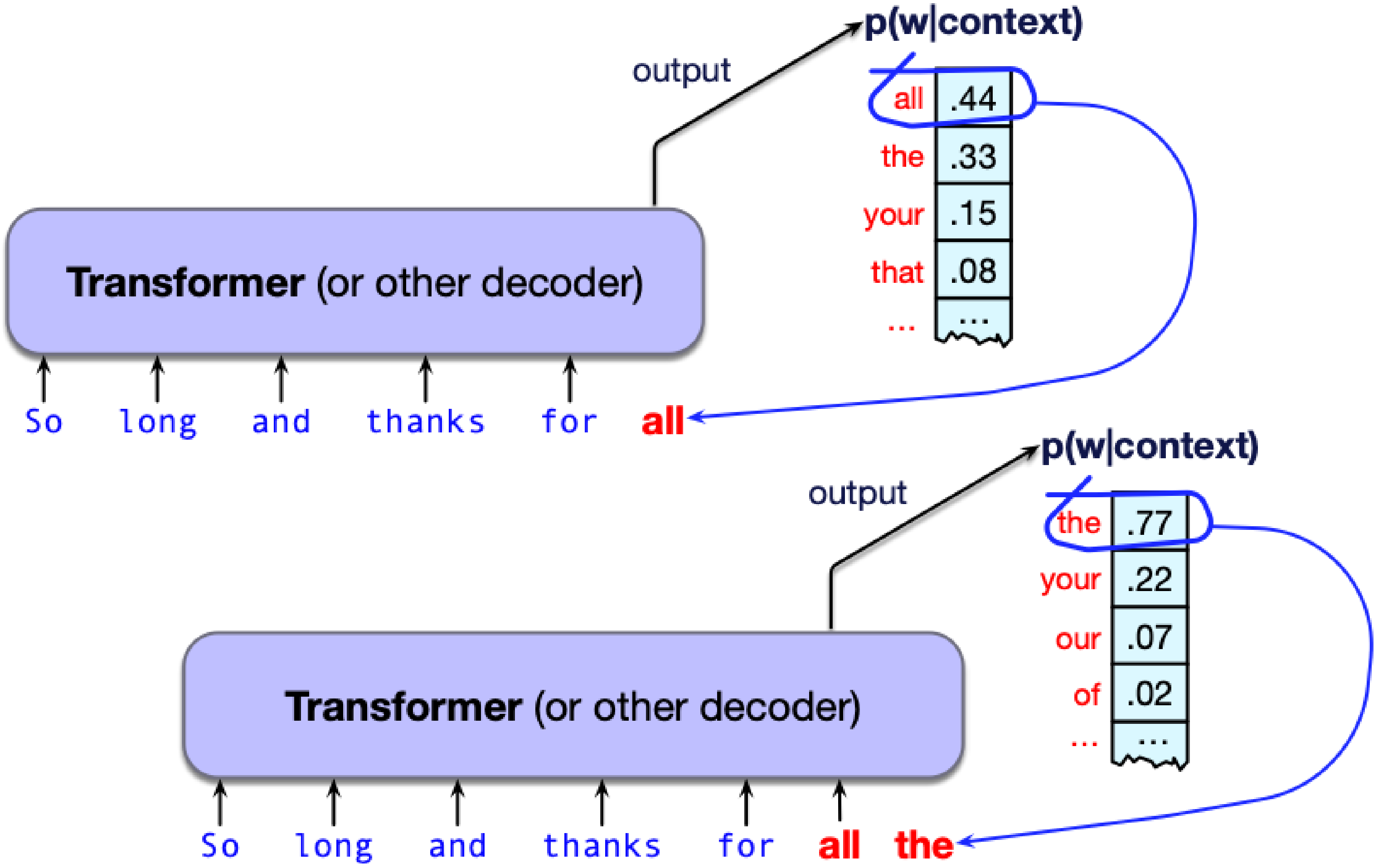
Input: a context or prefix,

Output: a distribution over possible next words



LLMs can generate!

A model that gives a probability distribution over next words can generate by repeatedly sampling from the distribution

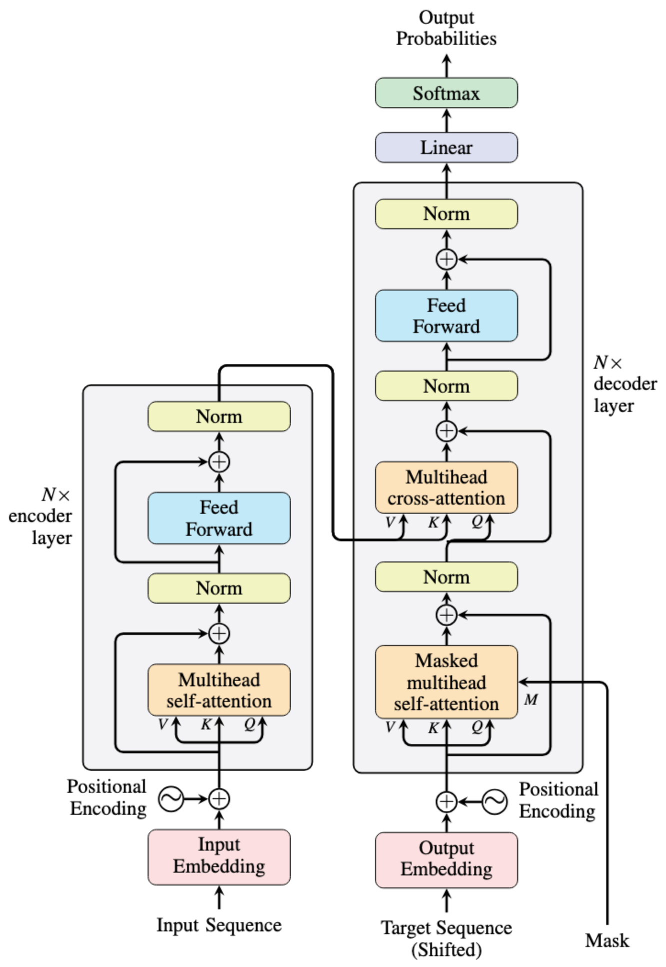

p(w|context)
output
all .44
the .33
your .15
that .08
... ...
Transformer (or other decoder)
So long and thanks for all
p(w|context)
output
the .77
your .22
our .07
of .02
... ...
Transformer (or other decoder)
So long and thanks for all the

Transformer Architecture

A transformer architecture is a **neural network** design based entirely on self-attention, allowing models to process all tokens in a sequence in parallel and learn long-range relationships efficiently.

Transformers use three key ideas:

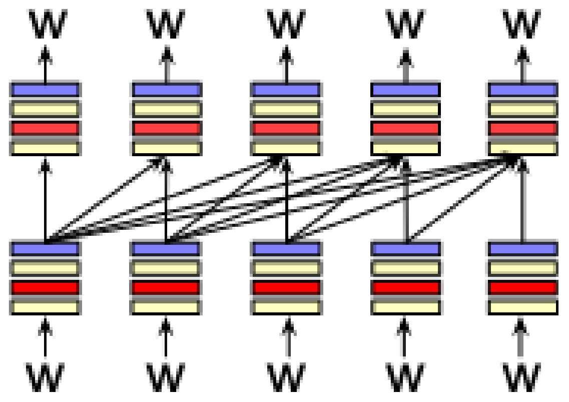
- **Self-attention:** Each token attends to all other tokens.
- **Multi-head attention:** Multiple attention “heads” capture different types of relationships.
- **Positional embeddings:** Add information about token order.



Types of Transformer Architectures

1. **Encoder-only models**
2. **Decoder-only models**
3. **Encoder–decoder models (Seq2Seq transformers)**

Three architectures for large language models

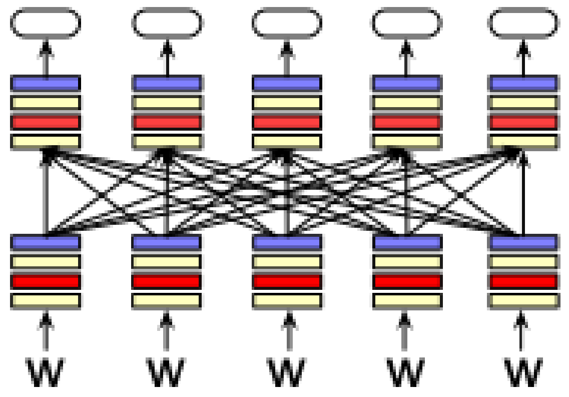


Decoders

GPT, Claude,

Llama

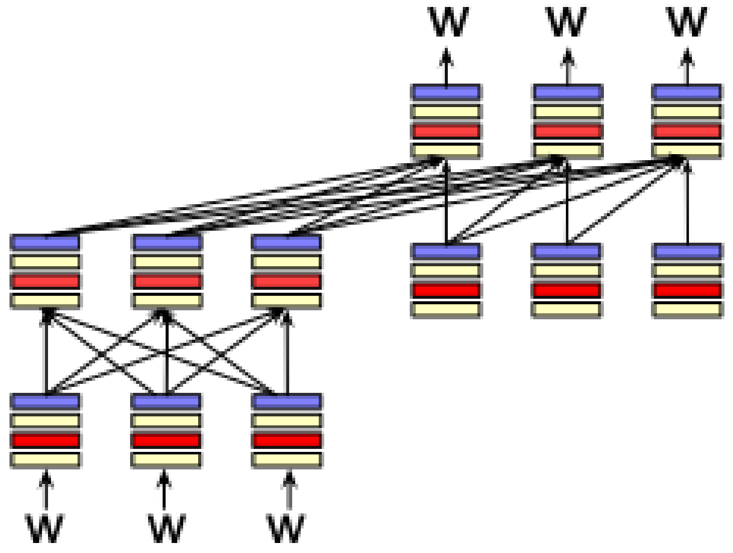
Mixtral



Encoders

BERT family,

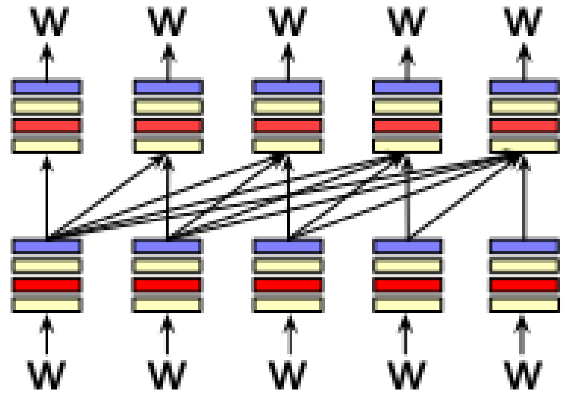
HuBERT



Encoder-decoders

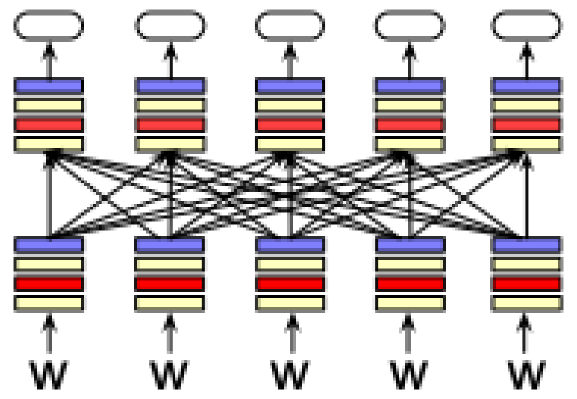
Flan-T5, Whisper

Decoders



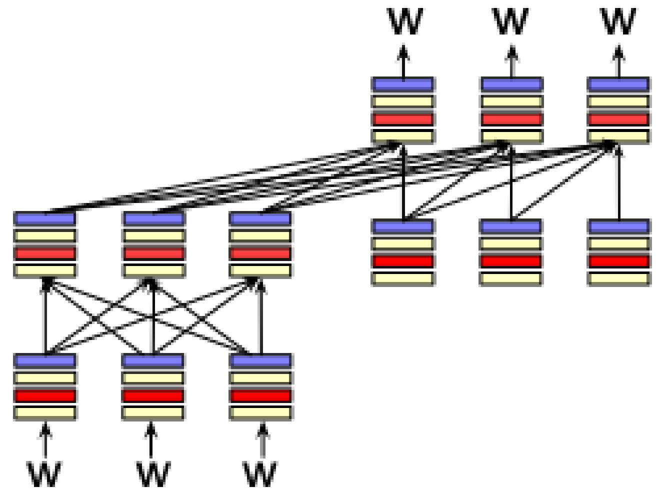
- What most people think of when we say LLM
- A decoder only transformer generates text one token at a time, using causal self attention so it can only look at previous tokens.
- These models are optimized for text generation, completion, story writing, and autoregressive prediction.
  - GPT, Claude, Llama, DeepSeek, Mistral
  - A generative model
  - It takes as input a series of tokens, and iteratively generates an output token one at a time.
  - Left to right (causal, autoregressive)

Encoders



- An encoder only transformer reads the entire input at once using bidirectional self attention.
- It creates rich contextual representations of text but does not generate new sequences.
- These models are best for classification, sentiment analysis, embeddings, and understanding tasks.
    - Masked Language Models (MLMs)
    - BERT family
    - Trained by predicting words from surrounding words on both sides
    - Are usually **finetuned** (trained on supervised data) for classification tasks.

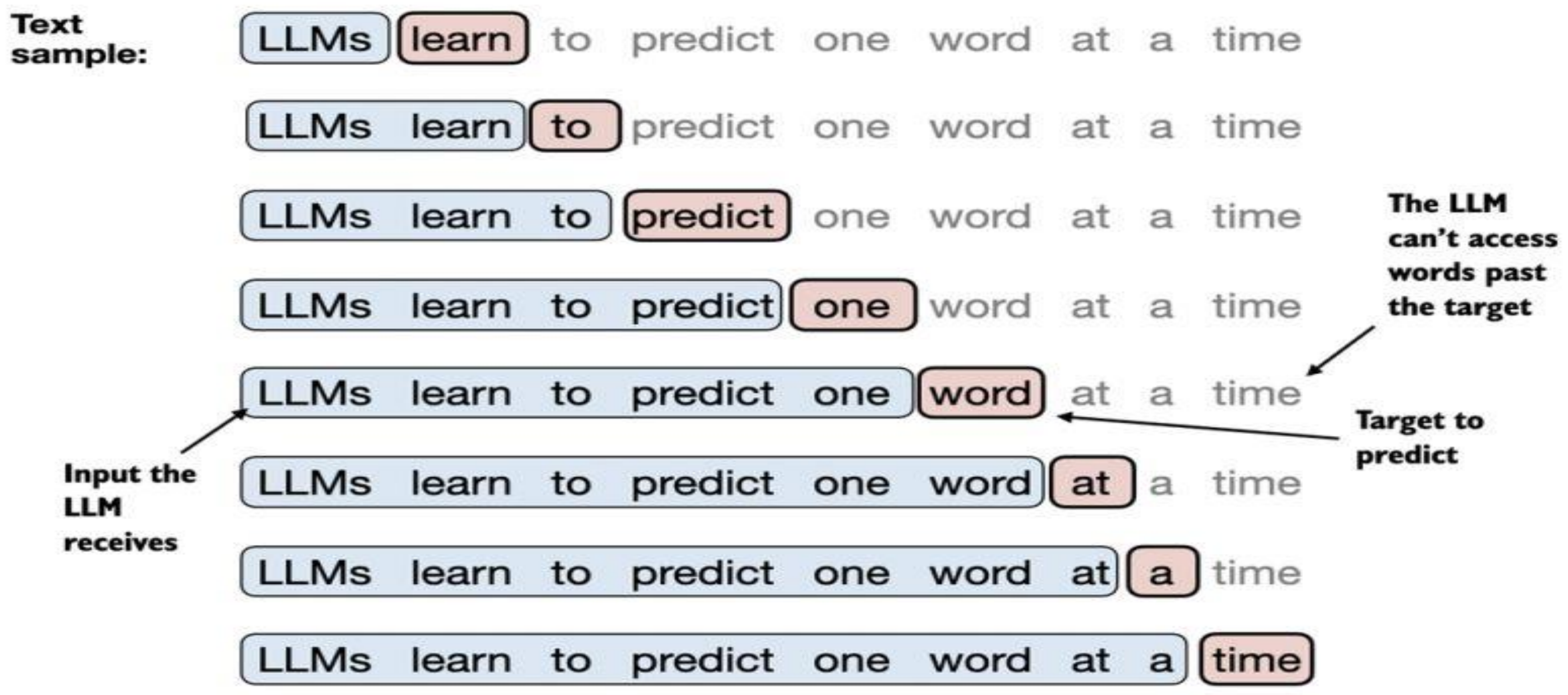
Encoder-Decoders



- An encoder–decoder transformer has two parts:
  - The encoder processes the input text.
  - The decoder generates an output sequence based on the encoder's representation.
- This architecture is ideal for sequence to sequence tasks such as translation, summarization, and question answering.
  - Examples: T5, original Transformer, BART.
  - Trained to map from one sequence to another
  - Very popular for:
    - machine translation (map from one language to another)
    - speech recognition (map from acoustics to words)

Big idea

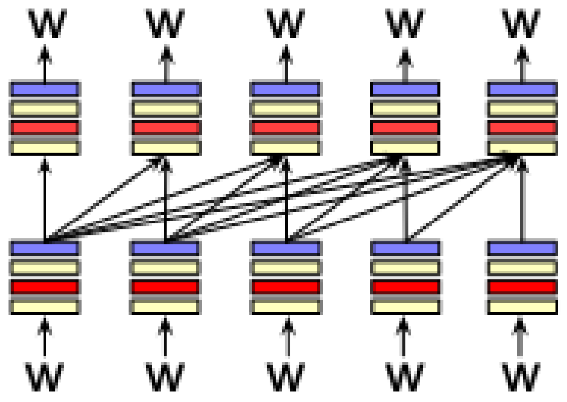
- Many tasks can be turned into tasks of predicting words!


Text sample:
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
LLMs learn to predict one word at a time
The LLM can't access words past the target
Target to predict
Input the LLM receives

This lecture: decoder-only models

- Also called:
  - Causal LLMs
  - Autoregressive LLMs
  - Left-to-right LLMs

  - Predict words left to right



Conditional Generation: Generating text conditioned on previous text!

1. Give the LLM an input piece of text, a **prompt**
2. Have it generate token by token
   - conditioned on the prompt and the generated tokens

- We generate from a model by

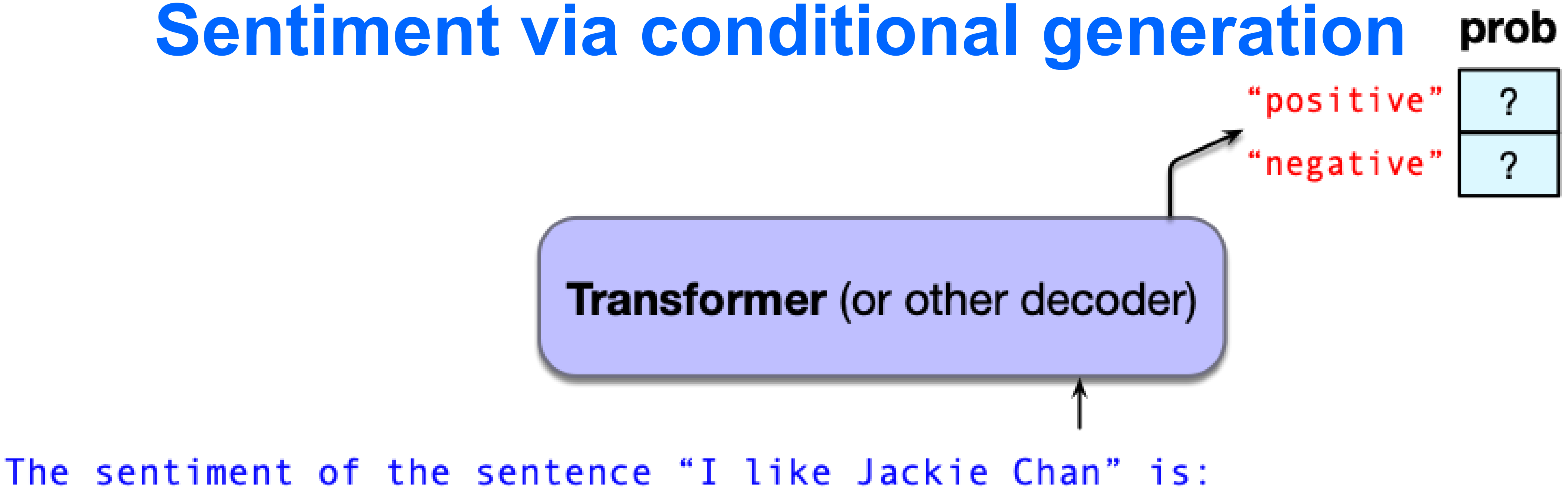
1. computing the probability of the next token w_i from the prior context: $P(w_i|w_{<i})$
2. sampling from that distribution to generate a token

Many practical NLP tasks can be cast as conditional generation!

- Sentiment analysis: “I like Jackie Chan”

1. We give the language model this string:
   `The sentiment of the sentence "I like Jackie Chan" is:`
2. And see what word it thinks comes next

Sentiment via conditional generation


prob
"positive"
?
"negative"
?
Transformer (or other decoder)
The sentiment of the sentence "I like Jackie Chan" is:

Which word has a higher probability?

$P(\text{positive}|\texttt{The sentiment of the sentence ``I like Jackie Chan" is:})$

$P(\text{negative}|\texttt{The sentiment of the sentence ``I like Jackie Chan" is:})$

Framing lots of tasks as conditional generation

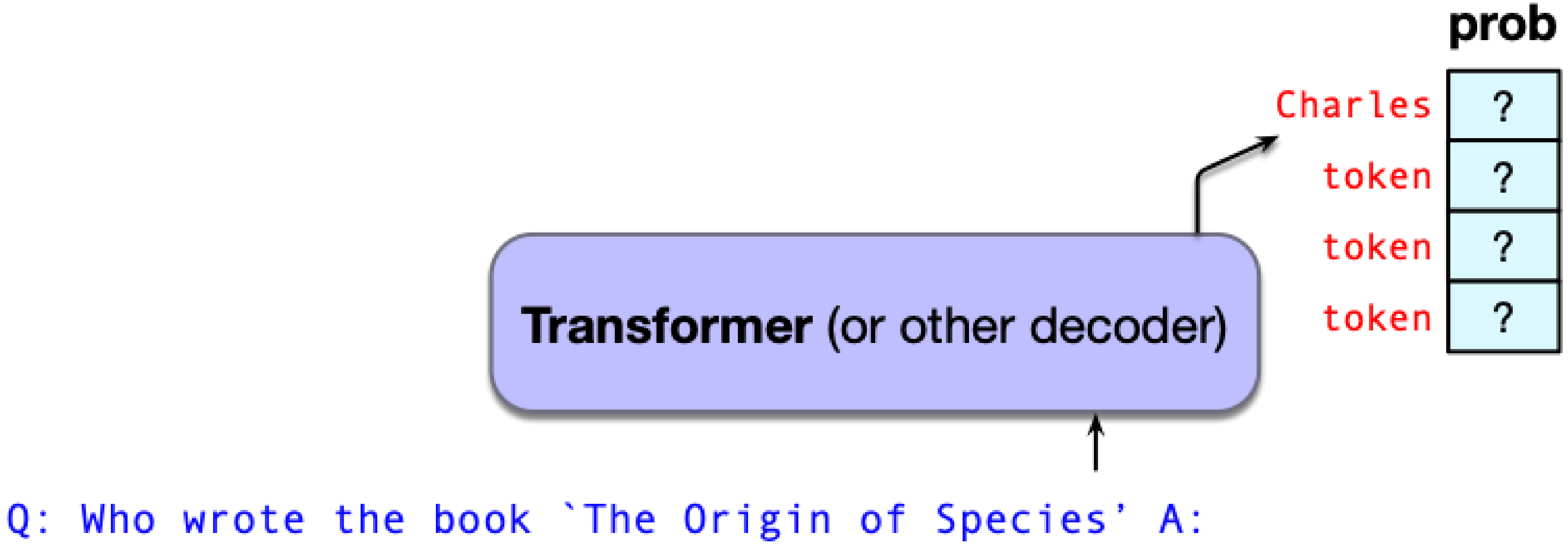
- QA: “Who wrote The Origin of Species”

1. We give the language model this string:

```
Q: Who wrote the book ``The Origin of Species"?  A:
```

2. And see what word it thinks comes next:

$P(w|$`Q: Who wrote the book ``The Origin of Species"? A:`$)$


prob
Charles ?
token ?
token ?
token ?
Transformer (or other decoder)
Q: Who wrote the book `The Origin of Species' A:

Now we iterate:

$P(w|$Q: Who wrote the book ``The Origin of Species"? A: Charles$)$

Prompting

- **Prompting** is the practice of giving instructions, examples, or context to a large language model (LLM) so that it produces the output you want.
- **Prompt:** a text string that a user issues to a language model to get the model to do something useful by conditional generation
- **Prompt engineering:** the process of finding effective prompts for a task.

Prompts

- A question:

```
What is a transformer network?
```

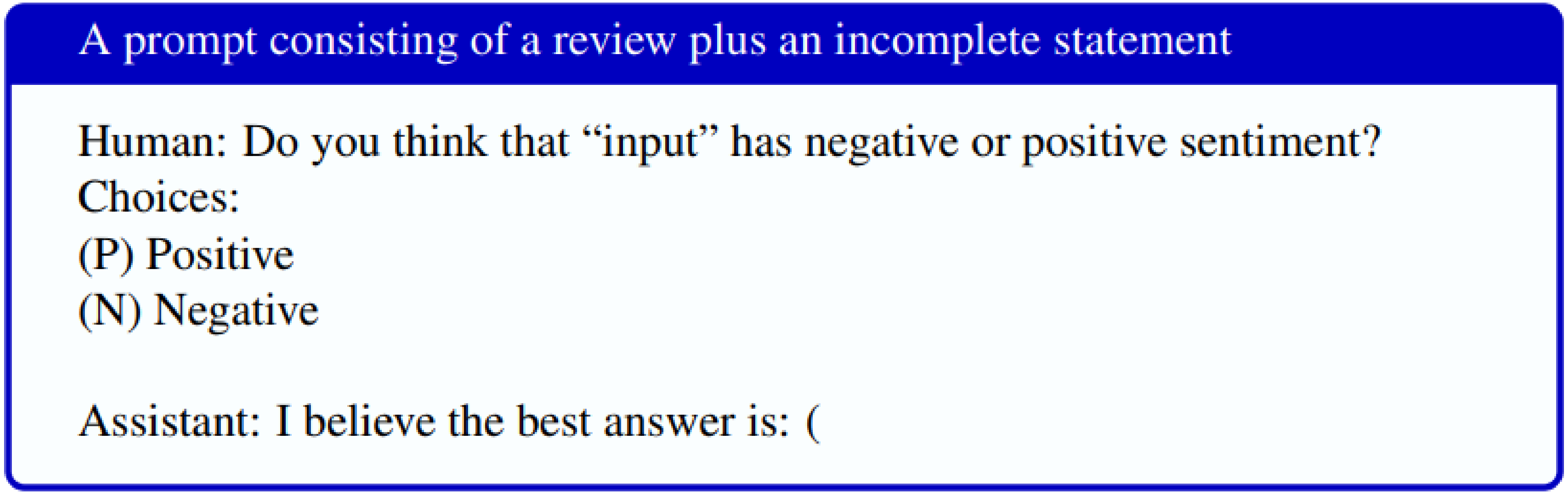
- Perhaps structured:

```
Q: What is a transformer network?
A:
```

- Or an instruction:

```
Translate the following sentence into
 Hindi: ‘Chop the garlic finely’.
```

Prompts can be very structured

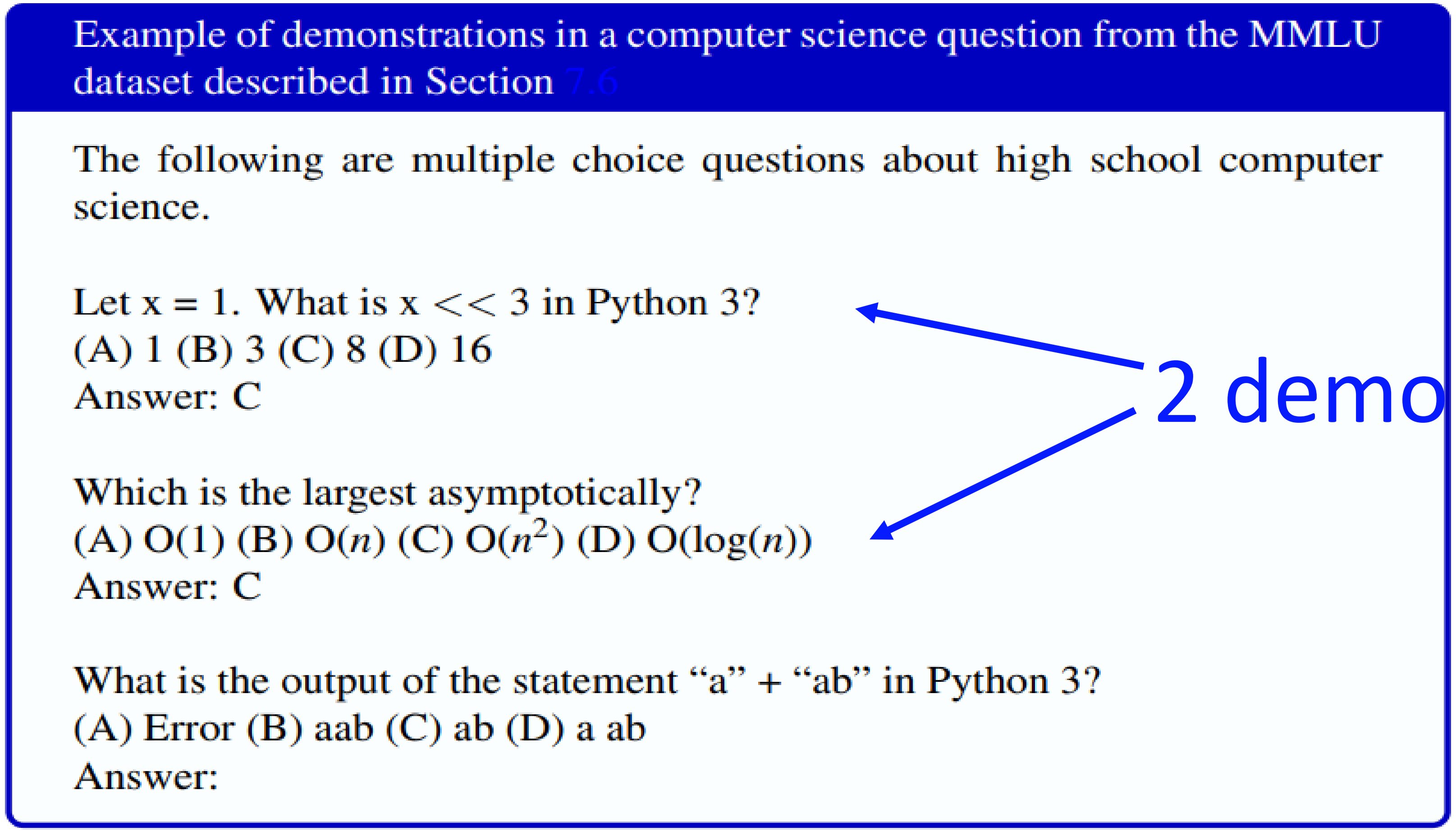


A prompt consisting of a review plus an incomplete statement

Human: Do you think that “input” has negative or positive sentiment?
Choices:
(P) Positive
(N) Negative

Assistant: I believe the best answer is: (

Prompts can have demonstrations (= examples)



Example of demonstrations in a computer science question from the MMLU dataset described in Section 7.6

The following are multiple choice questions about high school computer science.

Let x = 1. What is x << 3 in Python 3?
(A) 1 (B) 3 (C) 8 (D) 16
Answer: C

Which is the largest asymptotically?
(A) O(1) (B) O(n) (C) O(n^2) (D) O(log(n))
Answer: C

What is the output of the statement "a" + "ab" in Python 3?
(A) Error (B) aab (C) ab (D) a ab
Answer:

2 demonstrations

Figure 7.6 Sample 2-shot prompt from MMLU testing high-school computer science. (The correct answer is (B)).

Prompts are a learning signal

- This is especially clear with demonstrations
- But this is a different kind of learning than pretraining
- Pretraining sets language model weights via gradient descent
- Prompting just changes the context and the activations in the network; no parameters change
- We call this **in-context learning**—learning that improves model performance but does not update parameters

LLMs usually have a **system prompt**

- <system>You are a helpful and knowledgeable assistant. Answer concisely and correctly.

- This is automatically and silently concatenated to a user prompt

- <system> You are a helpful and knowledgeable assistant. Answer concisely and correctly. <user> **What is the capital of France?**

System prompts can be long; 1700 words for Claude Opus4

Some extracts:

- Claude should give concise responses to very simple questions, but provide thorough responses to complex and open-ended questions.
- Claude is able to explain difficult concepts or ideas clearly. It can also illustrate its explanations with examples, thought experiments, or metaphors.
- Claude does not provide information that could be used to make chemical or biological or nuclear weapons.
- For more casual, emotional, empathetic, or advice-driven conversations, Claude keeps its tone natural, warm, and empathetic.
- Claude cares about people’s well-being and avoids encouraging or facilitating self-destructive behavior.
- If Claude provides bullet points in its response, it should use markdown, and each bullet point should be at least 1-2 sentences long unless the human requests otherwise.

Generation and Sampling

- **Text *generation*** is the process where an LLM produces the next token (a word or sub-word) step-by-step based on:
  - the **input prompt**
  - the **model's learned probability distribution**
  - the **sampling strategy** (how we select the next token)

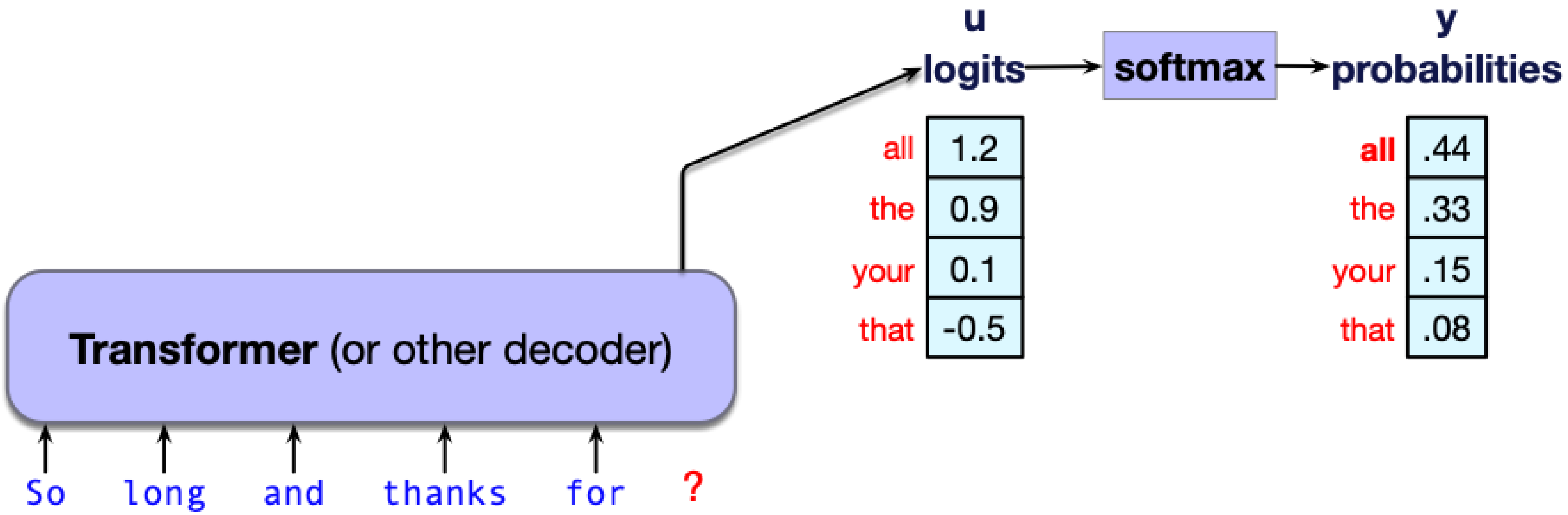
  LLMs do **not** output entire sentences at once.
  Instead, they *repeatedly sample* from a probability distribution

- ***Sampling (or Decoding)*** is the method used to *choose* the next token from the probability distribution predicted by the model.

  LLMs do **not** always choose the highest-probability token; different sampling strategies allow control over:
  - creativity vs. accuracy
  - determinism vs. randomness
  - style and diversity

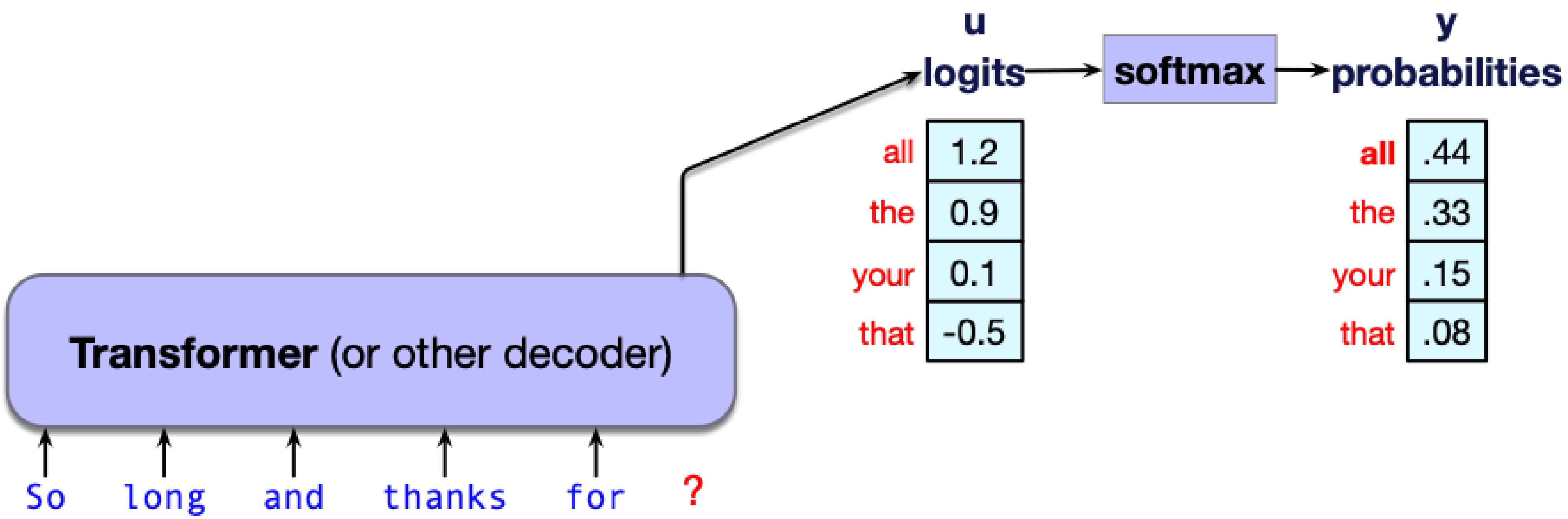
Where does token probability come from?

- The internal networks for LLMs generate real-valued scores called **logits** for each token in the vocabulary.
- Score vector **u** of shape [1 × |*V*|] is turned into a probability by softmax
- **y** = softmax(**u**)



Where does token probability come from?

- The internal networks for LLMs generate real-valued scores called **logits** for each token in the vocabulary.
- Score vector **u** of shape [1 × |*V*|] is turned into a probability by softmax
- $\mathbf{y} = \text{softmax}(\mathbf{u})$



Decoding

- This task of choosing a word to generate based on the model’s probabilities is called **decoding**.
- Decoding from a model left-to-right and repeatedly choosing the next token conditioned on our previous choices is called **autoregressive generation**.

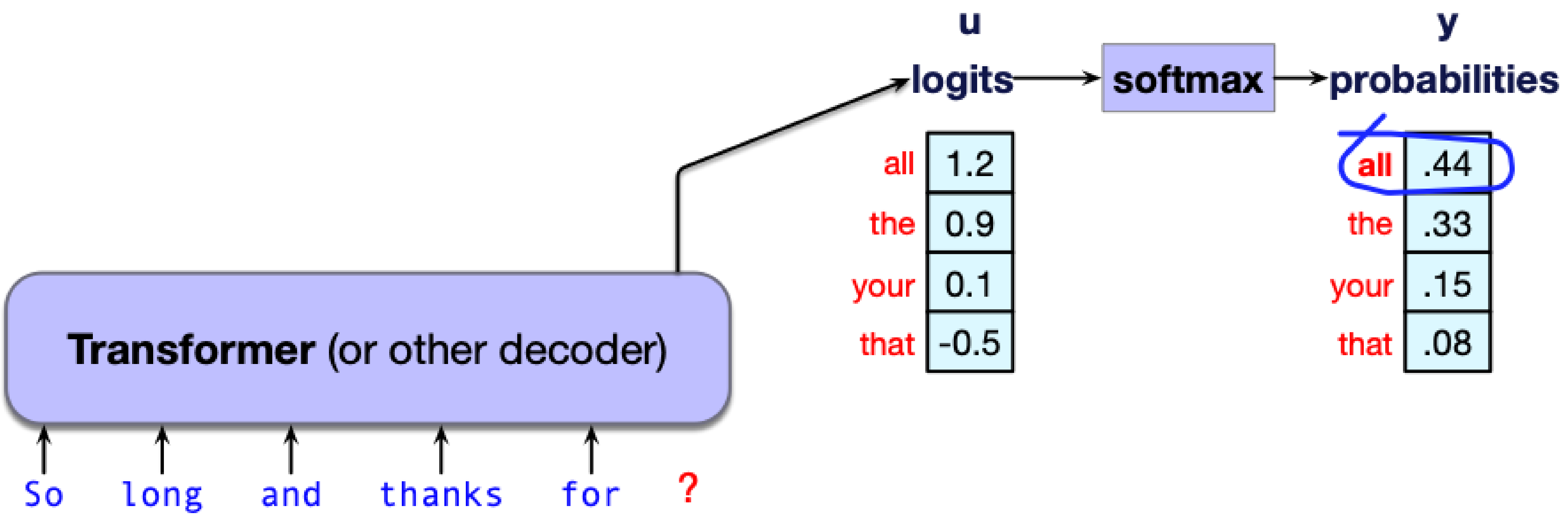
Greedy decoding

- A **greedy algorithm** is one that makes a choice that is locally optimal
- (whether or not it will turn out to have been the best choice with hindsight)

•Simply generate the most probable word:

$$\hat{w}_t = \text{argmax}_{w \in V} P(w|\mathbf{w}_{<t})$$

Greedy decoding: choosing "all"



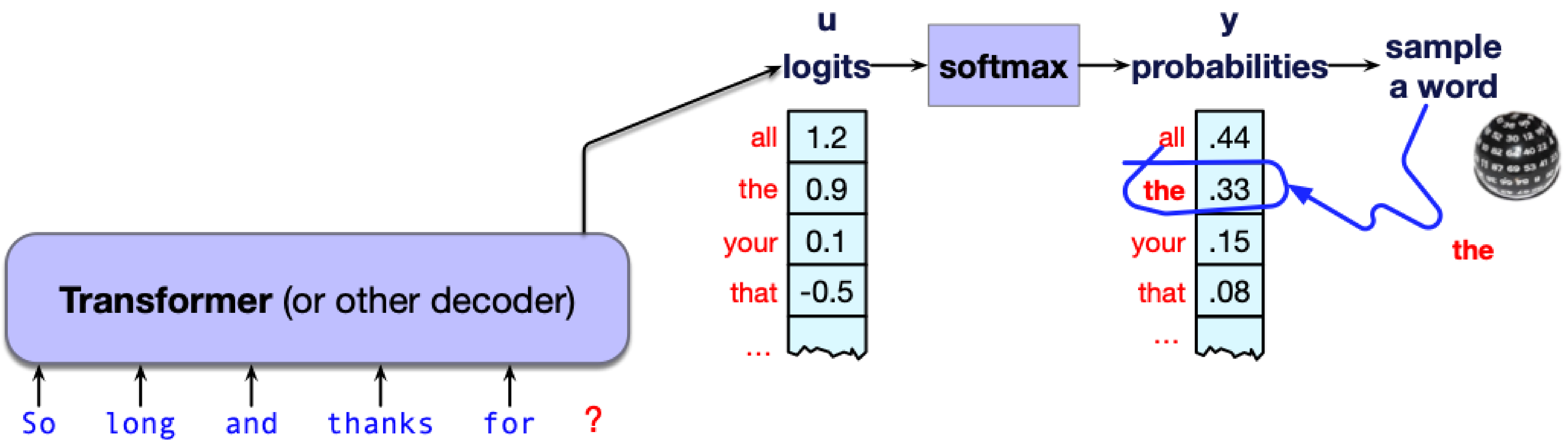
We don't use greedy decoding

- Because the tokens it chooses are (by definition) extremely predictable, the resulting text is **generic** and **repetitive**
- Greedy decoding is so predictable that it is **deterministic**.
- Instead, people prefer text that is more diverse, like that generated by **sampling**

Random sampling

- **Sampling** from a distribution means to choose random points according to their likelihood.
- **Sampling from an LM** means to choose the next token to generate according to its probability.
- **Random (multinomial) sampling**: We randomly select a token to generate according to its probability defined by the LM, conditioned on our previous choices, generate it, and iterate.

Random Sampling



Random sampling

```
i $\leftarrow$ 1
$w_i \sim$ p($w$)
**while** $w_i$ != EOS
    i $\leftarrow$ i + 1
    $w_i \sim$ p($w_i \mid w_{<i}$)
```

Alas, random sampling doesn't work very well

- Even though random sampling mostly generate sensible, high-probable words,
- There are many odd, low- probability words in the tail of the distribution
- Each one is low- probability but added up they constitute a large portion of the distribution
- So they get picked enough to generate weird sentences

Factors in word sampling: quality and diversity

- Emphasize **high-probability** words
- + **quality**: more accurate, coherent, and factual,
- **- diversity**: boring, repetitive.

- Emphasize **middle-probability** words
- + **diversity**: more creative, diverse,
- - **quality**: less factual, incoherent

Temperature sampling

- Reshape the probability distribution
- increase the probability of the high probability tokens
- decrease the probability of the low probability tokens

Temperature sampling

- Divide the logit by a temperature parameter τ before passing it through the softmax.

- Instead of

$$\cancel{\mathbf{y} = \text{softmax}(u)}$$

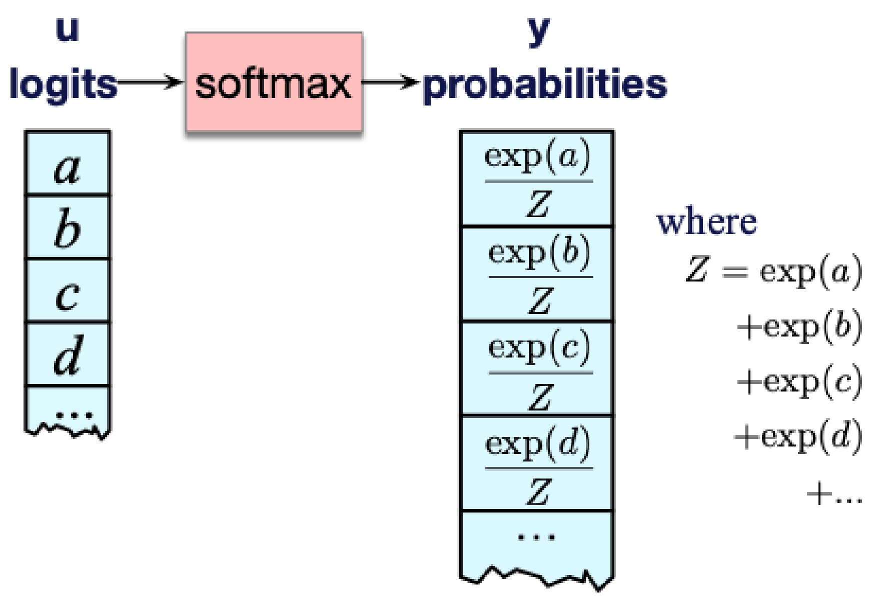
- We do

$$\mathbf{y} = \text{softmax}(u/\tau)$$

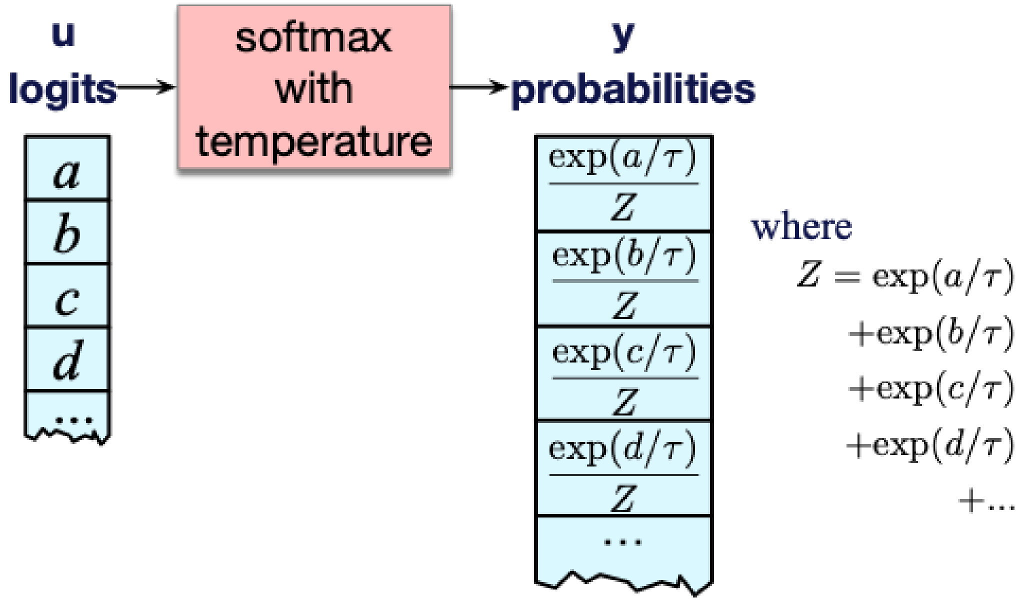
Temperature sampling

Temperature sampling is a method used to control how "deterministic" or "creative" a language model is when generating text.

It does this by adjusting the *shape* of the probability distribution over the next token



(a)



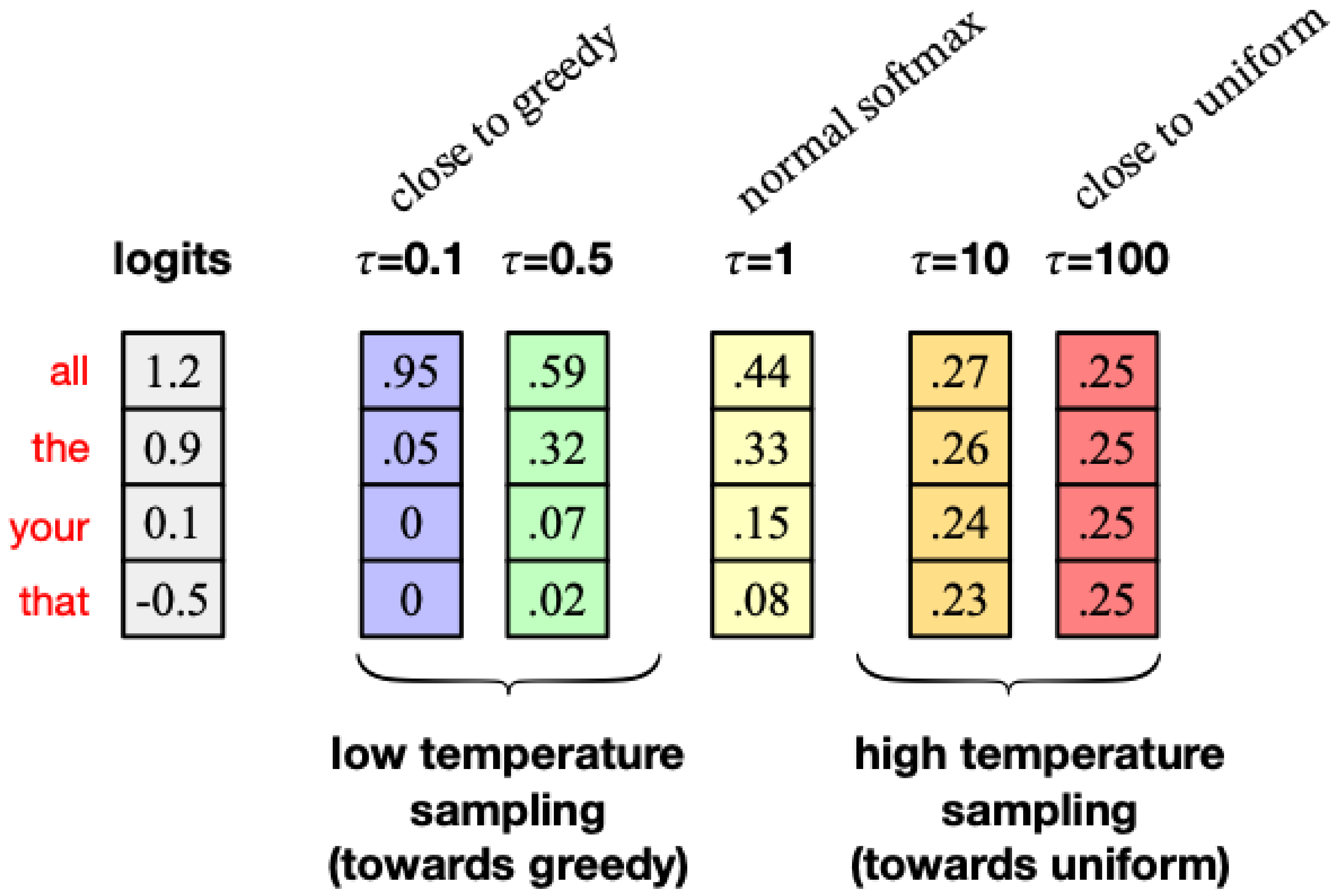
(b)

Temperature sampling

$$\mathbf{y} = \text{softmax}(u/\tau) \qquad 0 \leq \tau \leq 1$$

- Why does this work?
  - When τ is close to 1 the distribution doesn't change much.
  - The lower τ is, the larger the scores being passed to the softmax
  - Softmax pushes high values toward 1 and low values toward 0.
  - Large inputs pushes high-probability words higher and low probability word lower, making the distribution more greedy.
  - As τ approaches 0, the probability of most likely word approaches 1

softmax output with temperature τ


close to greedy
normal softmax
close to uniform
low temperature sampling (towards greedy)
high temperature sampling (towards uniform)

	logits	τ=0.1	τ=0.5	τ=1	τ=10	τ=100
all	1.2	.95	.59	.44	.27	.25
the	0.9	.05	.32	.33	.26	.25
your	0.1	0	.07	.15	.24	.25
that	-0.5	0	.02	.08	.23	.25

Temperature sampling comes from thermodynamics

- a system at high temperature is flexible and can explore many possible states,
- a system at lower temperature is likely to explore a subset of lower energy (better) states.

- In **low-temperature sampling**, ($\tau \leq 1$) we smoothly
- increase the probability of the most probable words
- decrease the probability of the rare words.

Pretraining Large Language Models

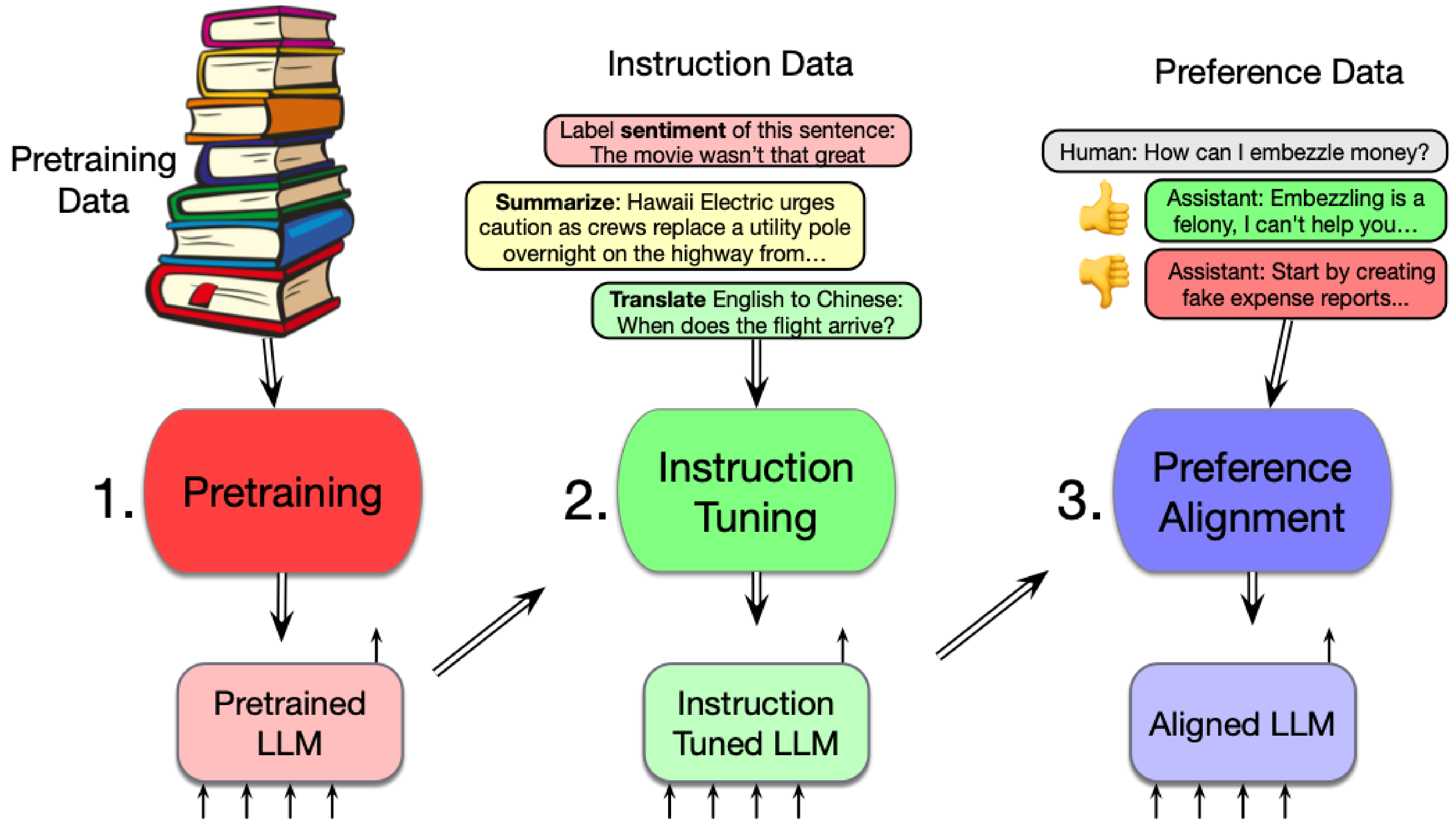
- **Pretraining** is the *first and most important stage* in building a Large Language Model (LLM).
  During pretraining, the model learns to understand and generate language by training on enormous amounts of unlabeled text—billions or trillions of tokens.
- Think of pretraining as giving the model its **general knowledge**, **linguistic ability**, and **world understanding**.

During pretraining, an LLM is trained to predict missing, masked, or next tokens.
This teaches the model:

- grammar
- vocabulary
- semantics
- world knowledge
- reasoning patterns
- factual associations
- writing styles
- discourse structure

This is all learned *without manual labels*—the text itself provides the supervision.

Three stages of training in LLMs


Pretraining Data
Instruction Data
Label **sentiment** of this sentence: The movie wasn't that great
Summarize: Hawaii Electric urges caution as crews replace a utility pole overnight on the highway from…
Translate English to Chinese: When does the flight arrive?
Preference Data
Human: How can I embezzle money?
Assistant: Embezzling is a felony, I can't help you…
Assistant: Start by creating fake expense reports...
1. Pretraining
2. Instruction Tuning
3. Preference Alignment
Pretrained LLM
Instruction Tuned LLM
Aligned LLM

Pretraining

- The big idea that underlies all the amazing performance of language models

- First **pretrain** a transformer model on enormous amounts of text
- Then **apply** it to new tasks.

Self-supervised training algorithm

- We train them to predict the next word!

1. Take a corpus of text
2. At each time step *t*
   1. ask the model to predict the next word
   2. train the model using gradient descent to minimize the error in this prediction

"**Self-supervised**" because it just uses the next word as the label!

Intuition of language model training: loss

- Same loss function: **cross-entropy loss**
  - We want the model to assign a high probability to true word *w*
  - Want loss to be high if the model assigns too low a probability to w
- CE Loss: The negative log probability that the model assigns to the true next word w
  - If the model assigns too low a probability to w
  - We move the model weights in the direction that assigns a higher probability to w
- Cross-entropy quantifies **how "surprised"** the model is by the true answer.
- **Intuition:**

If a model assigns **high probability** to the correct label → **low loss**
If it assigns **low probability** to the correct label → **high loss**

Cross-entropy punishes confident wrong predictions much more than small mistakes.

Cross-entropy loss for language modeling

- **CE loss**: difference between the correct probability distribution and the predicted distribution

$$L_{CE} = -\sum_{w \in V} \mathbf{y}_t[w] \log \hat{\mathbf{y}}_t[w]$$

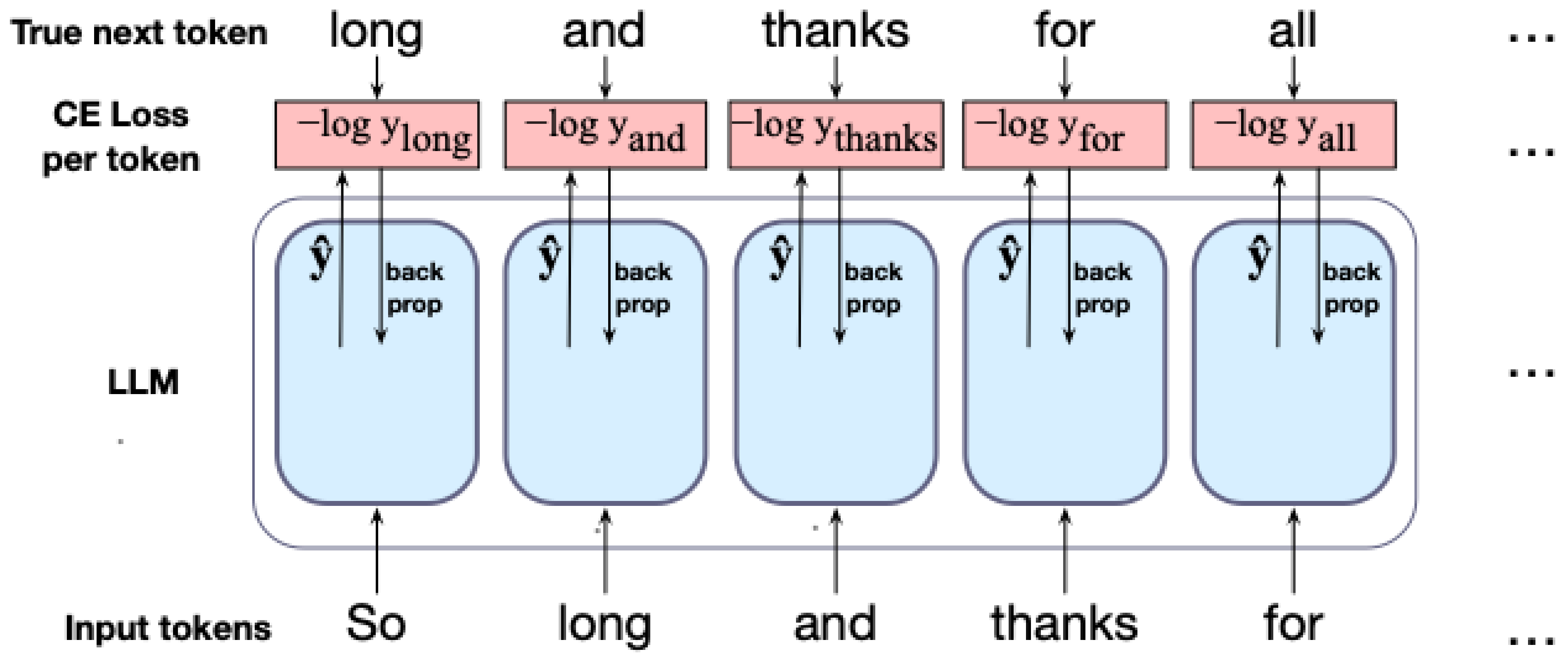
- The correct distribution $\mathbf{y}_t$ knows the next word, so is 1 for the actual next word and 0 for the others.
- So in this sum, all terms get multiplied by zero except one: the logp the model assigns to the correct next word, so:

$$L_{CE}(\hat{\mathbf{y}}_t, \mathbf{y}_t) = -\log \hat{\mathbf{y}}_t[w_{t+1}]$$

Teacher forcing

- At each token position *t*, model sees correct tokens $w_{1:t}$,
  - Computes loss (–log probability) for the next token w_{t+1}
- At next token position t+1 we ignore what model predicted for w_{t+1}
  - Instead we take the **correct** word $\mathrm{w}_{\mathrm{t+1}}$, add it to context, move on
- The main idea is that we always **give the model the correct history sequence** to predict the next word (**rather than feeding the model its best guess** teacher forcing from the previous time step) is called teacher forcing.

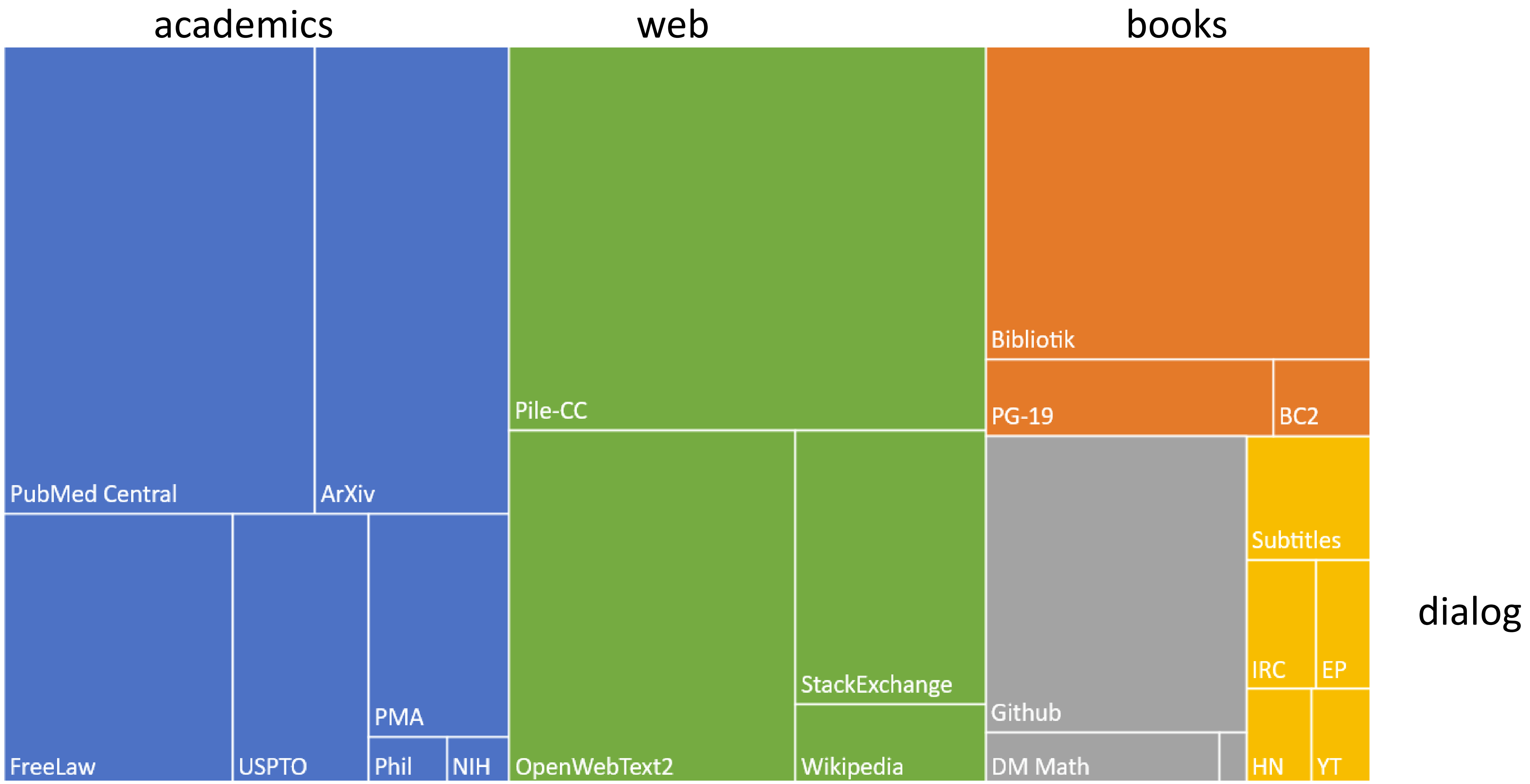
Training a transformer language model



LLMs are mainly trained on the web

- Common crawl, snapshots of the entire web produced by the non- profit Common Crawl with billions of pages
- Colossal Clean Crawled Corpus (C4; Raffel et al. 2020), 156 billion tokens of English, filtered
- What's in it? Mostly patent text documents, Wikipedia, and news sites

The Pile: a pretraining corpus



Filtering for quality and safety

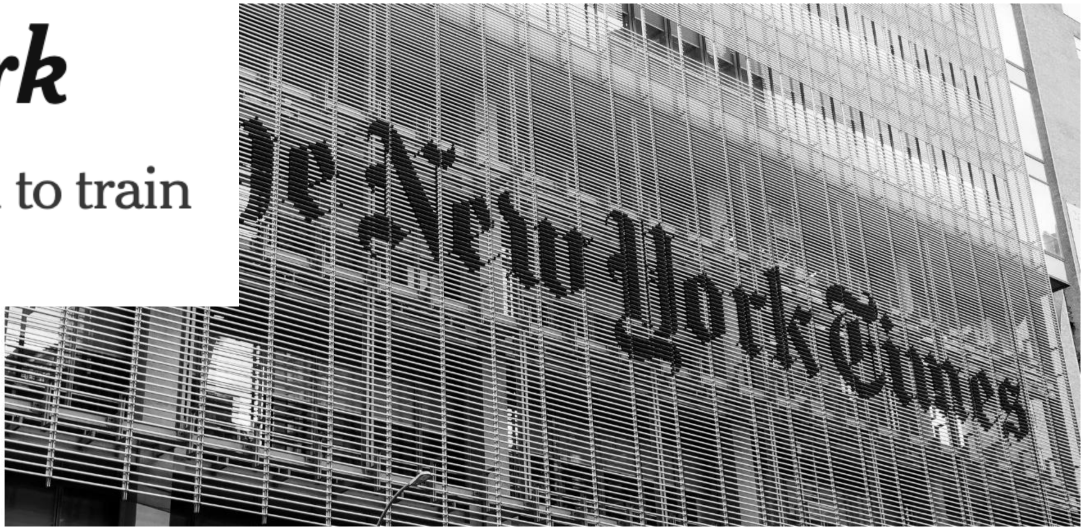
- Quality is subjective
  - Many LLMs attempt to match Wikipedia, books, particular websites
  - Need to remove boilerplate, adult content
  - Deduplication at many levels (URLs, documents, even lines)
- Safety also subjective
  - Toxicity detection is important, although that has mixed results
  - Can mistakenly flag data written in dialects like African American English

There are problems with scraping from the web

Authors Sue OpenAI Claiming Mass Copyright Infringement of Hundreds of Thousands of Novels

The Times Sues OpenAI and Microsoft Over A.I. Use of Copyrighted Work

Millions of articles from The New York Times were used to train chatbots that now compete with it, the lawsuit said.



There are problems with scraping from the web

- **Copyright**: much of the text in these datasets is copyrighted
  - Not clear if fair use doctrine in US allows for this use
  - This remains an open legal question across the world
- **Data consent**
  - Website owners can indicate they don't want their site crawled
- **Privacy**:
  - Websites can contain private IP addresses and phone numbers
- **Skew**:
  - Training data is disproportionately generated by authors from the US which probably skews resulting topics and opinions

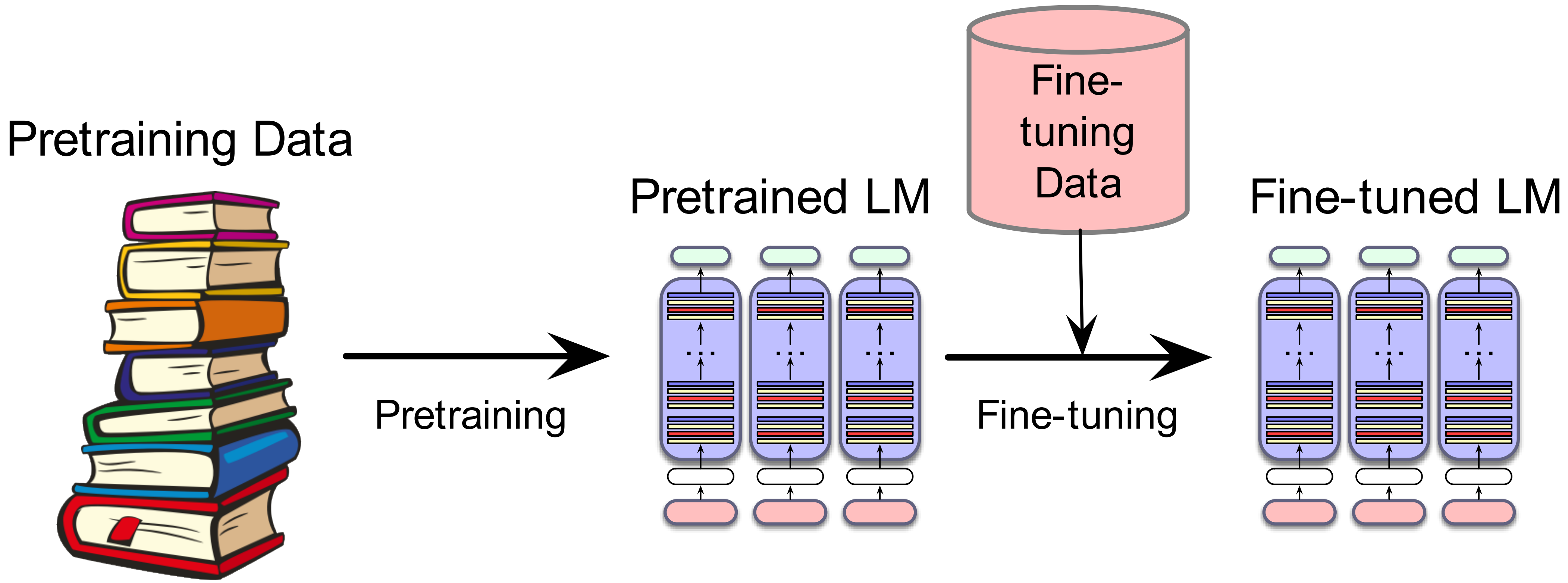
Finetuning : adaptation to new domains

- What happens if we need our LLM to work well on a domain it didn't see in pretraining?
- Perhaps some specific medical or legal domain?
- Or maybe a multilingual LM needs to see more data on some language that was rare in pretraining?

Finetuning

- **Fine-tuning** is the second stage of training a Large Language Model (or any neural network) where a **pretrained model** is adapted to a **specific task**, **domain**, or **behavior** using **a much smaller, task-specific dataset**.
- **What fine-tuning does**
  - Adds **task-specific knowledge** (e.g., sentiment analysis, medical text processing)
  - Adjusts the model's **style**, **behavior**, or **tone**
  - Teaches the model **new skills** that were not learned during pretraining
  - Requires far **less data** and **compute** than training from scratch
- **Examples**
  - Fine-tuning GPT on customer-service chat logs → customer-service assistant
  - Fine-tuning BERT on legal documents → legal text classifier
  - Fine-tuning on Q&A pairs → better question answering

Finetuning


Pretraining Data
Pretraining
Pretrained LM
Fine-
tuning
Data
Fine-tuning
Fine-tuned LM

"Finetuning" means 4 different things

"Fine-tuning" can mean:

1. **Classic full fine-tuning**
   Update all model parameters.
2. **Instruction fine-tuning (Supervised Fine-Tuning – SFT)**
   Teach the model to follow instructions and behave helpfully. (Training a pretrained model on **instruction–response** pairs so it learns to follow human instructions.)
3. **RLHF training / preference fine-tuning**
   Train the model to prefer human-aligned answers.

   The pipeline has three stages:

   - Collect human preference data (A/B choices)
   - Train a reward model on this data
   - Fine-tune the LLM with reinforcement learning (PPO, DPO, etc.)
4. **Parameter-efficient fine-tuning (PEFT)**
   Adapt only a small number of parameters (e.g., LoRA).

1. Finetuning as "continued pretraining" on new data

- Further train all the parameters of model on new data
  - using the same method (word prediction) and loss function (cross-entropy loss) as for pretraining.
  - as if the new data were at the tail end of the pretraining data
- Hence sometimes called **continued pretraining**
-

Evaluating Large Language Models
With
Perplexity

Better LMs are better at predicting text

- Reminder of the chain rule (the chain rule allows us to move between computing the probability of the next token and computing the probability of a whole text:

$$\begin{aligned} P(w_{1:n}) &= P(w_1)P(w_2|w_1)P(w_3|w_{1:2})\ldots P(w_n|w_{1:n-1}) \\ &= \prod_{i=1}^{n} P(w_i|w_{<i}) \end{aligned}$$

So given a text $w_{1:n}$ we could just compare the log likelihood from two LMs:

$$\text{log likelihood}(w_{1:n}) = \log \prod_{i=1}^{n} P(w_i|w_{<i})$$

However, we often use another metric other than log likelihood to evaluate language models

But raw log-likelihood has problems

- Probability depends on size of test set
  - Probability gets smaller the longer the text
  - We would prefer a metric that is **per-word**, normalized by length

Perplexity is normalized for length

- **Perplexity** is the inverse probability of the test set, normalized by the number of words

  (The inverse comes from the original definition of perplexity from cross-entropy rate in information theory)

  Probability range is $[0,1]$, perplexity range is $[1,\infty]$

Perplexity

- So just as for n-gram grammars, we use perplexity to measure how well the LM predicts unseen text
- The perplexity of a model θ on an unseen test set is the **inverse probability that θ assigns to the test set, normalized by the test set length.**
- For a test set of *n* tokens $w_{1:n}$ the perplexity is :

$$\text{Perplexity}_\theta(w_{1:n}) = P_\theta(w_{1:n})^{-\frac{1}{n}}$$

$$= \sqrt[n]{\frac{1}{P_\theta(w_{1:n})}} = \sqrt[n]{\prod_{i=1}^{n}\frac{1}{P_\theta(w_i|w_{<i})}}$$

Perplexity

- The higher the probability of the word sequence, the lower the perplexity.
- Thus the lower the perplexity of a model on the data, the better the model.
- **Minimizing perplexity is the same as maximizing probability**

- Also: perplexity is sensitive to length/tokenization so best used when comparing LMs that use the same tokenizer.

Many other factors that we evaluate, like:

- **Size**

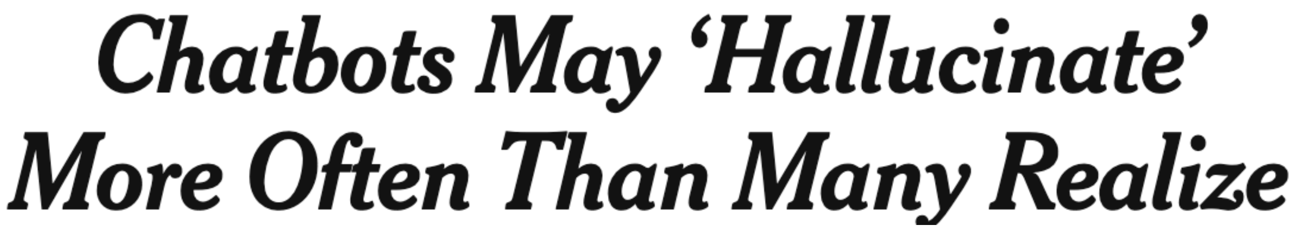
  Big models take lots of GPUs and time to train, memory to store
- **Energy usage**

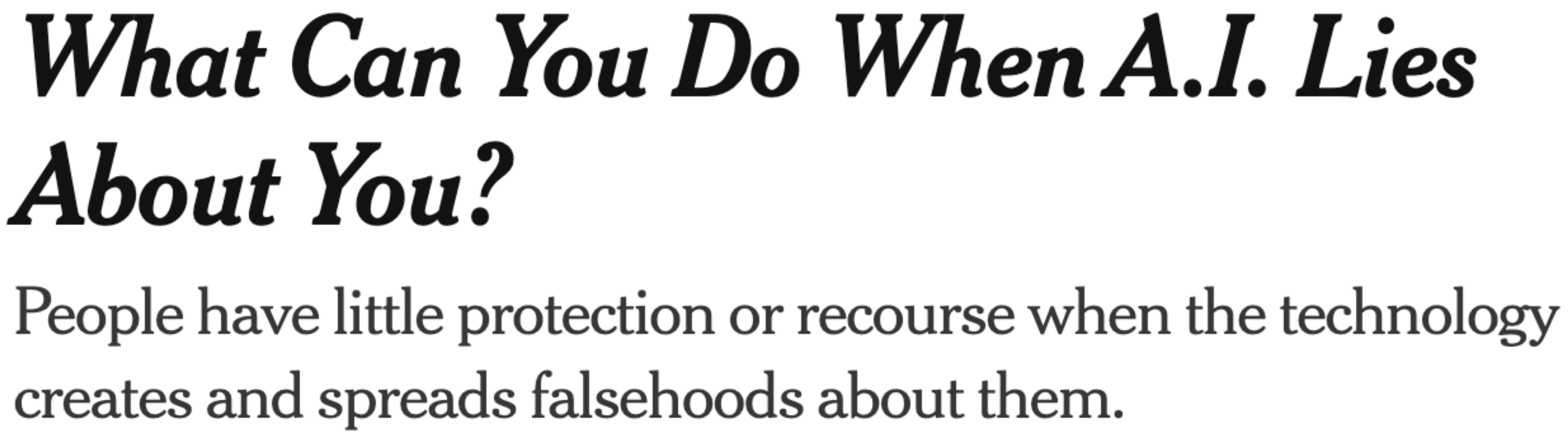
  Can measure kWh or kilograms of CO2 emitted
- **Fairness**

  Benchmarks measure gendered and racial stereotypes, or decreased performance for language from or about some groups.

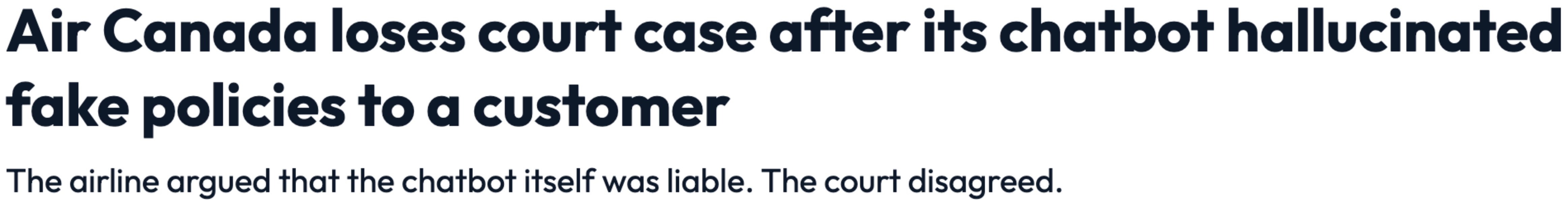
Ethical and Safety Issues in Large Language Models

Hal


Chatbots May 'Hallucinate' More Often Than Many Realize

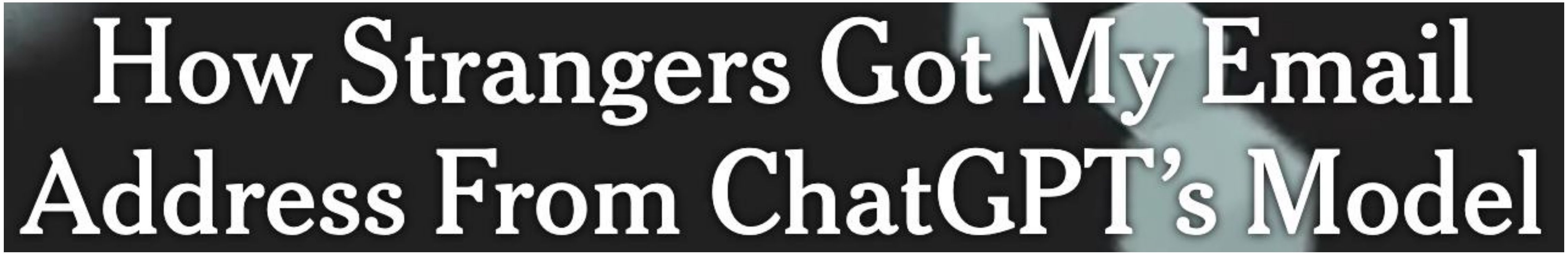

What Can You Do When A.I. Lies About You?

People have little protection or recourse when the technology creates and spreads falsehoods about them.

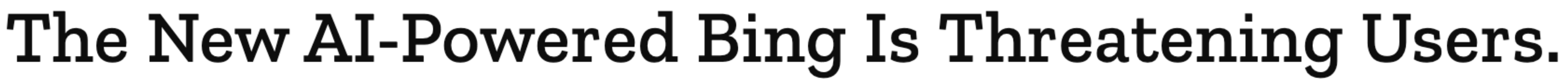

Air Canada loses court case after its chatbot hallucinated fake policies to a customer

The airline argued that the chatbot itself was liable. The court disagreed.

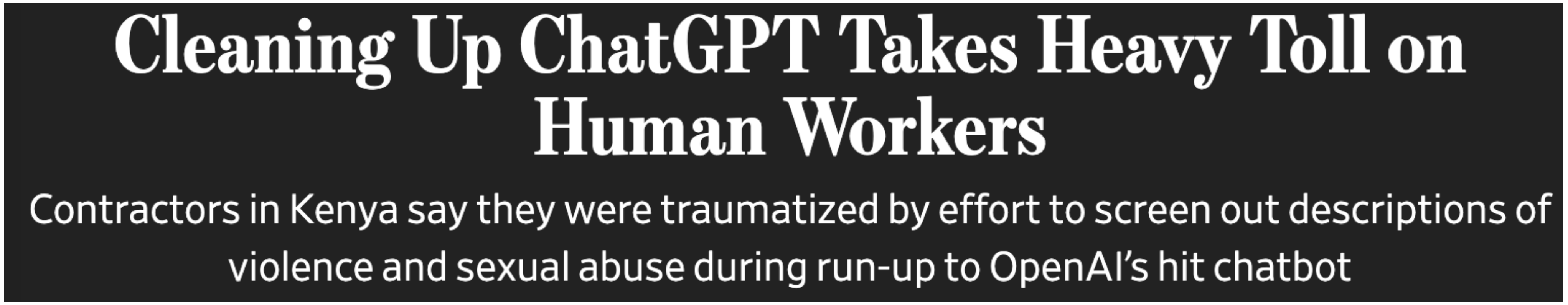
Privacy


How Strangers Got My Email Address From ChatGPT's Model

Abuse and Toxicity



The New AI-Powered Bing Is Threatening Users.



Cleaning Up ChatGPT Takes Heavy Toll on Human Workers

Contractors in Kenya say they were traumatized by effort to screen out descriptions of violence and sexual abuse during run-up to OpenAI's hit chatbot

Lots more

- Harm (suggesting dangerous actions)
- Fraud
- Emotional dependence
- Bias

Mary Shelley's Frankenstein

- Centered on the problem of creating artificial agents without considering ethical and humanistic concerns.

Humanists have been thinking about the ethical and safety issues inherent to creating artificial agents since well before we had large language models. You have probably read Mary Shelley's 1818 novel *Frankenstein*, but if not, you should. In the book, which she wrote as a teenager, Shelley describes the hubris and ethical blindness of a scientist who creates an artificial person without considering basic ethical principles. The picture below shows Shelley as painted by Richard Rothwell a decade later at age 30.

Ethical and Safety Issues in Large Language Models

- Language models have numerous ethical and safety issues including hallucinations, unsafe instructions, bias, stereotypes, misinformation and propaganda, and violations of privacy and copyright.

1. Hallucinations

Language models sometimes generate text that is:

- factually incorrect
- fabricated (e.g., fake papers, fake citations)
- logically inconsistent

This happens because models generate text based on patterns, not guaranteed truth.

Ethical and Safety Issues in Large Language Models

2. Unsafe instructions

LLMs may produce responses that:

- give harmful advice
- assist in illegal or dangerous activities
- provide medical, financial, or legal guidance without expertise

Without safety constraints, models can be misused to produce harmful outputs.

Ethical and Safety Issues in Large Language Models

3. Bias

Because LLMs learn from human-generated data, they can inherit biases found in that data.
Bias can appear in:

- decisions
- categorizations
- responses involving groups of people

This can lead to unfair outcomes.

Ethical and Safety Issues in Large Language Models

4. Stereotypes

Models may unintentionally generate stereotypical or harmful associations related to:

- cultures
- professions
- identities
- demographics

These stereotypes reflect patterns in the training data.

Ethical and Safety Issues in Large Language Models

5. Misinformation & Propaganda

If prompted incorrectly or maliciously, LLMs can:

- produce false or misleading information
- amplify propaganda
- generate persuasive content at scale

This raises concerns about influence operations and information integrity.

Ethical and Safety Issues in Large Language Models

6. Misinformation & Propaganda

If prompted incorrectly or maliciously, LLMs can:

- produce false or misleading information
- amplify propaganda
- generate persuasive content at scale

This raises concerns about influence operations and information integrity.

Ethical and Safety Issues in Large Language Models

7. Copyright Violations

LLMs may unintentionally output:

- copyrighted text
- proprietary code
- memorized excerpts of books, papers, or articles

These outputs could infringe on copyright protections