



Reviewing the Foundations

CSCI 601-771 (NLP: Advances in Self-Supervised Models)

Language Modeling: Definitions and History

The

The cat

The cat sat

The cat sat on

The cat sat on __?__

The cat sat on the mat.

$P(\text{mat} \mid \text{The cat sat on the})$

next word

context or prefix

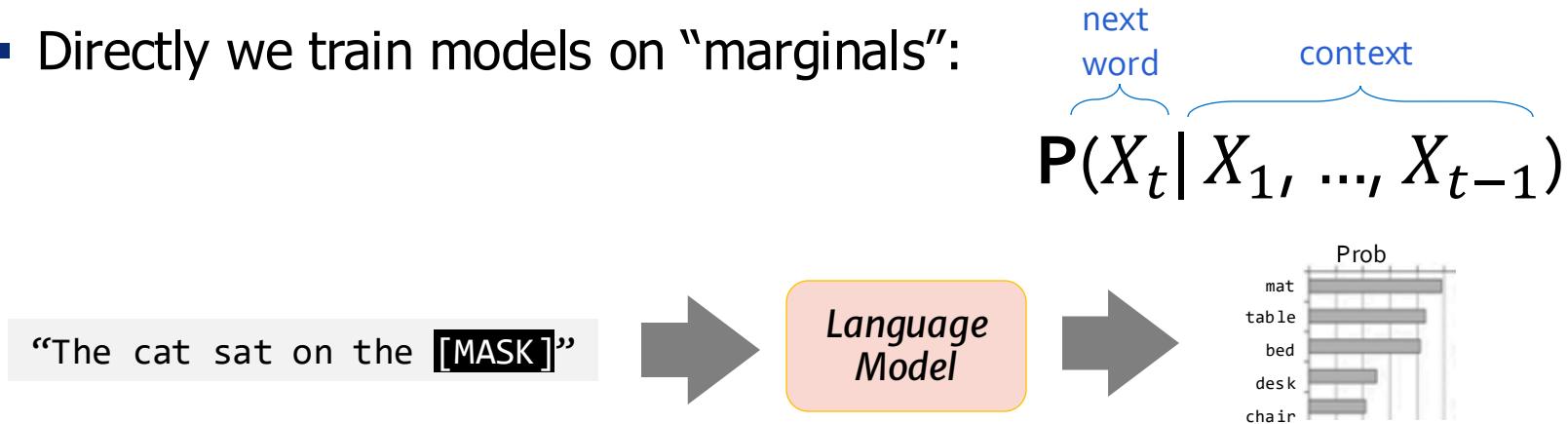
Probability of Upcoming Word

$$P(X_t | X_1, \dots, X_{t-1})$$



LMs as a Marginal Distribution

- Directly we train models on “marginals”:



LMs as Implicit Joint Distribution over Language

- While language modeling involves learning the marginals, we are implicitly learning the full/joint distribution of language.
 - Remember the chain rule:

$$P(X_1, \dots, X_t) = P(X_1) \prod_{i=1}^t P(X_i | X_1, X_2, \dots, X_{i-1})$$

- **Language Modeling** \triangleq learning prob distribution over language sequence.

Doing Things with Language Model

- What is the probability of

“I like Johns Hopkins University”

“like Hopkins I University Johns”

Doing Things with Language Model

- What is the probability of

“I like Johns Hopkins University”

“like Hopkins I University Johns”

- LMs assign a probability to every sentence (or any string of words).

$P(\text{"I like Johns Hopkins University EOS"}) = 10^{-5}$

$P(\text{"like Hopkins I University Johns EOS"}) = 10^{-15}$

Doing Things with Language Model (2)

- We can rank sentences.
- While LMs show “typicality”, this may be a proxy indicator to other properties:
 - Grammaticality, fluency, factuality, etc.

$$P(X_t | X_1, \dots, X_{t-1})$$

next word context



$P("I \ like \ Johns \ Hopkins \ University. \ EOS") > P("I \ like \ John \ Hopkins \ University \ EOS")$

$P("I \ like \ Johns \ Hopkins \ University. \ EOS") > P("University. \ I \ Johns \ EOS \ Hopkins \ like")$

$P("JHU \ is \ located \ in \ Baltimore. \ EOS") > P("JHU \ is \ located \ in \ Virginia. \ EOS")$

Doing Things with Language Model (3)

- Can also generate strings!

$$P(X_t | X_1, \dots, X_{t-1})$$

next word context



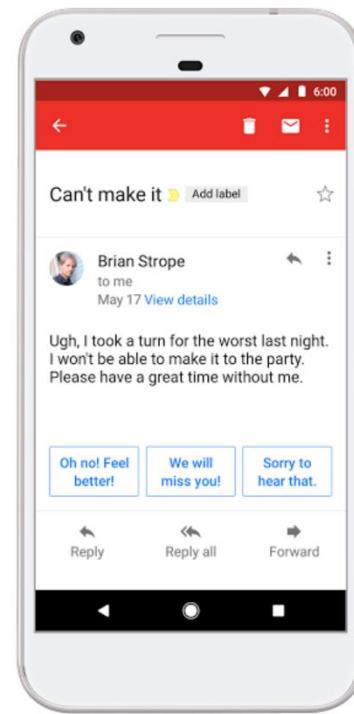
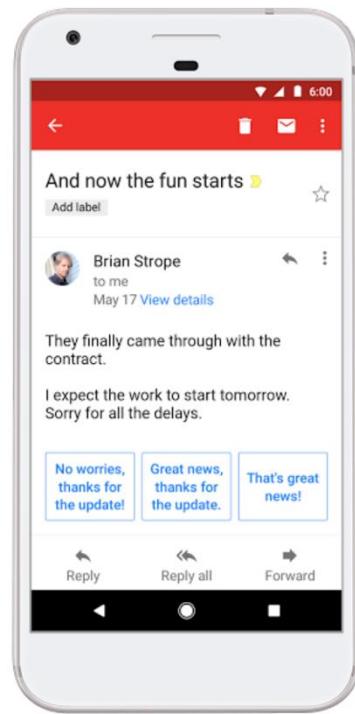
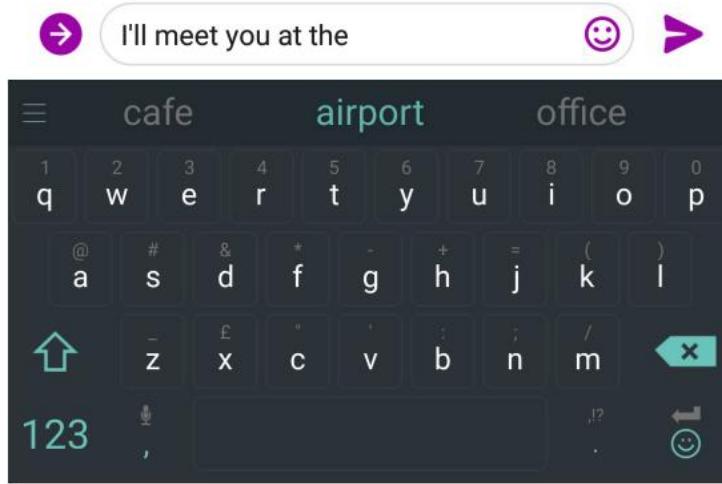
- Let's say we start "*Johns Hopkins is*"
- Using this prompt as an initial condition, recursively sample from an LM:

1. Sample from $P(X | "Johns\ Hopkins\ is") \rightarrow "located"$
2. Sample from $P(X | "Johns\ Hopkins\ is\ located") \rightarrow "at"$
3. Sample from $P(X | "Johns\ Hopkins\ is\ located\ at") \rightarrow "the"$
4. Sample from $P(X | "Johns\ Hopkins\ is\ located\ at\ the") \rightarrow "state"$
5. Sample from $P(X | "Johns\ Hopkins\ is\ located\ at\ the\ state") \rightarrow "of"$
6. Sample from $P(X | "Johns\ Hopkins\ is\ located\ at\ the\ state\ of") \rightarrow "Maryland"$
7. Sample from $P(X | "Johns\ Hopkins\ is\ located\ at\ the\ state\ of\ Maryland") \rightarrow "EOS"$

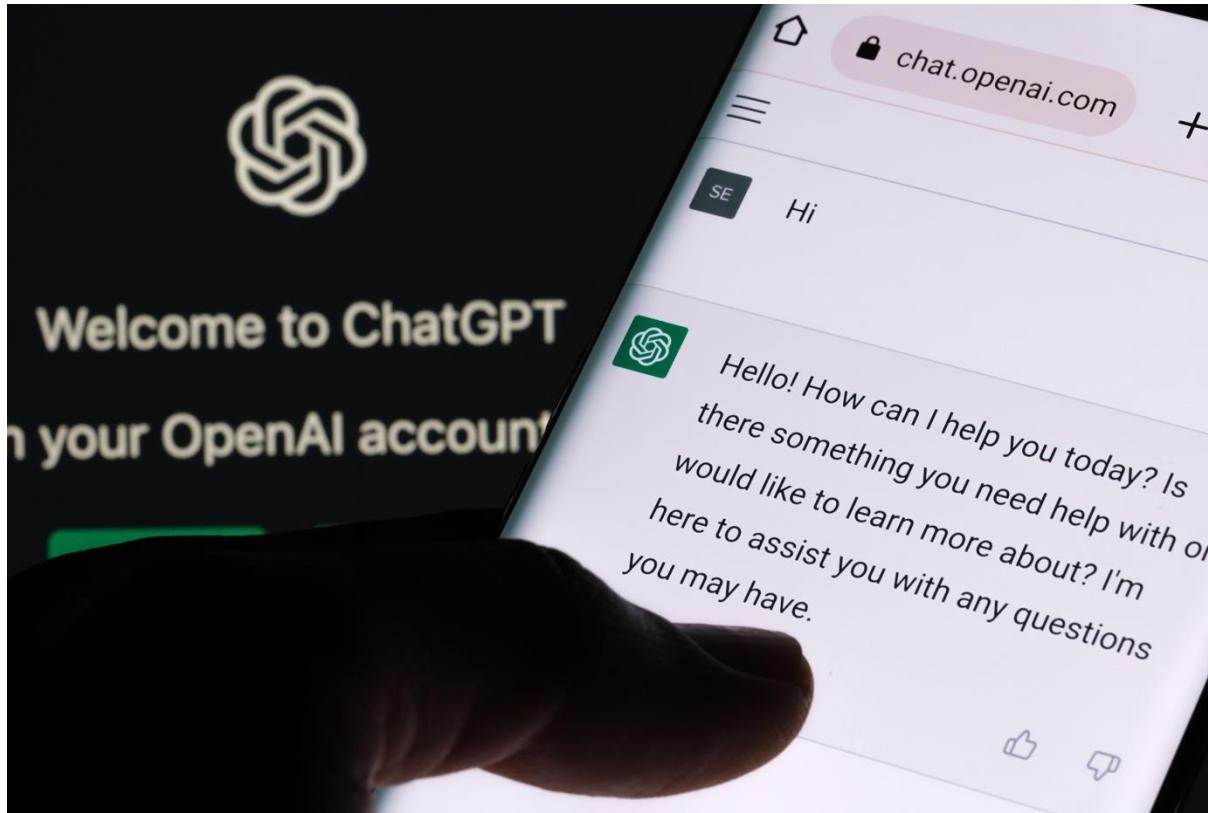
Why Care About Language Modeling?

- Language Modeling is a **subcomponent superset** of many tasks:
 - Summarization
 - Machine translation
 - Spelling correction
 - Dialogue etc.
- Language Modeling is an effective proxy for **language understanding**.
 - **Effective ability to predict forthcoming words** requires **understanding of context/prefix**.

You use Language Models every day!



You use Language Models every day!



Summary

- **Language modeling:** building probabilistic distribution over language.
- An accurate distribution of language enables us to solve many important tasks that involve language communication.
- **The remaining question:** how do you actually estimate this distribution?

Language Modeling with Counting

LMs as a Marginal Distribution

- Now the question is, how to estimate this distribution.

$$\mathbf{P}(X_t | X_1, \dots, X_{t-1})$$

How do we estimate these probabilities?

Let's just count!

$$P(\text{mat} | \text{the cat sat on the}) \approx \frac{\text{count}(\text{"the cat sat on the mat"})}{\text{count}(\text{"the cat sat on the"})}$$

Count how often
“**the cat sat on the mat**”
has appeared in the world (internet)!

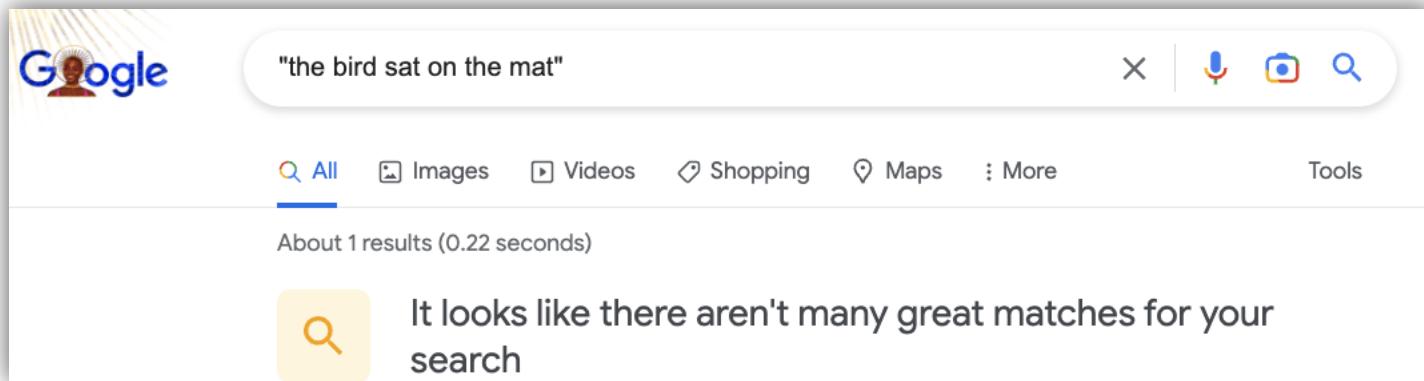
Divide that by, the count of
“**the cat sat on the**”
in the world (internet)!

$$\mathbf{P}(X_t | X_1, \dots, X_{t-1})$$

How do we estimate these probabilities?

Let's just count!

$$P(\text{mat} | \text{the cat sat on the}) \approx \frac{\text{count}(\text{"the cat sat on the mat"})}{\text{count}(\text{"the cat sat on the"})}$$



A screenshot of a Google search results page. The search bar contains the query "the bird sat on the mat". Below the search bar, there are navigation links for All, Images, Videos, Shopping, Maps, More, and Tools. A message indicates "About 1 results (0.22 seconds)". At the bottom, a yellow search icon is followed by the text: "It looks like there aren't many great matches for your search".

$$\mathbf{P}(X_t | X_1, \dots, X_{t-1})$$

How do we estimate these probabilities?

Let's just count!

$$P(\text{mat} | \text{the cat sat on the}) \approx \frac{\text{count}(\text{"the cat sat on the mat"})}{\text{count}(\text{"the cat sat on the"})}$$

Challenge: Increasing n makes **sparsity problems** worse.

Typically, we can't have n bigger than 5.

Some partial solutions (e.g., smoothing and backoffs)
though still an open problem.

Language Models: A History

- Shannon (1950): The redundancy and predictability (entropy) of English.



Prediction and Entropy of Printed English

By C. E. SHANNON

(Manuscript Received Sept. 15, 1950)

A new method of estimating the entropy and redundancy of a language is described. This method exploits the knowledge of the language statistics possessed by those who speak the language, and depends on experimental results in prediction of the next letter when the preceding text is known. Results of experiments in prediction are given, and some properties of an ideal predictor are developed.



$$P(X_t | X_1, \dots, X_{t-1})$$

Shannon (1950) built an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is **conditionally independent** of its non-descendants, **given its parents**.

1st order approximation:

$$P(\text{mat} | \text{the cat sat on the}) \approx P(\text{mat} | \text{the})$$

1 element



$$P(X_t | X_1, \dots, X_{t-1})$$

Andrey Markov

Shannon (1950) built an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is **conditionally independent** of its non-descendants, **given its parents**.

2nd order approximation:

$$P(\text{mat} | \text{the cat sat on the}) \approx P(\text{mat} | \text{on the})$$

2 elements



$$P(X_t | X_1, \dots, X_{t-1})$$

Andrey Markov

Shannon (1950) built an approximate language model with word co-occurrences.

Markov assumptions: every node in a Bayesian network is **conditionally independent** of its non-descendants, **given its parents**.

3rd order approximation:

$$P(\text{mat} | \text{the cat sat on the}) \approx P(\text{mat} | \text{sat on the})$$

3 elements



$$P(X_t | X_1, \dots, X_{t-1})$$

Andrey Markov

Shannon (1950) built an approximate language model with word co-occurrences.

Then, we can use counts of approximate conditional probability.
Using the 3rd order approximation, we can:

$$P(\text{mat} | \text{the cat sat on the}) \approx P(\text{mat} | \text{sat on the}) = \frac{\text{count("sat on the mat")}}{\text{count("on the mat")}}$$

Understanding Sparsity: A Thought Experiment

- How common are zero-probabilities? 😐
- **Example:** Shakespeare as a text corpus
 - The size vocab used by Shakespeare: $|V|=29,066$
 - Shakespeare produced: $\sim 300,000$ bigrams
 - Out of $|V|^2 = 844$ million possible bigrams
 - (some of them don't make sense, but ok!)
- So, **99.96%** of the possible bigrams are **never seen** (hence, have zero entries for bigram counts).

N-gram Language Models

- **Terminology:** *n*-gram is a chunk of *n* consecutive words:
 - unigrams: “cat”, “mat”, “sat”, ...
 - bigrams: “the cat”, “cat sat”, “sat on”, ...
 - trigrams: “the cat sat”, “cat sat on”, “sat on the”, ...
 - four-grams: “the cat sat on”, “cat sat on the”, “sat on the mat”, ...
- *n*-gram language model:

$$P(X_t | X_1, \dots, X_{t-1}) \approx P(X_t | \underbrace{X_{t-n+1}, \dots, X_{t-1}}_{n-1 \text{ elements}})$$

Generation from N-Gram Models

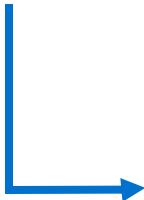
- You can build a simple **trigram** Language Model over a 1.7 million words corpus in a few seconds on your laptop*

Generation from N-Gram Models

- You can build a simple **trigram** Language Model over a 1.7 million words corpus in a few seconds on your laptop*

today the

get probability
distribution



company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

Sparsity problem: not
much granularity in the
probability distribution

Otherwise, seems reasonable!

Generation from N-Gram Models

- Now we can sample from this mode:

today the

get probability
distribution



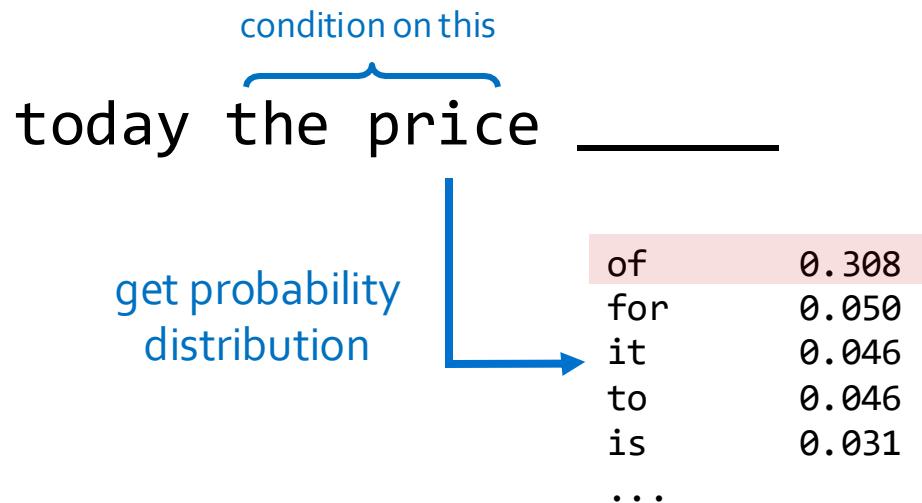
company	0.153
bank	0.153
price	0.077
italian	0.039
emirate	0.039
...	

Sparsity problem: not
much granularity in the
probability distribution

Otherwise, seems reasonable!

Generation from N-Gram Models

- Now we can sample from this mode:

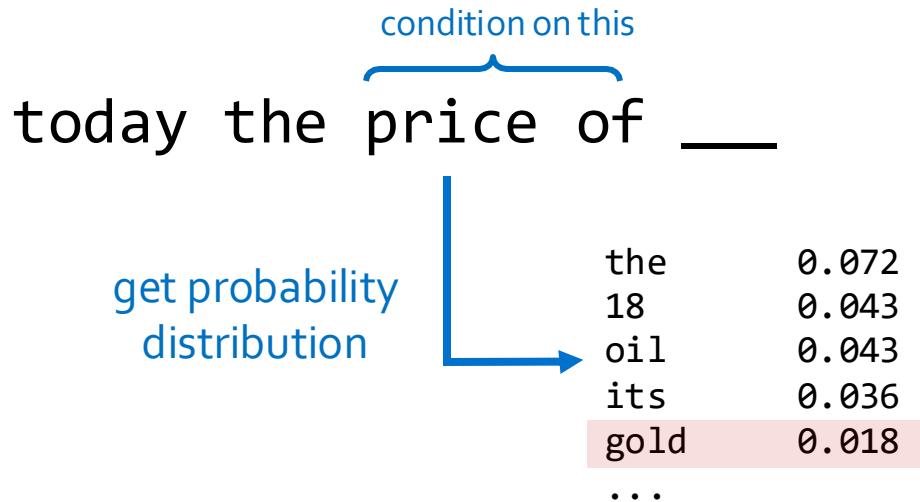


Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

Generation from N-Gram Models

- Now we can sample from this mode:



Sparsity problem: not much granularity in the probability distribution

Otherwise, seems reasonable!

N-Gram Models in Practice

- Now we can sample from this mode:

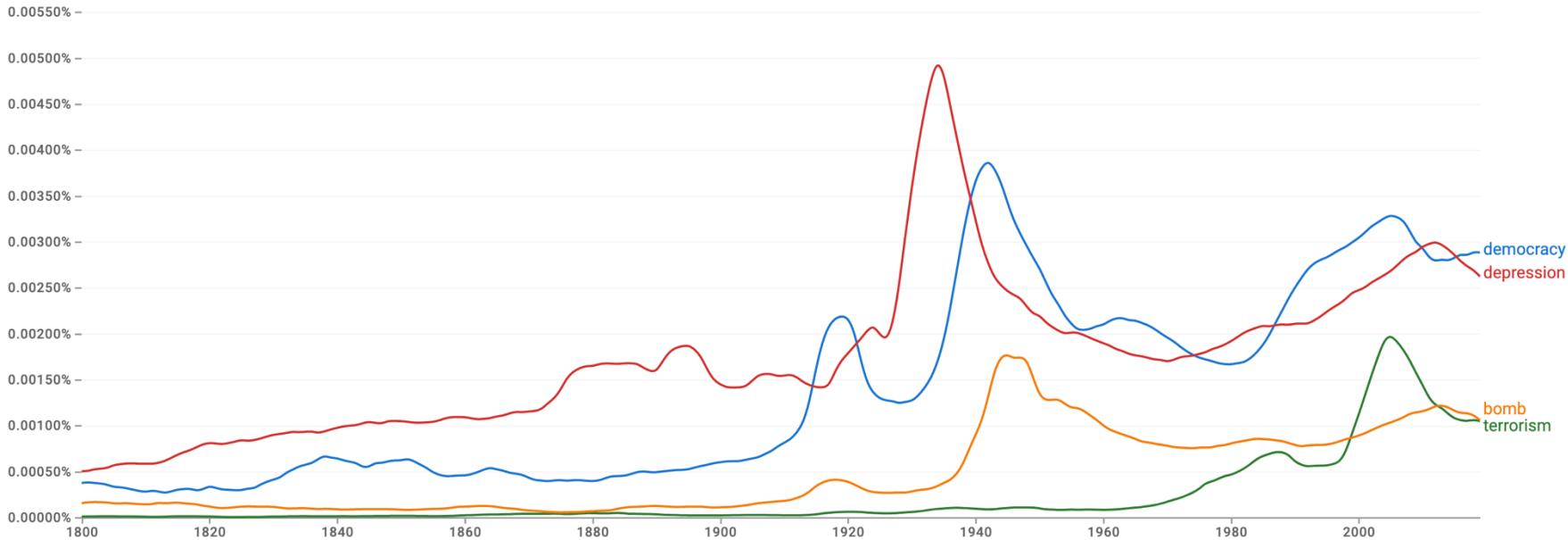
today the price of gold per ton , while production of shoe lasts and shoe industry , the bank intervened just after it considered and rejected an imf demand to rebuild depleted european stocks , sept 30 end primary 76 cts a share .

Surprisingly grammatical!

But quite incoherent! To improve coherence, one may consider increasing larger than 3-grams, but that would worsen the sparsity problem!

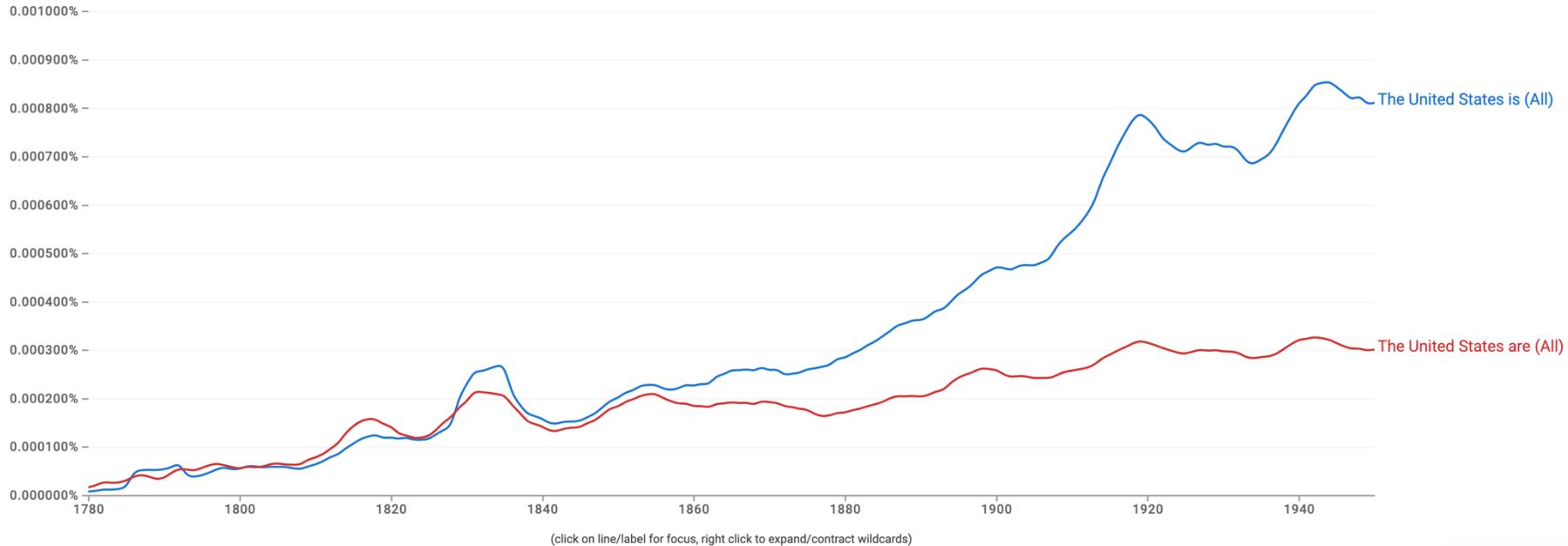
Pre-Computed N-Grams

Google Books Ngram Viewer



Pre-Computed N-Grams

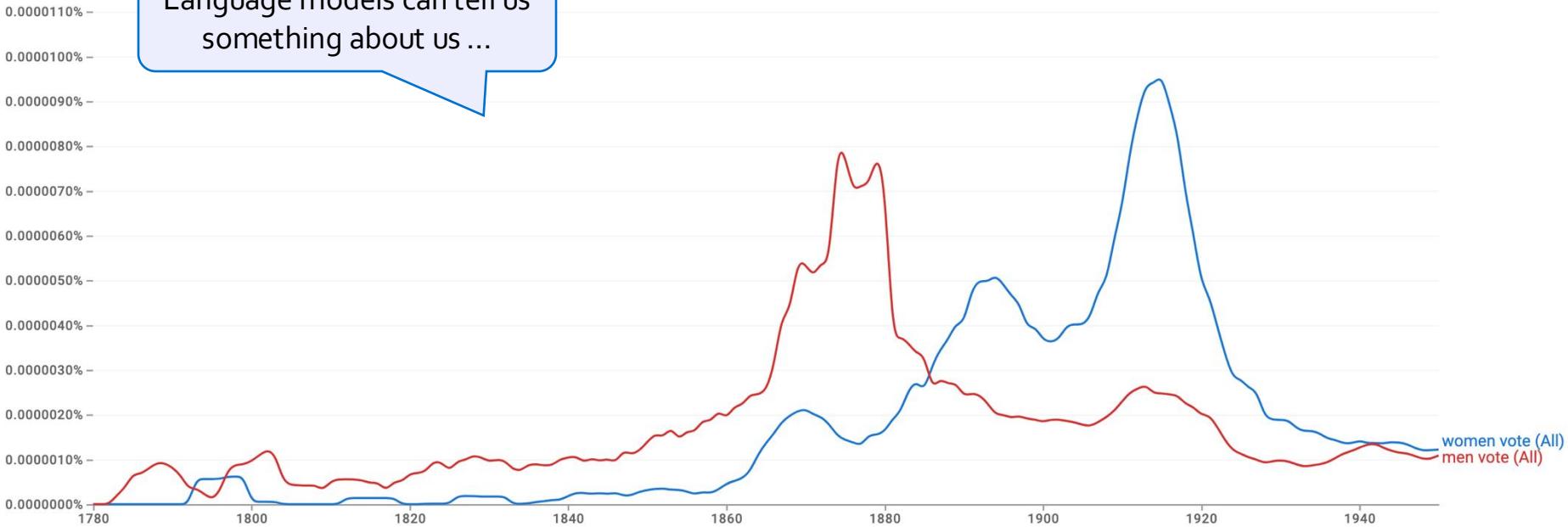
Google Books Ngram Viewer



Pre-Computed N-Grams

Google Books Ngram Viewer

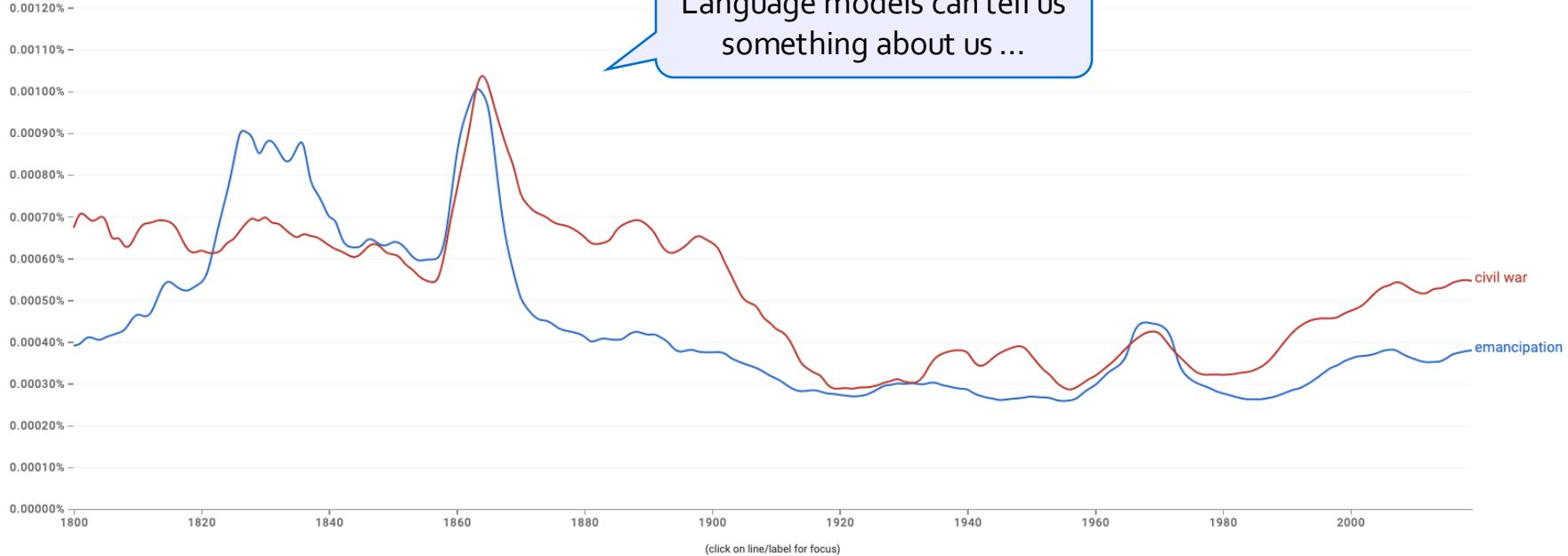
Language models can tell us something about us ...



Pre-Computed N-Grams

Google Books Ngram Viewer

Language models can tell us something about us ...



Limits of N-Grams LMs: Long-range Dependencies

- In general, count-based LMs are insufficient models of language because language has **long-distance dependencies**:

“**The computer** which I had just put into the machine room on the fifth floor **crashed.**”

N-Gram Language Models, A Historical Highlight

"Every time I fire a linguist, the performance of the speech recognizer goes up"!!



Fred Jelinek
(1932-2010)

- Probabilistic n-gram models of text generation [Jelinek+ 1980's, ...]
 - Applications: Speech Recognition, Machine Translation

532

PROCEEDINGS OF THE IEEE, VOL. 64, NO. 4, APRIL 1976

Continuous Speech Recognition by Statistical Methods

FREDERICK JELINEK, FELLOW, IEEE

Abstract—Statistical methods useful in automatic recognition of continuous speech are described. They concern modeling of a speaker and of an acoustic processor, extraction of the models' statistical parameters, and hypothesis search procedures and likelihood computations of linguistic decoding. Experimental results are presented that indicate the power of the methods.

utterance models used will incorporate more grammatical features, and statistics will have been grafted onto grammatical models. Most methods presented here concern modeling of the speaker's and acoustic processor's performance and should, therefore, be universally useful.

Automatic recognition of continuous (English) speech is an

Summary

- Learning a language model ~ learning **conditional probabilities** over language.
- One approach to estimating these probabilities: **counting word co-occurrences**.
- Challenges:
 - Word co-occurrences become **rare** for long sequences. (the sparsity issue)
 - But language understanding requires **long-range** dependencies.
- We need a better alternative! 😱
- **Next:** Measuring quality of language models.

How Good are Language Models?

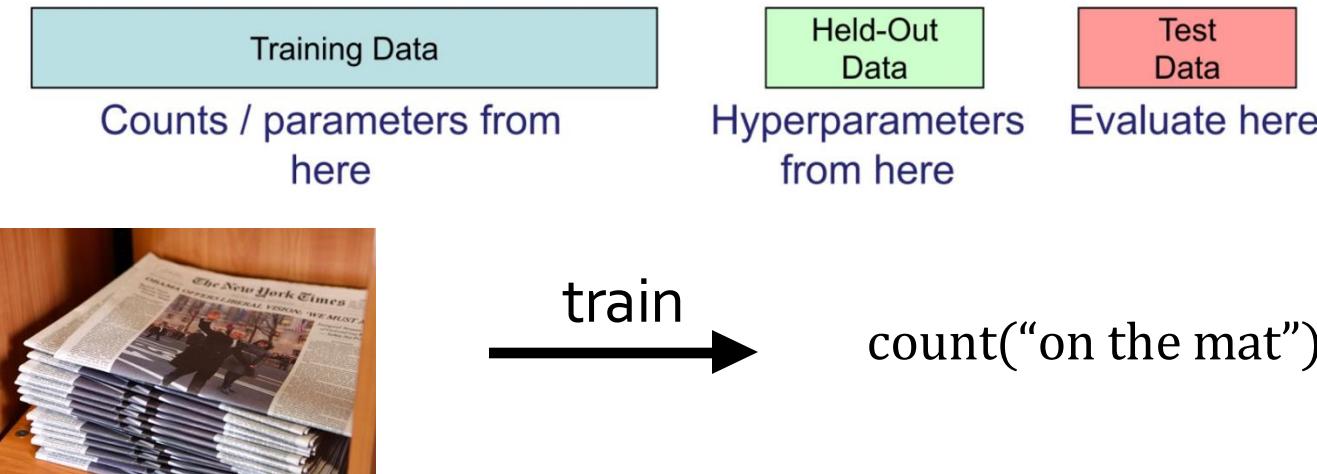
Evaluating Language Models

- Does our language model prefer good sentences to bad ones?
 - Assign higher probability to “real” or “frequently observed” sentences
 - Than “ungrammatical” or “rarely observed” sentences?
- We test the model’s performance on data we haven’t seen.

Evaluating Language Models

Setup:

- Train it on a suitable training documents.
- Evaluate their predictions on different, unseen documents.
- An evaluation metric tells us how well our model does on the test set.



Evaluating Language Models: Example

Setup:

- Train it on a suitable training documents.
- Evaluate their predictions on different, unseen documents.
- An evaluation metric tells us how well our model does on the test set.

Example: I use a bunch of New York Times articles to build a bigram probability table



A good language model
should assign a high
probability to held-out text!

train

count("on the mat")

eval



Now I'm going to evaluate the probability of some heldout data using our bigram table

Be Careful About Data Leakage!

Advice from a grandpa 🧑:

- Don't allow test sentences to leak into training set.
- Otherwise, you will assign it an artificially high probability (==cheating).

Example: I use a bunch of New York Times articles to build a bigram probability table



A good language model
should assign a high
probability to held-out text!

train →

count("on the mat")

eval →



Now I'm going to evaluate the probability of some heldout data using our bigram table

Evaluating Language Models: Intrinsic vs Extrinsic

- **Intrinsic:** measure how good we are at modeling language
- **Extrinsic:** build a new language model, use it for some task (MT, ASR, etc.)

Example: I use a bunch of New York Times articles to build a bigram probability table



train →



Now I'm going to evaluate the probability of some heldout data using our bigram table



Evaluation Metric for Language Modeling: Perplexity

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

$$\text{ppl}(w_1, \dots, w_n) = \mathbf{P}(w_1, w_2, \dots, w_n)^{-\frac{1}{n}}$$

- A measure of **predictive quality** of a language model.
- **Minimizing** perplexity is the same as **maximizing** probability

Evaluation Metric for Language Modeling: Perplexity

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

$$\text{ppl}(w_1, \dots, w_n) = \mathbf{P}(w_1, w_2, \dots, w_n)^{-\frac{1}{n}}$$

Evaluation Metric for Language Modeling: Perplexity

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

$$\begin{aligned} \text{ppl}(w_1, \dots, w_n) &= \mathbf{P}(w_1, w_2, \dots, w_n)^{-\frac{1}{n}} \\ &= \sqrt[n]{\frac{1}{\mathbf{P}(w_1, w_2, \dots, w_n)}} = \sqrt[n]{\prod_{i=1}^n \frac{1}{\mathbf{P}(w_i | w_{<i})}} \quad \text{chain rule} \\ &= 2^H, \text{ where} \end{aligned}$$

$$H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use **log**-probabilities (also known as “logits”)
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \dots, w_n) = 2^H, \text{ where } H = -\frac{1}{n} \sum_{i=1}^n \log_2 P(w_i | w_1, \dots, w_{i-1})$$

Recap: Definition of **cross-entropy** between two distributions:

$$H(p, q) = - \sum_{x \in \mathcal{X}} p(x) \log q(x)$$

Can be interpreted as **cross-entropy** between LM prob and language prob. **Why?**

Evaluation Metric for Language Modeling: Perplexity

- In practice, we prefer to use **log**-probabilities (also known as “logits”)
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \dots, w_n) = 2^H, \text{ where } H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

- **Perplexity** for n-grams:
 - Unigrams: $H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i)$
 - Bigrams: $H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_{i-1})$
 - Trigrams: $H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_{i-2}, w_{i-1})$
 - ...

Intuition-building Quizzes (1)

- In practice, we prefer to use **log**-probabilities (also known as “logits”)
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \dots, w_n) = 2^H, \text{ where } H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

- **Quiz:** let's suppose we have a sentence w_1, \dots, w_n and it's fixed. Our model will correctly guess each word with probability 1/5. What is perplexity of our model?

$$H = -\frac{1}{n} \left[\log_2 \left(\frac{1}{5} \right) + \dots + \log_2 \left(\frac{1}{5} \right) \right] = -\log \left(\frac{1}{5} \right) \Rightarrow \text{ppl}(D) = 5$$

Intuition: the model is indecisive among 5 choices.

Intuition-building Quizzes (2)

- In practice, we prefer to use **log**-probabilities (also known as “logits”)
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \dots, w_n) = 2^H, \text{ where } H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

- **Quiz:** let's we evaluate an **exact** (!! model of language, i.e., our model always knows what exact word should follow a given context. What is the perplexity of this model?

$$\forall w \in V: \mathbf{P}(w_i | w_{1:i-1}) = 1 \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{2} n \log_2 1} = 1$$

Intuition: the model is indecisive among 1 (the right!) choice!

Intuition-building Quizzes (3)

- In practice, we prefer to use **log**-probabilities (also known as “logits”)
- We can rewrite perplexity formula in terms of log-probs:

$$\text{ppl}(w_1, \dots, w_n) = 2^H, \text{ where } H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

- **Quiz:** let's evaluate a **confused** (!!) model of language, i.e., our model has no idea what word should follow each context—it always chooses a uniformly random word. What is the perplexity of this model?

$$\forall w \in V: \mathbf{P}(w | w_{1:i-1}) = \frac{1}{|V|} \Rightarrow \text{ppl}(D) = 2^{-\frac{1}{n} n \log_2 \frac{1}{|V|}} = |V|$$

Intuition: the model is indecisive among all the vocabulary terms.

Perplexity: Summary

$$\text{ppl}(w_1, \dots, w_n) = 2^H, \text{ where } H = -\frac{1}{n} \sum_{i=1}^n \log_2 \mathbf{P}(w_i | w_1, \dots, w_{i-1})$$

- Perplexity is a measure of model's uncertainty about next word (aka "average branching factor").
 - The larger the number of vocabulary, the more options there to choose from.
 - (the choice of atomic units of language impacts PPL — more on this later)
- Perplexity ranges between 1 and $|V|$.
- We prefer LMs with lower perplexity.

Lower perplexity == Better Model

- Training on 38 million words, test 1.5 million words, Wall Street Journal

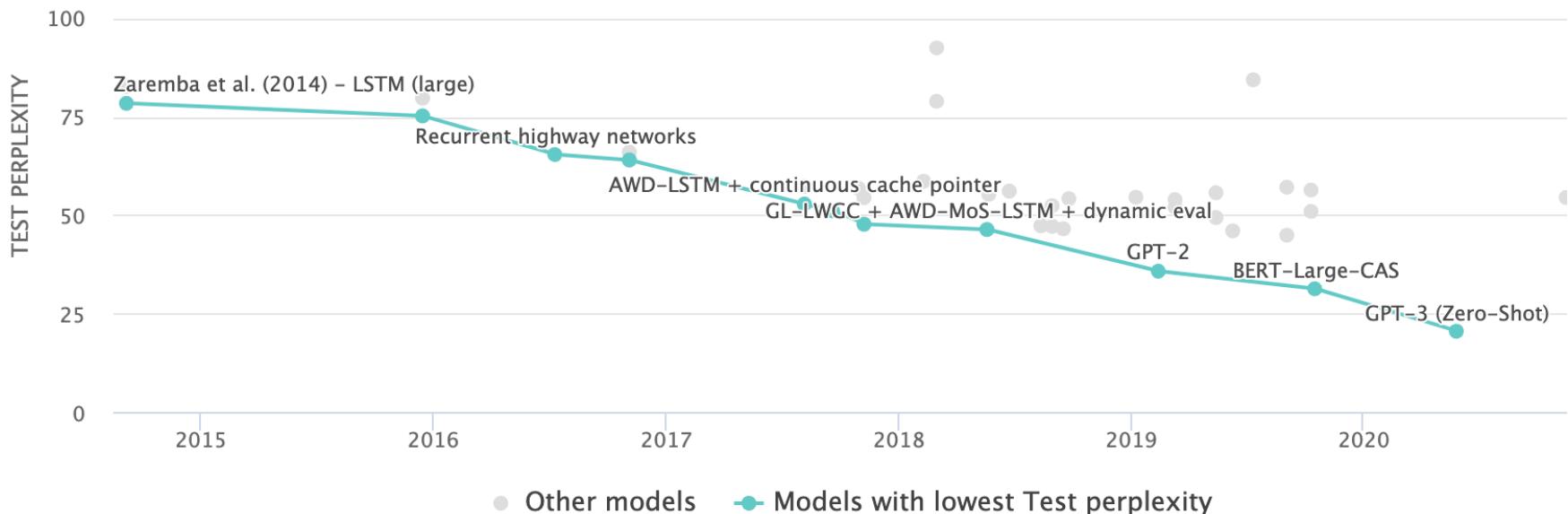
N-gram Order	Unigram	Bigram	Trigram
Perplexity	962	170	109

Lower is better

Note these evaluations are done on data that was not used for “counting.” (no cheating!!)

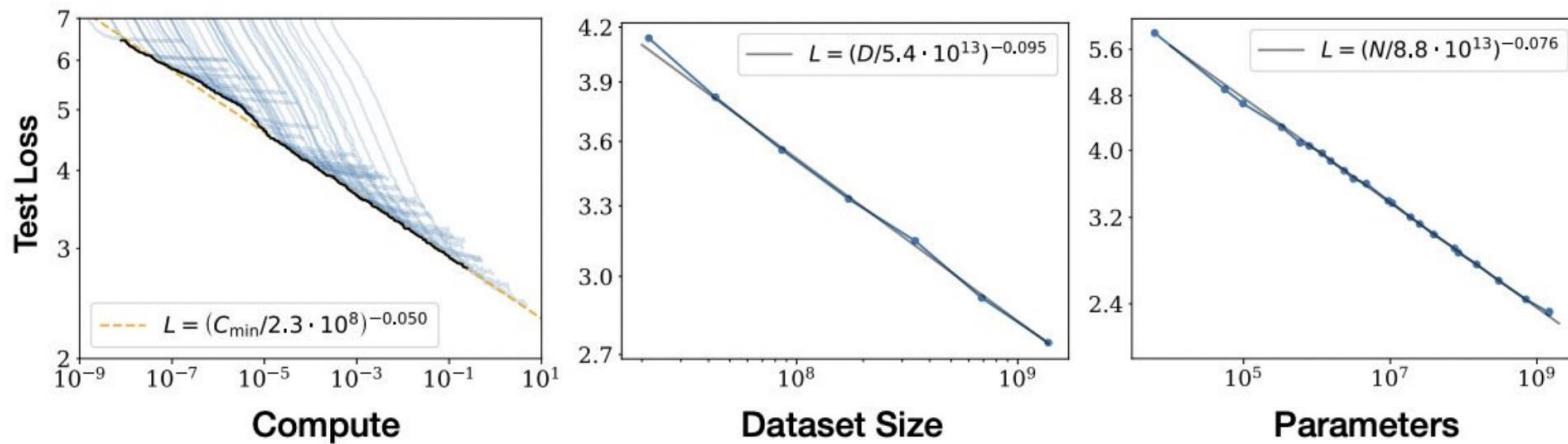
Lower perplexity == Better Model

The PPL of modern language models have consistently been going down.



Lower perplexity == Better Model

The PPL of modern language models have consistently been going down.



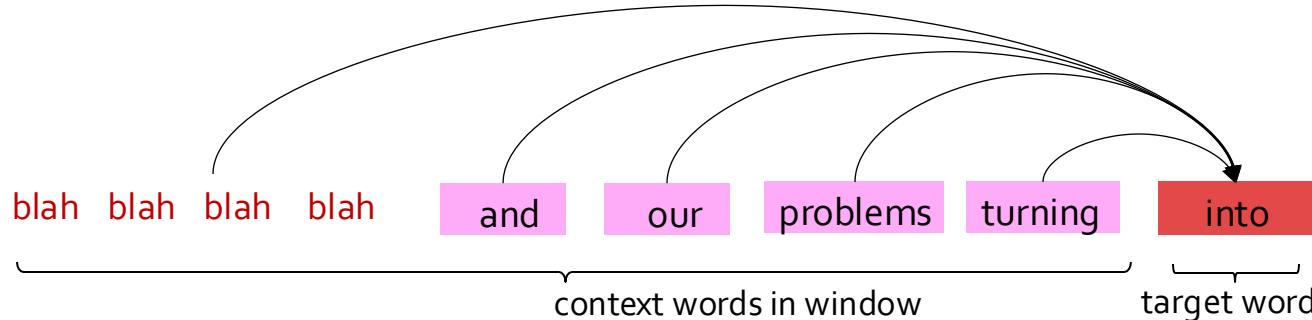
Summary

- Language Models (LM): distributions over language
- Measuring LM quality: use perplexity on held-out data.
- Count-based LMs have limitations.
 - Challenge with large N's: sparsity problem — many zero counts/probs.
 - Challenge with small N's: lack of long-range dependencies.
- Next: Rethinking language modeling as a statistical learning problem.

Beyond Counting: Language Models as a Learning Problem

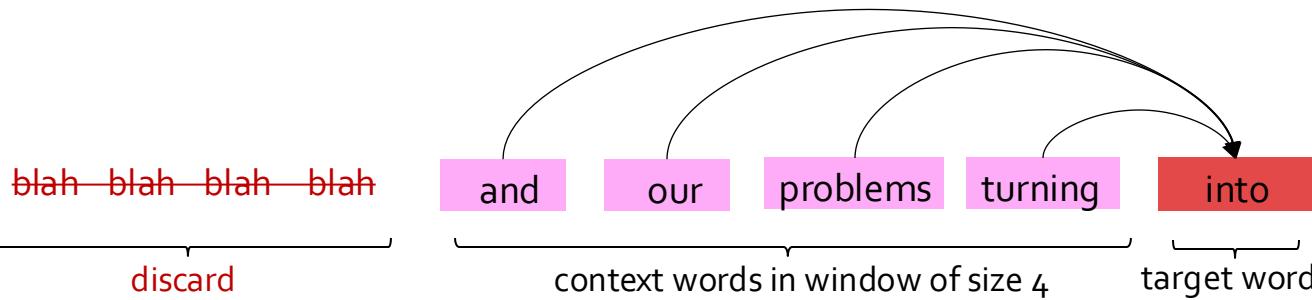
LM as a Machine Learning Problem

- Given the embeddings of the context, predict the word on the right side.
 - Dropping the right context for simplicity -- not a fundamental limitation.
- Discard anything beyond its context window



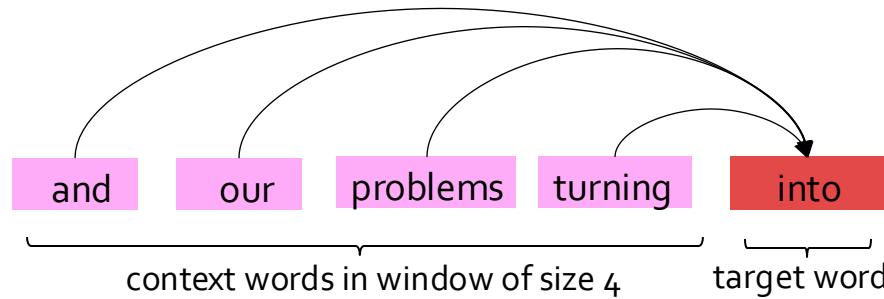
LM as a Machine Learning Problem

- Given the embeddings of the context, predict the word on the right side.
 - Dropping the right context for simplicity -- not a fundamental limitation.
- Discard anything beyond its context window



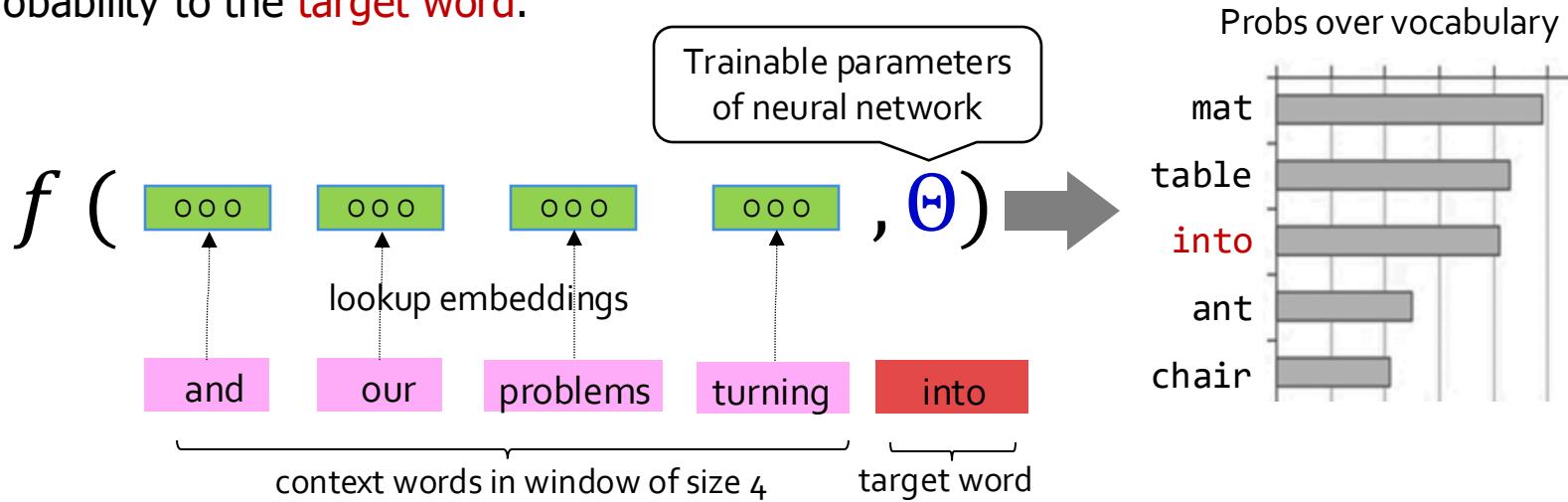
LM as a Machine Learning Problem

- Given the embeddings of the context, predict the word on the right side.
 - Dropping the right context for simplicity -- not a fundamental limitation.
- Discard anything beyond its context window



A Fixed-Window Neural LM

- Given the embeddings of the **context**, predict a **target word** on the right side.
 - Dropping the right context for simplicity -- not a fundamental limitation.
- Training this model is basically optimizing its parameters Θ such that it assigns high probability to the **target word**.



A Fixed-Window Neural LM

- It will also lay the foundation for the future models (recurrent nets, transformers, ...)
- But first we need to figure out how to train neural networks!

How do you build
this function?

$$f($$



Trainable parameters
of neural network



$$, \Theta)$$



Neural Networks
for rescue!

lookup embeddings

our

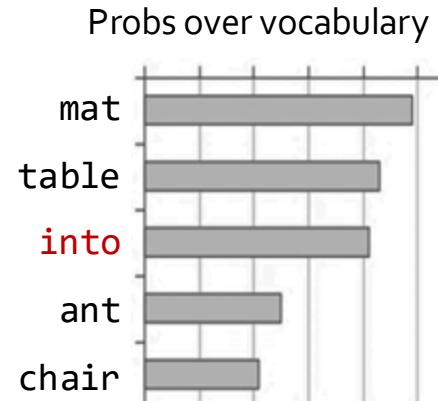
problems

turning

into

context words in window of size 4

target word



From Counting (N-Gram) to Neural Models

- n-gram models of text generation [Jelinek+ 1980's, ...]
 - Applications: Speech Recognition, Machine Translation
 - “Shallow” statistical/neural language models (2000’s) [Bengio+ 1999 & 2001, ...]
-

NeurIPS 2000

A Neural Probabilistic Language Model

Yoshua Bengio*, Réjean Ducharme and Pascal Vincent
Département d'Informatique et Recherche Opérationnelle
Centre de Recherche Mathématiques
Université de Montréal
Montréal, Québec, Canada, H3C 3J7
{bengioy,ducharme,vincentp}@iro.umontreal.ca

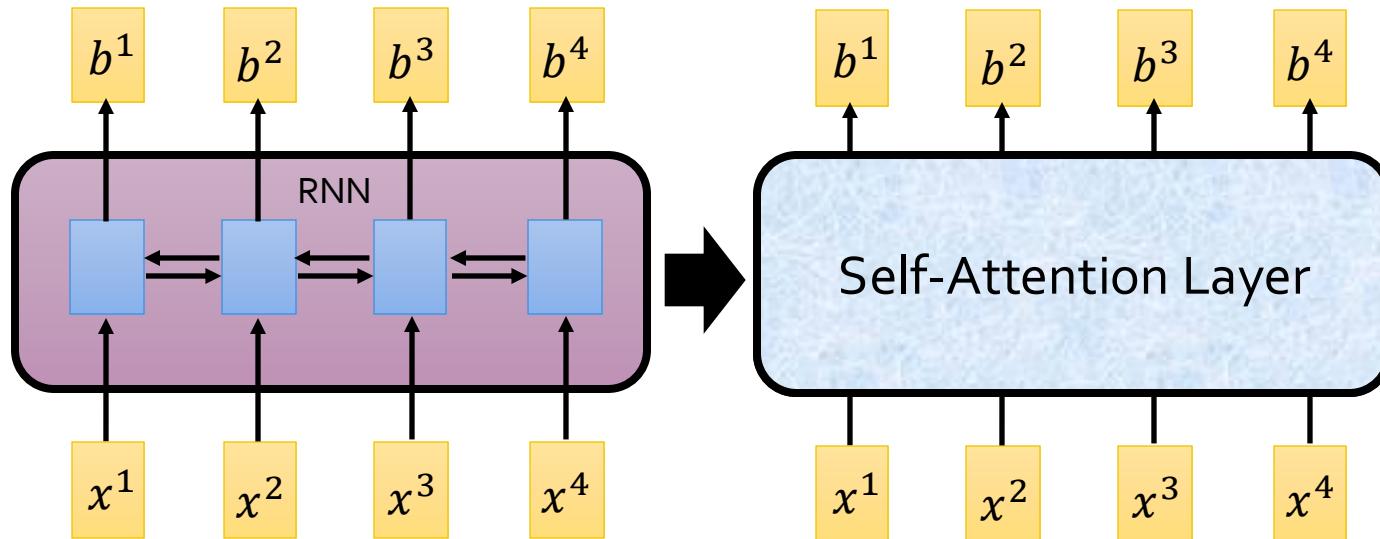
Summary

- **Language Modeling (LM)**, a useful predictive objective for language
- **Perplexity**, a measure of an LM's predictive ability
- **N-gram models** (~1980 to early 2000's),
 - Early instances of LMs
 - Difficult to scale to large window sizes
- **Shallow neural LMs** (early and mid-2000's),
 - These will be effective predictive models based on feed-forward networks
- **Recurrent neural LMs** (2010s),
 - Compact models used recursively

Self-Attention

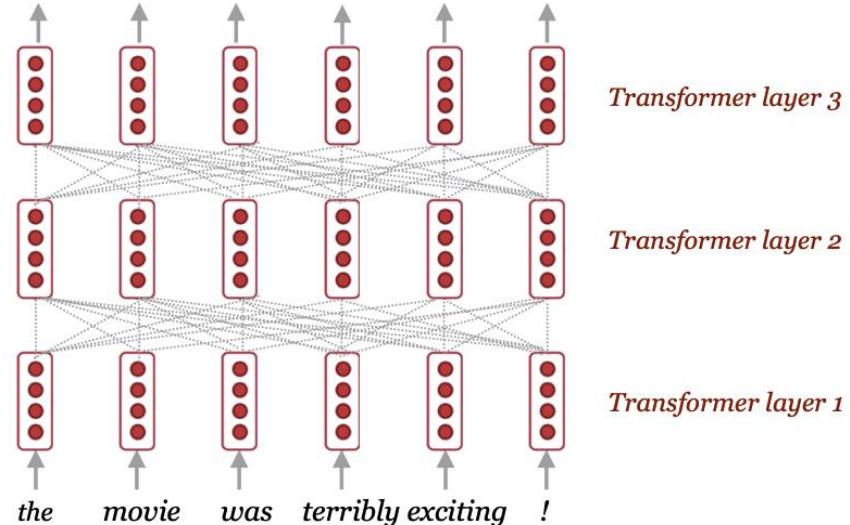
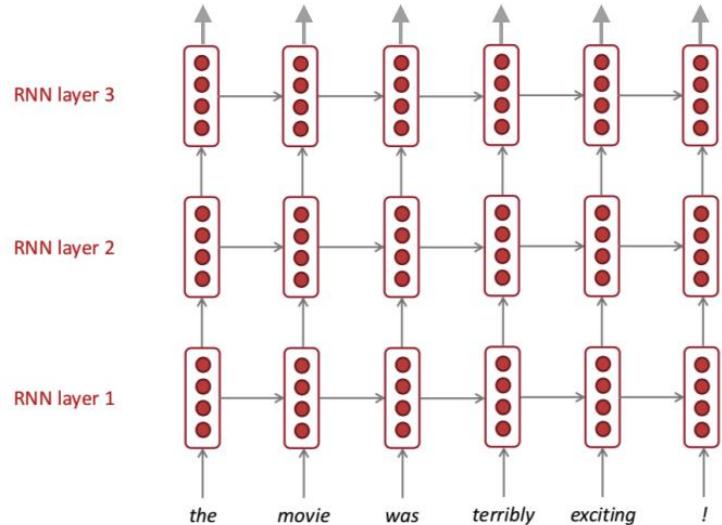
Self-Attention

- b^i is obtained based on the whole input sequence.
- can be parallelly computed.



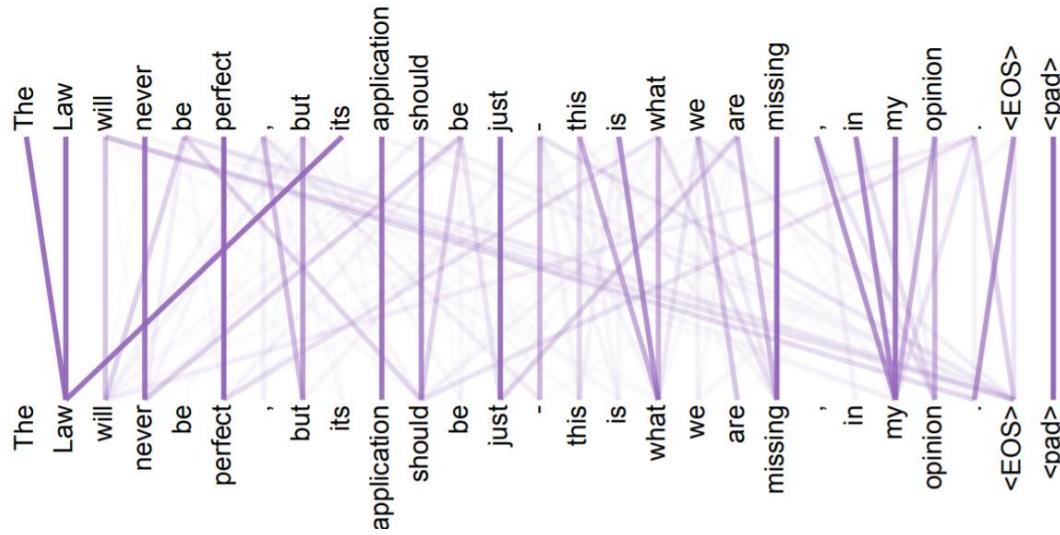
Idea: replace any thing done by RNN with **self-attention**.

RNN vs Transformer



Attention

- Core idea: build a mechanism to focus ("attend") on a particular part of the context.



Defining Self-Attention

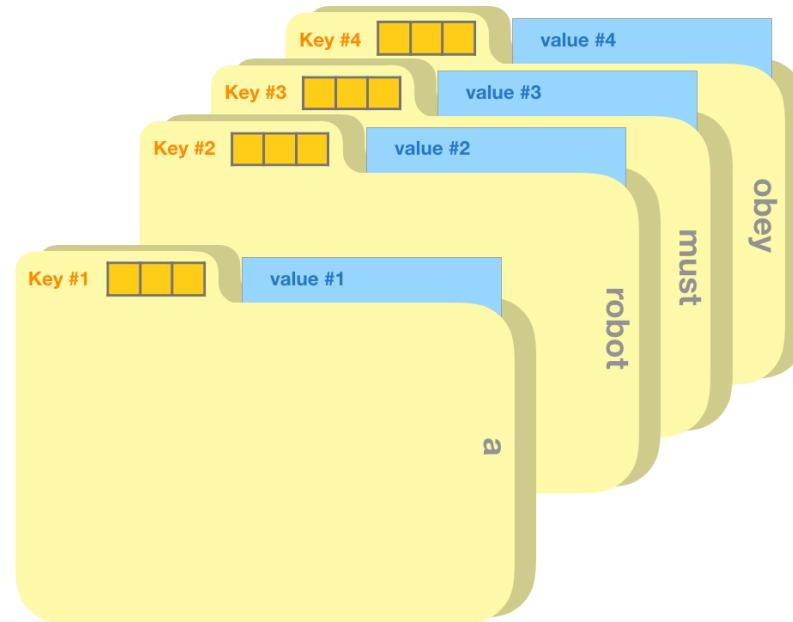
- Terminology:
 - **Query**: to match others
 - **Key**: to be matched
 - **Value**: information to be extracted

Defining Self-Attention

- Terminology:
 - **Query**: to match others
 - **Key**: to be matched
 - **Value**: information to be

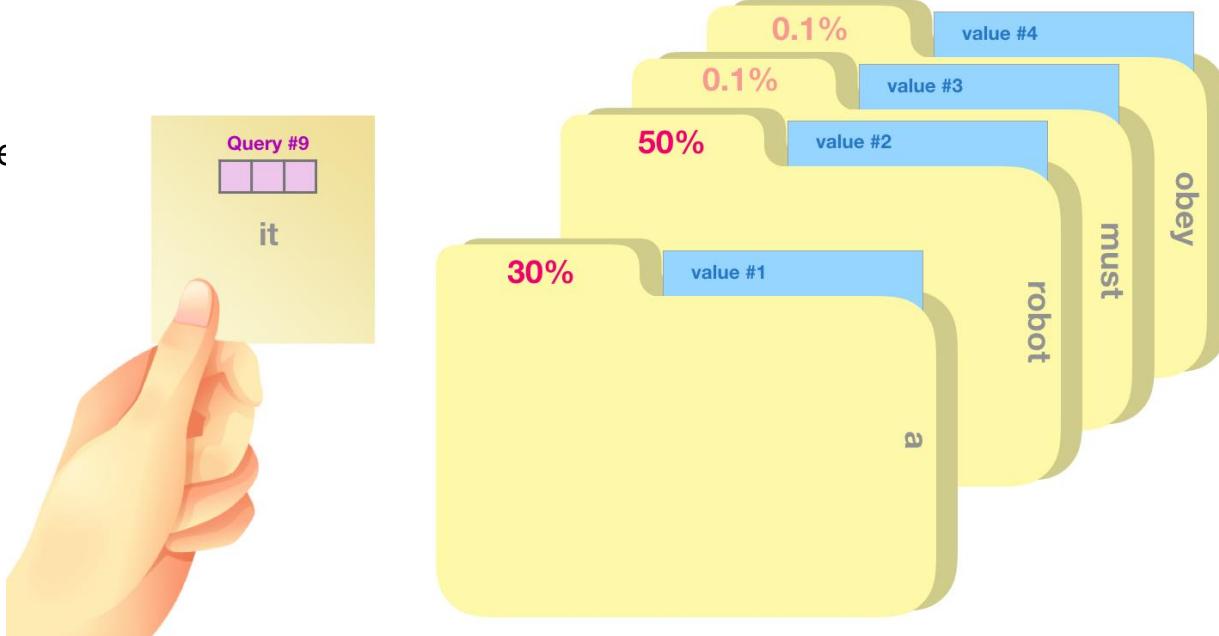


An analogy



Defining Self-Attention

- Terminology:
 - **Query**: to match others
 - **Key**: to be matched
 - **Value**: information to be



q: query (to match others)

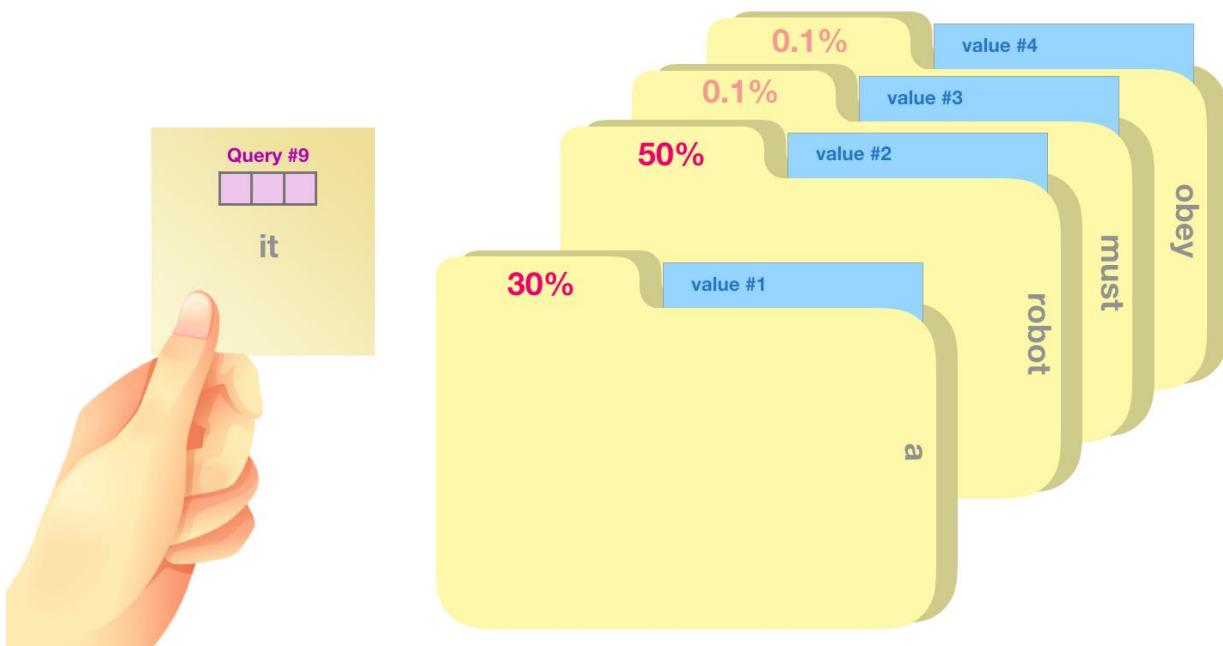
$$q_i = W^q x_i$$

k: key (to be matched)

$$k_i = W^k x_i$$

v: value (information to be extracted)

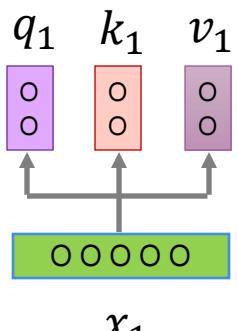
$$v_i = W^v x_i$$



q : query (to match others)
 $q_i = W^q x_i$

k : key (to be matched)
 $k_i = W^k x_i$

v : value (information to be extracted)
 $v_i = W^v x_i$

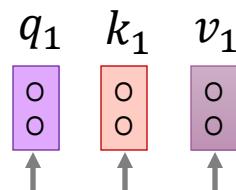


The

q: query (to match others)
 $q_i = W^q x_i$

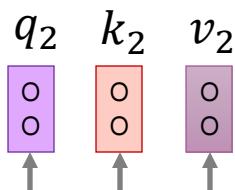
k: key (to be matched)
 $k_i = W^k x_i$

v: value (information to be extracted)
 $v_i = W^v x_i$



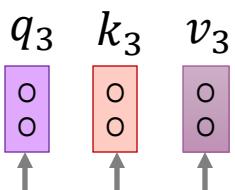
x_1

The



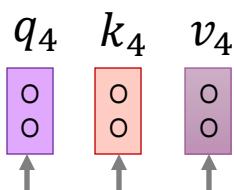
x_2

cat



x_3

sat



x_4

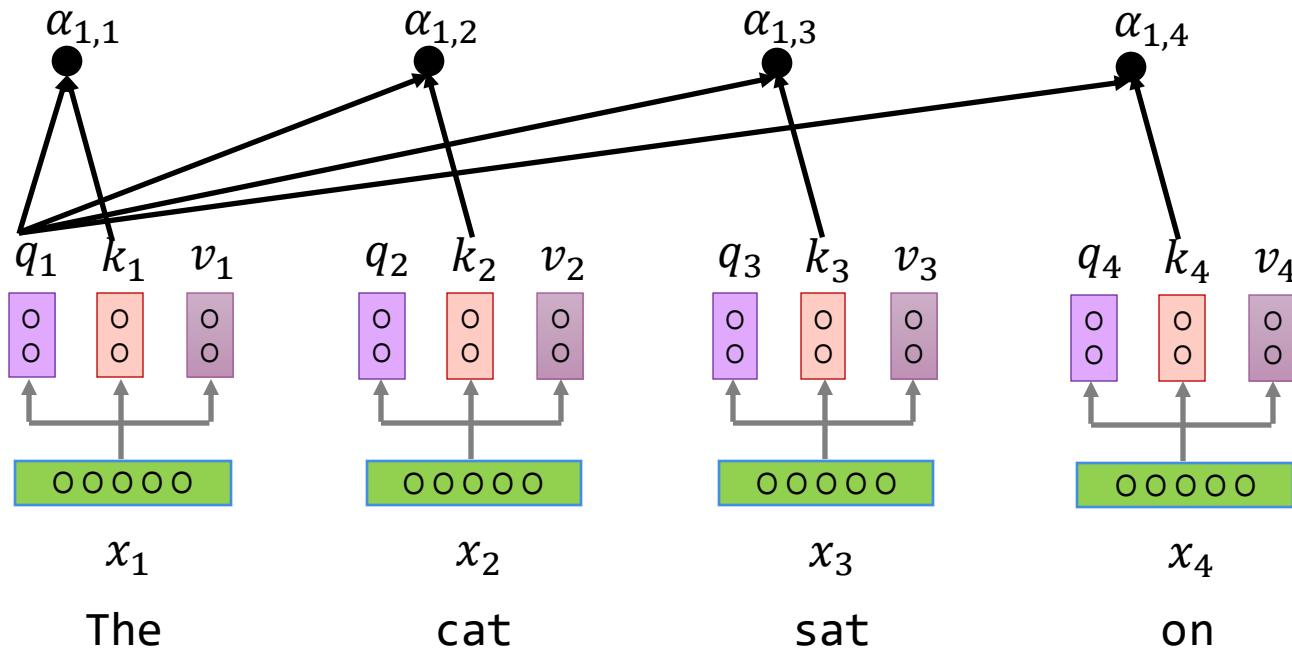
on

$$\alpha_{1,i} = \frac{q^1 \cdot k^i}{\sqrt{d}}$$

Scaled dot product

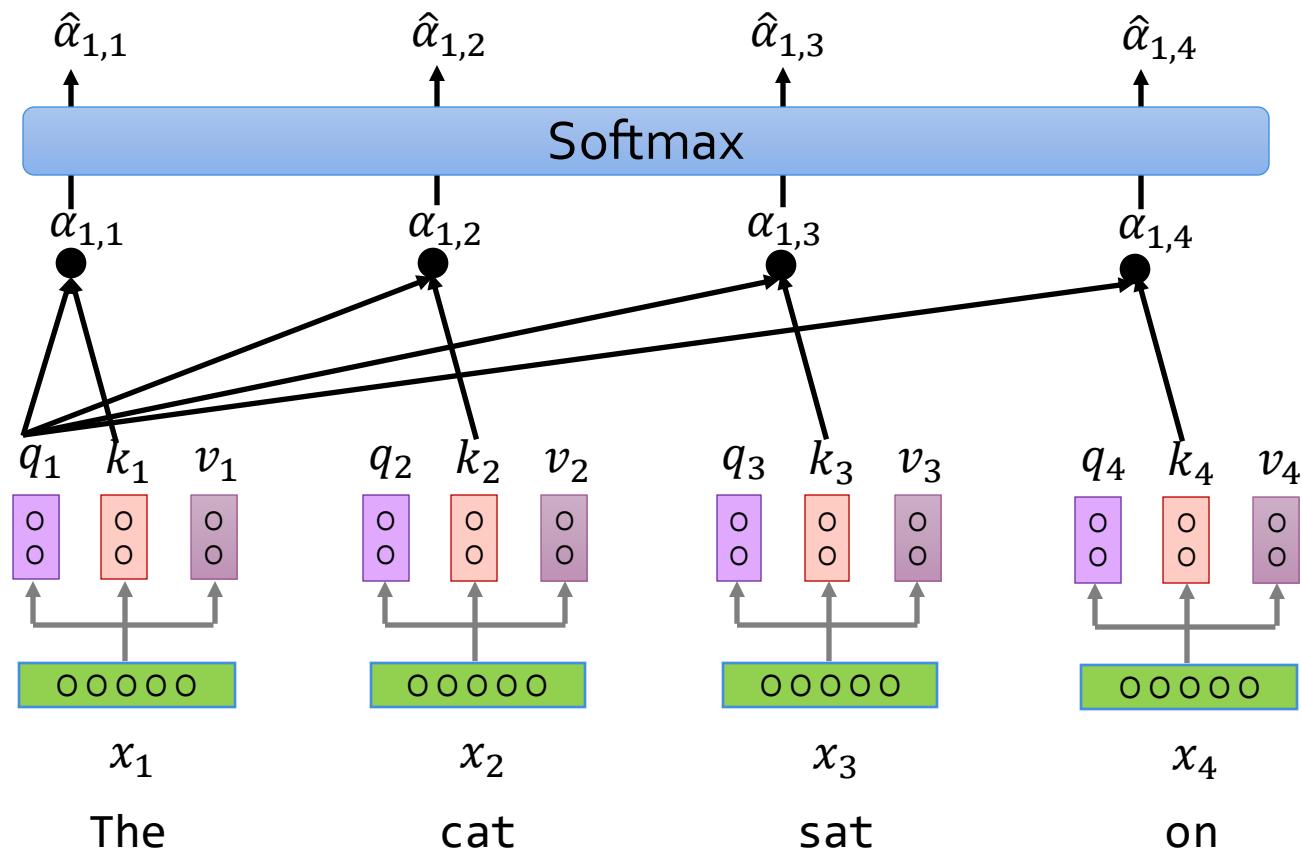
How much
should "The"
attend to other
positions?

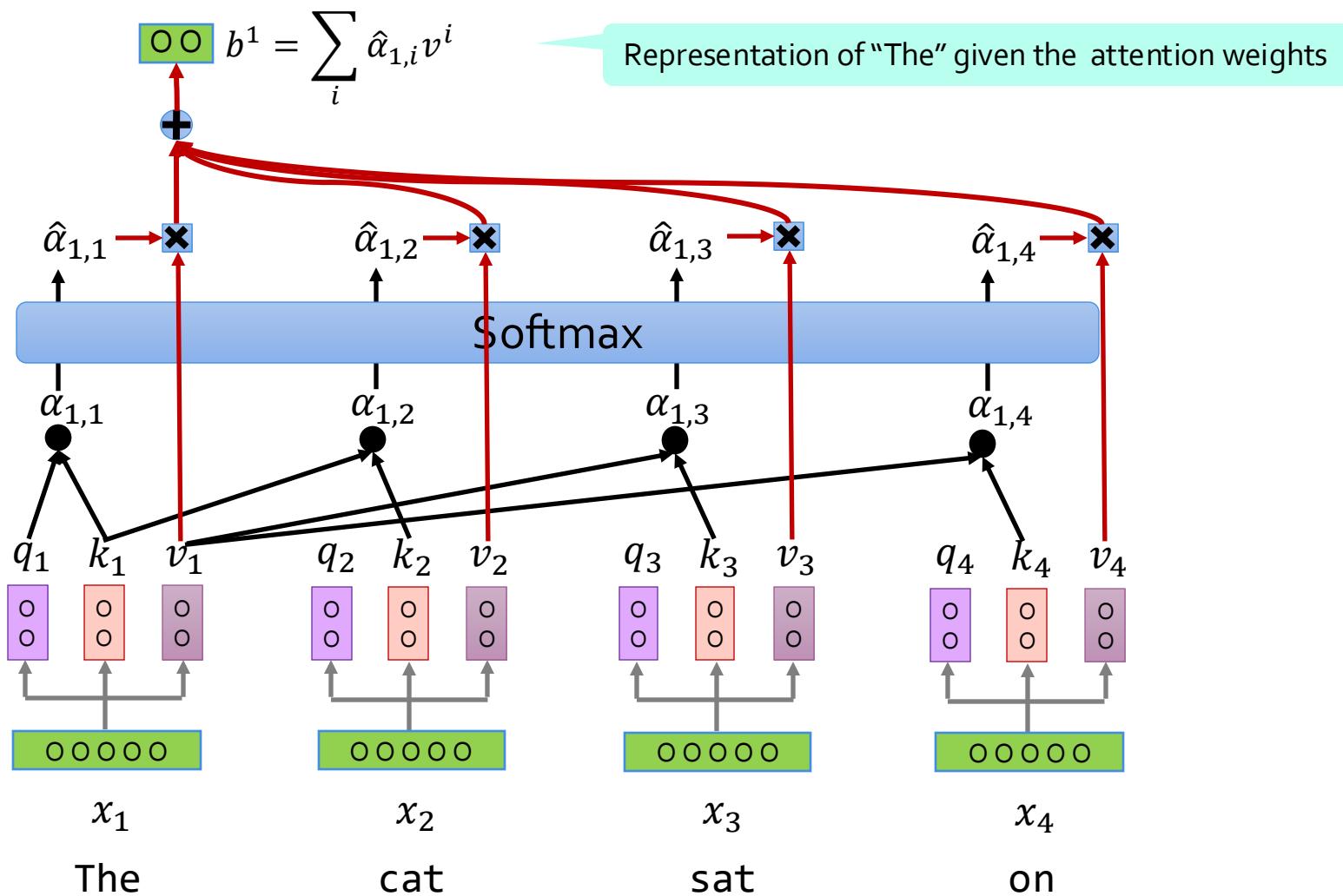
q: query (to match others)
k: key (to be matched)
v: value (information to be extracted)



$$\sigma(z)_i = \frac{\exp(z_i)}{\sum_j \exp(z_j)}$$

How much
should "The"
attend to other
positions?





Self-Attention

- Can write it in matrix form:
- Given input \mathbf{x} :

$$Q = \mathbf{W}^q \mathbf{x}$$

$$K = \mathbf{W}^k \mathbf{x}$$

$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



hardmaru

@hardmaru

...

The most important formula in deep learning after 2018

Self-Attention

What is self-attention? Self-attention calculates a weighted average of feature representations with the weight proportional to a similarity score between pairs of representations. Formally, an input sequence of n tokens of dimensions d , $X \in \mathbf{R}^{n \times d}$, is projected using three matrices $W_Q \in \mathbf{R}^{d \times d_q}$, $W_K \in \mathbf{R}^{d \times d_k}$, and $W_V \in \mathbf{R}^{d \times d_v}$ to extract feature representations Q , K , and V , referred to as query, key, and value respectively with $d_k = d_q$. The outputs Q , K , V are computed as

$$Q = XW_Q, \quad K = XW_K, \quad V = XW_V. \quad (1)$$

So, self-attention can be written as,

$$S = D(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_q}}\right)V, \quad (2)$$

where softmax denotes a *row-wise* softmax normalization function. Thus, each element in S depends on all other elements in the same row.

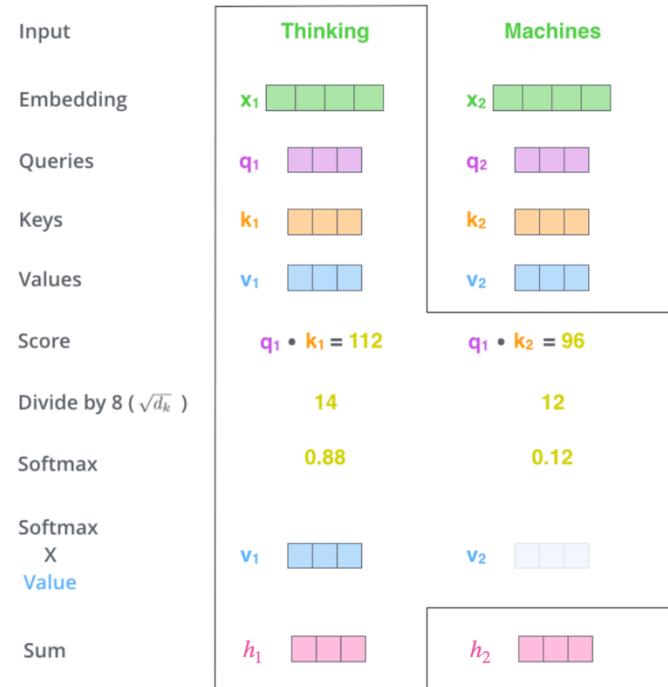
9:08 PM · Feb 9, 2021 · Twitter Web App

553 Retweets 42 Quote Tweets 3,338 Likes

Question

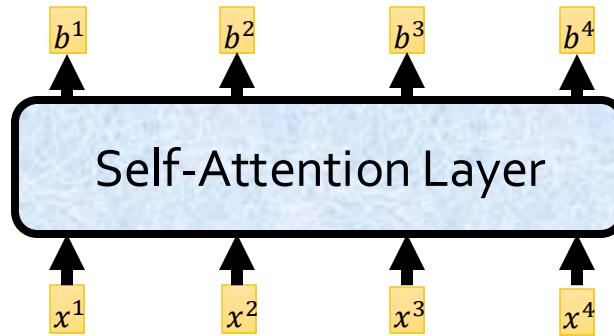
- What would be the output vector for the word “Thinking”?

- (a) $0.5\mathbf{v}_1 + 0.5\mathbf{v}_2$
- (b) $0.54\mathbf{v}_1 + 0.46\mathbf{v}_2$
- (c) $0.88\mathbf{v}_1 + 0.12\mathbf{v}_2$
- (d) $0.12\mathbf{v}_1 + 0.88\mathbf{v}_2$



Self-Attention: Back to Big Picture

- **Attention** is a powerful mechanism to create context-aware representations
- A way to focus on select parts of the input



- Better at maintaining **long-distance dependencies** in the context.

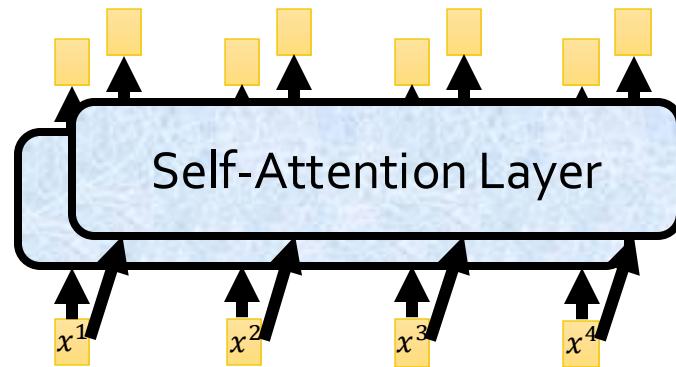
Properties of Self-Attention

Layer Type	Complexity per Layer	Sequential Operations
Self-Attention	$O(n^2 \cdot d)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$

- Per-layer statistics; n = sequence length, d = hidden dimension
- Complexity per layer: Quadratic function of n for SA
- Sequential operations: # of operations that must be performed sequentially
 - $O(1)$ sequential operations for SA.
 - SA layers computes all the operations in parallel across all tokens in the sequence
 - **Efficient** implementations

Multi-Headed Self-Attention

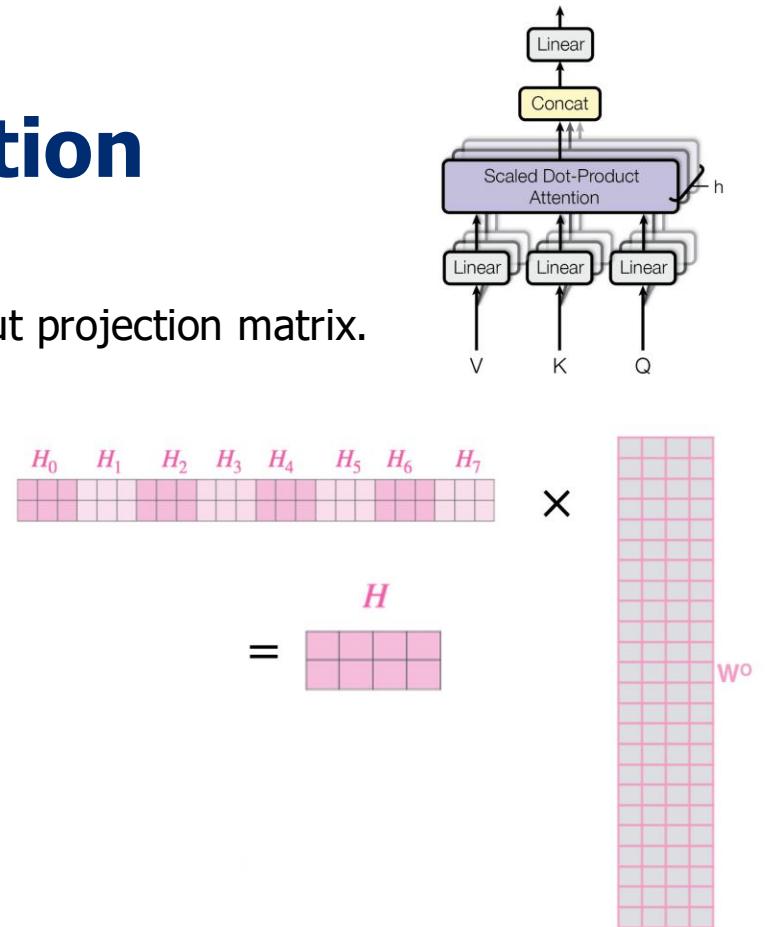
- Multiple parallel attention layers is quite common.
 - Each attention layer has its own parameters.
 - Concatenate the results and run them through a linear projection.



Multi-Headed Self-Attention

- Just concatenate all the heads and apply an output projection matrix.

$$\text{MultiHeadedAttention}(x) = \text{Concat}(\text{head}_1, \dots, \text{head}_h)W^0$$
$$\text{head}_i = \text{Attention}(xW_i^q, xW_i^k, xW_i^v)$$



- In practice, we can use a reduced dimension for each head:

$$W_i^q, W_i^k, W_i^v \in \mathbb{R}^{d/h}$$

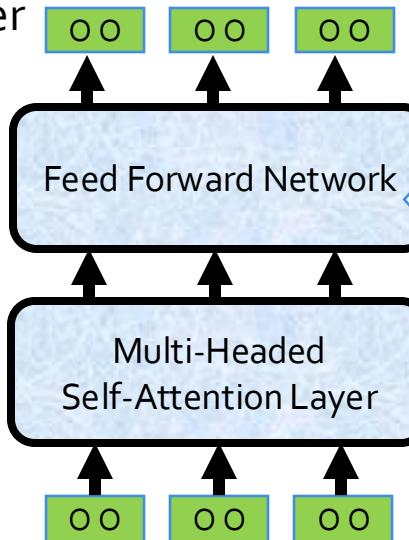
d = hidden dimension

h = # of heads

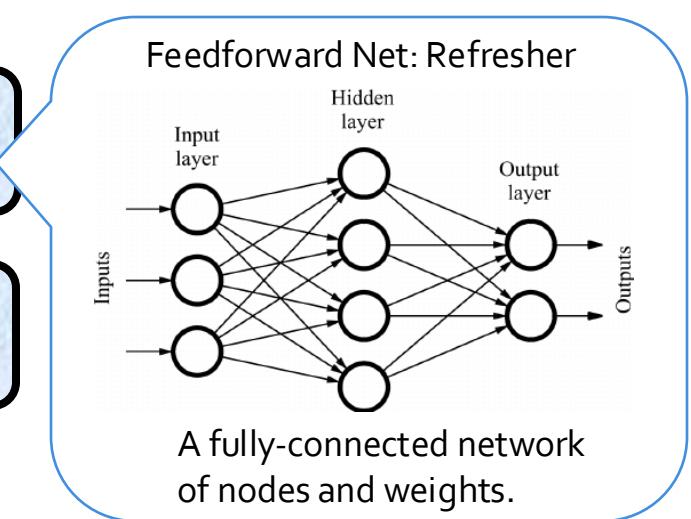
- The total computational cost is similar to that of a single-head attention with full dimensionality.

Combine with FFN

- Add a **feed-forward network** on top it to add more expressivity.
 - This allows the model to apply another transformation to the contextual representations (or “post-process” them).
 - Usually, the dimensionality of the hidden feedforward layer is 2-8 times larger than the input dimension.

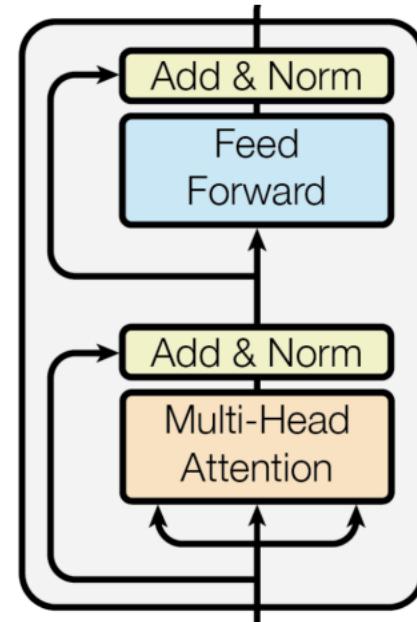


$$\text{FFN}(\mathbf{x}) = f(cW_1 + b_1)W_2 + b_2$$



How Do We Prevent Vanishing Gradients?

- Residual connections let the model “skip” layers
 - These connections are particularly useful for training deep networks
- Use layer normalization to stabilize the network and allow for proper gradient flow



Putting it Together: Self-Attention Block

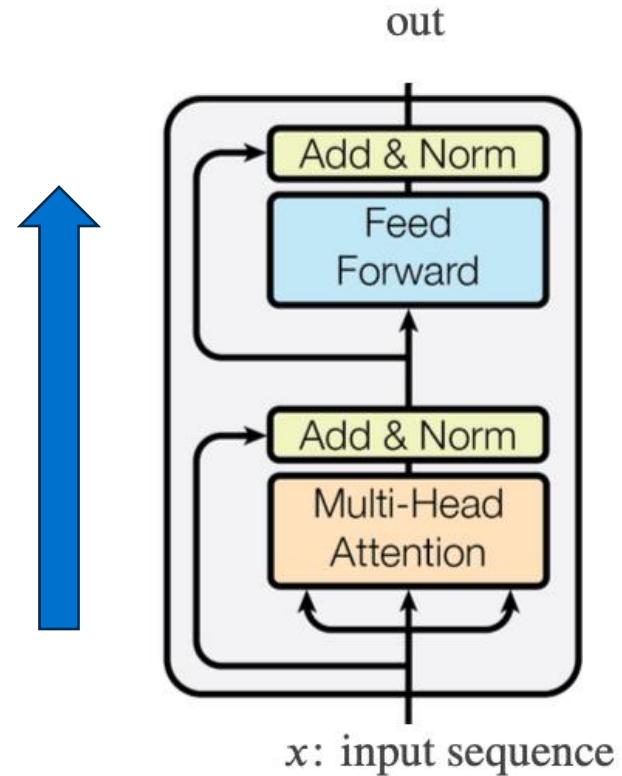
Given input x :

$$\text{out} = \text{LN}(\tilde{\mathbf{c}} + \mathbf{c}')$$

$$\tilde{\mathbf{c}} = \text{FFN}(\mathbf{c}') = f(\mathbf{c}'W_1 + b_1)W_2 + b_2$$

$$\mathbf{c}' = \text{LN}(\mathbf{c} + x)$$

$$\mathbf{c} = \text{MultiHeadedAttention}(x)$$



Summary: Self-Attention Block

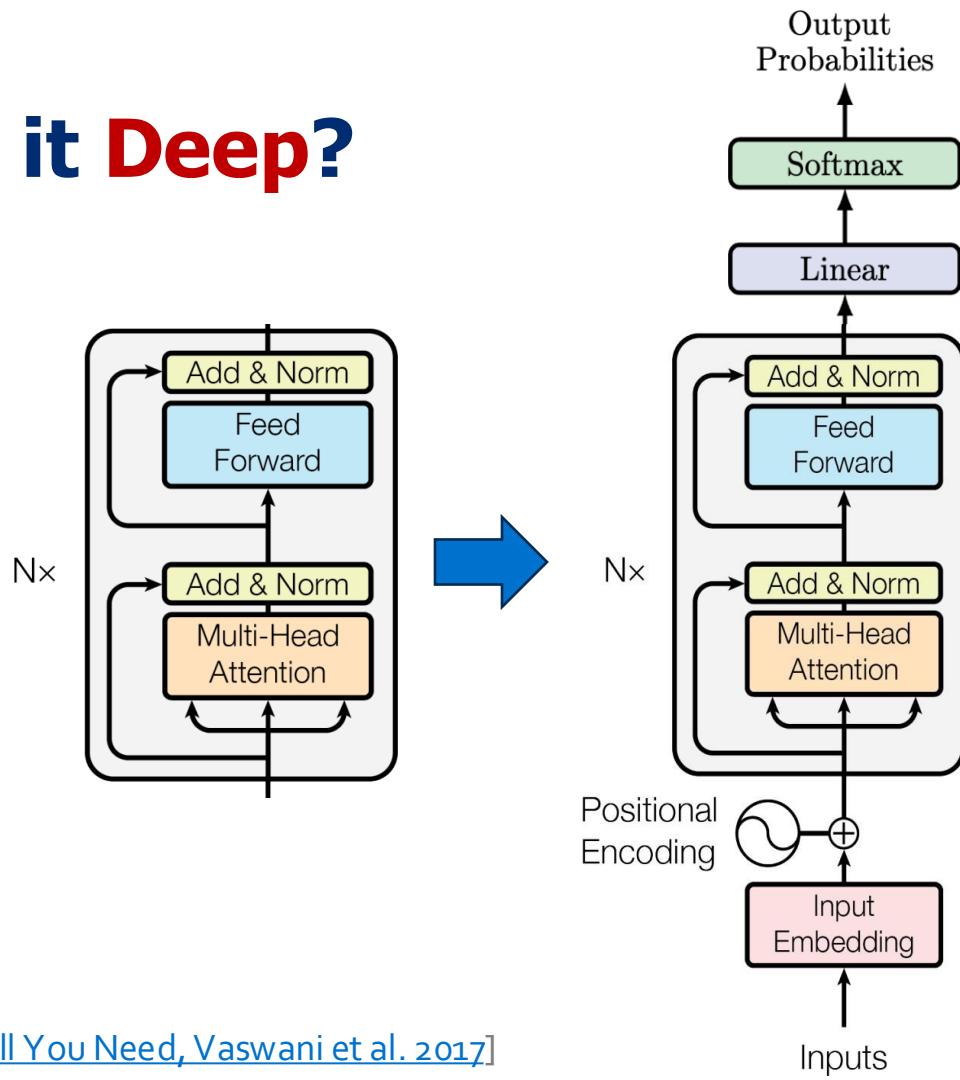
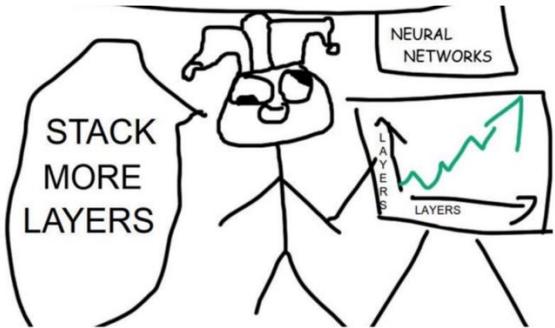
- **Self-Attention:** A critical building block of modern language models.
 - The idea is to compose meanings of words weighted according some similarity notion.
- **Next:** We will combine self-attention blocks to build various architectures known as Transformer.



Transformer

How Do We Make it Deep?

- Stack more layers!



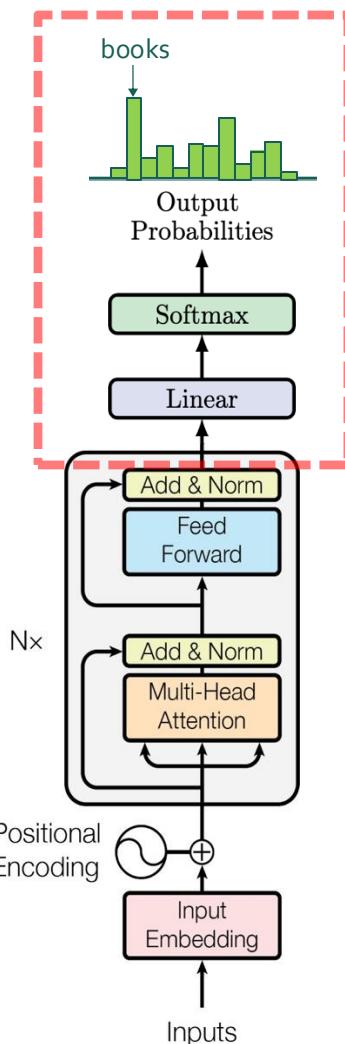
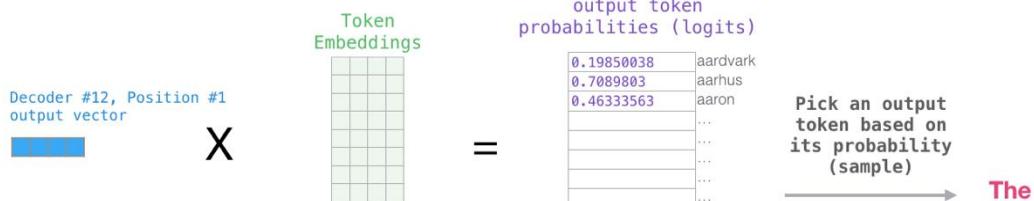
From Representations to Prediction

- To perform prediction, add a classification head on top of the final layer of the transformer.
- This can be per token (Language modeling)
- Or can be for the entire sequence (only one token)

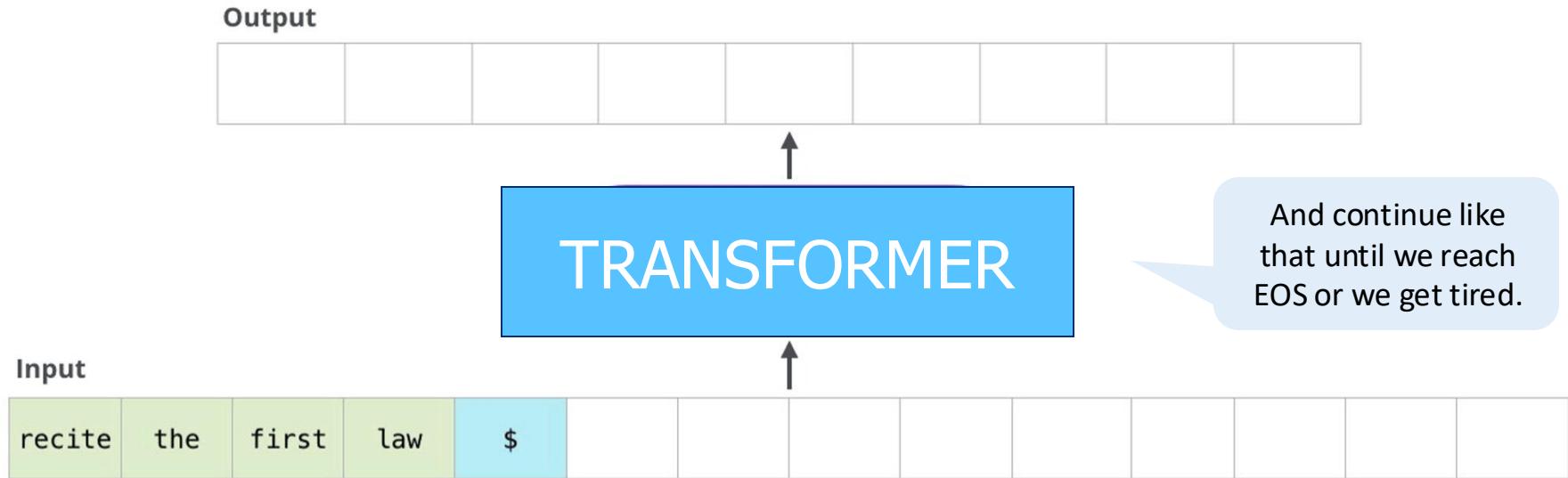
$\text{out} \in \mathbb{R}^{S \times d}$ (S: Sequence length)

$\text{logits} = \text{Linear}_{(d, V)}(\text{out}) = f(\text{out} \cdot W_V) \in \mathbb{R}^{S \times V}$

$\text{probabilities} = \text{softmax}(\text{logits}) \in \mathbb{R}^{S \times V}$



Transformer-based Language Modeling



Training a Transformer Language Model

- **Goal:** Train a Transformer for language modeling (i.e., predicting the next word).
- **Approach:** Train it so that each position is predictor of the next (right) token.
 - We just shift the input to right by one, and use as labels

(gold output) $Y = \text{cat sat on the mat } </s>$



EOS special token

```
X = text[:, :-1]  
Y = text[:, 1:]
```

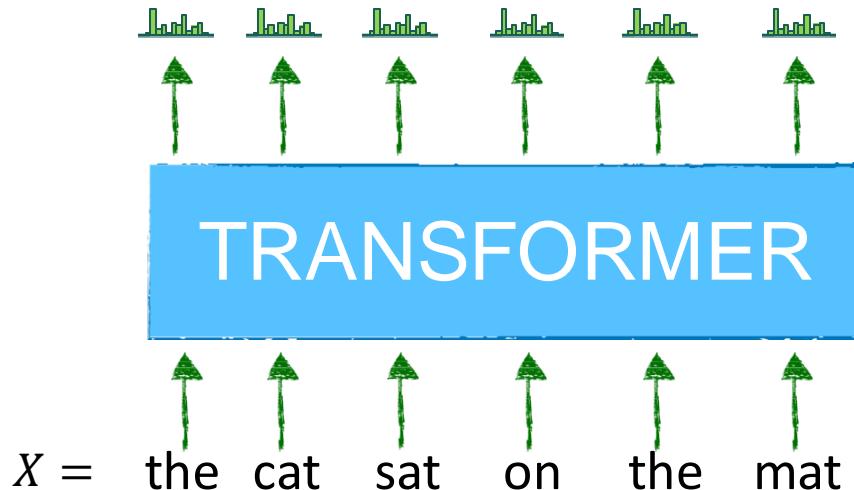
$X = \text{the cat sat on the mat}$

[Slide credit: Arman Cohan]

Training a Transformer Language Model

- For each position, compute their corresponding **distribution** over the whole vocab.

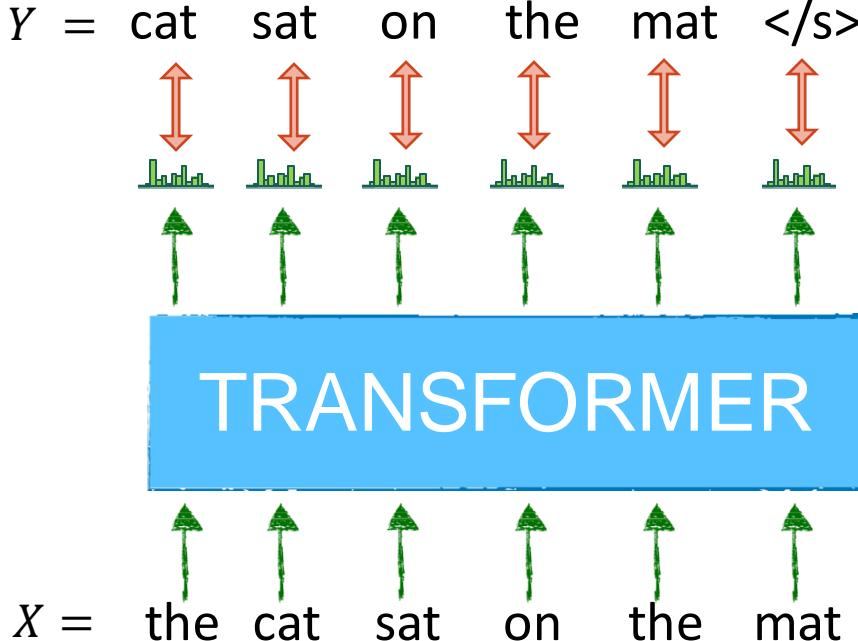
(gold output) $Y = \text{cat sat on the mat } </s>$



Training a Transformer Language Model

- For each position, compute the **loss** between the distribution and the gold output label.

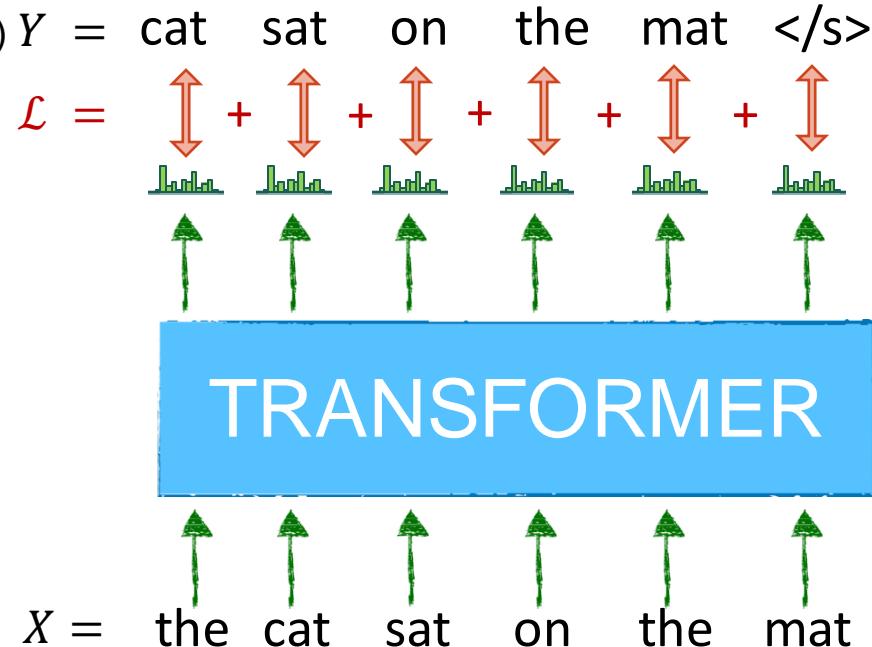
(gold output) $Y = \text{cat sat on the mat } </s>$



Training a Transformer Language Model

- Sum the position-wise loss values to obtain a **global loss**.

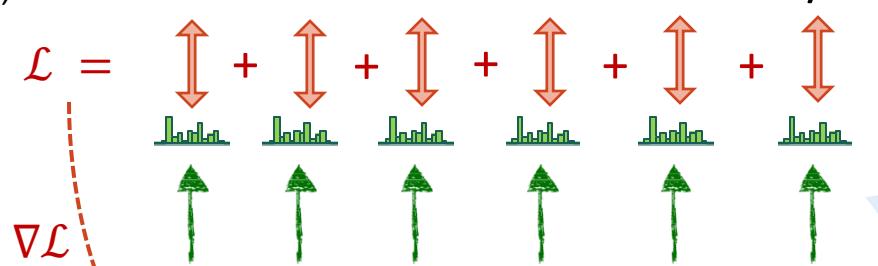
(gold output) $Y = \text{cat sat on the mat } </s>$



Training a Transformer Language Model

- Using this loss, do **Backprop** and **update** the Transformer parameters.

(gold output) $Y = \text{cat sat on the mat } </s>$

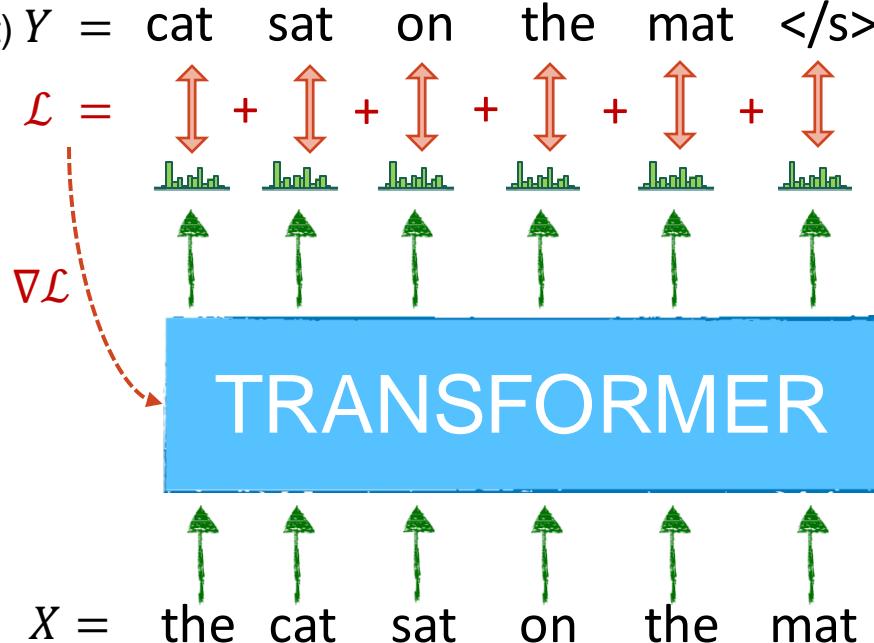


Well, this is not quite right 😊
...
what is the problem with this?

Training a Transformer Language Model

- The model would solve the task by **copying** the next token to output (data leakage).
 - Does **not** learn anything useful

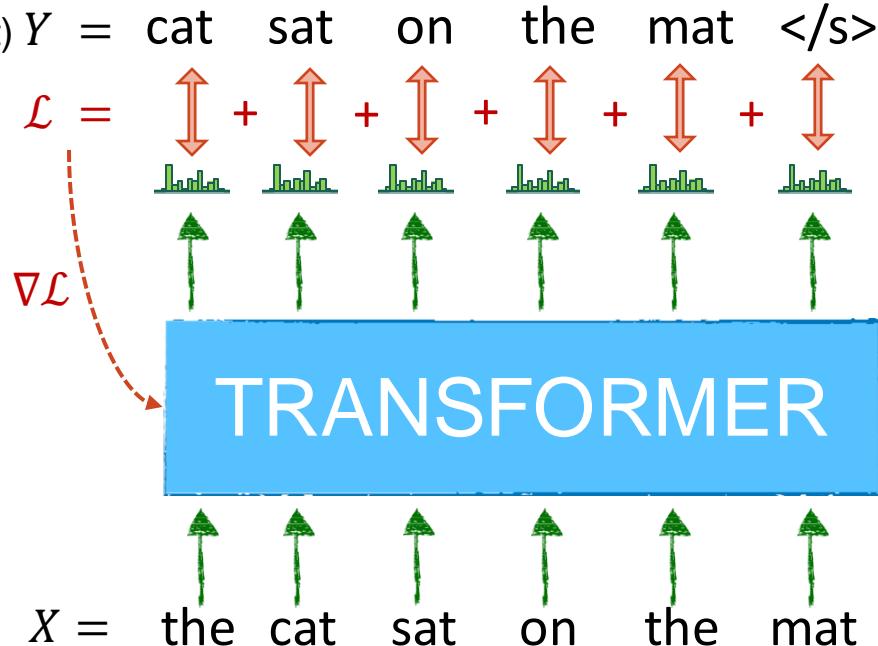
(gold output) $Y = \text{cat sat on the mat } </s>$



Training a Transformer Language Model

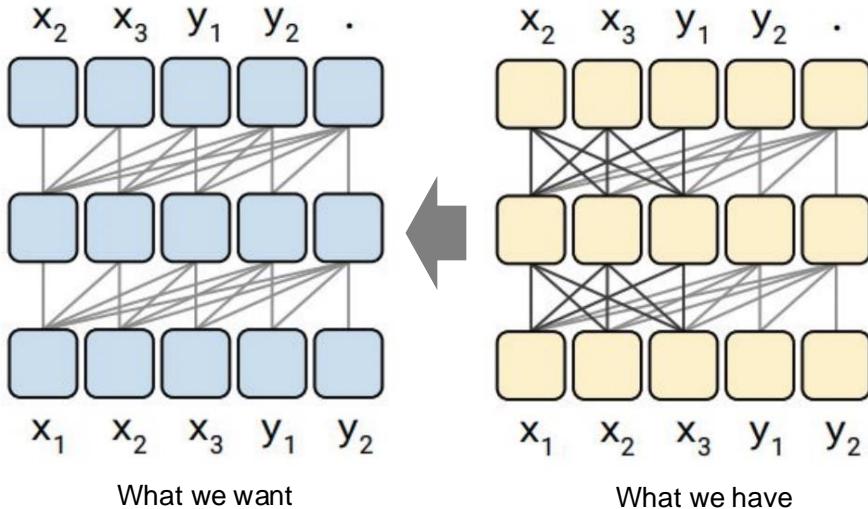
- We need to **prevent information leakage** from future tokens! How?

(gold output) $Y = \text{cat sat on the mat } </s>$

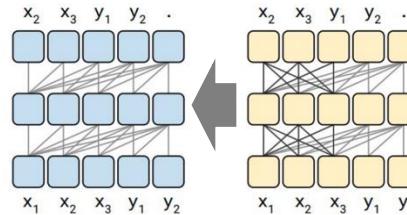


Attention mask

Attention raw scores												
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11



Attention mask



Attention raw scores

	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

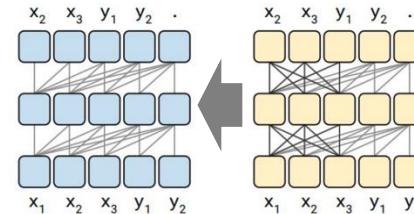
Attention mask



Attention mask

Attention raw scores												
0	-0.08	1.24	0.69	-0.98	1.43	-0.6	0.7	0.16	0.93	1.28	-1.61	-1.1
1	-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	-1.01	1.01
2	0.86	1.19	1.59	0.86	-0.13	-0.15	-2.13	-0.98	-0.87	-1.72	1.87	-0.72
3	0.12	-0.03	-0.02	0.88	-0.46	-0.7	0.54	-0.42	-1.89	-0.38	0.04	-0.84
4	0.51	0.17	0.13	-1.64	0.24	-0.02	1.68	-0.36	0.64	0.36	0.27	0.66
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-0.31	1.54	1.66	1.14	0.58	-1.44
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-0.96	-0.17	-0.9	-1.57	0.22
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-0.7	-0.04	1.54	0.81
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	0.3	0.57	0.74
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	0.5	-0.3
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-0.43
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11

X



Attention mask

0	1	2	3	4	5	6	7	8	9	10	11
0	1.0	-inf									
1	1.0	1.0	-inf								
2	1.0	1.0	1.0	-inf							
3	1.0	1.0	1.0	1.0	-inf						
4	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf	-inf
5	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf	-inf
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf	-inf
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf	-inf
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf	-inf
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	-inf
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0



Note matrix multiplication is quite fast in GPUs.

Presented by: Arman Cohan

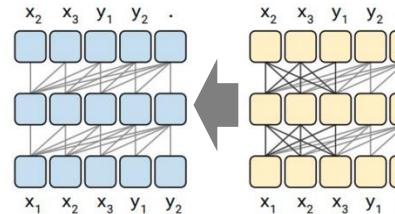
Attention mask

Attention raw scores												
	1	2	3	4	5	6	7	8	9	10	11	12
1	-0.09	1.34	0.69	0.98	3.43	-0.1	0.7	0.16	0.93	1.28	1.81	-1.13
2	-0.09	-0.7	0.06	0.25	0.26	0.26	0.18	0.78	-0.21	1.01	1.01	1.01
3	-0.09	1.19	1.09	0.96	-0.15	0.15	-0.11	-0.06	-0.07	1.73	1.67	-0.23
4	0.12	-0.03	-0.02	0.88	-0.46	-0.1	0.54	-0.42	1.86	-0.38	0.54	-0.84
5	0.51	0.17	0.51	0.34	0.24	-0.02	1.68	-0.36	0.84	0.36	0.27	0.86
6	0.24	-1.64	0.43	0.78	0.96	1.2	-0.31	1.54	1.88	1.14	0.58	-1.44
7	0.26	-0.1	0.63	0.72	-0.36	1.84	0.47	0.96	-0.17	-0.5	0.58	0.22
8	0.55	0.81	0.71	1.7	-0.8	1.16	-0.32	1.78	-0.7	-0.54	1.54	0.81
9	0.74	0.76	-0.48	-0.68	-0.38	0.13	0.13	1.25	1.37	1.84	0.3	0.57
10	0.87	0.91	0.15	0.36	-0.81	0.11	1.14	-0.51	1.08	1.87	0.5	-0.37
11	0.52	0.74	0.44	0.71	1.19	0.89	0.29	2.06	0.31	-0.28	1.51	0.11
12	1	2	3	4	5	6	7	8	9	10	11	12

X

=

Raw attention scores												
	1	2	3	4	5	6	7	8	9	10	11	12
1	1.11	inf										
2	0.1	1.0	inf									
3	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
12	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0



Masked attention raw scores

	0	1	2	3	4	5	6	7	8	9	10	11	12
0	-0.08	-inf	-inf	-inf	-inf	-inf	-inf						
1	-0.09	-0.0	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
2	0.86	1.19	1.59	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
3	0.12	-0.03	-0.02	0.88	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
4	0.51	0.17	0.13	-1.64	0.24	-inf	-inf	-inf	-inf	-inf	-inf	-inf	-inf
5	0.24	-1.44	0.43	0.74	0.96	-1.21	-inf	-inf	-inf	-inf	-inf	-inf	-inf
6	0.26	-0.1	0.93	0.72	-0.38	1.65	0.47	-inf	-inf	-inf	-inf	-inf	-inf
7	-0.55	0.81	0.71	1.7	-0.8	-1.14	-0.32	1.78	-inf	-inf	-inf	-inf	-inf
8	0.74	-0.76	-0.44	-0.08	-1.38	-0.13	1.25	-1.37	1.84	-inf	-inf	-inf	-inf
9	-0.97	-0.91	0.15	0.35	-0.81	0.11	1.14	-1.52	1.06	1.87	-inf	-inf	-inf
10	1.56	0.9	0.39	1.46	1.44	-1.05	0.9	-0.73	0.36	-0.67	-0.62	-inf	-inf
11	0.32	0.74	0.44	-0.1	1.19	0.83	0.29	2.06	0.51	-0.26	1.51	0.11	-inf
12	1	2	3	4	5	6	7	8	9	10	11	12	

Slide credit: Arman Cohan

Attention mask

The effect is more than just pruning out some of the wirings in self-attention block.

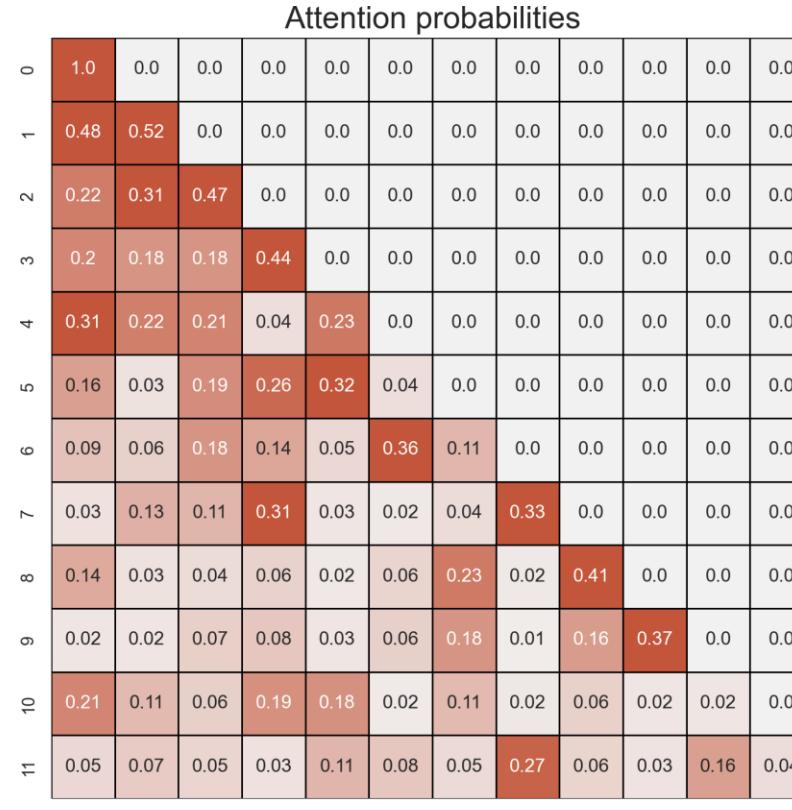
Attention raw scores											
-0.09	1.24	0.69	0.38	3.43	-0.1	0.7	1.16	0.93	1.28	1.81	-1.13
-0.09	-0.0	-0.7	0.06	0.25	0.23	0.26	0.18	0.78	-0.21	1.01	1.01
-0.06	1.19	1.59	0.98	-0.15	0.15	1.11	0.96	-0.97	1.22	1.67	-0.72
-0.12	-0.03	-0.42	0.88	-0.46	-0.1	0.54	-0.42	1.86	-0.38	0.94	-0.84
-0.51	0.17	0.55	0.34	0.24	0.02	1.68	0.36	0.04	0.37	0.66	-0.01
-0.24	-1.64	0.43	0.78	0.96	1.2	-0.31	1.54	1.88	1.14	0.58	-1.44
-0.26	-0.1	0.63	0.72	0.36	1.84	0.47	1.96	-0.17	-0.5	0.67	0.22
-0.55	0.81	0.71	1.7	-0.18	0.32	1.78	-0.7	-0.94	1.54	0.81	-0.74
-0.74	-0.48	-0.48	0.08	-0.33	-0.13	1.25	1.37	1.84	0.3	0.57	0.74
-0.87	0.91	0.15	0.30	-0.81	0.11	1.14	-0.50	1.08	1.87	0.5	-0.37
-0.52	0.74	0.44	0.1	1.19	0.89	0.29	2.06	0.31	-0.28	1.51	0.11

X

Raw attention scores											
0	1.0	ad									
1	1.0	1.2	ad								
2	1.0	1.2	1.0	1.0	ad						
3	1.0	1.0	1.0	1.0	1.0	1.0	ad	ad	ad	ad	ad
4	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	ad	ad	ad
5	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	ad
6	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
7	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
8	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
9	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
10	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0
11	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0	1.0

Masked attention raw scores											
-0.08	ad	ad	ad	ad	ad	ad	ad	ad	ad	ad	ad
-0.09	-0.0	ad	ad	ad	ad	ad	ad	ad	ad	ad	ad
-0.06	1.19	1.59	ad	ad	ad	ad	ad	ad	ad	ad	ad
-0.12	-0.03	0.88	ad	ad	ad	ad	ad	ad	ad	ad	ad
-0.51	0.17	0.33	-1.64	0.24	ad	ad	ad	ad	ad	ad	ad
-0.24	1.64	1.43	0.74	0.68	1.21	ad	ad	ad	ad	ad	ad
-0.26	-0.1	0.93	0.72	-0.38	1.65	0.67	ad	ad	ad	ad	ad
-0.56	0.81	0.71	1.7	-0.8	1.14	-0.26	1.78	ad	ad	ad	ad
-0.74	-0.76	-0.44	-0.36	-0.36	1.13	1.37	1.84	ad	ad	ad	ad
-0.87	-0.91	0.15	0.35	-0.81	0.11	1.14	1.52	1.06	1.87	ad	ad
-1.56	0.8	0.39	1.40	1.44	1.00	0.8	-0.73	1.36	-0.67	-0.62	ad
-0.32	0.74	0.44	-0.1	1.19	0.89	0.29	2.06	0.51	-0.28	1.51	0.11

softmax

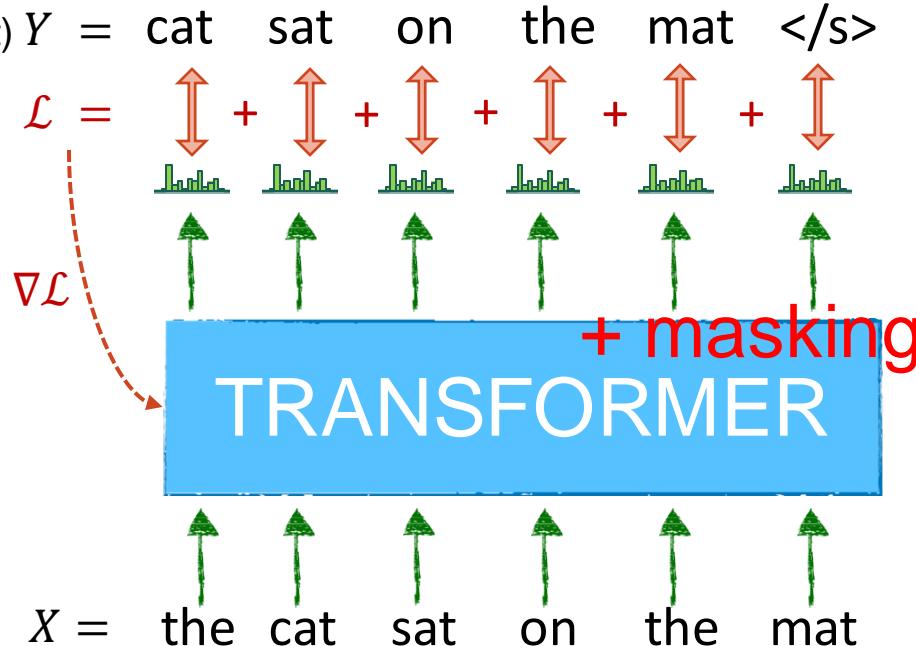


Slide credit: Alman Cohan

Training a Transformer Language Model

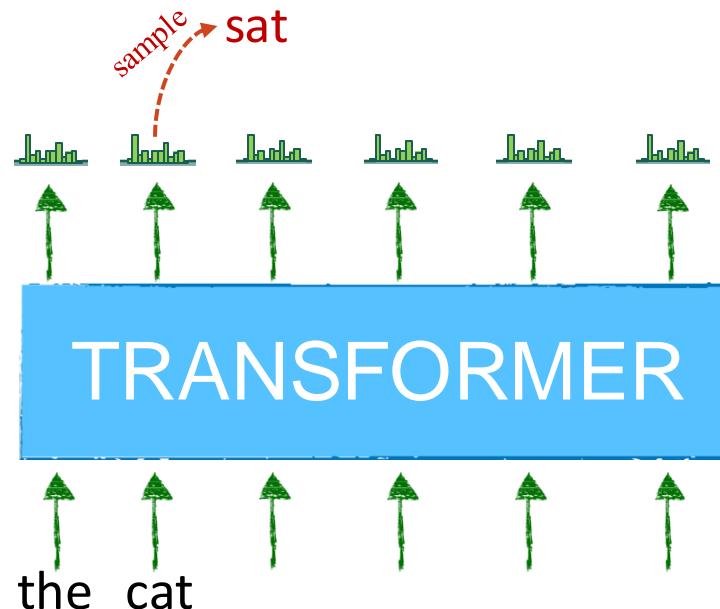
- We need to **prevent information leakage** from future tokens! How?

(gold output) $Y = \text{cat sat on the mat } </s>$



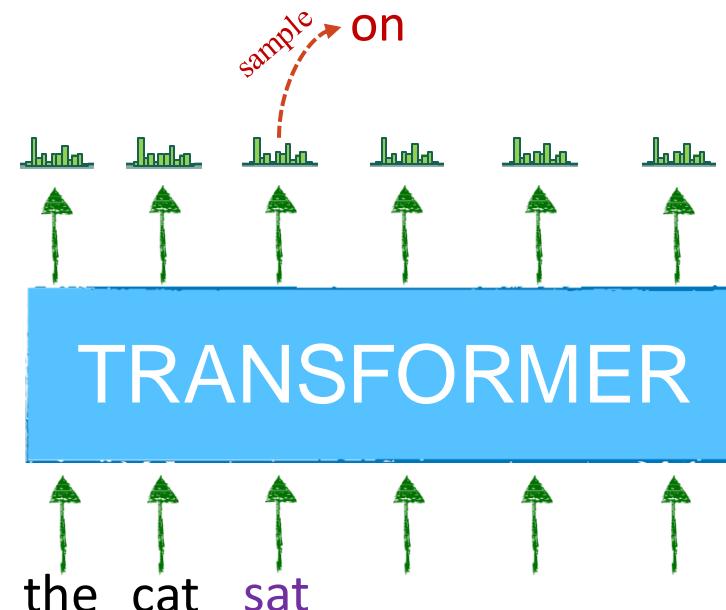
How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly



How to use the model to generate text?

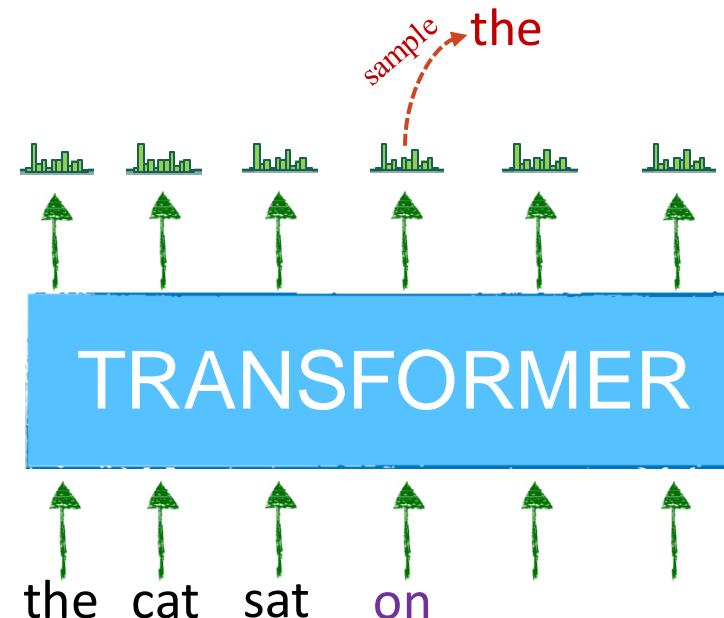
- Use the output of previous step as input to the next step repeatedly



The probabilities get revised upon adding a new token to the input.

How to use the model to generate text?

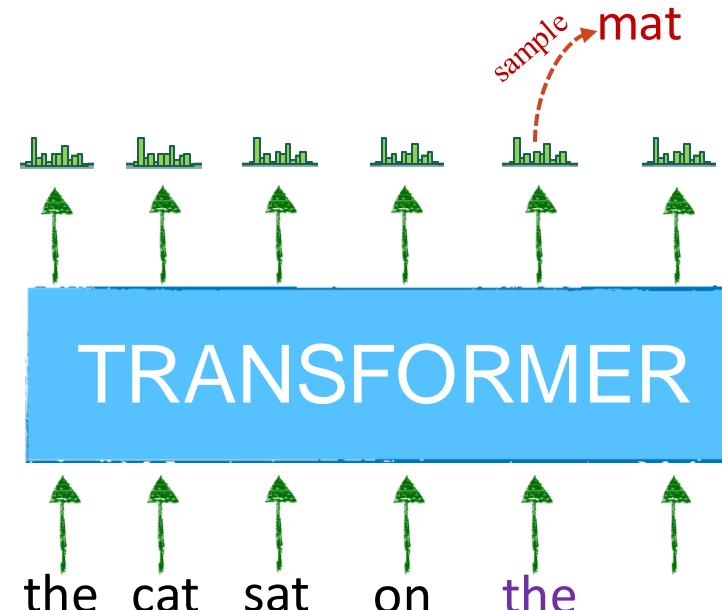
- Use the output of previous step as input to the next step repeatedly



The probabilities get revised upon adding a new token to the input.

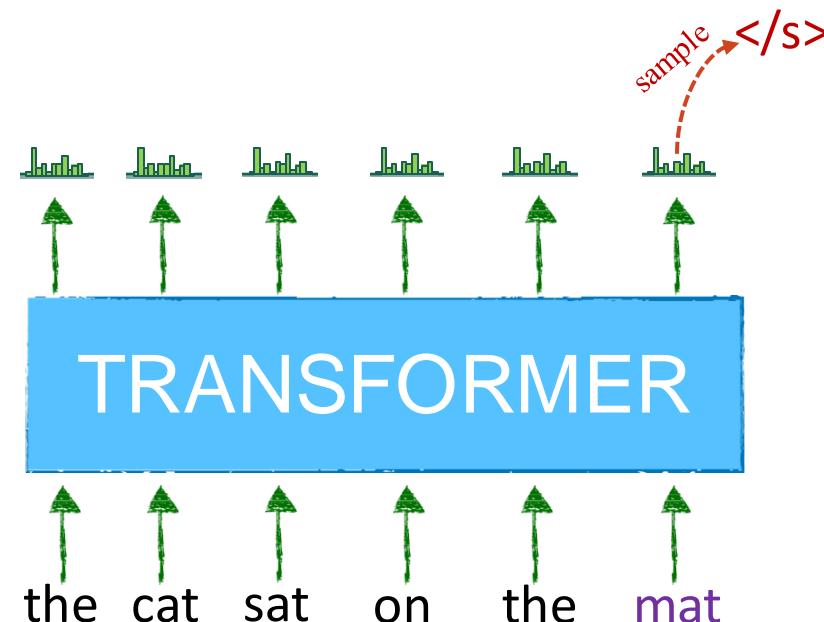
How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly



How to use the model to generate text?

- Use the output of previous step as input to the next step repeatedly





How efficient is this decoding?



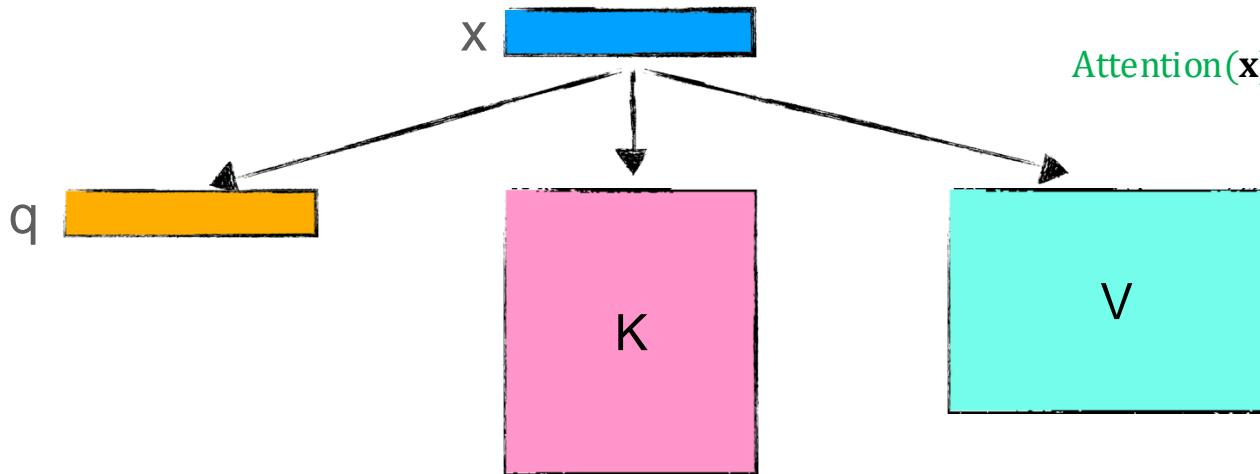
Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$

$$K = \mathbf{W}^k \mathbf{x}$$

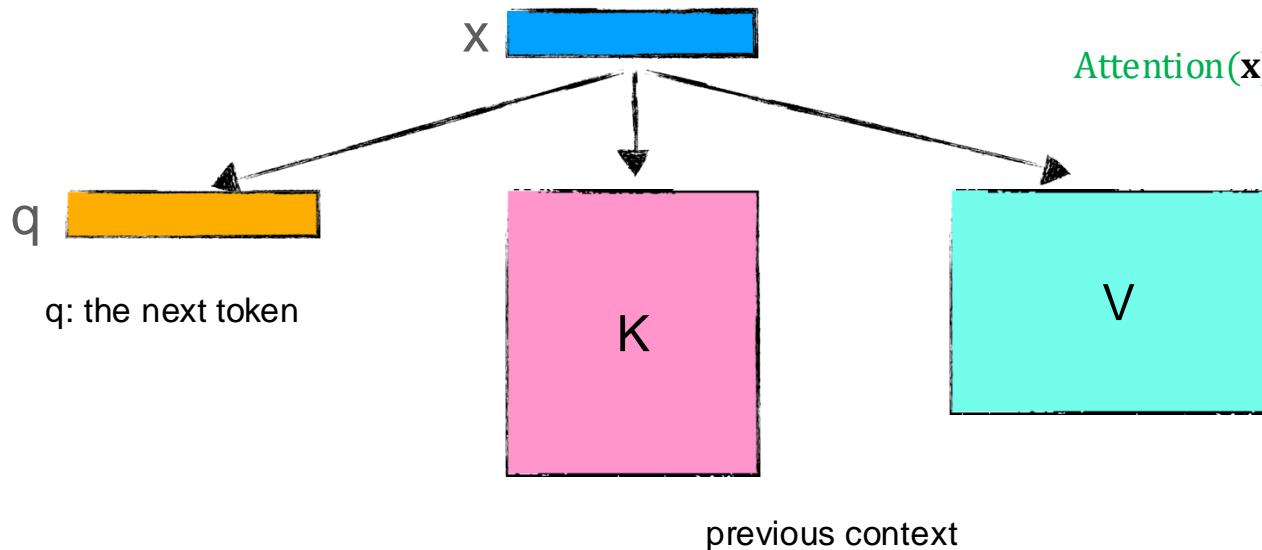
$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient



$$\begin{aligned} Q &= \mathbf{W}^q \mathbf{x} \\ K &= \mathbf{W}^k \mathbf{x} \\ V &= \mathbf{W}^v \mathbf{x} \end{aligned}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

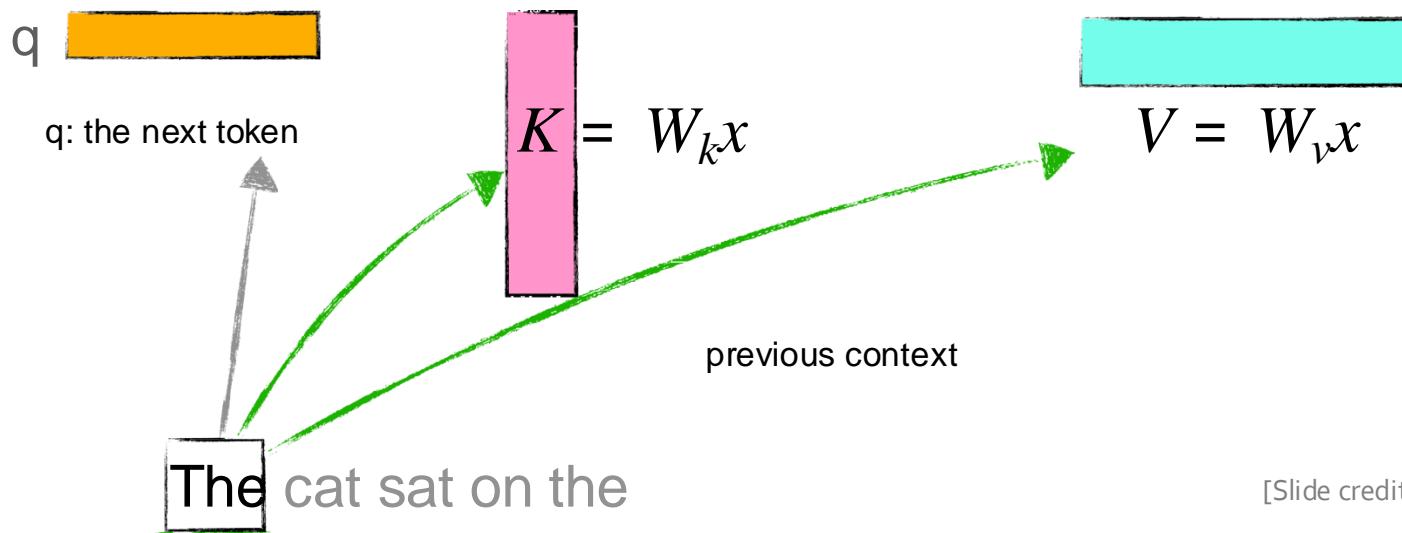
Making decoding more efficient

$$Q = W^q \mathbf{x}$$

$$K = W^k \mathbf{x}$$

$$V = W^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

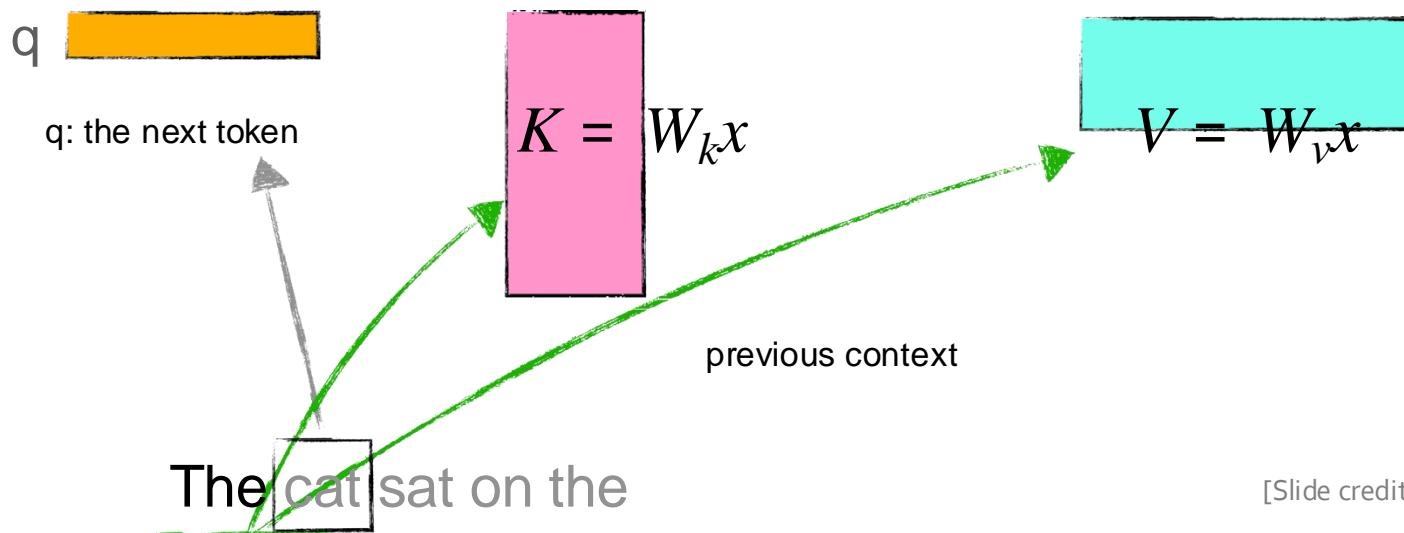
Making decoding more efficient

$$Q = W^q x$$

$$K = W^k x$$

$$V = W^v x$$

$$\text{Attention}(x) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

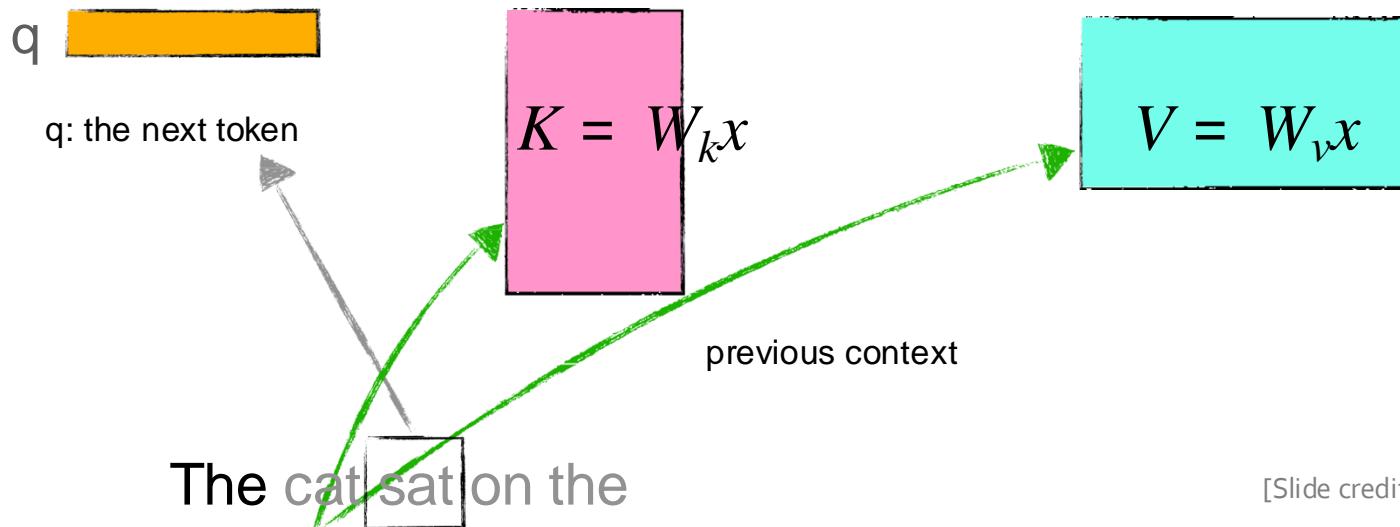
Making decoding more efficient

$$Q = W^q x$$

$$K = W^k x$$

$$V = W^v x$$

$$\text{Attention}(x) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

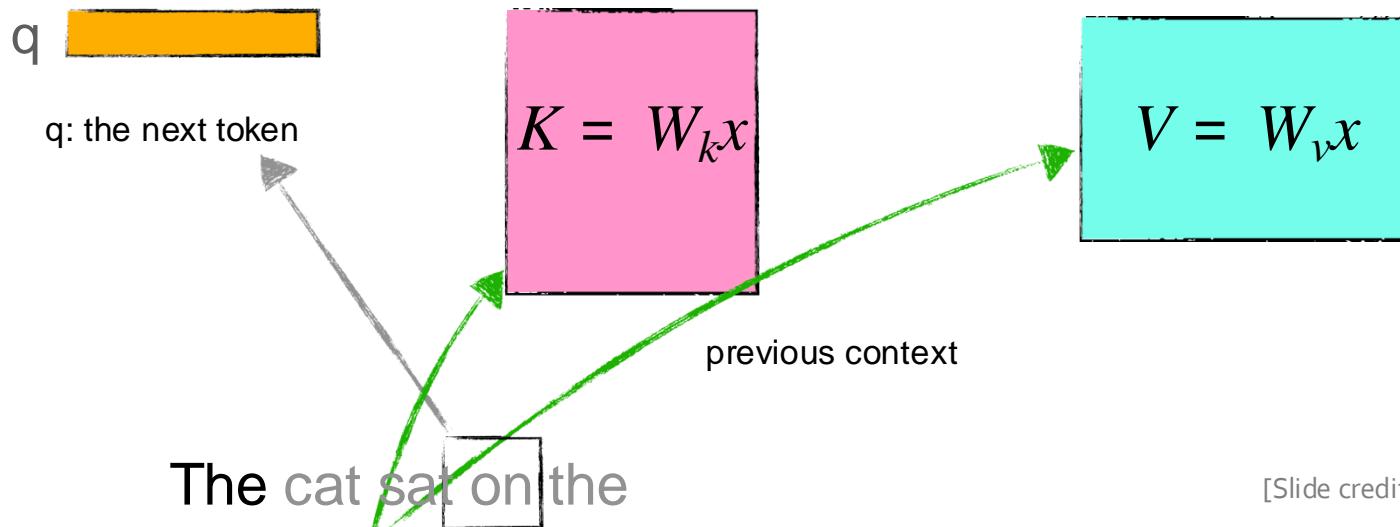
Making decoding more efficient

$$Q = \mathbf{W}^q \mathbf{x}$$

$$K = \mathbf{W}^k \mathbf{x}$$

$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

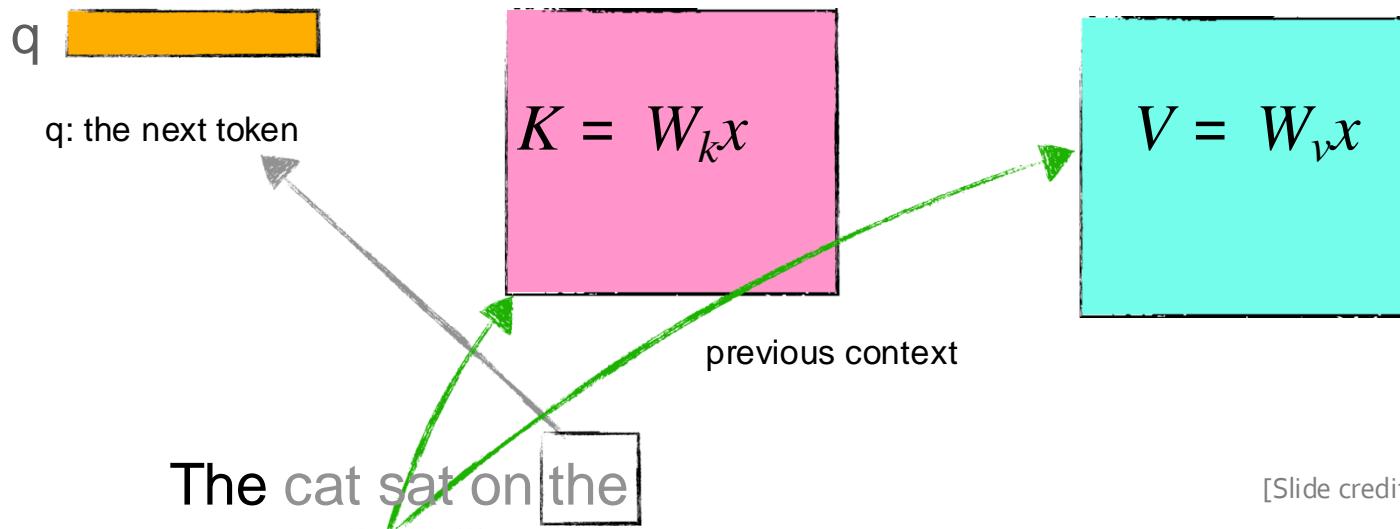
Making decoding more efficient

$$Q = W^q \mathbf{x}$$

$$K = W^k \mathbf{x}$$

$$V = W^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$



[Slide credit: Arman Cohan]

Making decoding more efficient

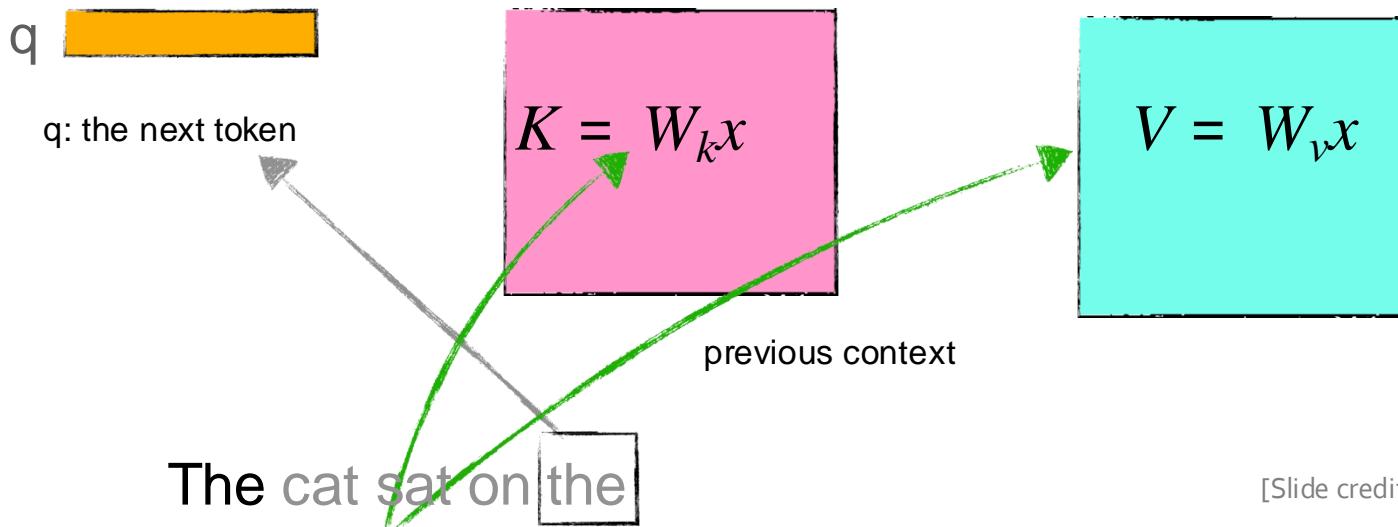
- We are computing the Keys and Values many times!
 - Let's reduce redundancy! 😊

$$Q = \mathbf{W}^q \mathbf{x}$$

$$K = \mathbf{W}^k \mathbf{x}$$

$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$



[Slide credit: Arman Cohan]

Making decoding more efficient

- We are computing the Keys and Values many times!
 - Let's reduce redundancy! 😊

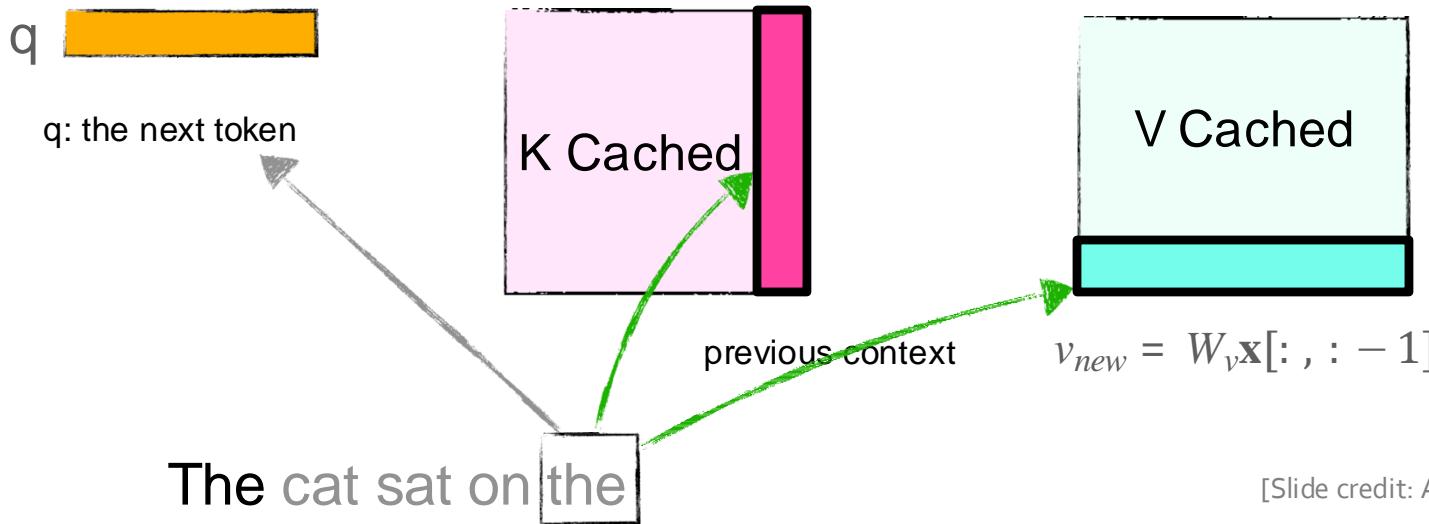
$$Q = \mathbf{W}^q \mathbf{x}$$

$$K = \mathbf{W}^k \mathbf{x}$$

$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

$$k_{new} = W_k \mathbf{x}[:, :, -1]$$



Making decoding more efficient

- **Question:** How much memory does this K, V cache require?

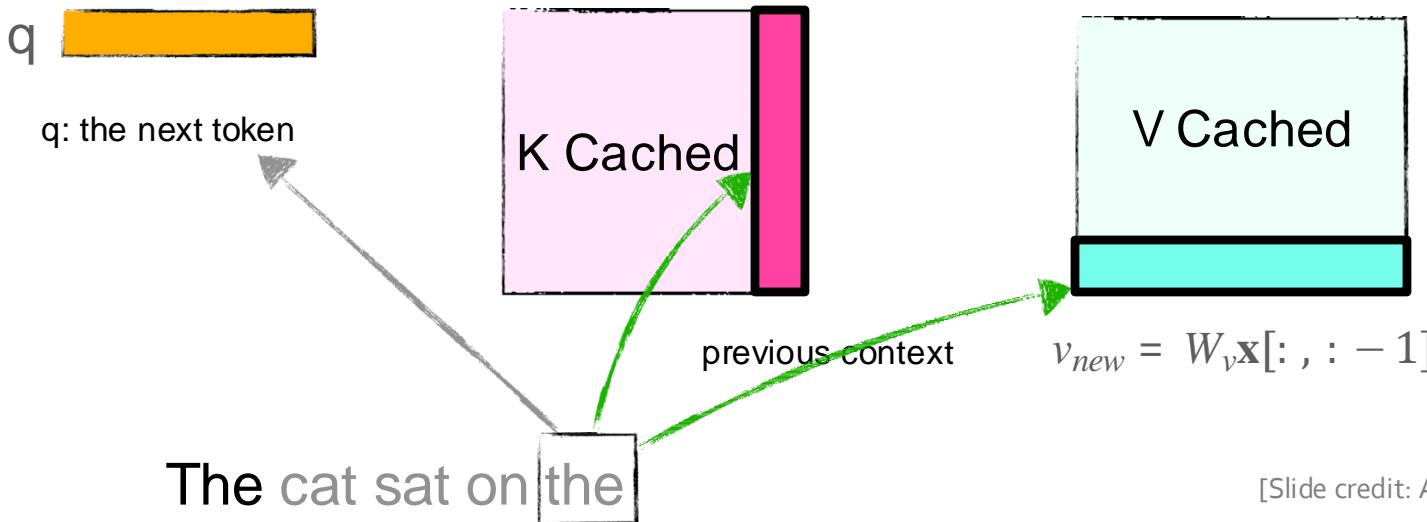
$$Q = \mathbf{W}^q \mathbf{x}$$

$$K = \mathbf{W}^k \mathbf{x}$$

$$V = \mathbf{W}^v \mathbf{x}$$

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\mathbf{Q}\mathbf{K}^T}{\sqrt{d}}\right)\mathbf{V}$$

$$k_{new} = W_k \mathbf{x}[:, : - 1]$$



[Slide credit: Arman Cohan]

Summary

- This is a very generic Transformer!

- **Next:**

- Positional encodings
 - ...

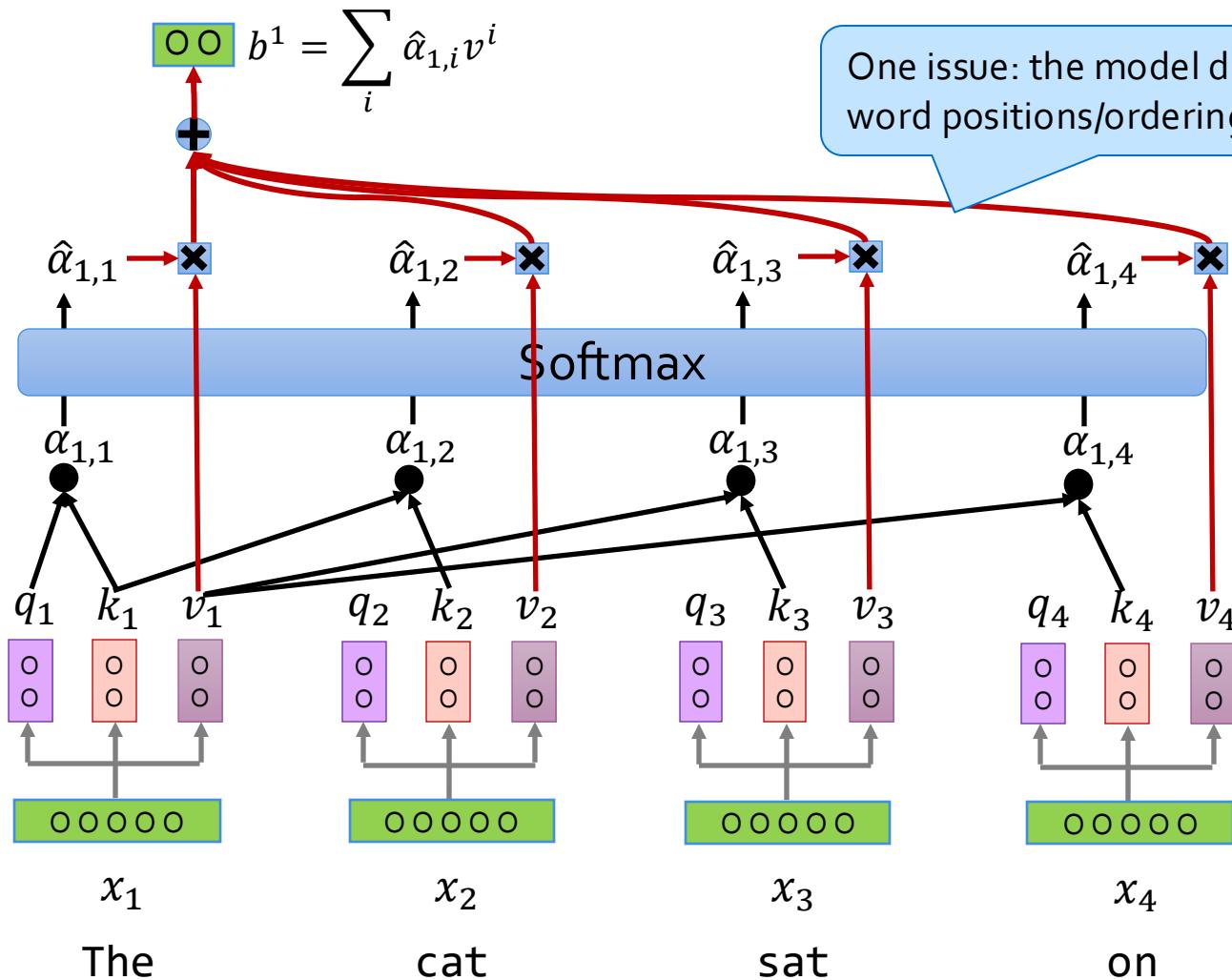


Encoding Positional Information



Why do we need positional encoding?

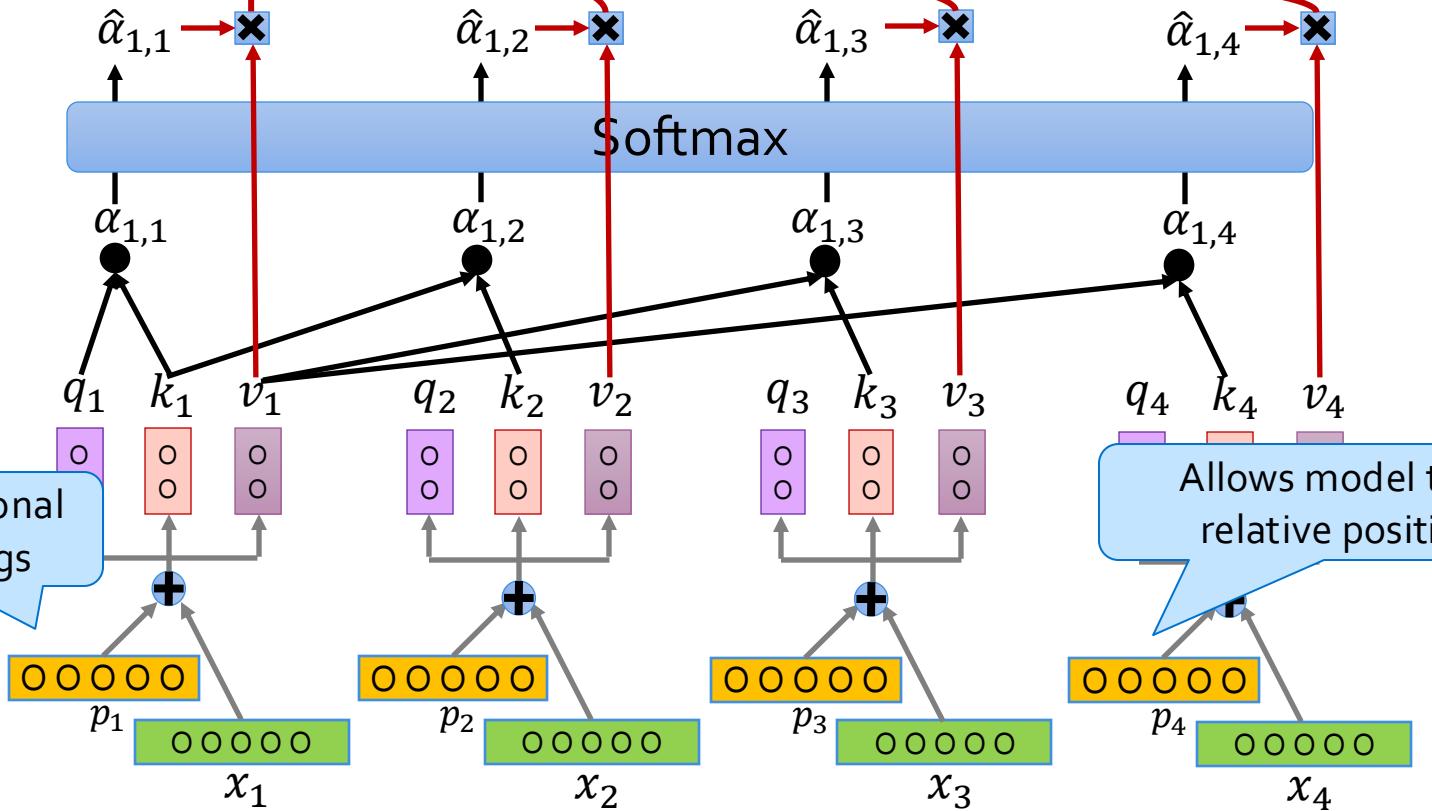




We will discuss various choices for these embedding!

$$\begin{array}{c} \text{OO} \\ \uparrow \end{array} b^1 = \sum_i \hat{\alpha}_{1,i} v^i$$

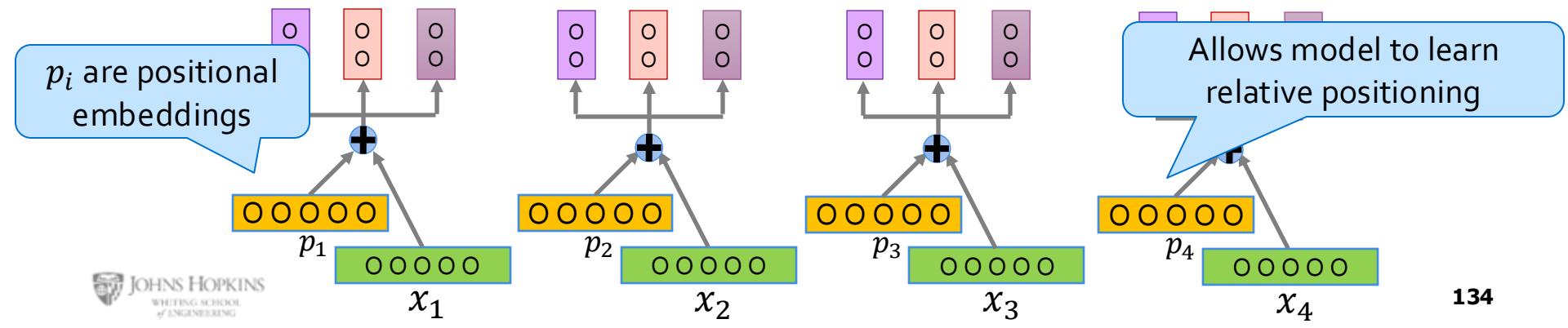
One issue: the model doesn't know word positions/ordering.



p_i are positional embeddings

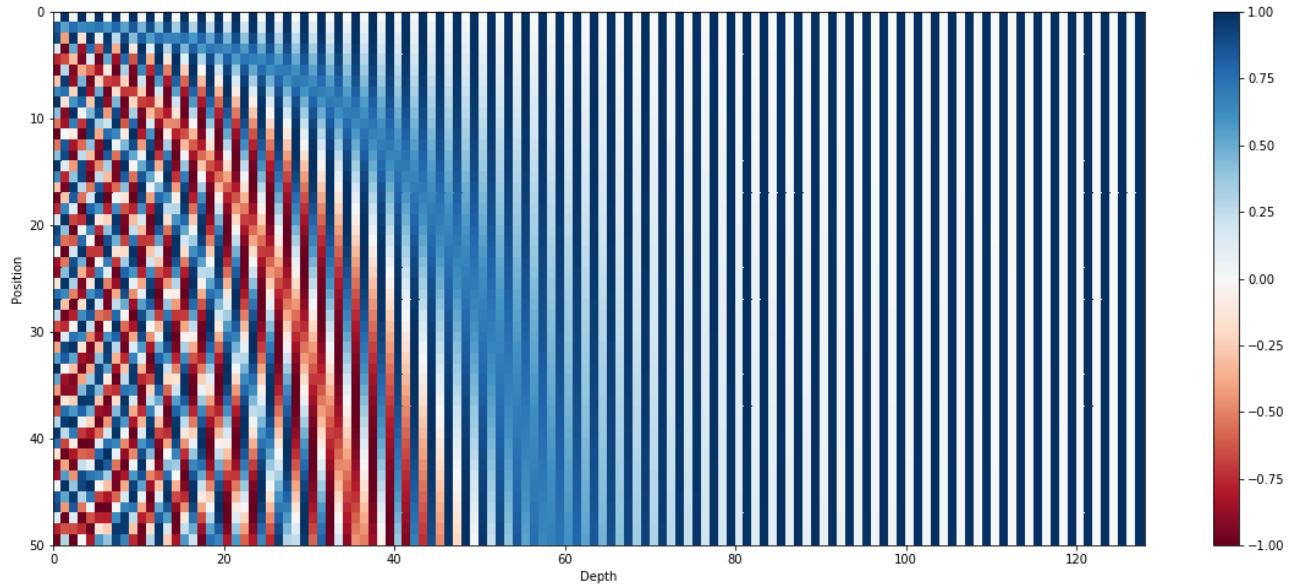
Positional Embeddings: The Flavors

- Absolute encoding: vectors that uniquely encode each position.
- Relative encoding: the positional encoding for each position is determined based on its distance from the other positions it is attending to.

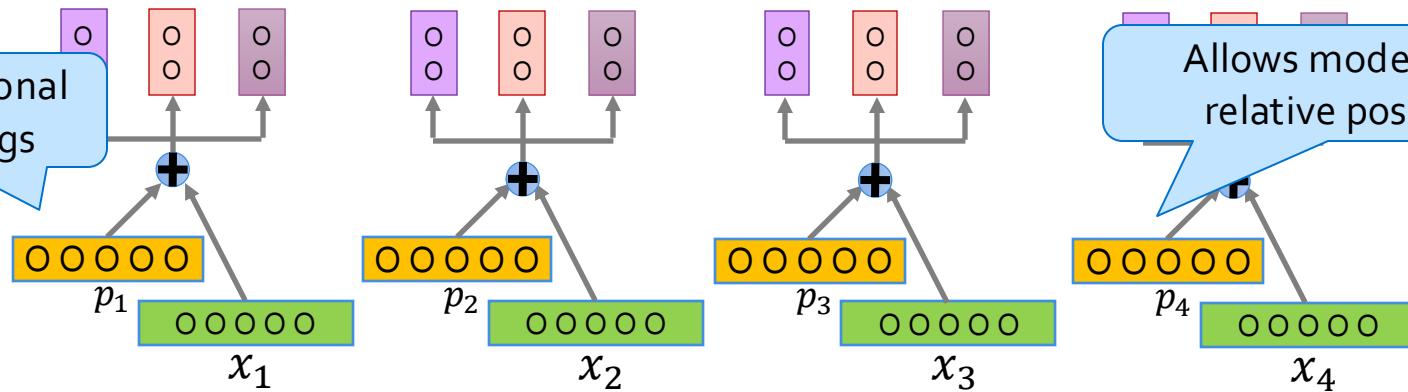


An approach:
Sine/Cosine encoding

$$p_i = \begin{cases} \sin(i/10000^{2*1/d}) \\ \cos(i/10000^{2*1/d}) \\ \vdots \\ \sin(i/10000^{2*\frac{d}{2}/d}) \\ \cos(i/10000^{2*\frac{d}{2}/d}) \end{cases}$$



p_i are positional embeddings



Limits of Absolute Positional Encoding

- We can have fixed positional embeddings for each index training position (e.g., 1, 2, 3, ... 1000).
 - What happens if we get a sequence with 5000 words at test time?
- We want something that can generalize to arbitrary sequence lengths.
 - Approach: encoding the relative positions, for example based on the distance of the tokens in a local window to the current token.

Relative Positional Encoding

- You can rewrite the statement from the previous slide in the following form:

$$QK_{ij} = (W_q[x_i + p_i])^T (W_k[x_j + p_j]) = x_i^T W_q^T W_k x_j + P_{ij}$$

- Note, the values of P_{ij} encode the relative of i and j .
- How should we construct P_{ij} ?

How much attention
should position i should
attend to position j

Relative Positional Encoding

- There have been various choices:
 - T5 models simplify this into learnable relative embeddings \mathbf{P}_{ij} such that:
$$QK_{ij} = \mathbf{x}_i^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{x}_j + \mathbf{P}_{ij}$$
 - DeBERTa learns relative positional embeddings $\tilde{\mathbf{p}}_{i-j}$ such that:
$$QK_{ij} = \mathbf{x}_i^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{x}_j + \mathbf{x}_i^T \mathbf{W}_q^T \mathbf{W}_k \tilde{\mathbf{p}}_{i-j} + \tilde{\mathbf{p}}_{i-j}^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{x}_j$$
 - Transformer-XL learns relative positional embeddings $\tilde{\mathbf{p}}_{i-j}$ and trainable vectors \mathbf{u}, \mathbf{v} s.t.:
$$QK_{ij} = \mathbf{x}_i^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{x}_j + \mathbf{x}_i^T \mathbf{W}_q^T \mathbf{W}_k \tilde{\mathbf{p}}_{i-j} + \mathbf{u}^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{x}_j + \mathbf{v}^T \mathbf{W}_q^T \mathbf{W}_k \tilde{\mathbf{p}}_{i-j}$$
 - ALiBi learns a scalar m such that:
$$QK_{ij} = \mathbf{x}_i^T \mathbf{W}_q^T \mathbf{W}_k \mathbf{x}_j - m |i - j|$$

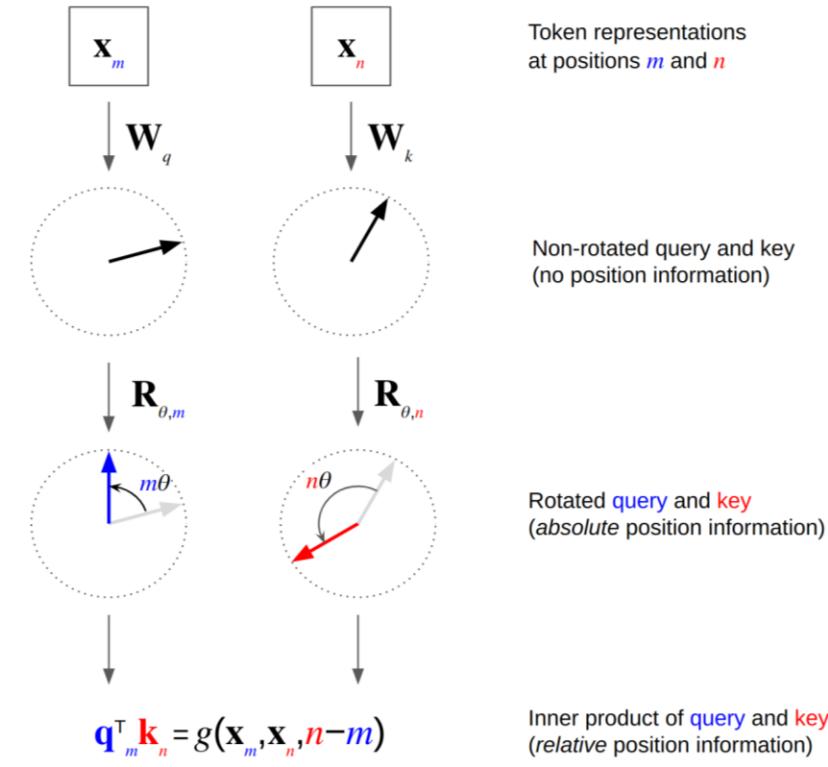
Relative Encoding via Multiplication: Rotary Positional Encoding (RoPE)

- Drop the additive positional encoding and make it multiplicative.

$$\begin{aligned} qk_{mn} &= (R_{\theta,m}W_qx_m)^T(R_{\theta,n}W_kx_n) \\ &= x_m^TW_q^TR_{\theta,m}^TR_{\theta,n}W_kx_n \end{aligned}$$

- θ : the size of rotation
- $R_{\theta,m}$: rotation matrix, rotates a vector it gets multiplied to proportional to θ and the position index m .

- Intuition: **nearby** words have **smaller relative rotation**.



Summary

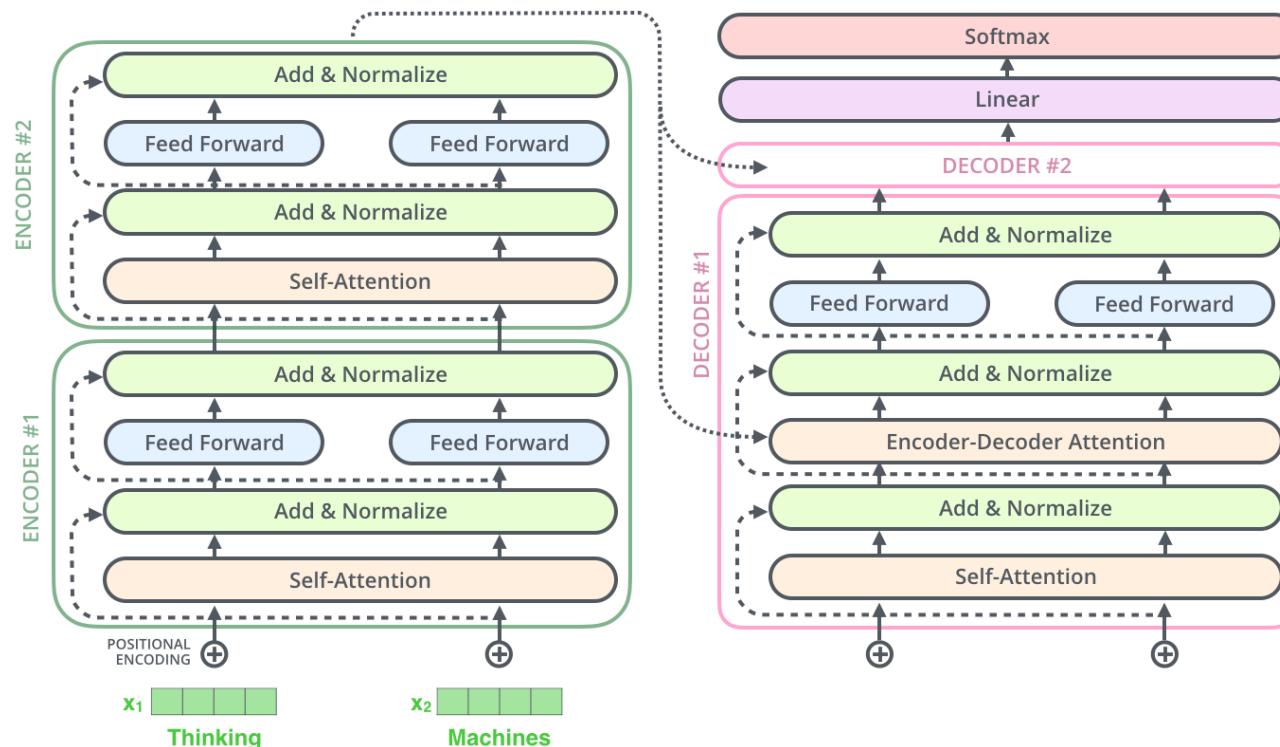
- Encoding positional information in language models is a non-trivial problem.
 - We discussed various proposal: learned, absolute, relative encoding, NoPos, etc.
- This is an important literature related to the length generalization of Transformers.
- This is an active research area and likely to change in the coming years.



Transformer Architectural Variants

Transformer [Vaswani et al. 2017]

- An encoder-decoder architecture built with attention modules.



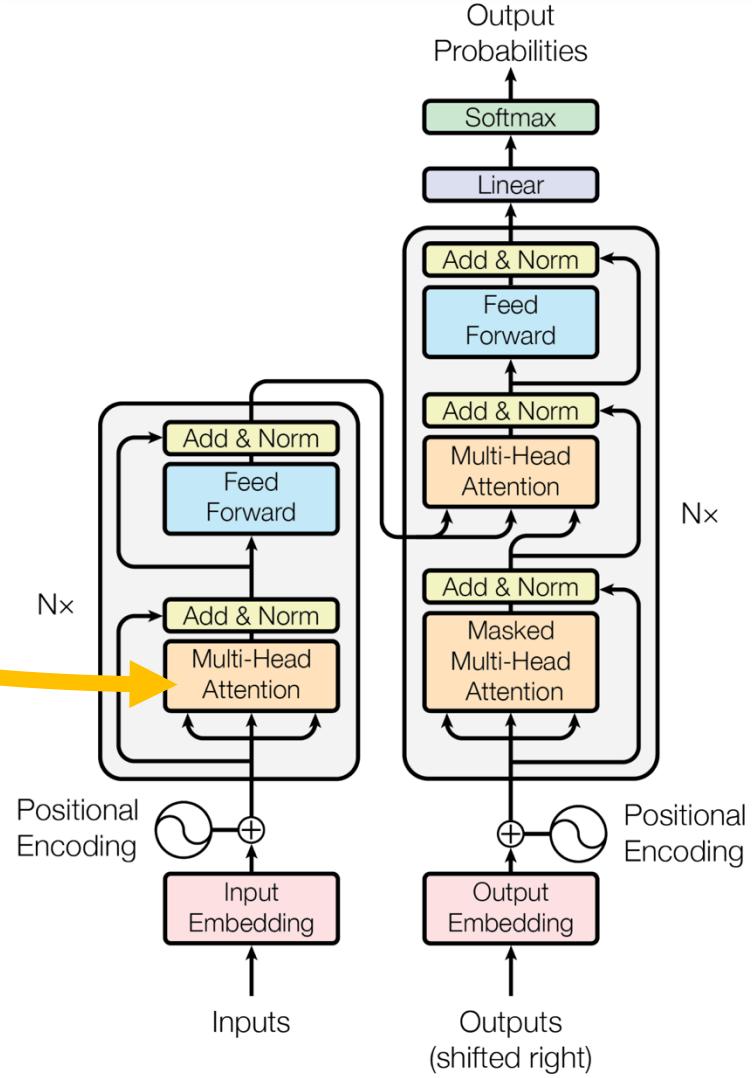
Transformer

[Vaswani et al. 2017]

- Computation of **encoder** attends to both sides.



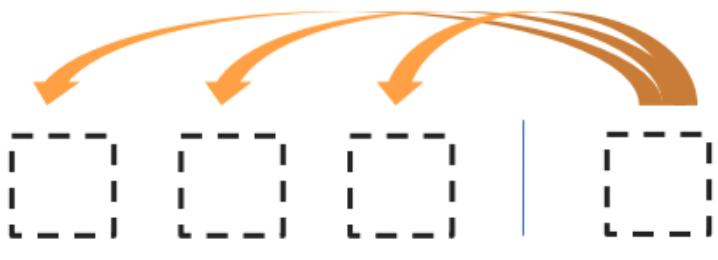
Encoder Self-Attention



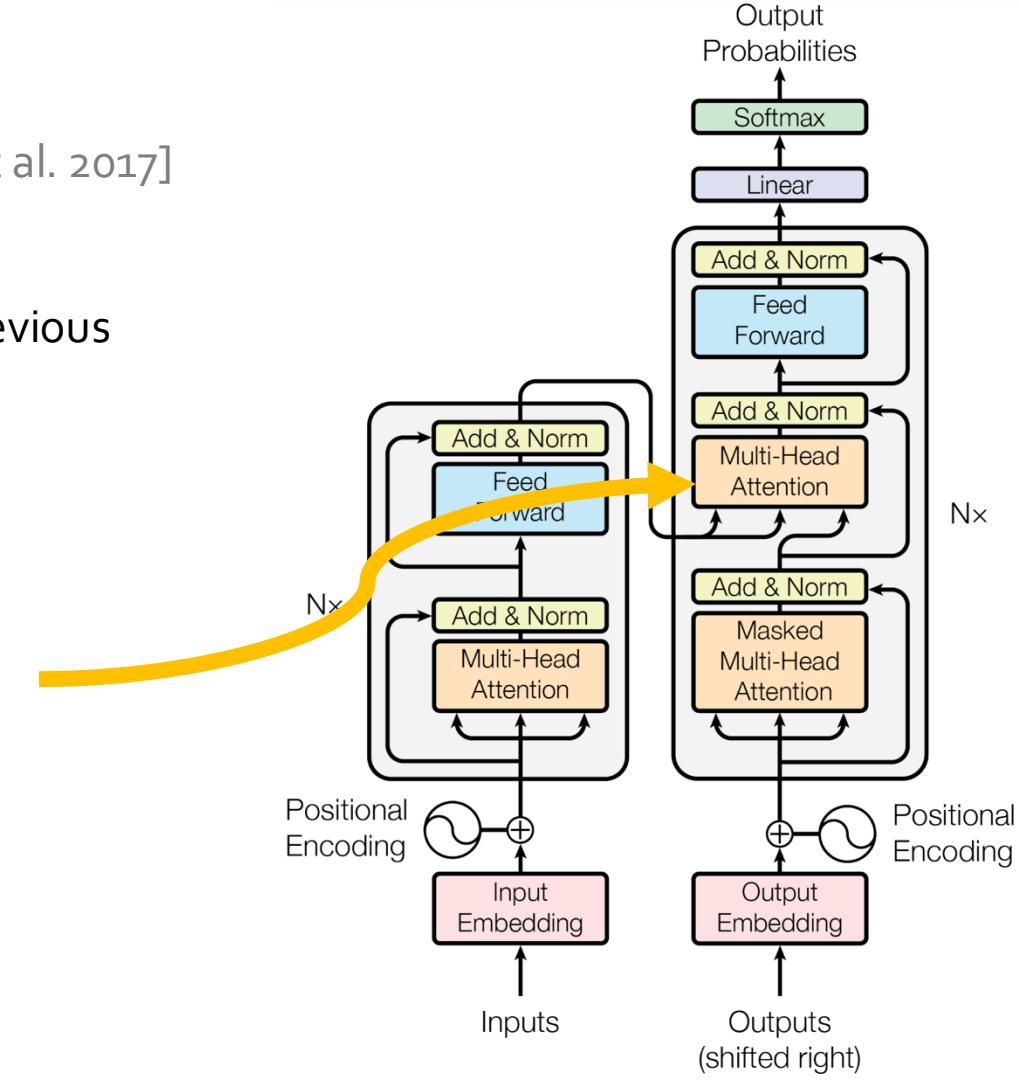
Transformer

[Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder**



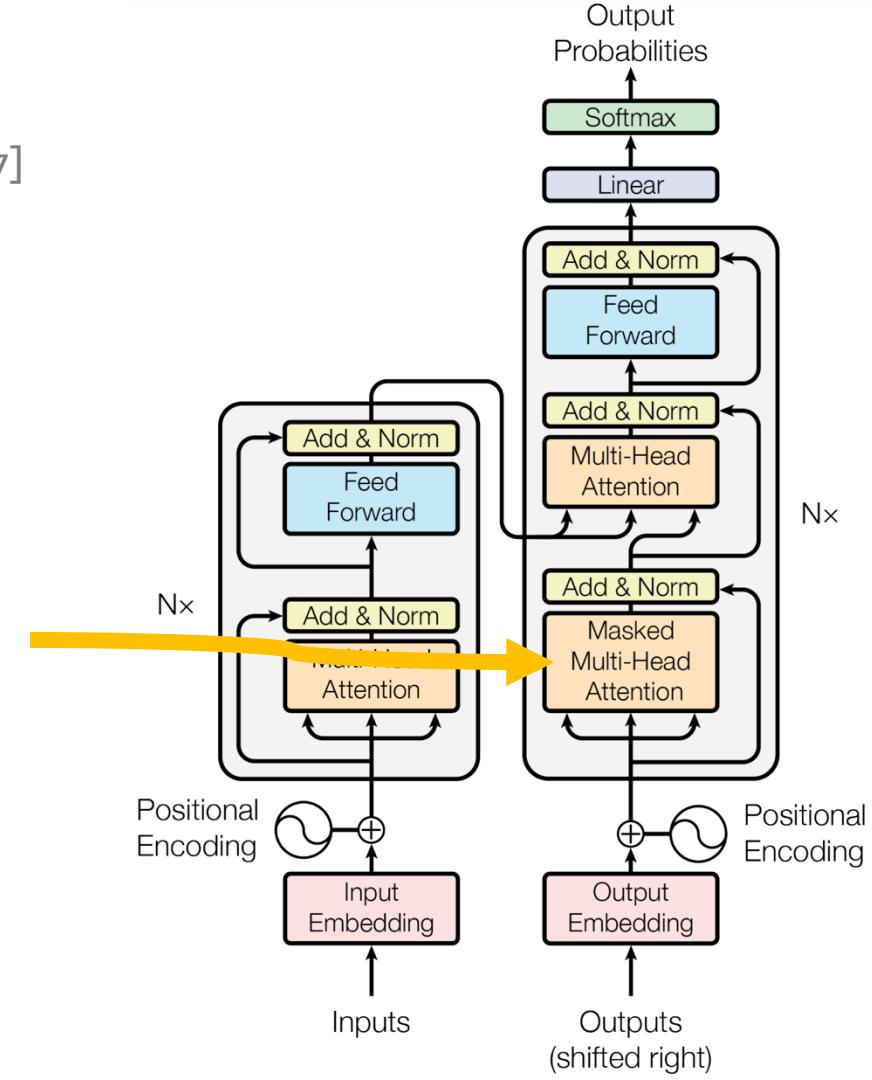
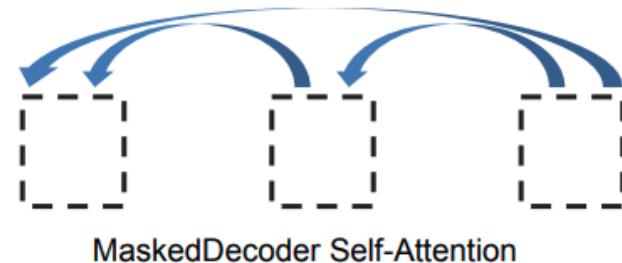
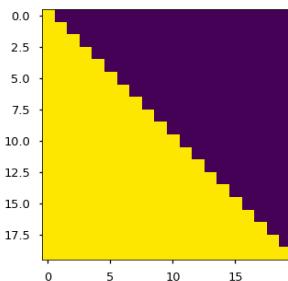
Encoder-Decoder Attention



Transformer

 [Vaswani et al. 2017]

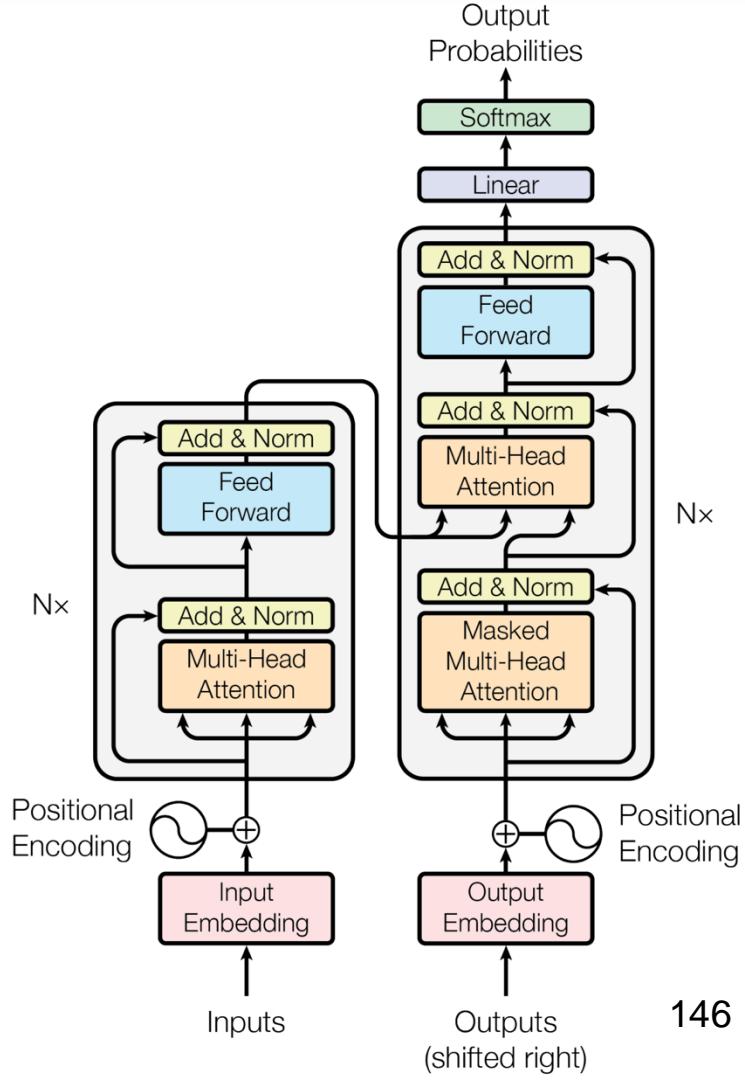
- At any step of **decoder**, it attends to previous computation of **encoder** as well as **decoder's** own generations



Transformer

[Vaswani et al. 2017]

- At any step of **decoder**, it attends to previous computation of **encoder** as well as **decoder's** own generations
- At any step of **decoder**, **re-use** previous computation of **encoder**.
- Computation of **decoder** is **linear**, instead of quadratic.



Quiz: Enc-Dec Cost

- Goal: We are building an encoder-decoder Transformer for summarizing passages to summaries.
- For a passage of length N and a summary of length M , the complexity of the attention is:
 - $O(N) + O(M)$
 - $O(N) + O(M) + O(NM)$
 - $O(N^2) + O(M^2) + O(NM)$
 - $O(N^2) + O(M^2)$

Quiz: Enc-Dec Cost

- Goal: We are building an encoder-decoder Transformer for summarizing passages to summaries.
- For a passage of length N and a summary of length M, the complexity of the attention is:
 - $O(N) + O(M)$
 - $O(N) + O(M) + O(NM)$
 - $O(N^2) + O(M^2) + O(NM)$
 - $O(N^2) + O(M^2)$

No, self attention is all-to-all
and so quadratic.

Quiz: Enc-Dec Cost

- Goal: We are building an encoder-decoder Transformer for summarizing passages to summaries.
- For a passage of length N and a summary of length M , the complexity of the attention is:
 - $O(N) + O(M)$
 - $O(N) + O(M) + O(NM)$
 - $O(N^2) + O(M^2) + O(NM)$
 - $O(N^2) + O(M^2)$

No, self attention is all-to-all
and so quadratic in M and N .

Quiz: Enc-Dec Cost

- Goal: We are building an encoder-decoder Transformer for summarizing passages to summaries.
- For a passage of length N and a summary of length M , the complexity of the attention is:
 - $O(N) + O(M)$
 - $O(N) + O(M) + O(NM)$
 - $O(N^2) + O(M^2) + O(NM)$
 - $O(N^2) + O(M^2)$

No, self attention is all-to-all
and so quadratic in M and N .

Quiz: Enc-Dec Cost

- Goal: We are building an encoder-decoder Transformer for summarizing passages to summaries.
- For a passage of length N and a summary of length M, the complexity of the attention is:
 - $O(N) + O(M)$
 - $O(N) + O(M) + O(NM)$
 - $O(N^2) + O(M^2) + O(NM)$
 - $O(N^2) + O(M^2)$

No, cross attention is missing.

Quiz: Enc-Dec Cost

- Goal: We are building an encoder-decoder Transformer for summarizing passages to summaries.
- For a passage of length N and a summary of length M, the complexity of the attention is:
 - $O(N) + O(M)$
 - $O(N) + O(M) + O(NM)$
 - $O(N^2) + O(M^2) + O(NM)$
 - $O(N^2) + O(M^2)$

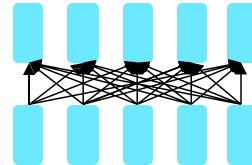
Yes. The three terms are respectively the Encoder self-attention, Decoder self-attention, and Cross attention.

After Transformer ...



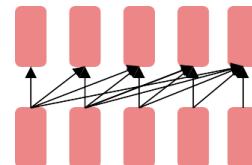
Impact of Transformers

- A building block for a variety of LMs



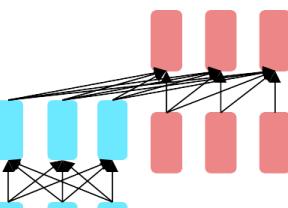
Encoders

- ❖ Examples: BERT, RoBERTa, SciBERT.
- ❖ Captures bidirectional context. Wait, how do we pretrain them?



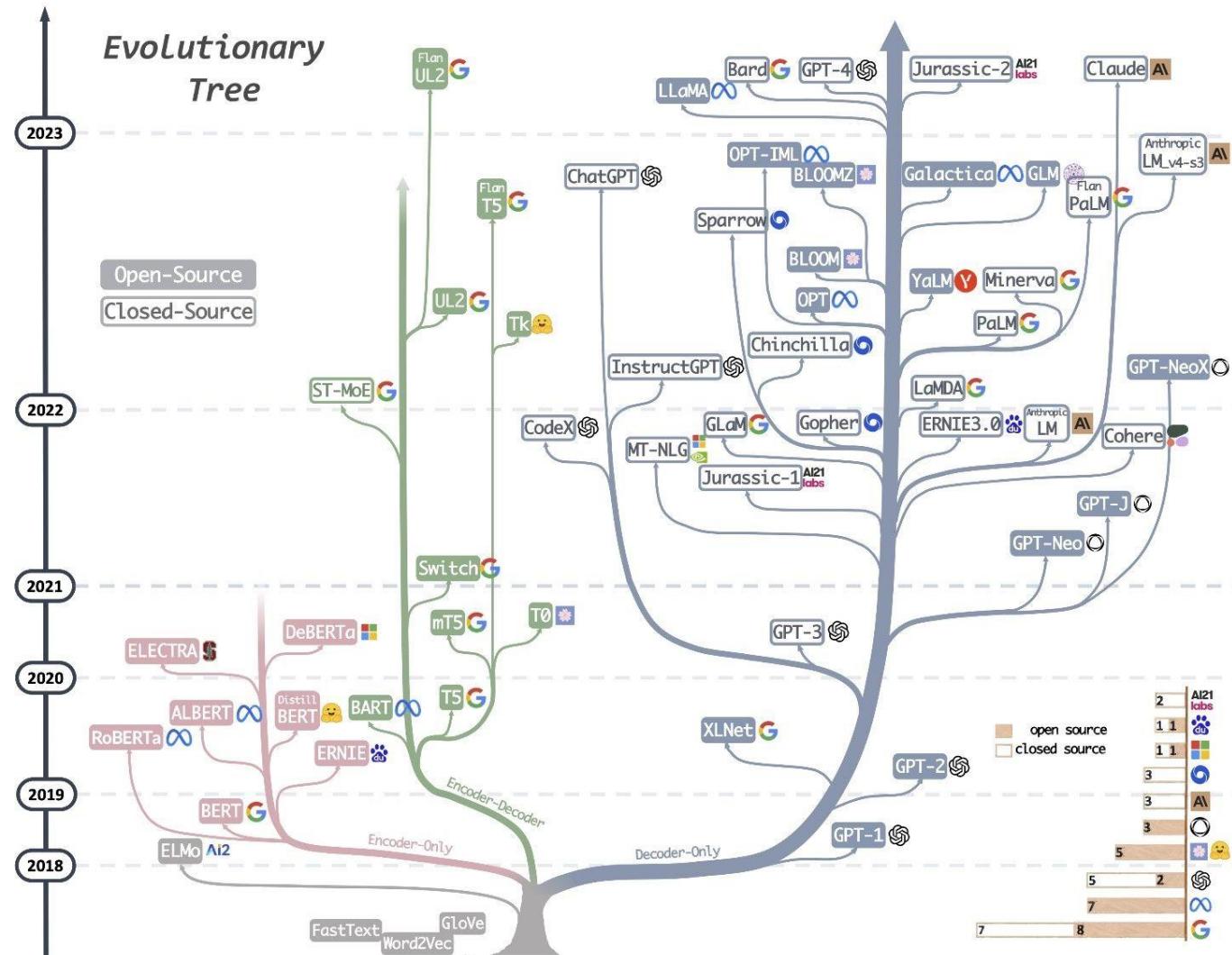
Decoders

- ❖ Examples: GPT-3, GPT-4, LaMMA, Mistral
- ❖ Other name: causal or auto-regressive language model
- ❖ Nice to generate from; can't condition on future words



Encoder-
Decoders

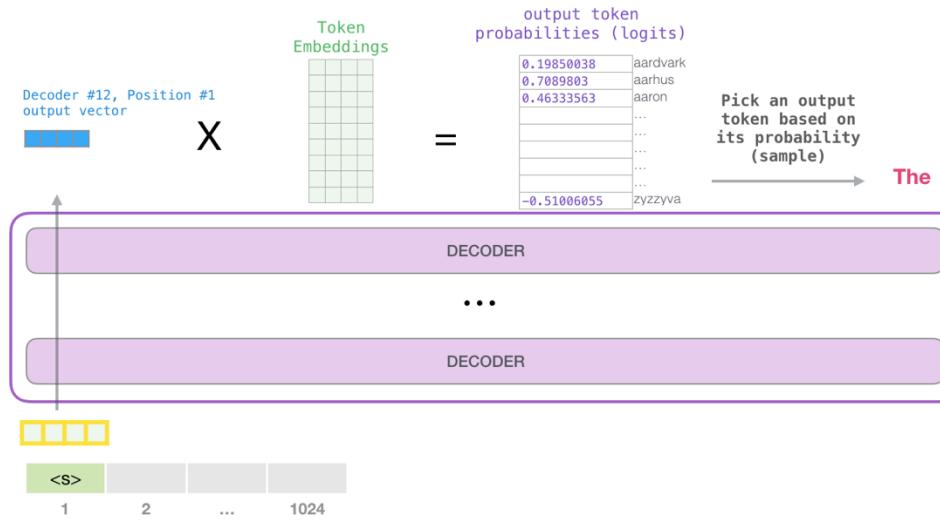
- ❖ Examples: T5, Meena
- ❖ What's the best way to pretrain them?



Few notable models

GPT-2

- GPT-2 uses only Transformer Decoders (no Encoders) to generate new sequences from scratch or from a starting sequence
- As it processes each subword, it masks the “future” words and conditions on and attends to the previous words



GPT-3: Just Scale

- More layers & parameters
- Bigger dataset
- Longer training
- Larger embedding/hidden dimension
- Larger context window

GPT-3 = A **very big** GPT-2



GPT4

- Transformer-based
 - The rest is mystery! 😊
 - If we're going based on costs, GPT4 is ~15-30 times costlier than GPT3. That should give you an idea how its likely size!
- Note, these language models involve more than just pre-training.
 - Pre-training provides the foundation based on which we build the model.
 - We will discuss the later stages (post hoc alignment) in a 2-3 weeks.

Model	Usage	
davinci-002	\$0.0020 / 1K tokens	
Model	Input	Output
gpt-4	\$0.03 / 1K tokens	\$0.06 / 1K tokens

Other Available [Decoder] LMs

EleutherAI: GPT-Neo (6.7B), GPT-J (6B), GPT-NeoX (20B)

<https://huggingface.co/EleutherAI>

<https://6b.eleuther.ai/>

LLaMA, 65B: <https://github.com/facebookresearch/llama>

Mistral and Mixtral:

<https://huggingface.co/mistralai/Mistral-7B-Instruct-v0.2>

<https://huggingface.co/mistralai/Mixtral-8x7B-Instruct-v0.1>

LMSys ChatArena

<https://lmarena.ai/>

Rank* (UB)	Model	Arena Score	95% CI	Votes	Organization	License	Knowledge Cutoff
1	ChatGPT-4o-latest...(2024-08-08)	1317	+5/-5	20805	OpenAI	Proprietary	2023/10
2	Gemini-1.5-Pro-Exp-0801	1298	+4/-4	23232	Google	Proprietary	2023/11
2	Grok-2.08-13	1293	+7/-6	6686	xAI	Proprietary	2024/3
3	GPT-4o-2024-05-13	1286	+3/-3	80741	OpenAI	Proprietary	2023/10
5	GPT-4o-mini-2024-07-18	1275	+5/-4	21621	OpenAI	Proprietary	2023/10
5	Claude-3.5-Sonnet	1271	+3/-3	51097	Anthropic	Proprietary	2024/4
5	Grok-2-Mini-08-13	1268	+7/-7	7266	xAI	Proprietary	2024/3
6	Gemini_Advanced_App_(2024-05-14)	1267	+4/-3	52136	Google	Proprietary	Online
6	Meta_llama-3.1-405b-Instruct	1266	+4/-4	22312	Meta	Llama 3.1 Community	2023/12
7	GPT-4o-2024-08-06	1262	+5/-5	13703	OpenAI	Proprietary	2023/10
8	Gemini-1.5-Pro-001	1260	+3/-2	72623	Google	Proprietary	2023/11
10	Gemini-1.5-Pro-Preview-0409	1257	+3/-3	55604	Google	Proprietary	2023/11
10	GPT-4-Turbo-2024-04-09	1257	+2/-3	86648	OpenAI	Proprietary	2023/12
12	Mistral-Large-2407	1250	+5/-5	14793	Mistral	Mistral Research	2024/7
12	Athene-70b	1250	+5/-5	13655	NexusFlow	CC-BY-NC-4.0	2024/7
14	GPT-4-1106-preview	1251	+3/-3	93540	OpenAI	Proprietary	2023/4

Training Transformer LMs: Empirical Considerations

Pre-training Transformer LMs

- There is so much empirical knowledge/experiences that goes into training these models.
- Various empirical issues about:
 - Preparation/pre-processing data
 - Efficient training of models
 - ...

Data Cleaning: Examples

Remove any:

- References to Javascript
- “Lorem ipsum” text — placeholder text commonly used to demonstrate the visual form of a document

Menu

Lemon

Introduction

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae. The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Article

The origin of the lemon

vn, though

Retain:

- Sentences with terminal punctuation marks
- Pages with at least 5 sentences, sentences with at least 3 words

Please enable JavaScript to use our site.

Home
Products
Shipping
Contact
FAQ

Dried Lemons, \$3.59/pound

Organic dried lemons from our farm in California.
Lemons are harvested and sun-dried for maximum flavor.
Good in soups and on popcorn.

The lemon, Citrus Limon (L.) Osbeck, is a species of small evergreen tree in the flowering plant family rutaceae. The tree's ellipsoidal yellow fruit is used for culinary and non-culinary purposes throughout the world, primarily for its juice, which has both culinary and cleaning uses. The juice of the lemon is about 5% to 6% citric acid, with a ph of around 2.2, giving it a sour taste.

Lorem ipsum dolor sit amet, consectetur adipiscing elit. Curabitur in tempus quam. In mollis et ante at consectetur. Aliquam erat volutpat. Donec at lacinia est. Duis semper, magna tempor interdum suscipit, ante elit molestie urna, eget efficitur risus nunc ac elit. Fusce quis blandit lectus. Mauris at mauris a turpis tristique lacinia at nec ante. Aenean in scelerisque tellus, a efficitur ipsum. Integer justo enim, ornare vitae sem non, mollis fermentum lectus. Mauris ultrices nisl at libero porta sodales in ac orci.

```
function Ball(r) {  
    this.radius = r;  
    this.area = pi * r ** 2;  
    this.show = function(){  
        drawCircle(r);  
    }  
}
```

Pre-training Data: Experiment

- Takeaway:
 - Clean and compact data is better than large, but noisy data.
 - Pre-training on in-domain data helps.

Data set	Size	GLUE	CNNDM	SQuAD	SGLUE	EnDe	EnFr	EnRo
★ C4	745GB	83.28	19.24	80.88	71.36	26.98	39.82	27.65
C4, unfiltered	6.1TB	81.46	19.14	78.78	68.04	26.55	39.34	27.21

Pre-training Data Duplicates

- There is a non-negligible number of duplicates in any pre-training data.

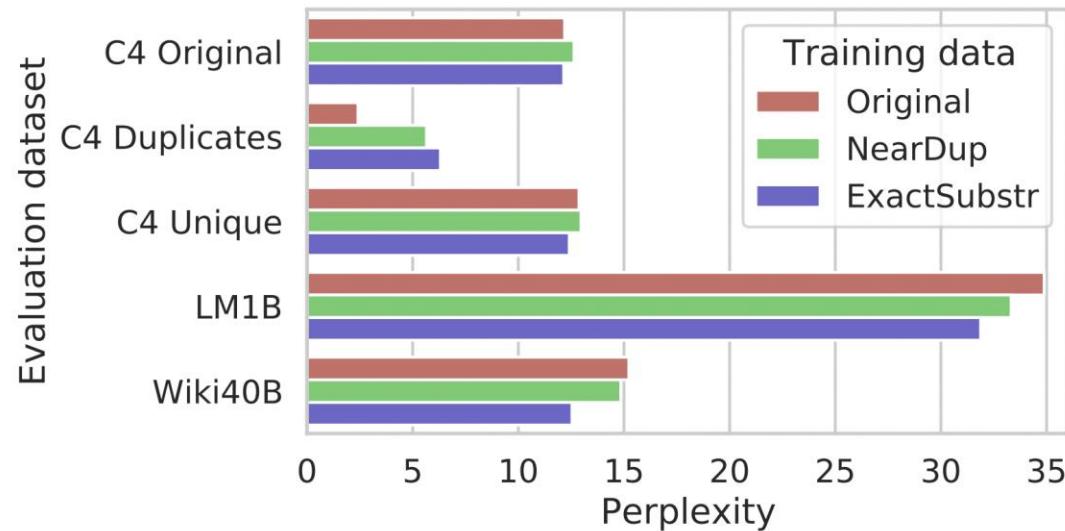
	% train examples with dup in train	% valid with dup in train	% valid with dup in valid
C4	3.04%	1.59%	4.60%
RealNews	13.63%	1.25%	14.35%
LM1B	4.86%	0.07%	4.92%
Wiki40B	0.39%	0.26%	0.72%

Dataset	Example	Near-Duplicate Example
Wiki-40B	\n_START_ARTICLE_\nHum Award for Most Impactful Character \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]	\n_START_ARTICLE_\nHum Award for Best Actor in a Negative Role \n_START_SECTION_\nWinners and nominees\n_START_PARAGRAPH_\nIn the list below, winners are listed first in the colored row, followed by the other nominees. [...]
LM1B	I left for California in 1979 and tracked Cleveland 's changes on trips back to visit my sisters .	I left for California in 1979 , and tracked Cleveland 's changes on trips back to visit my sisters .
C4	Affordable and convenient holiday flights take off from your departure country, "Canada". From May 2019 to October 2019, Condor flights to your dream destination will be roughly 6 a week! Book your Halifax (YHZ) - Basel (BSL) flight now, and look forward to your "Switzerland" destination!	Affordable and convenient holiday flights take off from your departure country, "USA". From April 2019 to October 2019, Condor flights to your dream destination will be roughly 7 a week! Book your Maui Kahului (OGG) - Dubrovnik (DBV) flight now, and look forward to your "Croatia" destination!

Deduplicating Data Improves LMs

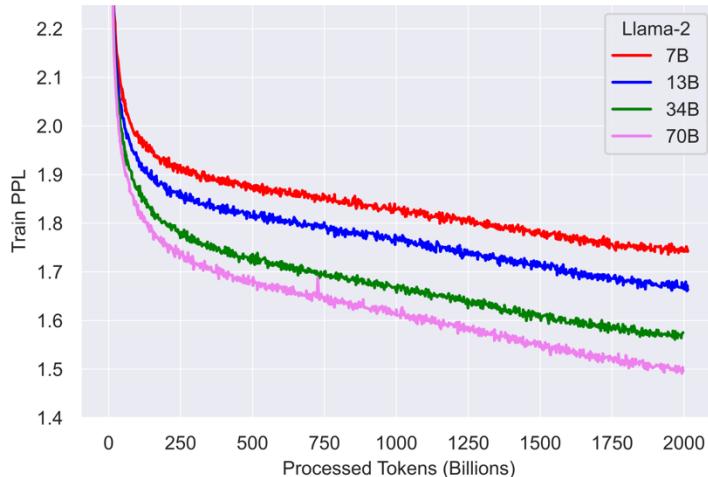
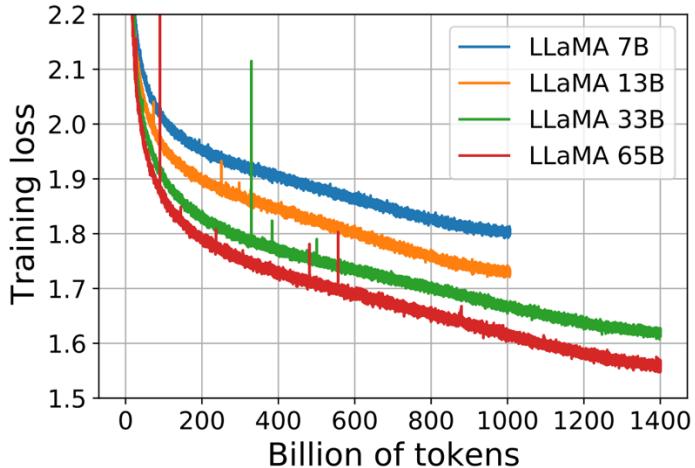
- C4 : the original training data
- C4-NearDup: C4 excluding exact duplicates
- C4-ExactSubs: C4 excluding near-duplicates

Training on deduplicated data almost always leads to lower PPL!



Convergence

- In practice, your model's loss should continue to go down with more training on more data.
- So, the real bottlenecks are:
 - (1) compute
 - (2) data
- Sometimes training diverges (spikes in the loss), at which point practitioners usually restart training from an earlier checkpoint.

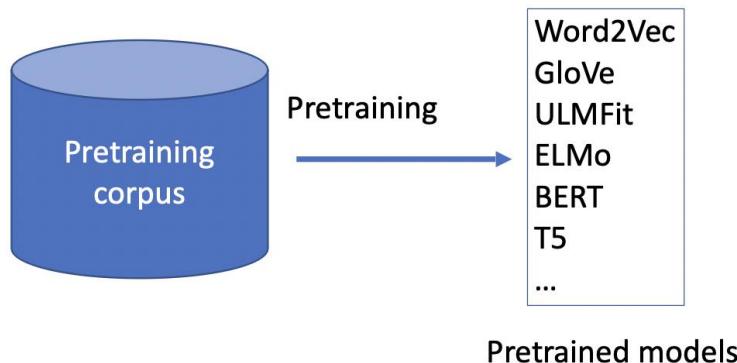


Summary

- There is much empirical knowledge that goes into engineering LMs.
- Here we covered basic topics about data and architecture engineering.
- Various topics are forthcoming: scaling laws, efficient training, etc.

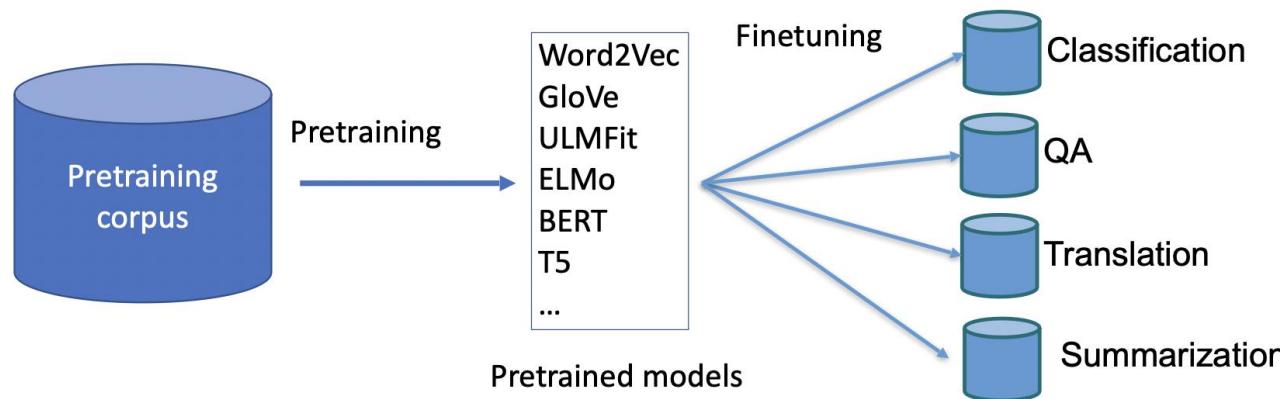
Adaptation via Fine-Tuning

- At this point, we have built a pre-trained model.

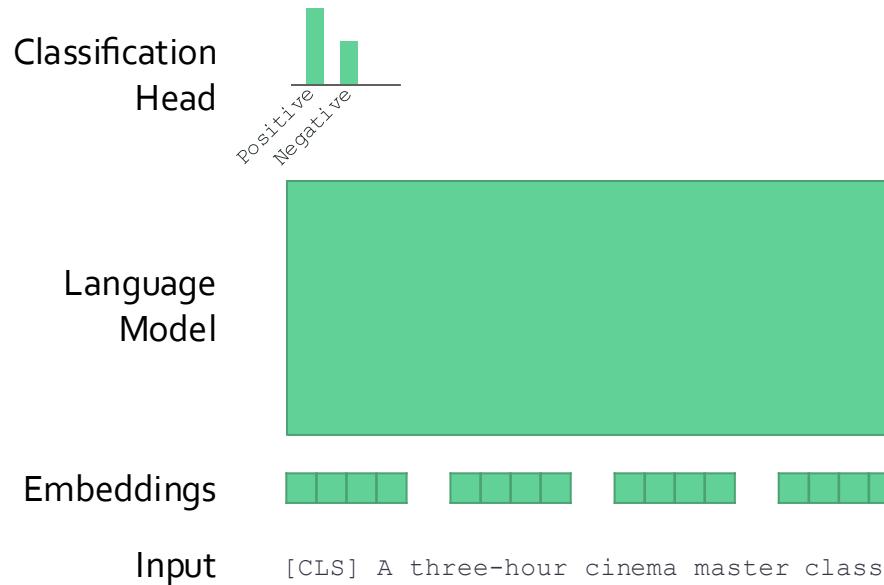


Fine-Tuning for Tasks

- Now we want to “adapt” it for specific tasks with labeled data.



Fine-tuning Pre-trained Models

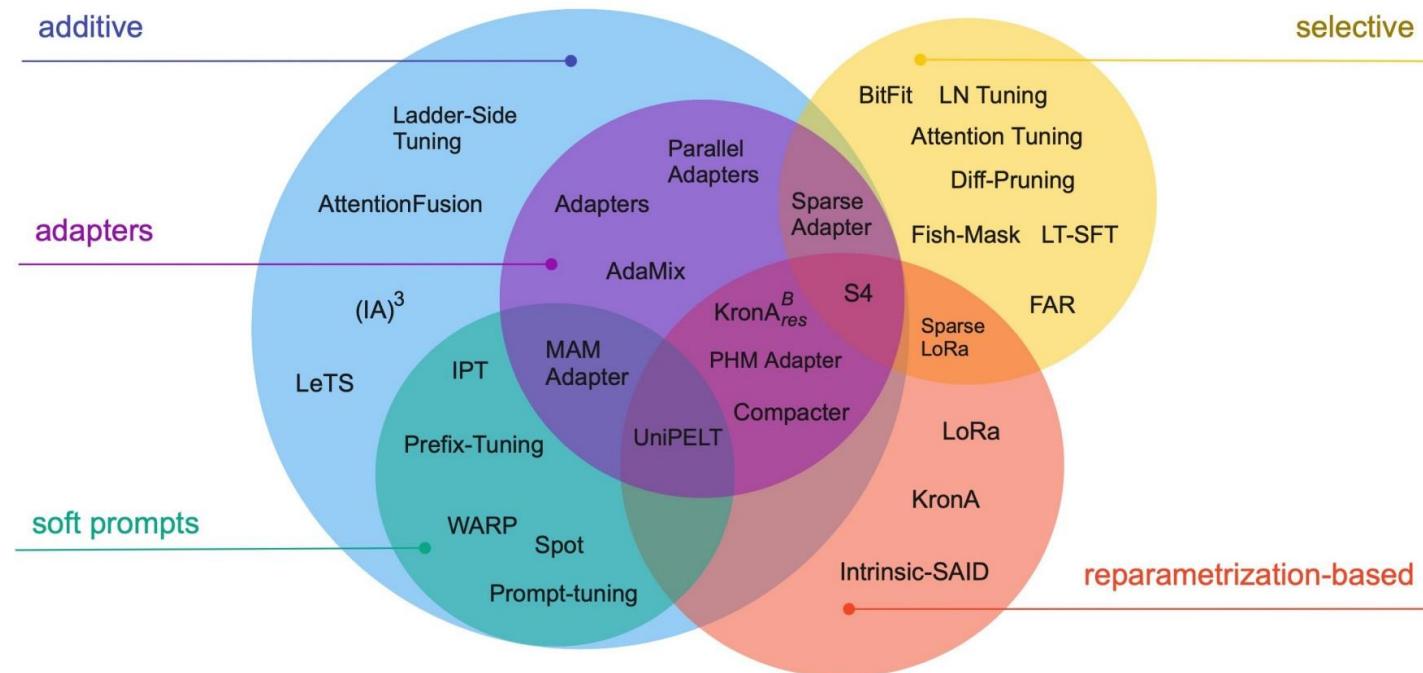


- Whole model tuning:
 - Run an optimization defined on your task data that updates **all** model parameters
- Head-tuning:
 - Run an optimization defined on your task data that updates the parameters of the model “head”

Parameter-efficient Fine-tuning

- In fine-tuning we need to updating and storing all the parameters of the LM
 - We would need to store a copy of the LM for each task
- With large models, storage management becomes difficult.
 - E.g., A model of size 170B parameters requires ~340Gb of storage
 - If you fine-tune a separate model for 100 tasks:
 - $340 * 100 = 34 \text{ TB}$ of storage!

Parameter-efficient Fine-tuning



LoRA: Low-Rank Adaptation

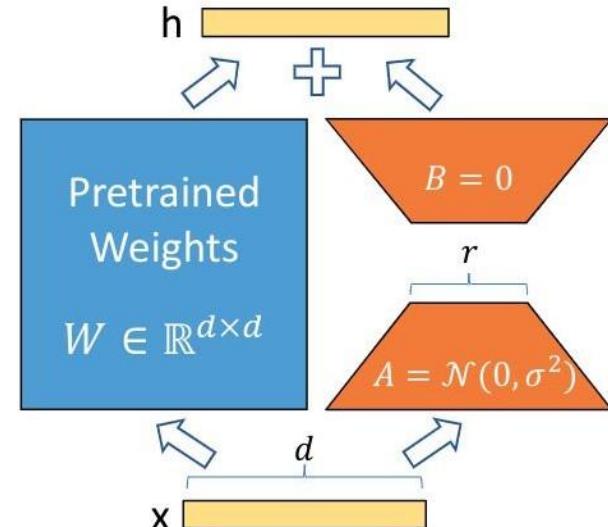
- Hypothesis: the intrinsic rank of the weight matrices in a large language model is low
- Parameter update for a weight matrix is decomposed into a product of two low-rank matrices

$$W \leftarrow W + \Delta W$$

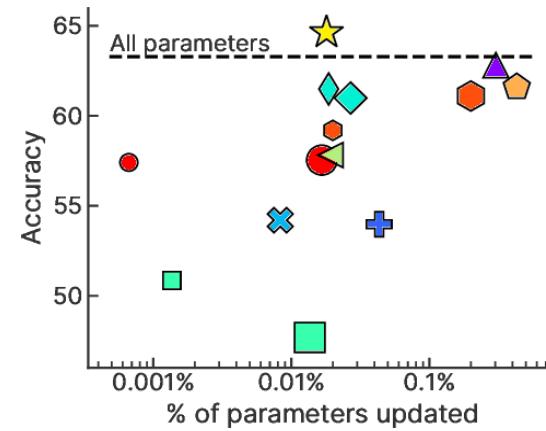
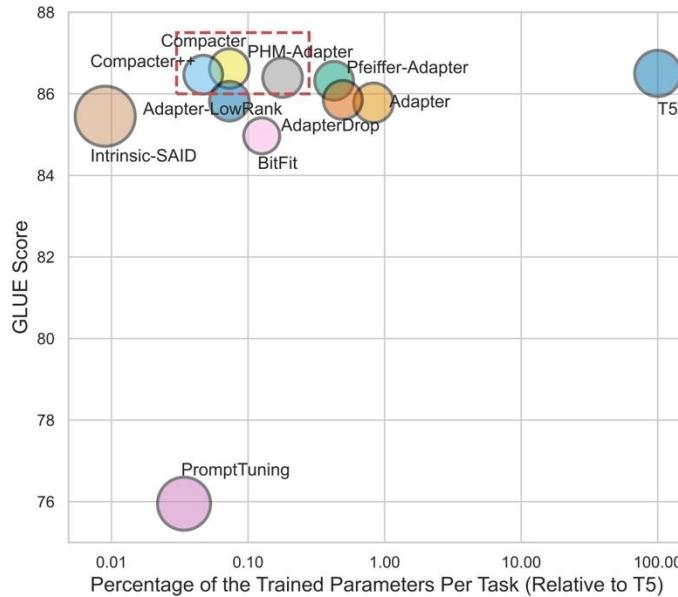
$$\Delta W = BA$$

$$B \in \mathbb{R}^{d,r}, A \in \mathbb{R}^{r,k}, r \ll \min(k, d)$$

- A is initialized with random Gaussian Initialization, B is initialized to zero



Performance/compactness comparison



- ★ (IA)³
- ▲ LoRA
- ✚ BitFit
- ✖ Layer Norm
- ◆ Compacter
- ◆ Compacter++
- Prompt Tuning
- ◀ Prefix Tuning
- Adapter
- ◇ FISH Mask
- Intrinsic SAID

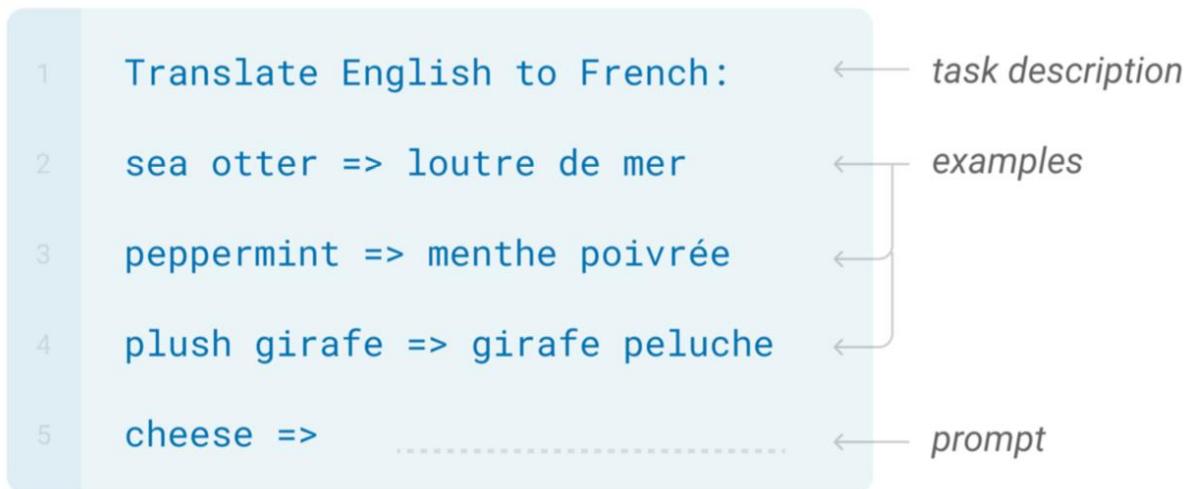
Summary

- Parameter efficient optimization — optimize fewer parameters than the whole model.
 - Space efficiency — fewer parameters to store
 - Computation efficiency? Some gains since you're storing less trainable parameters.

Prompting and In-Context Learning

In-Context Learning

- Learns to do a downstream task by conditioning on input-output examples!
- **No weight update** — our model is not **explicitly pre-trained** to learn from examples
 - The underlying models are quite general



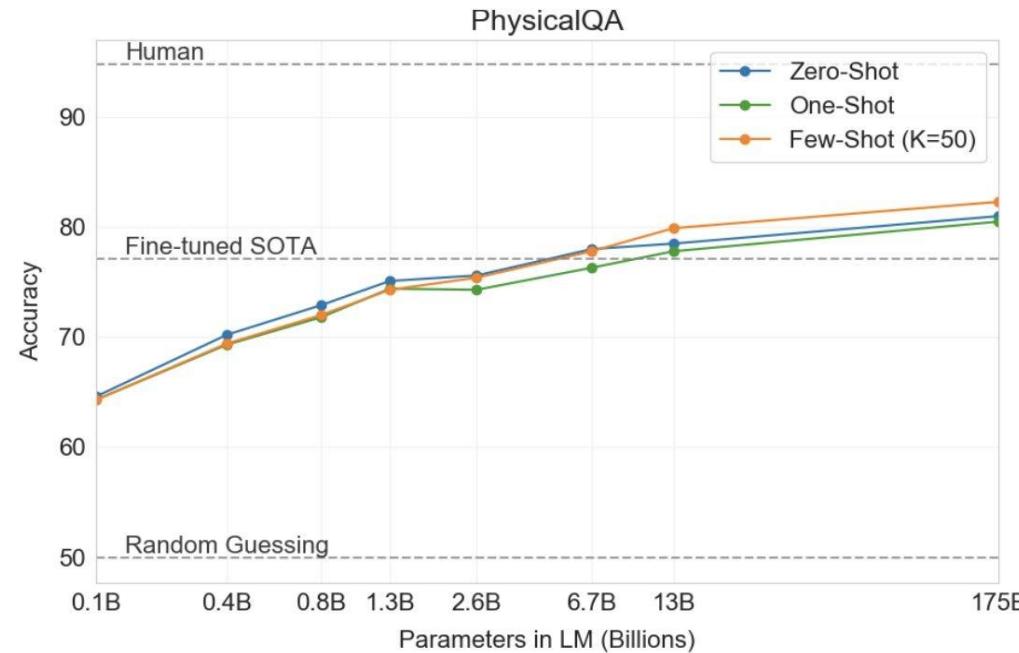
In-Context learning Results



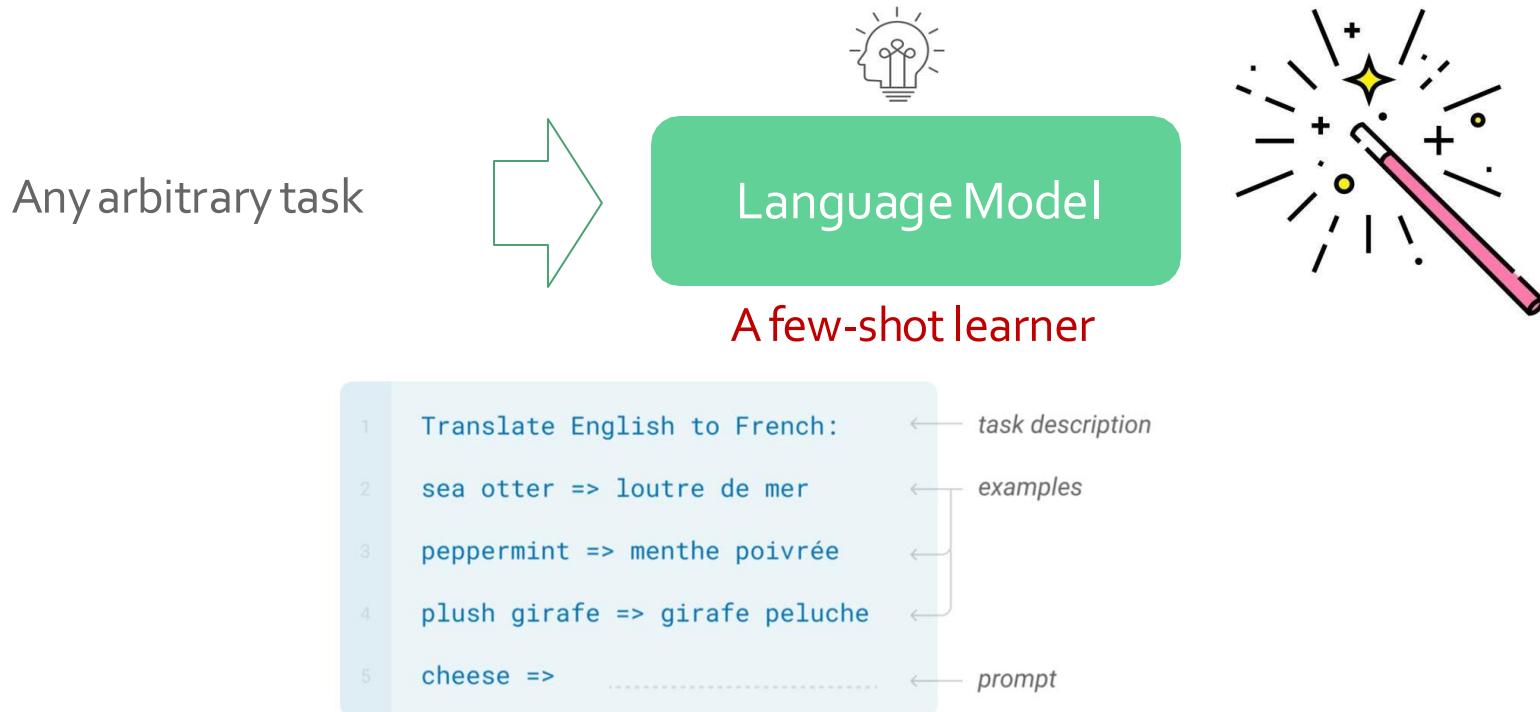
To separate egg whites from the yolk using a water bottle, you should...

a. **Squeeze** the water bottle and press it against the yolk. **Release**, which creates suction and lifts the yolk.

b. **Place** the water bottle and press it against the yolk. **Keep pushing**, which creates suction and lifts the yolk.

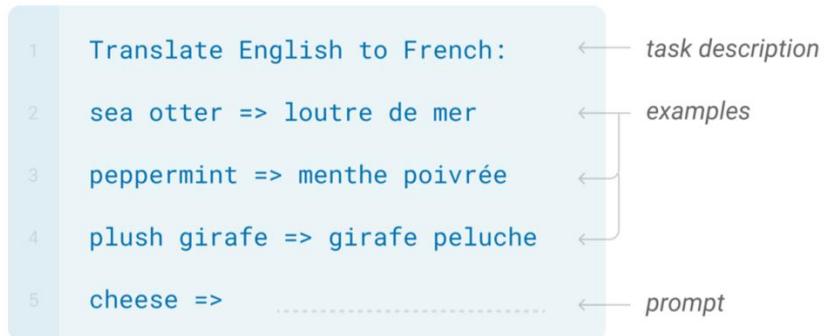


ICL as a General-Purpose Few-Shot Learning Mechanism



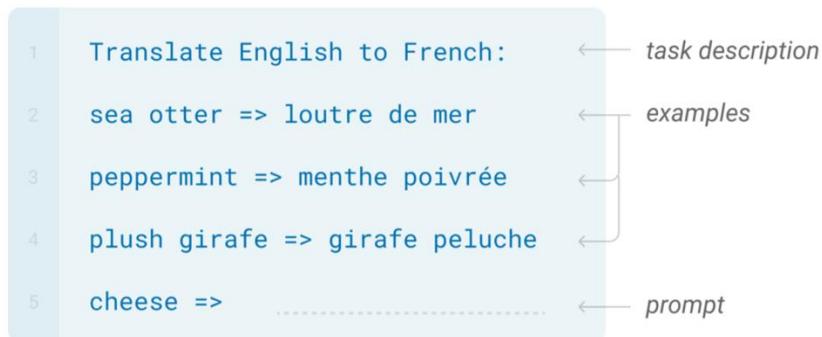
In-Context Learning: Practically Useful

- Labeling data is costly
 - You don't want to get more data
 - Emergent, time-sensitive scenarios
 - Something new happened—need to react quickly!
- Finetuning can tricky
 - Not enough validation data
 - Expensive to train, time and memory



In-Context Learning: Intellectually Intriguing

- Potential test for “Intelligent Behavior”
 - Generalization from few examples
 - Fundamental piece of intelligence
 - What does an LLM “know”?
 - What are the biases/limitations of LLMs?
 - ...
- Insights into Language Modeling



ICL's is quite sensitive!

LM Prompting: Choices of Encoding

Prompt

Input: Subpar acting. Sentiment: negative

Input: Beautiful film. Sentiment: positive

Input: Amazing. Sentiment:

LM Prompting: Choices of Encoding

Prompt

Input: Subpar acting. Contimont: negative

Sentence: Subpar acting. Label: bad

Sentence: Beautiful film. Label: good

Sentence: Amazing. Label:

LM Prompting: Choices of Encoding

Prompt

Input: Subpar acting. Sentiment: negative.

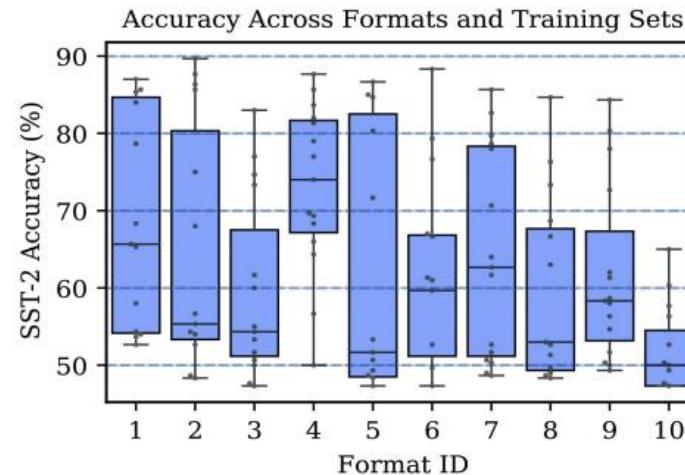
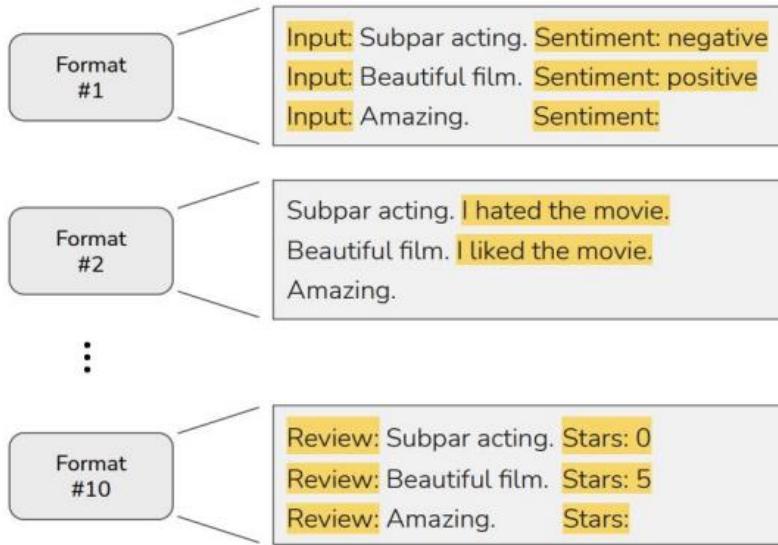
Sentence: Subpar acting. Label: bad.

Q: What's the sentiment of "Subpar acting"?
A: negative

Q: What's the sentiment of "Beautiful film"?
A: positive

Q: What's the sentiment of "Amazing"?
A:

In-Context Learning: Sensitivity to Encoding



In-context learning is highly sensitive to prompt format (training sets and patterns/verbalizers)

What Causes These Variances?

- Here we will provide several factors ...

Majority Label Bias

Prompt

Input: Subpar acting. Sentiment: negative

Input: Beautiful film. Sentiment: positive

Input: Amazing. Sentiment:

- Among 4 demonstrations, count how many are “positive”.
- Then check if the model output correlates with the number of “positive” demos.

Majority Label Bias

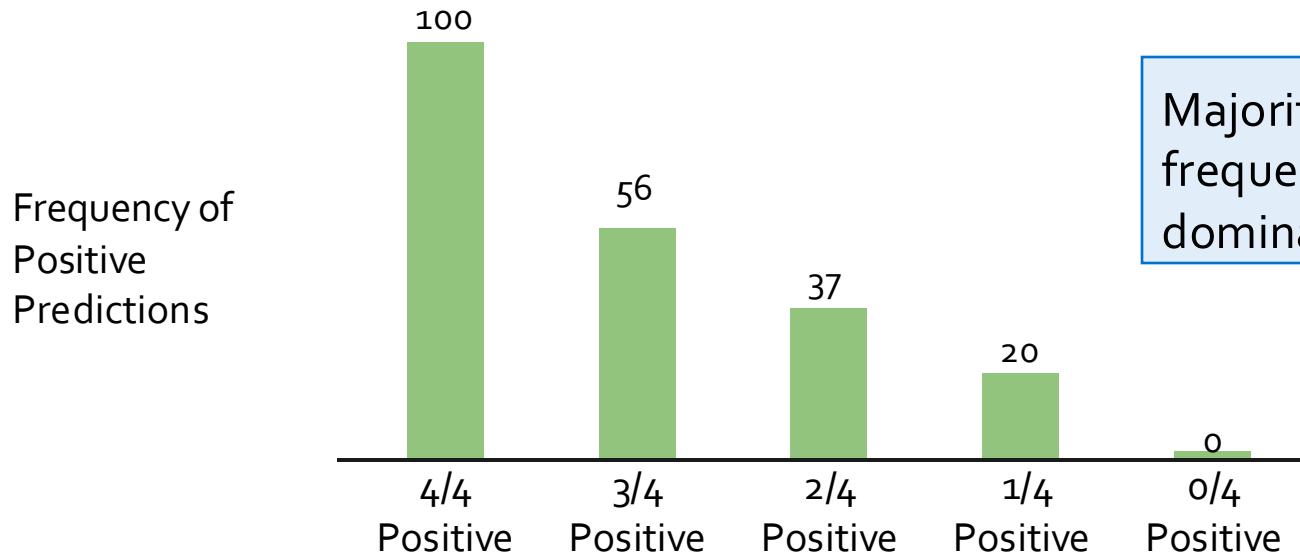
Prompt

Input: Subpar acting. Sentiment: negative

Input: Beautiful film. Sentiment: positive

Input: Amazing. Sentiment:

- Among 4 demonstrations, count how many are “positive”.
- Then check if the model output correlates with the number of “positive” demos.



Majority label bias:
frequent training answers
dominate predictions.

Recency Bias

Prompt

Input: Subpar acting. Sentiment: negative

Input: Beautiful film. Sentiment: positive

Input: Amazing. Sentiment:

- Check if the label of the most-recent demo biases the model output.

Recency Bias

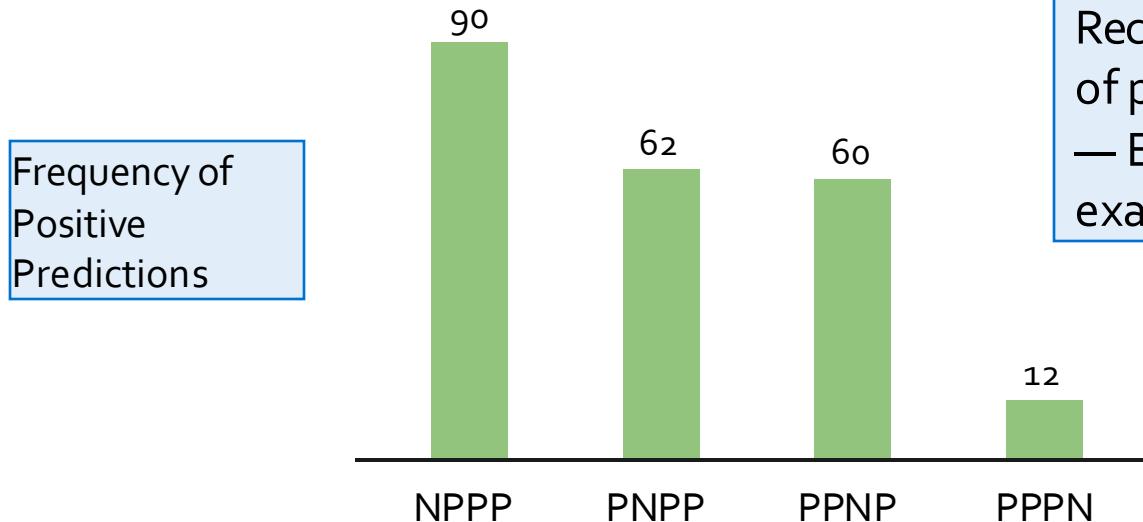
Prompt

Input: Subpar acting. Sentiment: negative

Input: Beautiful film. Sentiment: positive

Input: Amazing. Sentiment:

- Check if the label of the most-recent demo biases the model output.

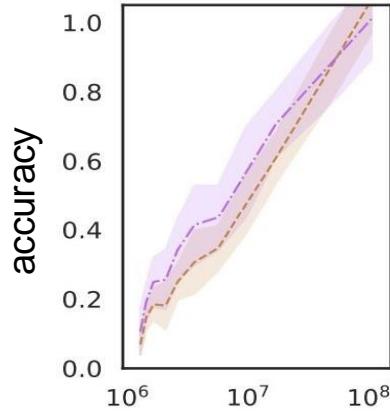


Recency bias: examples near end of prompt dominate predictions
— Explains variance across example permutations!

Impact of Pretraining Term Frequencies

- For each task, identify relevant terms from each instance—numbers and units
- Count co-occurrences of these terms in the pretraining data (term pairs or triples within a fixed window)

Impact of Pretraining Term Frequencies



Q: What is 24 times 18?

A: 432 ✓

Q: What is 23 times 18?

A: 462 ✗

In-context learning performance is highly correlated with term frequencies during pretraining

Why Does ICL Emerge?

- We don't know!
- We have partial empirical explanations.
- And some theoretical analogies.
- But none of them fully explain ICL.

Prompting to Solve Multi-step Problems

Some Problems Involve Reasoning

Q: If there are 3 cars in the parking lot and 2 more cars arrive, how many cars are in the parking lot?

A: The answer is **5**

Q: Take the last letters of the words in "Elon Musk" and concatenate them

A: The answer is **nk**.

Q: What home entertainment equipment requires cable?
Answer Choices: (a) radio shack
(b) substation (c) television (d) cabinet

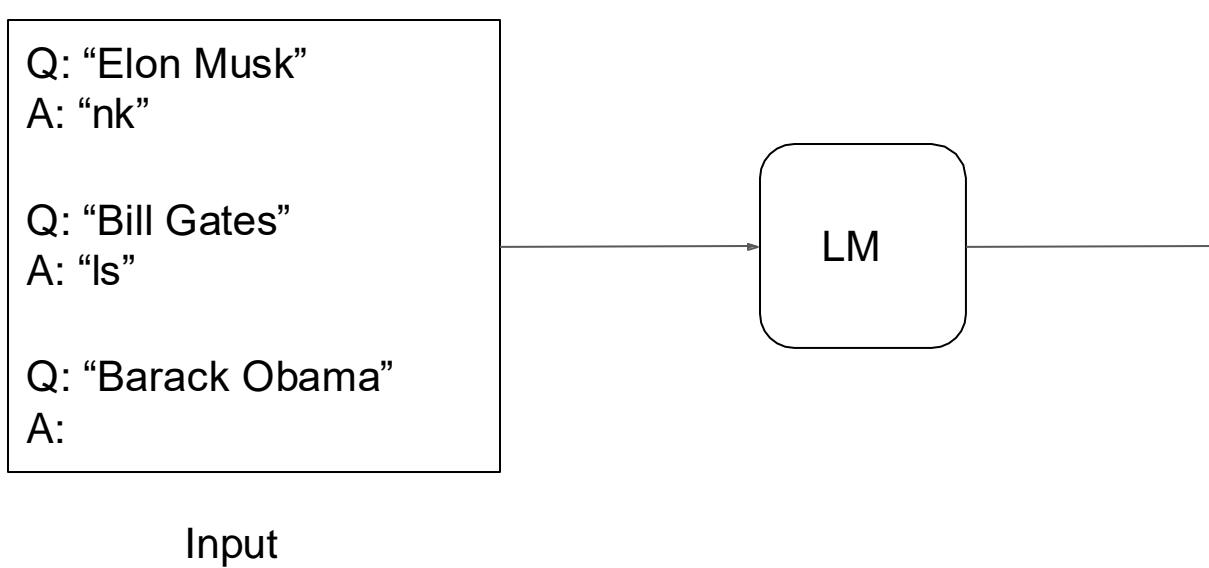
A: The answer is **(c)**.

Arithmetic Reasoning (AR)
 $(+ - \times \div \dots)$

Symbolic Reasoning (SR)

Commonsense Reasoning (CR)

Vanilla ICL on Reasoning Problems



Playground

Load a preset...



Save

View code

Share

...

Q: "Elon Musk"

A: "nk"

Q: "Bill Gates"

A: "Is"

Q: "Barack Obama"

A: "ma"



Complete



Model

text-davinci-003



Temperature 0



Maximum length 256



Stop sequences

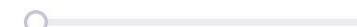
Enter sequence and press Tab



Top P 1



Frequency penalty 0



Presence penalty 0



Submit



54

Playground

Load a preset...



Save

View code

Share

...

Q: "Elon Musk"



A: "nk"

Q: "Bill Gates"

A: "Is"

Q: "Barack Obama"

A: "ma"

How about adding more examples?

Submit



Complete



Model

text-davinci-003



Temperature

0

Maximum length

256

Stop sequences

Enter sequence and press Tab

Top P

1

Frequency penalty

0

Presence penalty

0

Playground

Load a preset...

Save

View code

Share

...

Q: "Elon Musk"

A: "nk"

Q: "Bill Gates"

A: "ls"

Q: "Steve Jobs"

A: "es"

Q: "Larry Page"

A: "ye"

Q: "Jeff Bezos"

A: "fs"

Q: "Barack Obama"

A: "ma"



Submit



94

[Denny Zhou]

Complete

Model

text-davinci-003

Temperature

0

Maximum length

256

Stop sequences

Enter sequence and press Tab

Top P

1

Frequency penalty

0

Presence penalty

0

CoT: Adding “thought” before “answer”

Q: “Elon Musk”

A: the last letter of "Elon" is "n". the last letter of "Musk" is "k". Concatenating "n", "k" leads to "nk". so the output is "nk".

thought

Q: “Bill Gates”

A: the last letter of "Bill" is "l". the last letter of "Gates" is "s". Concatenating "l", "s" leads to "ls". so the output is "ls".

Q: “Barack Obama”

A:

CoT: Adding “thought” before “answer”

Q: “Elon Musk”

A: the last letter of "Elon" is "n". the last letter of "Musk" is "k". Concatenating "n", "k" leads to "nk". so the output is "nk".

thought

Q: “Bill Gates”

A: the last letter of "Bill" is "l". the last letter of "Gates" is "s". Concatenating "l", "s" leads to "ls". so the output is "ls".

Q: “Barack Obama”

A: the last letter of "Barack" is "k". the last letter of "Obama" is "a". Concatenating "k", "a" leads to "ka". so the output is "ka".

CoT: Adding “thought” before “answer”

(a) Few-shot

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: 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:

(Output) The answer is 8. X

Apply CoT to Any Task

Though each task's demonstrations need to be "engineered" manually! 😞

StrategyQA

Q: Yes or no: Would a pear sink in water?

A: The density of a pear is about 0.6 g/cm³, which is less than water. Thus, a pear would float. So the answer is no.

Date Understanding

Q: The concert was scheduled to be on 06/01/1943, but was delayed by one day to today. What is the date 10 days ago in MM/DD/YYYY?

A: One day after 06/01/1943 is 06/02/1943, so today is 06/02/1943. 10 days before today is 05/23/1943. So the answer is 05/23/1943.

Sports Understanding

Q: Is the following sentence plausible? "Joao Moutinho caught the screen pass in the NFC championship."

A: Joao Moutinho is a soccer player. The NFC championship is part of American football, not soccer. So the answer is no.

SayCan (Instructing a robot)

Human: How would you bring me something that isn't a fruit?

Explanation: the user wants something to eat that isn't a fruit. An energy bar is not a fruit, so I will bring the user an energy bar.

Plan: 1. find(energy bar) 2. pick(energy bar) 3. find(user) 4. put(energy bar) 5. done().

Last Letter Concatenation

Q: Take the last letters of the words in "Lady Gaga" and concatenate them.

A: The last letter of "Lady" is "y". The last letter of "Gaga" is "a". Concatenating them is "ya". So the answer is ya.

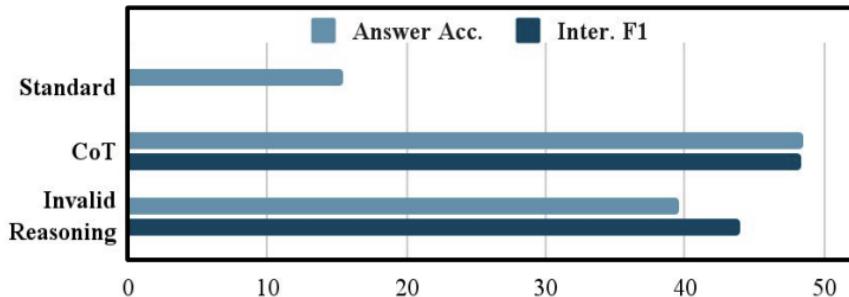
Coin Flip (state tracking)

Q: A coin is heads up. Maybelle flips the coin. Shalonda does not flip the coin. Is the coin still heads up?

A: The coin was flipped by Maybelle. So the coin was flipped 1 time, which is an odd number. The coin started heads up, so after an odd number of flips, it will be tails up. So the answer is no.

Multi-Step Prompting: Steps Don't Have to Be Correct!

- It is possible even with invalid demonstration
- Prompting with invalid reasoning steps can act with correct reasoning steps.



	In-context Demonstration	Inference by LLM
Query	Leah had 32 chocolates and her sister had 42. If they ate 35, how many pieces do they have left in total?	Julie is reading a 120-page book. Yesterday, she was able to read 12 pages and today, she read twice as many pages as yesterday. If she wants to read half of the remaining pages tomorrow, how many pages should she read?
Standard	The answer is 39.	The answer is 18. X
CoT	Originally, Leah had 32 chocolates and her sister had 42. So in total they had $32 + 42 = 74$. After eating 35, they had $74 - 35 = 39$ pieces left in total. The answer is 39.	Julie is reading a 120-page book. Yesterday, she read 12 pages and today, she read 24 pages. So she read a total of $12 + 24 = 36$ pages. Now she has $120 - 36 = 84$ pages left. Since she wants to read half of the remaining pages, she should read $84 / 2 = 42$ pages. The answer is 42. ✓
Invalid Reasoning	Originally, Leah had 32 chocolates and her sister had 42. So her sister had $42 - 32 = 10$ chocolates more than Leah has. After eating 35, since $10 + 35 = 45$, they had $45 - 6 = 39$ pieces left in total. The answer is 39.	Yesterday, Julie read 12 pages. Today, she read $12 * 2 = 24$ pages. So she read a total of $12 + 24 = 36$ pages. Now she needs to read $120 - 36 = 84$ more pages. She wants to read half of the remaining pages tomorrow, so she needs to read $84 / 2 = 42$ pages tomorrow. The answer is 42. ✓

Summary

- Prompting language models is a powerful way to adapt them to our desired tasks.
 - We saw prompting via in-context demonstrations
 - We also saw various variants and extensions
- They also serve as a gateway to understand the underlying dynamics inside models.
- Lots of activity in this area and room for a lot of research progress.

Prompt Engineering

- Reformulating tasks to a language that is easier to for the models.
 - Show demonstrations
 - Decompose your problem
 - Ask for rationales (a la CoT)
 - Check for consistency
 - ...
- **Question for you:** will “prompt engineering” be relevant topic in the coming years?

AI 'prompt engineer' jobs can pay up to \$375,000 a year and don't always require a background in tech

Britney Nguyen May 1, 2023, 11:34 PM GMT+8



The rise of generative AI tools like ChatGPT is creating a hot market for "prompt engineers" who test and improve chatbot answers. Getty Images

Alignment

Things that Generative LMs Can Do

- Johns Hopkins University is in _____. [Trivia]
- I put _____ fork down on the table. [syntax]
- The woman walked across the street, checking for traffic over _____ shoulder. [coreference]
- I went to the ocean to see the fish, turtles, seals, and _____. [lexical semantics/topic]
- What I got from the two hours watching it was popcorn. The movie was _____. [sentiment]
- Thinking about the sequence 1, 1, 2, 3, 5, 8, 13, 21, ____ [basic arithmetic]

Language Modeling ≠ Following Human Instructions

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.

There is a mismatch between LLM pre-training and user intents.

Language Modeling ≠ Following Human Instructions

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

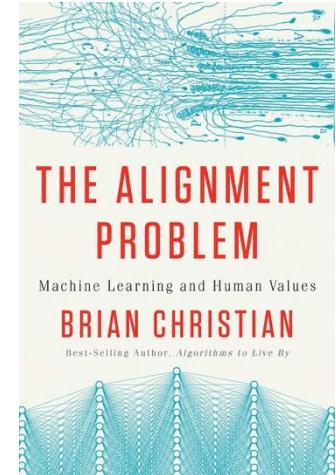
COMPLETION Human

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.

There is a mismatch between LLM pre-training and user intents.

[Mis]Alignment in Language Models

- There is clearly a mismatch between what **pre-trained** models can do and what we want.
- Addressing this gap is the focus of “alignment” research.
 - Making sure it does what its designers **intended**.
 - Making sure its outputs comply with **rules**.
 - Making sure it produces outputs that comply with **moral principles**.
 -



[Mis]Alignment in a Broad Sense

- “The result of arranging in or along a line, or into appropriate relative positions; the layout or orientation of a thing or things disposed in this way” — Oxford Dictionary



Alignment Problem is Everywhere!

- This is a fundamental problem of human society.
- Most things we do in our day-to-day life is an alignment problem.

- **Example 1:** Alignment mechanisms in this class:
 - Me giving lectures; You asking questions; You solving homework assignments, ...
- **Example 2:** Alignment mechanisms in our society:
 - Law and its enforcement; norms and cultures; markets, democracy, ...



Aligning Language Models: Instruction-tuning

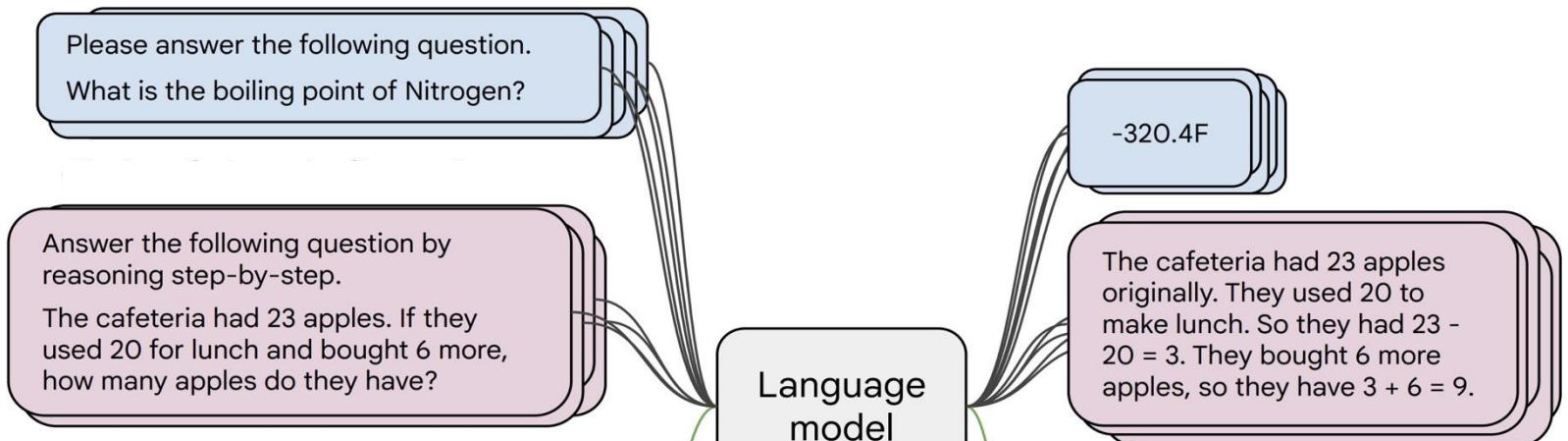
Instruction-tuning

- Finetuning language models on a collection of datasets that involve mapping language instructions to their corresponding desirable generations.

Instruction-tuning

[Weller et al. 2020; Mishra et al. 2021; Wang et al. 2022,
Sanh et al. 2022; Wei et al., 2022, Chung et al. 2022, many others]

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



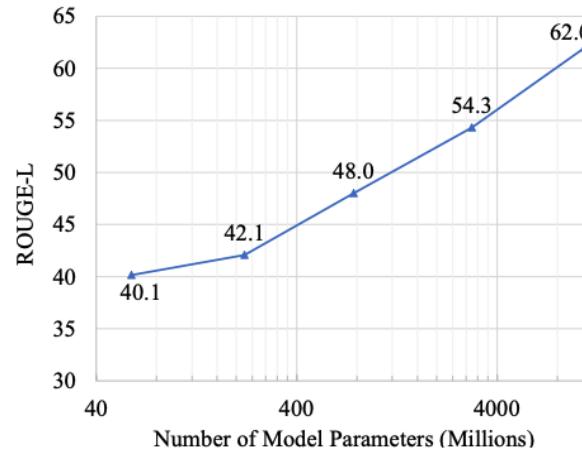
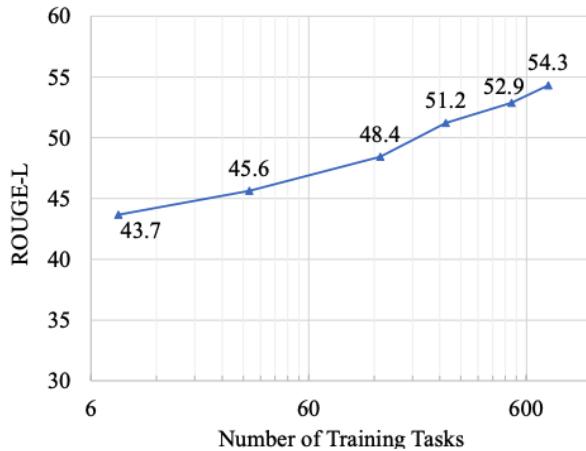
2. Evaluate on unseen tasks

Inference: generalization to unseen tasks

Q: Can Geoffrey Hinton have a conversation with George Washington?
Give the rationale before answering.

Geoffrey Hinton is a British-Canadian computer scientist born in 1947. George Washington died in 1799. Thus, they could not have had a conversation together. So the answer is "no".

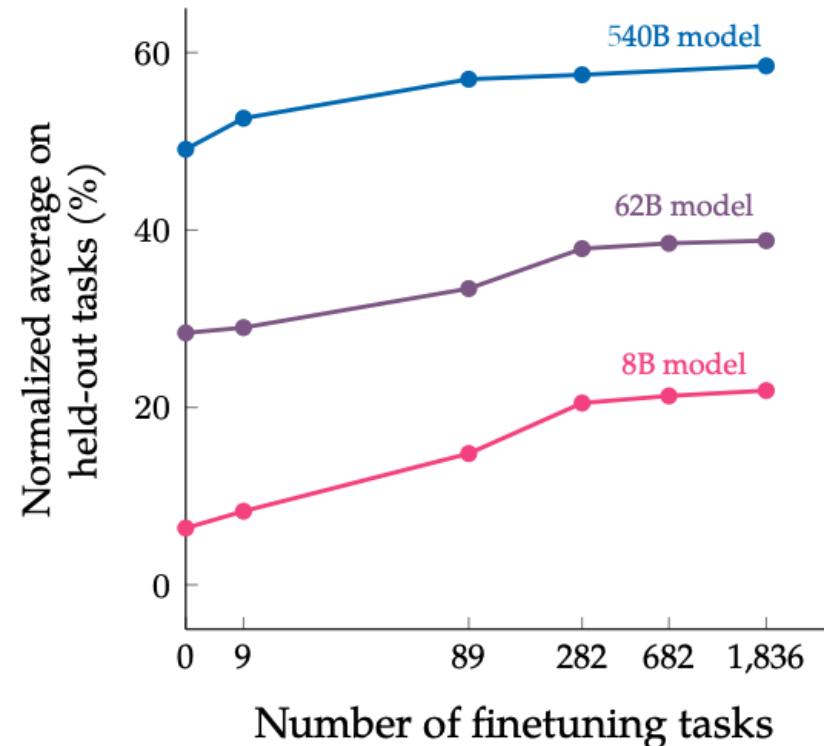
Scaling Instruction-Tuning



Linear growth of model performance
with exponential increase in observed tasks and model size.

Scaling Instruction-Tuning

- Instruction finetuning improves performance by a large margin compared to no finetuning
- Increasing the number of finetuning tasks improves performance
- Increasing model scale by an order of magnitude (i.e., 8B → 62B or 62B → 540B) improves performance substantially for both finetuned and non-finetuned models



Limits of Instruction-Tuning

1. Difficult to collect diverse labeled data

Limited/sparse feedback—usually considered a curse, but now a blessing.

2. Rote learning (token by token) —
 - limited creativity

“don't give a man fish rather teach him how to fish by himself”

3. Agnostic to model's knowledge —
 - may encourage hallucinations

The model itself should be involved in the alignment loop.



Aligning Language Models: Reinforcement Learning w/ Feedback



Reinforcement Learning: Intuition

Action here: generating responses/token

agent



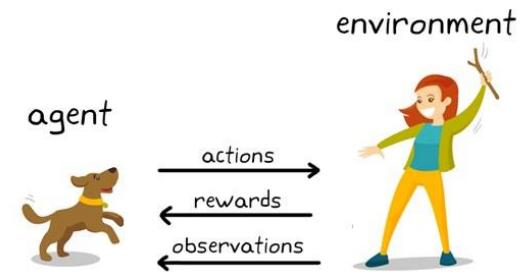
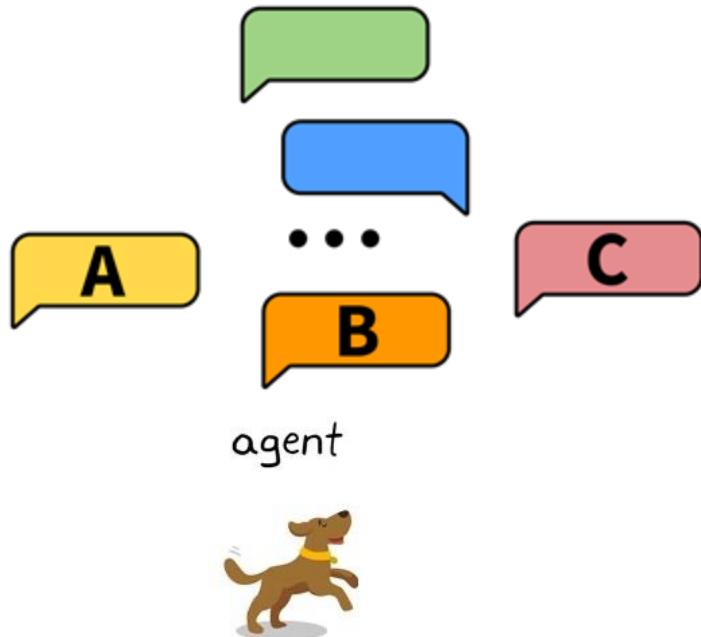
environment



Reward here: whether humans liked the generation (sequence of actions=tokens)

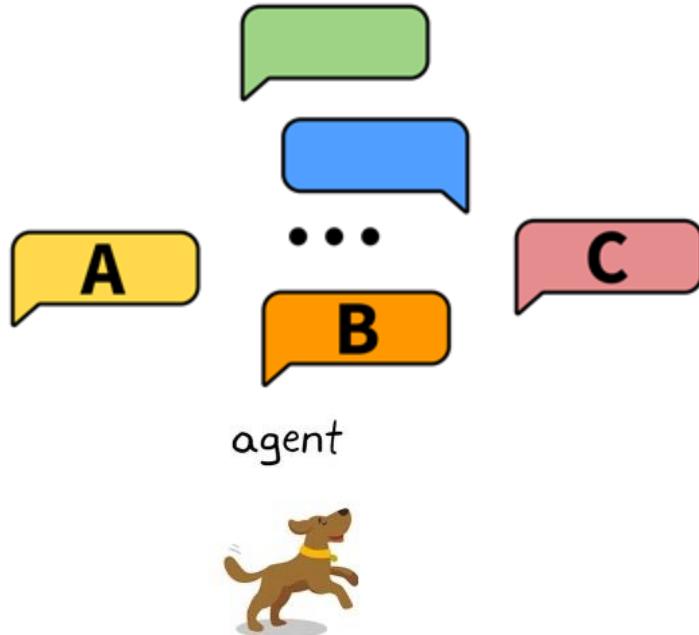
Goal

Task: choose the better next message in a conversation



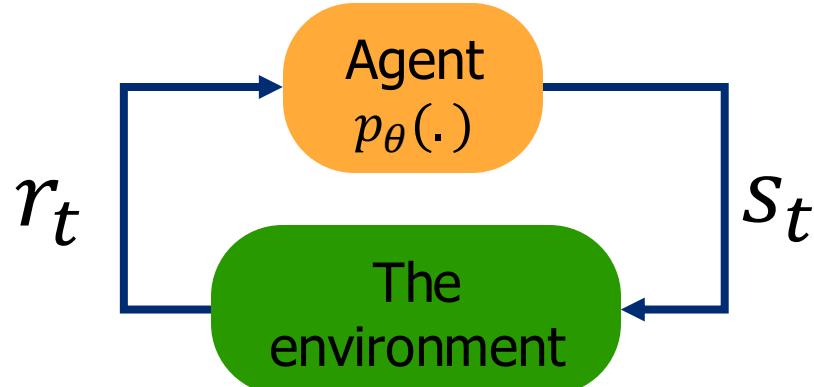
Feedback Mechanism

Scoring interface: Likert scale or rankings

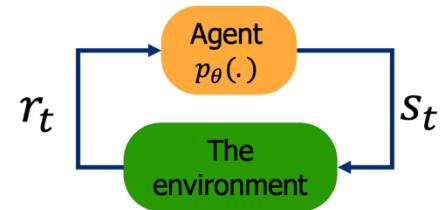


Reinforcement Learning: Formalism

- An agent **interacts** with an environment by taking **actions** s_t .
- The environment returns a **reward** r_t for the **action** s_t .
- Agent uses a **policy function** p_θ to choose an action at a given **state**.
- We need to figure out: (1) reward function r_t and (2) the policy function p_θ

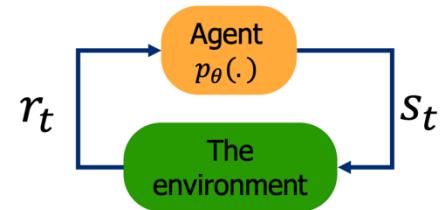


Reinforcement Learning from Human Feedback



- Imagine a reward function: $R(s; \text{prompt}) \in \mathbb{R}$ for any output s to a prompt.
- The reward is higher when humans prefer the output.
- Good generation is equivalent to finding reward-maximizing outputs:

Reinforcement Learning from Human Feedback



- Imagine a reward function: $R(s; \text{prompt}) \in \mathbb{R}$ for any output s to a prompt.
- The reward is higher when humans prefer the output.
- Good generation is equivalent to finding reward-maximizing outputs:

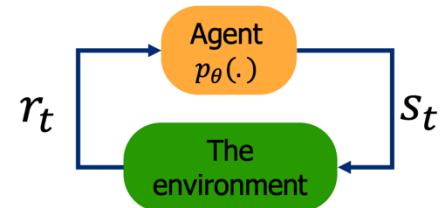
Expected reward over the course of sampling from our policy (generative model)

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

$p_\theta(s)$ is a pre-trained model with params θ we would like to optimize (policy function)

- On the notation:
 - “ \mathbb{E} ” in practice is estimated empirically (i.e., average).
 - “ \sim ” indicates sampling from a given distribution.

Reinforcement Learning from Human Feedback



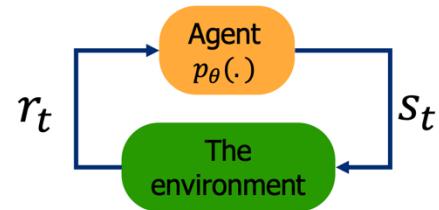
- Imagine a reward function: $R(s; \text{prompt}) \in \mathbb{R}$ for any output s to a prompt.
- The reward is higher when humans prefer the output
- Good generation is equivalent to finding reward-maximizing outputs:

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

- What we need to do:
 - (1) Estimate the reward function $R(s; \text{prompt})$.
 - (2) Find the best generative model p_θ that maximizes the expected reward:

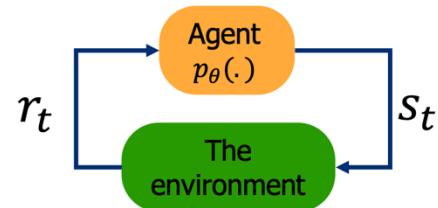
$$\hat{\theta} = \operatorname{argmax}_\theta \mathbb{E}_{\hat{s} \sim p_\theta}[R(\hat{s}; \text{prompt})]$$

Step 1: Estimating the Reward R



- Obviously, we don't want to use human feedback directly since that could be 💰💰💰
- Alternatively, we can build a model to mimic their preferences [[Knox and Stone, 2009](#)]

Step 1: Estimating the Reward R



- Obviously, we don't want to use human feedback directly since that could be 💰💰💰
- Alternatively, we can build a model to mimic their preferences [[Knox and Stone, 2009](#)]
- Approach 2: ask for pairwise comparisons [Phelps et al. 2015; Clark et al. 2018]

Bradley-Terry [1952]
paired comparison model

Pairwise comparison of multiple
provides which can be more reliable

prompt
Explain "space elevators" to
a 6-year-old.



s_1

It is like any typical elevator,
but it goes to space. ...



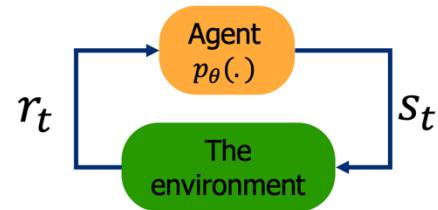
s_2

Explain gravity to a 6-year-old. ...



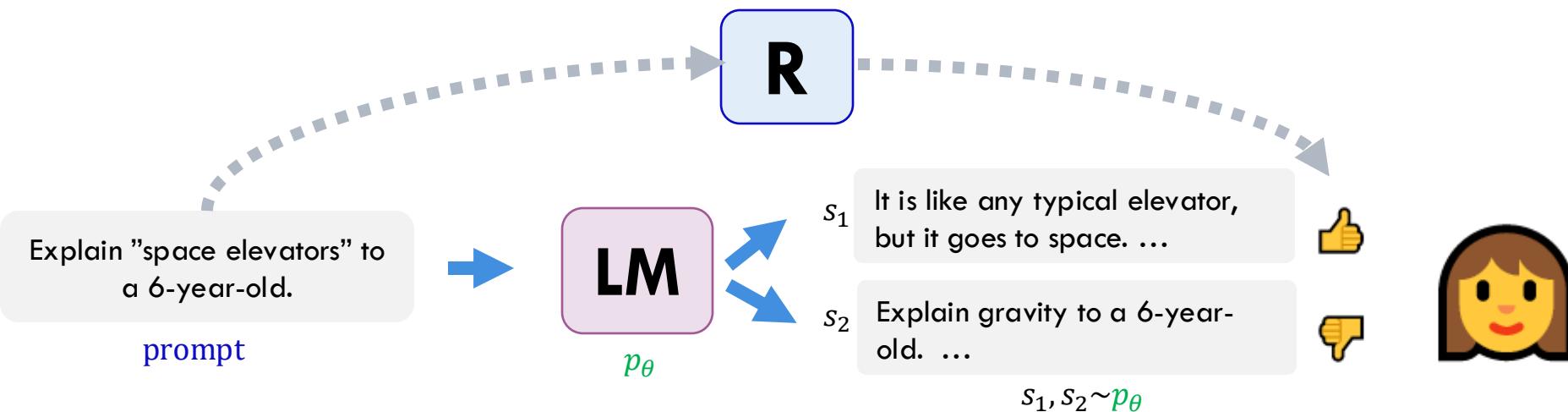
$s_1, s_2 \sim p_\theta$

Step 1: Estimating the Reward R

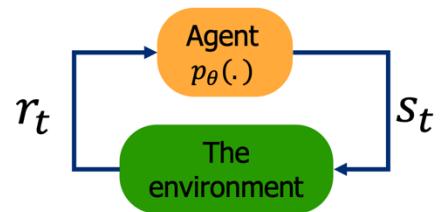


$$J(\phi) = -\mathbb{E}_{(s^+, s^-)} [\log \sigma(R(s^+; \text{prompt}) - R(s^-; \text{prompt}))]$$

"winning" sample "losing" sample

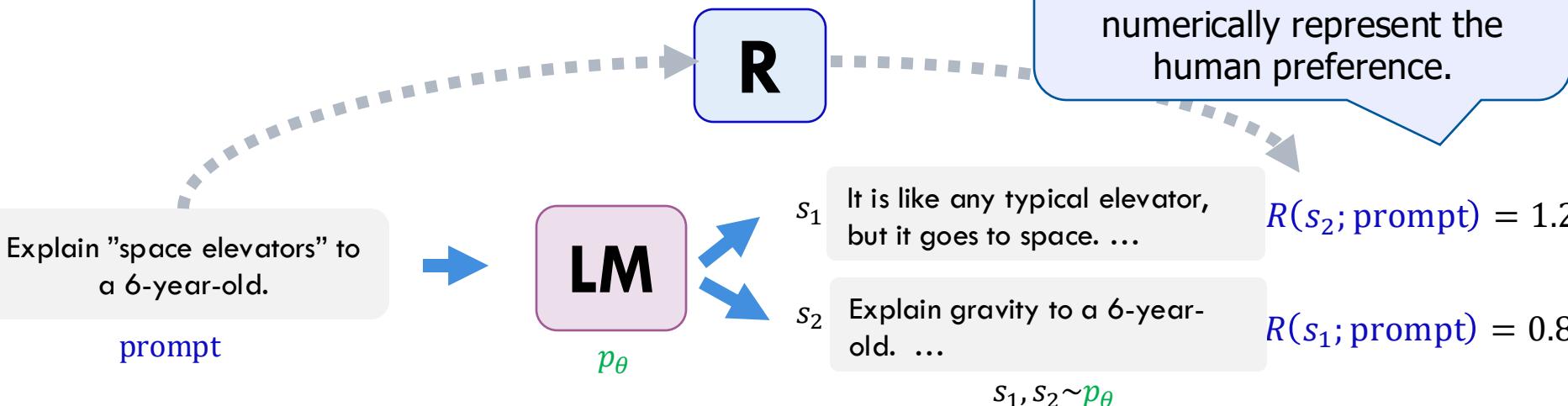


Step 1: Estimating the Reward R



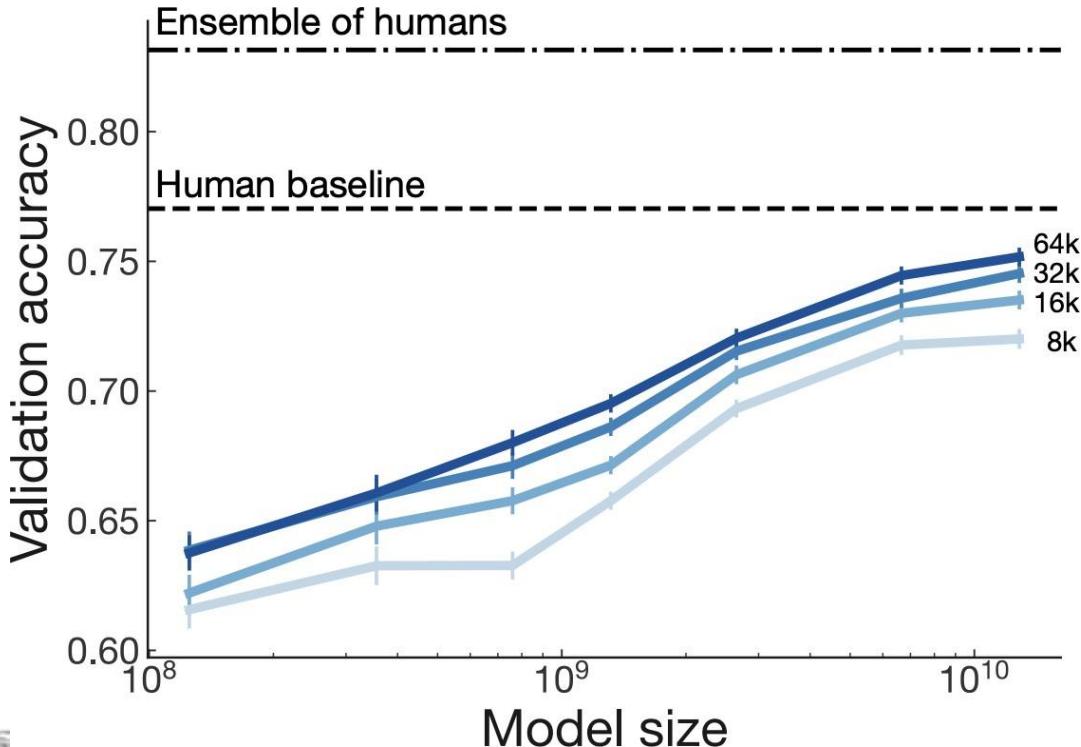
$$J(\phi) = -\mathbb{E}_{(s^+, s^-)} [\log \sigma(R(s^+; \text{prompt}) - R(s^-; \text{prompt}))]$$

"winning" sample "losing" sample



R

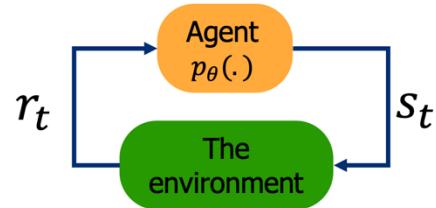
Scaling Reward Models



Large enough reward trained on large enough data approaching human performance.

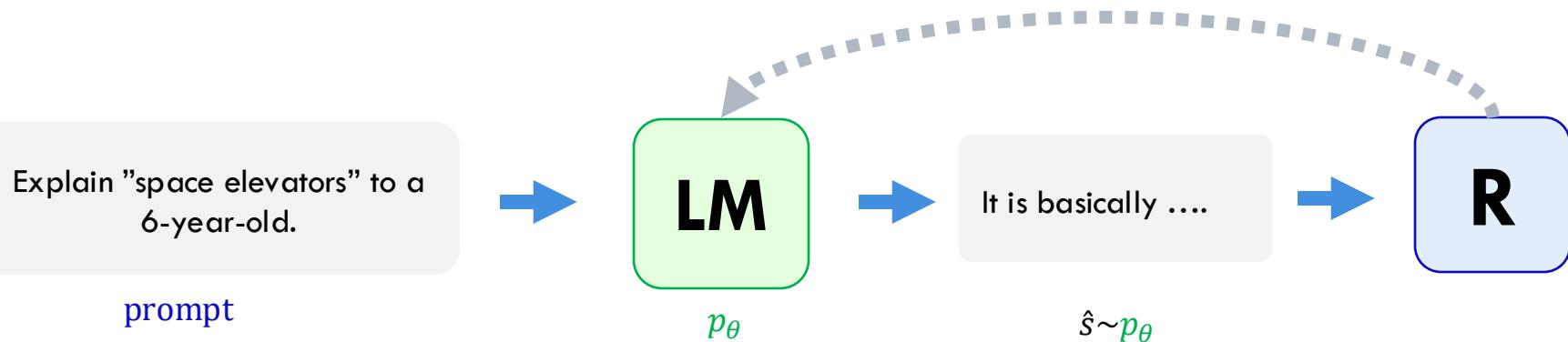
[Stiennon et al., 2020]

Step 2: Optimizing the Policy Function

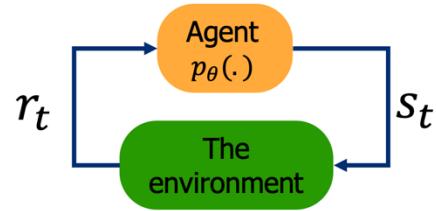


- Policy function := The model that makes decisions (here, generates responses)
- How do we change our LM parameters θ to maximize this?

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$



Step 2: Optimizing the Policy Function



- Policy function := The model that makes decisions (here, generates responses)
- How do we change our LM parameters θ to maximize this?

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

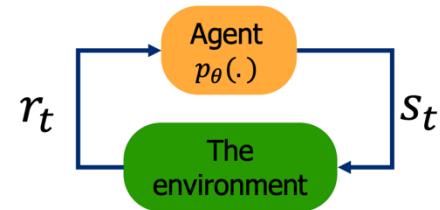
- Let's try doing gradient ascent!

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

How do we estimate
this expectation?

- Turns out that we can write this “gradient of expectation” to a simpler form.

Policy Gradient [Williams, 1992]



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

$$\hat{\theta} = \operatorname{argmax}_{\theta} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

- Let's try doing gradient ascent!

$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_{\theta}} [R(\hat{s}; \text{prompt})]$$

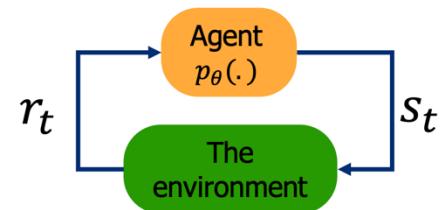
- With a bit of math, this can be approximated as Monte Carlo samples from $p_{\theta}(s)$:

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

Proof next slide; check it later in your own time!

- This is “policy gradient”, an approach for estimating and optimizing this objective.
- Oversimplified. For full treatment of RL see [701.741](#) course, or [Huggingface’s course](#)

Derivations (check it later in your own time!)



- Let's compute the gradient:

$$\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p)] = \nabla_{\theta} \sum_s p_{\theta}(s) R(s; p) = \sum_s R(s; p) \cdot \nabla_{\theta} p_{\theta}(s)$$

Def. of "expectation" Gradient distributes over sum

- Log-derivative trick $\nabla_{\theta} p_{\theta}(s) = p_{\theta}(s) \cdot \nabla_{\theta} \log p_{\theta}(s)$ to turn sum back to expectation:

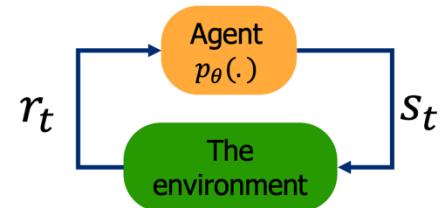
$$\nabla_{\theta} \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p)] = \sum_s R(s; p) p_{\theta}(s) \nabla_{\theta} \log p_{\theta}(s) = \mathbb{E}_{s \sim p_{\theta}(s)} [R(s; p) \nabla_{\theta} \log p_{\theta}(s)]$$

Log-derivative trick

- Approximate this expectation with Monte Carlo samples from $p_{\theta}(s)$:

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

Policy Gradient [Williams, 1992]



- This gives us the following update rule:

$$\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n} \sum_{i=1}^n R(s; p) \nabla_\theta \log p_\theta(s)$$

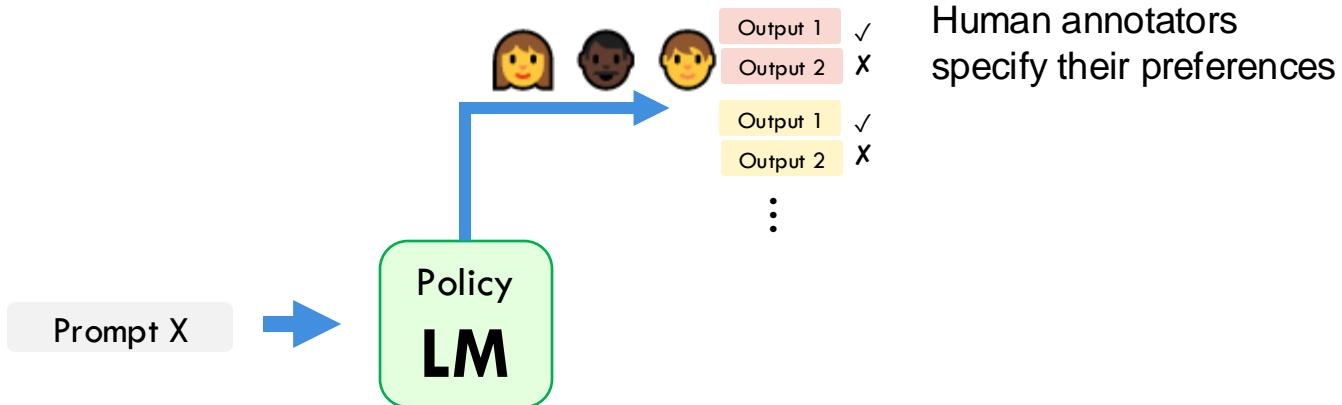
Note, $R(s; p)$ could be any arbitrary, non-differentiable reward function that we design.

- If $R(s; p)$ is **large**, we take proportionately **large** steps to maximize $p_\theta(s)$
- If $R(s; p)$ is **small**, we take proportionately **small** steps to maximize $p_\theta(s)$

This is why it's called "reinforcement learning":
we reinforce good actions, increasing the chance they happen again.

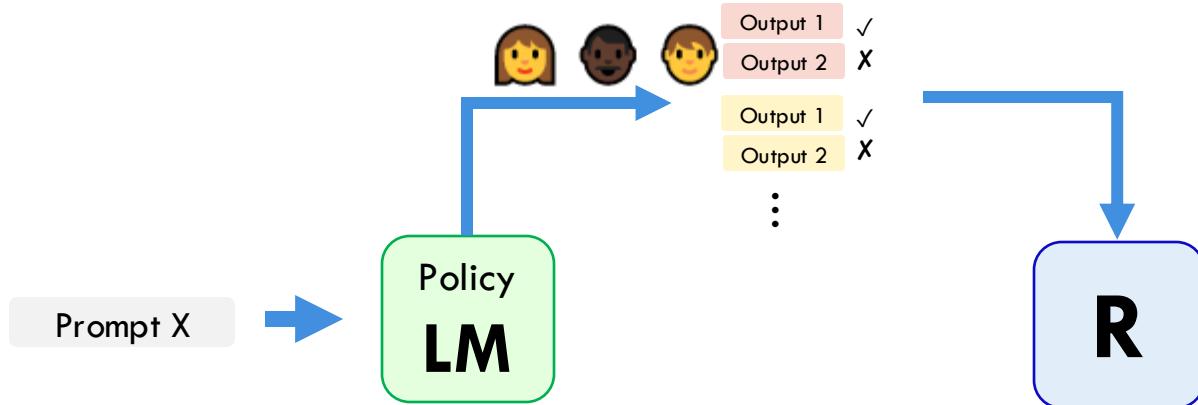
Putting it Together

- First collect a dataset of human preferences
 - Present multiple outputs to human annotators and ask them to rank the output based on preferability



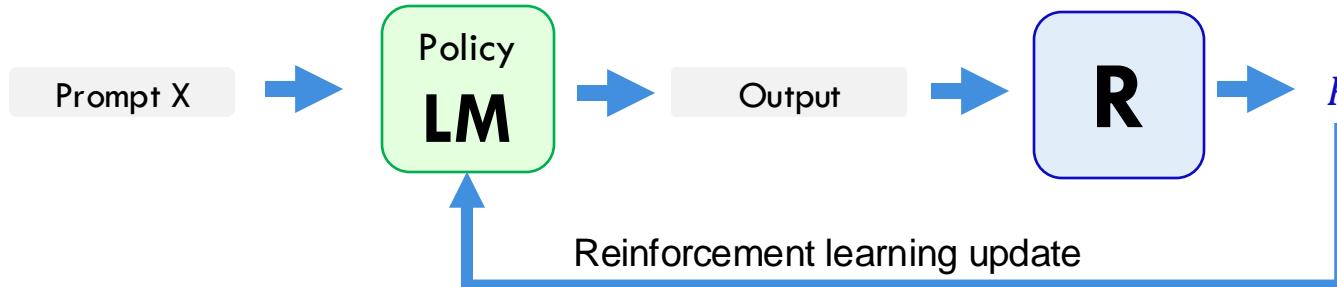
Putting it Together (2)

- Using this data, we can train a reward model
 - The reward model returns a scalar reward which should numerically represent the human preference.



Putting it Together (3)

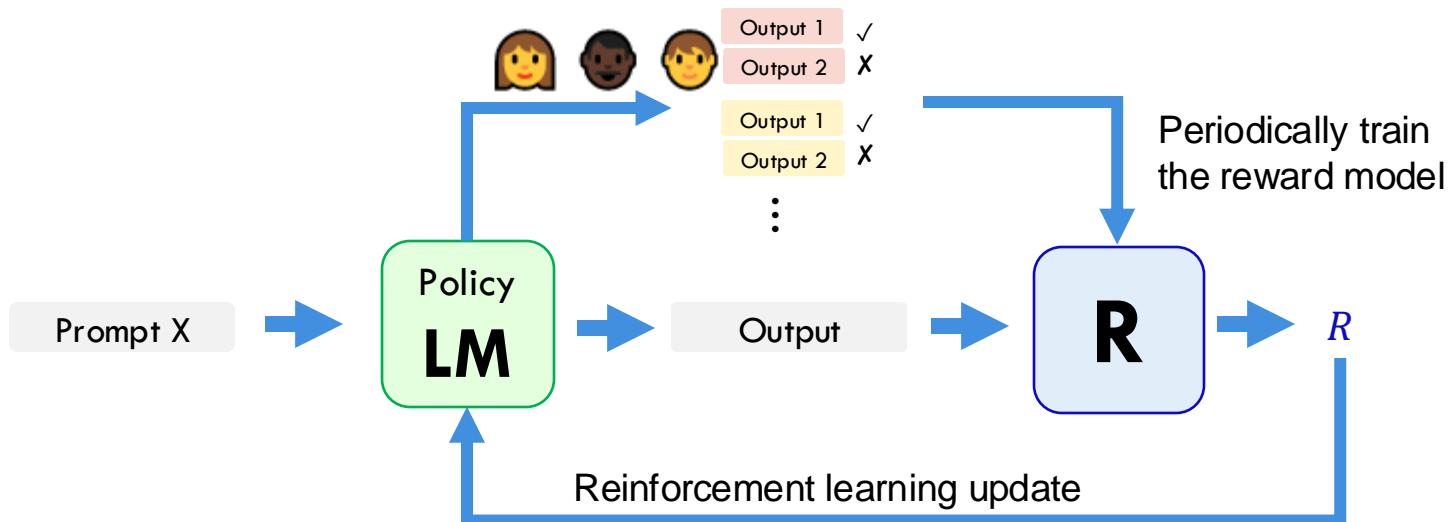
- We want to learn a policy (a Language Model) that optimizes against the reward model



$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_\theta} [R(\hat{s}; p)]$$

Putting it Together (4)

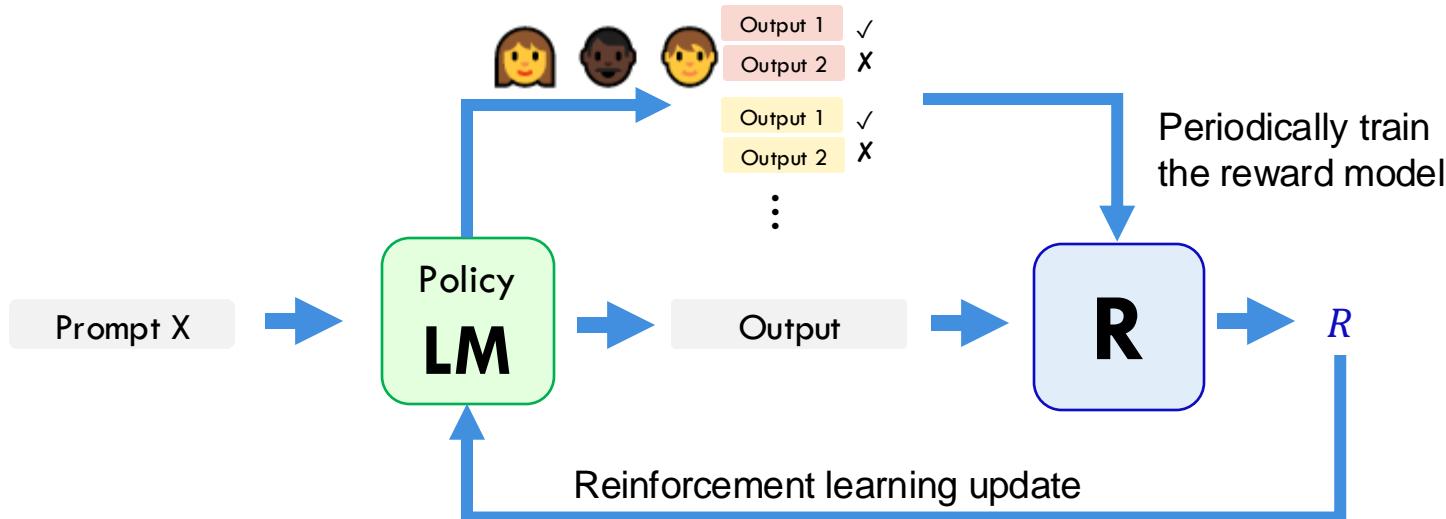
- Periodically train the reward model with more samples and human feedback



$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_\theta} [R(\hat{s}; p)]$$

One missing ingredient

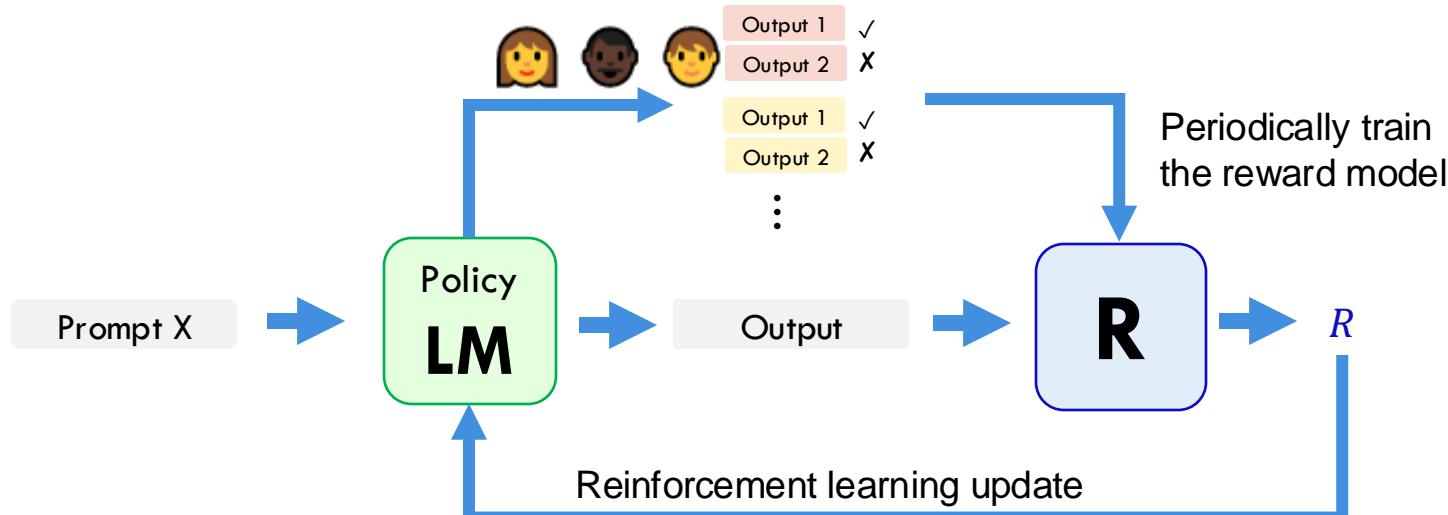
- It turns out that this approach doesn't quite work. (Any guesses why?)
 - The policy will learn to "cheat".



One missing ingredient

How do you resolve this? 😐

- Will learn to produce an output that would get a **high** reward but is **gibberish** or **irrelevant** to the prompt.
- Note, since $R(s; p)$ is trained on natural inputs, it may not generalize to unnatural inputs.



$$\theta_{t+1} \leftarrow \theta_t + \alpha \nabla_{\theta_t} \mathbb{E}_{\hat{s} \sim p_\theta} [R(\hat{s}; p)]$$

Regularizing with Pre-trained Model

- **Solution:** add a penalty term that penalizes too much deviations from the distribution of the pre-trained LM.

pay a price when
 $p^{RL}(s) < p^{PT}(s)$

$$\hat{R}(s; p) := R(s; p) - \beta \log \left(\frac{p^{RL}(s)}{p^{PT}(s)} \right)$$

Policy model

Pre-trained
model

- Prevents the policy model from diverging too far from the pretrained model.
- This regularization is equivalent to adding a KL-divergence regularization term.

RLHF: Putting it All Together [Stiennon et al. 2020]

1. Select a pre-trained generative model as your base: $p_{\theta}^{PT}(s)$
2. Build a reward model $R(s; p)$ that produces scalar rewards for outputs, trained on a dataset of human comparisons
3. Regularize the reward function:

$$\hat{R}(s; p) \coloneqq R(s; p) - \beta \log \left(\frac{p_{\theta}^{RL}(s)}{p_{\theta}^{PT}(s)} \right)$$

4. Iterate:

1. Fine-tune the policy $p_{\theta}^{RL}(s)$ to maximize our reward model $R(s; p)$

$$\theta_{t+1} \leftarrow \theta_t + \alpha \frac{1}{n} \sum_{i=1}^n \hat{R}(s; p) \nabla_{\theta} \log p_{\theta}^{RL}(s)$$

2. Occasionally repeat steps 2-3 to update the reward model.

GPT3 vs. InstructGPT3 (RLHF-ed)

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.

GPT3 vs. InstructGPT3 (RLHF-ed)

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

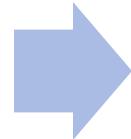
The overall recipe



Pre-train

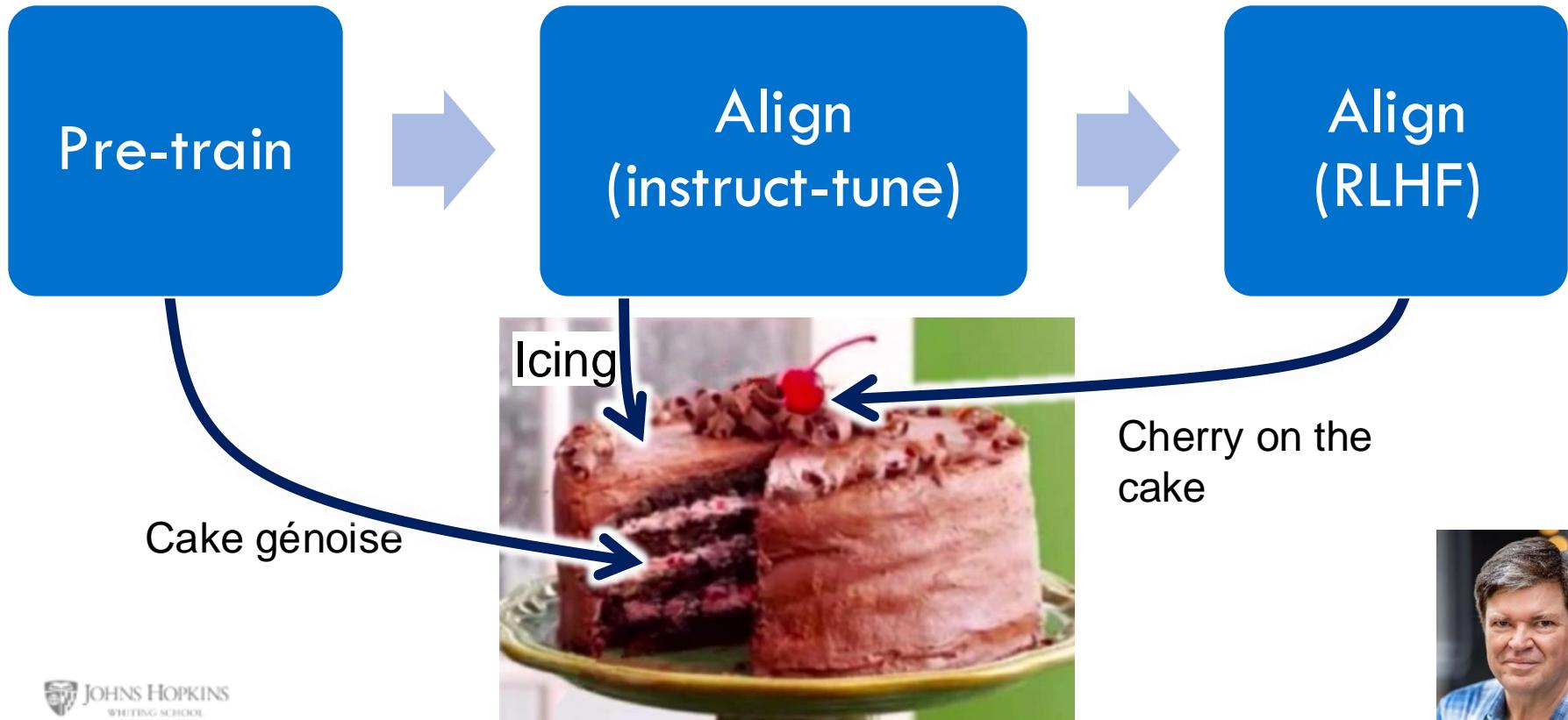


Align
(instruct-tune)



Align
(RLHF)

The overall recipe 🧑‍🍳: Yann's Three-layered cake



Failures and Challenges

RL Failure: Reward Hacking

- “Reward hacking” is a common problem in RL
- Goodhart’s law— when a measure becomes a target, it ceases to be a good measure. (i.e., the proxy ceases to track the actual thing that you care about)

Humanoid: Baseball Pitch - Throw



Throwing a ball to a target.

[<https://openai.com/blog/faulty-reward-functions/>]

[Concrete Problems in AI Safety, 2016]

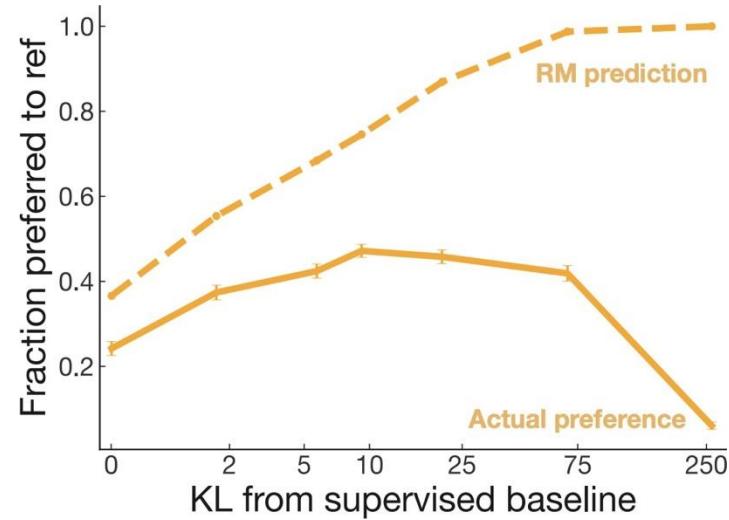
Reward Over-optimization

- Regularizing reward model is a delicate dance balancing:
 - Distance to the prior
 - Following human preferences

$$J(\pi_\theta) = \mathbb{E}_{\hat{s} \sim \pi_\theta} [R(\hat{s}; p)] - \beta D_{KL}(\pi_\theta || \pi_{\text{ref}})$$

- The reward might be over-optimized; the reward might be increasing but the actual preferences may degrade.
- Why does over-optimization happen?
 - The proxy reward is estimated and there are parts of input space that are poorly estimated.

Reward model over-optimization



Direct Policy Optimization

Simplifying RLHF: Direct Policy Optimization (DPO)

- DPO directly optimizes for human preferences
 - avoiding RL and fitting a separate reward model
- One can use mathematical derivations to simplify the RLHF objective to an **equivalent** objective that is **simpler** to optimize.

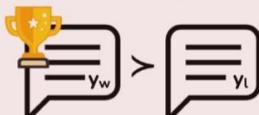
RLHF objective



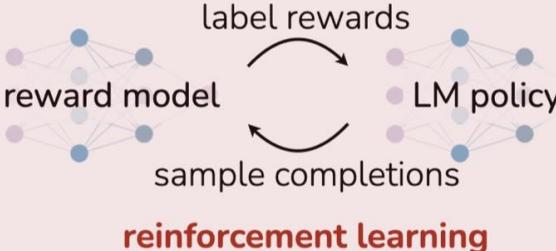
DPO objective

Reinforcement Learning from Human Feedback (RLHF)

x: "write me a poem about
the history of jazz"



maximum
likelihood



Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"



maximum
likelihood



DPO Algorithm

- Algorithm:
 - Create a preference data
 - Optimize the language model to minimize the DPO objective.

Direct Preference Optimization (DPO)

x: "write me a poem about
the history of jazz"



maximum likelihood

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

DPO Limitations

- You're trying to optimize multiple things which can potentially **override** each other.

Direct Preference Optimization (DPO)

x: "write me a poem about the history of jazz"



maximum likelihood

$$\mathcal{L}_{\text{DPO}}(\pi_\theta; \pi_{\text{ref}}) = -\mathbb{E}_{(x, y_w, y_l) \sim \mathcal{D}} \left[\log \sigma \left(\beta \log \frac{\pi_\theta(y_w | x)}{\pi_{\text{ref}}(y_w | x)} - \beta \log \frac{\pi_\theta(y_l | x)}{\pi_{\text{ref}}(y_l | x)} \right) \right]$$

- In practice, when using DPO practitioners constantly monitor these terms.

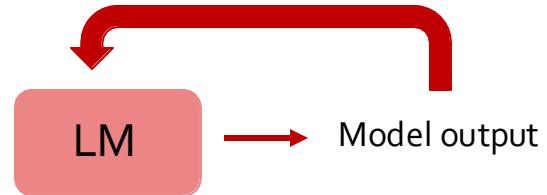


Alignment with Model-Generated (Synthetic) Data



RLHF/Instruction-tuning is Data Hungry

- **Idea:** Use LMs to generate data for aligning them with intents.
 - **Self-Instruct** [[Wang et al. 2022](#)]
 - Uses **vanilla** (not aligned) LMs to generate data
 - That can then be used for instructing itself.
- More related work:
 - Unnatural Instructions [[Honovich et al. 2022](#)] — Similar to “Self-Instruct”
 - Self-Chat [[Xu et al. 2023](#)] — “Self-Instruct” extended to dialogue
 - RL from AI feedback [[Bai et al., 2022](#)],
 - Finetuning LMs on their own outputs [[Huang et al., 2022; Zelikman et al., 2022](#)]



Get humans to write “seed” tasks



- I am planning a 7-day trip to Seattle. Can you make a detailed plan for me?
- Is there anything I can eat for breakfast that doesn't include eggs, yet includes protein and has roughly 700-100 calories?
- Given a set of numbers find all possible subsets that sum to a given number.
- Give me a phrase that I can use to express I am very happy.

175 seed
tasks



Put them your task bank



- I am planning a 7-day trip to Seattle. Can you make a detailed plan for me?
- Is there anything I can eat for breakfast that doesn't include eggs, yet includes protein and has roughly 700-100 calories?
- Given a set of numbers find all possible subsets that sum to a given number.
- Give me a phrase that I can use to express I am very happy.

175 seed
tasks

task pool



Sample and get LLM to expand it

- I am planning a 7-day trip to Seattle. Can you make a detailed plan for me?
- Is there anything I can eat for breakfast that doesn't include eggs, yet includes protein and has roughly 700-100 calories?
- Given a set of numbers find all possible subsets that sum to a given number.
- Give me a phrase that I can use to express I am very happy.

LM

Pre-trained, but **not aligned yet**

- Create a list of 10 African countries and their capital city?
- Looking for a job, but it's difficult for me to find one. Can you help me?
- Write a Python program that tells if a given string contains anagrams.

175 seed tasks

task pool



LM suggests
new tasks



Get LLM to answers the new tasks

- Task: Convert the following temperature from Celsius to Fahrenheit.
- Input: 4 °C
- Output: 39.2 °F
- Task: Write a Python program that tells if a given string contains anagrams.

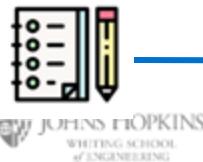
LM

Pre-trained, but **not aligned yet**

- Input: -
- Output:
`def isAnagram(str1, str2): ...`

175 seed tasks

task pool

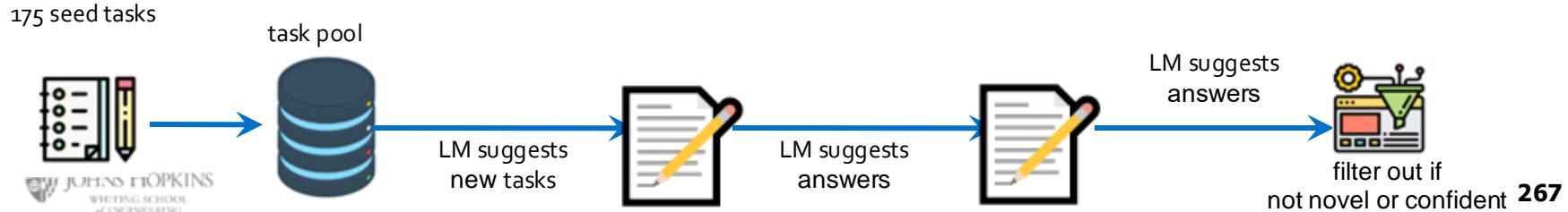


LM suggests
new tasks

LM suggests
answers

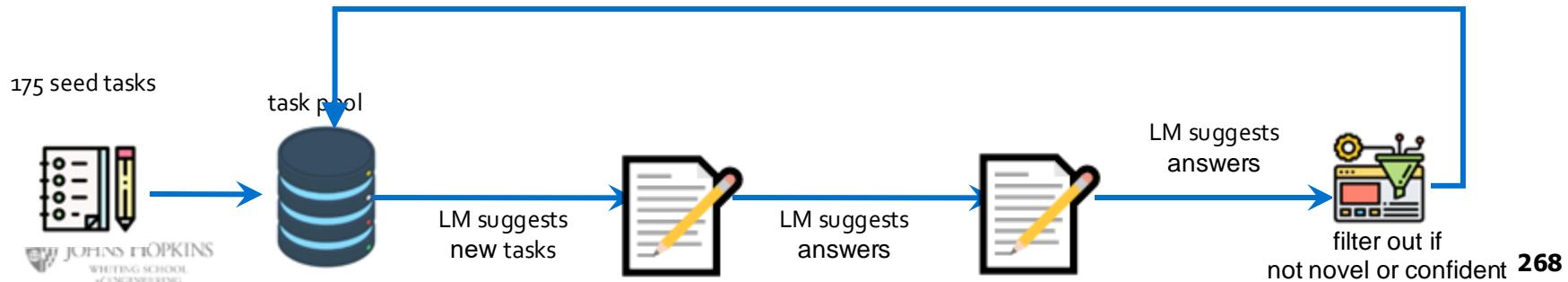
Filter tasks

- Drop tasks if LM assigns **low probability** to them.
- Drop tasks if they have a high overlap with one of the existing tasks in the task pool.
 - Otherwise, common tasks become more common — **tyranny of majority**.



Close the loop

- Add the filtered tasks to the task pool.
- Iterate this process (generate, filter, add) until yield is near zero.



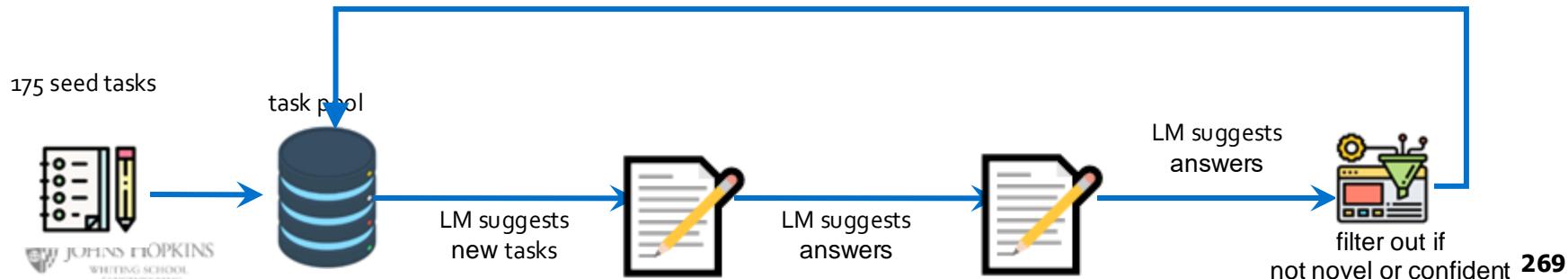
Self-Instructing GPT3 (base version)

- **Generate:**

- GPT3 ("davinci" engine).
- We generated 52K instructions and 82K instances.
- API cost ~\$600

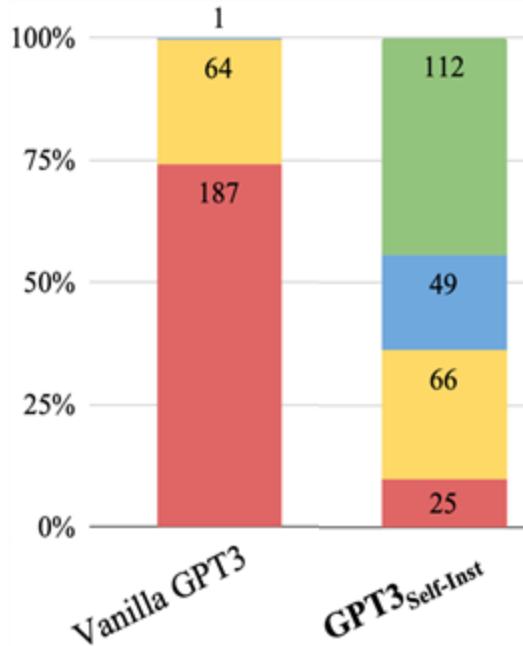
- **Align:**

- We finetuned GPT3 with this data via OpenAI API (2 epochs). **
- API cost: ~\$338 for finetuning



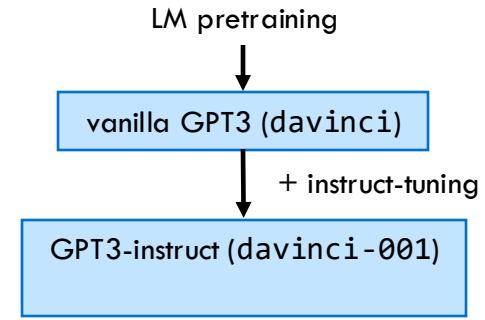
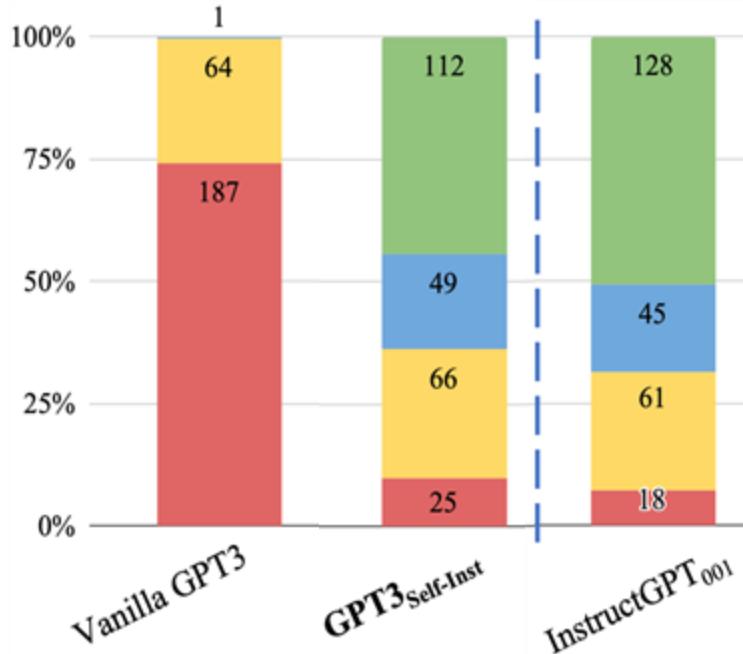
Evaluation on User-Oriented Instructions

- A: correct and satisfying response ■ B: acceptable response with minor imperfections
- C: responds to the instruction but has significant errors ■ D: irrelevant or invalid response



Evaluation on User-Oriented Instructions

- A: correct and satisfying response ■ B: acceptable response with minor imperfections
- C: responds to the instruction but has significant errors ■ D: irrelevant or invalid response



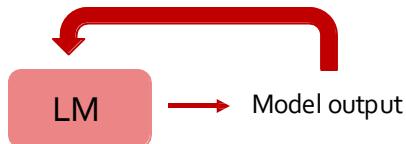
Noisy, but diverse “self-instruct” data ~ thousands of clean human-written data

Summary Thus Far

- ❖ Evidence suggest that we probably can reduce the reliance on **human** annotations in the “alignment” stage
 - **Data diversity** seems to be necessary for building successful generalist models.
- ❖ Self-Instruct: Rely on creativity induced by an LLM’s themselves.
 - Applicable to a broad range of LLMs.
 - Several open-source models utilize “Self-Instruct” data.

Impact: Learning from AI Feedback

- Open-source models adopted Self-Instruct data generation.
 - Alphaca, Zephyr, etc. [Taori et al. 2023; Tunstall et al. 2023]
- LLMs used directly as a reward during alignment, skipping the data generation.
[Lee et al. 2023; many others]



RLAIF: Scaling Reinforcement Learning from Human Feedback with AI Feedback

Harrison Lee, Samrat Phatale, Hassan Mansoor, Thomas Mesnard, Johan Ferret, Kellie Lu,
Colton Bishop, Ethan Hall, Victor Carbune, Abhinav Rastogi, Sushant Prakash
Google Research

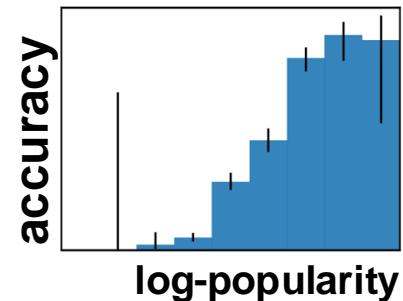
{harrisonlee,samratph,hassan}@google.com

Training LLMs with LLM Feedback: The Bottleneck

- Model feedback is a powerful idea, but ...
- It has many limitations ...
 - It amplifies existing biases.
 - It is still confined to the [implicit] boundaries defined by the its prompts.
 - LLMs work best in high-data regime. They fail when data is thin.

[Mallen et al. 2022; Razeghi et al. 2022; many others]

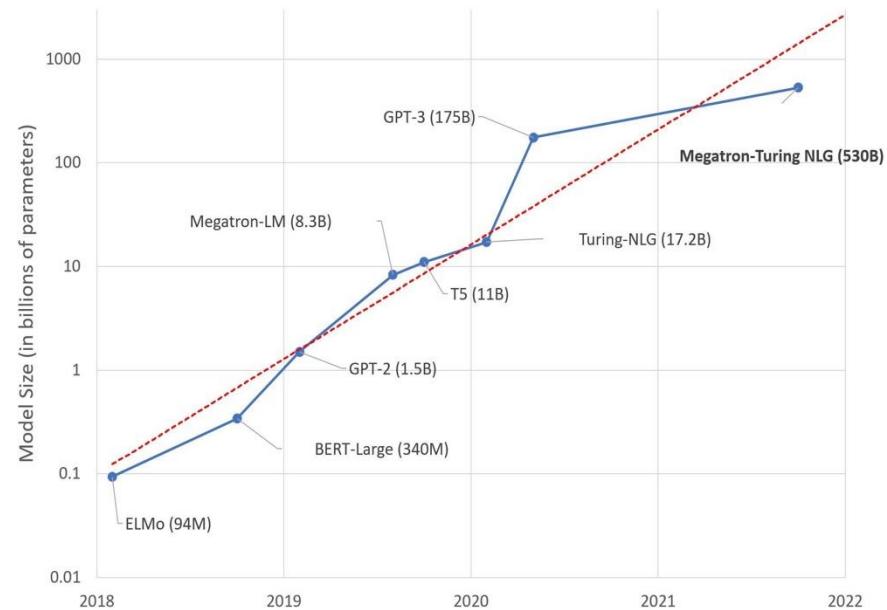
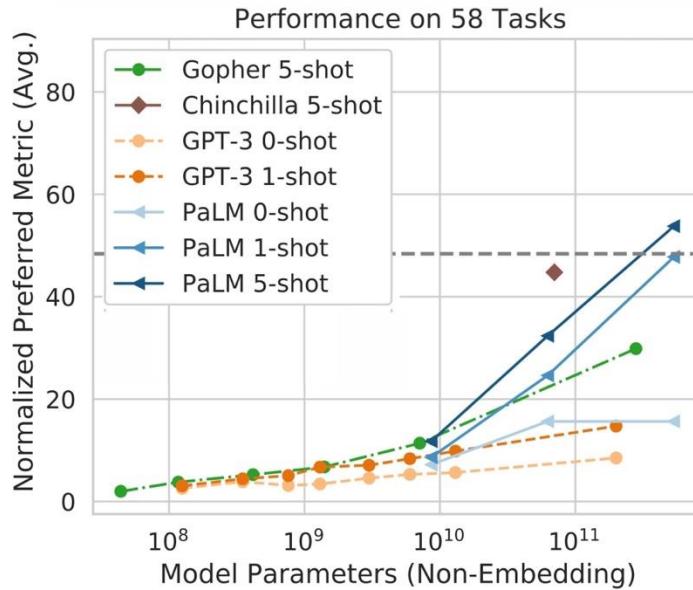
- Training with self-feedback is unlikely to be the way to the moon!



Brief on “Scaling”

Model Size vs. Accuracy

Photo credit: PaLM, Chowdhery et.al, 2022



Larger LMs \Rightarrow better zero/few-shot performance

"More is Different"

4 August 1972, Volume 177, Number 4047

SCIENCE

- The idea that complex physical systems can behave in ways that can't be understood by the laws that govern their microscopic parts.
- Anderson also gives an example of "More is Different" at the molecular level.
 - He describes a peculiar broken symmetry that appears in larger-scale molecules, which seems to go against a law defined at the smaller scale.
 - This broken symmetry is a new effect that appears when the scale changes.
- Anderson argues that new properties appear at each level of complexity.
 - For example, although chemistry is subject to the laws of physics, we can't infer chemistry from our knowledge of physics.

More Is Different

Broken symmetry and the nature of the hierarchical structure of science.

P. W. Anderson

The reductionist hypothesis may still be a topic for controversy among philosophers, but among the great majority of active scientists I think it is accepted explanation of phenomena in terms of known fundamental laws. As always, distinctions of this kind are not unambiguous, but they are clear in most cases. Solid state physics, plasma physics, and perhaps

less relevance they seem to have to the very real problems of the rest of science, much less to those of society.

The constructionist hypothesis breaks down when confronted with the twin difficulties of scale and complexity. The behavior of large and complex aggregates of elementary particles, it turns out, is not to be understood in terms of a simple extrapolation of the properties of a few particles. Instead, at each level of complexity entirely new properties appear, and the understanding of the new behaviors requires research which I think is as fundamental in its nature as any other. That is, it seems to me that one may array the sciences roughly linearly in a hierarchy, according to the idea: The elementary entities of science X obey the laws of science Y.

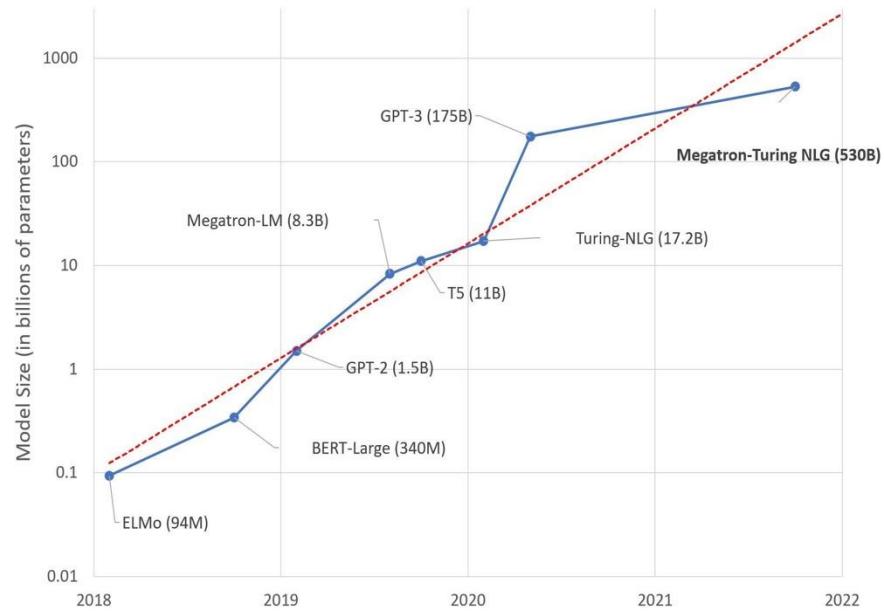


Scaling Laws

- **Hypothesis:** there are fundamental principles that govern effective scaling
- **Importance:** understanding these “laws” would allow us to find optimal models for a given data/compute budget.
- Think of Newton’s laws
 - Provide the basis for understanding and analyzing the motion of objects in the physical world
 - Can be used to calculate the trajectory of a rocket, the speed of a car, or the motion of a planet.

Constraints of Real World

- Even massive companies have their own constraints.
- Examples of constraints:
 - The total amount of data
 - The total computing budget.
 - Time
 -
- **Given a set of constraints, how do you choose which LM to train?**
 - Note, trial and error is wasteful.



Quantifying Computation Cost of Models

- How do you compute computational cost of a single-layer NN with one matrix multiplication?

FLOPS

- Floating point operations per second (FLOPS, flops or flop/s)
 - Each FLOP can represent an addition, subtraction, multiplication, or division of floating-point numbers,
- We want to compute the total FLOP of a model (e.g., Transformer)
 - Provides a basic approximation of computational costs associated with that model.
- Our models are just a bunch of matrix multiplications.
Let's estimate the FLOPS of matrix operations... 😊

FLOPS: Matrix Multiplication

Inference FLOPs for multiplying by a matrix W
 $\approx 2 \times (\text{batch size}) \times (\text{size of } W)$

Training FLOPs for multiplying by a matrix W
 $\approx 6 \times (\text{batch size}) \times (\text{size of } W)$

(Why? Think about FLOPS for forward and backward separately ...)

Computing the computational cost of Transformer

Transformer FLOPs: The Quick Estimate

- The Weight FLOPs Assumption
 - The FLOPs that matter the most are weight FLOPs, that is ones performed when intermediate states are multiplied by weight matrices.
 - The weight FLOPs are the majority of Transformer FLOPs
 - We can ignore FLOPs for
 - Bias vector addition
 - layer normalization
 - residual connections
 - non-linearities
 - Softmax

Transformer FLOPs: The Quick Estimate

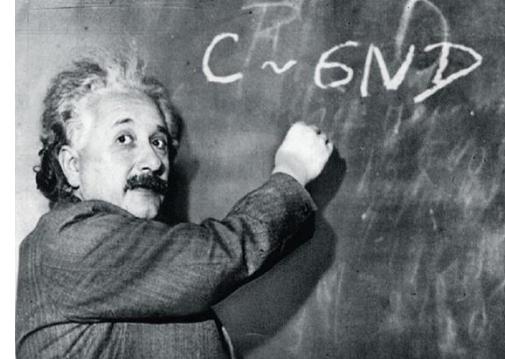
- Let N be number of parameters (the sum of size of all matrices)
- Let D be the number of tokens in pre-training dataset.
- The total cost of pre-training on this dataset is:

$$C \sim 6ND$$

- We are ignoring the non-matrix operators (normalization, non-linearities, etc.)

Transformer FLOPs

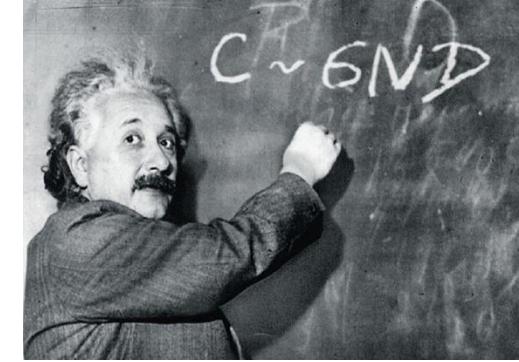
- Given the pre-training data with 400B tokens.



n_{layer}	d_{model}	Parameters (N)	Training FLOPs
4	512	13M	3.0e19
6	768	42M	1.0e20
10	1280	197M	4.7e20
16	2048	810M	1.9e21
24	3072	2.7B	6.5e21
40	5120	13B	3.0e22
64	8192	52B	1.2e23

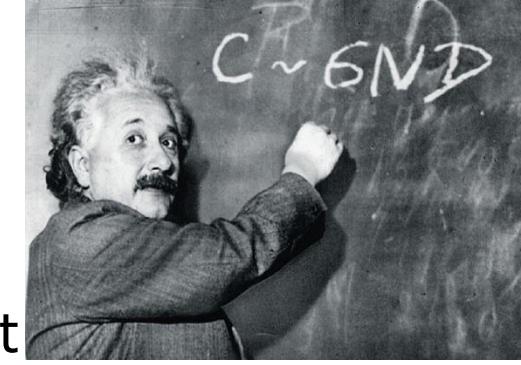
Training cost (FLOPs):

$$\begin{aligned}C &\approx 6ND \\&= 6 \times (400 \times 10^9) \\&\times (52 \times 10^9) \\&= 1.24 \times 10^{23}\end{aligned}$$



Estimating training time

- This is a very practical question in real world.
- We will use our formula earlier to estimate training time.
- Consider HyperCLOVA, an 82B parameter model that was pre-trained on 150B tokens, using a cluster of 1024 A100 GPUs.



Estimating training time

- Consider HyperCLOVA, an 82B parameter model that was pre-trained on 150B tokens, using a cluster of 1024 A100 GPUs.
- Training cost (FLOPs):

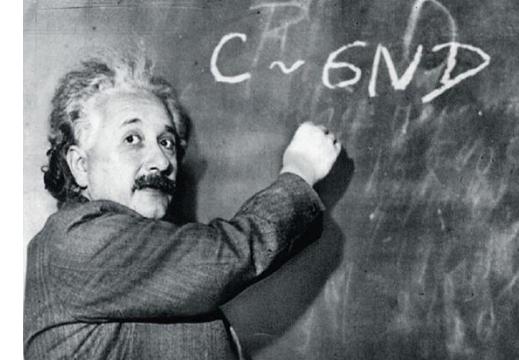
$$\begin{aligned} C &\approx 6ND \\ &= 6 \times (150 \times 10^9) \times (82 \times 10^9) = 7.3 \times 10^{22} \end{aligned}$$

- The peak throughput of A100 GPUs is 312 teraFLOPS or 3.12×10^{14} .
- **How long would this take?**

$$\text{Duration} = \frac{\text{model compute cost}}{\text{cluster throughput}} = \frac{7.3 \times 10^{22}}{3.12 \times 10^{14} \times 1024} = 2.7 \text{ days}$$

Intensive Study on HyperCLOVA: Billions-scale Korean Generative Pretrained Transformers, 2021

<https://arxiv.org/pdf/2109.04650.pdf>



Estimating training time

- How long would this take?

$$\text{Duration} = \frac{\text{model compute cost}}{\text{cluster throughput}} = \frac{7.3 \times 10^{22}}{3.12 \times 10^{14} \times 1024} = 2.7 \text{ days}$$

- According to the white paper, training took 13.4 days. Our estimate is 5 times off (why?), but we did get the order of magnitude right! 🤝

Factors We Did Not Consider

- Note that these estimates can be slightly off in practice
 - Theoretical peak throughput is not achievable with distributed training.
(unless your model only does large matrix multiplications).
 - We ignored many additional operations like softmax, ReLU/GeLU activations, self-attention, Layer Norm etc.
 - Training divergence and restarting from earlier checkpoints are not uncommon.
- There are various factors that contribute to computation latency
 - Communication latency, memory bandwidth, caching, etc.
 - See <https://kipp.ly/transformer-inference-arithmetic/> for an excellent discussion.

Summary

- One can measure the computational cost of training neural networks in terms of FLOPS.
- Such estimates allow you to estimate the training time of your model, given your GPU specs.
- What else can we do?

Optimal Scaling

Optimal Scaling

- **A real problem:** Your boss gives you a compute budget \$\$. What is the best model you can build with this budget?
- We know from the literature that **larger** models generally lead to **better** models.
 - Does that mean that you should aim to build the largest model possible?
- Intuitively, if you choose a model that is **too large** for your budget, you need to **cut your training** cycles that **may reduce its quality**.
- **This chapter:** principled approach to selecting optimal data/model scaling.

Scaling

Experimental Setup:

- Pre-train various models of different sizes
- Plot their validation loss throughout training

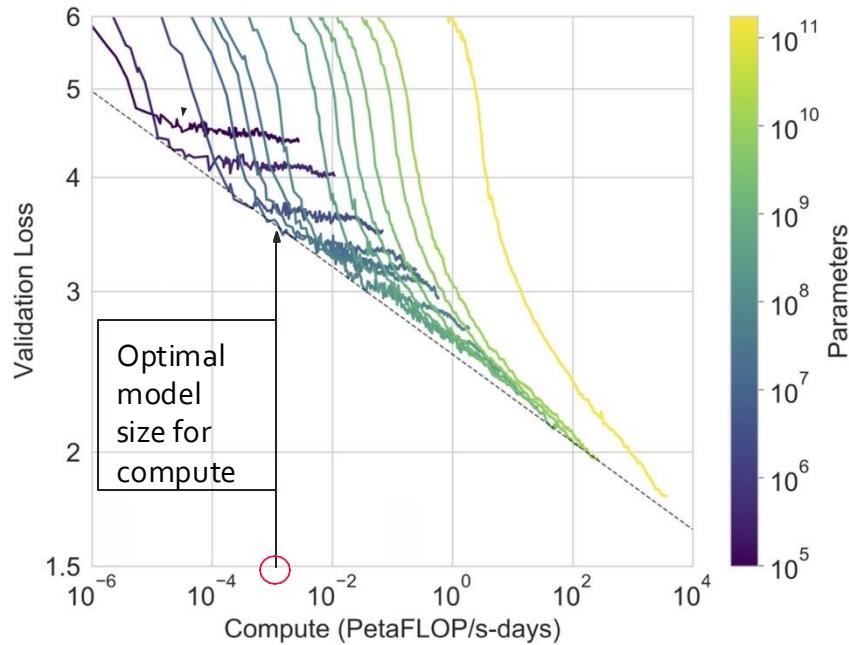


Photo credit: GPT3, Brown et. al., 2020

Scaling

- **Smaller** models don't have enough capacity to utilize the extra compute. They plateau early.
- **Larger** models are initially slower to train, but with more compute they reach lower losses.

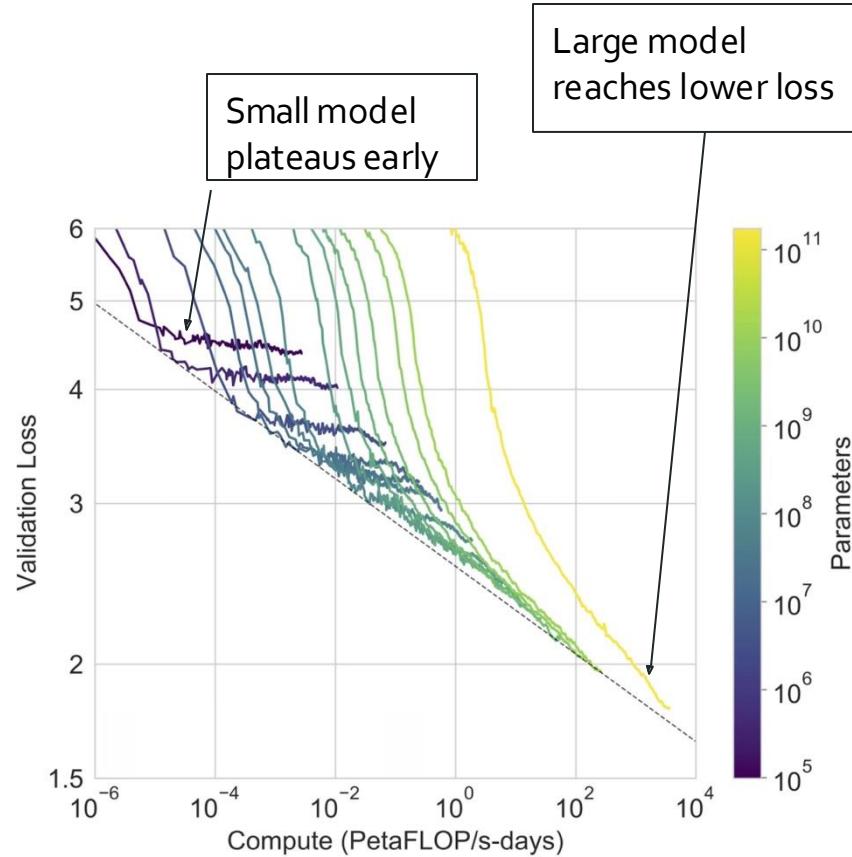


Photo credit: GPT3, Brown et.al., 2020

Scaling - Optimal Model Size

- Let's say our compute budget is $C = 10^{-2}$ PetaFLOPs-days.
 - The **optimal model** is the one that plateaus at exactly C .
 - If we train a **larger** model than optimality point, we won't reach the best performance.
 - If we train a **smaller** model the performance wouldn't be optimal
- The idea of "**optimal model size for given compute**" was introduced by Kaplan et. al.
- If we have the equations ("laws") describing the behavior, we can compute it **analytically**.

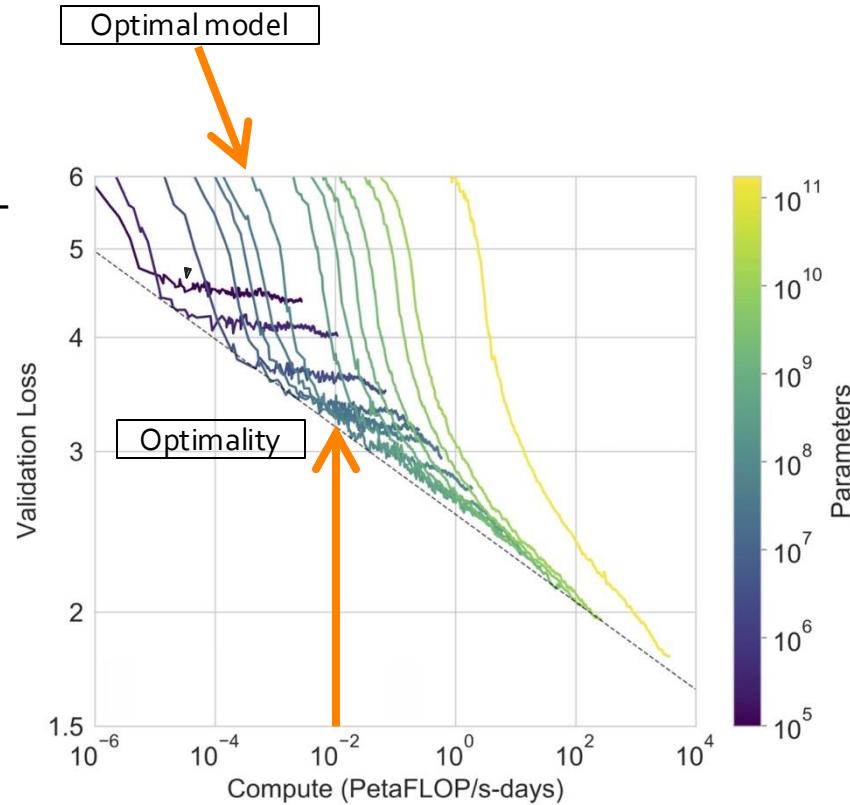


Photo credit: GPT3, Brown et. al., 2020

N: number of model parameters
C: compute
D: dataset size

Scaling Laws: Kaplan et al.

- Optimal model size and optimal number of tokens, for a given compute budget

Kaplan et. al. 2020

$$N_{\text{opt}} \propto C^{0.73}$$

$$D_{\text{opt}} \propto C^{0.27}$$

N_{opt} exponent $\gg D_{\text{opt}}$ exponent

- Takeaway:** grow the model size faster than growing the number of tokens.
 - Example:** Given 10x compute, increase N by [MASK] and D by [MASK]

However ...

- In 2022 a Hoffmann et al. from DeepMind showed a different set of scaling laws.

Scaling Laws: Hoffmann et al.

N: number of model parameters
C: compute
D: dataset size

- Optimal model size and optimal number of tokens, for a given compute budget

Kaplan et. al. 2020	$N_{\text{opt}} \propto C^{0.73}$	$D_{\text{opt}} \propto C^{0.27}$
Hoffmann et. al., 2021	$N_{\text{opt}} \propto C^{0.5}$	$D_{\text{opt}} \propto C^{0.5}$

$$N_{\text{opt}} \text{ exponent} \cong D_{\text{opt}} \text{ exponent}$$

- Compute and tokens should increase **at the same rate**.
 - Example 1:** Given 10x compute, grow N by 3.2x and D by 3.2x
 - Example 2:** Given 100x compute, grow N by [MASK] and D by [MASK]

Recap

- We used to train “oversized” and “under-trained” models.
- You should scale your model at the same rate as your data.
- For example, if you get a 100x increase in compute,
 - you should make your model 10x bigger and your data 10x bigger.

A Word of Caution

- While we kept referring to these as “law”, one should take them with grain of salt.
- There are various confounding factors here:
 - Different optimizer: AdamW vs. Adam vs. others
 - Different tokenizers
 - Different numerical representation (e.g., bfloat16 vs float32)
 -

Is Scale All You Need?

Is Scale All We Need?

- For what purpose?
 - For building useful applications (answering simple questions, translating simple sentences) we already have good models. Not our focus.
 - General intelligence: think of an assistant that is always with you, knows what you want, assists you with anything you need.



Nando de Freitas 
@NandoDF

...

Solving these scaling challenges is what will deliver AGI. Research focused on these problems, eg S4 for greater memory, is needed. Philosophy about symbols isn't. Symbols are tools in the world and big nets have no issue creating them and manipulating them 2/n

4:50 AM · May 14, 2022 · Twitter for iPhone

23 Retweets 5 Quote Tweets 153 Likes

Do you agree with Nando?

Argument: Not Enough Compute

Limitations regarding compute:

- There is simply not enough compute available.
 - Models have been increasing 10x every year
 - Moore's law: # of transistors on an IC doubles about every two years.
 - There are physical limits to how much faster computers can get.
- Even if we have the compute, scaling the compute will be quite costly.
- Scaling compute is simply infeasible. [QED]

Are you convinced?

Rebutting “Not Enough Compute”

- On insufficiency of compute resource:
 - Hardware technologies continue to progress at a rapid pace.
 - Huang’s law: advancements in GPUs happen at much faster rate than what Moore predicted.
 - So much potentials in parallel computing.
- On cost-[in]efficiency of scaling:
 - While models like GPT3 cost a lot (monetary or otherwise), their availability prevent training MANY smaller, mediocre models.
 - Therefore, it might be that the net cost of scaling large models is negative.
 - It is the case within Microsoft according to its CTO, Kevin Scott.

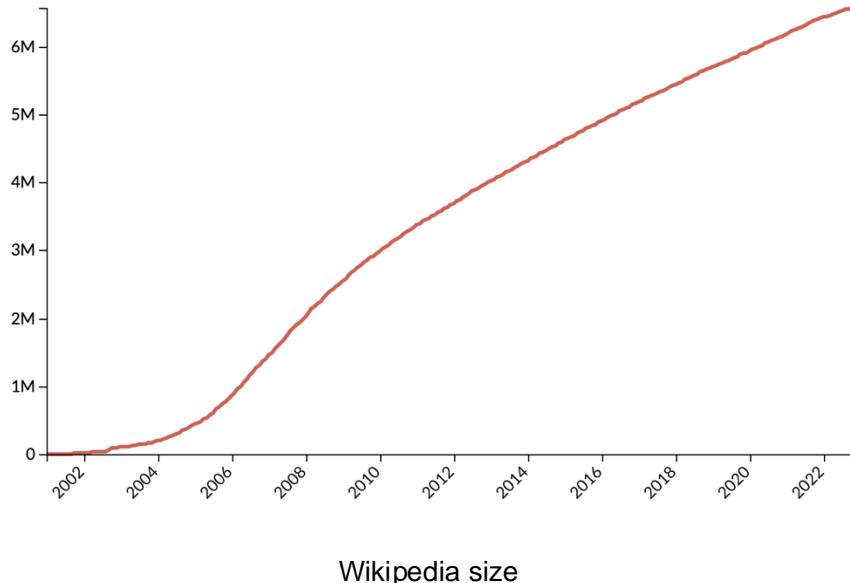
Argument: Not Enough Data

- Hoffmann et al showed that, to be compute-optimal, **model size and training data must be scaled equally**.
- It shows that existing LLMs are severely data-starved and under-trained.
- Given the new scaling law, even if you pump a billions of params into a model, the gains will **not** compensate for more training tokens.
- There is simply not enough [language] data. [QED]

Are you convinced?

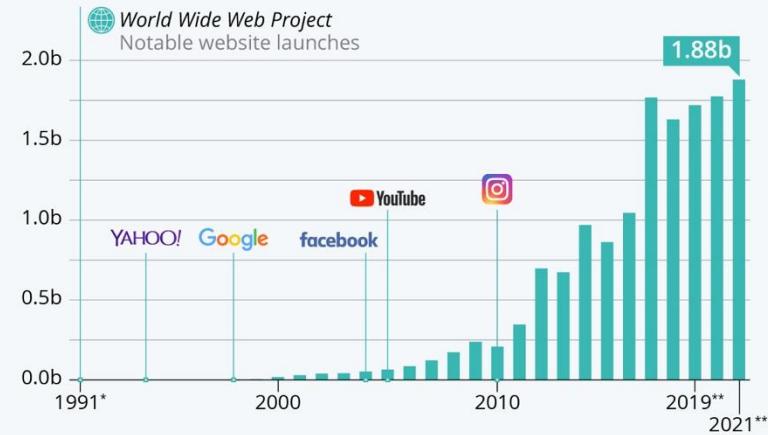
Rebutting “Not Enough Data”

- Data is growing exponentially (?)



How Many Websites Are There?

Number of websites online from 1991 to 2021



* As of August 1, 1991.

** Latest available data for 2019: October 28, for 2020: June 2, for 2021: August 6.

Source: Internet Live Stats

Rebutting “Not Enough Data” (2)

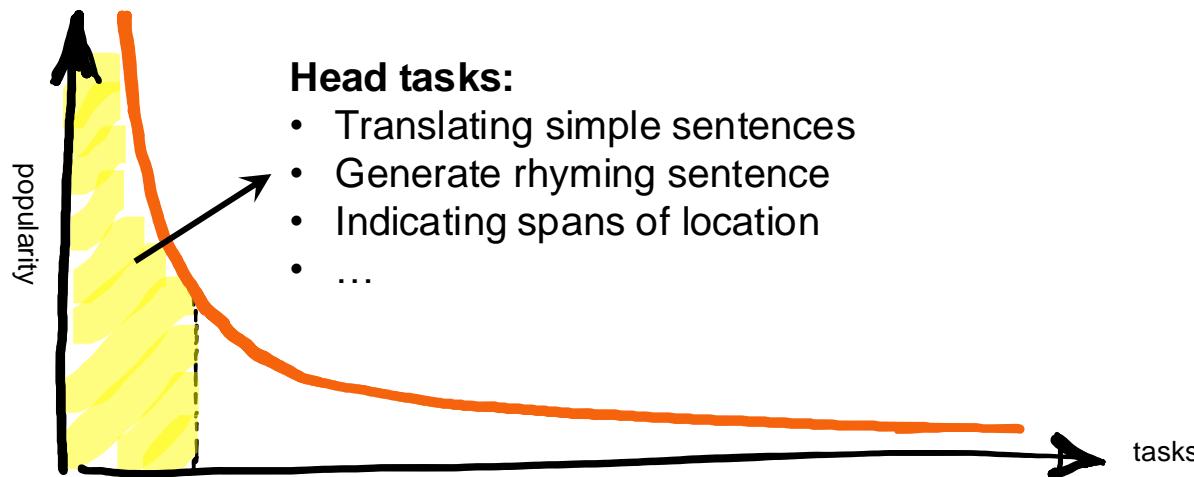
- You can harness data from other modalities.
 - For example, to get more text data we can build a solid speech processor model that converts speech to text.
 - (aside: more than 80% if internet traffic is video)

SKYQUEST

Global Online Video Platforms Market Drives over 80% of Total Internet Traffic ||
Skyquest Technology
 - (aside2: is that why OpenAI built Whisper?!)

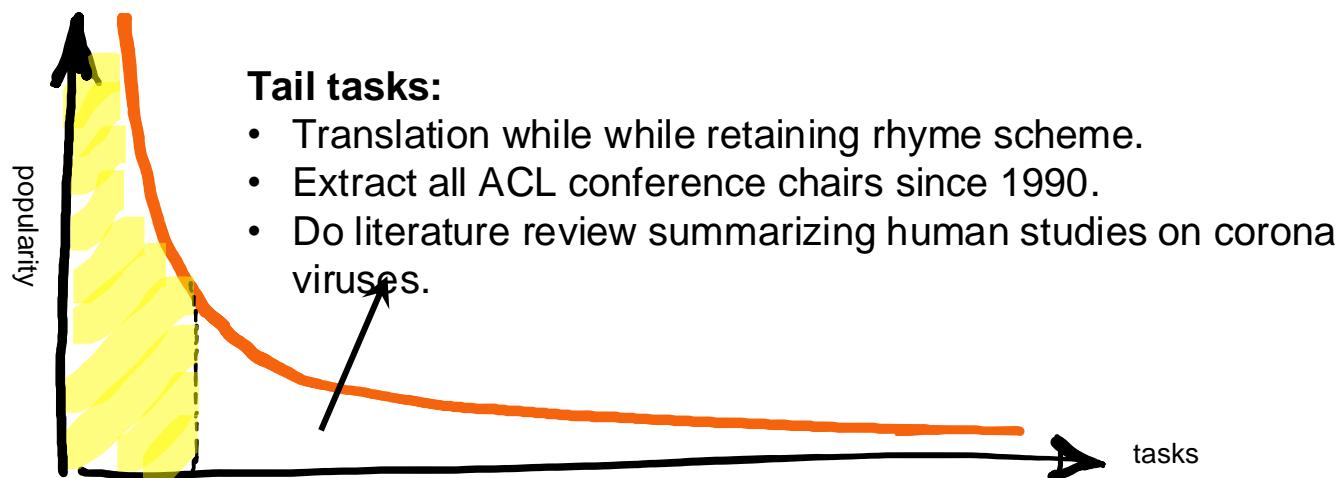
Argument: Scale is Not all You Need Because of Tail Phenomena

- Tail phenomena will never go away!



Argument: Scale is Not all You Need Because of Tail Phenomena

- Tail phenomena will never go away!



Argument: Scale is Not all You Need Because of Tail Phenomena

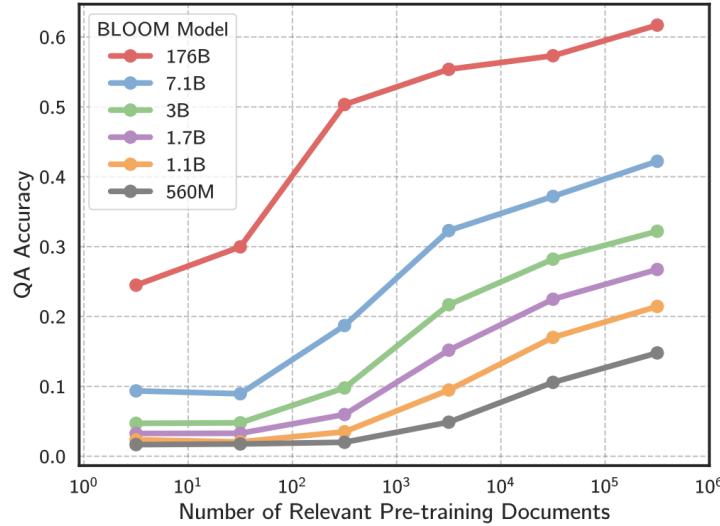
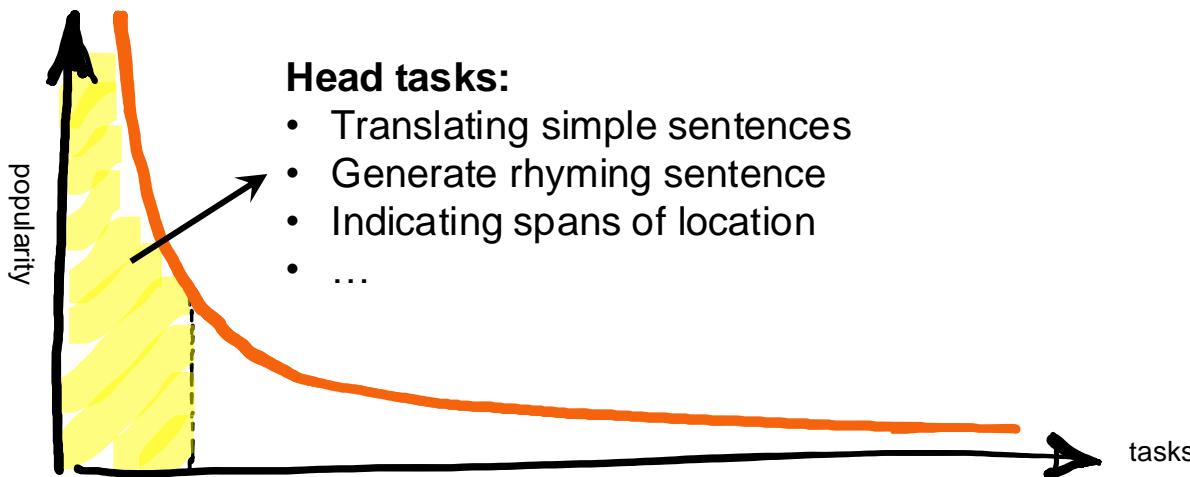


Figure 1: Language models struggle to capture the long-tail of information on the web. Above, we plot accuracy for the BLOOM model family on TriviaQA as a function of how many documents in the model's pre-training data are relevant to each question.

Argument: Scale is Not all You Need Because of Tail Phenomena

- Hence, scale won't solve the tail phenomena. [QED]

Let's do a poll!



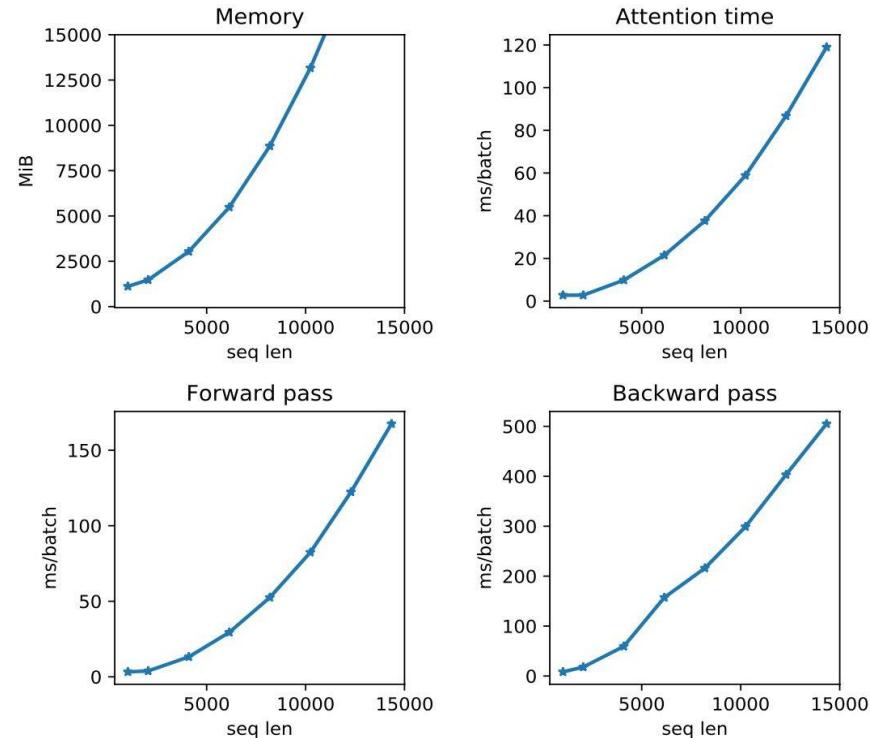
Argument: Scale is Not all You Need Because of Tail Phenomena

- How do you rebut this??

Long Context: Efficiency and Generalization

Transformer LMs and Long Inputs

- **Length generalization:** Do Transformers work **accurately** on long inputs?
 - We will read papers on this topic.
- **Efficiency considerations:** How **efficient** are LMs on long inputs?



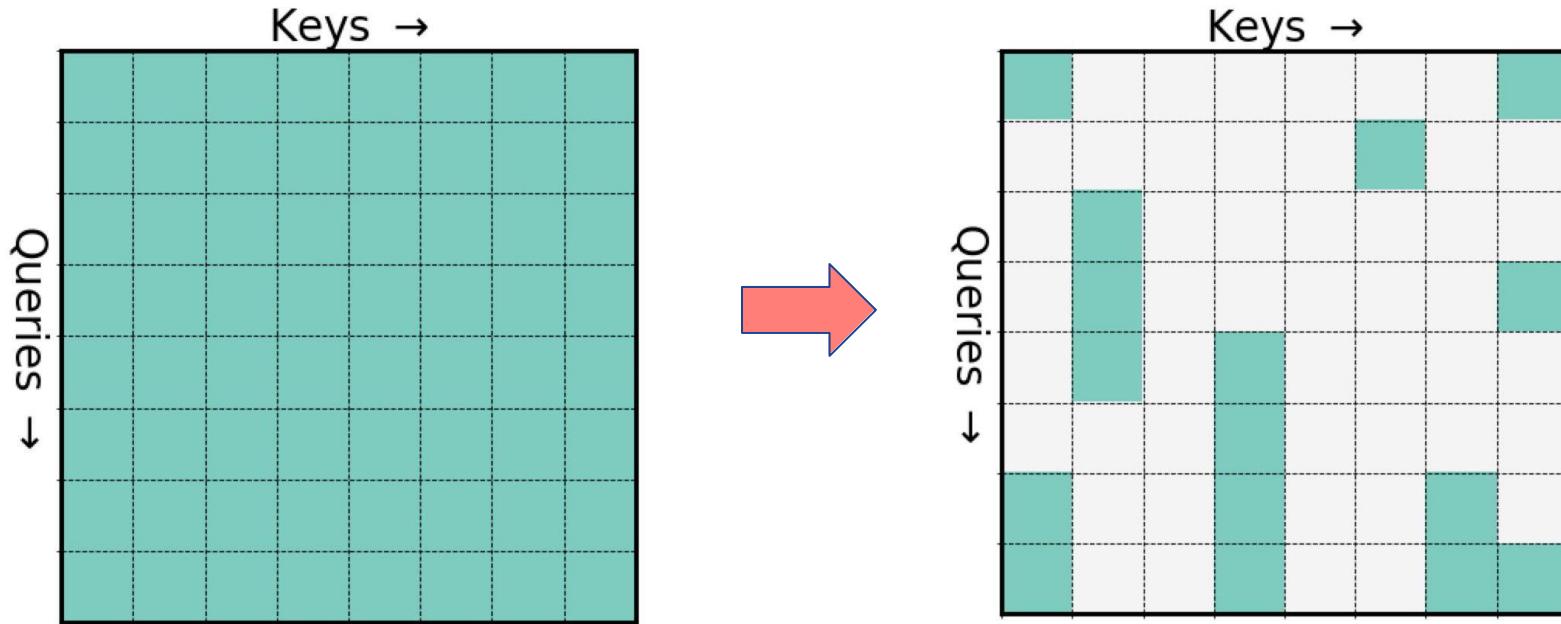


Efficiency via sparsity



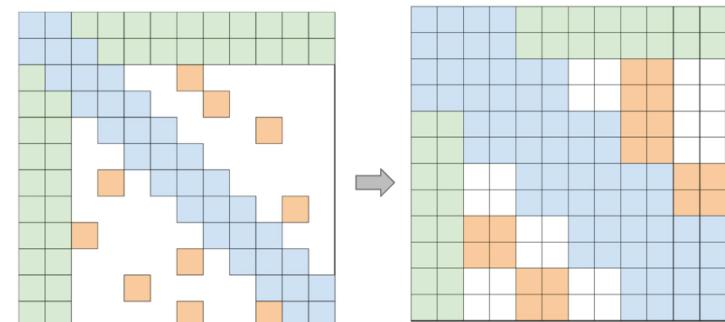
Sparse Attention Patterns

- The idea is to make the attention operation sparse



Sparse Attention Patterns: Challenge

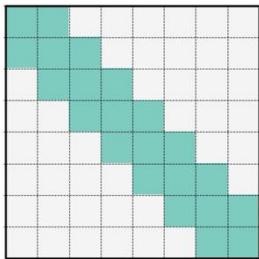
- Ok sparsity is great, but how to efficiently implement this?
- **Challenge:** Arbitrary sparse matrix multiplication is not supported in DL libraries
- **A solution:** Perform computations in blocks
- There are libraries for implementing blockified sparse matrix multiplication.
 - Can be hardware specific
 - Block Sparse ([Gray et al., 2017](#))
 - TVM toolkit ([Chen et al., 2018](#))
 - cuSPARSE



Pre-specified Sparsity Patterns

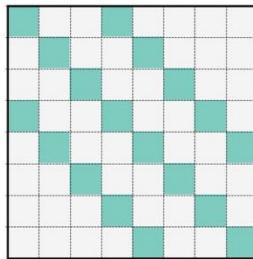
- A variety of patterns has been explored in the past work
 - Longformer ([Beltagy et al., 2020](#)), Sparse Transformer ([Child et al., 2019](#)), ...

Sliding window



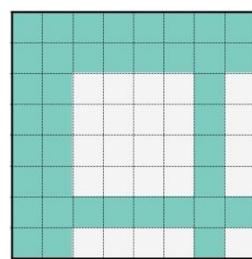
Sparse Transformer
Longformer

Dilated



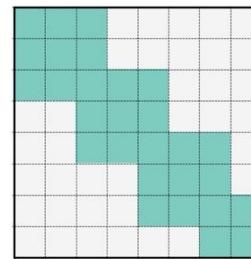
Longformer

Global



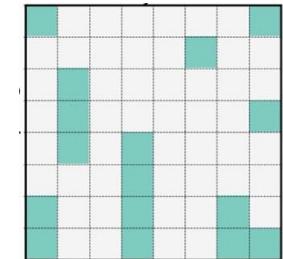
Big Bird

Blocked



Big Bird
Sinkhorn

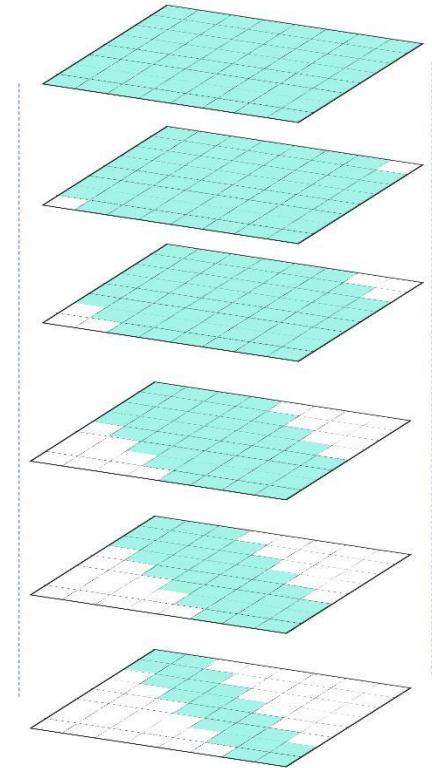
Random



Big Bird

Pre-specified Sparsity Patterns

- Different layers and attention heads can follow different patterns
- A common setup is to have earlier layers with sparser attention pattern.
 - Longformer ([Beltagy et al., 2020](#))



A Notable Adoption: GPT-3

- Sparse patterns also used in GPT-3 ([Brown et al., 2020](#))

2.1 Model and Architectures

We use the same model and architecture as GPT-2 [RWC⁺19], including the modified initialization, pre-normalization, and reversible tokenization described therein, with the exception that we use alternating dense and locally banded sparse attention patterns in the layers of the transformer, similar to the Sparse Transformer [CGRS19]. To study the dependence of ML performance on model size, we train 8 different sizes of model, ranging over three orders of magnitude from 125 million parameters to 175 billion parameters, with the last being the model we call GPT-3. Previous work [KMH⁺20]

Summary

- How well do Transformers work on long sequences? Not so well.
- How can we make them more efficient? Induce sparsity.
- We will see papers on other aspects of efficiency:
 - Quantization
 - Effective use of compute (GPUs)
 - Architectural variations
 - ...

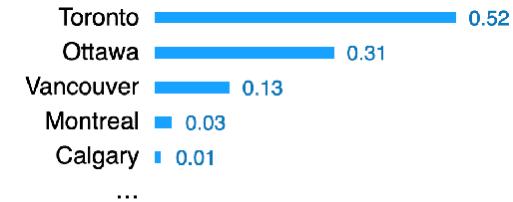
Retrieval-augmented LMs

Retrieval-based Language Models

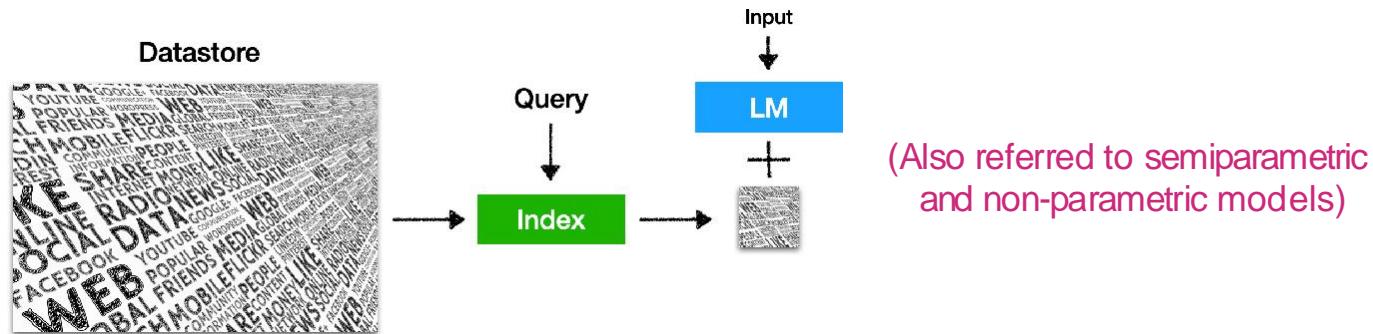
- It is a **language model** $P(x_n | x_1, x_2, \dots, x_{n-1})$

The capital city of Ontario is __

(can be broadly extended to masked language models or encoder-decoder models)



- It retrieves from an **external datastore** (at least during inference time)



Why retrieval?

- LLMs can't memorize all (long-tail) knowledge in their parameters



List 5 important papers authored by Geoffrey Hinton



Geoffrey Hinton is a renowned computer scientist ...
Here are five important papers authored by him:

1. "**Learning Internal Representations by Error Propagation**" (with D. E. Rumelhart and R. J. Williams) - This paper, published in **1986**, ..



2. "**Deep Boltzmann Machines**" (with R. Salakhutdinov) - Published in **2009**, ..



4. "**Deep Learning**" (with Y. Bengio and A. Courville) - Published as a book in **2016**, ...



Why retrieval?

- LLMs can't memorize all (long-tail) knowledge in their parameters
- LLMs' knowledge is easily outdated and hard to update



Who is the CEO of Twitter?



ChatGPT

As of my **knowledge cutoff in September 2021**, the CEO of Twitter is **Jack Dorsey**....

- The datastore can be easily **updated** and **expanded** - even without retraining!



Who is the CEO of Twitter? X Microphone Smartphone Search

All News Images Shopping Videos More Tools

About 1,090,000,000 results (0.45 seconds)

Twitter / CEO

Linda Yaccarino

Jun 5, 2023~

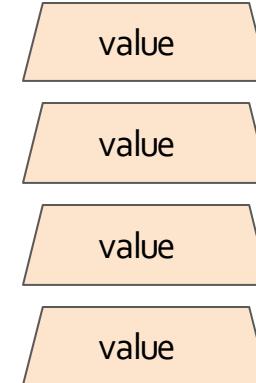


Anatomy of a Neural Retriever

Given an input, we want to
find the relevant docs
from the datastore

input

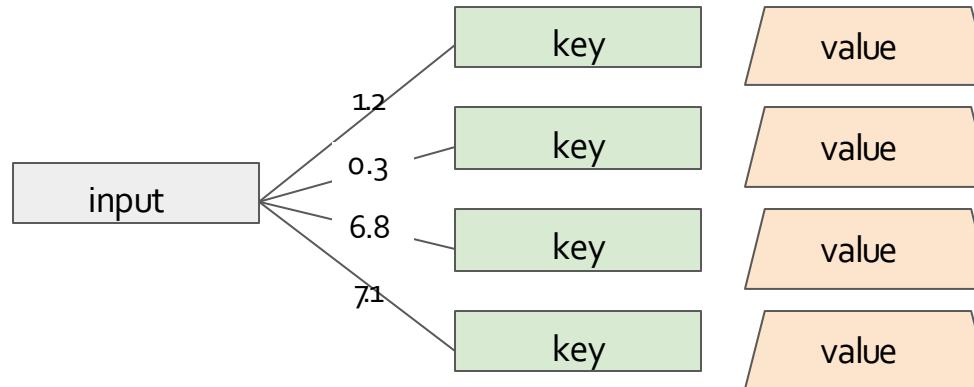
We have a “datastore” that
contains a variety
of documents



WIKIPEDIA
The Free Encyclopedia

Anatomy of a Neural Retriever

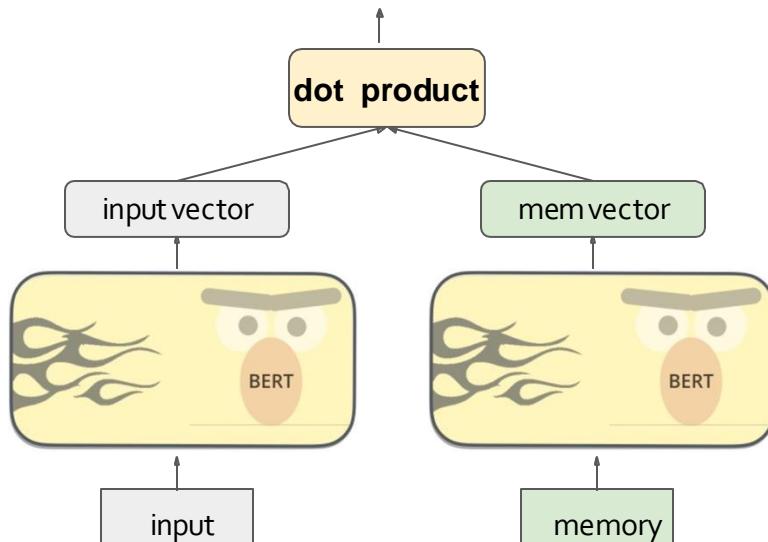
1. Score the input against each key.
2. Return the value for the highest scoring key.



A similarity function: $\text{sim}(\text{input}, \text{key}) \rightarrow \text{score}$

Similarity via Sentence Embeddings

$$\text{sim}(I, M) = \text{Encoder}(I) \times \text{Encoder}(M)$$



- **Advantages:**

- Differentiable -- can optimize with gradient descent.

- **Disadvantages:**

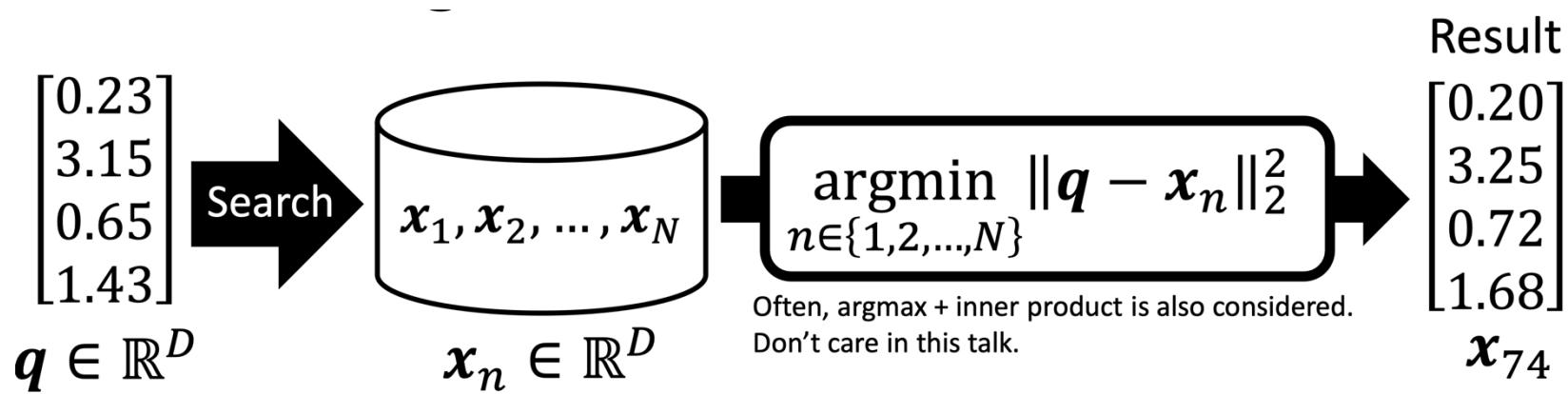
- Works well for data on which your LM is pre-trained on.



Thus far: Data store documents can be represented as word embeddings.
Now how do we find the most relevant ones?

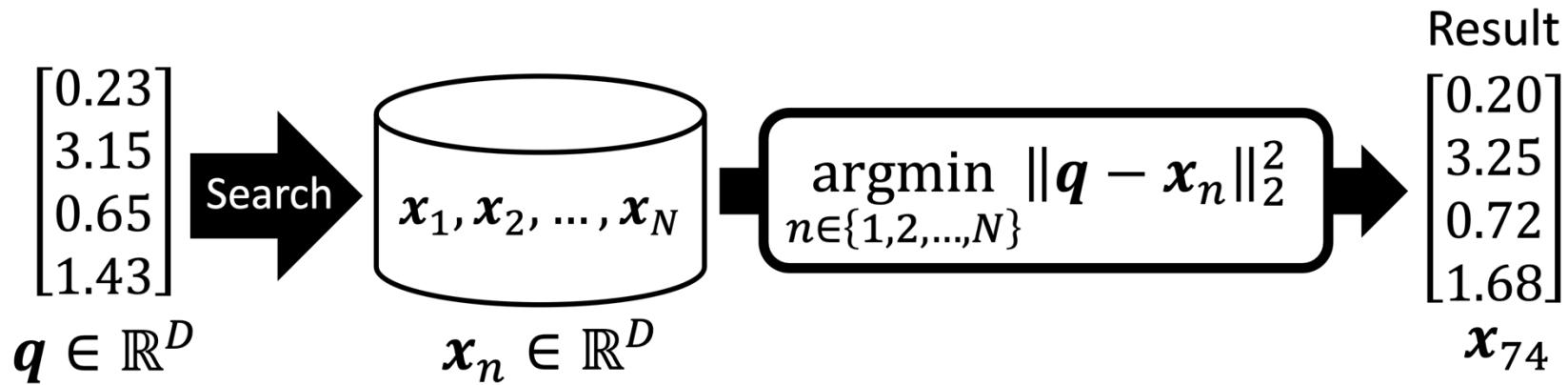


Finding Nearest Neighbors



- N D -dim database vectors: $\{x_n\}_{n=1}^N$
- Given a query q , find the closest vector from the database
- One of the fundamental problems in computer science
- Solution: linear scan, $O(ND)$, slow ☹

Approximate Finding Nearest Neighbors



- Faster search
- Don't necessarily have to be exact neighbors
- Trade off: runtime, accuracy, and memory-consumption

Approximate NNs: Algorithms, Libraries, Services

Algorithm

- Scientific paper
- Math
- Often, by researchers

Product Quantization +
Inverted Index (PQ, IVFPQ)
[Jégou+, TPAMI 2011]

Hierarchical Navigable
Small World (HNSW)
[Malkov+, TPAMI 2019]

ScaNN (4-bit PQ)
[Guo+, ICML 2020]

Library

- Implementations of algorithms
- Usually, a search function only
- By researchers, developers, etc

faiss

NMSLIB

hnswlib

ScaNN

Pinecone

Qdrant

Milvus

jina

Vald

Vertex AI
Matching Engine

Weaviate

Three levels of technology

Product Quantization +
Inverted Index (PQ, IVFPQ)
[Jégou+, TPAMI 2011]

Hierarchical Navigable
Small World (HNSW)
[Malkov+, TPAMI 2019]

faiss

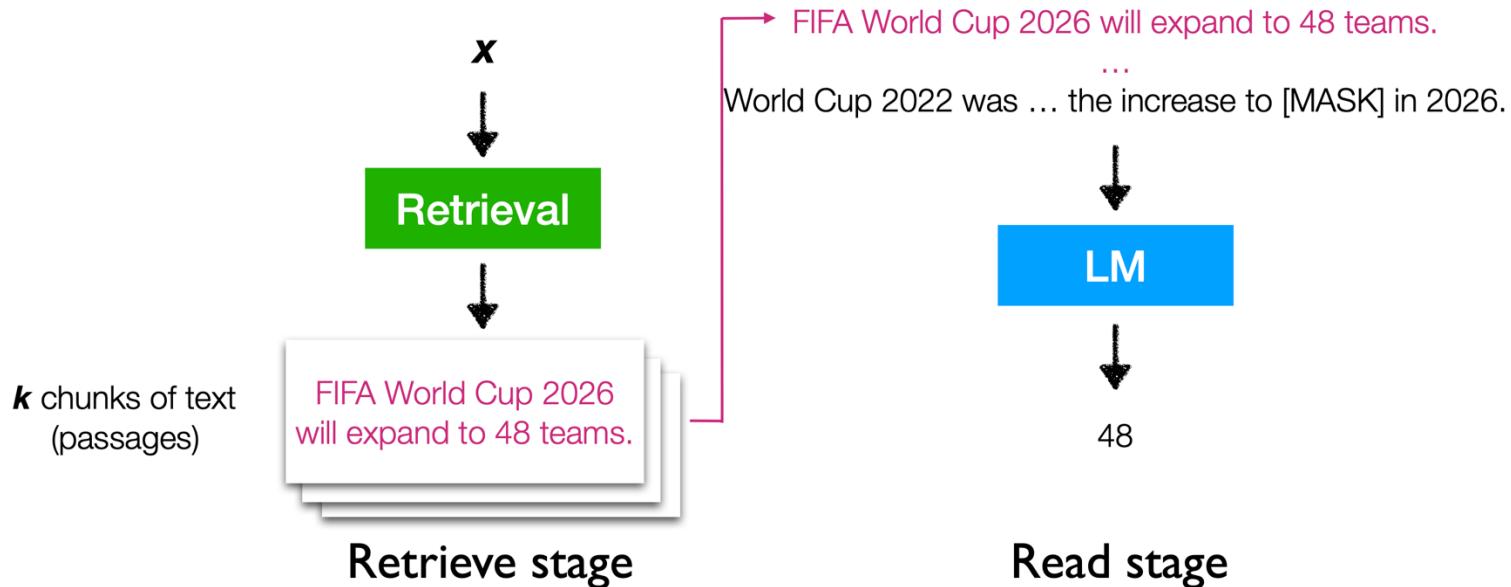
One library may implement
multiple algorithms

- “I benchmarked faiss”
- “I benchmarked PQ in faiss”

Let's assume that we have our retrieval engine and data ready

Retrieval-Augmented LM

- x = World Cup 2022 was the last before the increase to [MASK] in the 2026 tournament.



Summary

- How do we enable LMs to utilize external knowledge?
 - Retrieval-augmented language models
- A retriever is a function, $f(\text{input}, \text{memory}) \rightarrow \text{score}$
- What we did not discuss:
 - Attribution: Tracing decisions to the source knowledge
 - How to modify the knowledge
 - Conflicting knowledge
 - Editing knowledge
 - More efficient scaling
 -