Building LLM Reasoners

Lecture 1: Introduction, Transformer LMs

Greg Durrett

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Administrative details

Administrivia

- Lecture: Fridays 12:30pm-2:30pm CIWW 101
- Course website (including **syllabus**): http://gregdurrett.github.io/courses/sp2026/index.html
- Discourse: see Brightspace
- Office hours: see Brightspace. Greg's are hybrid, some TA OHs are hybrid too. **Office hours start next Monday**
- TAs: Shaswat Patel, Harry Wang

Course Requirements

- CSCI-GA.2590 Natural Language Processing or equivalent. You should be familiar with the Transformer architecture (fairly comfortable with the material in The Illustrated Transformer).
- CSCI-GA.2565 Machine Learning or equivalent
- Deep learning or equivalent
- Python programming experience, including PyTorch
- Experience with concepts from probability and linear algebra

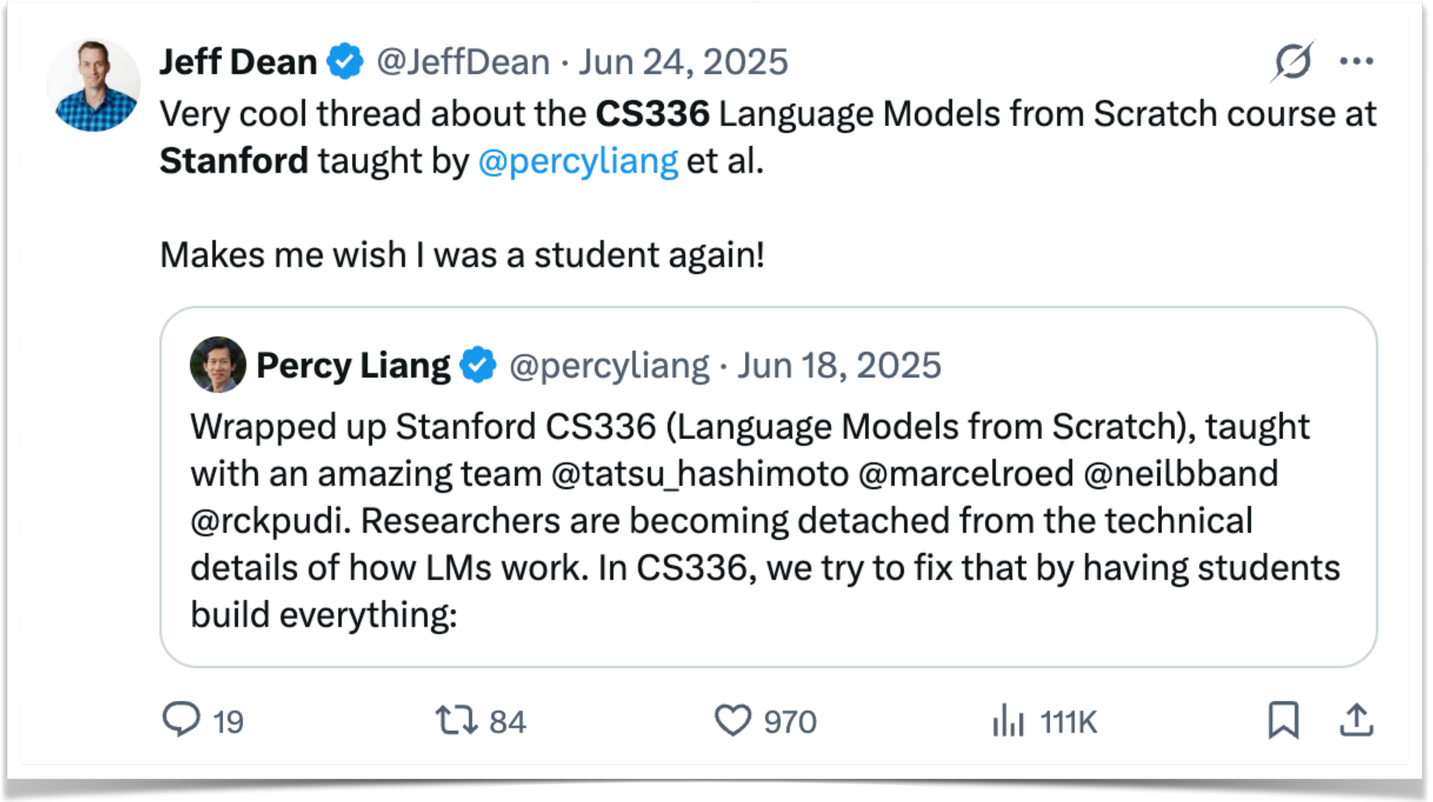
DATE	TOPIC	DETAILS	ASSIGNMENT
Jan 23	Intro, Transformers Review	Course overview; expectations; review of the Transformer architecture.	Assignment 1 released
Jan 30	Tokenizers, Optimizers, and Tricks	Tokenizers, positional encodings, optimizers, and other stuff needed to make Transformer LLMs work.	—
Feb 6	Making LLMs Fast I	GPUs, memory layouts, and FlashAttention.	—
Feb 13	Making LLMs Fast II	Inference-time optimization, KV caching	Assignment 1 due · Assignment 2 released · In-class quiz 1
Feb 20	Scaling Laws	Empirical scaling laws with connections to convergence rates of estimators, model/data/compute tradeoffs.	—
Feb 27	Training I: SFT & RLHF	Supervised fine-tuning, reinforcement learning from human feedback, and how they're used to train LLMs	—
Mar 6	Training II: GRPO & RLVR	Training methods for modern reasoners	Assignment 2 due · Assignment 3 released · In-class quiz 2
Mar 13	Midterm	In-class midterm exam	—

Mar 20	No Class	Spring Break.	—
Mar 27	TBD	Topic to be announced.	Final project proposals due
Apr 3	LLM Evaluation	Principles for evaluating LLMs, including statistical testing and benchmark design practices	Assignment 3 due · In-class quiz 3
Apr 10	Multimodality	LLMs beyond text: vision, audio, and other modalities.	—
Apr 17	Agents	Tool calling, MCP, computer use agents, and agentic workflows.	—
Apr 24	TBD	Topic to be announced.	—
May 1	Project Presentations	Final project presentations and wrap-up.	Final project due date TBD around this time

Assignments and Compute

- Assignment 1 is released now, due in 3 weeks. **Get started early.** It took Greg about 10-12 hours.

- These assignments are adapted from CS336 at Stanford (taught by Percy Liang and Tatsu Hashimoto). **Thanks to them for developing the assignments and letting us use them!**



Coursework

- Three assignments, worth 25% of grade
  - Mix of writing and implementation
  - ~3 weeks per assignment
  - 5 “slip days” throughout the semester to turn in assignments 24 hours late
  - Submission on Gradescope
- Three homework quizzes, worth 25% of the grade
  - If you did the homework and didn’t just get Claude to do it, you should be fine.

These assignments require understanding the concepts, writing performant code, and thinking about how to debug complex systems. **They are challenging; start early!**

Office hours: please come! However, **the course staff are not here to debug your code!** We **will** help you understand the concepts and come up with debugging strategies!

Coursework

- Midterm (25% of grade)
  - You can see the style of exams I give via my online MS course at UT Austin (linked from my webpage)

- Final project (25% of grade), groups of 2 or 3, proposal due in advance

Academic Honesty

Students are encouraged to discuss lecture material, homework problems, and coding assignments with others! However, **your final written solution or source code must be your own**, excluding the final project, which may be completed in groups.

You may not:

- Copy all or part of an assignment from a fellow student, the Internet, or AI
- Bring answers into an exam, or consult an AI during an exam
- Submit any work without appropriate attribution. In the final project, you must make clear when you are using code, data, figures, concepts, or any other artifacts from prior work.

You may consult external resources such as blog posts, YouTube videos, academic papers, GitHub repositories, AI assistants, and more. However, your use of such resources, particularly GitHub repositories, must be limited in the same way as discussions with other students: you can look at these to get an idea of how to solve a problem, but you should not take external code and submit it as part of your assignment, **except for the final project when it is appropriately attributed.**

AI Assistants

Policy on AI Assistants

Understanding the capabilities of these systems and their boundaries is a major focus of this class, and there's no better way to do that than by using them!

- **We strongly encourage you to use AI assistants to understand concepts in AI and machine learning.** You should see it as another tool like web search that can supplement understanding of the course material.
- **You are allowed to use Claude Code/ChatGPT/etc. for programming assignments, but your usage must be limited in the same way as usage of other resources.** You should come up with the high-level skeleton of the solution and an initial implementation yourself and use these tools primarily as coding assistants for debugging. If you and another student submit exactly the same code because you both generated it with Claude Code, we will treat that as if you had copied each others' solutions.
- **You are permitted to use AI assistants for conceptual questions on assignments, but discouraged from doing so.** These questions are meant to deepen your understanding of the course content, particularly in preparation for the midterm. Heavily relying on ChatGPT for your answers will negatively impact your learning.

An example of a good question is, *"Write a line of Python code to reshape a Pytorch tensor x of [batch size, seqlen, hidden dimension] to be a 2-dimensional tensor with the first two dimensions collapsed."* An example of a bad question would be to try to feed in a large chunk of the assignment code and copy-paste the problem specification from the assignment PDF. As a heuristic, it should be possible for you to explain what each line of your code is doing. If you have code in your solution that is only included because an assistant told you to put it there, then it is no longer your own work.

Survey

- See “quiz” on Brightspace to be released at end of class (please remind me if I forget!)

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Understanding Language

Dialog Systems

Machine Translation

The Political Bureau of the CPC Central Committee

July 30

hold a meeting

中共中央政治局7月30日召开会议，会议分析研究当前经济形势，部署下半年经济工作。

People's Daily, August 10, 2020

Translate

The Political Bureau of the CPC Central Committee held a meeting on July 30 to analyze and study the current economic situation and plan economic work in the second half of the year.

Question Answering

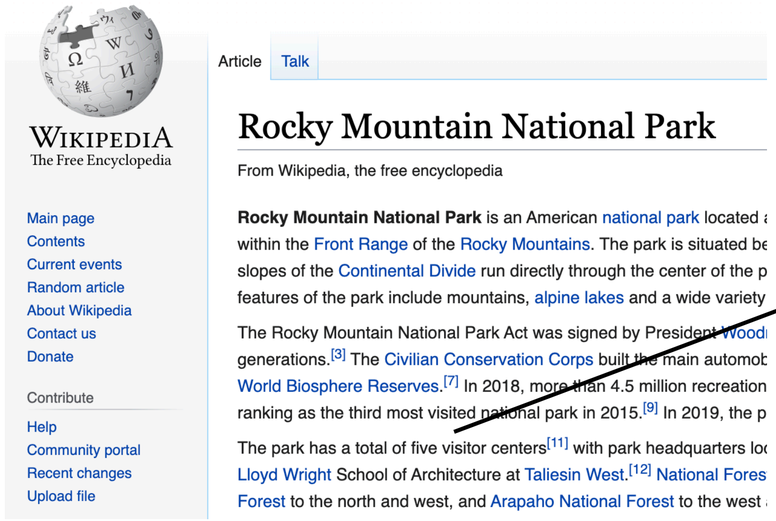
When was Abraham Lincoln born?

map to `Birthday` field

Name	`Birthday`
Lincoln, Abraham	2/12/1809
Washington, George	2/22/1732
Adams, John	10/30/1735

→ **February 12, 1809**

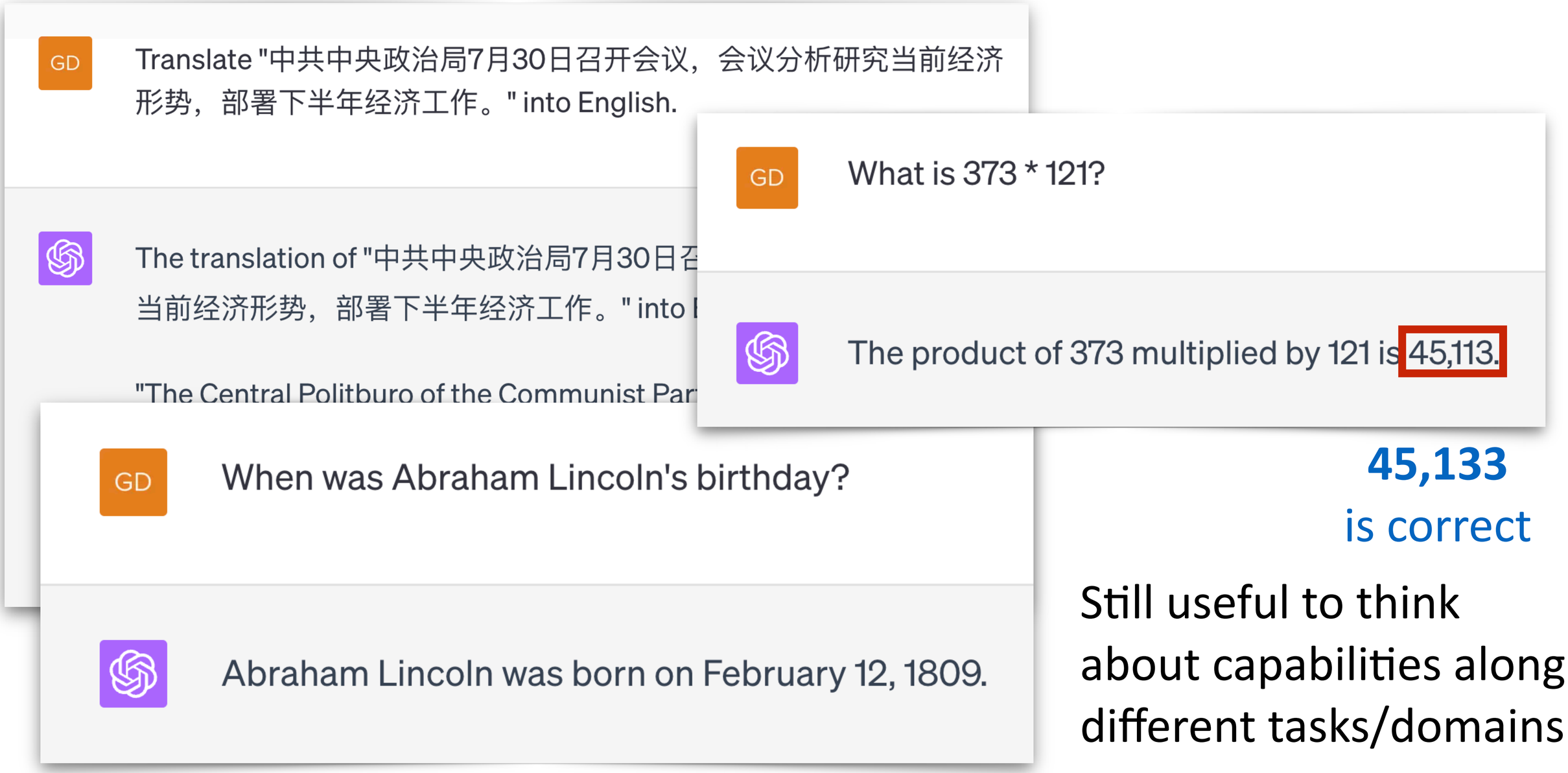
How many visitors centers are there in Rocky Mountain National Park?



The park has a total of five visitor centers

five

Generalist Systems?



45,133
is correct

Still useful to think about capabilities along different tasks/domains.

Language is Ambiguous!

- Hector Levesque (2011): “Winograd schema challenge” (named after Terry Winograd, the creator of SHRDLU)

The city council refused the demonstrators a permit because they advocated violence

The city council refused the demonstrators a permit because they feared violence

The city council refused the demonstrators a permit because they _____ violence

- Referential ambiguity

Language is Ambiguous!

Teacher Strikes Idle Kids

Ban on Nude Dancing on Governor's Desk

Iraqi Head Seeks Arms

- Syntactic and semantic ambiguities: parsing needed to resolve these, but need context to figure out which parse is correct

example credit: Dan Klein

Language is **Really** Ambiguous!

- There aren't just one or two possibilities which are resolved pragmatically

il fait vraiment beau ⟶

It is really nice out

It's really nice

The weather is beautiful

It is really beautiful outside

He makes truly beautiful

It fact actually handsome

- Combinatorially many possibilities, many you won't even register as ambiguities, but systems still have to resolve them
- In machine learning, we learn to do things with labeled data…or in this case, web-scale "unlabeled" data

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Why Language Modeling?

Pretraining

Language modeling: predict the next word in a text $P(w_i|w_1, \ldots, w_{i-1})$

P(*w* | I want to go to) = 0.01 Hawai'i

0.005 LA

0.0001 class

P(*w* | the acting was horrible, I think the movie was) = 0.1 bad

0.001 good

- Train a neural network to do language modeling on massive unlabeled text, can use it to do other tasks or further fine-tune it (alignment)

The Power of Language Modeling

My name _____

- One good option (*is*)?

My name is _____

- Flat distribution over many alternatives. But hard to get a good distribution?

I visited San _____

- Requires some knowledge but not one right answer

The capital of Texas is _____

- Requires more knowledge (one answer…or is there?)

The casting and direction were top notch. Overall I thought the movie was ___

- Requires implicitly doing sentiment analysis!

Language Models are Unsupervised Multitask Learners

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

”I’m not the cleverest man in the world, but like they say in French: **Je ne suis pas un imbecile [I’m not a fool].**

In a now-deleted post from Aug. 16, Soheil Eid, Tory candidate in the riding of Joliette, wrote in French: ”**Mentez mentez, il en restera toujours quelque chose**,” which translates as, ”**Lie lie and something will always remain.**”

- Is pre-training enough? What happens if we sample from a pre-trained model?

I think it should be fixed on either UTC standard or UTC+1 year around, with the current zone offsets. Moving timescales add a lot of complexity to the implementation of timekeeping systems and have dubious value

I think seasonal shifting time made sense in the pre-electric past, when timekeeping was more flexible and artificial light was inefficient and often dangerous.

[…]

tl;dr: Shifting seasonal time is no longer worth it.

reddit.com

What was the "shot heard round the world"?

The shot that was heard round the world was fired in Lexington, MA. It was fired because, at the time, America was still under British rule. The British soldiers had been taking firewood by throwing it down from the top of their fort at Lexington. Someone who was not a soldier, by the name of Joseph Warren (who had ties to the Sons of Liberty or the Sons of Freedom) shouted to the British to stop doing it. The British soldiers ignored him, not

What do you think of this completion?

What was the "shot heard round the world"?

"What to the slave is the 4th of July?"

What was the sideline occupation of Thomas Jefferson's wife, Martha, and their enslaved girl, Sally Hemmings?

What was the name of the first African American regiment from Vermont?

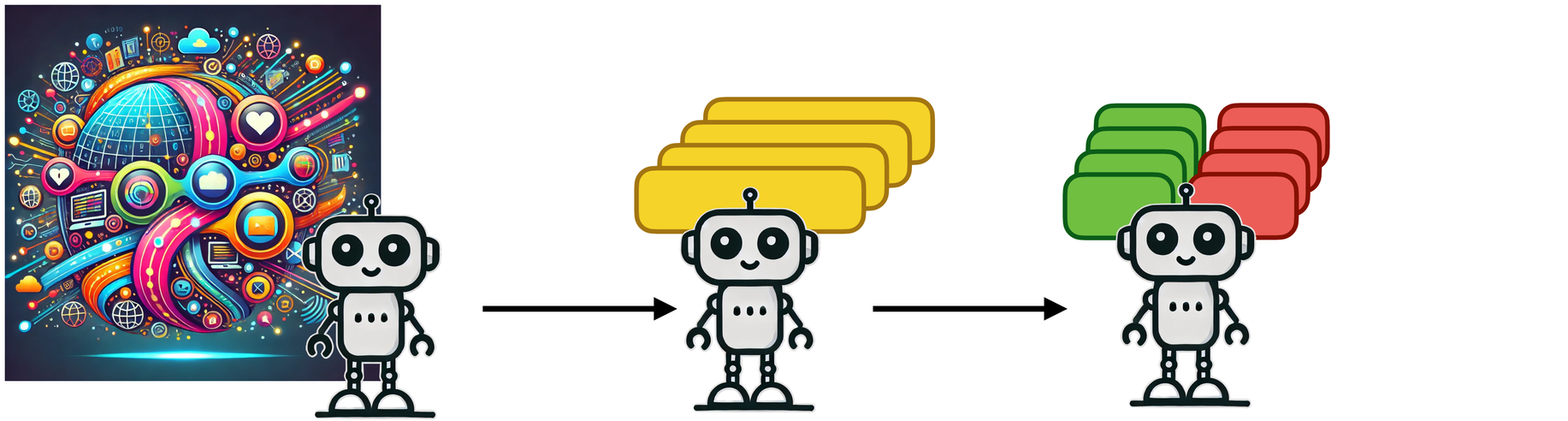
What do you think of this one?

Samples from GPT-3

Other shortcomings of pre-training

Alignment

- Is pre-training enough? No!



Pretraining

Supervised fine-tuning (SFT)

Reinforcement learning (e.g., RLHF, RLVR)

Alignment

- Is pre-training enough? No!
- Increasingly complex training flows

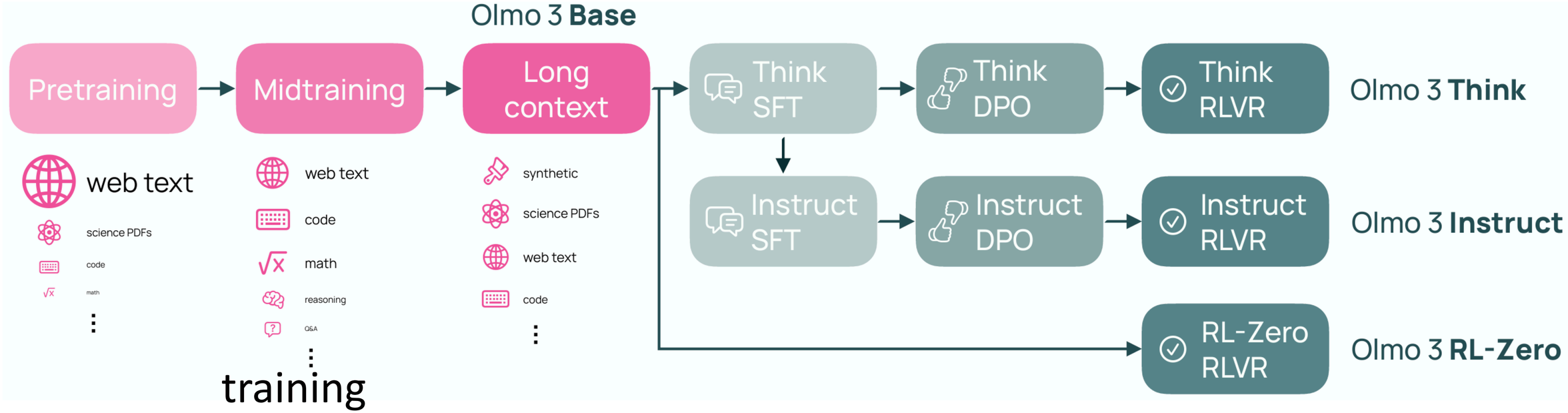
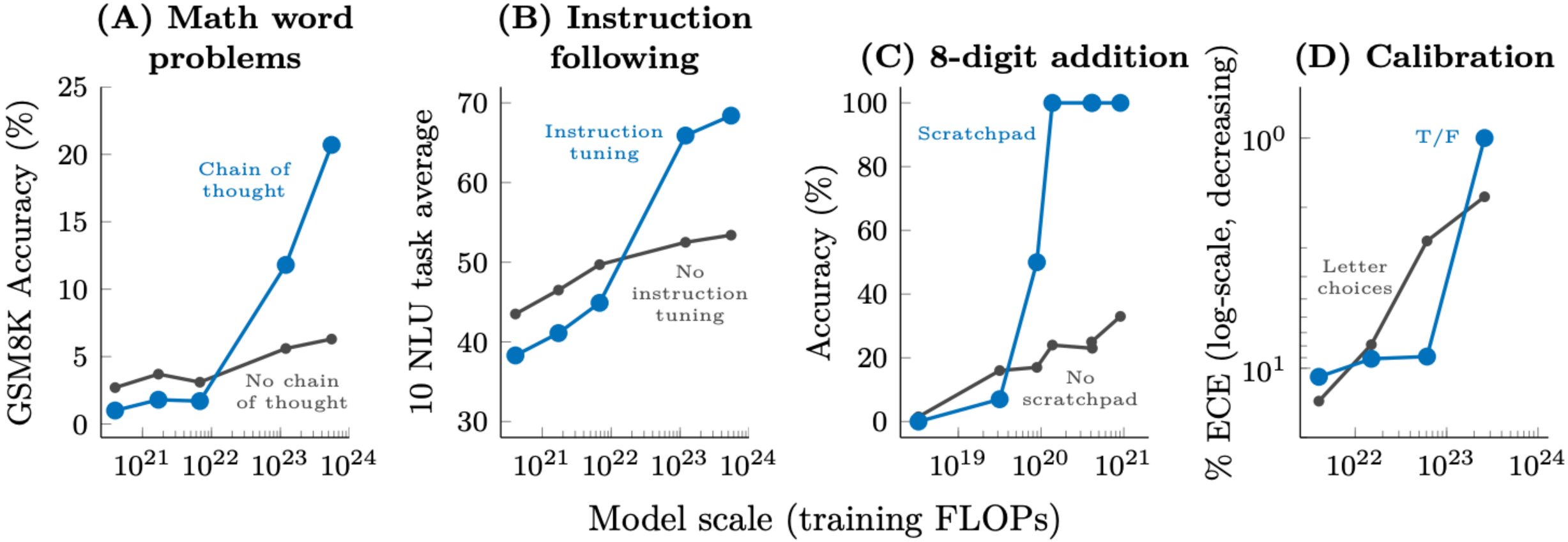


Figure 2 Depiction of model flow for Olmo 3. Development is divided into major **base model training (left)** and **post-training (right)** stages, each further divided into sub-stages with their own recipes (i.e., training data and method).

OLMo 3 (2025)

Scaling Laws



- Many of the methods that work in LLMs today only make sense and only work because the models are so big!

Kaplan et al. (2020), Jason Wei et al. (2022)

Efficiency

- Modern LLM development is compute-constrained: we would train LLMs for longer on the same data (or more data we could get cheaply) if we could
- Transformers and their implementation on GPUs reflects many design decisions arising from these constraints and the need to train on as many tokens as possible as quickly as possible
- First: we'll understand the Transformer. Then we'll dive into how to make it fast.

Goals of this course

- Understand how a language model works all the way down. **Build it from scratch to really learn this.**

- According to Percy Liang: three important things to know:
  - Mechanics: how things work (Transformer arch., model parallelism on GPUs, etc.)
  - Mindset: squeezing the most out of the hardware, taking scale seriously
  - Intuitions: which data and modeling decisions yield good accuracy

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Language Modeling

Language Modeling

- Fundamental task in both linguistics and NLP: can we determine of a sentence is *acceptable* or not?
- Related problem: can we evaluate if a sentence is grammatical? Plausible? Likely to be uttered?
- Language models: place a distribution P(***w***) over strings ***w*** in a language. This is related to all of these tasks but doesn't exactly map onto them
- Autoregressive models $P(\mathbf{w}) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\ldots$

N-gram Language Models

$$P(\mathbf{w}) = P(w_1)P(w_2|w_1)P(w_3|w_1, w_2)\dots$$

- n-gram models: distribution of next word is a categorical conditioned on previous n-1 words $P(w_i|w_1, \dots, w_{i-1}) = P(w_i|w_{i-n+1}, \dots, w_{i-1})$

- Markov property: only consider a few previous words

I visited San _____ put a distribution over the next word

2-gram: P(w | San)

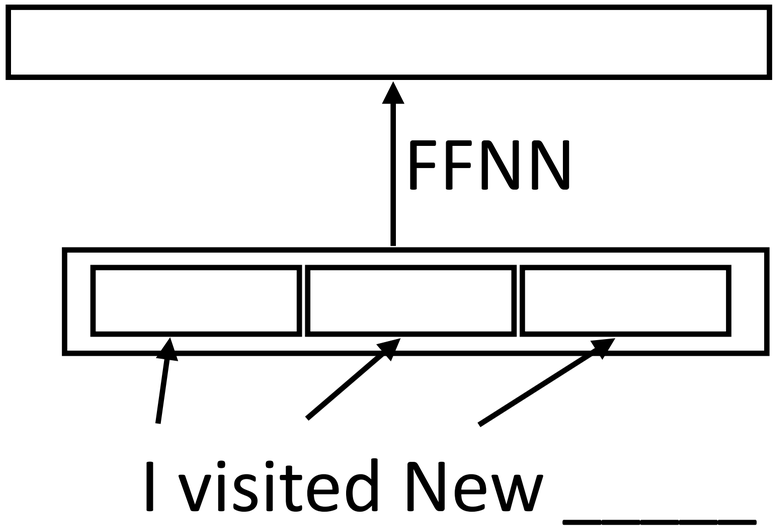
3-gram: P(w | visited San)

4-gram: P(w | I visited San)

- n-gram LLMs don't generalize or abstract over related words, and don't handle contexts beyond a few words

Challenges of Neural Language Modeling

Use a feedforward network?
(Bengio et al, 2003)



- Language models based purely on feedforward networks can abstract over words (using embeddings), but still fail to use large context
- Need to handle more context: RNNs or CNNs can do this, but current best is Transformers using **self-attention**

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Transformers

Running Example

- Fixed-length sequence of As and Bs

AAAAAAA

ABAAAAB

ABAABAB

AAAABAB

BAAAAAAAAAAAAAAAAAAAAAAAAB

- All As = last letter is A; any B = last letter is B
- **Attention**: method to access arbitrarily* far back in context from this point
- RNNs generally struggle with this; remembering context for many positions is hard (though of course they can do this simplified example — you can even hand-write weights to do it!)

Keys and Query

- Keys: embedded versions of the sentence; query: what we want to find

Assume A = [1, 0]; B = [0, 1] (one-hot encodings of the tokens); call these e_i

Step 1: Compute scores for each key

keys k_i

[1, 0] [1, 0] [0, 1] [1, 0]

A A B A

$s_i = k_i^T q$

0 0 1 0

query: q = [0, 1] (we want to find Bs)

Attention

Step 1: Compute scores for each key

keys k_i

[1, 0] [1, 0] [0, 1] [1, 0]

query: q = [0, 1] (we want to find Bs)

A A B A

$s_i = k_i^T q$

0 0 1 0

Step 2: softmax the scores to get probabilities α

0 0 1 0 => (1/6, 1/6, 1/2, 1/6) if we assume e=3

Step 3: compute output values by multiplying embs. by alpha + summing

result = sum($\alpha_i e_i$) = 1/6 [1, 0] + 1/6 [1, 0] + 1/2 [0, 1] + 1/6 [1, 0] = [1/2, 1/2]

Attention

keys k_i

[1, 0] [1, 0] [0, 1] [1, 0]

A A B A

query: q = [0, 1] (we want to find Bs)

(1/6, 1/6, 1/2, 1/6) if we assume e=3

result = sum($\alpha_i e_i$) = 1/6 [1, 0] + 1/6 [1, 0] + 1/2 [0, 1] + 1/6 [1, 0] = [1/2, 1/2]

How does this differ from just averaging the vectors?

What if we have a very very long sequence?

New Keys

keys k_i

[1, 0] [1, 0] [0, 1] [1, 0]

A A B A

query: q = [0, 1] (we want to find Bs)

We can make attention more peaked by not setting keys equal to embeddings.

$k_i = W^K e_i$ $\quad W^K = \begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$

[10, 0][10, 0][0, 10][10, 0]

0 0 1 0

What will new attention values be with these keys?

Attention, Formally

- Original “dot product” attention: $s_i = k_i^T q$
- Scaled dot product attention: $s_i = k_i^T W q$
- Equivalent to having two weight matrices: $s_i = (W^K k_i)^T (W^Q q)$
- Other forms exist: Luong et al. (2015), Bahdanau et al. (2014) present some variants (originally for machine translation)

Self-Attention

Self-attention: every word is both a key and a query simultaneously

Q: seq len x d matrix (d = embedding dimension = 2 for these slides)

K: seq len x d matrix

$W^Q = \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix}$ no matter what the value is, we're going to look for Bs

$W^K = \begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$ "booster" as before

Note: many ways to set up these weights that will be equivalent to this!

Self-Attention

$$E = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad W^Q = \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \qquad W^K = \begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$$

$$Q = E\,(W^Q) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad K = E\,(W^K) = \begin{pmatrix} 10 & 0 \\ 10 & 0 \\ 0 & 10 \\ 10 & 0 \end{pmatrix}$$

Scores $S = QK^T$ $\qquad S_{ij} = q_i \cdot k_j$

len x len = (len x d) x (d x len)

Let's compute these now!

Self-Attention

$$E = \begin{pmatrix} 1 & 0 \\ 1 & 0 \\ 0 & 1 \\ 1 & 0 \end{pmatrix} \qquad W^Q = \begin{matrix} 0 & 1 \\ 0 & 1 \end{matrix} \qquad W^K = \begin{matrix} 10 & 0 \\ 0 & 10 \end{matrix}$$

$$Q = E\,(W^Q) = \begin{pmatrix} 0 & 1 \\ 0 & 1 \\ 0 & 1 \\ 0 & 1 \end{pmatrix} \qquad K = E\,(W^K) = \begin{pmatrix} 10 & 0 \\ 10 & 0 \\ 0 & 10 \\ 10 & 0 \end{pmatrix}$$

Scores $S = QK^T$ $\quad S_{ij} = q_i \cdot k_j$

len x len = (len x d) x (d x len)

Final step: softmax to get attentions A, then output is AE

*technically it's A (EW^V), using a values matrix $V = EW^V$

Self-Attention (Vaswani et al.)

$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

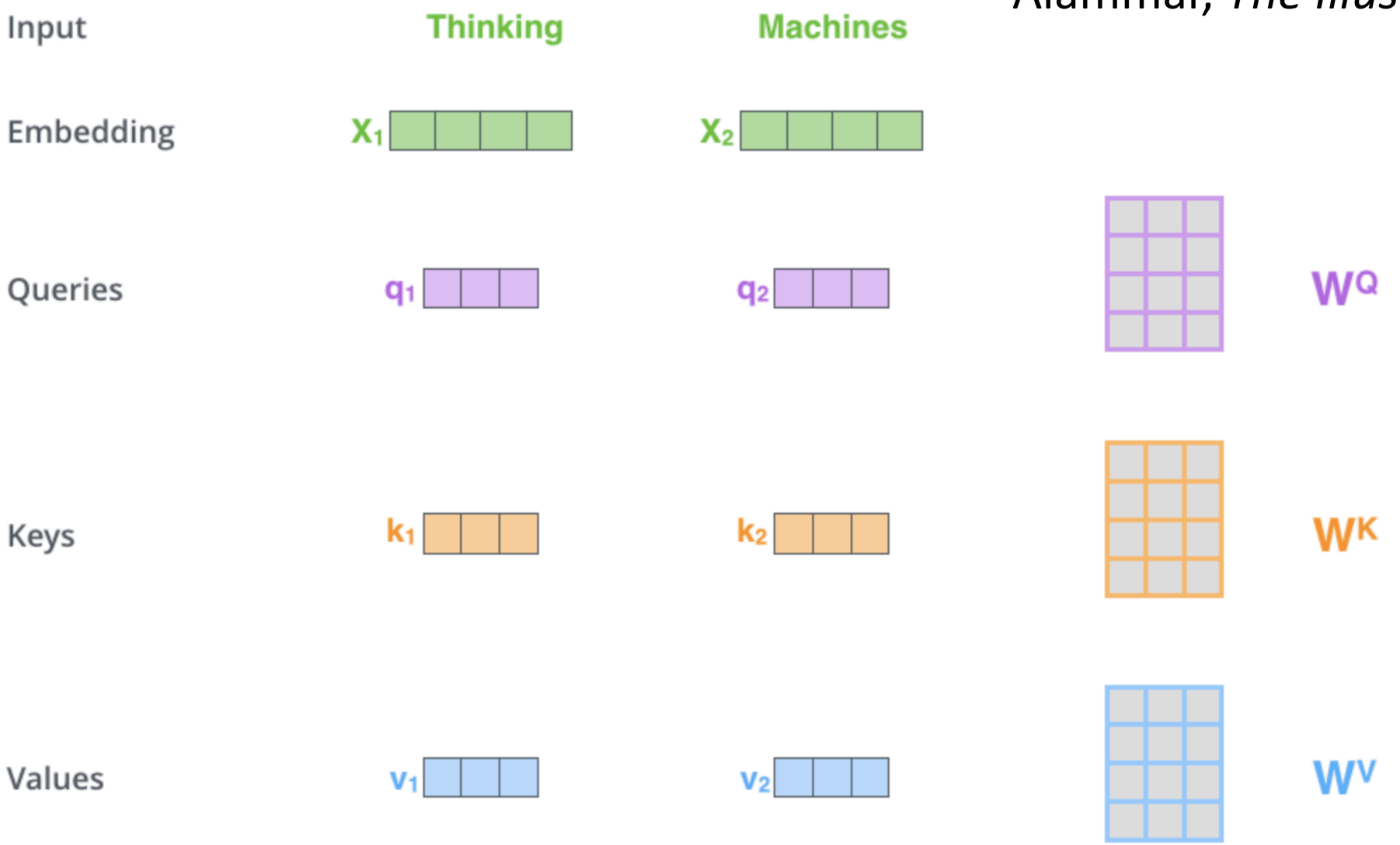
$Q = EW^Q$, $K = EW^K$, $V = EW^V$

- Normalizing by $\sqrt{d_k}$ helps control the scale of the softmax, makes it less peaked
- This is just one head of self-attention — produce multiple heads via randomly initialize parameter matrices (more in a bit)

Vaswani et al. (2017)

Self-Attention

Alammar, *The Illustrated Transformer*



Self-Attention

Alammar, *The Illustrated Transformer*

sent len x sent len (attn for each word to each other)

$$\text{softmax}\left(\frac{Q \times K^T}{\sqrt{d_k}}\right) V = Z$$

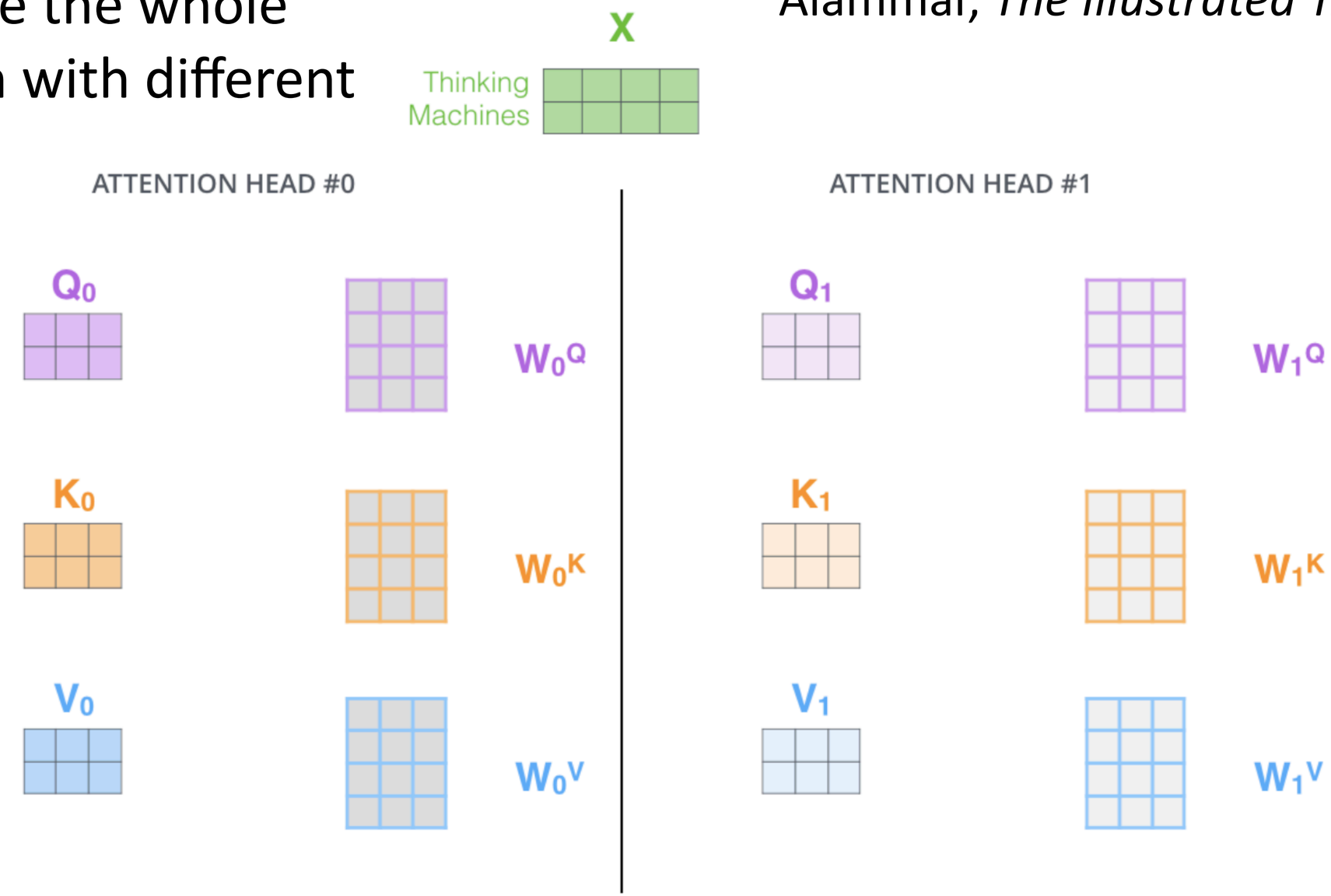
sent len x hidden dim

Z is a weighted combination of V rows

Multi-head Self-Attention

Just duplicate the whole computation with different weights:

Alammar, *The Illustrated Transformer*



Multi-head Self-Attention

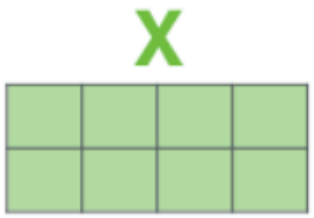
1) This is our input sentence*

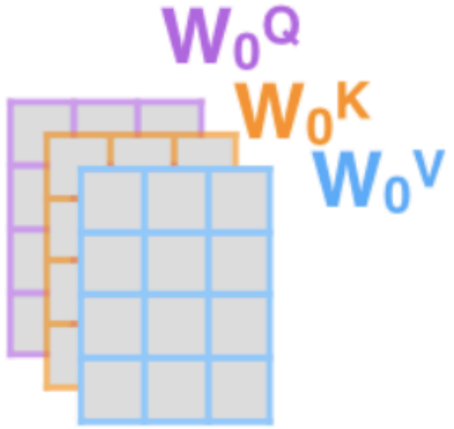
2) We embed each word*

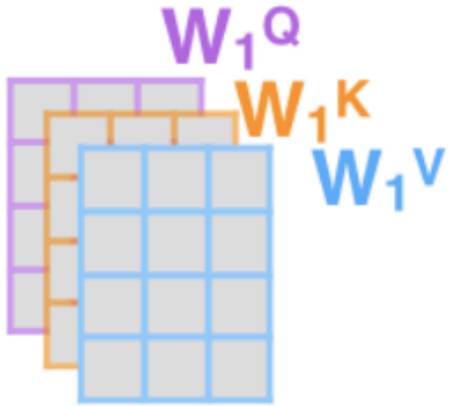
3) Split into 8 heads. We multiply X or R with weight matrices

Alammar, *The Illustrated Transformer*

Thinking Machines







* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

Multi-head Self-Attention

1) This is our input sentence*

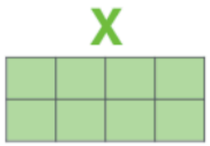
2) We embed each word*

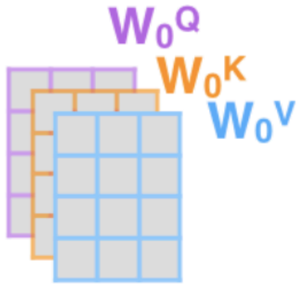
3) Split into 8 heads. We multiply X or R with weight matrices

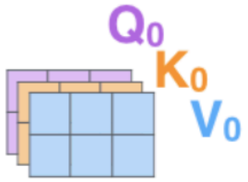
4) Calculate attention using the resulting Q/K/V matrices

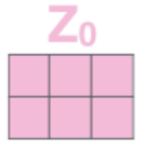
5) Concatenate the resulting Z matrices, then multiply with weight matrix W^O to produce the output of the layer

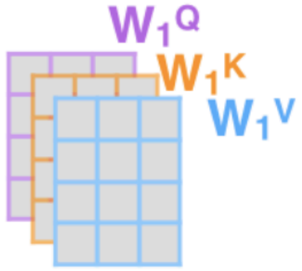
Thinking Machines

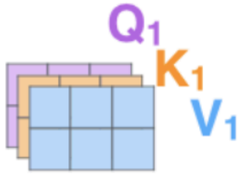


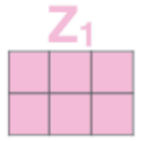


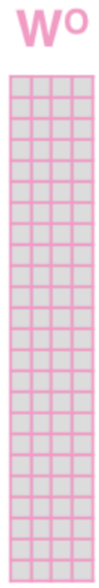


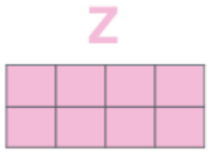






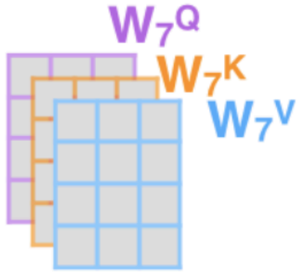




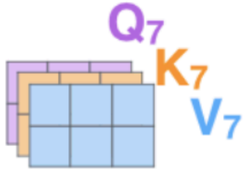


* In all encoders other than #0, we don't need embedding. We start directly with the output of the encoder right below this one

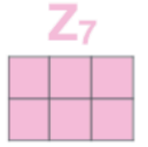
...



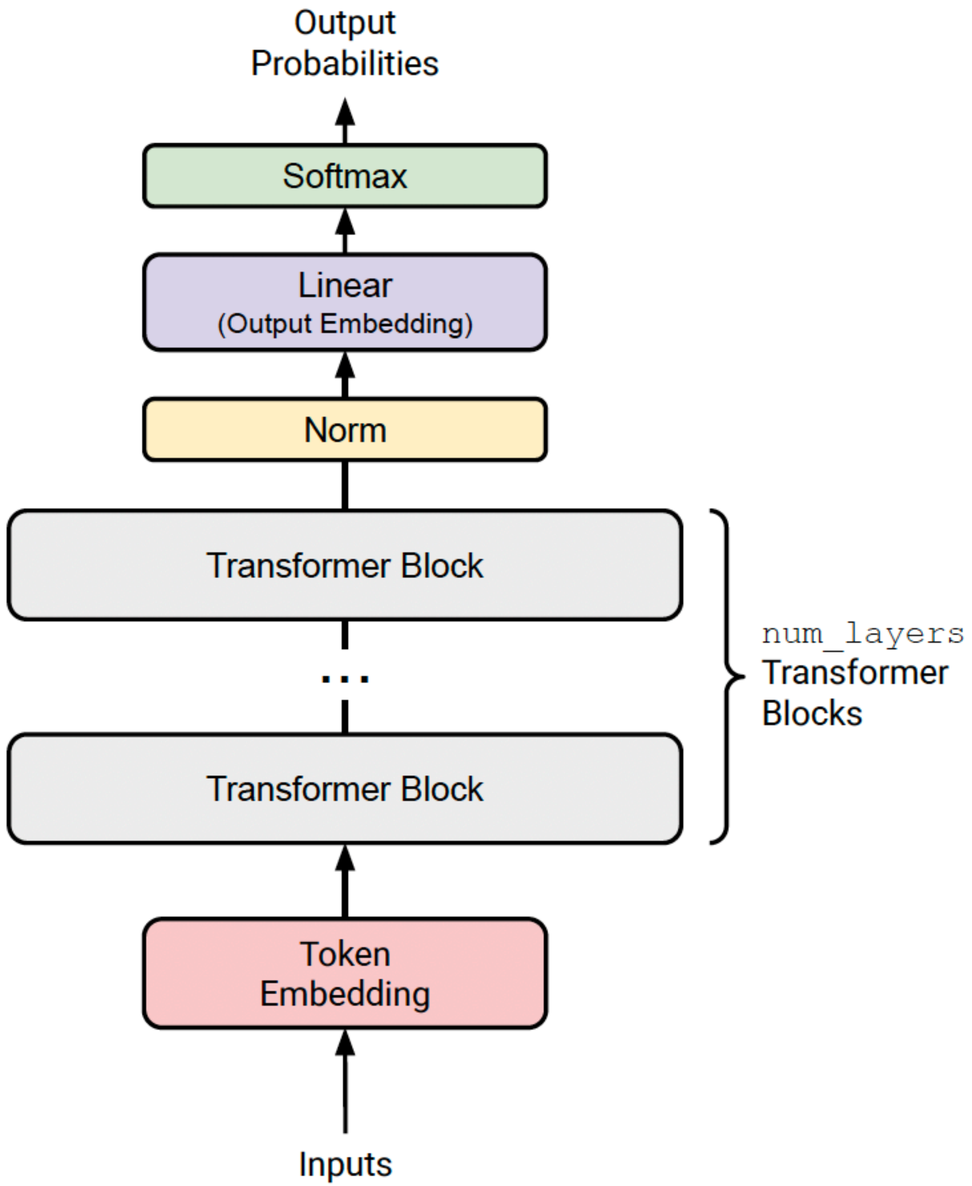
...



...



Architecture



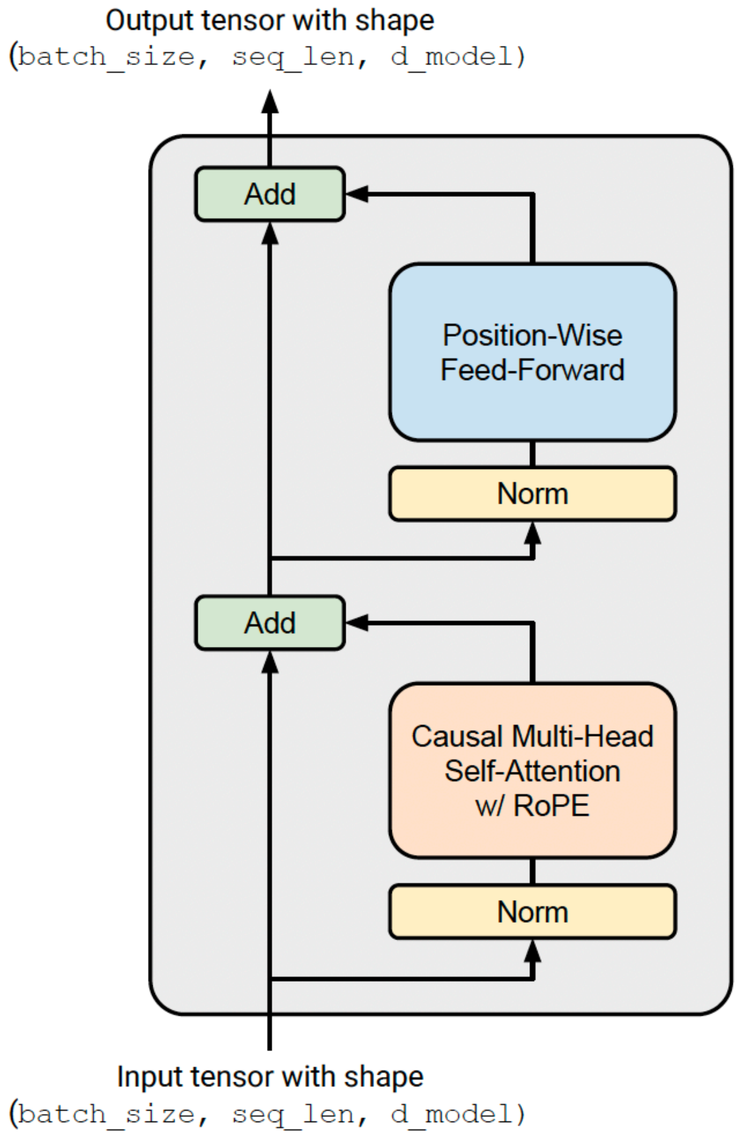
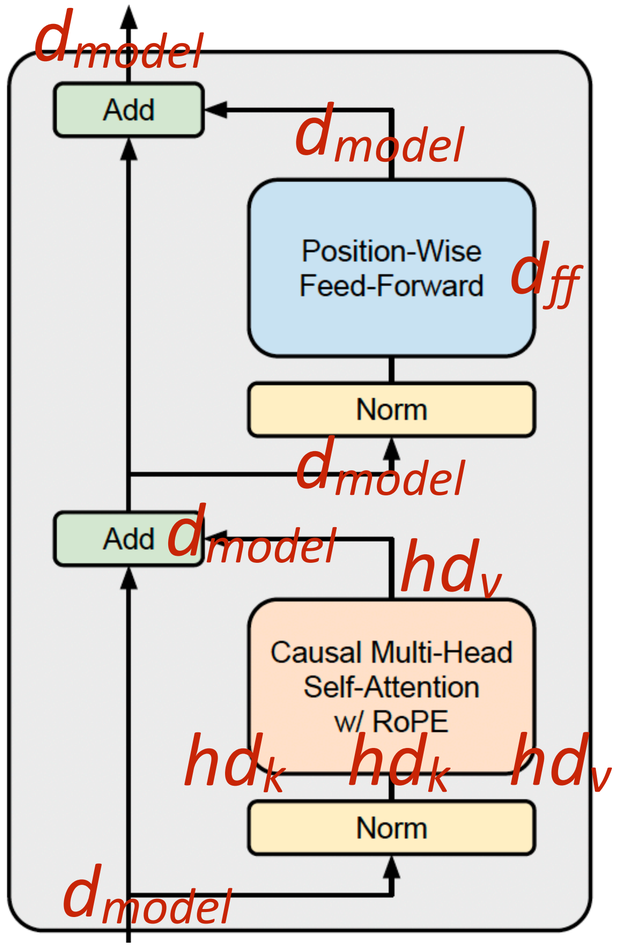


Figure credit:
Stanford CS336 A1

Dimensions

- Main vector size d_{model}
- Queries/keys: d_k , always smaller than d_{model}, often d_{model}/h (number of heads)
- Values: separate dimension d_v , output is multiplied by W^O which is $(d_v \, x \, h) \, x \, d_{model}$ so we can get back to d_{model}
- FFN can use a higher latent dimension



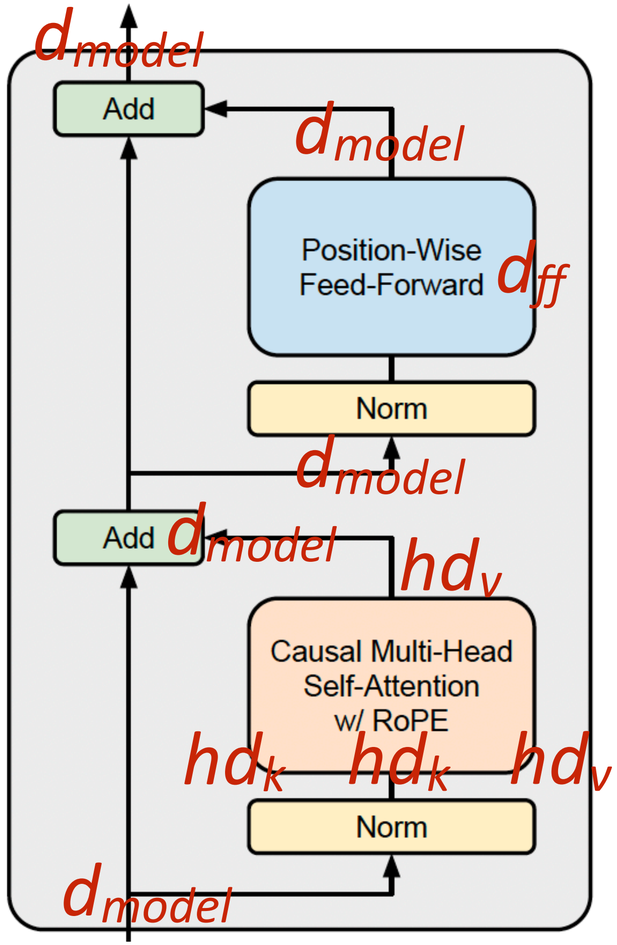
$$\mathrm{FFN}(x) = \mathrm{SwiGLU}(x, W_1, W_2, W_3) = W_2(\mathrm{SiLU}(W_1 x) \odot W_3 x)$$

Figure credit: Stanford CS336 A1

Transformer Architecture

Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}
GPT-3 Small	125M	12	768	12	64
GPT-3 Medium	350M	24	1024	16	64
GPT-3 Large	760M	24	1536	16	96
GPT-3 XL	1.3B	24	2048	24	128
GPT-3 2.7B	2.7B	32	2560	32	80
GPT-3 6.7B	6.7B	32	4096	32	128
GPT-3 13B	13.0B	40	5140	40	128
GPT-3 175B or "GPT-3"	175.0B	96	12288	96	128

From GPT-3; d_{head} is our d_k



FLOPs

1	description	FLOPs / update	% FLOPS MHA	% FLOPS FFN	% FLOPS attn	% FLOPS logit
8	OPT setups					
9	760M	4.3E+15	35%	44%	14.8%	5.8%
10	1.3B	1.3E+16	32%	51%	12.7%	5.0%
11	2.7B	2.5E+16	29%	56%	11.2%	3.3%
12	6.7B	1.1E+17	24%	65%	8.1%	2.4%
13	13B	4.1E+17	22%	69%	6.9%	1.6%
14	30B	9.0E+17	20%	74%	5.3%	1.0%
15	66B	9.5E+17	18%	77%	4.3%	0.6%
16	175B	2.4E+18	17%	80%	3.3%	0.3%

Credit: Stephen Roller on Twitter (Meta's OPT models)

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

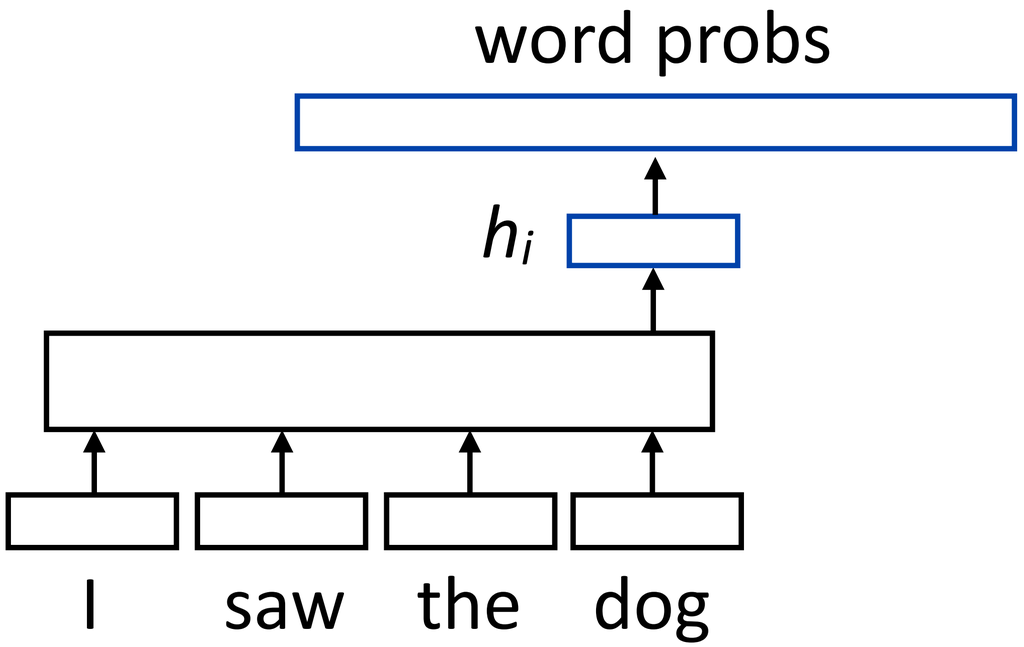
Transformer Language Modeling

Positional Encodings

Decoding

Transformer Language Modeling

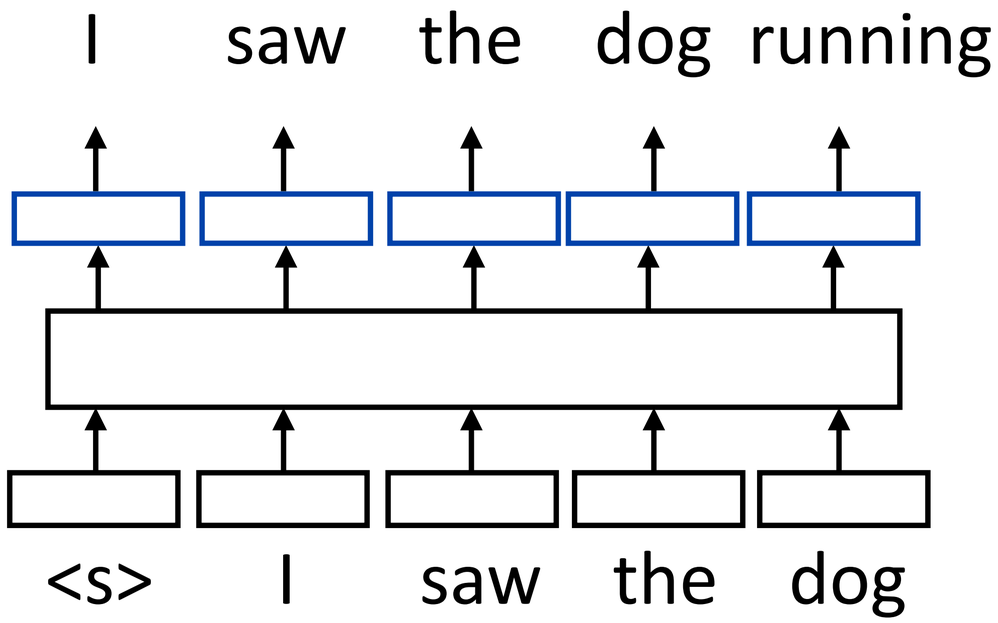
Transformer Language Modeling



$$P(w|\text{context}) = \text{softmax}(W\mathbf{h}_i)$$

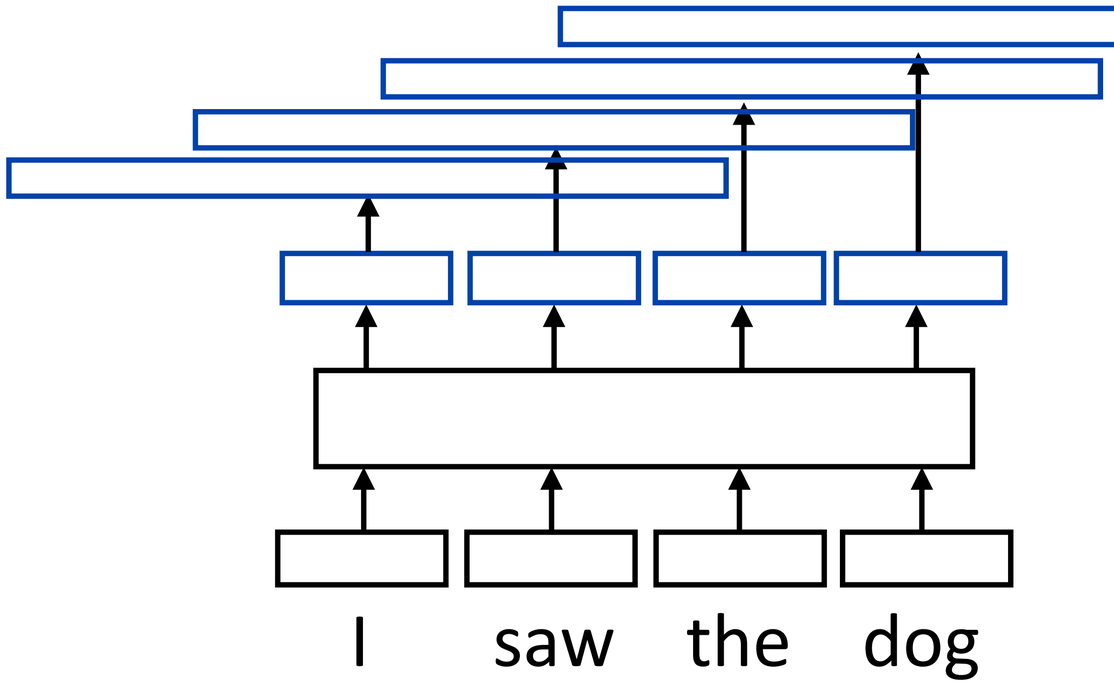
W is a (vocab size) x (hidden size) matrix

Training Transformer LMs



- Input is a sequence of words, output is those words shifted by one,
- Allows us to train on predictions across several timesteps simultaneously (similar to batching but this is NOT what we refer to as batching)

Training Transformer LMs



P(w|context)

loss = — log P(w*|context)

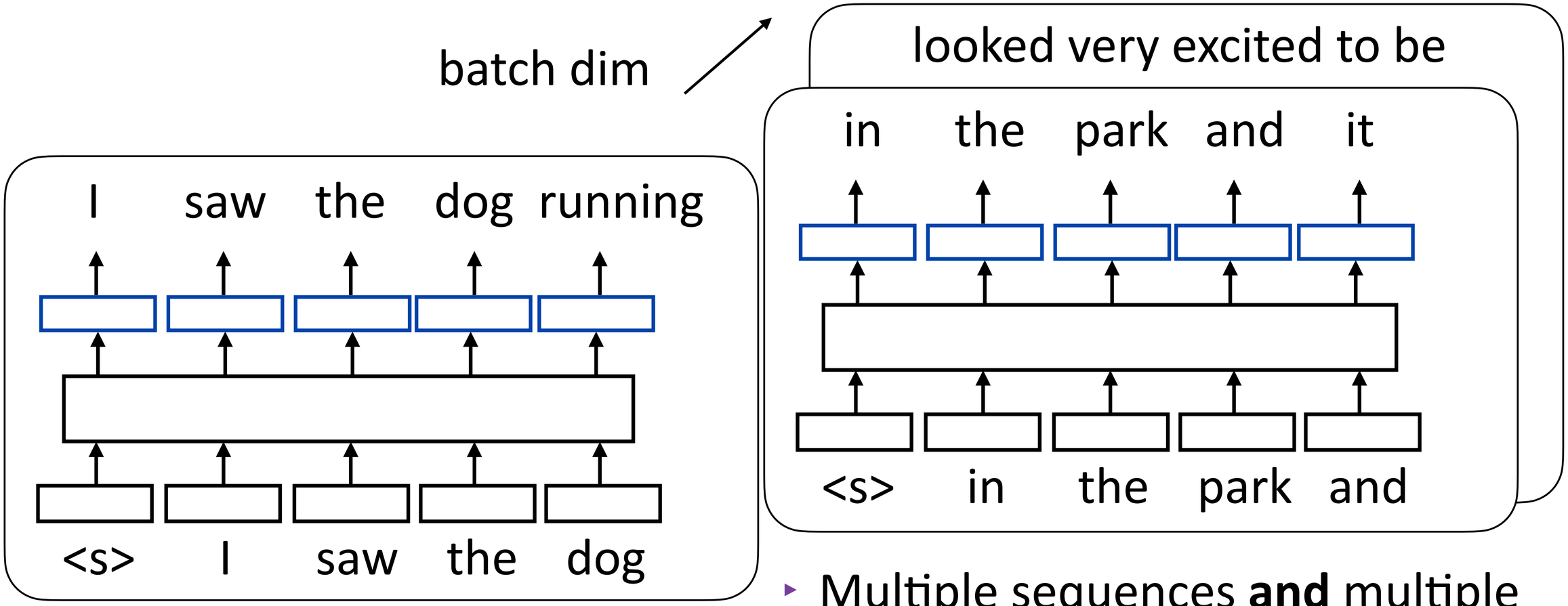
Total loss = sum of negative log likelihoods at each position

- Parallel inference across several tokens at training time, but at decoding time, tokens are generated one at a time

Batched LM Training

I saw the dog running | in the park and it | looked very excited to be | there

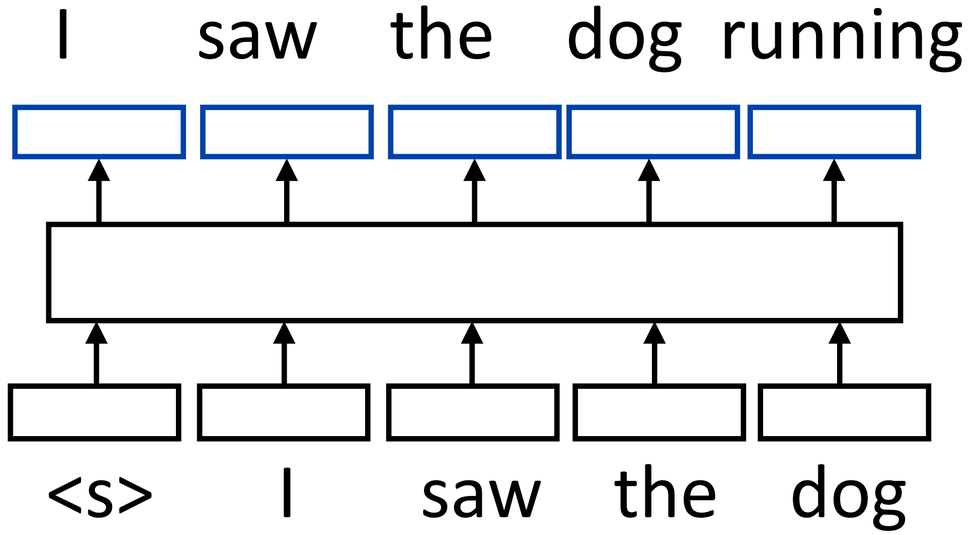
batch dim



- Multiple sequences **and** multiple timesteps per sequence

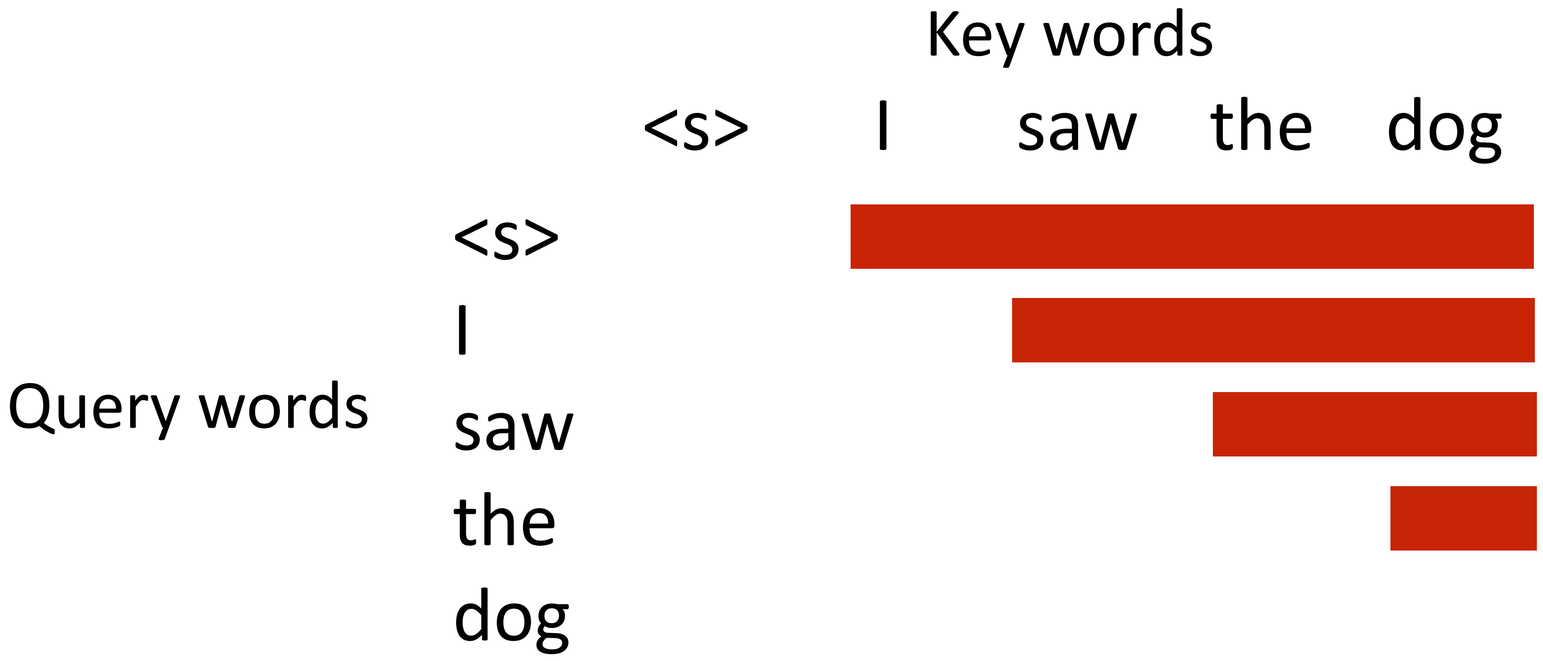
A Small Problem with Transformer LMs

- This Transformer LM as we've described it will *easily* achieve perfect accuracy. Why?



- With standard self-attention: "I" attends to "saw" and the model is "cheating". How do we ensure that this doesn't happen?

Attention Masking



- We want to mask out everything in red (an upper triangular matrix)

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Positional Encodings

Running Example

AAAAAAA

ABAAAAB

ABAABAB

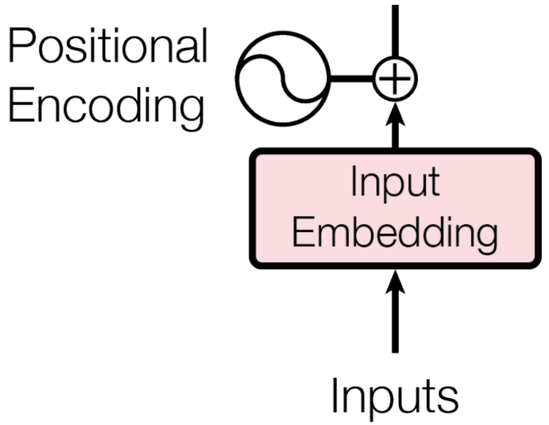
AAAABAB

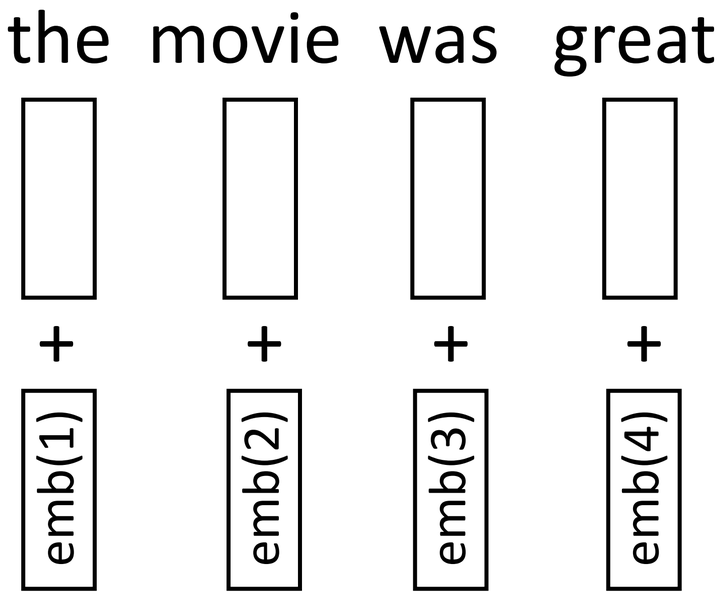
BAAAAAAAAAAAAAAAAAAAAAAAAAB

- All As = last letter is A; any B = last letter is B

- This didn't rely on self-attention being able to distinguish different positions.
- If we want to detect "B followed by 3 As", our approach so far may* not work well. (*we'll come back to this)

Absolute Position Encodings (BERT, etc.)



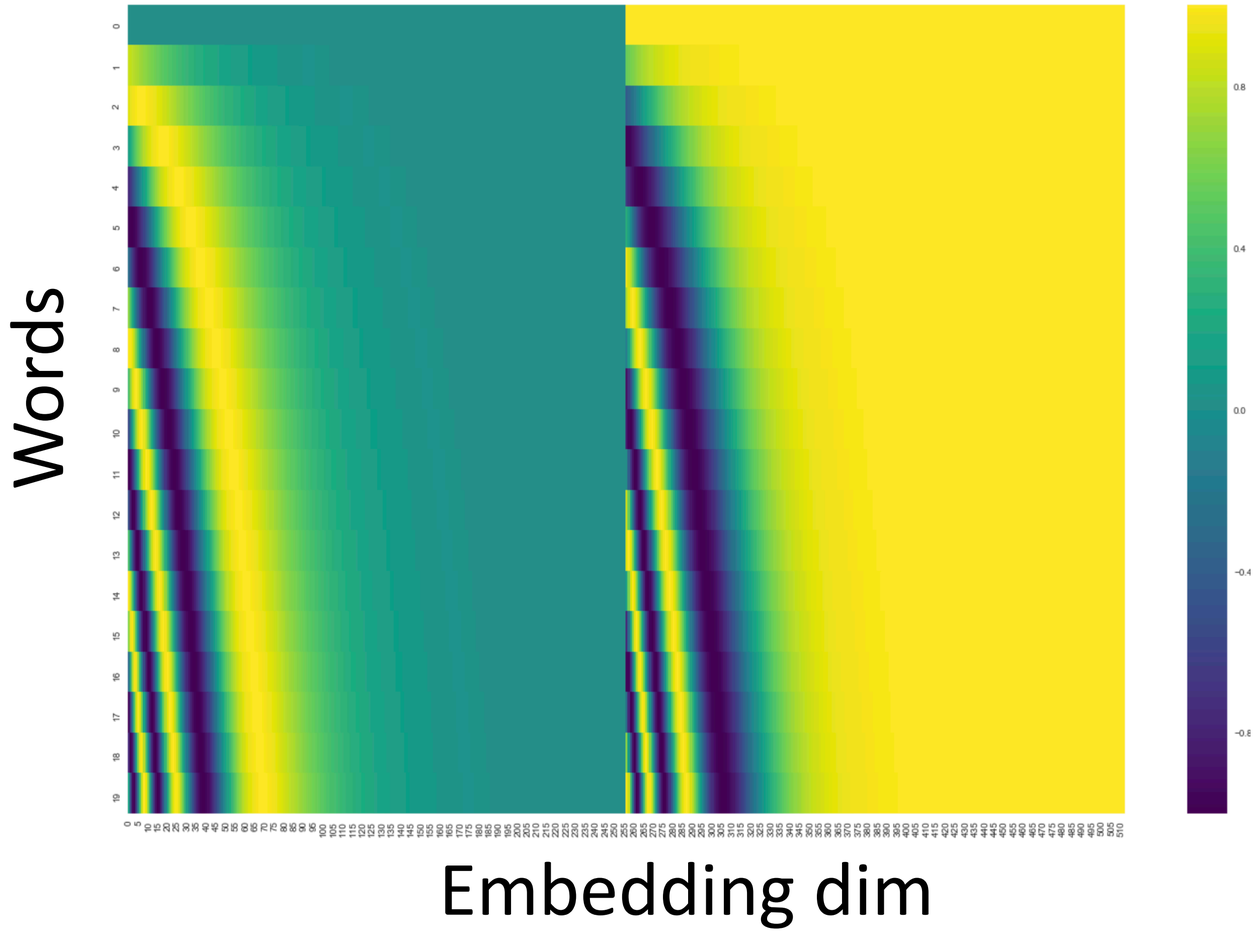


- Encode each sequence position as an integer, add it to the word embedding vector
- Why does this work?

Sinusoidal PE RoPE (Vaswani et al., 2017)

Alammar, *The Illustrated Transformer*

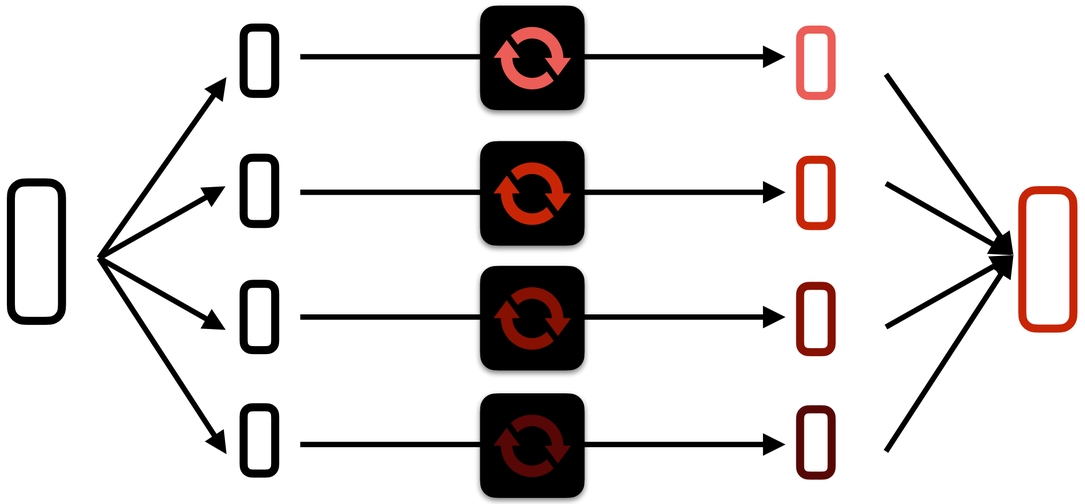
- Alternative from Vaswani et al.: sines/cosines of different frequencies (closer words get higher dot products by default)



RoPE (Jianlin Su et al., 2021)

Goal: encode positional information in each vector.

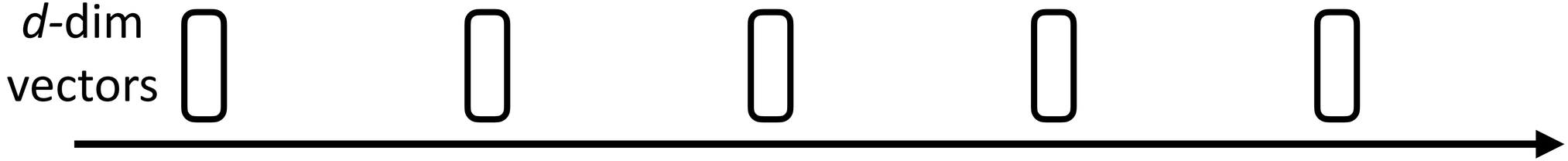
For vector at position *i*:



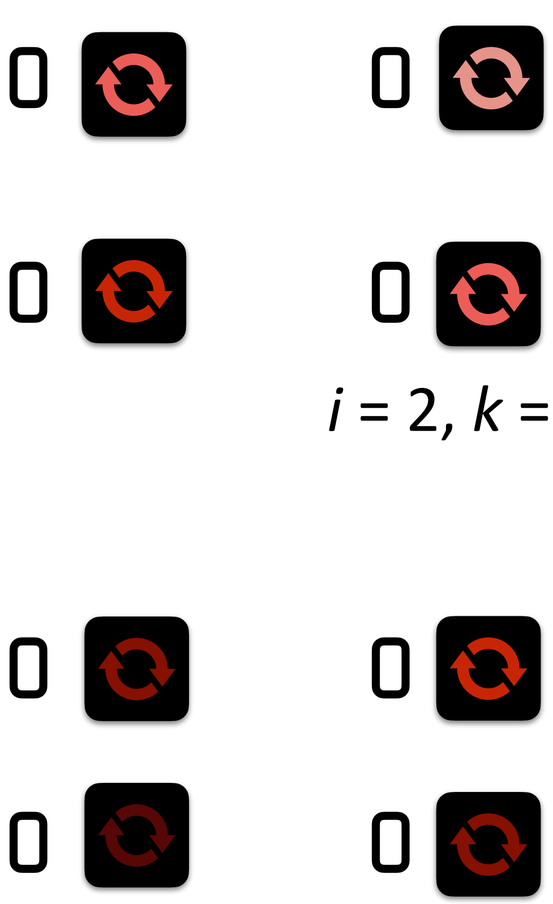
Step 1: Break into $d/2$ vectors in $\mathbb{R}^2$

Step 2: Rotate each one by an amount depending on *i* and the vector index

RoPE (Jianlin Su et al., 2021)



increasing token position *i* (going through sentence)



i = 2, *k* = 2

$$\theta_{i,k} = \frac{i}{\Theta^{(2k-2)/d}}$$

equation credit:
Stanford CS336 A1

$$R_k^i = \begin{bmatrix} \cos(\theta_{i,k}) & -\sin(\theta_{i,k}) \\ \sin(\theta_{i,k}) & \cos(\theta_{i,k}) \end{bmatrix}$$

Treat this element as a point in 2D space and rotate it by $\theta_{i,k}$

RoPE (Jianlin Su et al., 2021)

$$\theta_{i,k} = \frac{i}{\Theta^{(2k-2)/d}}$$

What happens as *i* increases?

What happens as *k* increases?

RoPE Angle Visualizer

$$\theta_{i,k} = i \,/\, \Theta^{(2k-2)/d}$$

i (position index)

3

k (dimension pair, 1 to d/2)

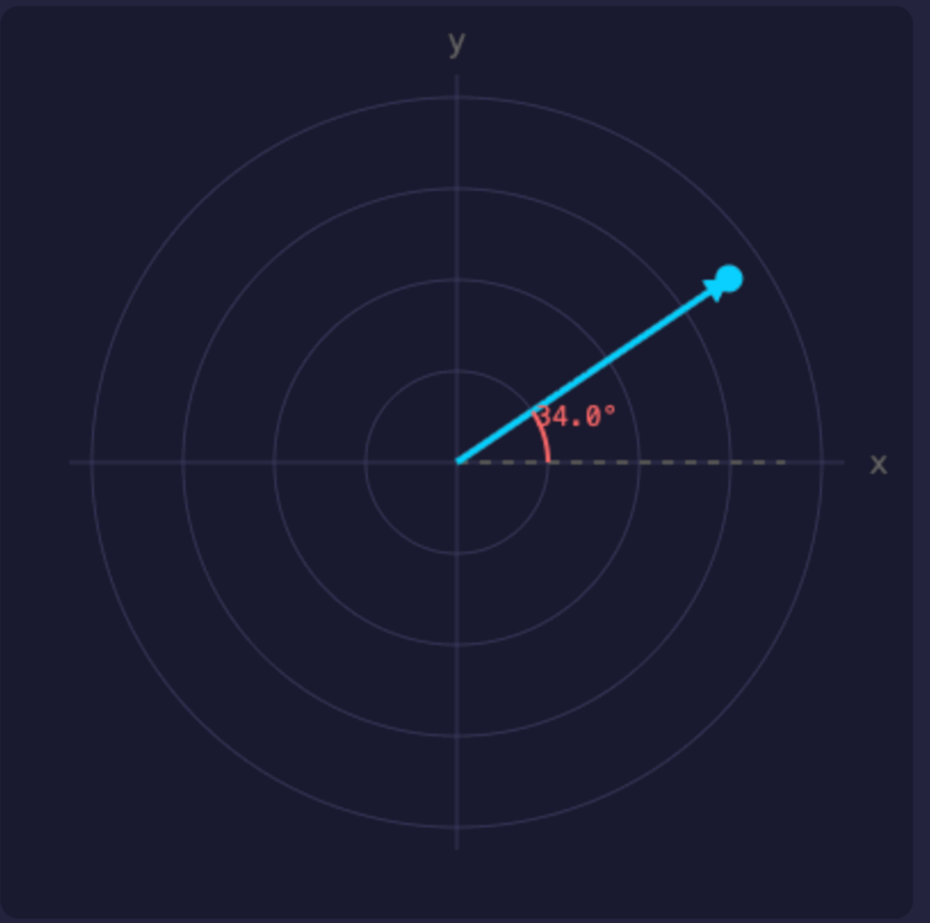
2

Θ (base theta)

3290

d (embedding dimension)

10



$\theta_{3,2} = 0.5939$ rad

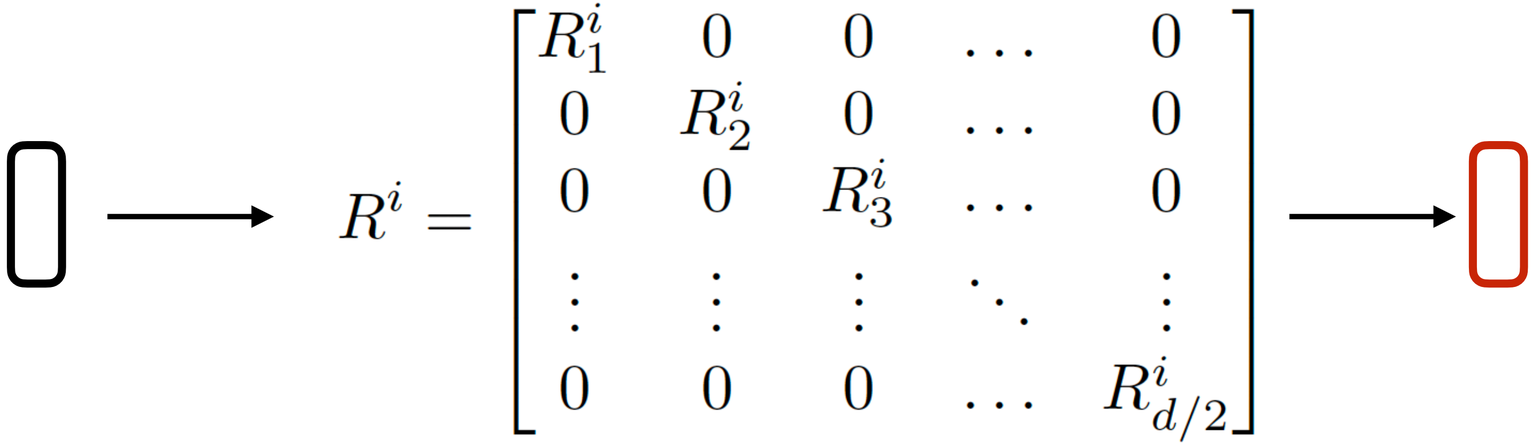
$= 34.03°$

RoPE (Jianlin Su et al., 2021)

$$\theta_{i,k} = \frac{i}{\Theta^{(2k-2)/d}}$$

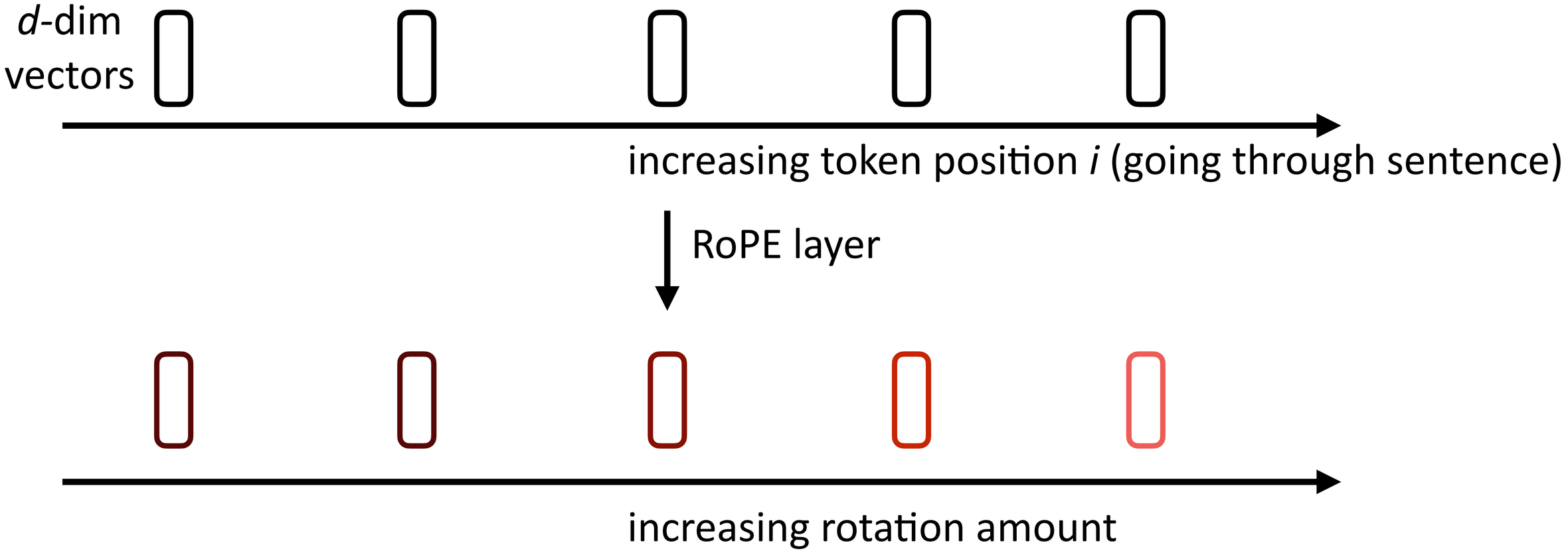
What happens as *i* increases?

What happens as *k* increases?



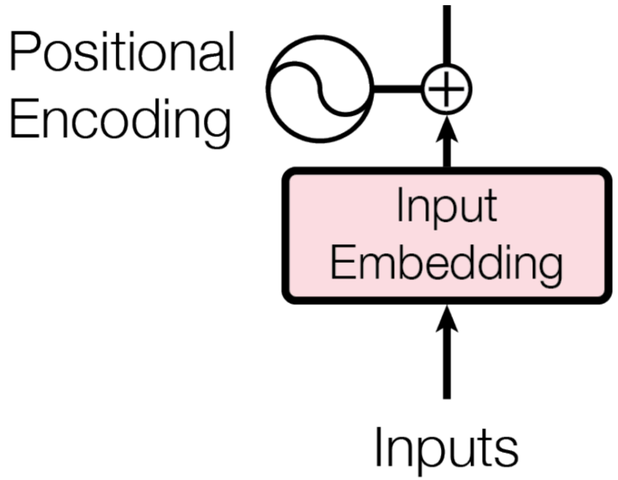
$R^i x^i$: rotate initial positions heavily, further-on positions less so

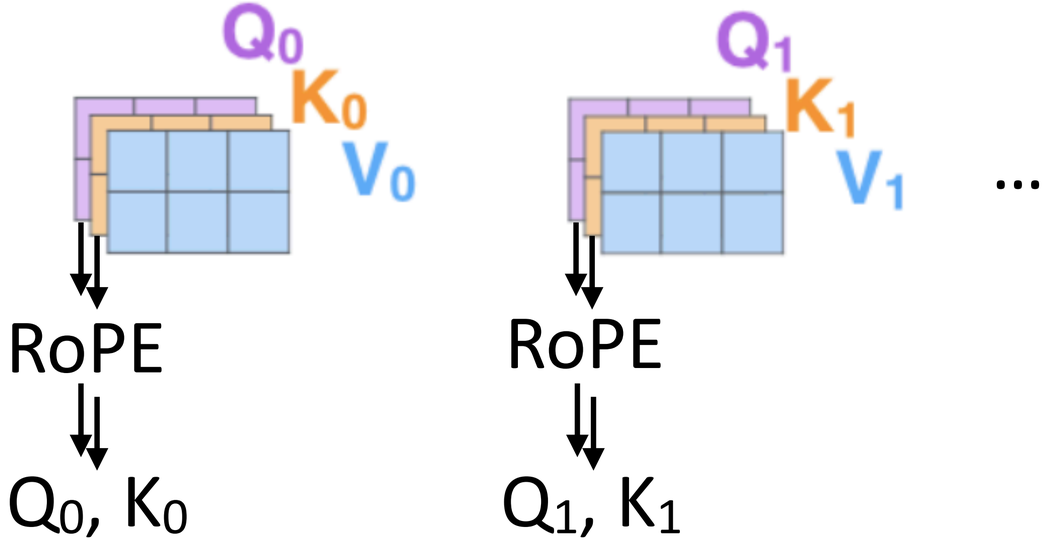
RoPE (Jianlin Su et al., 2021)



How does this help encode position for a Transformer?

Where are PEs used?





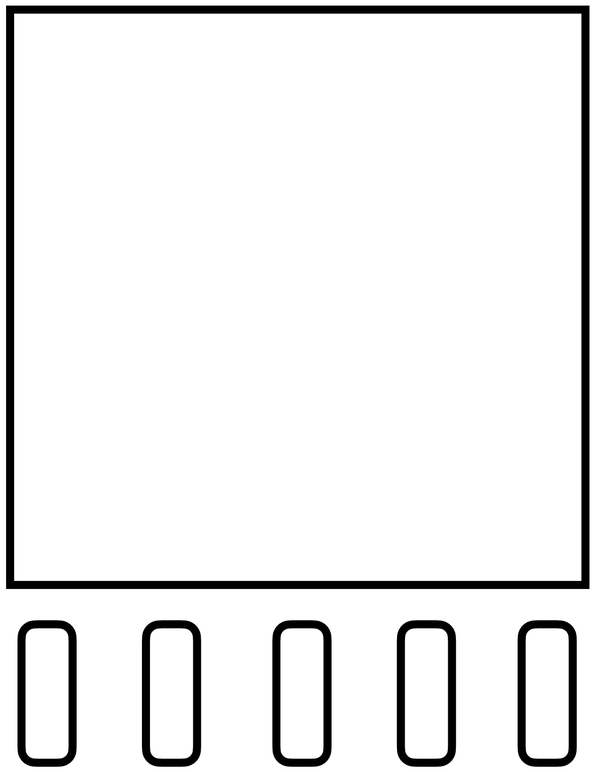
Classical Vaswani et al. Transformer (2017): added to input

Modern practice: Apply RoPE to Qs and Ks right before self-attention

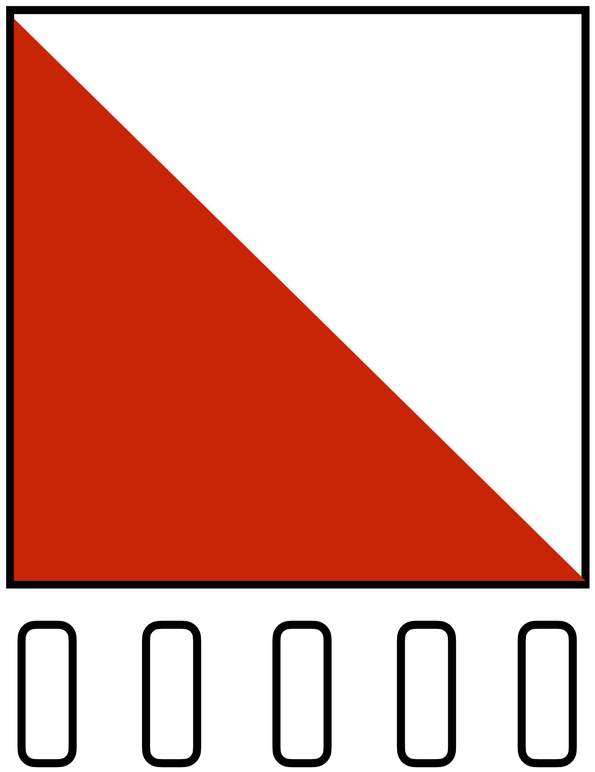
- How do these methods differ?

NoPE (Kazemnejad et al., 2023)

- Do we *actually* need positional encodings?



Full attention



Causal mask

- See what happens in Assignment 1!

YaRN (Bowen Peng et al., 2023)

Position interpolation: if RoPE is trained with encodings up to token *L*, you can expand it to *L'* by

$$\theta_{i,k} = \frac{i\,L/L'}{\Theta^{(2k-2)/d}}$$

Notion of wavelength at position *k*: the number of tokens needed to do a full rotation

$$\lambda_k = \frac{2\pi}{\theta_k} = 2\pi\,\Theta^{\frac{2k}{|D|}}$$

Two ideas in YaRN:

1. If wavelength is small (*r* is large), we do not use position interp. If wavelength is large, then we use position interp.
2. Introducing a temperature *t* on the attention computation to further rescale it

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Decoding

Decoding Strategies

- LMs place a distribution $P(y_i \mid y_1, \ldots, y_{i-1})$
- seq2seq models place a distribution $P(y_i \mid \mathbf{x}, y_1, \ldots, y_{i-1})$
- Generation from both models looks similar; how do we do it?
  - Option 1: max y_i $P(y_i \mid y_1, \ldots, y_{i-1})$ — take greedily best option
  - Option 2: use beam search to find the sequence with the highest prob.
  - Option 3: sample from the model; draw y_i from that distribution
- When should we use these different approaches?

Decoding Strategies

- Story generation with GPT-2:

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

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town, and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV reporters. We don't even stick around to be interviewed by TV reporters. Maybe that's how they figured out that they're cosplaying as the Bolivian Cavalleros."

- Beam search degenerates and starts repeating. If you see a fragment repeated 2-3x, it has very high probability to keep repeating
- Sampling is too noisy — introduces many grammatical errors

Holtzman et al. (2019)

Degeneration

- Beam search fails because the model is *locally normalized*
- Let's look at all the individual decisions that get made here

Beam Search, *b*=32:
"The study, published in the Proceedings of the National Academy of Sciences of the United States of America (PNAS), was conducted by researchers from the Universidad Nacional Autónoma de México (UNAM) and the Universidad Nacional Autónoma de México (UNAM/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de México/Universidad Nacional Autónoma de ..."

P(Nacional | … Universidad) is high

P(Autónoma | … Universidad Nacional) is high

P(de | … Universidad Nacional Autónoma) is high

P(México | Universidad Nacional Autónoma de) is high

P(/ | … México) and P(Universidad | … México /) — these probabilities may be low. But those are just 2/6 words of the repeating fragment

- **Each word is likely given the previous words but the sequence is bad**

Holtzman et al. (2019)

Drawbacks of Sampling

- Sampling is “too random”

Pure Sampling:
They were cattle called Bolivian Cavalleros; they live in a remote desert uninterrupted by town and they speak huge, beautiful, paradisiacal Bolivian linguistic thing. They say, 'Lunch, marge.' They don't tell what the lunch is," director Professor Chuperas Omwell told Sky News. "They've only been talking to scientists, like we're being interviewed by TV

P(*y* | … they live in a remote desert uninterrupted by)

0.01 roads
0.01 towns
0.01 people
0.005 civilization

Good options, maybe accounting for 90% of the total probability mass. So a 90% chance of getting something good

…
0.0005 town

Long tail with 10% of the mass

Holtzman et al. (2019)

Nucleus Sampling

P(*y* | … they live in a remote desert uninterrupted by)

0.01 roads

0.01 towns

0.01 people

0.005 civilization

cut off after *p*% of mass → renormalize and sample

- Define a threshold *p.* Keep the most probable options account for *p*% of the probability mass (the *nucleus*), then sample among these.
- To implement: sort options by probability, truncate the list once the total exceeds *p*, then renormalize and sample from it

Holtzman et al. (2019)

Decoding Strategies

- LMs place a distribution $P(y_i \mid y_1, \ldots, y_{i-1})$
- seq2seq models place a distribution $P(y_i \mid \mathbf{x}, y_1, \ldots, y_{i-1})$
- How to generate sequences?
  - Option 1: max y_i $P(y_i \mid y_1, \ldots, y_{i-1})$ — take greedily best option
  - Option 2: use beam search to find the sequence with the highest prob.
  - ~~Option 3: sample from the model; draw y_i from that distribution~~
  - Option 4: nucleus sampling

Holtzman et al. (2019)

Administrative details

Understanding Language

Why Language Modeling?

Language Modeling

Transformers

Transformer Language Modeling

Positional Encodings

Decoding

Next time

Tokenizers
Optimizers
Tricks
Efficiency