# Course Overview / Foundations

CSE 5539: Special Topics in Large Language Models

https://go.osu.edu/cse-5539-fall-2026

THE OHIO STATE UNIVERSITY

# Logistics

- Instructor: Sachin Kumar
  - Primary area of research: Natural Language Processing.

- Office Hours: Mondays 5-6pm in DL 581 or by appointment.

# First week attendance

- TopHat

# Course structure

- This is a seminar course.
  - The course is primarily based on presentations & discussion of research papers.

# Course structure

- This is a seminar course.
  - The course is primarily based on presentations & discussion of research papers.

- Main goals of the course:
  - Get students up to speed with the latest developments in language modeling.
    - Overall theme: Constraint-Driven Progress in Language Modeling Research.
    - Main foci this semester: reasoning, long-context/efficiency, tool-use/agents, high-stakes applications.
  - Help students build or improve research skills (from literature reviews and critiquing prior work, to brainstorming ideas and implementing them).

# Preliminaries: What I Expect From You

- Familiar / comfortable with language modeling foundations.
  - **Modeling:** neural networks, language models, transformers, …
  - **Training:** gradient descent, backpropagation, optimizers, Supervised Finetuning, LoRA, basics of "alignment."
  - **Measuring quality:** perplexity, accuracy, benchmarking, LLM-as-a-judge, …
- Being open to reading papers and presenting their gist to the class.
- Participate in discussions in the classroom.

# Homework to test foundational knowledge

- Later today, a homework will be released on Canvas and will be due mid next week (Wednesday, September 9).
  - The only homework in this course.

- It is intended to measure your understanding of the foundational concepts of language modeling.
  - If you took the last 5539 I taught, don't worry, it is not as tedious.

- This is to make sure that when coming in, you know all the pre-requisites needed for the class.

# Questions so far?

# Class Format

- The class will be **in-person**.
- Each session will involve **the presentation/discussion** of recent important papers on NLP / Language Models.
- The course also involves **a project**.

# Class Presentations

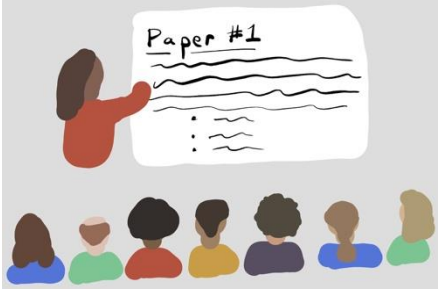
- Role-based presentation

**Role-Playing Paper-Reading Seminars**

Alec Jacobson and Colin Raffel
March 17th, 2021
colinraffel.com/blog

https://colinraffel.com/blog/role-playing-seminar.html


- Role-based presentation vs.
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.
  - The "role" defines **the lens** through which you read/present a paper.




- One-to-Many presentations
  - A single (subgroup of) student(s) presenting a paper to the class.
- Pro:
  - Easy division of labor
- Cons:
  - Too much work for one person
  - Audience easy to disengage


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** Stakeholder ✍

Act as if you're the author of this paper. Try to sell it!


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** Scientific Reviewer 🔎

Do a complete conference-style critical peer-review of the paper.


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** Archaeologist

Determine the [prior and recent] literature that inspired and was inspired by this work.


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** Visionary

Propose an imaginary follow-up -- research project or a new application.


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** Reproducer

Suppose you are tasked with reproducing the paper as faithfully as possible. Is it possible with the published artifact?


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** AI Reviewer

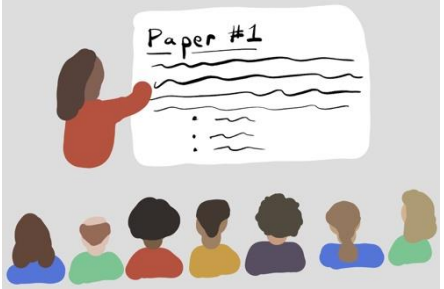
Before reading the paper, prompt an AI system to review it for you, then read it, and critique the AI review.


- Role-based presentation
  - Many students **cooperatively** present a paper.
  - Each subgroup of students takes a specific **"role"**.

**Role:** Theme Context

Explain how this paper relates to the others we have read and the theme of the seminar.


- Role-based presentation
  - Many students **cooperatively** present a paper.
- Pro:
  - More engagement
  - Distributed and less workload
  - Present more frequently
- Cons:
  - Need to manage role assignment




- One-to-Many presentations
  - A single (subgroup of) student(s) presenting a paper to the class.
- Pro:
  - Easy division of labor
- Cons:
  - Too much work for one person
  - Audience easy to disengage
  - Present a 1-2 times only.

# Role-Based Presentation

- We will discuss two (thematically related) papers each week.
- Each member of the presenting group will be given a random role every week.
  - The presenters will be assigned at least 10 days before the class.
- Each role has a time budget:
  - ~15 mins for Stakeholder 
  - ~5-10 mins for reviewer, visionary, archaeologist.
  - AI reviewer, reproducer, theme-context(er) will not present but submit their findings on Canvas/TopHat before the lecture.
- Each paper will take around ~45-50 minutes (~5-10 min break between two presentations)

# Non-presenter Activity

- **Before the class: All students are expected to read the 2 papers**
  - Students who are not presenting, need to prepare at least one question/thought about each paper:
    - Could be anything you are confused about or something you'd like to hear discussed more, or an open-ended question
  - Submit your questions the night before the class (due midnight EST)
    - Where? TBD
  - We will use these questions partly as discussion points
    - Avoid generic questions/statements (e.g., What is their learning rate? How long did they train? Didn't understand their intro)
    - Aim to be probing, analytical, and thought-provoking by offering specific critical comments or questions.
- **During the class:** come to class ready to participate in the discussions.
  - You may come up with other questions in the class as the paper is being presented.
  - I may randomly call upon you to ask a question / add to the discussion, please be prepared to participate.

# Questions so far?

# Class Project

- Group projects (team size = 2 to 3 students)
  - 3 students are allowed for projects with a larger proposed scope

- What is the goal of the final project?
  - Conduct research on problem related to the course theme and submit a written report.
  - I will post example projects/topics on teams with requirements etc this week.

# Class project timeline

- **September 11:** Form teams (and inform me).
- **September 21:** project proposal (1-2 page)
  - Should describe what is the main goal of the project, the proposed research, and how it connects to existing work in the field.
- **November 2:** progress report (~4 pages)
  - Describe the main problem, project goal and related work, what has been done so far, any initial results, and the plan continuing the project.
- **December 7:** project presentations in class.
- **December 14**: final reports due (~8 pages).

# Generative AI Policy

- AI use is permitted but please be careful. You are ultimately responsible for every part of your submission. Acknowledge any and all AI use in your submissions.

- For paper presentations/reading: If you are using AI to understand the papers / prepare presentation / questions, please verify all the details. The reviewer role should NOT be using AI. AI-reviewer role (obviously) MUST use AI. I might randomly call upon you to make sure you read the paper(s).

- For the projects, feel free to use any AI coding tools (again, you are responsible for any code you write). For writing reports, only allowed use is for minor editing/polishing, content must be yours. Learning how to communicate is part of your doing research and writing helps organize your thoughts. Blatantly AI-written reports will get a failing grade.

# Resources

- Teams: primary medium of communication, find the link on course website and on Canvas (announcement).
- CarmenCanvas: all instructor posts will be cross-posted on Canvas but please don't post questions on there, I do plan to look. Foundation homework should be submitted on Canvas.
- TopHat: For ocassional quizzes.
- For more course details / logistics, please make sure to read the course website: https://go.osu.edu/cse-5539-fall-2026

# Questions?

# Language Models

# The cat sat on the __

# The cat sat on the mat

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

next word

context or prefix

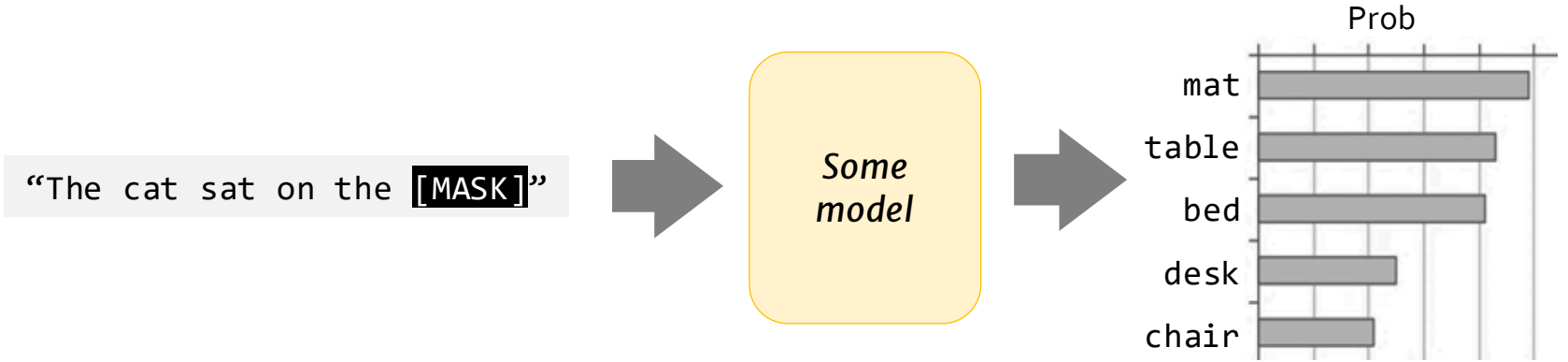
$$\mathbf{P}(\underbrace{X_t}_{\text{next word}} | \underbrace{X_1, \ldots, X_{t-1}}_{\text{context}})$$

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

next word — context

But more broadly,

$$\mathrm{P}(X_1, \ldots, X_t) = \mathrm{P}(X_1)P(X_2|\mathrm{X}_1) \ldots \mathrm{P}(\mathrm{X}_\mathrm{t}|\mathrm{X}_1, \ldots, X_{t-1})$$




Language Modeling ≙ learning prob distribution over language sequence.

# Doing Things with Language Model

- What is the probability of ….

"I like The Ohio State University"

"like State I University The Ohio State"

# Doing Things with Language Model

- What is the probability of ….

  "I like The Ohio State University"

  "like State I University The Ohio State"

- LMs assign a probability to any string of tokens.

  P("I like The Ohio State University") = 10^-5

  P("like State I University The Ohio State") = 10^-15

# Doing Things with Language Model (2)

- We can rank sentences.

next word context

$$\mathbf{P}(X_t \mid X_1, \ldots, X_{t-1})$$

- While LMs show "typicality", this may be a proxy indicator to other properties:
  - Grammaticality, fluency, factuality, etc.

**P**("*I like The Ohio State University. EOS*") > **P**("*I like Ohio State University EOS*")
**P**("*OSU is located in Columbus. EOS*") > **P**("*OSU is located in Pittsburgh. EOS*")

# Doing Things with Language Model (3)

- Can also generate strings!

next word context

$$\mathbf{P}(X_t \mid X_1, \ldots, X_{t-1})$$

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

1. Sample from **P**(X | *"Ohio State is "*) → "located"
2. Sample from **P**(X | *"Ohio State is located"*) → "in"
3. Sample from **P**(X | *"Ohio State is located in"*) → "the"
4. Sample from **P**(X | *"Ohio State is located in the"*) → "state"
5. Sample from **P**(X | *"Ohio State is located in the state"*) → "of"
6. Sample from **P**(X | *"Ohio State is located in the state of"*) → "Ohio"
7. Sample from **P**(X | *"Ohio State is located in the state of Ohio"*) → "EOS"

# Why Care About Language Modeling?

- Language Modeling is/used-to-be a part of many tasks:
  - Summarization
  - Machine translation
  - Spelling correction
  - General purpose Instruction following (ala ChatGPT/Claude/Gemini etc) – most of the course will focus on developments related to this.

- Language Modeling is a reasonably effective proxy for language understanding – there is still a lot of debate about the term.
  - Effective ability to predict forthcoming words requires on understanding of context/prefix.

# Summary

- **Language modeling:** building probabilistic distribution over language.

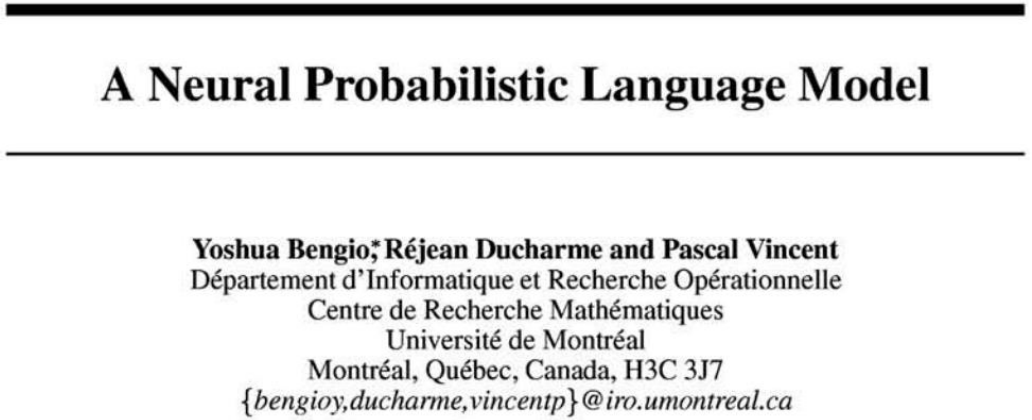
- An accurate distribution of language enables us to solve many important tasks that involve language communication.

- **Question**: how do you actually estimate this distribution?

# “Historical” Approaches to Language Modeling

- Probabilistic n-gram models of text generation [Jelinek+ 1980’s, …]
- “Shallow” neural language models using feed-forward networks (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*

# LMs w/ Recurrent Neural Nets

Example: the cat sat on the mat

- Core idea: apply a model repeatedly

outputs — output distribution

$$\hat{\boldsymbol{y}}^{(t)} = \text{softmax}\left(\boldsymbol{U}\boldsymbol{h}^{(t)} + \boldsymbol{b}_2\right) \in \mathbb{R}^{|V|}$$

hidden states

$$\boldsymbol{h}^{(t)} = \sigma\left(\boldsymbol{W}_h\boldsymbol{h}^{(t-1)} + \boldsymbol{W}_e\boldsymbol{e}^{(t)} + \boldsymbol{b}_1\right)$$

$\boldsymbol{h}^{(0)}$ is the initial hidden state

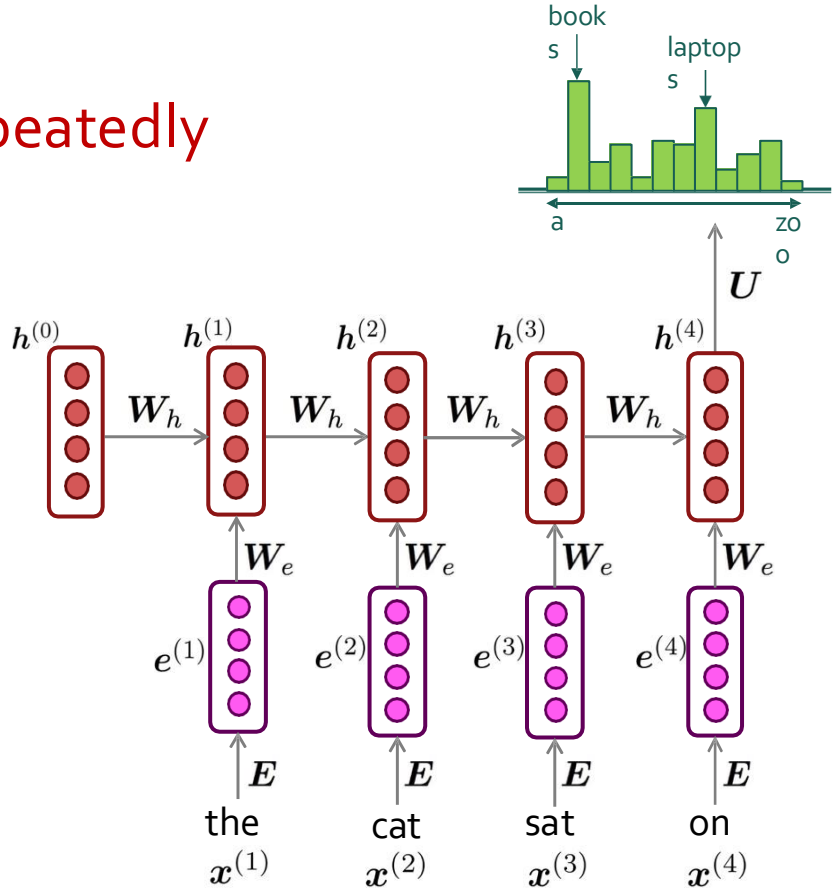
Input embedding

word embeddings

$$\boldsymbol{e}^{(t)} = \boldsymbol{E}\boldsymbol{x}^{(t)}$$

words / one-hot vectors

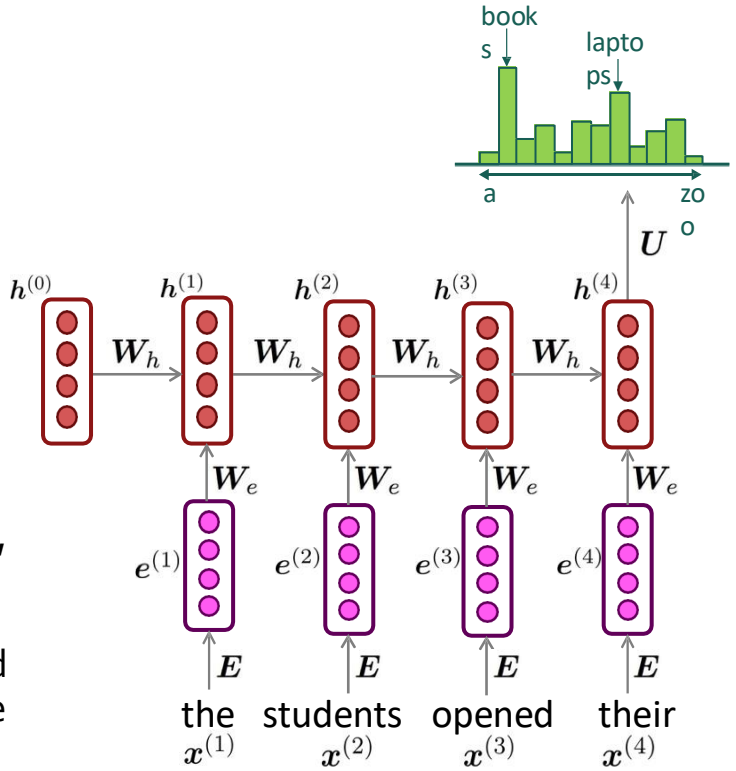
$$\boldsymbol{x}^{(t)} \in \mathbb{R}^{|V|}$$



# RNNs: Pros and Cons

- **Advantages**:
  - Model size is constant with respect to sequence length
  - Computation for step *t* can (in theory) use information from many steps back

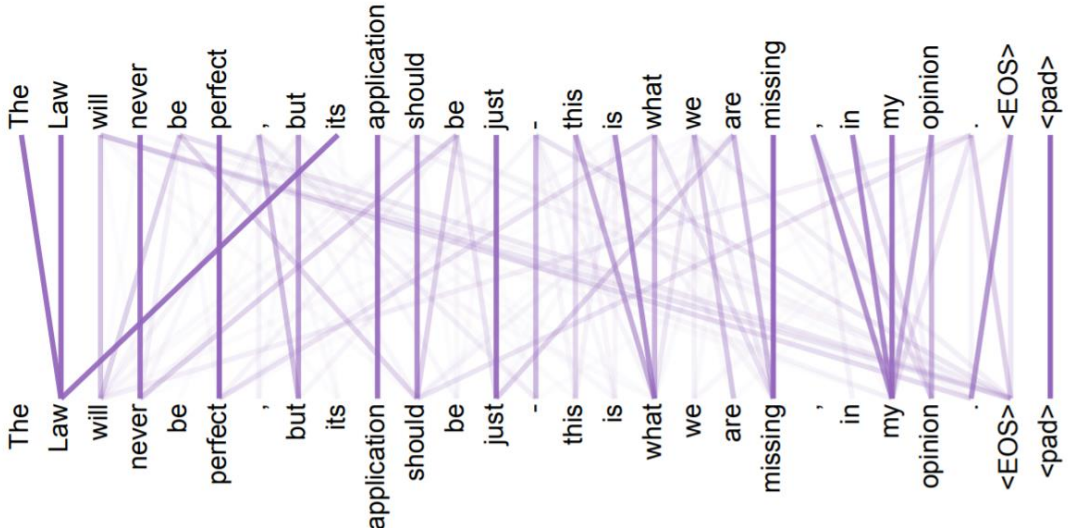
- **Disadvantages**:
  - Recurrent computation is slow.
  - While RNNs in theory can represent long sequences, they quickly forget portions of the input.
  - Vanishing/exploding gradients.
  - To fix these issues, many variations were experimented with and one that has stood the test of time are attention-based models i.e. transformers.
  - But RNNs (in different forms) are coming back now.




[adopted from Chris Manning]

# Attention

- Core idea: on each step of the generation, use *direct connection* to *focus ("attend") on a particular part* of the context.




[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

# Defining Self-Attention

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

- **Definition:** Given a set of vector **values**, and a vector **query**, *attention* is a technique to compute a weighted sum of the **value**, dependent on the **query**.

[Vaswani et al. 2017: https://arxiv.org/abs/1706.03762]

$q$: query (to match others)

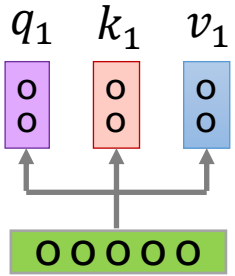
$$q_i = W^q x_i$$

$k$: 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)

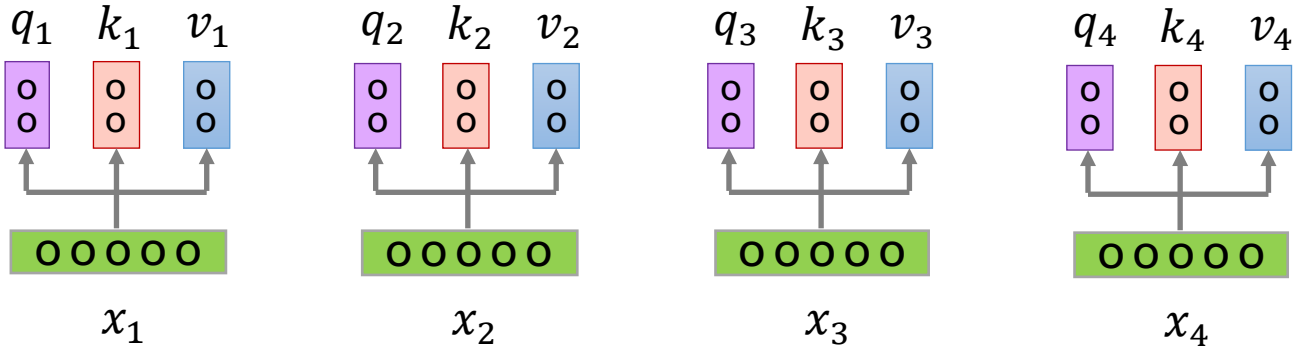
$$q_i = W^q x_i$$

$k$: key (to be matched)

$$k_i = W^k x_i$$

$v$: value (information to be extracted)

$$v_i = W^v x_i$$



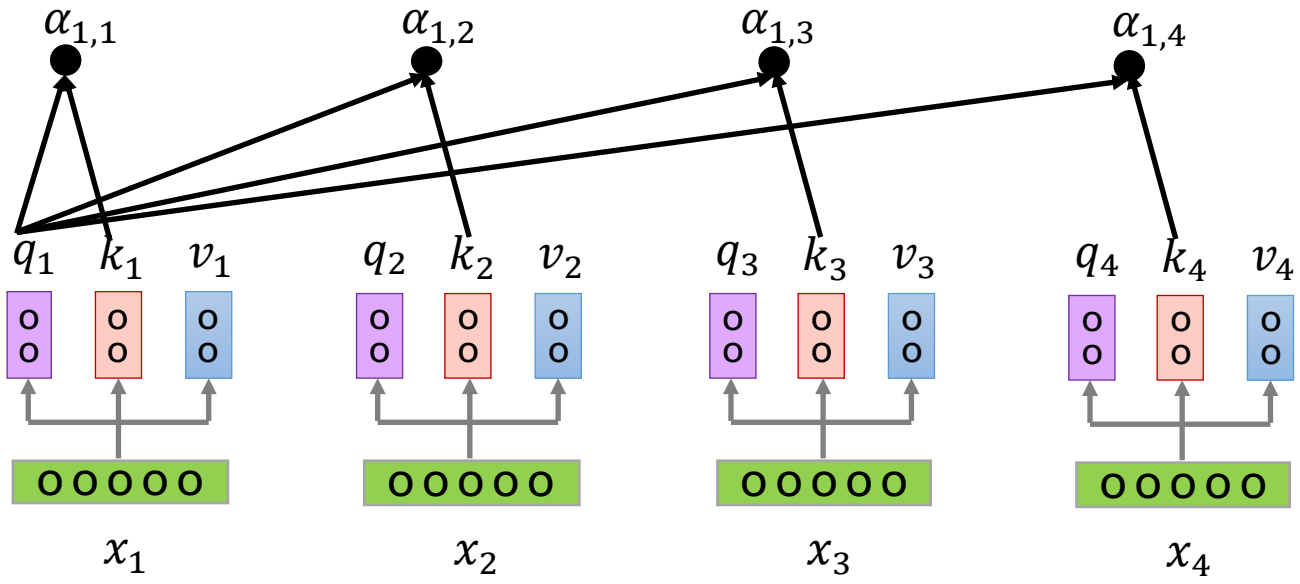
$$\alpha_{1,i} = \underbrace{q^1 \cdot k^i / \alpha}_{\text{Scaled dot product}}$$

$q$: query (to match others)

$k$: key (to be matched)

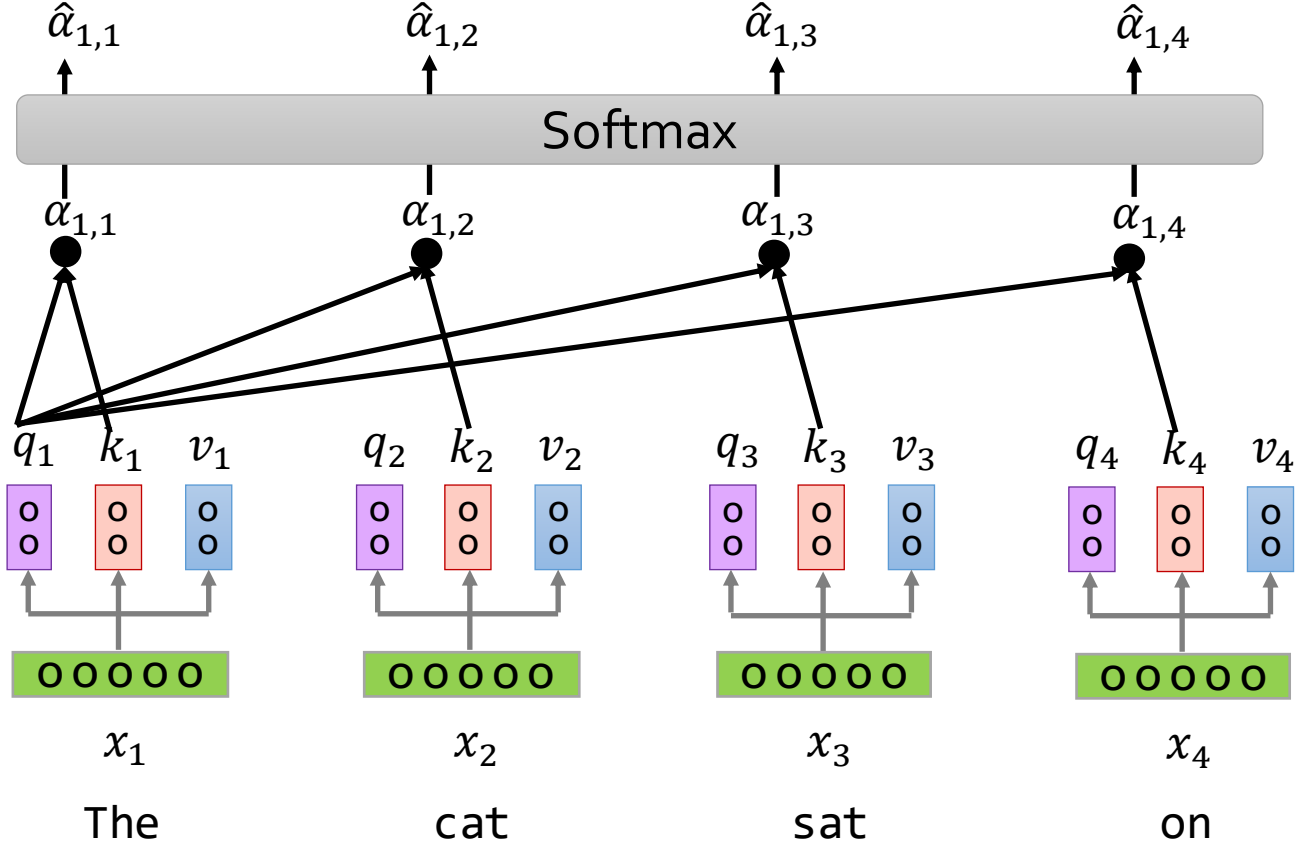
$v$: value (information to be extracted)

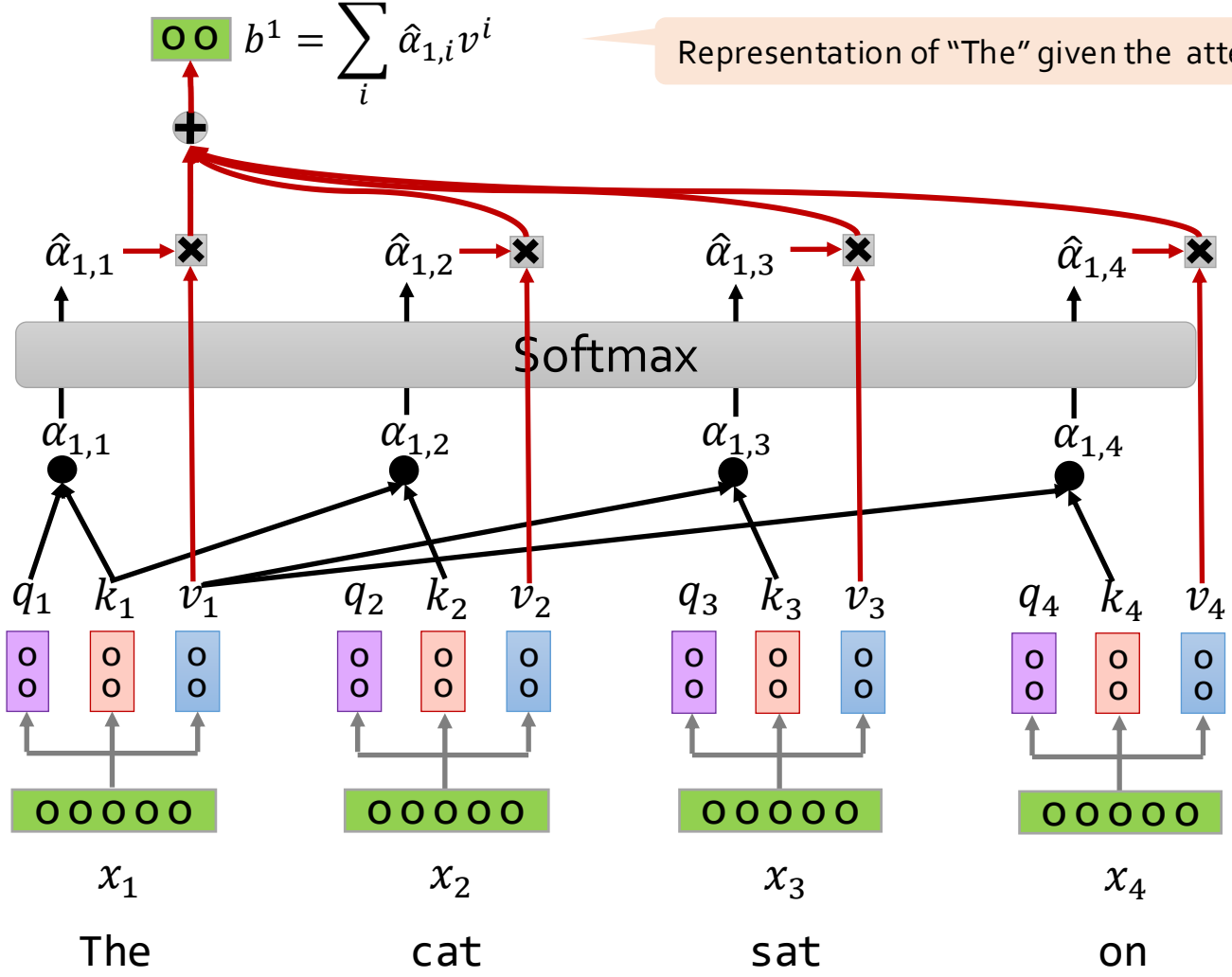
How much should "The" attend to other positions?

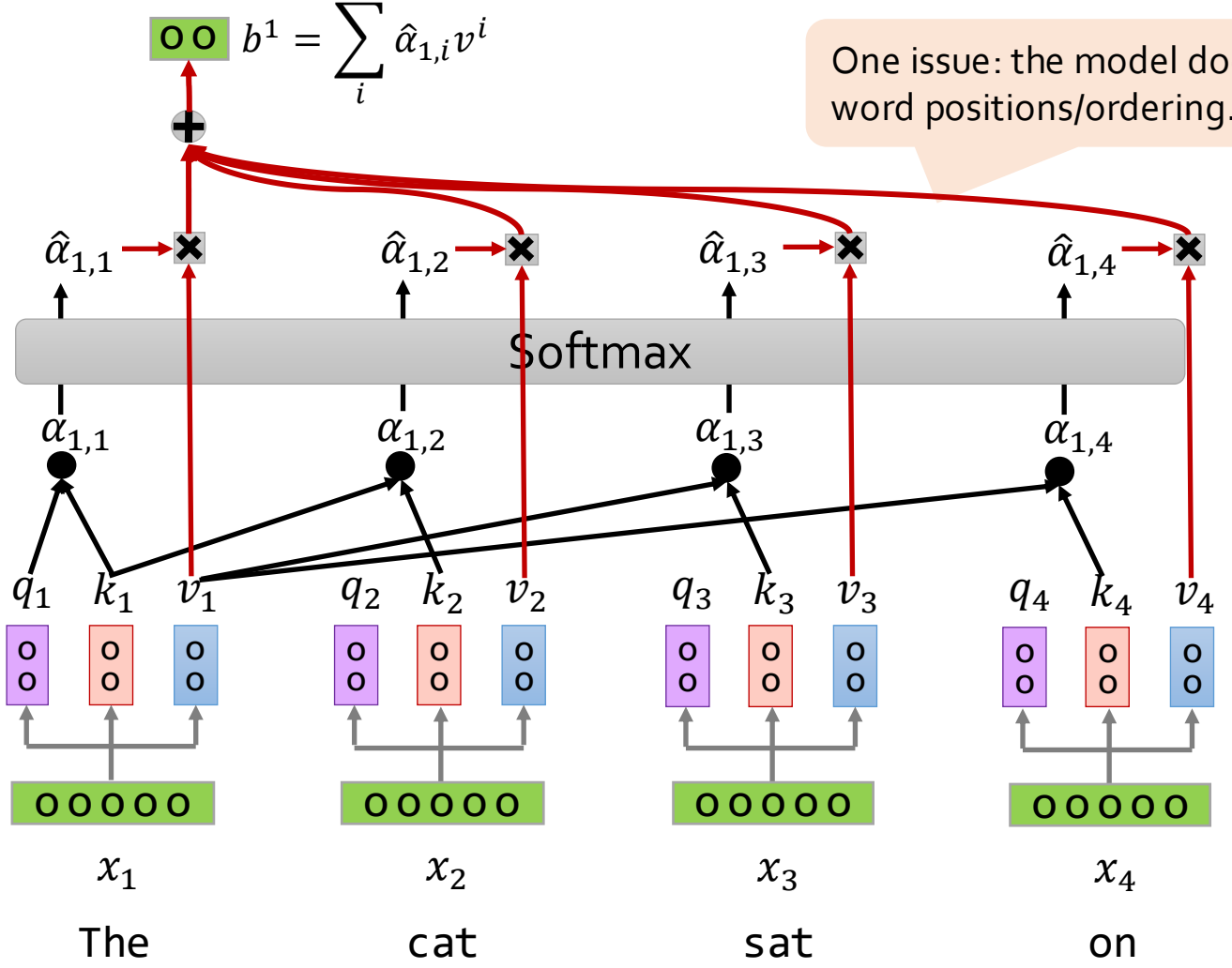


$$\sigma(z)_i = \frac{exp(z_i)}{\sum_j exp(z_j)}$$

How much should "The" attend to other positions?



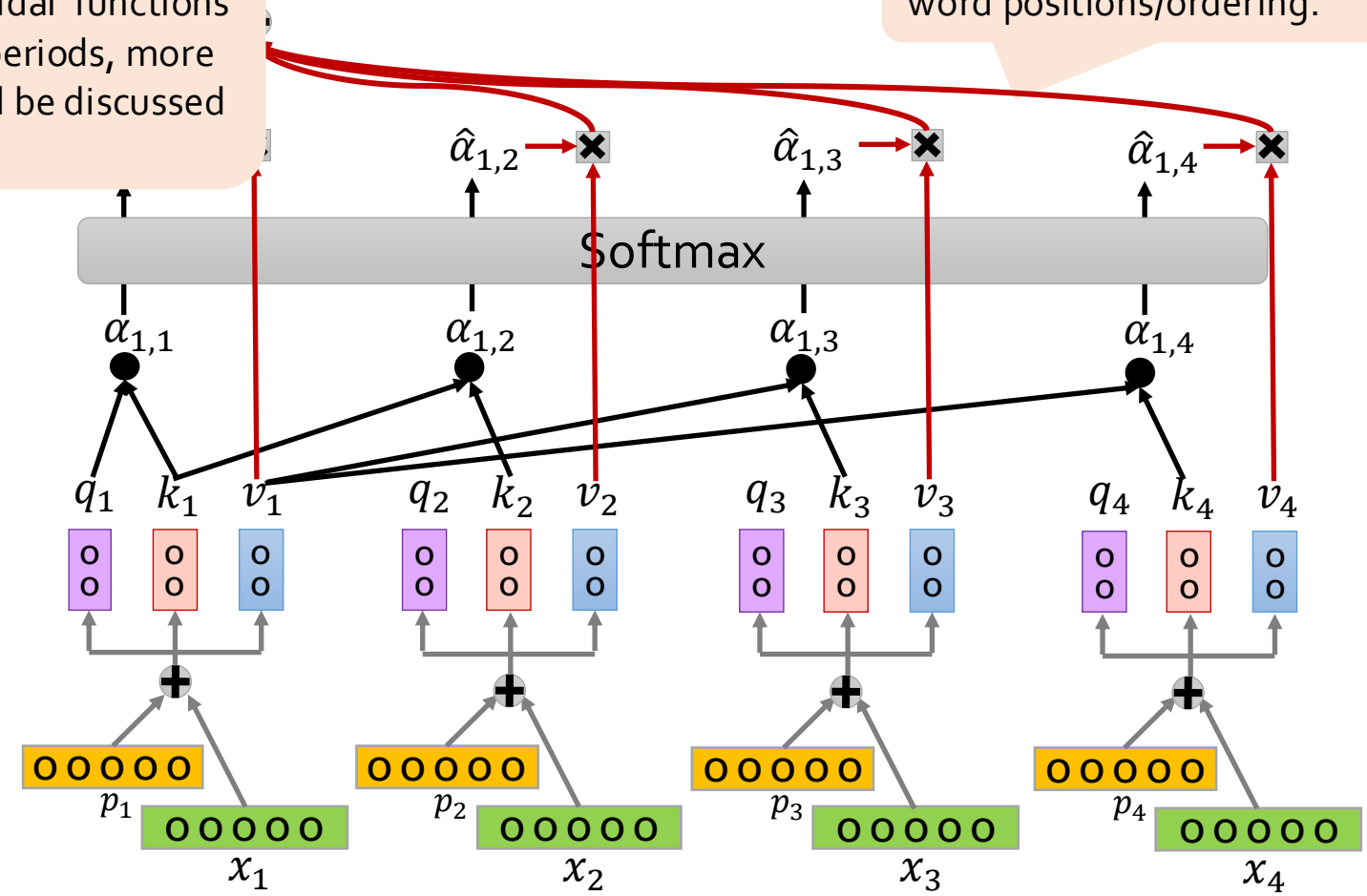

$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
Representation of "The" given the attention weights
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
$q_1$ $k_1$ $v_1$
$q_2$ $k_2$ $v_2$
$q_3$ $k_3$ $v_3$
$q_4$ $k_4$ $v_4$
$x_1$
$x_2$
$x_3$
$x_4$
The
cat
sat
on


$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$
One issue: the model doesn't know word positions/ordering.
$\hat{\alpha}_{1,1}$
$\hat{\alpha}_{1,2}$
$\hat{\alpha}_{1,3}$
$\hat{\alpha}_{1,4}$
Softmax
$\alpha_{1,1}$
$\alpha_{1,2}$
$\alpha_{1,3}$
$\alpha_{1,4}$
$q_1$ $k_1$ $v_1$
$q_2$ $k_2$ $v_2$
$q_3$ $k_3$ $v_3$
$q_4$ $k_4$ $v_4$
$x_1$
$x_2$
$x_3$
$x_4$
The
cat
sat
on

$b^1 = \sum_i \hat{\alpha}_{1,i} v^i$

$p_i$ are unique fixed vectors (e.g. sinusoidal functions of varying periods, more variants will be discussed later).

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



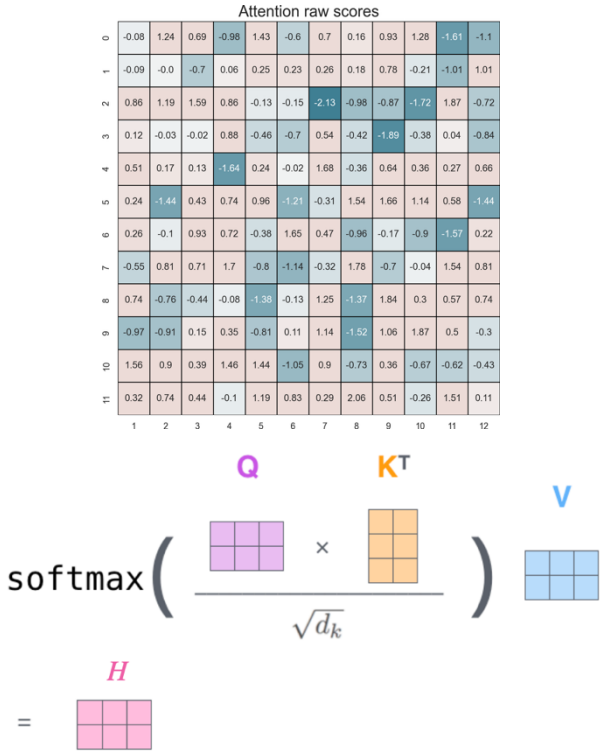
# Self-Attention: Matrix Notation

- Input sequence: $\mathbf{x} = [x_1, x_2, \ldots, x_n] \in \mathbb{R}^{n \times d}$
- Calculate:

$$\boldsymbol{Q} = \mathbf{x}\mathbf{W}^q \in \mathbb{R}^{n \times d}$$
$$\boldsymbol{K} = \mathbf{x}\mathbf{W}^k \in \mathbb{R}^{n \times d}$$
$$\boldsymbol{V} = \mathbf{x}\mathbf{W}^v \in \mathbb{R}^{n \times d}$$

- Pairwise similarity matrix between queries and keys: $\boldsymbol{Q}\boldsymbol{K}^T \in \mathbb{R}^{n \times n}$
- Attention output:

$$\text{Attention}(\mathbf{x}) = \text{softmax}\left(\frac{\boldsymbol{Q}\boldsymbol{K}^{\mathrm{T}}}{\sqrt{d}}\right)\boldsymbol{V} \in \mathbb{R}^{n \times d}$$

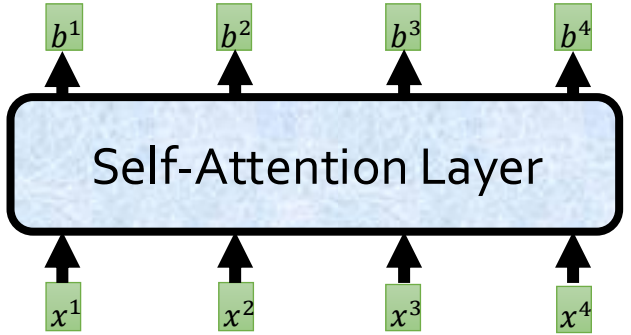


# Self-Attention: Back to Big Picture

- Attention is a way to focus on particular parts of the input
- Can write it in matrix form:

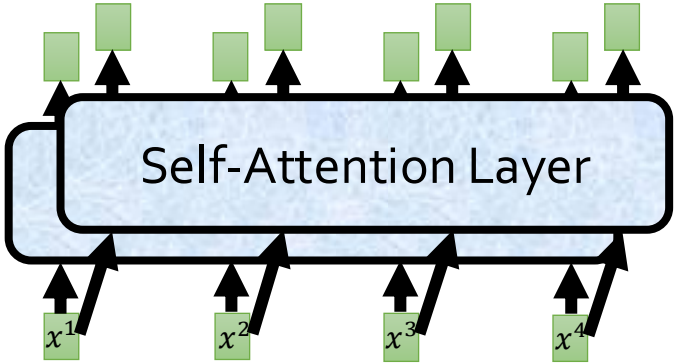
$$\boldsymbol{b} = \text{softmax}\left(\frac{QK^{\mathrm{T}}}{\alpha}\right)V$$

- Efficient implementations, can be batched easily.
- Better than RNNs at maintaining long-distance dependencies in the context.



# Multi-Headed Self-Attention

- Multiple parallel attention layers is quite common.
  - Each attention layer has its own parameters.

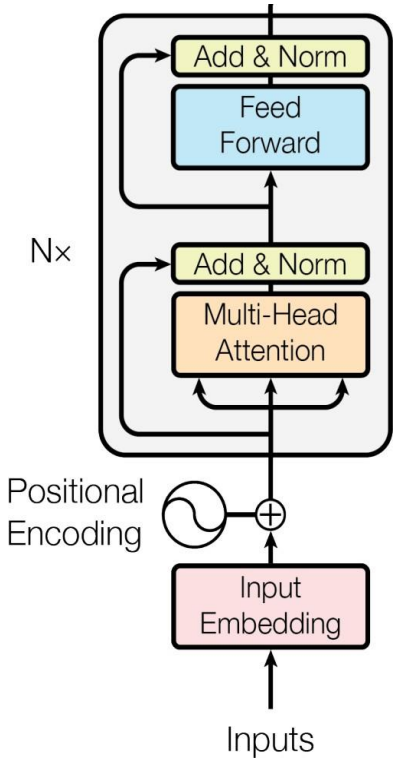



[Vaswani et al. 2017]

# Summary: Self-Attention Block

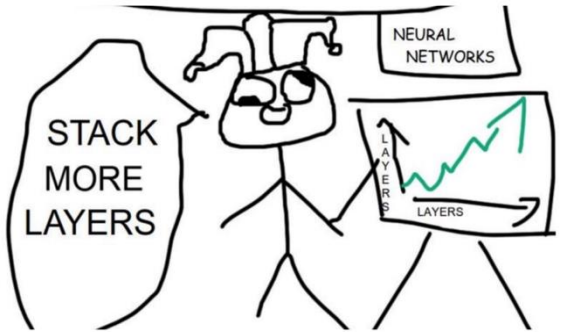
- **Self-Attention:** A critical building block of modern language models.
  - The intuitive idea is to compose meanings of words weighted according some similarity notion.

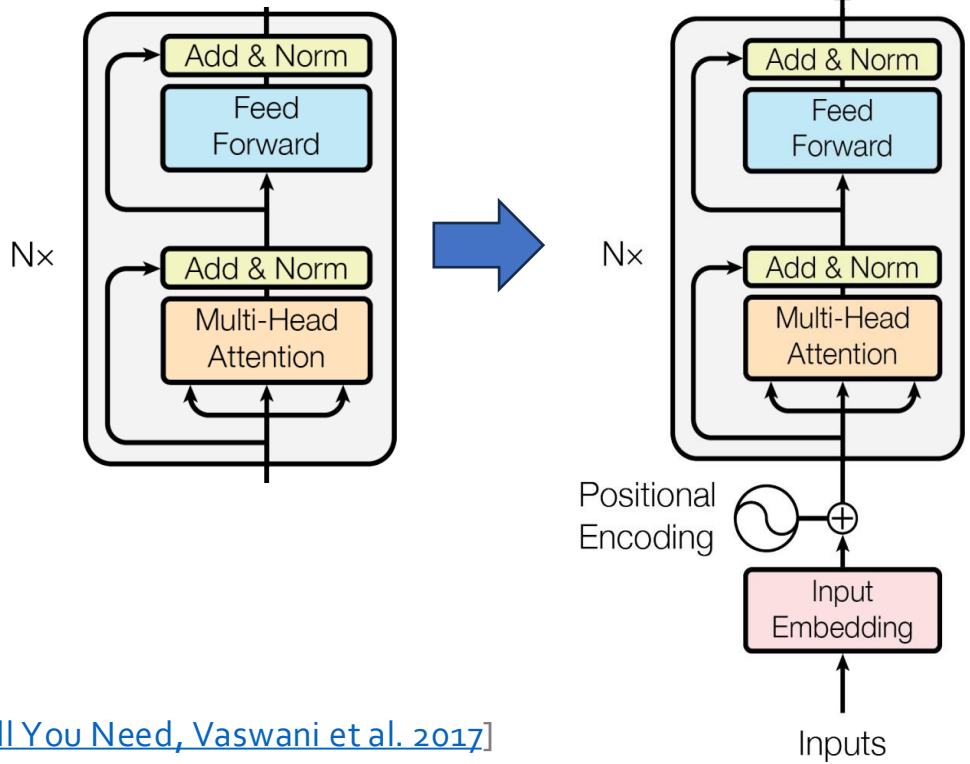
- Other additions added on top to make training feasible (look at suggested readings for more details):
  - Feedforward network (to add more capacity)
  - Positional embeddings (for positional information)
  - Residual connections (to improve gradient flow)
  - Layer Normalization (to stabilize gradient values)



# How Do We Make it Deep?

- Stack more layers!




[Attention Is All You Need, Vaswani et al. 2017]

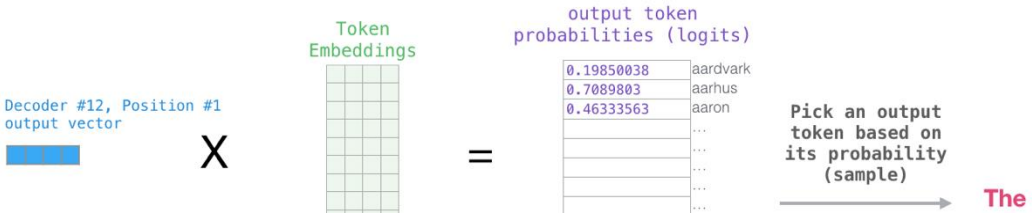
# From Representations to Prediction

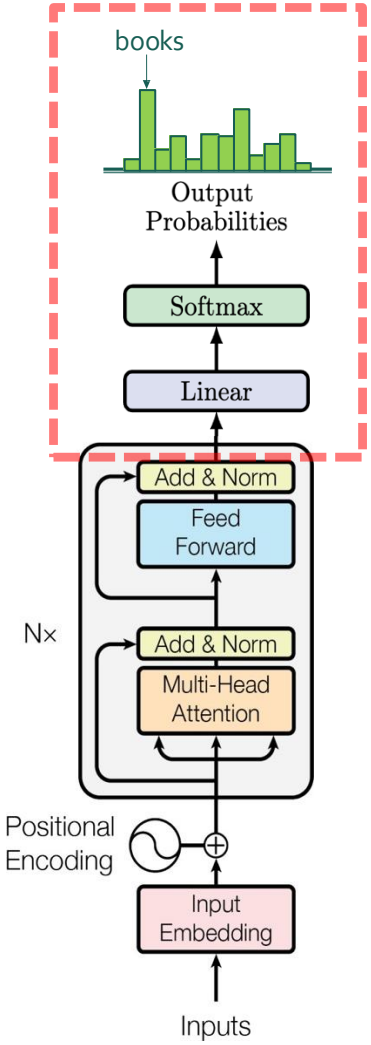
- To perform prediction, add a classification head on top of the final layer of the transformer: $\mathbf{x} \in \mathbb{R}^{n \times d}$.
  - Where $n$ is the length of your sequence.
- To obtain logits, we can apply a linear transformation with token embedding matrix $\mathbf{W_V} \in \mathbb{R}^{V \times d}$:

$$\mathbf{x}\mathbf{W}_\mathbf{V}^T \in \mathbb{R}^{n \times V}$$

- To obtain probabilities, run this through softmax:

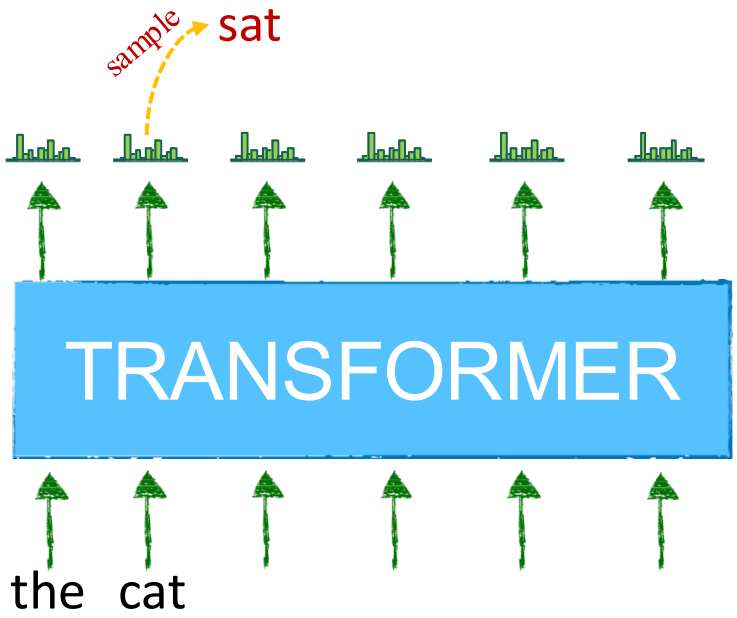
$$\text{softmax}(\mathbf{x}\mathbf{W_V}) \in \mathbb{R}^{n \times V}$$



# Generating text via Transformer

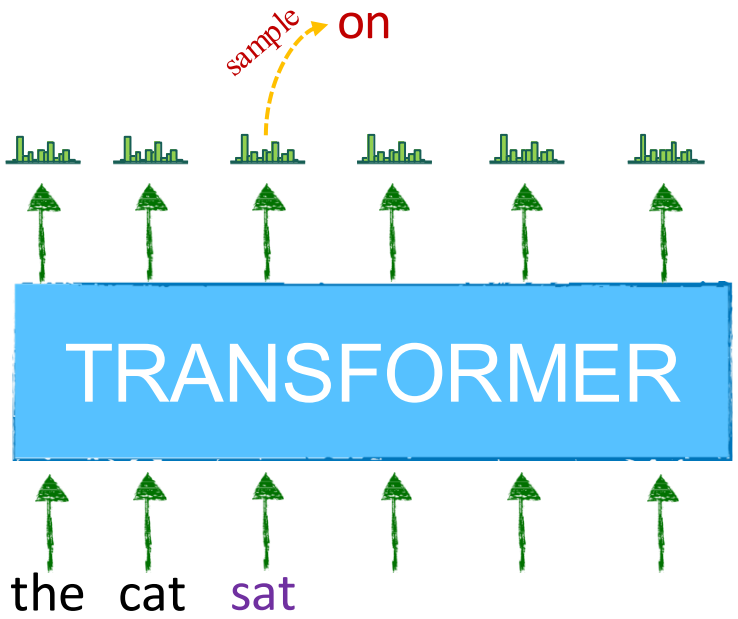
- Use the output of previous step as input to the next step repeatedly



# Generating text via Transformer

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

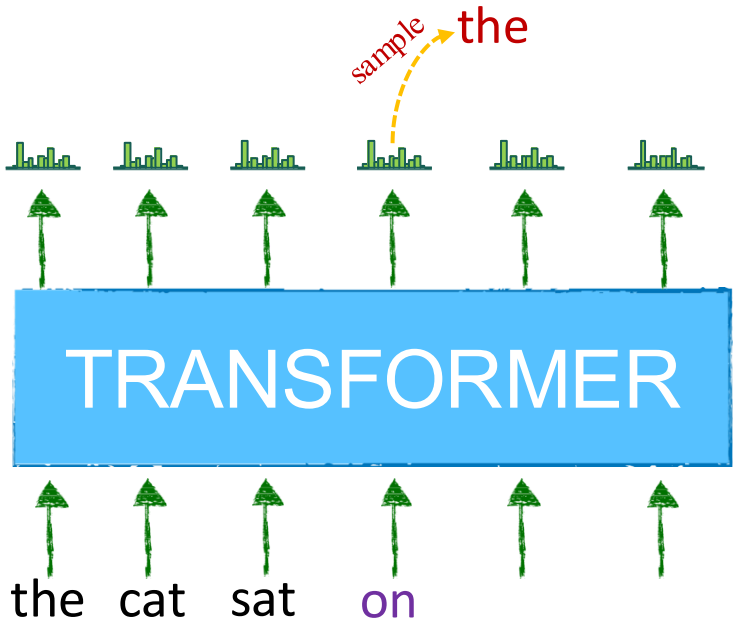
The probabilities get revised upon adding a new token to the input.



# Generating text via Transformer

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

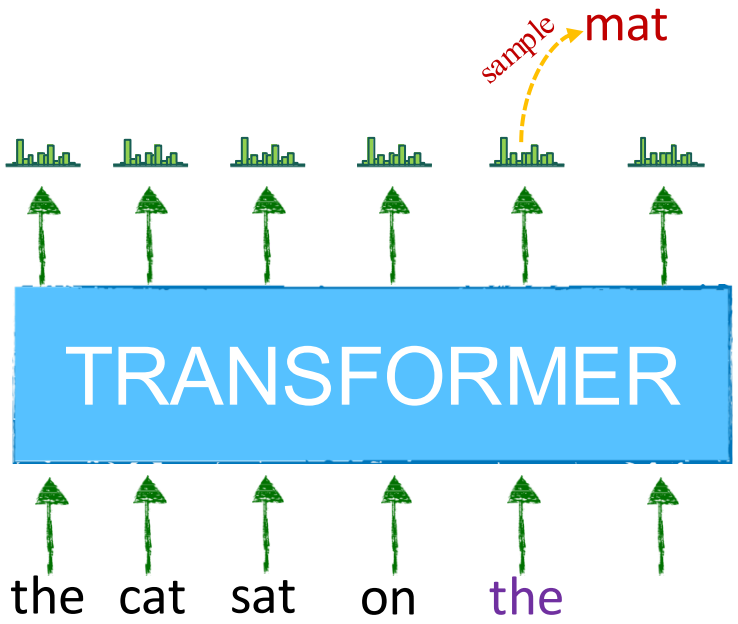
The probabilities get revised upon adding a new token to the input.



# Generating text via Transformer

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

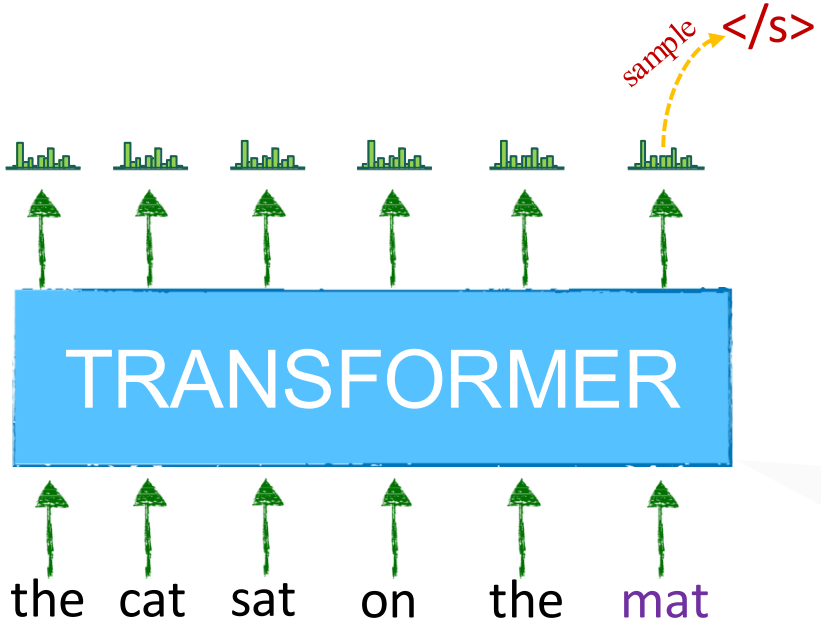
The probabilities get revised upon adding a new token to the input.



# Generating text via Transformer

- 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.



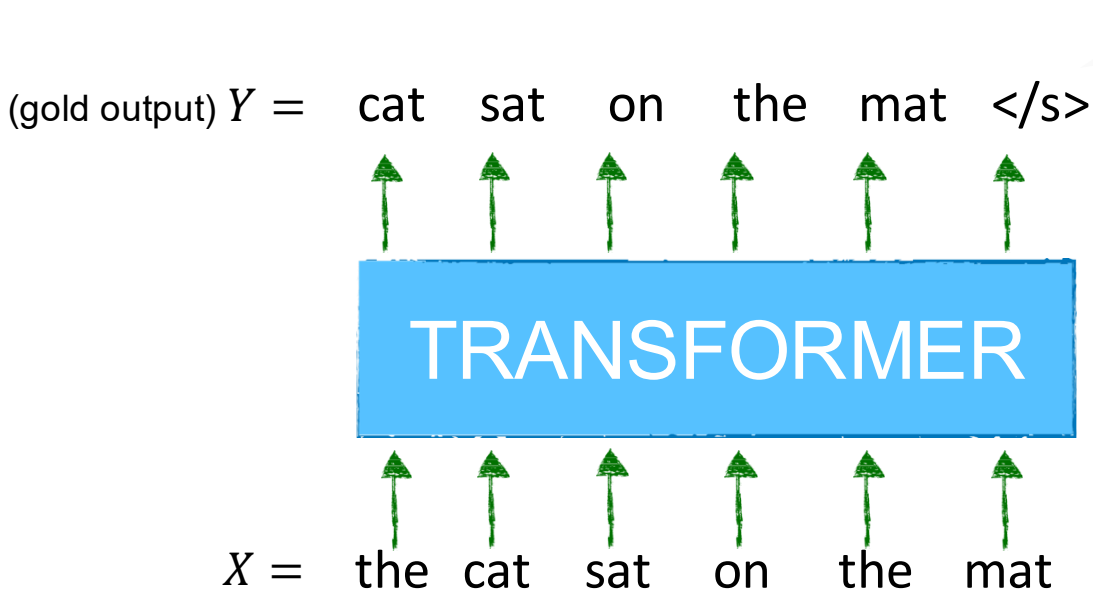

Note that each token generation is a forward pass – a lot of computations!!

And continue like that until we reach EOS or we get tired.

# How does training work?

# 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



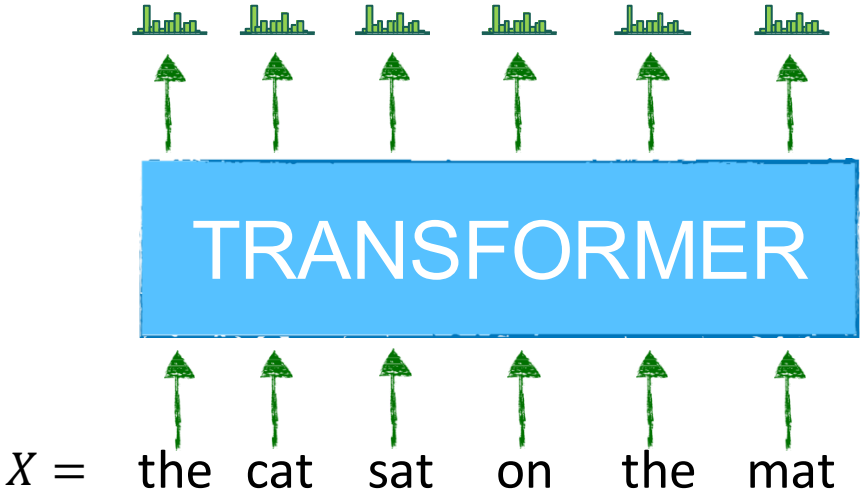

```
X = text[:, :-1]
Y = text[:, 1:]
```

[Slide credit: Arman Cohan]

# Training a Transformer Language Model

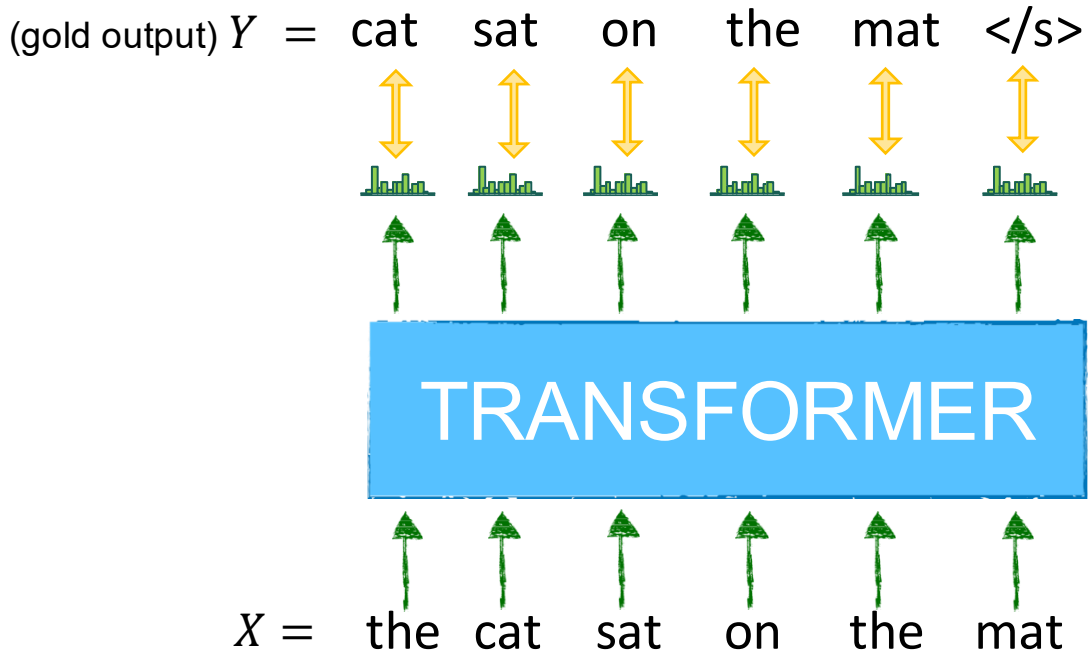
- For each position, compute their corresponding **distribution** over the whole vocab.

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



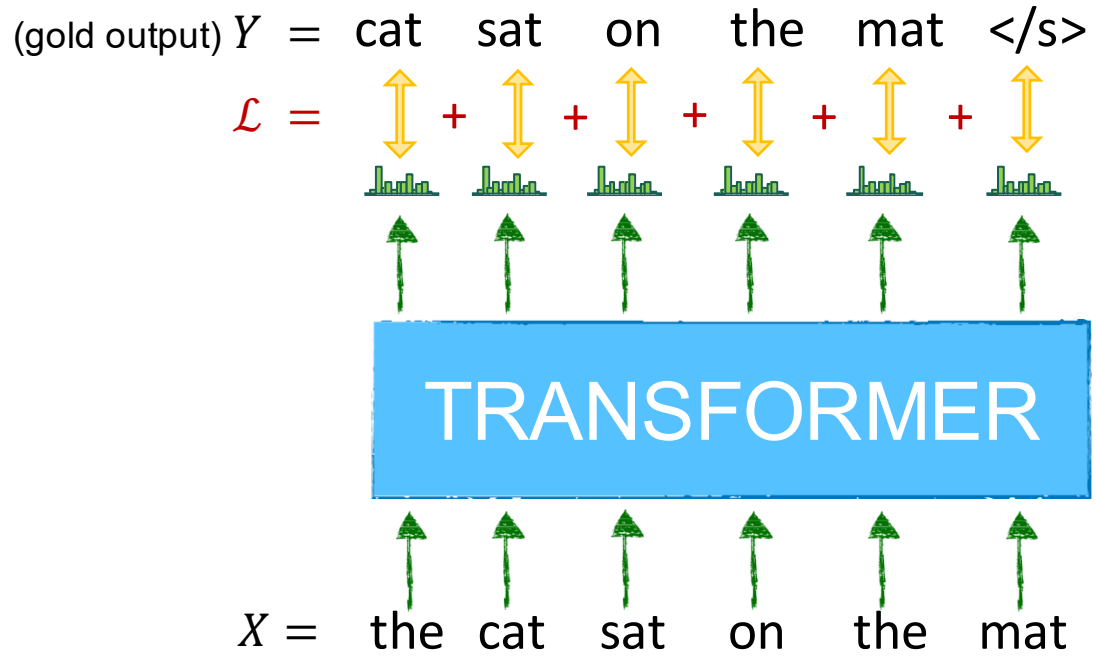
# Training a Transformer Language Model

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



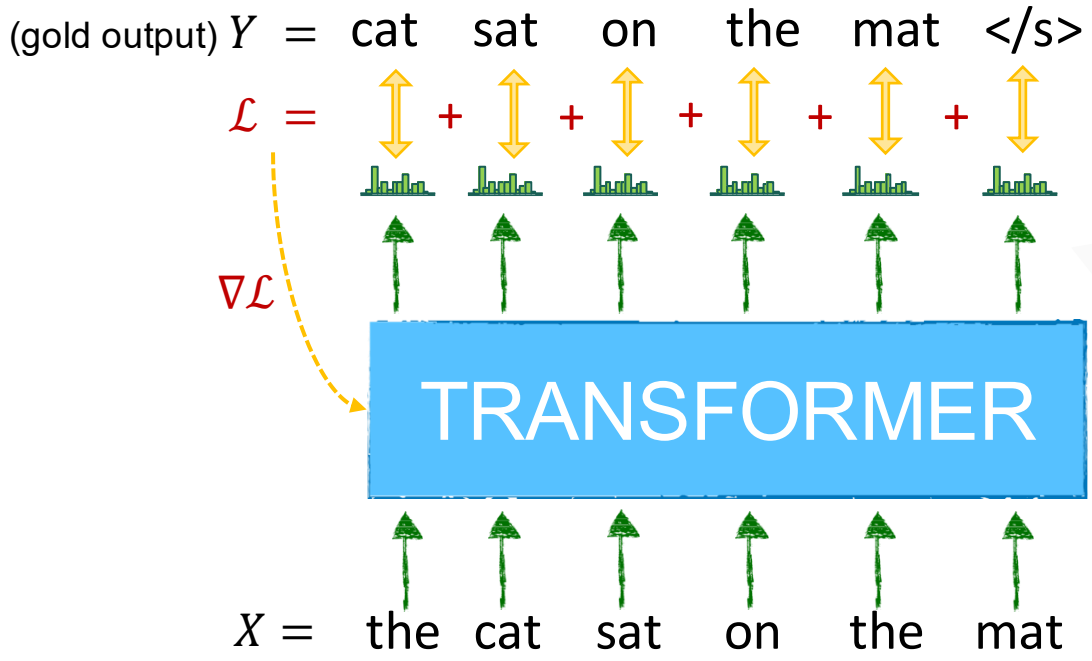
# Training a Transformer Language Model

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



# Training a Transformer Language Model

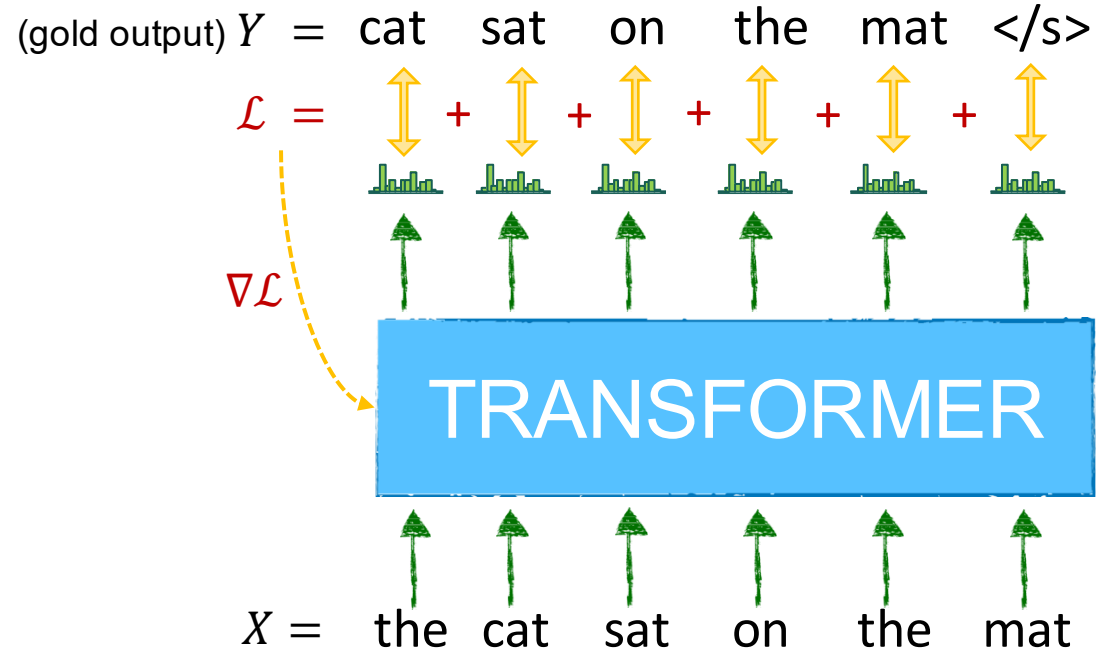
- Using this loss, do **Backprop** and **update** the Transformer parameters.




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

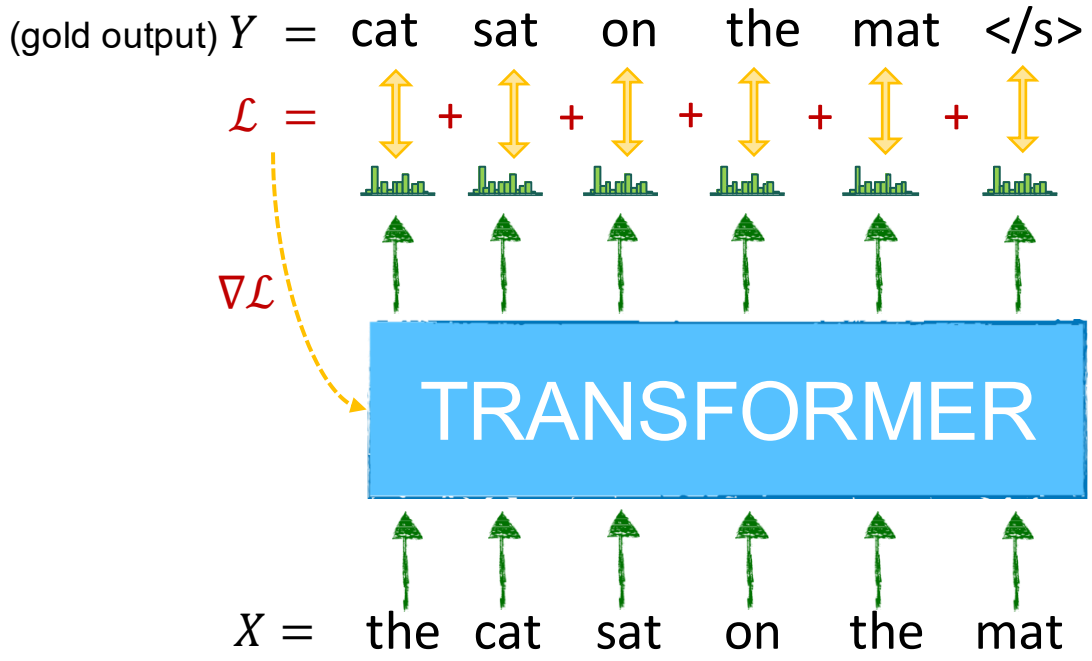
# Training a Transformer Language Model

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



# Training a Transformer Language Model

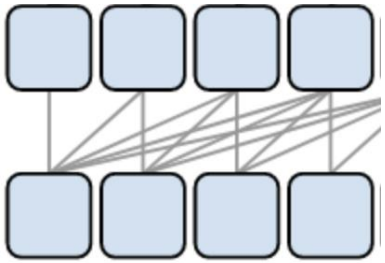
- We need to **prevent information leakage** from future tokens! How?



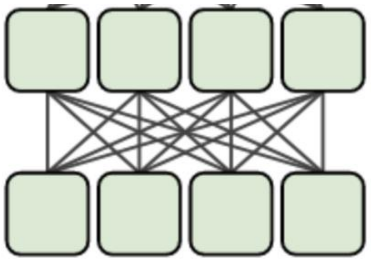
# Attention mask

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 |



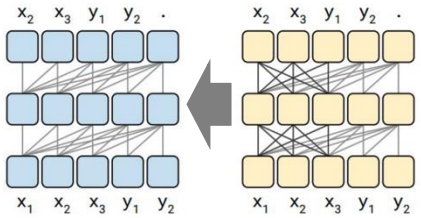
What we want



What we have

Slide credit: Arman Cohan

# 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

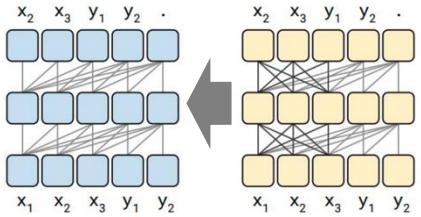
| | 0 | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 1 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 2 | 1.0 | 1.0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

0.30
0.25
0.20
0.15
0.10
0.05
0.00

large negative numbers,
which leads to softmax($-\infty$) $\approx$ 0

Slide credit: Arman Cohan

# 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 |

X

Element-wise product

Attention mask

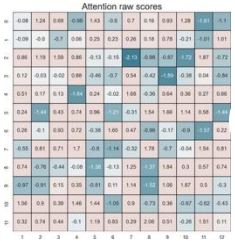
| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 1 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 2 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 3 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 4 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 5 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf | -inf |
| 6 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf | -inf | -inf |
| 7 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf | -inf |
| 8 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf | -inf |
| 9 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | 1.0 | -inf | -inf |
| 10 | 1.0 | 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 | 1.0 |

0.30 0.25 0.20 0.15 0.10 0.05 0.00

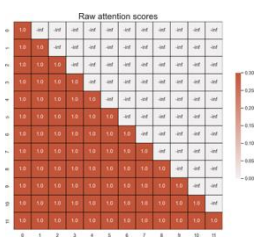
Note matrix operations is quite fast in GPUs.

Slide credit: Arman Cohan

# Attention mask


x




=

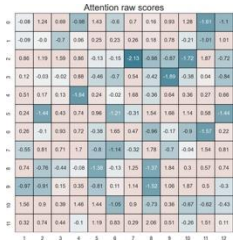
Masked attention raw scores

| | 1 | 2 | 3 | 4 | 5 | 6 | 7 | 8 | 9 | 10 | 11 | 12 |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| 0 | -0.08 | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf | -inf |
| 1 | -0.09 | -0.0 | -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 |
| 3 | 0.12 | -0.03 | -0.02 | 0.88 | -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 |
| 5 | 0.24 | -1.44 | 0.43 | 0.74 | 0.96 | -1.21 | -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 |
| 7 | -0.55 | 0.81 | 0.71 | 1.7 | -0.8 | -1.14 | -0.32 | 1.78 | -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 |
| 9 | -0.97 | -0.91 | 0.15 | 0.35 | -0.81 | 0.11 | 1.14 | -1.52 | 1.06 | 1.87 | -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 |
| 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 |

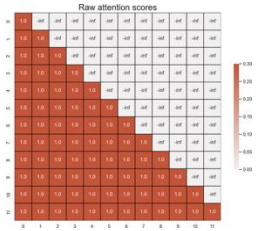
Slide credit: Arman Cohan

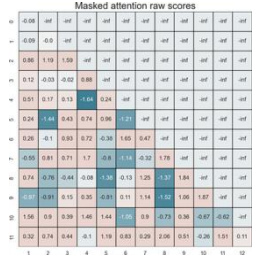
# Attention mask

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

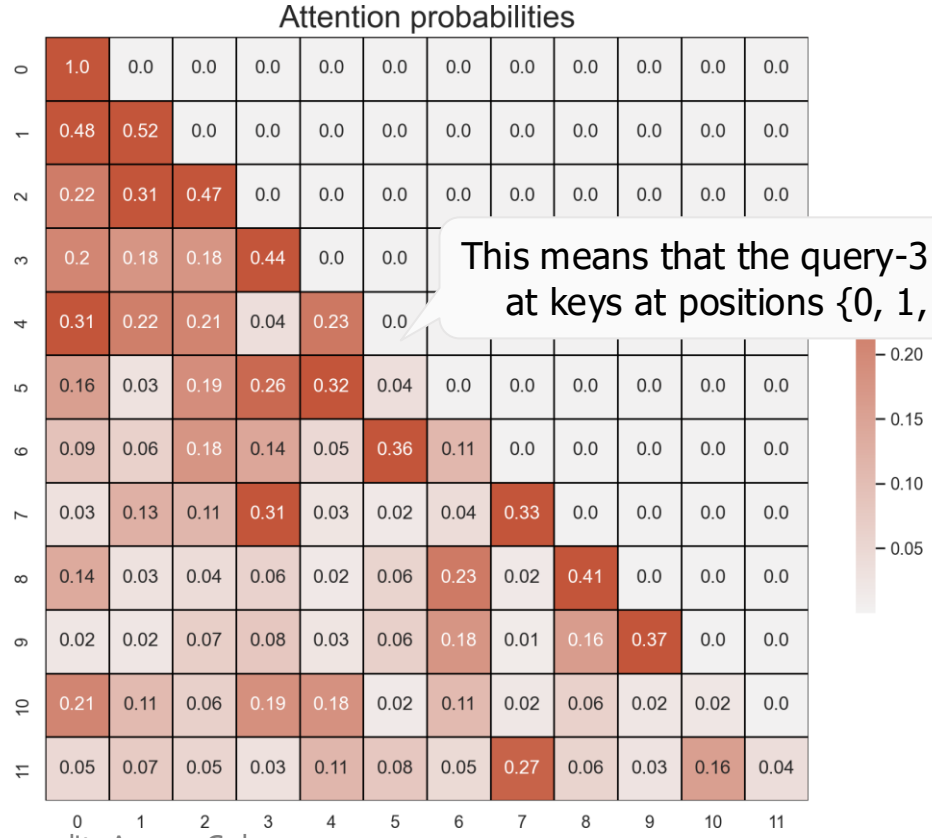



x




softmax

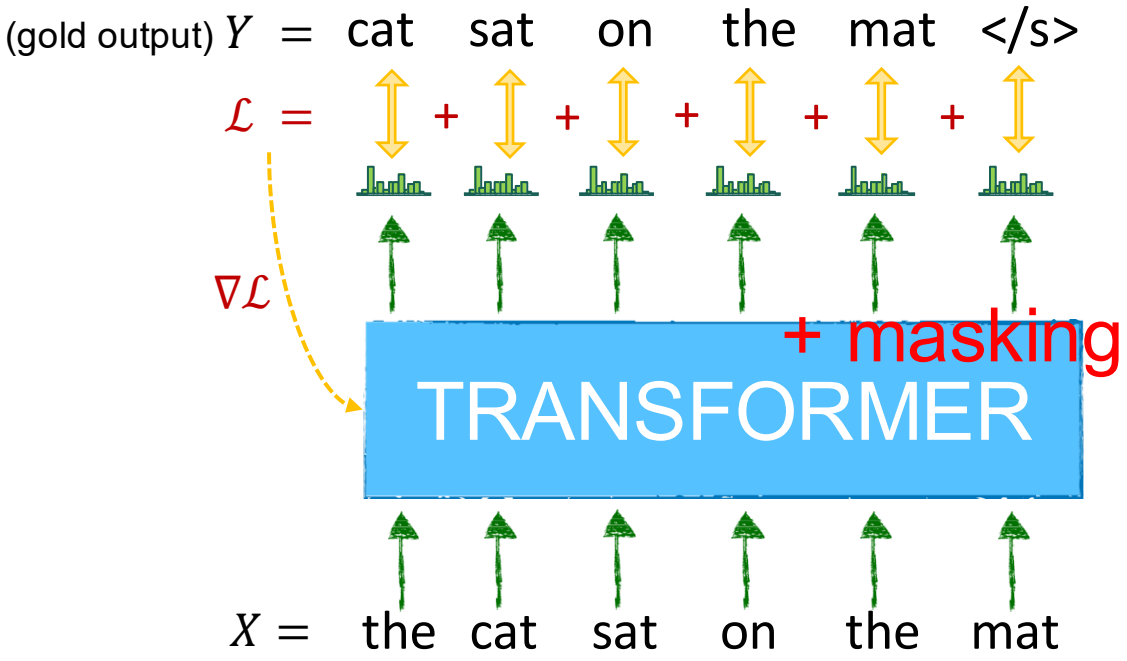



This means that the query-3 can look at keys at positions {0, 1, 2, 3}.

Slide credit: Arman Cohan

# Training a Transformer Language Model

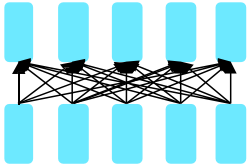
- We need to **prevent information leakage** from future tokens! How?




A transformer model with this type of masking is called a "decoder only" or "causal" or "autoregressive" language model.

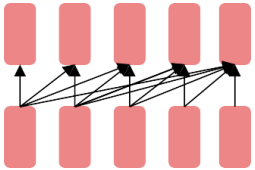
# Transformer variants

- A building block for a variety of LMs



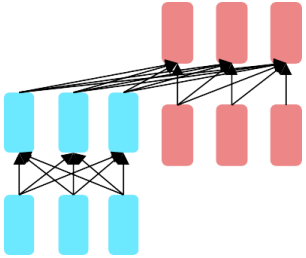
**Encoders**

- **Examples:** BERT, RoBERTa, SciBERT, ModernBERT, NeoBERT
- Captures bidirectional context.



**Decoders**

- **Examples:** GPT-2, and more, Llama, Qwen, etc.
- Other name: **causal or auto-regressive language model**
- Nice to generate from; can't condition on future words



**Encoder-Decoders**

- **Examples:** Transformer, T5, BART,

# Considerations about computational cost in Transformers

# Diversion: Floating-point Ops: 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,
- The total FLOP of a model (e.g., Transformer) provides a basic approximation of computational costs associated with that model.

$b$: batch size,
$n$: sequence length,
$d$: feature dimension in output of self-attention

# The bottlenecks

| Computations | IO |
|---|---|
| $O(bnd^2 + bn^2d)$ | $O(bnd + bmn^2 + d^2)$ |

- In total, we have ⟶
- The quadratic terms are based on $n$ and $d$
- $d$ is fixed (part of architecture) but $n$ changes with input.
- **Bottlenecks #1:** If $n$ (sequence length) $\gg d$ (feature dimension), the time and space complexity would be dominated by $O(n^2)$.
- **However,** these despite this quadratic dependence these are parallelizable operations which can be computed efficiently in GPUs.
  - In comparison, RNNs perform less arithmetic ops but they're not all parallelizable.
- **Bottlenecks #2:** Another potential bottleneck is how fast we can run IO.

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

[Vaswani et al. 2017]

# Summary of computational considerations.

- Transformers **computation of a full sequence** is bounded by $O(bn$ $bn^2d)$.
  - Generally, the quadratic term that depends on seq len $n$ is more concerning.

- We have not discussed yet how **IO may impose** other limits on this (later in the course).

- Also, the above calculations is for encoding a given input.
  - Later in the course, we will discuss computational issues during inference.

# A very brief review of the modern language modeling pipeline

# Outline

- Pretraining
- Post-training or “Alignment”
  - SFT
  - RLHF / RLVR / rest of the course basically.

# Pretraining

Train a randomly initialized model on a text corpora

# Pre-training data

# Where do we begin to collect data?

- Where do I find a very large dataset?
  - Crawling web is non-trivial (unless you're OpenAI / Anthropic / Google with ton of resources).
  - But if you have to do it, be aware that websites have their own permissions regarding which parts of their content, if any, can be crawled. Most companies don't respect these permissions. (search for copyright and related cases against all frontier AI companies).

  - The alternative is to look for websites that have done the crawling for you, such a commoncrawl.
    - A non-profit organization that release a new crawl of the internet **every month.**

# Pretraining data will not be a major part of this course

- Data does not fall from the sky. You have to work to get it!
- **Finding large data:** CommonCrawl has a ton of crawled dumps, but not the only one.
- **Cleaning data** can save tons of compute and even give you gains.
- **Repetitions** are often a waste of compute and deteriorate model quality.
- **Scaling deduplication** requires advanced data structures.
- **Old data** old data may skew your model predictions, but it depends on your application.
- **Data mixtures** are quite important, though depend on your downstream application.

# The pre-training data size and sources

- They vary quite a bit!
- They used to be in billions of tokens; now they're north of trillions.

| Model Name | Release | Pre-training data #Tokens | Training Dataset |
|---|---|---|---|
| **BERT** | 2018 | 3.3B | BooksCorpus (800M), English Wikipedia (2.5B) |
| **GPT-1** | 2018 | 13B | BooksCorpus |
| **GPT-2** | 2019 | 40B | WebText: scraping outbound links from Reddit post with >= 3 karma |
| **T5** | 2019 | 34B | C4 which is the cleaned up version of CommonCrawl |
| **GPT-3** | 2020 | 400B | Common Crawl (filtered), WebText2, Myrstry books!! (Books1, Books2), Wikipedia |
| **Gopher** | 2021 | 1.4T | MassiveText |
| **BLOOM** | 2022 | 350B | ROOTS corpus, a dataset comprising hundreds of sources in 46 natural and 13 programming languages (59 in total) |
| **PaLM** | 2022 | 2.81T | Web documents, books, Wikipedia, conversations, GitHub code |
| **LaMDA** | 2022 | 1.56T | Public dialog data and web documents |
| **Chinchilla** | 2022 | 1.4T | MassiveText |
| **LLaMA2** | 2023 | 2.0T | A new mix of publicly available online data |
| **GPT-4** | 2023 | ? | ? |
| **Claude-3** | 2023 | ? | ? |
| **OLMo 2** | 2024 | 5.6T | OLMo-Mix-1124(stage1) + Dolmino-Mix-1124(stage 2) |
| **Qwen2.5** | 2024 | 7T | |
| **DeepSeek (V3)** | 2024 | 14.8T | GitHub's Markdown and StackExchange |
| **LLaMA3** | 2024 | 15T | A new mix of publicly available online data |

# The actual pre-training

How should we select the right hyperparameters?

# Q: What would you do?

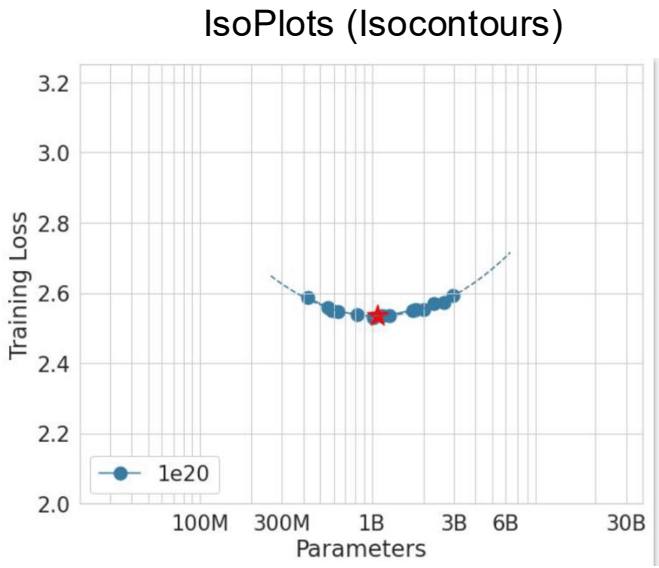
- Zuckerberg gave you a $500M budget for training Muse Spark 3.
- You set aside $10M for finding the best architecture at smaller scale, assuming that your ultimate model will be much larger.
- This way, you pick your parameters with rigorous experiments at small scale:
  - Optimal training params: Learning rate, warmup, weight decay, etc.
  - Architecture configs by scaling (each x50) the optimal values at small scale.

- Q: What you like (or don't) about this recipe?

- Optimal depth/width, lr, batch size, weight decay, init, and residual scaling are **not scale-invariant**.

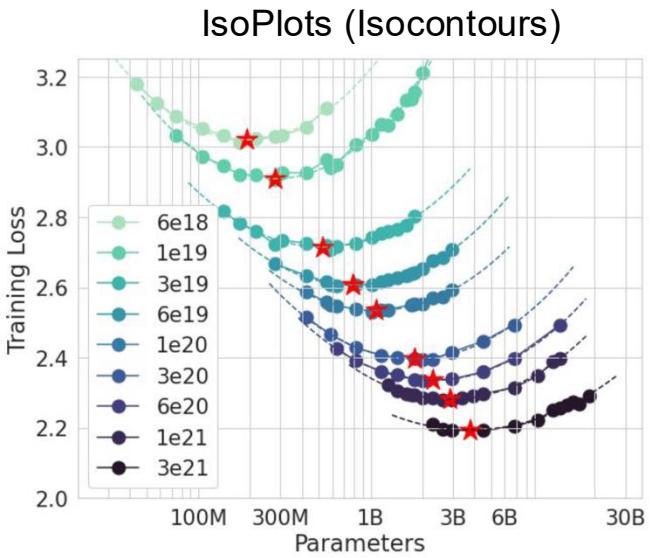
# IsoPlots: Tradeoffs at a smaller scale

- The performance of your model depends on a complex combination of many factors.
- Goal: find the best combinations, for a fixed compute.
- Approach:
  1. Fix a compute budget FLOP
  2. Train a few models and vary their size
  3. Fit a curve and find the minimum



# IsoPlots: Tradeoffs at a smaller scale

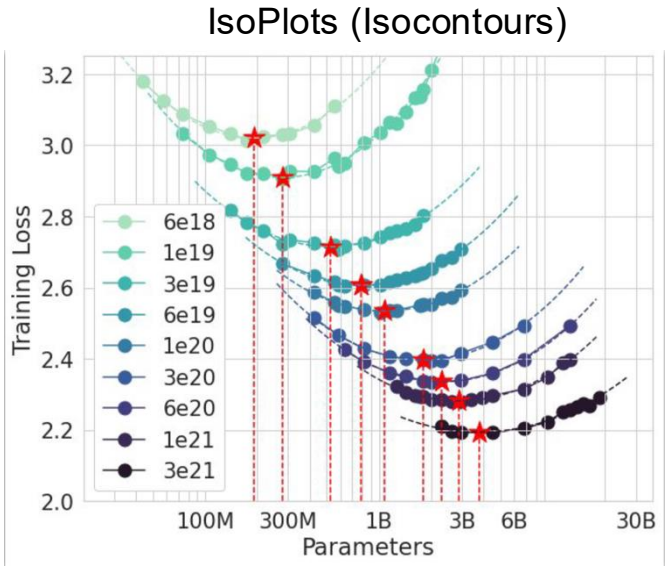
- The performance of your model depends on a complex combination of many factors.
- Goal: find the best combinations, for a fixed compute.
- Approach:
  1. Fix a compute budget FLOP
  2. Train a few models and vary their size
  3. Fit a curve and find the minimum
  4. Repeat 1-3 for various FLOP budgets



# IsoPlots: Tradeoffs at a smaller scale

- The performance of your model depends on a complex combination of many factors.
- Goal: find the best combinations, for a fixed compute.
- Approach:
  1. Fix a compute budget FLOP
  2. Train a few models and vary their size
  3. Fit a parabola and find the minimum
  4. Repeat 1-3 for various FLOP budgets

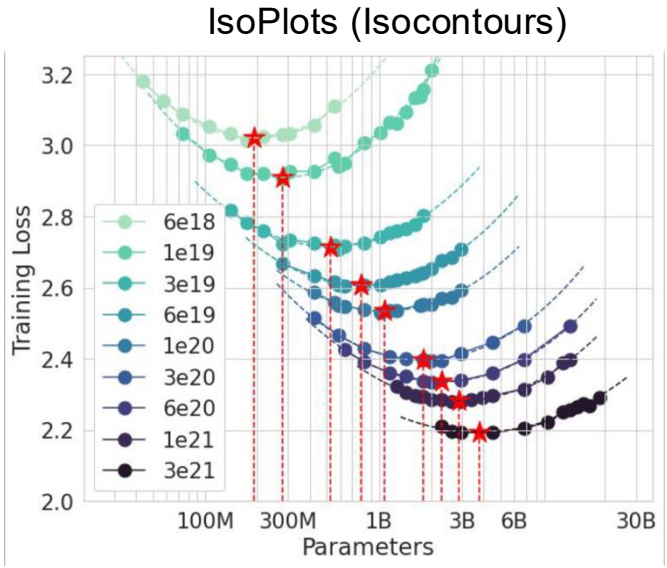
- It's good to change various parameter (e.g., training data, size, or other hyperparams) and see how it's quality (loss) changes.



# Predictive models for parameters

- Overall, IsoPlots show how loss depends on three axes: model size (parameters N), dataset size (tokens D), and compute budget C.
- They're a tool for reasoning about *how to spend your training budget efficiently*.
- But it also allows one to see track effective hyperparams (LR, batch size, etc.) changes with N, C, or D.

- **Lesson:**
  - Don't overtune your hyperparameters at small scales and expect to use them at large
  - Instead develop predictive metrics based on parameters:



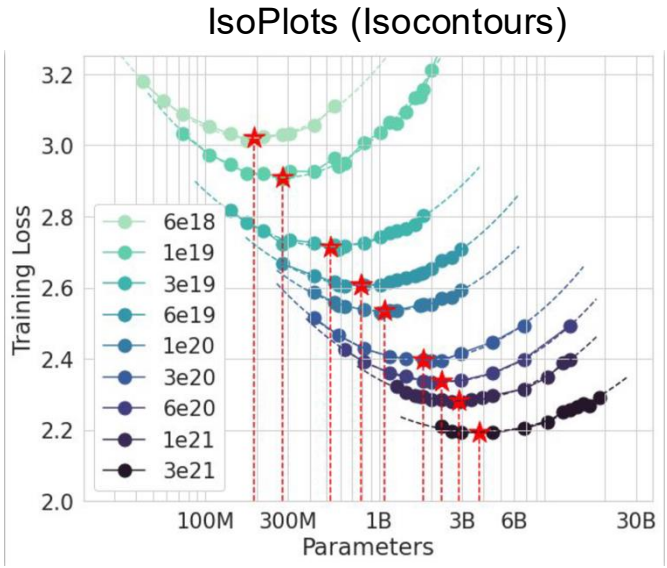
# Predictive models for parameters

- Overall, IsoPlots show how loss depends on three axes: model size (parameters N), dataset size (tokens D), and compute budget C.
- They're a tool for reasoning about *how to spend your training budget efficiently*.
- But it also allows one to see track effective hyperparams (LR, batch size, etc.) changes with N, C, or D.

- **Lesson:**
  - Don't overtune your hyperparameters at small scales and expect to use them at large
  - Instead develop predictive metrics based on parameters:

- Most of this work has typically been done at industry labs who have the compute to run these experiments.

- Recent open-source efforts from academia/non-profit:
  - OLMo series of models from Ai2.
  - Marin project from Stanford/Together Compute:
    - https://marin.community/



# Pretraining summary

- **IsoPlots:** for a fixed compute, which combination of parameters give you the best bang for the buck.
- For different compute budgets, plot IsoPlots and observe trends. Make predictions about larger budgets before actually training them.
- Very hard to do large scale training in academia but you can still find meaningful trends at smaller scale to predict performance at larger scale even with modest academic compute.

# Other pretraining related concepts, we will not cover in detail

- **In-context Learning:** Pretrained models can solve many NLP tasks using demonstrations in the input.
  - Input:  (x1, y1), (x2, y2), (x3
  - Expected model output: y3.

- **Chain-of-thought Prompting:** Demonstrations can include "step-by-step" reasoning (r) of how to solve the task.
  - Input: (x1, r1, y1), (x2, r2, y2), (x3
  - Expected model output: r3, followed by y3.
  - Why: shown to improve performance over just demonstrating with x and y.
  - We will study the modern version of this in detail where you post-train models to "reason".