A Brief Introduction to Large Language Models, ChatGPT, & GenAI

Joseph E. Gonzalez
jegonzal@berkeley.edu

slido

Please download and install the Slido app on all computers you use

How often do you use GenAI (e.g., Gemini, ChatGPT, Claude)

ⓘ Start presenting to display the poll results on this slide.

slido

Please download and install the Slido app on all computers you use

Which GenAI technologies have you used

ⓘ Start presenting to display the poll results on this slide.

What is Chat GPT?

Chat: natural language system

Explained Today

G: **Generatively** – Designed to model the creation of text

P: **Pretrained** – Trained on lots of naturally occurring data

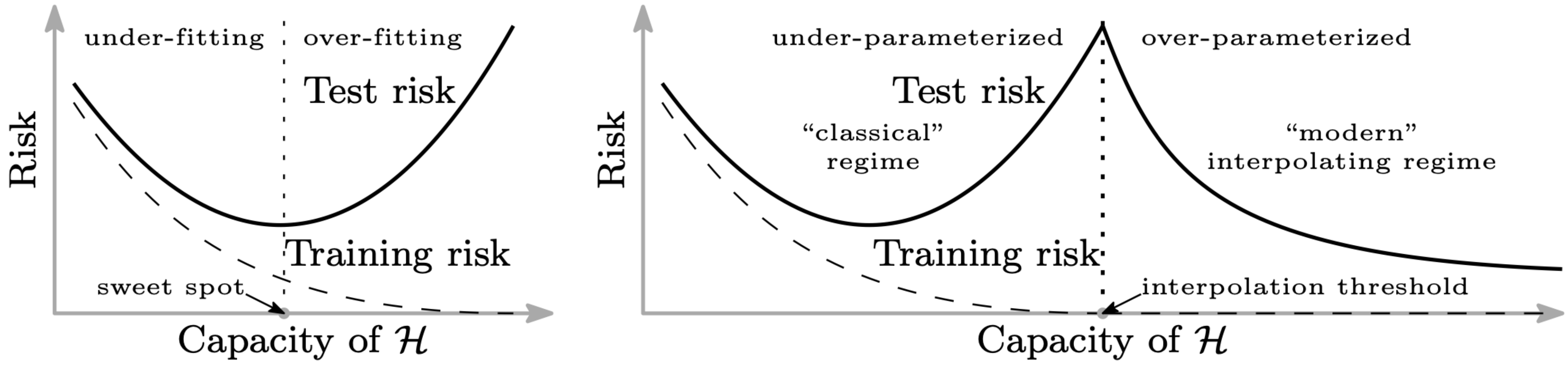
T: **Transformer** – A kind of neural network architecture

Chat GPT is just one example of a

Large Language Model (LLM)

What is a **Large Language Model (LLM)**?

- **Large:** The model parameters (θ) are ***BIG!***
  - ***BILLIONS of PARAMETERS!!!!***



(a) **(b)**

[Reconciling modern machine learning practice and the bias-variance trade-off]

[Deep Double Descent: Where Bigger Models and More Data Hurt]

What is a
Large Language Model (LLM)?

- **Large:** The model parameters (θ) are ***BIG!***
  - ***BILLIONS of PARAMETERS!!!!***
- **Language Model:** predicting language (e.g., words)

The capital of California is __**Sacramento**_____

San Francisco (1862)
Benicia (1853)
Vallejo (1852)

Tell a short story about a fairy princess named Alice.

Once

Predicting the Next Word is *Knowledge*

The capital of California is **Sacramento**

San Francisco (1862)
Benicia (1853)
Vallejo (1852)

Predicting the next word allows you to:

- Answer questions
- Tell stories
- Accomplish tasks

→ **Generative AI**

How do we *model* the next ~~word~~ token?

Modeling **Tokens** not **Words**

- **Tokens** represent **words**, **word parts,** and **special characters**

  “The smallest tokenizer!” →

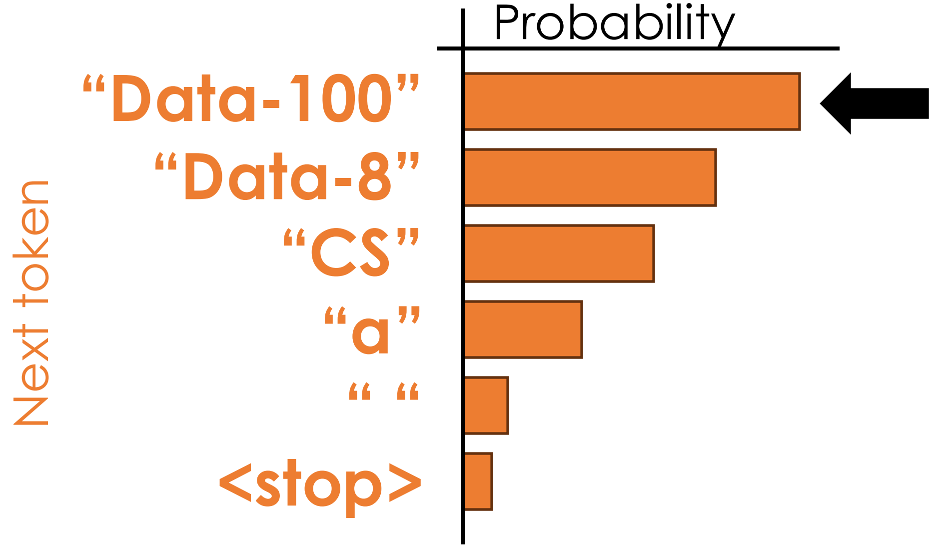
  Tokens: [“The”, “ small”, “est”, “ token”, “izer”, “!”]

- Constructed based on **frequency of char. sequences**
- Allows for **new words**, **misspelling**, and **numbers**
- **Vocabulary Sizes:** Llama-2: 32K → Llama-3: 128K tokens

Causal Language Modeling

The best class at UC Berkeley is ___________

Model: $\Pr(\underbrace{\text{“Data-100”}}_{\textbf{Next Token}} \mid \underbrace{\text{“The best class at UC Berkeley is”}}_{\textbf{Context (ordered tokens)}}])$



- ➢ Conditioned on the **context**
- ➢ Model probability of **next token**
  - ➢ **Sample** or **pick most likely**

Causal Language Modeling

The best class at UC Berkeley is ___________

$$\textbf{Model:}\quad \Pr(\underbrace{\text{“Data-100”}}_{\textbf{Next Token}} \mid \underbrace{\text{“The best class at UC Berkeley is”}}_{\textbf{Context (ordered tokens)}}])$$

- Conditioned on the **context**
- Model probability of **next token**
  - **Sample** or **pick most likely**

How do we go from **predicting a single token** to **writing an essay?**

One token at a time!

Auto Regressive Decoding

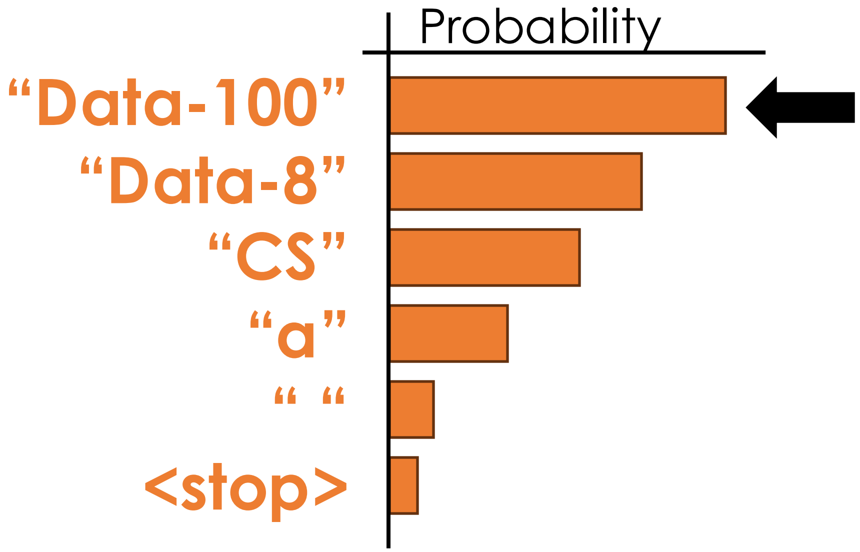
1. **Compute the probability** over the next token
2. **Select** the next token
   1. **Most likely** next token (temperature 0)
   2. **Sample** over the top few most likely tokens
3. **Append the selected token** to the context
4. **Repeat until** the **<stop>** token is reached.

Auto-regressive Decoding

➢ Sample **one token at a time** and add to context

Model: Pr("Data-100" | "The best class at UC Berkeley is"])

Next Token **Context (ordered tokens)**



Decode one word:

The best class at UC Berkeley is **Data-100**

Auto-regressive Decoding

- Sample **one token at a time** and **add to context**

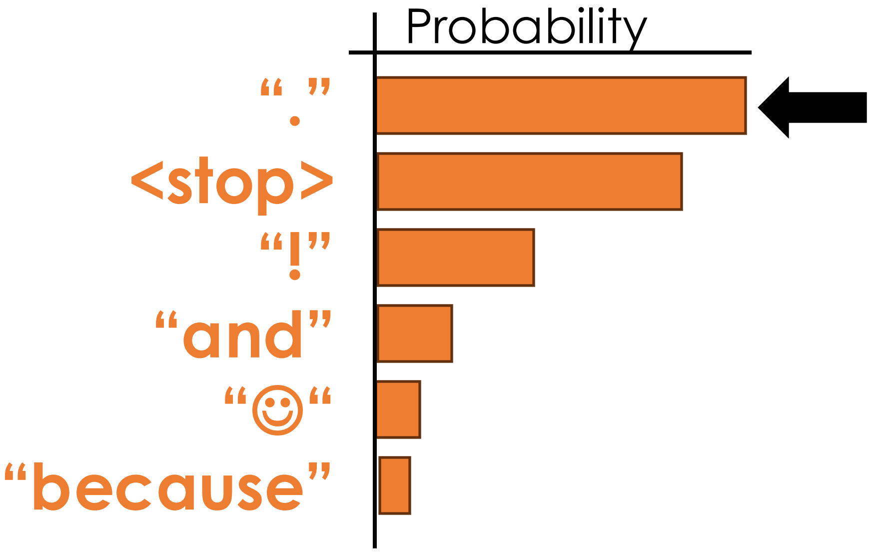
Model: **Pr(** $\underbrace{\quad\quad}_{\text{Next Token}}$ **|** “$\underbrace{\text{The best class at UC Berkeley is } \textbf{Data-100}}_{\text{Context (ordered tokens)}}$”])

Auto-regressive Decoding

- Sample **one token at a time** and **add to context**

Model: **Pr(** | "The best class at UC Berkeley is **Data-100**"])

Next Token **Context (ordered tokens)**



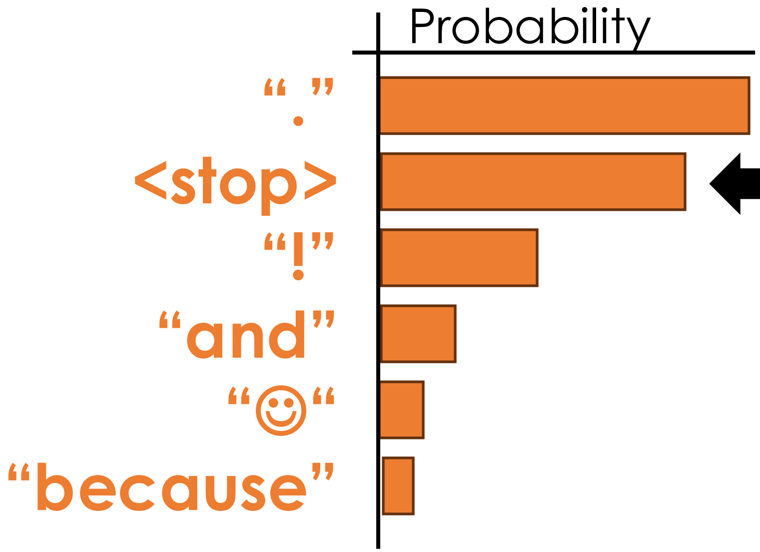
Decode one word:

The best class at UC Berkeley is Data-100.

Auto-regressive Decoding

➢ Sample **one token at a time** and **add to context**

Model: **Pr(** $\underbrace{\quad}_{\text{Next Token}}$ | "$\underbrace{\text{The best class at UC Berkeley is } \textbf{Data-100}}_{\text{Context (ordered tokens)}}$"])



Sample according to probabilities

Decode one word:

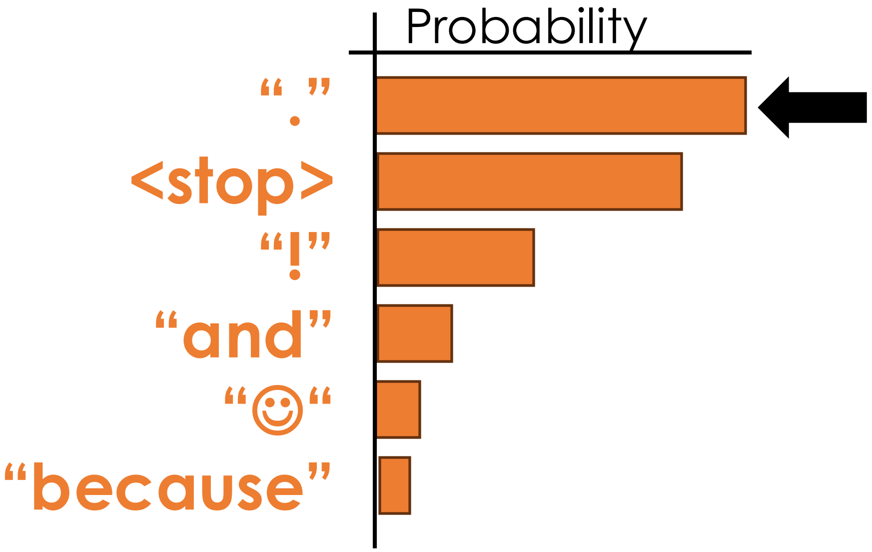
The best class at UC Berkeley is Data-100<stop>

Auto-regressive Decoding

- Sample **one token at a time** and add to context

Model: Pr("." | "The best class at UC Berkeley is **Data-100**"])

Next Token **Context (ordered tokens)**



Decode one word:

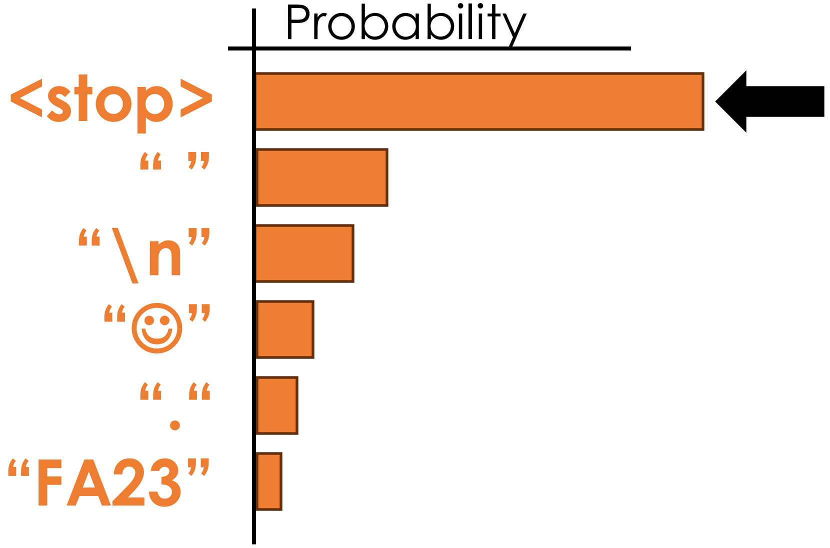
The best class at UC Berkeley is Data-100.

Auto-regressive Decoding

➢ Sample **one token at a time** and add to context

Model: Pr(<stop> | "The best class at UC Berkeley is Data-100."])

Next Token **Context (ordered tokens)**



Decode one word:

The best class at UC Berkeley is Data-100.<stop>

Stop decoding!

Quick Recap

- **Causal language modeling** predict next token given context

$$\underline{\textbf{Model:}}\quad \Pr(\underbrace{\text{"Data-100"}}_{\textbf{Next Token}} \mid \underbrace{\text{"The best class at UC Berkeley is"}}_{\textbf{Context (ordered tokens)}}])$$

- **Auto-regressive** (iterative) decoding:

Call model many times.
Slow to compute!

The best class at UC Berkeley **is**
The best class at UC Berkeley is **Data-100**
The best class at UC Berkeley is Data-100**!**
The best class at UC Berkeley is Data-100!**<stop>**

slido

Please download and install the Slido app on all computers you use

Can we predict the next several tokens in parallel (at the same time)?

ⓘ Start presenting to display the poll results on this slide.

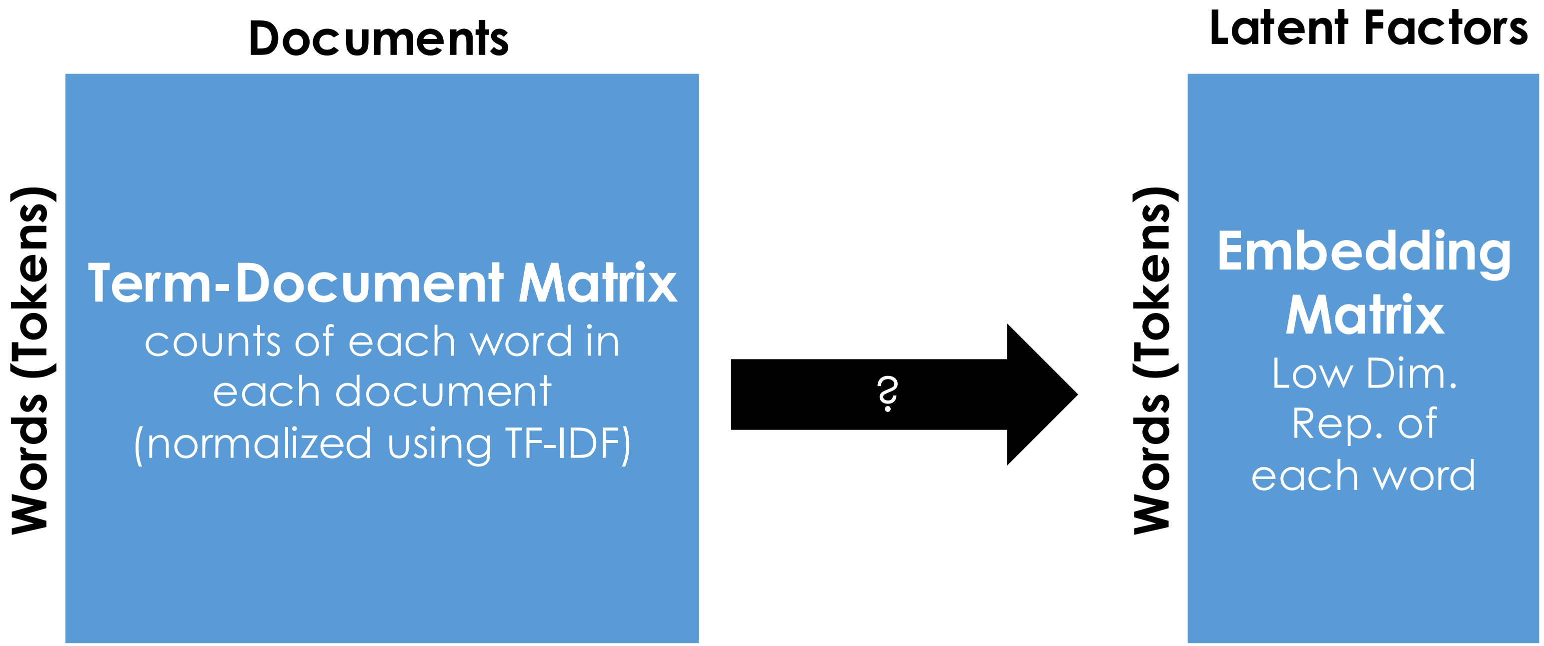
Building a basic Language Model

Model: $\Pr(\underbrace{\text{"Data-100"}}_{\text{Next Token}} \mid \underbrace{\text{"The best class at UC Berkeley is"}}_{\text{Context (ordered tokens)}}])$

- How do we implement this model?
- Solution builds on simple classification ideas

*I am going to give a **high-level intuition** and "explain" some of the basic parts.*

Language in High Dimensions



How would we go from a **high-dimension** term **frequency matrix** to a **low-dimensional term embedding matrix**?

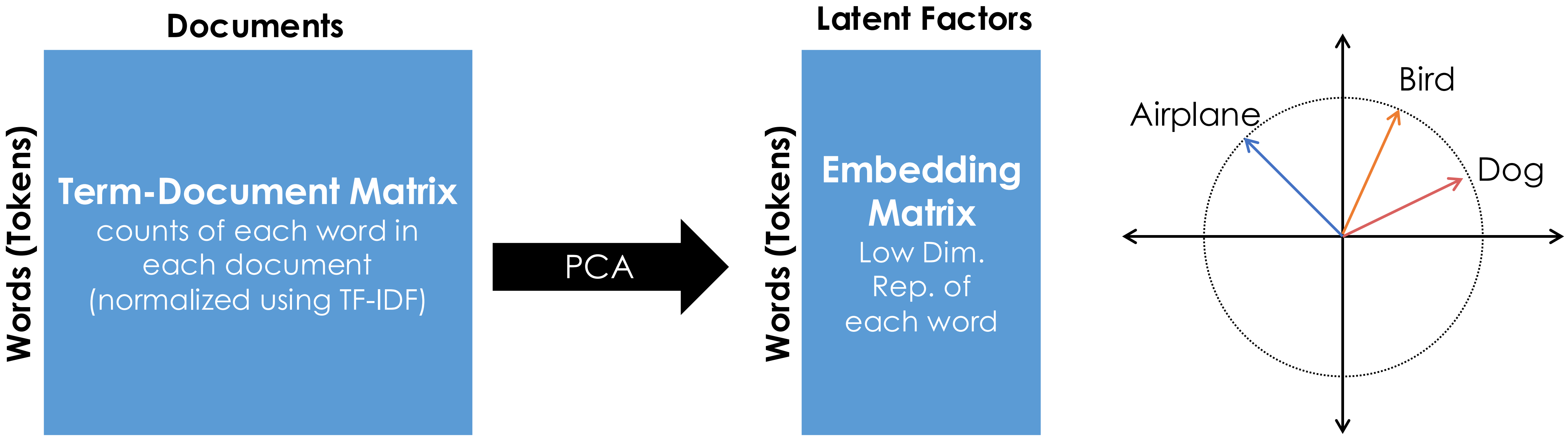
slido

Please download and install the Slido app on all computers you use

How would we go from a high-dimension term frequency matrix to a low-dimensional term embedding matrix?

ⓘ Start presenting to display the poll results on this slide.

Language in High Dimensions



- Ideas here date back to [Latent Semantic Analysis](#) (1988)
  - **PCA (you already know how to do this)**
  - Earlier in [1950s Linguistics](#) using

Building a basic Language Model

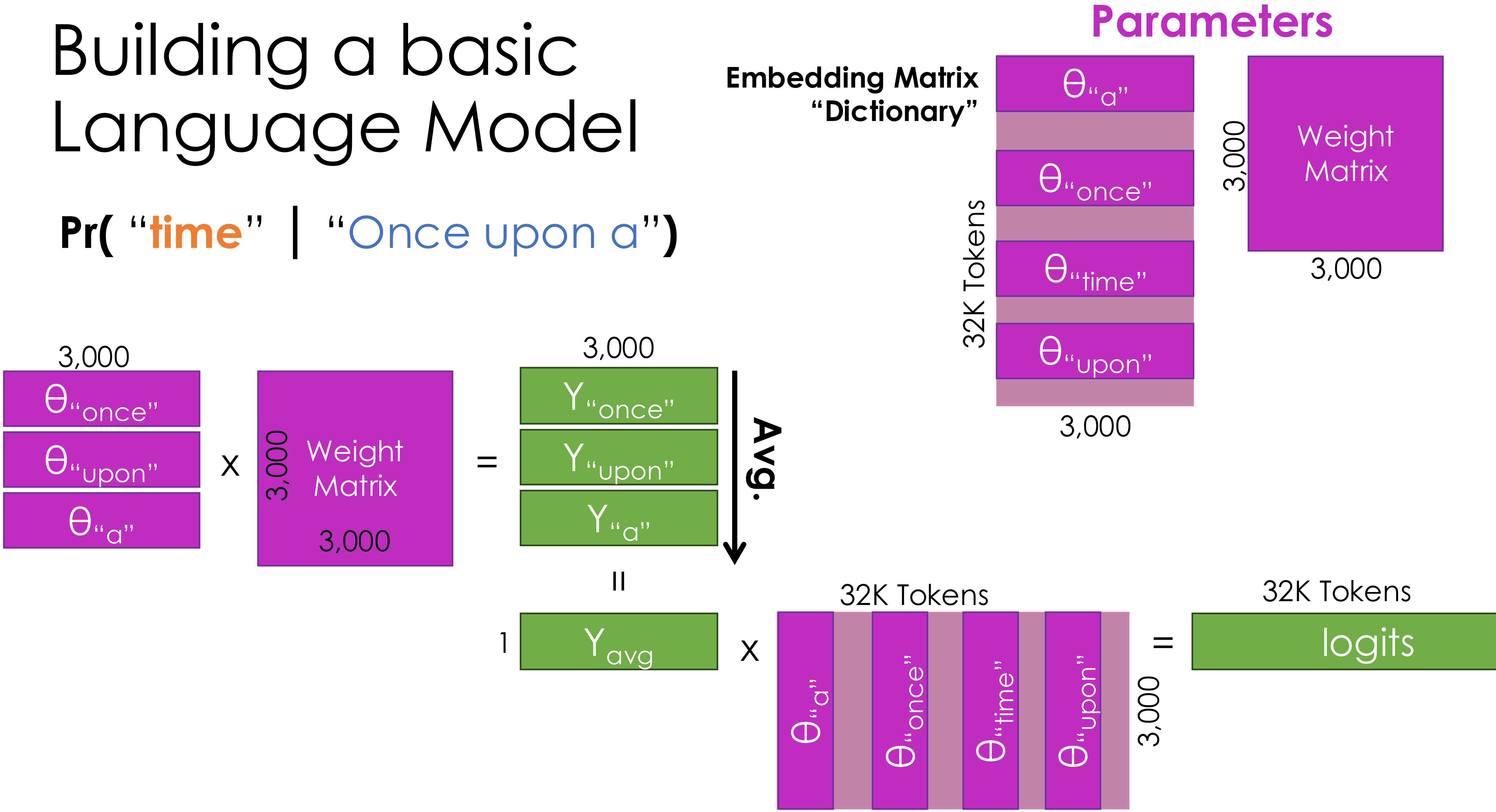
Pr("time" | "Once upon a")

Modeling Goal:

Predict the **next word** (*prob. of all possible next words*) **given the context** (*all the previous words*).

Building a basic Language Model

Pr("time" | "Once upon a")

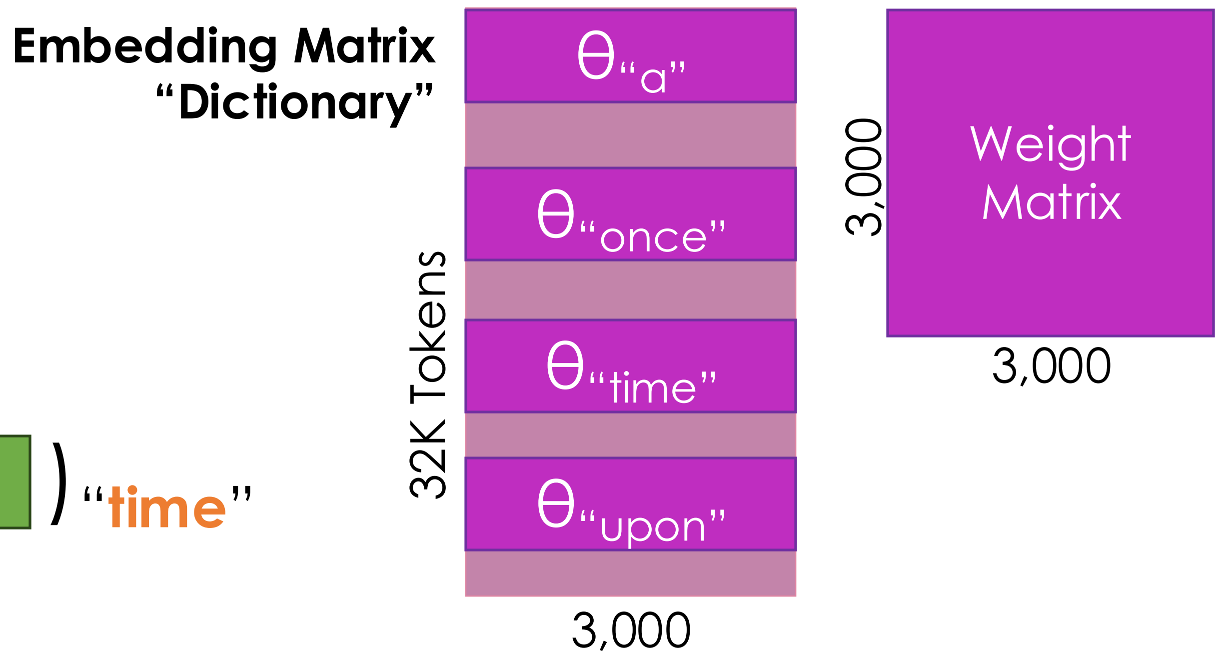


Building a basic Language Model

Pr("time" | "Once upon a")

= SoftMax(logits)$_{\text{"time"}}$

Parameters



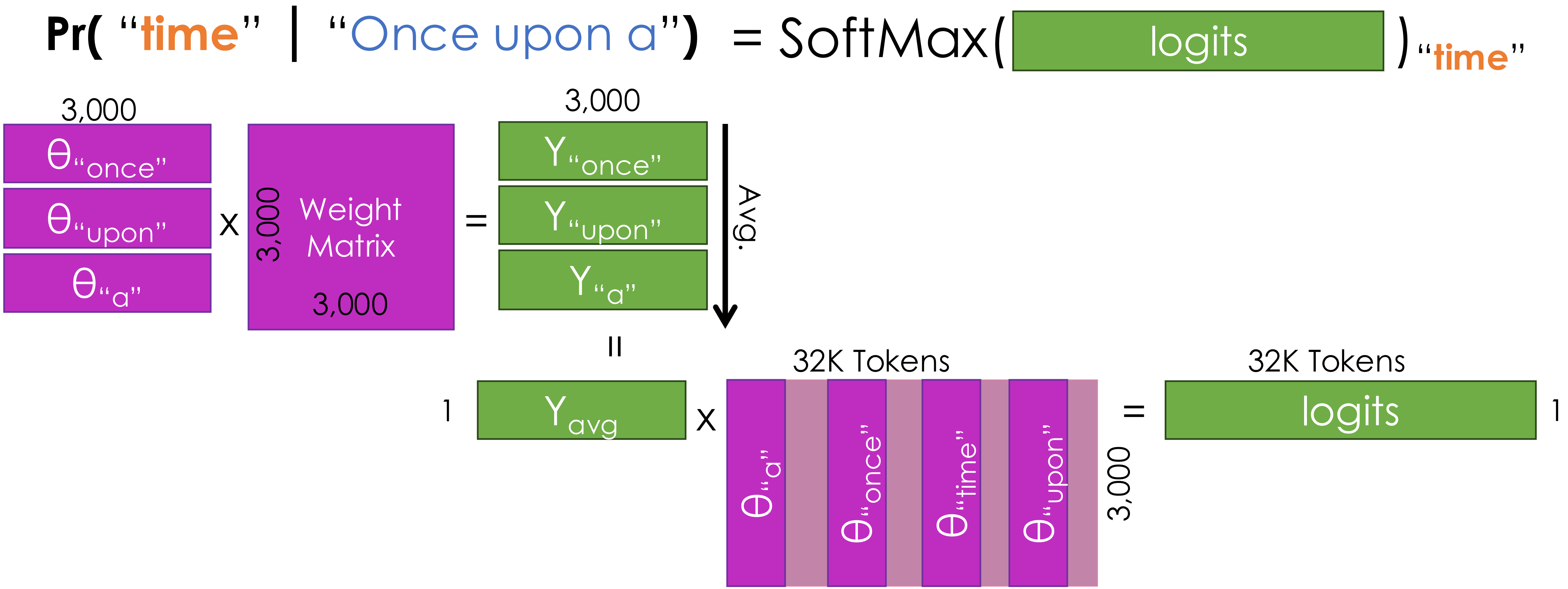
These are real numbers
Need a probability.

Converting real numbers to prob.:

$$\mathrm{SoftMax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$

Question:

- Does the order of the words in the context affect our prediction (the prob. of each next word)?



slido

Please download and install the Slido app on all computers you use

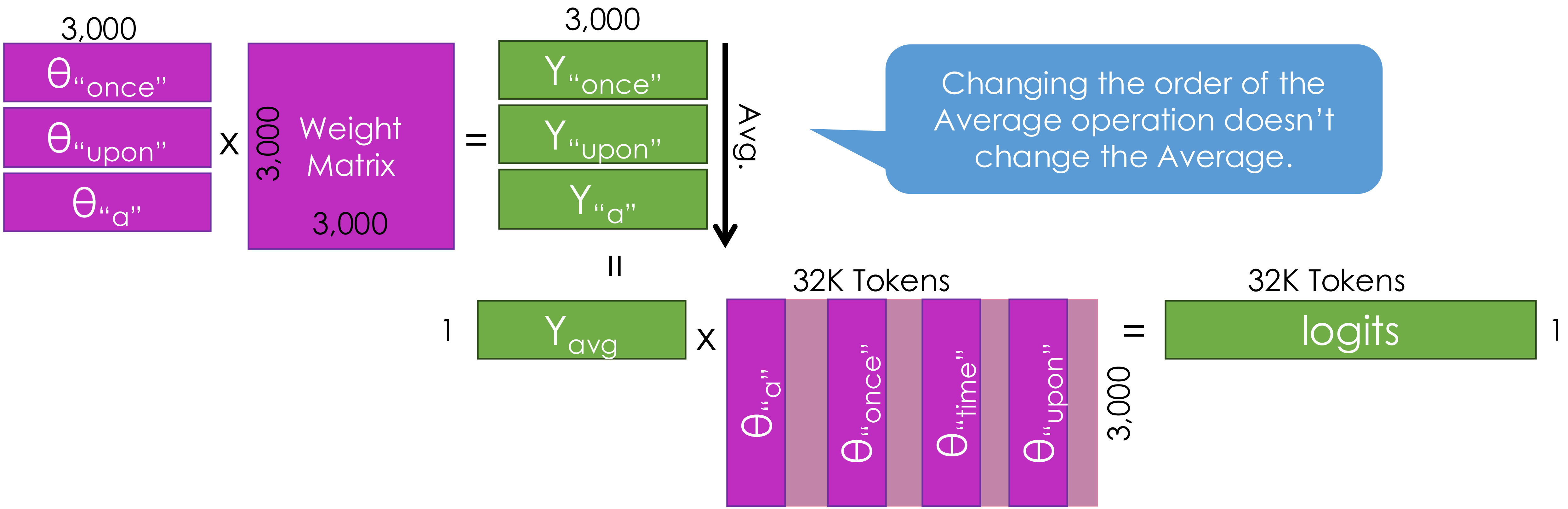
Does the order of the words in the context affect our prediction?

ⓘ Start presenting to display the poll results on this slide.

Question:

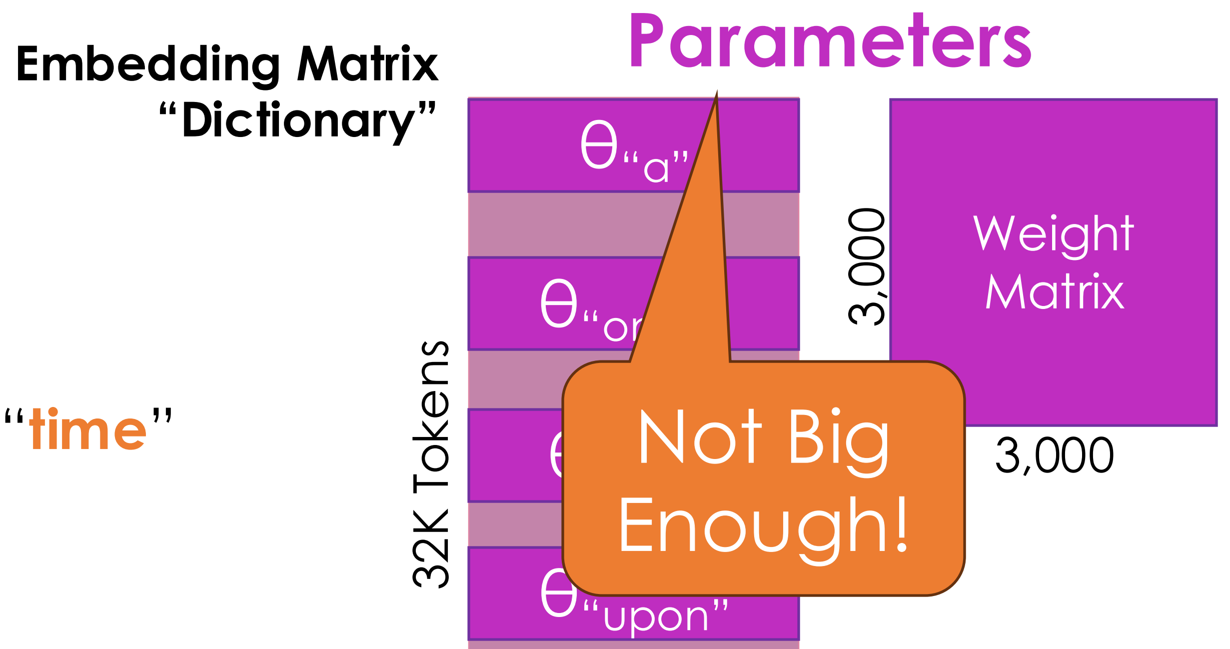
➢ Does the order of the words in the context affect our prediction (the prob. of each next word)?

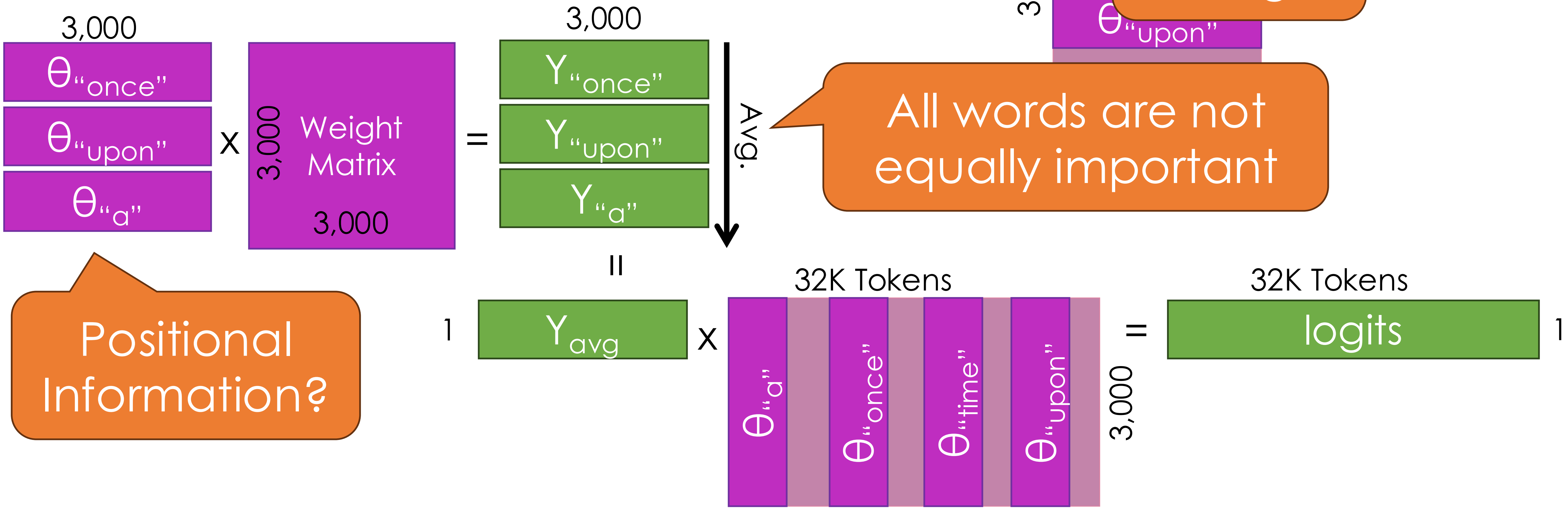
Pr("time" | "Once upon a") = SoftMax(logits)$_{\text{"time"}}$



Issues with our model

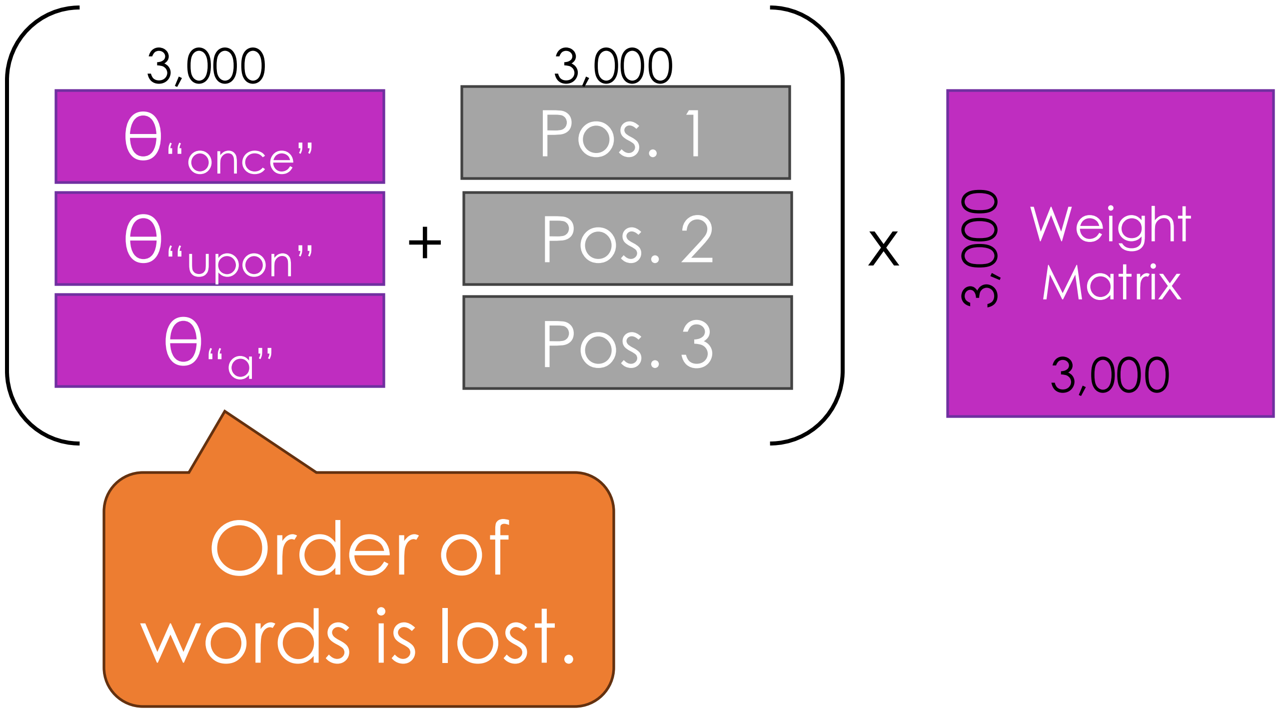
$$\Pr(\text{“time”} \mid \text{“Once upon a”}) = \mathrm{SoftMax}(\text{logits})_{\text{“time”}}$$





Encoding Positional Information

Pr(“time” **|** “Once upon a” **)**

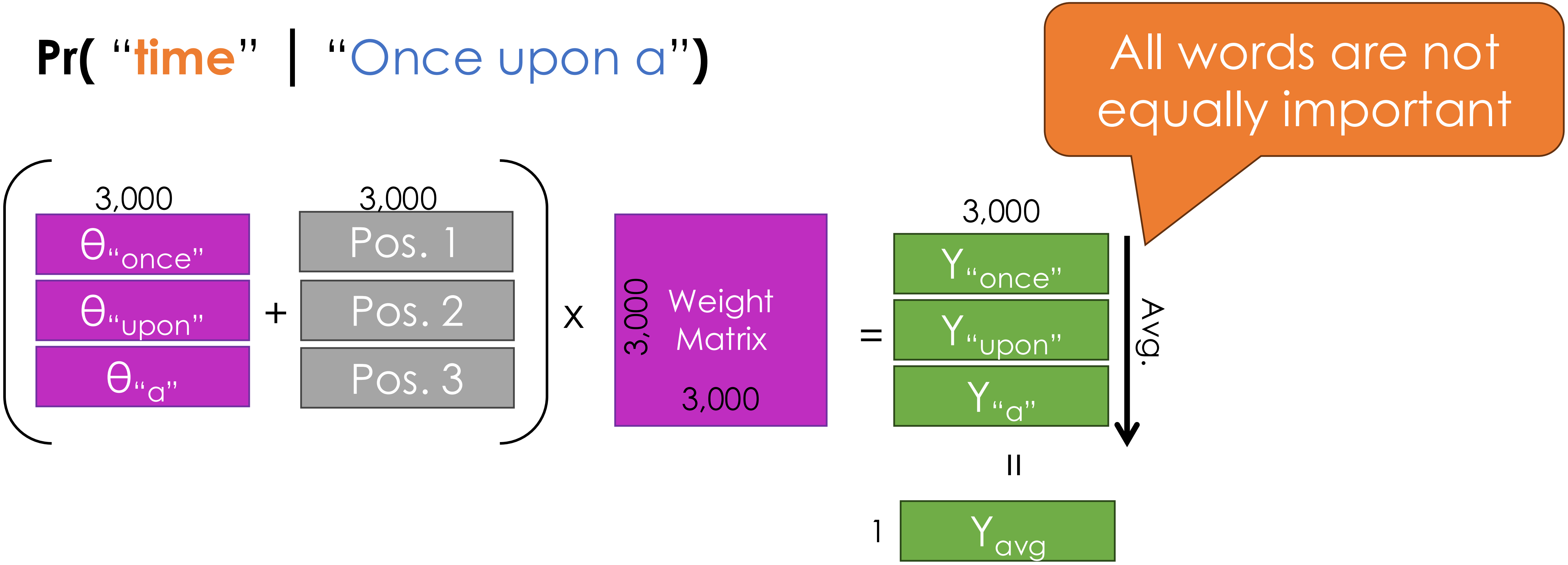


<u>Feature Engineering!</u>

- Add **positional encoding** to each token embedding
- Often **based on trig. fns.** so that nearby encodings are similar

Issues with our model

Pr("time" | "Once upon a")



What words in the context most predict the next word? (Where should we attend?)

Tell a short story about a fairy princess named Alice.

Once upon a time there was a fairy princess named ____

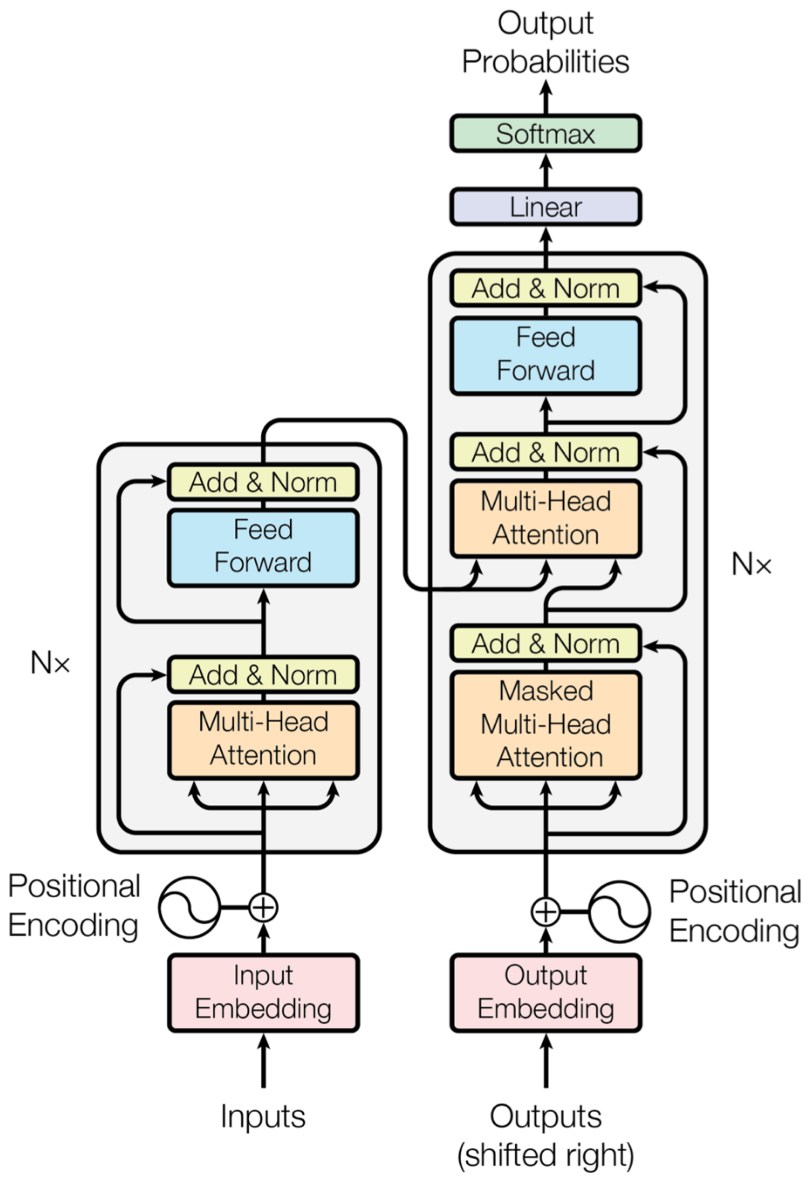
What words in the context most predict the next word? (Where should we attend?)

Tell a short story about a fairy princess named Alice.

Once upon a time there was a fairy princess named ______

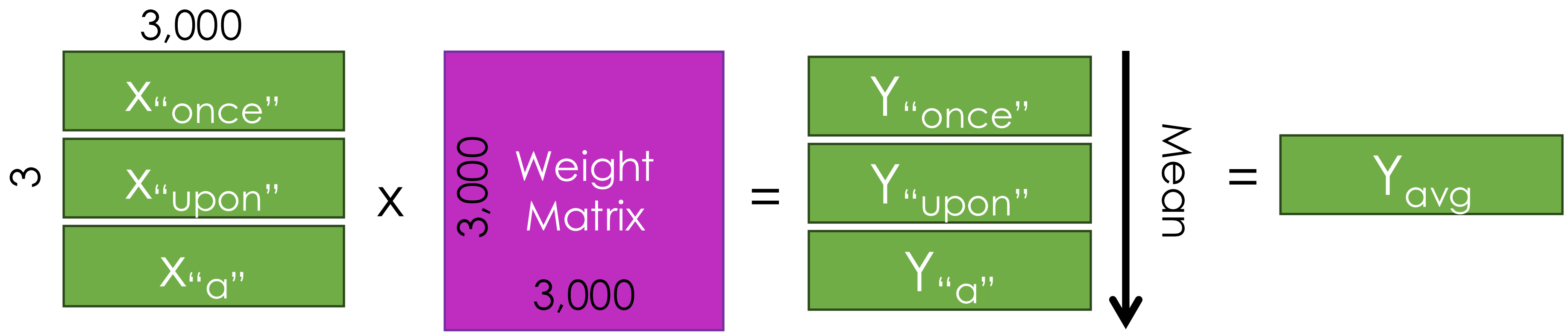
The Transformer Model

A somewhat simplified explanation of self-attention*



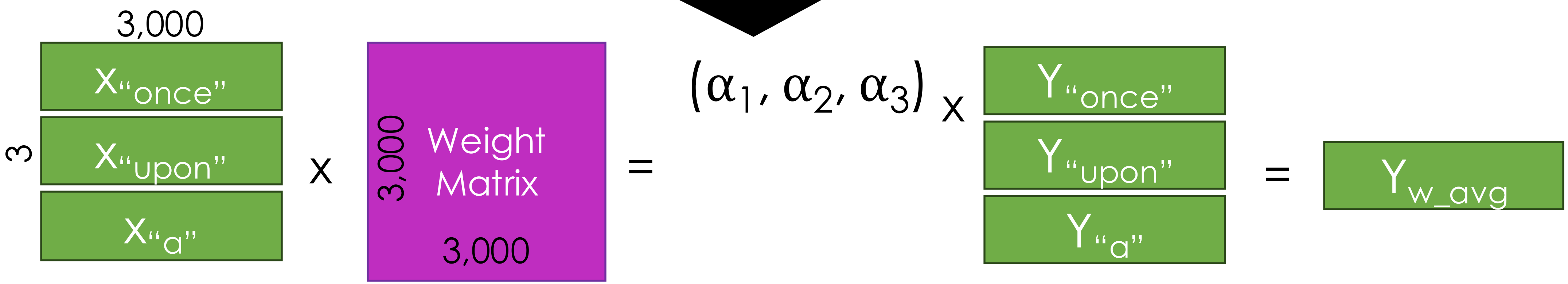
*Which is famously difficult to explain.

Computing a Weighted Average

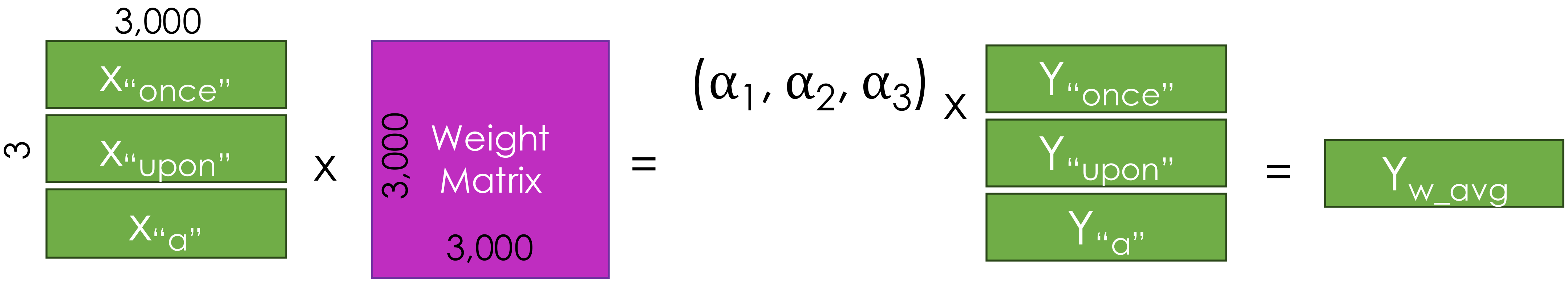


Computing a Weighted Average

Probabilities: should be between 0 and 1 and sum to 1

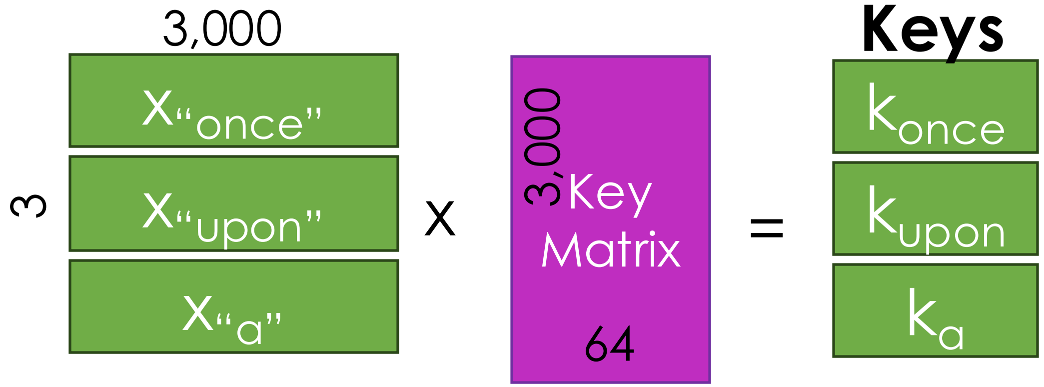


- How do we compute α?
  - Self Attention!

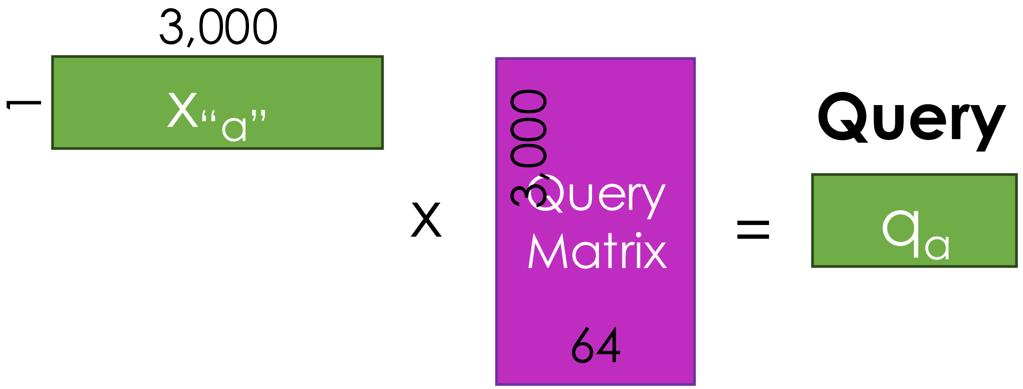


Computing Attention Weights ($\alpha_1, \alpha_2, \alpha_3$) – w.r.t. the last word

(1) Compute “Keys”

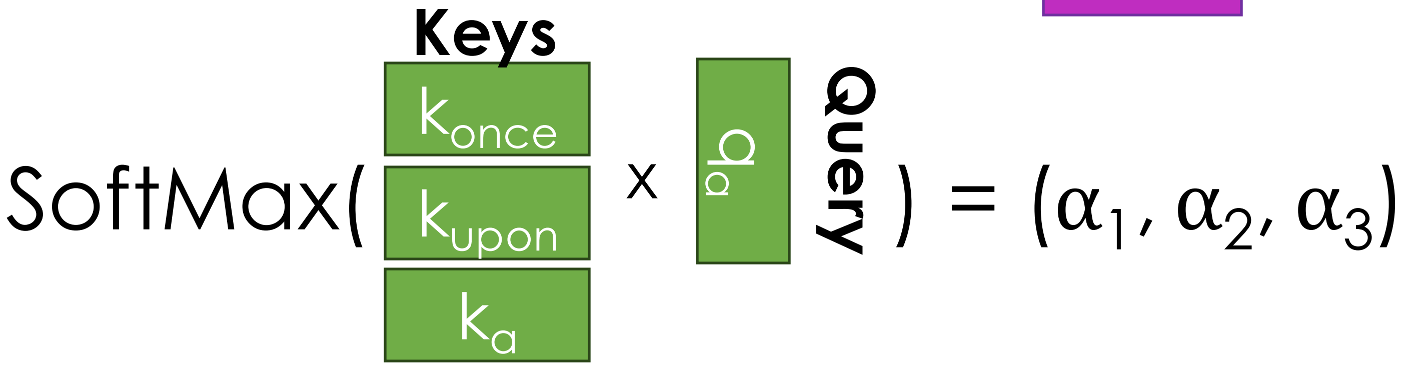


(2) Compute “Query” for the last word.



(3) Multiply keys by the query
And then take the soft-max

$$\mathrm{SoftMax}(x)_i = \frac{e^{x_i}}{\sum_{j=1}^{n} e^{x_j}}$$



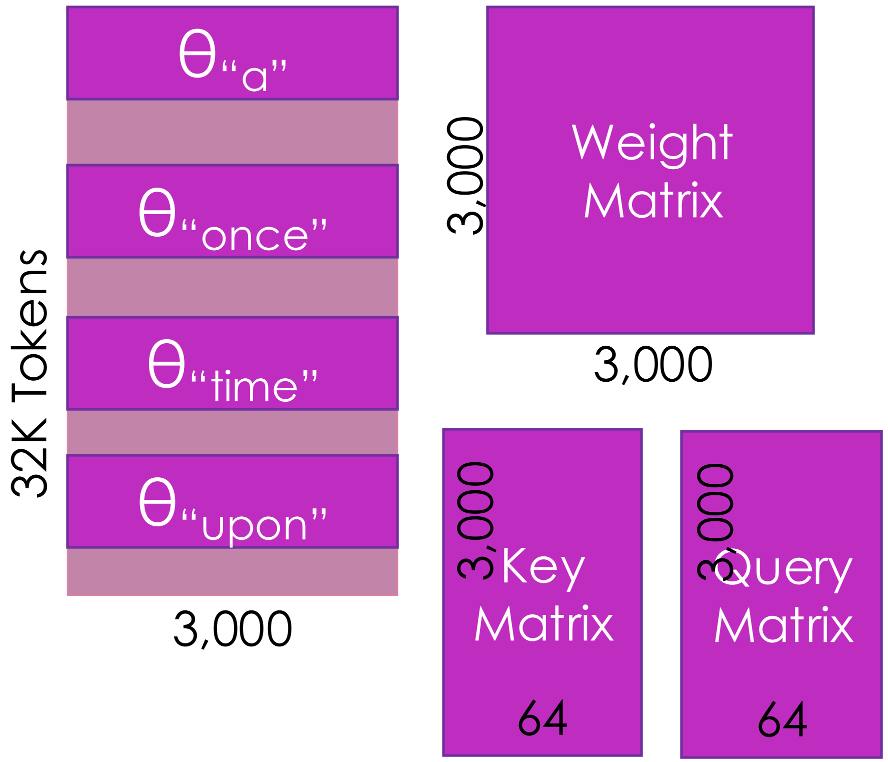
Transformer Recap

$$(\alpha_1, \alpha_2, \alpha_3) \times \begin{bmatrix} Y_{\text{“once”}} \\ Y_{\text{“upon”}} \\ Y_{\text{“a”}} \end{bmatrix}$$

- Computed a **weighted average** over the **“output” embeddings**
- Weights $(\alpha_1, \alpha_2, \alpha_3)$ were computed by
  - computing **keys** for each input token
  - computing **query** for the last token
  - taking the **soft max** of the product

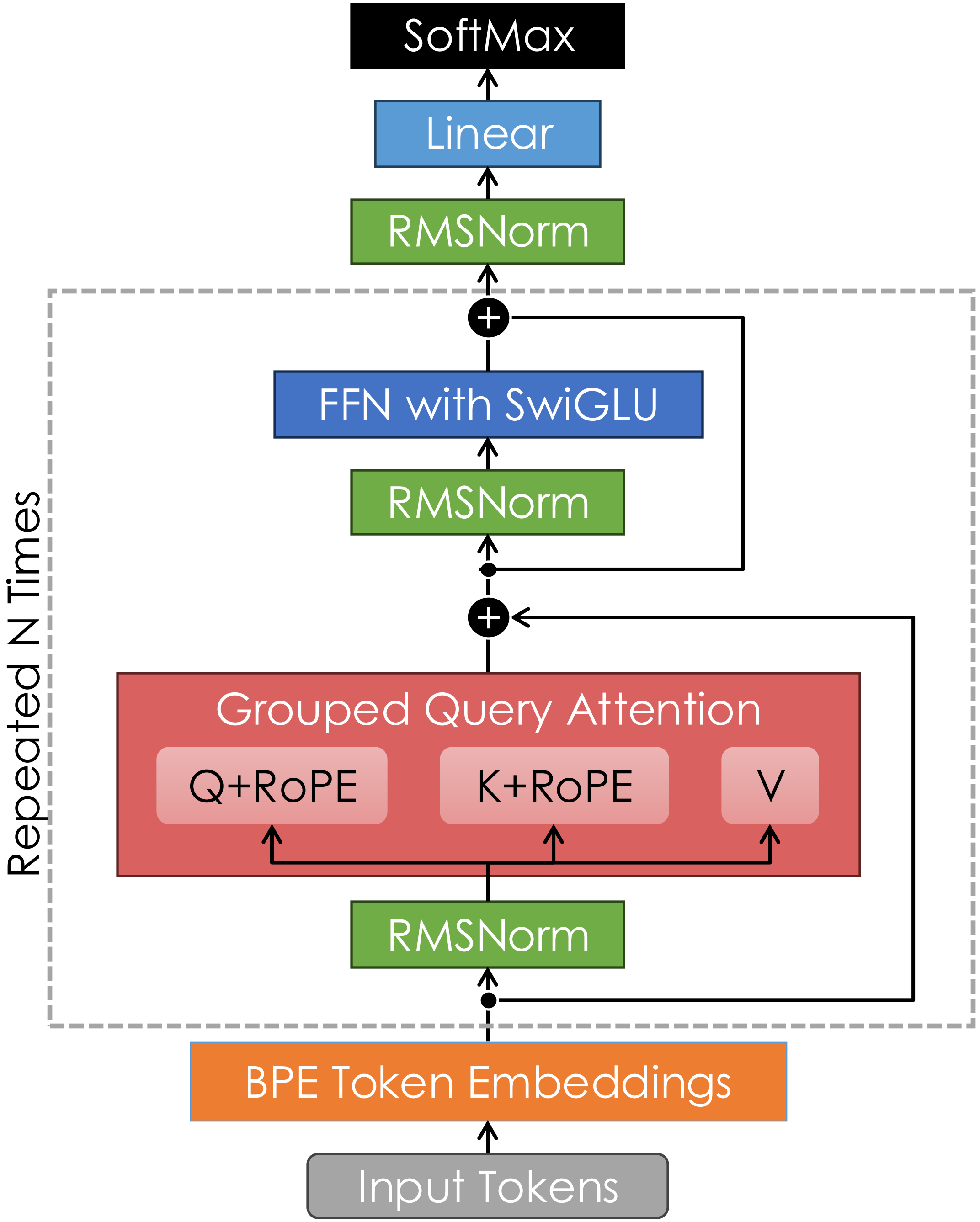
$$\text{SoftMax}\left(\overset{\textbf{Keys}}{\begin{bmatrix} k_{\text{once}} \\ k_{\text{upon}} \\ k_{\text{a}} \end{bmatrix}} \times \underset{\textbf{Query}}{q_{\text{a}}} \right) = (\alpha_1, \alpha_2, \alpha_3)$$

Many Parameters



Llama-3 Architecture

- **RMSNorm** (Layer Norm.) – improve training stability
- **FFN with SwiGLU** – Feed forward network
- **Residual Connections** – improve training stability
- **Repeated N Times** – increase model size



See [Actual Code](#) (its just one python script!)

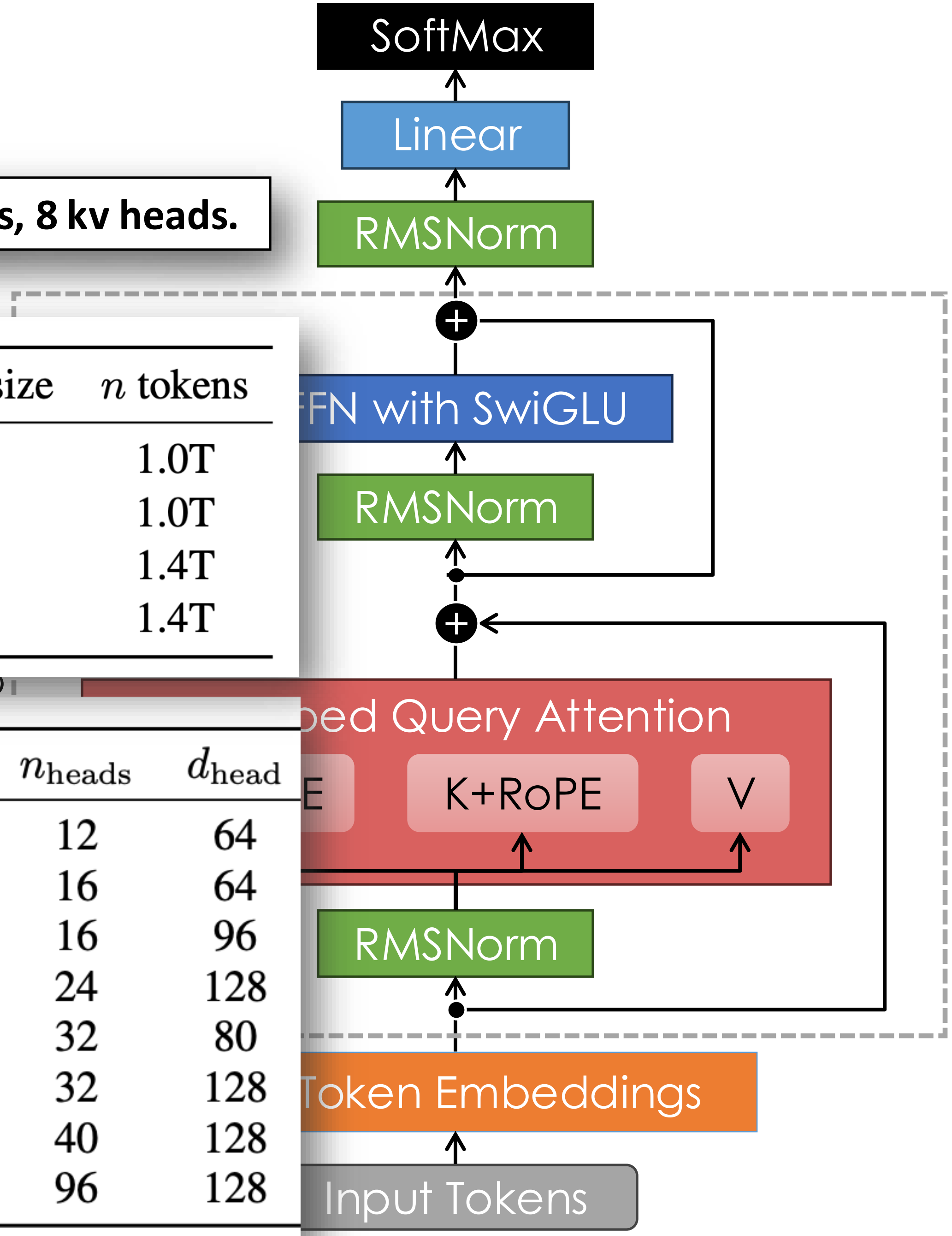
Going Big!

Llama-3 70b Instruct: 8192 hidden size, 80 layers, 64 query heads, 8 kv heads.

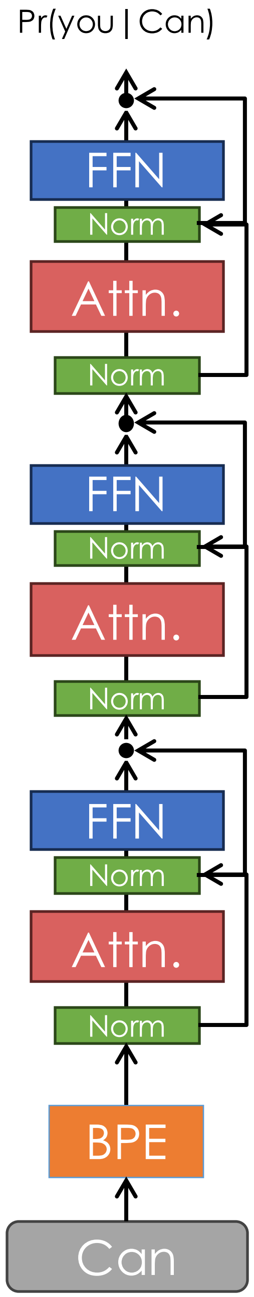
Llama 2

params	dimension	n heads	n layers	learning rate	batch size	n tokens
6.7B	4096	32	32	$3.0e^{-4}$	4M	1.0T
13.0B	5120	40	40	$3.0e^{-4}$	4M	1.0T
32.5B	6656	52	60	$1.5e^{-4}$	4M	1.4T
65.2B	8192	64	80	$1.5e^{-4}$	4M	1.4T

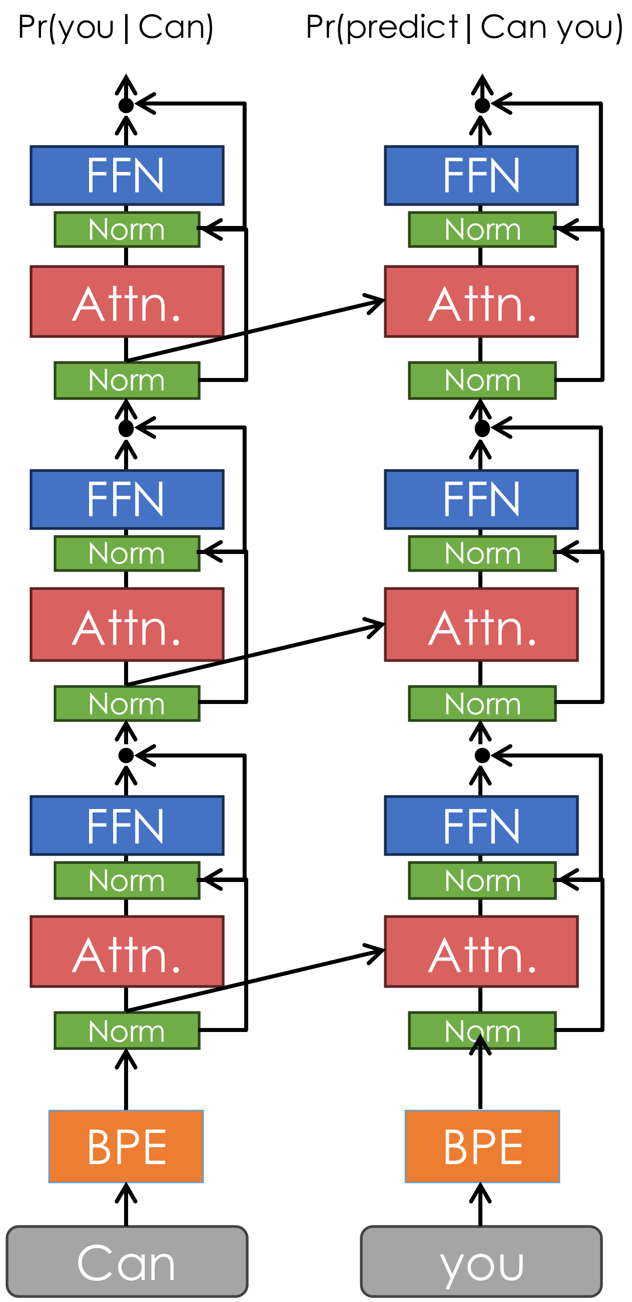
Model Name	n_{params}	n_{layers}	d_{model}	n_{heads}	d_{head}
GPT-3 Small	125M	12	768	12	64
GPT-3 Medium	350M	24	1024	16	64
GPT-3 Large	760M	24	1536	16	96
GPT-3 XL	1.3B	24	2048	24	128
GPT-3 2.7B	2.7B	32	2560	32	80
GPT-3 6.7B	6.7B	32	4096	32	128
GPT-3 13B	13.0B	40	5140	40	128
GPT-3 175B or "GPT-3"	175.0B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



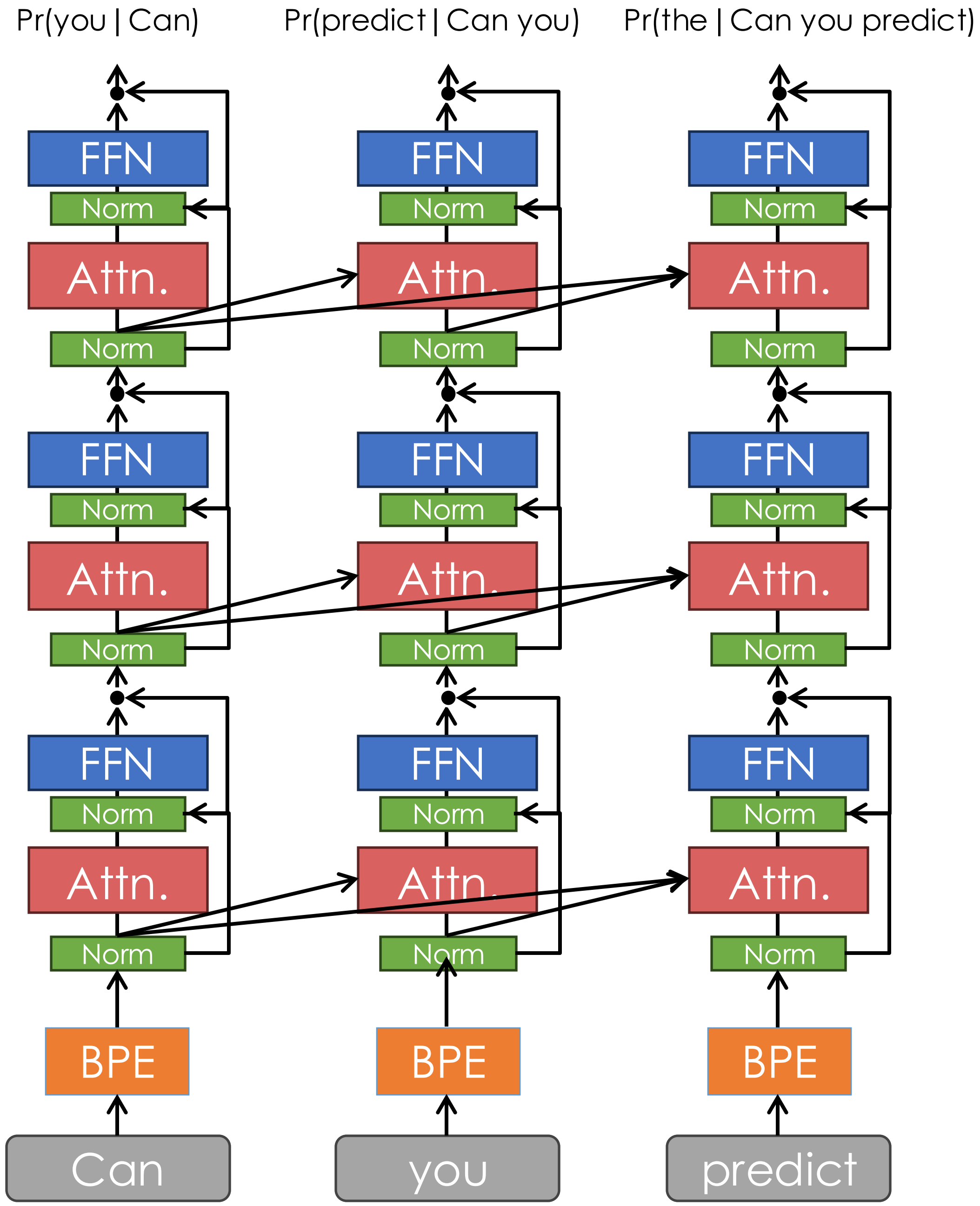
Unrolling the Model



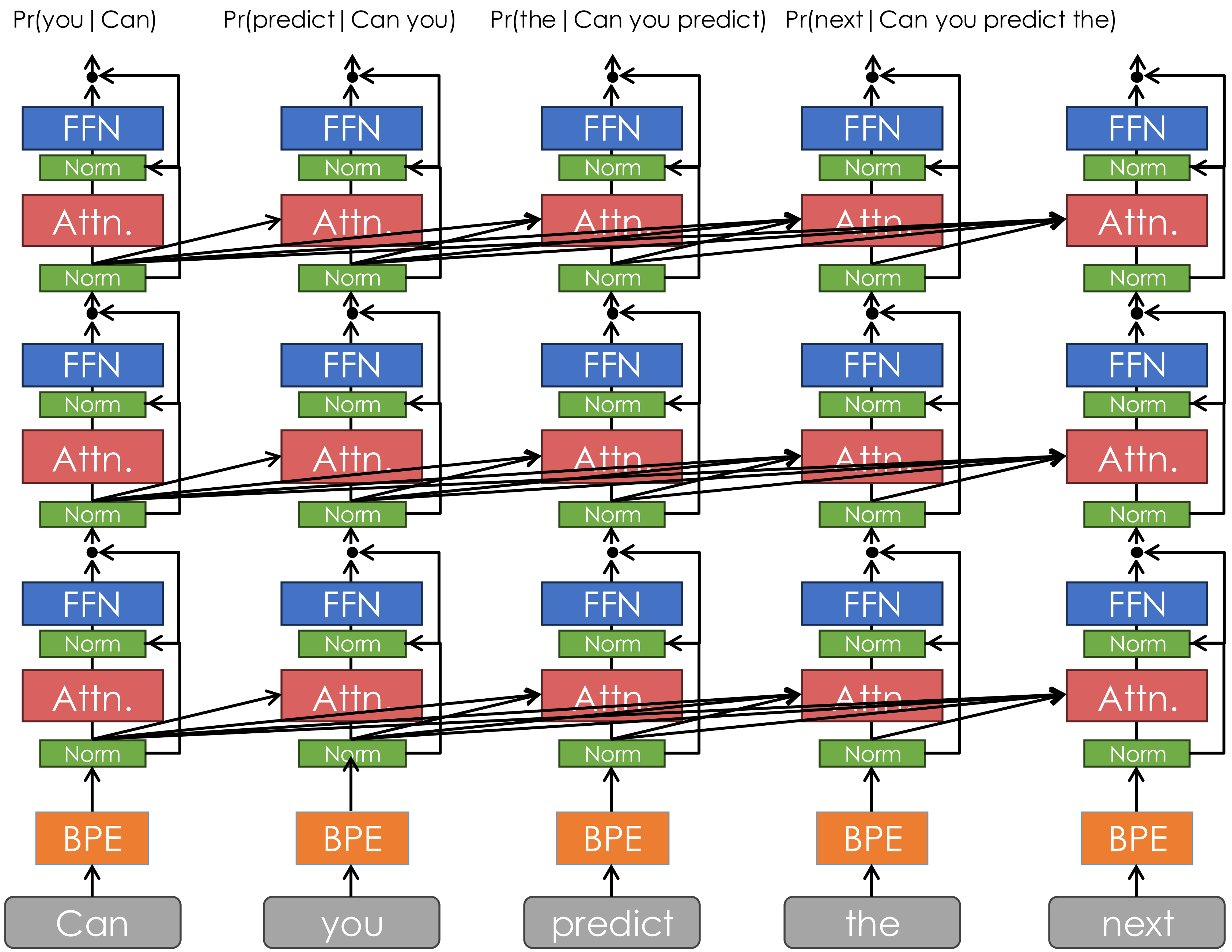
Unrolling the Model



Unrolling the Model



Unrolling the Model

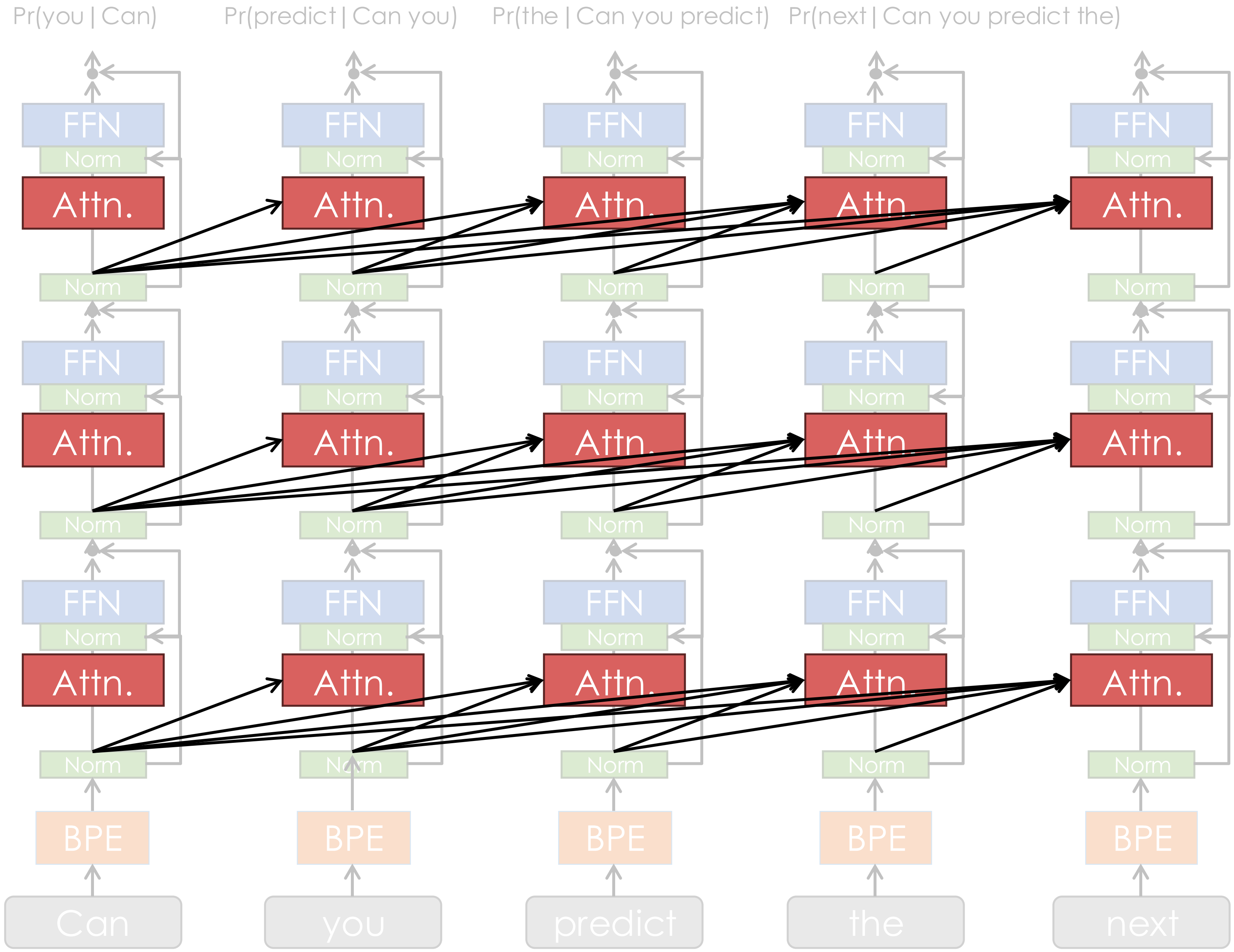


Masked Attention

Auto-regressive

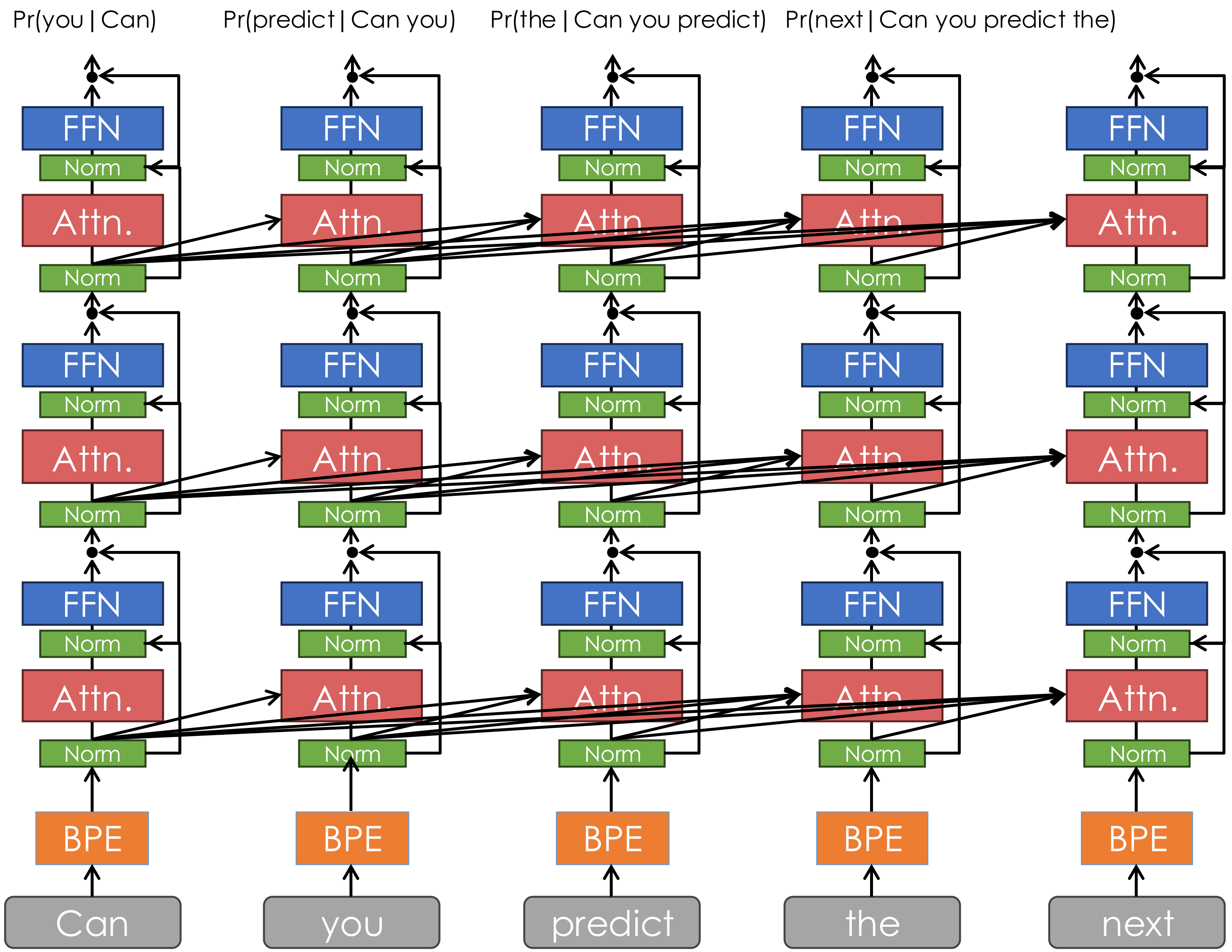
Attend only to previous tokens

Decoder only Transformer



Question

Which token requires the most computation?



slido

Please download and install the Slido app on all computers you use

Which token requires the most computation to predict?

ⓘ Start presenting to display the poll results on this slide.

What is Chat GPT?

Chat: natural language system

- ✅ **G**: **Generatively** – Designed to model the creation of text
- **P**: **Pretrained** – Trained on lots of naturally occurring data
- ✅ **T**: **Transformer** – A kind of neural network architecture

Chat GPT is just one example of a

✅ **Large Language Model (LLM)**

Generative Pre-training

Single Passage of Text

We the People of the United States, in Order to form a more perfect Union, establish Justice, …

Tune the model parameters to **maximize the likelihood** of the next token

Each token
is a training example

Pr(“We” | “”**)**

Pr(“the” | “We”**)**

Pr(“People” | “We the”**)**

Pr(“of” | “We the People”**)**

Pr(“the” | “We the People of”**)**

…

Pre-training on *Everything**

- Train the model on a **large collection** of data to learn **generalizable patterns**

Dataset	Quantity (tokens)	Weight in training mix	Epochs elapsed when training for 300B tokens
Common Crawl (filtered)	410 billion	60%	0.44
WebText2	19 billion	22%	2.9
Books1	12 billion	8%	1.9
Books2	55 billion	8%	0.43
Wikipedia	3 billion	3%	3.4

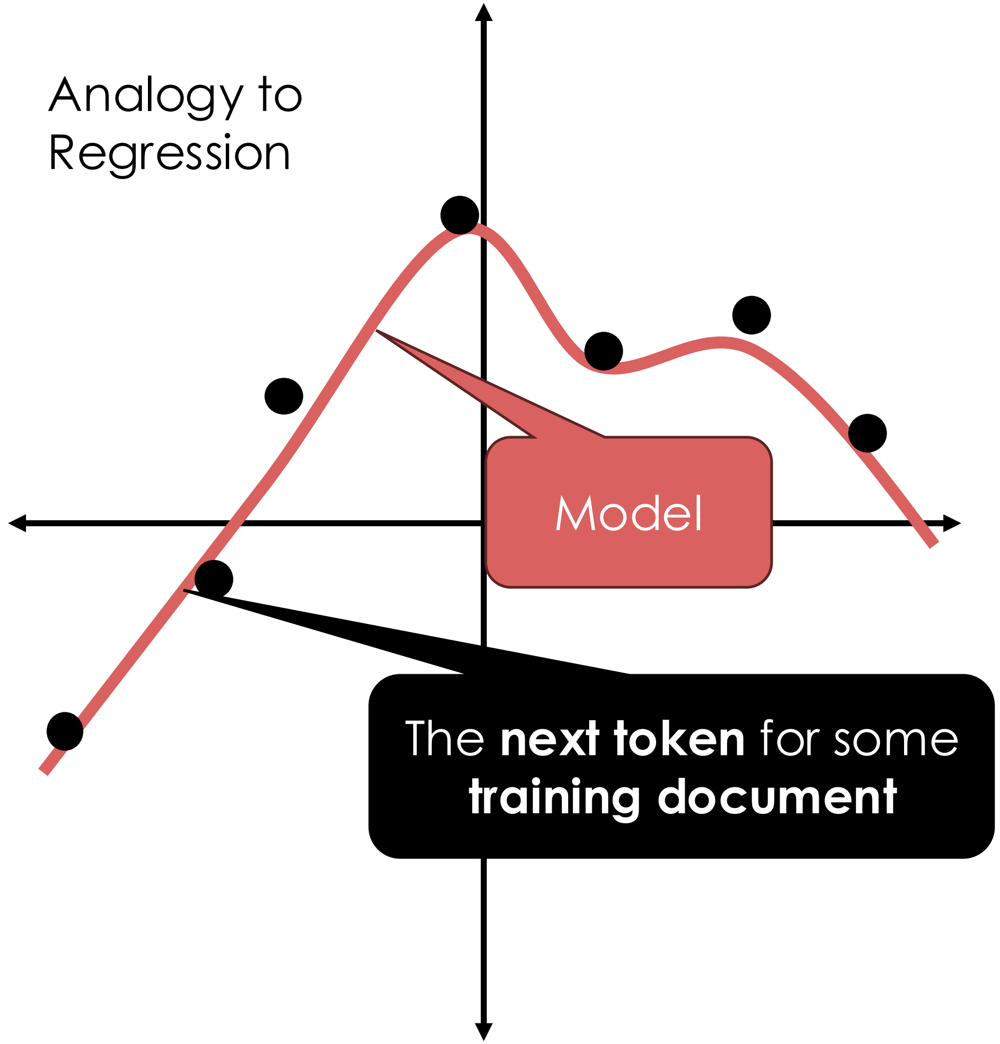
Reddit Links

OpenAI GPT3 Data Mix

- Llama-3 "open-source" models trained on **15.6T tokens from an unknown data mix.**
  - [405-billion parameter model]() trained on **16K H100s ($25K each)**
  - **39.3M GPU hours**

*Everything that is legal to use for training hopefully…

What have we learned?



- Model **approximates the data**
  - Doesn't fit perfectly.
- **Goal:** capture the **underlying structure** of all **language** (and therefore **human knowledge**)
- **Interpolation:** we can **generate** all the likely documents **between the documents**

Now you know

What is Chat GPT

There is still one more thing

Chat: natural language system

- ✅ **G**: **Generatively** – Designed to model the creation of text
- ✅ **P**: **Pretrained** – Trained on lots of naturally occurring data
- ✅ **T**: **Transformer** – A kind of neural network architecture

GPT alone can't chat!

Need to Teach Model to **Follow Instructions (and be fun!)**

- Generative pre-training **captures knowledge**
  - To finish the statement "Four score and seven years ago …" you need to learn to memorize the text
- Resulting model predicts the rest of a statement.
  - "What is attorney client privilege?" the model might generate "Provide a concise answer using an example from class."
- Use **Supervised Fine-Tuning** or **RLHF** to adjust "behavior" of model to **follow instructions** and **chat like a human.**

Fine-tuning

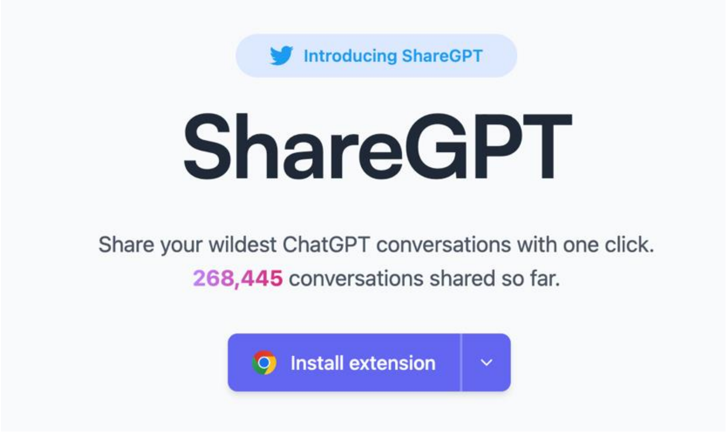
Running **additional training iterations** with a specific task

- The task differs from the original pre-training task
  - **New objective**: translate sentence, follow instruction
  - **New training data** from new source domain
- **Smaller learning rate** (as you get older you learn slower?)
  - Avoid "catastrophic forgetting" (new information causes forgetting pre-training information)

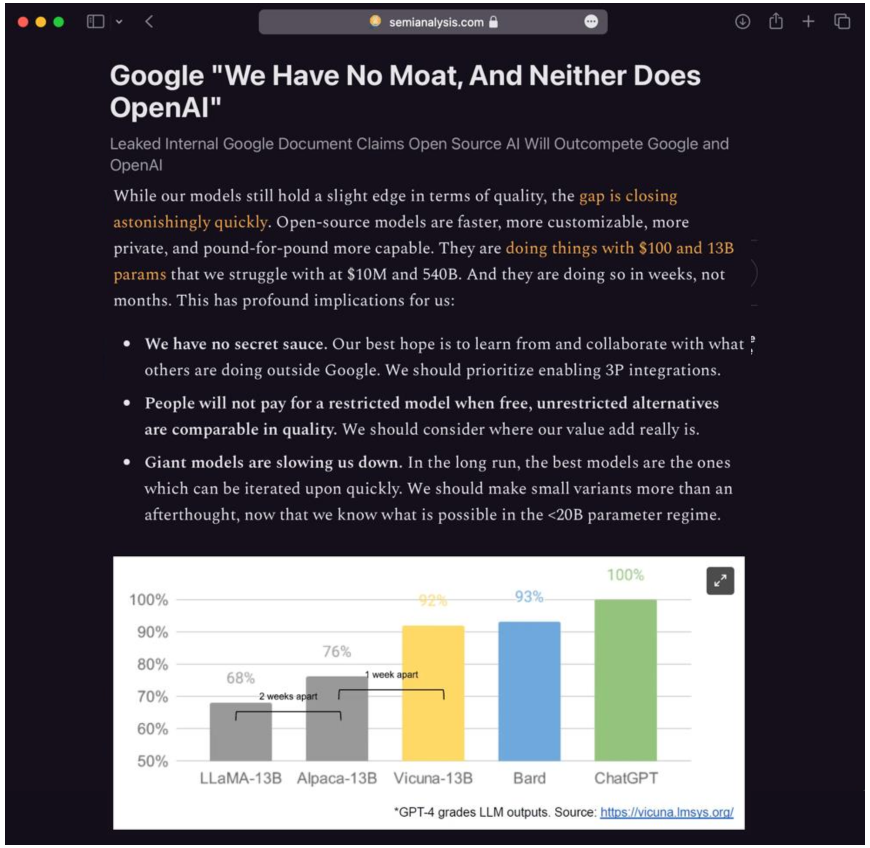
Vicuña

Open-source Instruction Fine-Tuned LLMs and LLM-as-a-judge

- ➢ First open-source model that was **“comparable” to ChatGPT**
  - ➢ Fine-tuned **LlaMA-13B** on **ShareGPT Data**



Tiny
High Quality Data
70K conversations
(~800MB)


semianalysis.com

Google "We Have No Moat, And Neither Does OpenAI"

Leaked Internal Google Document Claims Open Source AI Will Outcompete Google and OpenAI

While our models still hold a slight edge in terms of quality, the gap is closing astonishingly quickly. Open-source models are faster, more customizable, more private, and pound-for-pound more capable. They are doing things with $100 and 13B params that we struggle with at $10M and 540B. And they are doing so in weeks, not months. This has profound implications for us:

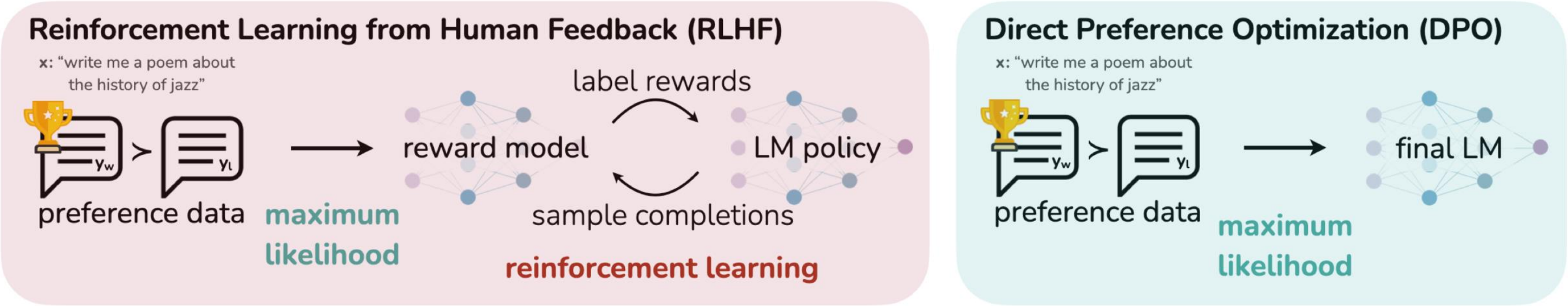
- **We have no secret sauce.** Our best hope is to learn from and collaborate with what others are doing outside Google. We should prioritize enabling 3P integrations.
- **People will not pay for a restricted model when free, unrestricted alternatives are comparable in quality.** We should consider where our value add really is.
- **Giant models are slowing us down.** In the long run, the best models are the ones which can be iterated upon quickly. We should make small variants more than an afterthought, now that we know what is possible in the <20B parameter regime.

*GPT-4 grades LLM outputs. Source: https://vicuna.lmsys.org/

- ➢ Helped launch academic open-source **GenAI research**

Reward Optimization

Can further **align model with human preferences** using **human preference data**: *"this is better than that"*



Help to **make models more robust** and perform better in **safety situations**.

Now you know

What is Chat GPT

- ✅ **Chat**: natural language system
  - ✅ **G**: **Generatively** – Designed to model the creation of text
  - ✅ **P**: **Pretrained** – Trained on lots of naturally occurring data
  - ✅ **T**: **Transformer** – A kind of neural network architecture

How are people using it?

Zero-shot

The model predicts the answer given only a natural language description of the task. No gradient updates are performed.

```
Translate English to French:
cheese =>
```

task descriptio
prompt

Zero-shot relies on model already "knowing" how to complete the task.

One-shot

In addition to the task description, the model sees a single example of the task. No gradient updates are performed.

```
Translate English to French:
sea otter => loutre de mer
cheese =>
```

task description
example
prompt

Few-shot

In addition to the task description, the model sees a few examples of the task. No gradient updates are performed.

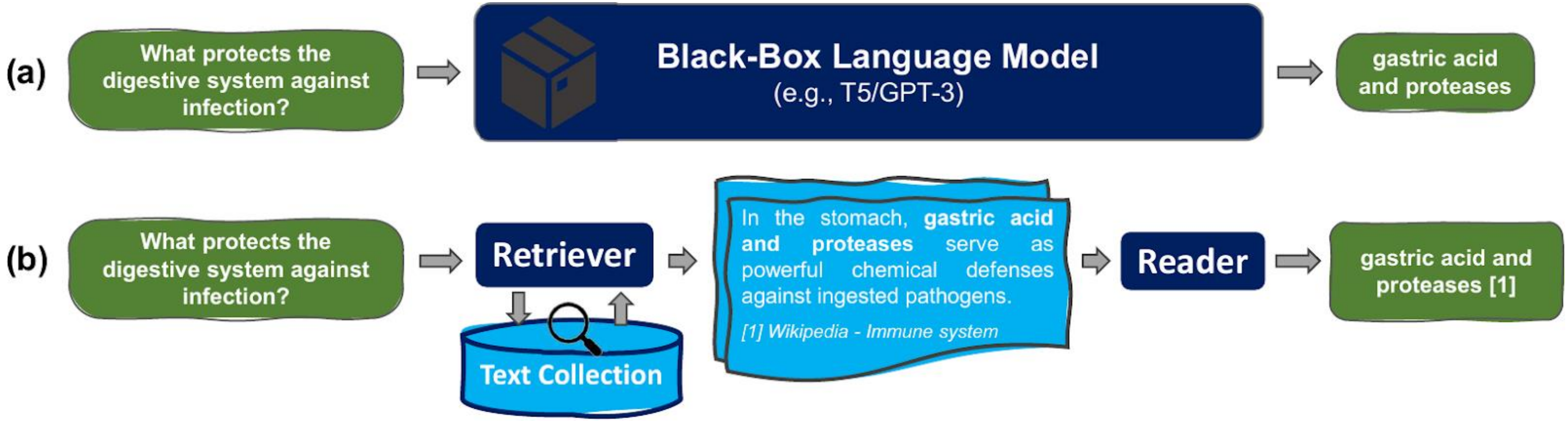
```
Translate English to French:
sea otter => loutre de mer
peppermint => menthe poivrée
plush girafe => girafe peluche
cheese =>
```

task description
examples
prompt

In-context Learning

Retrieval Augmented Generation (RAG)

- Use external data to augment LLMs



https://acl2023-retrieval-lm.github.io

Large Multi-Modal Models (LMMs)

- Combining with vision models to enable visual reasoning

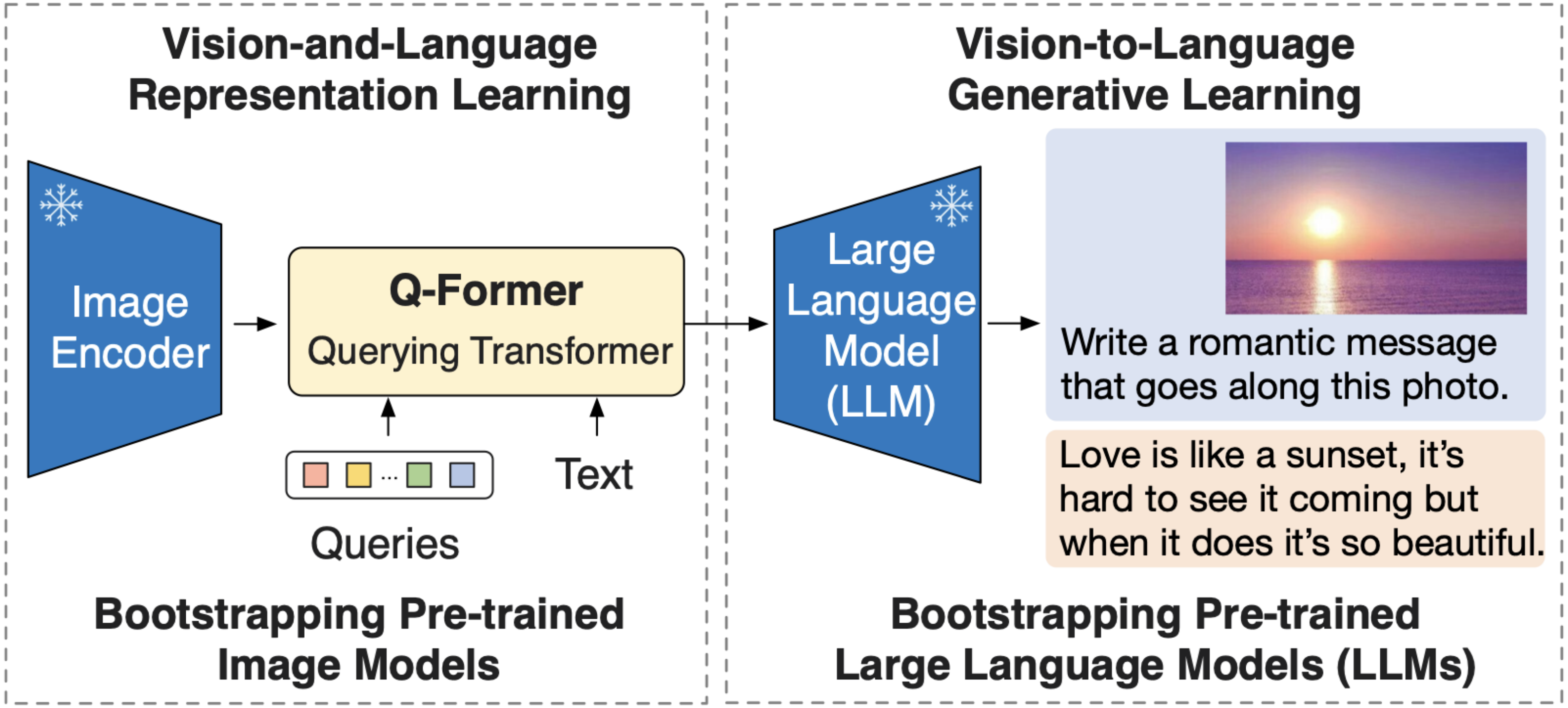


Figure from [BLIP-2 Paper]

Commercial and Open-Source Models

- **Use commercial services:** OpenAI, Google, Anthropic, …
  - State-of-the-art accuracy and fast
  - Constantly changing black box
  - Often affordable: priced per token (1M tokens ~ $10USD)
- **Use open-source models:** Llama3, Vicuña (mine!), Mixtral, …
    - Often built from **Meta's open-source Llama (1,2,3) models**
    - **Varying sizes** (7B, 13B, 30B, 70B) and **quantization levels** (low bit precision)
  - **Variable accuracy** and **speed depends on hardware**
    - Bigger is more accurate and slower

Most organizations are using a mix of these technologies.

Demo (if time): Chatbot Arena

https://chat.lmsys.org

Conclusion

- **Chat**: natural language system
  - **G**: **Generatively** – Designed to model the creation of text
  - **P**: **Pretrained** – Trained on lots of naturally occurring data
  - **T**: **Transformer** – A kind of neural network architecture
- Use cases leveraging **in-context learning**, **retrieval**, and **image reasoning**

Thank you!

Contact Info
Joseph E. Gonzalez
jegonzal@berkeley.edu