How Large Language Models Operate

Or: Transformers, an Introduction

David Maxwell

November 7, 2024

University of Alaska Fairbanks

Nicrophorus vespilloides

Nicrophorus vespilloides

Please tell me about this insect.

This insect appears to be a type of burying beetle, likely belonging to the genus *Nicrophorus*. These beetles are known for their striking black and orange coloration and their behavior of burying small carcasses as a food source for their larvae. The distinct orange markings on its back are characteristic of many species within this genus. Burying beetles play an important role in the ecosystem by recycling nutrients and helping to control the populations of small animals.

Goals For This Talk

- A little historical background on LLMs (large language models)
- Big picture overview on what these machines are
- Some details on the main architecture: transformers

Goals For This Talk

- A little historical background on LLMs (large language models)
- Big picture overview on what these machines are
- Some details on the main architecture: transformers
- Full disclosure: GPT 2.0

Goals For This Talk

- A little historical background on LLMs (large language models)
- Big picture overview on what these machines are
- Some details on the main architecture: transformers
- Full disclosure: GPT 2.0

Not this talk:

- How you train them (pretraining, RLHF, etc)
- Multi-modal models
- Chain of Thought (ChatGPT `o1-preview`)

History

- 1950s: Perceptron
- 1980s: Backpropagation making deep networks trainable

History

- 1950s: Perceptron
- 1980s: Backpropagation making deep networks trainable
- 2014: Bahdanau et. al.
  - attention mechanism

History

- 1950s: Perceptron
- 1980s: Backpropagation making deep networks trainable
- 2014: Bahdanau et. al.
  - attention mechanism
- 2017: Attention is All You Need
  - transformer architecture

History

- 1950s: Perceptron
- 1980s: Backpropagation making deep networks trainable
- 2014: Bahdanau et. al.
  - attention mechanism
- 2017: Attention is All You Need
  - transformer architecture
- 2019: GPT-2
  - focus of talk

History

- 1950s: Perceptron
- 1980s: Backpropagation making deep networks trainable
- 2014: Bahdanau et. al.
    - attention mechanism
- 2017: Attention is All You Need
    - transformer architecture
- 2019: GPT-2
    - focus of talk
- 2022: GPT-3.5
    - we all start paying attention
- 2024: GPT-4o & Claude 3.5 Sonnet

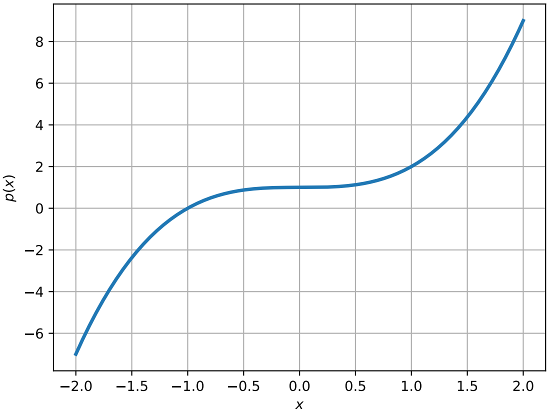
Machines that Depend on Numerical Parameters

Cubic polynomial:

$$p(x) = ax^3 + bx^2 + cx + d$$

Change the parameters and you get different behavior.

$a = 1 \quad b = 0 \quad c = 0 \quad d = 1$



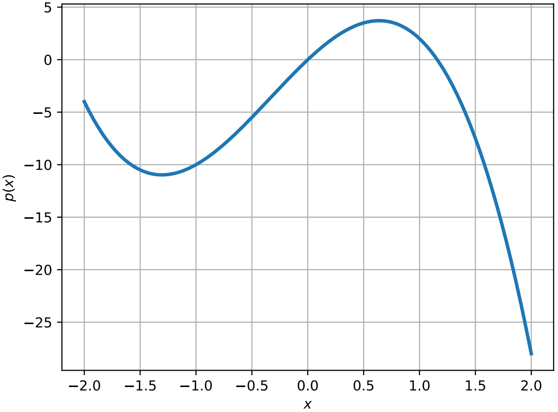
Machines that Depend on Numerical Parameters

$$a = -4 \quad b = -4 \quad c = 10 \quad d = 0$$

Cubic polynomial:

$$p(x) = ax^3 + bx^2 + cx + d$$

Change the parameters and you get different behavior.



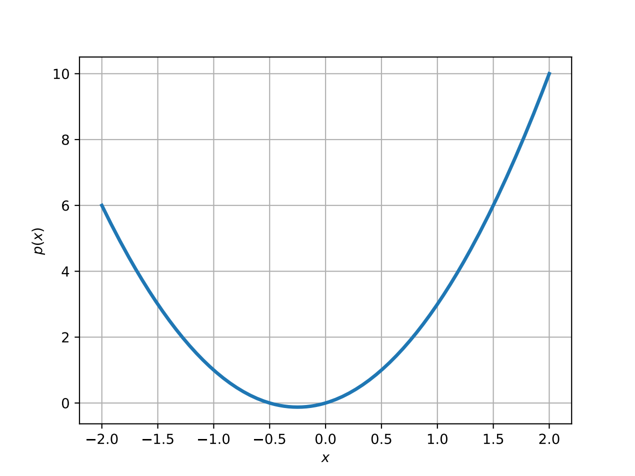
Machines that Depend on Numerical Parameters

$$a = 0 \quad b = 2 \quad c = 1 \quad d = 0$$

Cubic polynomial:

$$p(x) = ax^3 + bx^2 + cx + d$$

Change the parameters and you get different behavior.



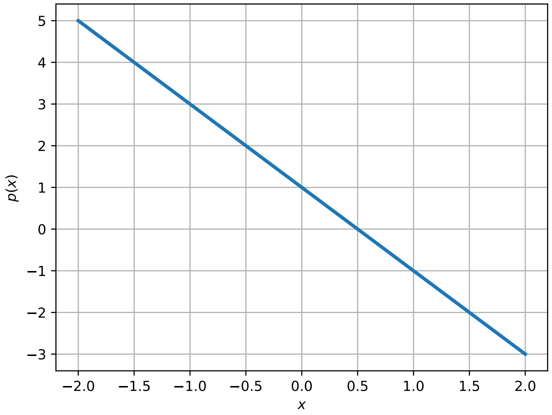
Machines that Depend on Numerical Parameters

$$a = 0 \quad b = 0 \quad c = -2 \quad d = 1$$

Cubic polynomial:

$$p(x) = ax^3 + bx^2 + cx + d$$

Change the parameters and you get different behavior.



Tokens

Input text is broken into a sequence of tokens:

Can	adians	are	friendly	and	approach	able	.

Tokens

Input text is broken into a sequence of tokens:

Can	adians	are	friendly	and	approach	able	.

Tokens are represented by small integers:

6854	21398	527	11919	323	5603	481	13

Tokens

Input text is broken into a sequence of tokens:

Can	adians	are	friendly	and	approach	able	.

Tokens are represented by small integers:

6854	21398	527	11919	323	5603	481	13

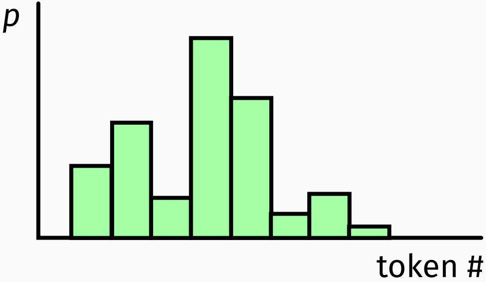
- Roundtrip unicode text to token sequence to unicode text is lossless.
- GPT-2 has a vocabulary of roughly 50000 tokens.
- GPT-3.5: 100000 tokens.

The basic function

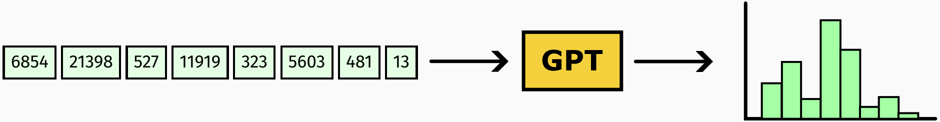
GPT-2 is a function, depending on a very large number of numerical parameters.

Input: A sequence of up to 1024 tokens (the **context window**).

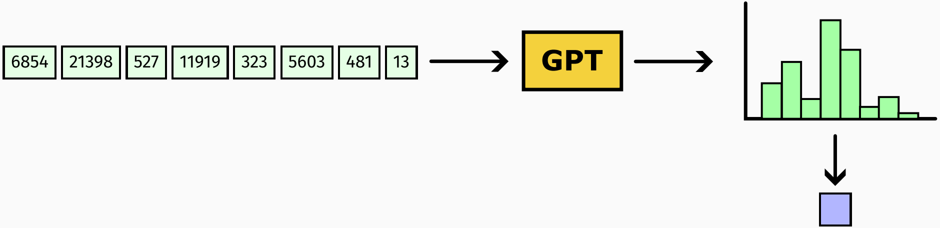
Output: A probability distribution over tokens.



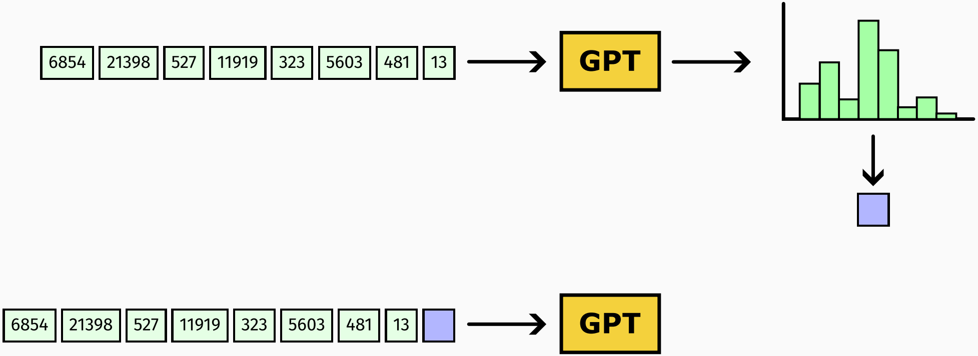
The Iteration



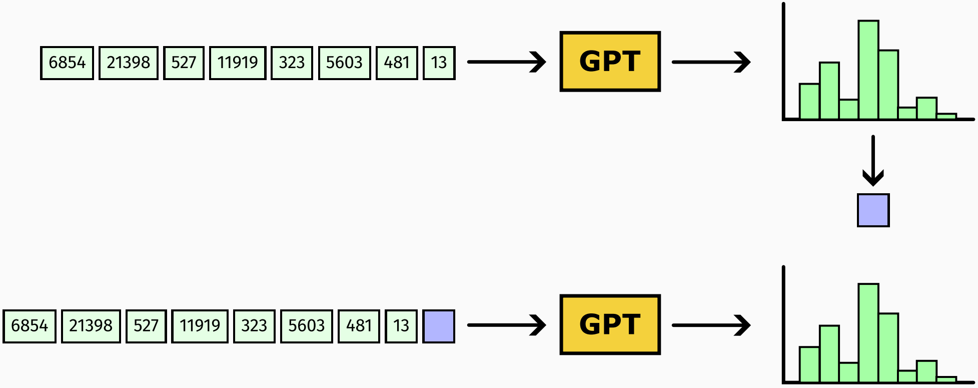
The Iteration



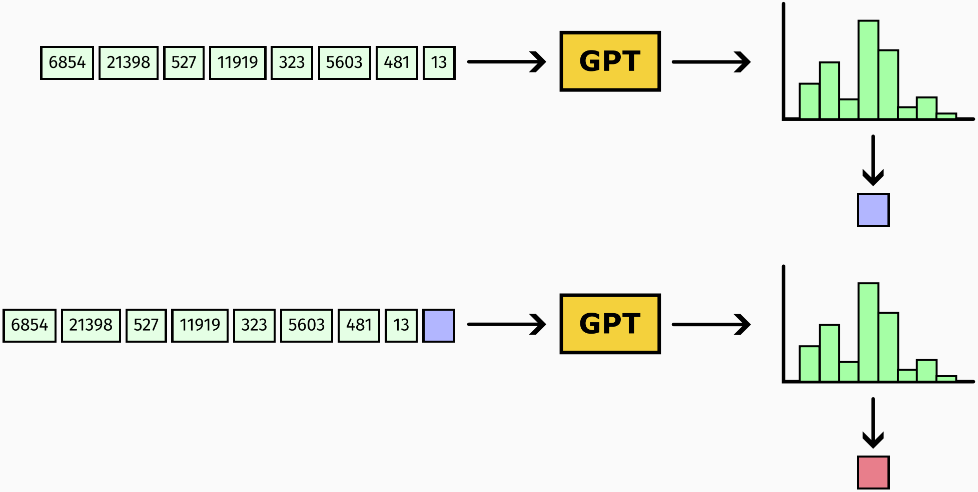
The Iteration



The Iteration



The Iteration



Implications

- The GPTs have no **persistent** state except for:
  - The fixed model parameters (i.e. ‘coefficients’)
  - The tokens of the conversation thus far

Implications

- The GPTs have no **persistent** state except for:
  - The fixed model parameters (i.e. ‘coefficients’)
  - The tokens of the conversation thus far
- No advanced planning of what to say
- No memory
- No new ‘learning’
- No ruminating

Implications

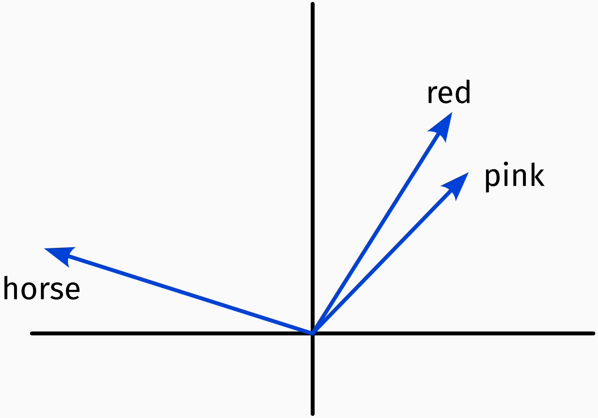
- The GPTs have no **persistent** state except for:
  - The fixed model parameters (i.e. ‘coefficients’)
  - The tokens of the conversation thus far
- No advanced planning of what to say
- No memory
- No new ‘learning’
- No ruminating
- Constant reprocessing of past tokens
  - $O(n^2)$ in the length of the context window

Implications

- The GPTs have no **persistent** state except for:
  - The fixed model parameters (i.e. ‘coefficients’)
  - The tokens of the conversation thus far
- No advanced planning of what to say
- No memory
- No new ‘learning’
- No ruminating
- Constant reprocessing of past tokens
  - $O(n^2)$ in the length of the context window
- Very static $\implies$ very safe

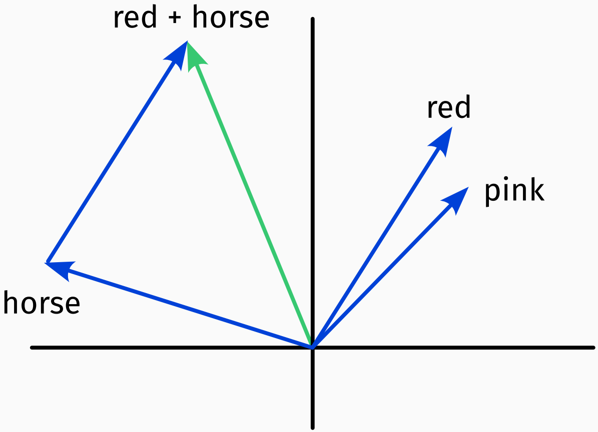
Latent Space

Tokens are immediately converted to vectors.



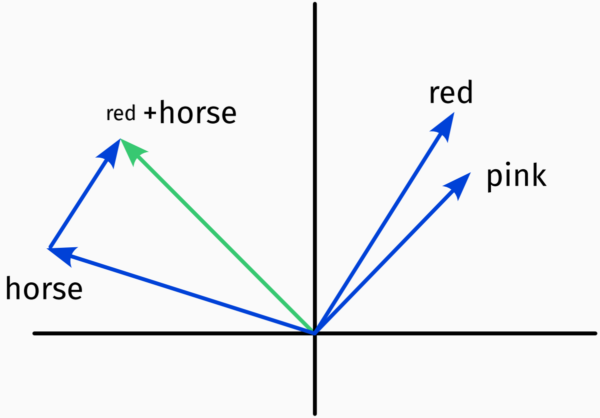
Latent Space

Tokens are immediately converted to vectors.



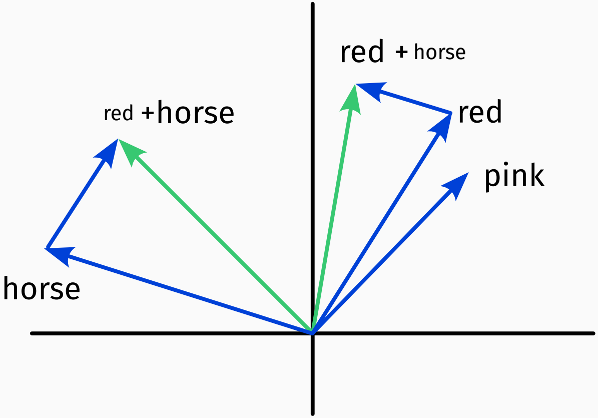
Latent Space

Tokens are immediately converted to vectors.



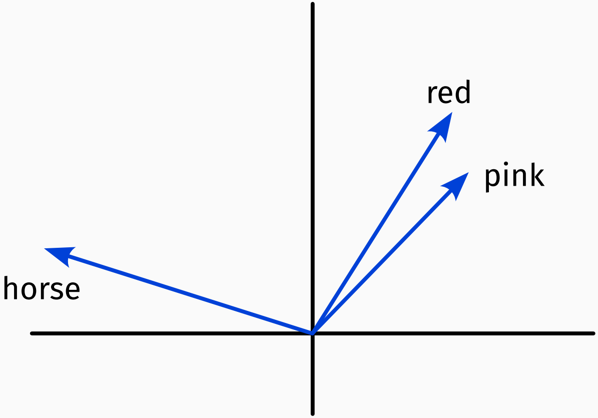
Latent Space

Tokens are immediately converted to vectors.

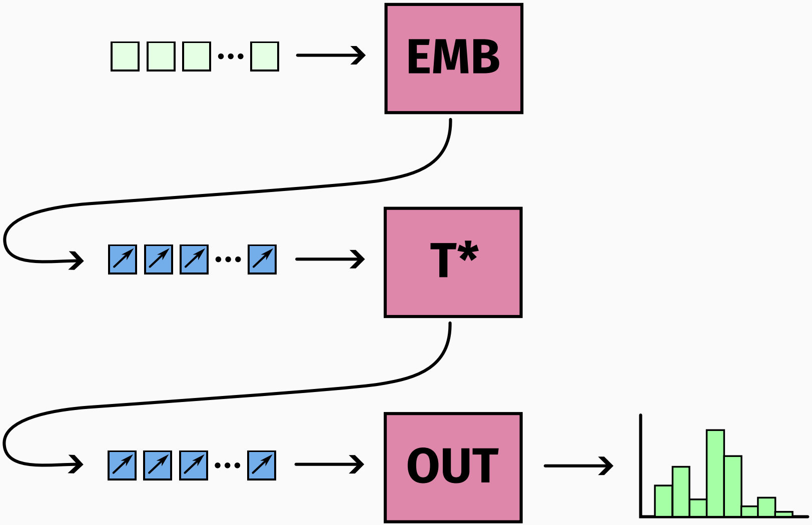


Latent Space

Tokens are immediately converted to vectors. (Actual dimension: 768 for GPT-2)



Main Pipeline

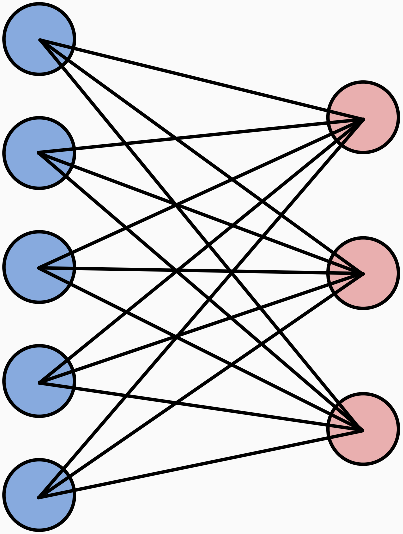


Ingredient: Linear Maps

A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear if:

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
$$f(c\mathbf{x}) = cf(\mathbf{x})$$

for all inputs $\mathbf{x}$ and $\mathbf{y}$ and all numbers c.



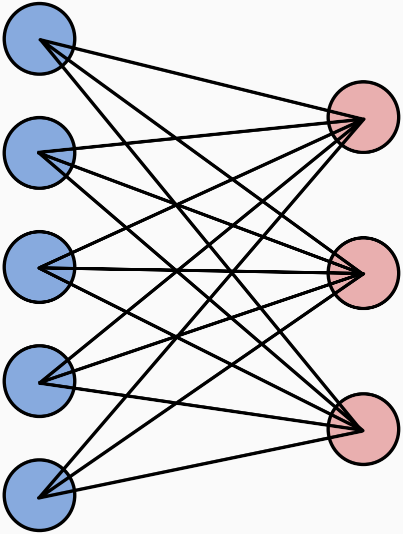
Ingredient: Linear Maps

A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear if:

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
$$f(c\mathbf{x}) = cf(\mathbf{x})$$

for all inputs $\mathbf{x}$ and $\mathbf{y}$ and all numbers c.

- We can represent such a map via a collection of $n \cdot m$ weights



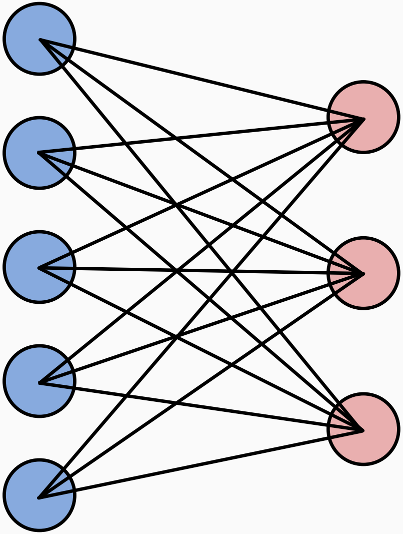
Ingredient: Linear Maps

A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear if:

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
$$f(c\mathbf{x}) = cf(\mathbf{x})$$

for all inputs $\mathbf{x}$ and $\mathbf{y}$ and all numbers c.

- We can represent such a map via a collection of $n \cdot m$ weights
- GPUs are great at computing these



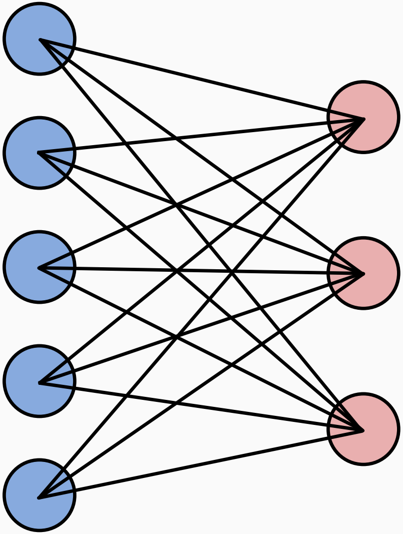
Ingredient: Linear Maps

A map $f : \mathbb{R}^n \to \mathbb{R}^m$ is linear if:

$$f(\mathbf{x} + \mathbf{y}) = f(\mathbf{x}) + f(\mathbf{y})$$
$$f(c\mathbf{x}) = cf(\mathbf{x})$$

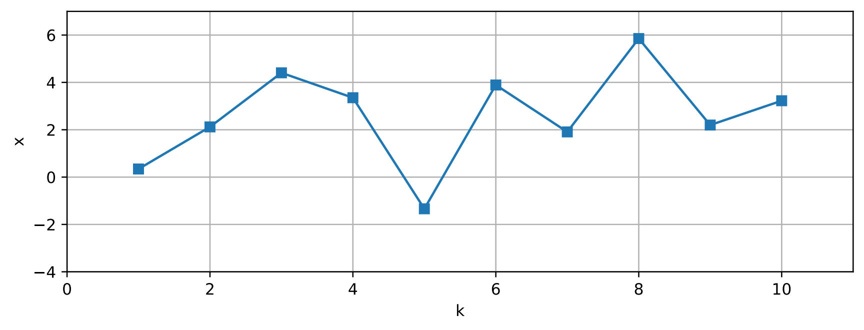
for all inputs $\mathbf{x}$ and $\mathbf{y}$ and all numbers c.

- We can represent such a map via a collection of $n \cdot m$ weights
- GPUs are great at computing these
- Shameless plug: Math 314!



Ingredient: Weight and Balance

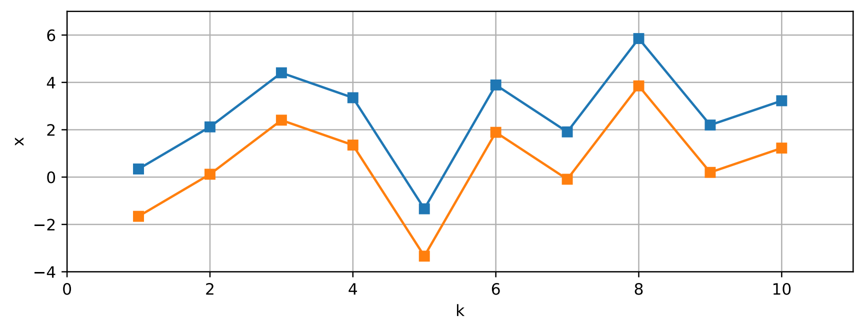
“Weight and balance” steps keep vectors in latent space at a reasonable size.



Ingredient: Weight and Balance

"Weight and balance" steps keep vectors in latent space at a reasonable size.

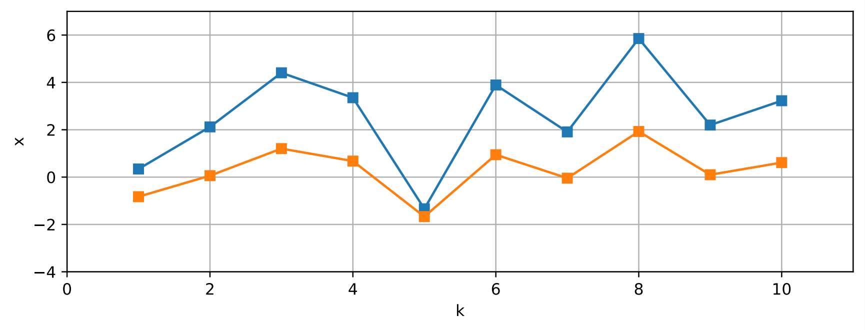
1. Remove the mean



Ingredient: Weight and Balance

"Weight and balance" steps keep vectors in latent space at a reasonable size.

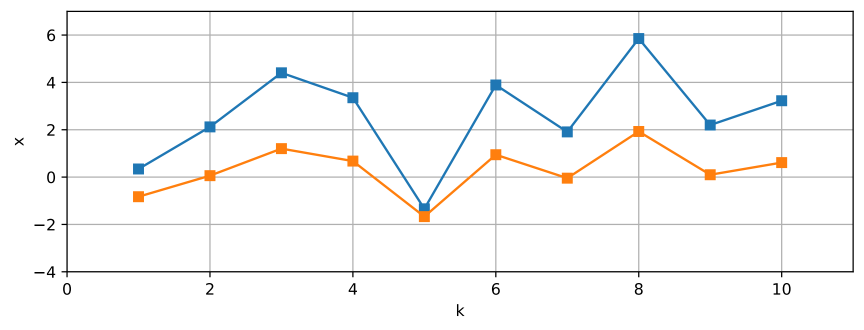
1. Remove the mean
2. Scale to unit variance



Ingredient: Weight and Balance

“Weight and balance” steps keep vectors in latent space at a reasonable size.

1. Remove the mean
2. Scale to unit variance

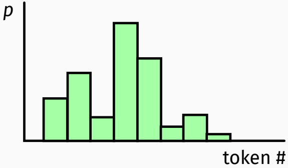


3. Training: new scale, new “zero vector”

Ingredient: softmax

The final output is a probability distribution:

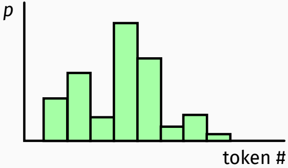
$$(p_1, p_2, \ldots, p_{50000}), \quad p_i \geq 0, \quad \sum_{i=1}^{50000} p_i = 1$$



Ingredient: softmax

The final output is a probability distribution:

$$(p_1, p_2, \ldots, p_{50000}), \quad p_i \geq 0, \quad \sum_{i=1}^{50000} p_i = 1$$



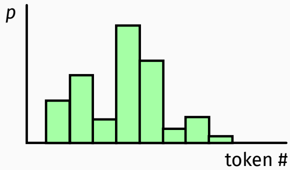
softmax

1. Start with arbitrary $(w_1, w_2, \ldots, w_{50000})$

Ingredient: softmax

The final output is a probability distribution:

$$(p_1, p_2, \ldots, p_{50000}), \quad p_i \geq 0, \quad \sum_{i=1}^{50000} p_i = 1$$



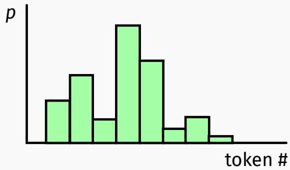
softmax

1. Start with arbitrary $(w_1, w_2, \ldots, w_{50000})$
2. Make $(q_1, q_2, \ldots, q_{50000})$ with $q_i = e^{w_i}$
    - Observe $q_i \geq 0$

Ingredient: softmax

The final output is a probability distribution:

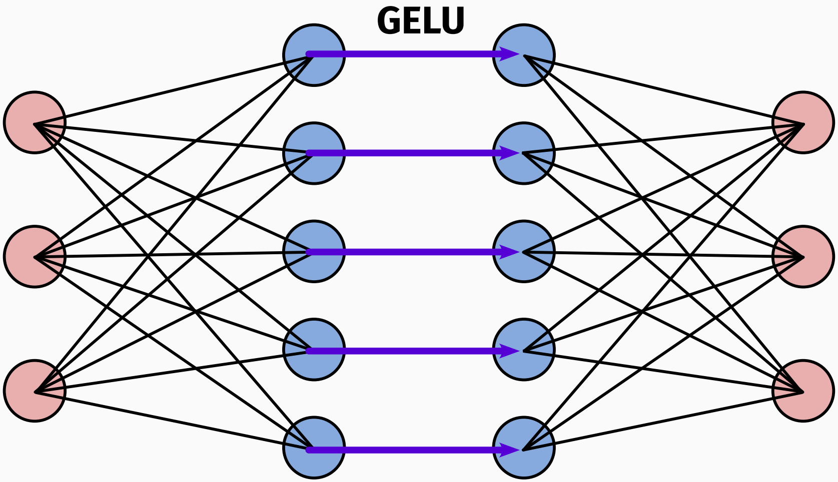
$$(p_1, p_2, \ldots, p_{50000}), \quad p_i \geq 0, \quad \sum_{i=1}^{50000} p_i = 1$$



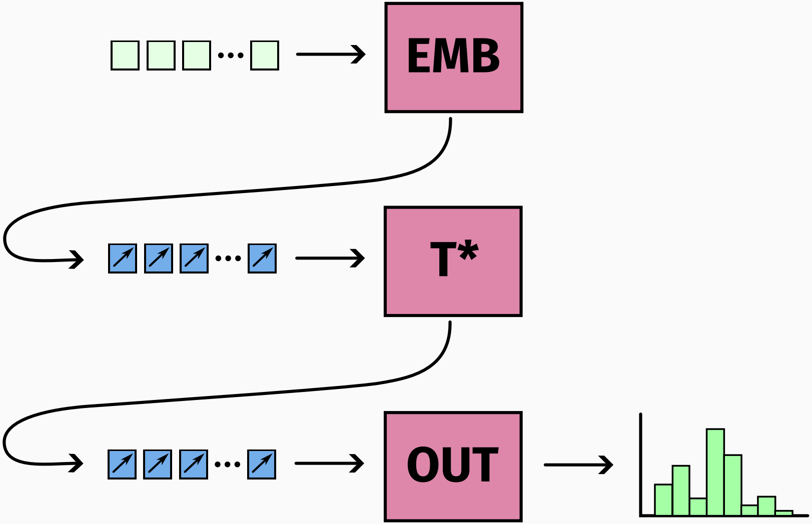
softmax

1. Start with arbitrary $(w_1, w_2, \ldots, w_{50000})$
2. Make $(q_1, q_2, \ldots, q_{50000})$ with $q_i = e^{w_i}$
   - Observe $q_i \geq 0$
3. Let $q_{\text{total}} = q_1 + q_2 + \cdots + q_{50000}$
4. Then $p_i = q_i / q_{\text{total}}$

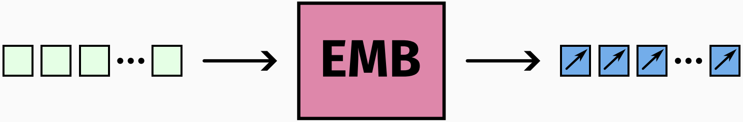
Ingredient: Thin Neural Net (Feedforward Layer)



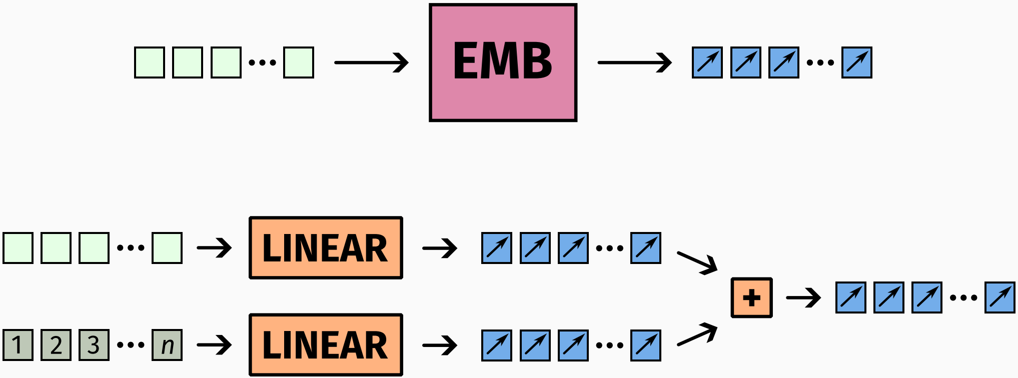
Main Pipeline



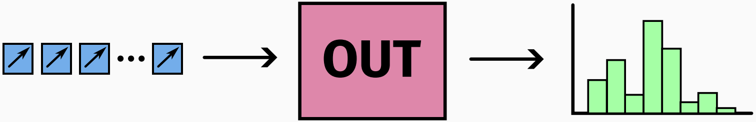
Token Embedding



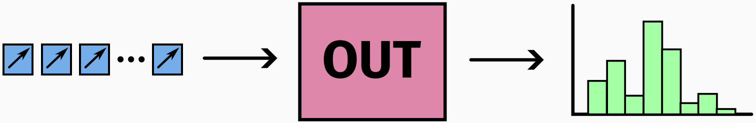
Token Embedding

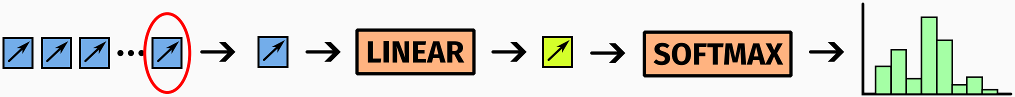


Final Output

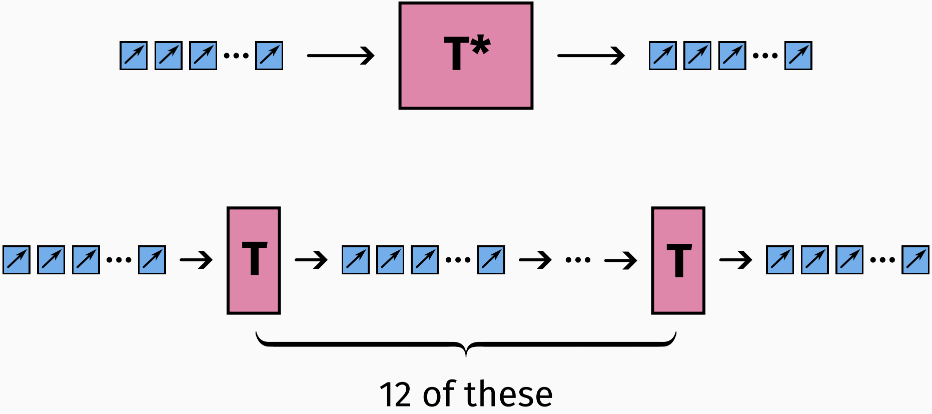


Final Output





Stack of Transformers



Motivation for Attention

- Natural language translation
- Distant information needs to be associated

I get up at 6:30 in the morning.

Morgens stehe ich um halb sieben auf.

Motivation for Attention

- Natural language translation
- Distant information needs to be associated

I get up at 6:30 in the morning.

Morgens stehe ich um halb sieben auf.

Motivation for Attention

- Natural language translation
- Distant information needs to be associated

I get up at 6:30 in the morning.

Morgens stehe ich um halb sieben auf.

Motivation for Attention

- Natural language translation
- Distant information needs to be associated

I get up at 6:30 in the morning.

Morgens stehe ich um halb sieben auf.

Motivation for Attention

- Natural language translation
- Distant information needs to be associated

I get up at 6:30 in the morning.

Morgens stehe ich um halb sieben auf.

Motivation for Attention

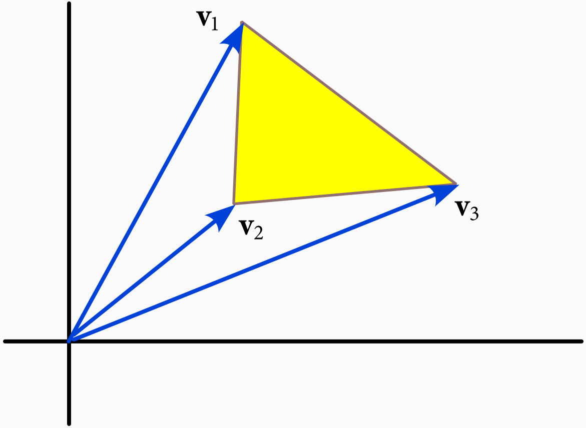
- Natural language translation
- Distant information needs to be associated

I get up at 6:30 in the morning.

Morgens stehe ich um halb sieben auf.

Bahdanau et. al, Neural Machine Translation by Jointly Learning to Align and Translate, 2014.

Combining information = convex combinations

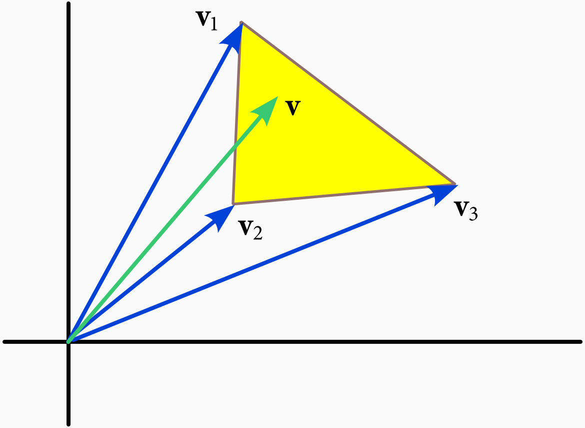


$$\mathbf{v} = b_1\mathbf{v}_1 + b_2\mathbf{v}_2 + b_3\mathbf{v}_3$$

$$0 \le b_i \le 1$$

$$b_1 + b_2 + b_3 = 1$$

Combining information = convex combinations



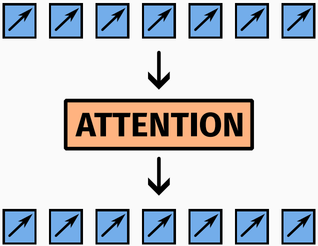
$$\mathbf{v} = b_1\mathbf{v}_1 + b_2\mathbf{v}_2 + b_3\mathbf{v}_3$$

$$0 \leq b_i \leq 1$$

$$b_1 + b_2 + b_3 = 1$$

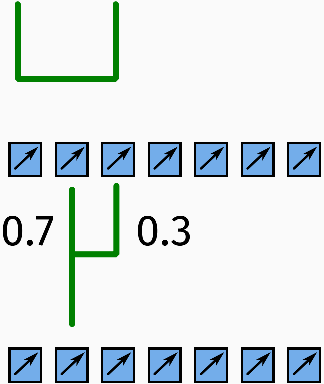
Attention Module

Morgens stehe ich um halb sieben auf



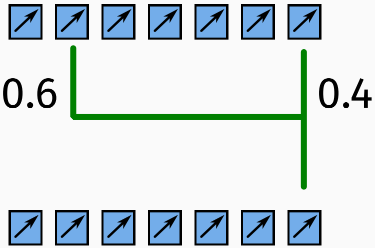
Attention Module

Morgens stehe ich um halb sieben auf



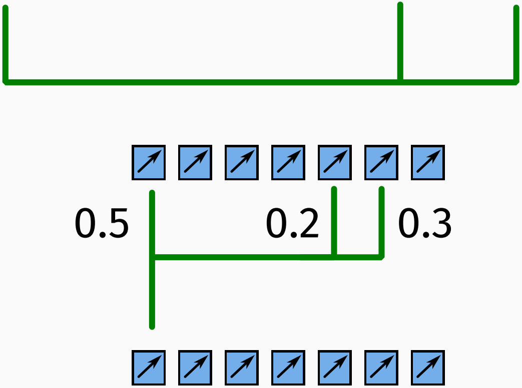
Attention Module

Morgens stehe ich um halb sieben auf



Attention Module

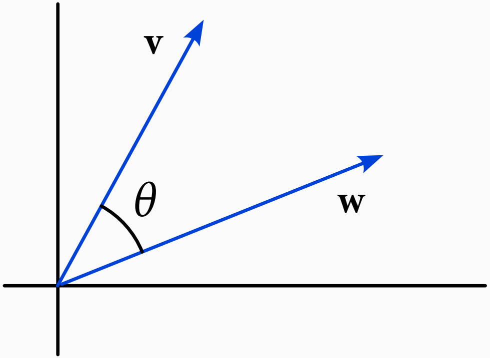
Morgens stehe ich um halb sieben auf



Determination of Weights I

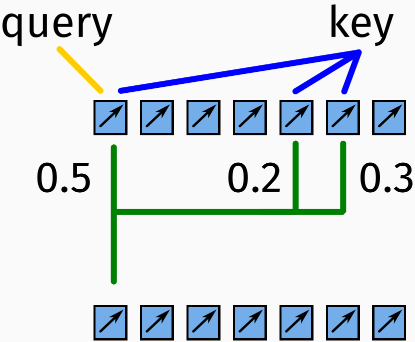
Dot product measures alikeness of vectors.

$$\mathbf{v} \cdot \mathbf{w} = ||\mathbf{v}||\mathbf{w}|| \cos\theta$$
$$= v_1 w_1 + v_2 w_2 + \cdots + v_n w_n$$



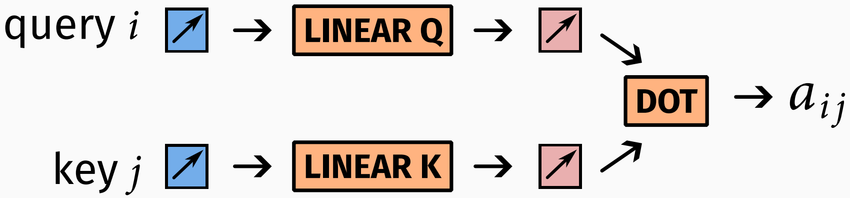
Determination of Weights II

Computing the weights a_{ij} where key j contributes to query i:



Determination of Weights II

Computing the weights a_{ij} where key j contributes to query i:



Determination of Weights III

Two details:

- The weights so far don't make a convex combination.
  - Use softmax: $a_{ij} \to b_{ij}$ to ensure $0 \leq b_{ij} \leq 1$ and $\sum_j b_{ij} = 1$

Determination of Weights III

Two details:

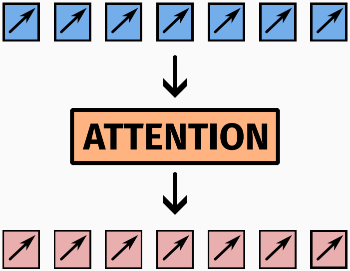
- The weights so far don't make a convex combination.
  - Use softmax: $a_{ij} \to b_{ij}$ to ensure $0 \leq b_{ij} \leq 1$ and $\sum_j b_{ij} = 1$
- We don't actually make convex combinations of the original vectors.

value j  → LINEAR V → 

Determination of Weights III

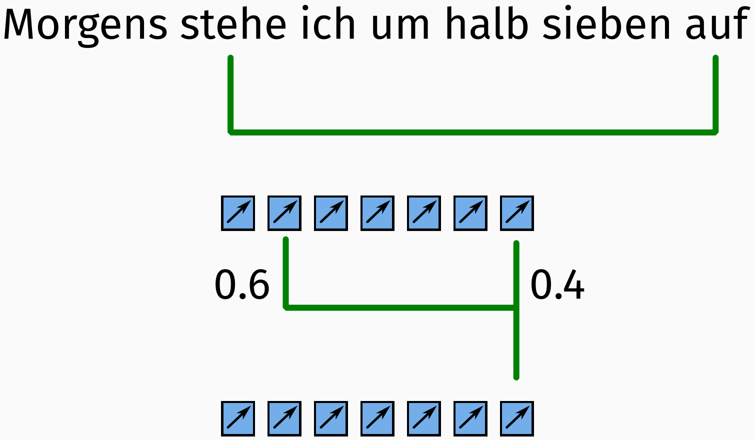
Two details:

- The weights so far don't make a convex combination.
  - Use softmax: $a_{ij} \to b_{ij}$ to ensure $0 \leq b_{ij} \leq 1$ and $\sum_j b_{ij} = 1$
- We don't actually make convex combinations of the original vectors.



Causality

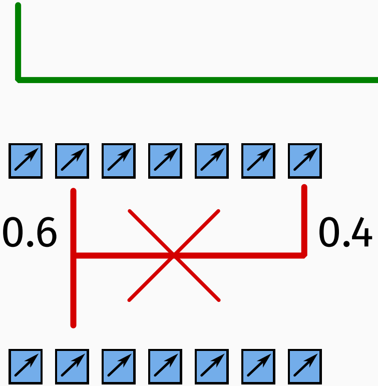
One more detail: each slot can only use information from the past



Causality

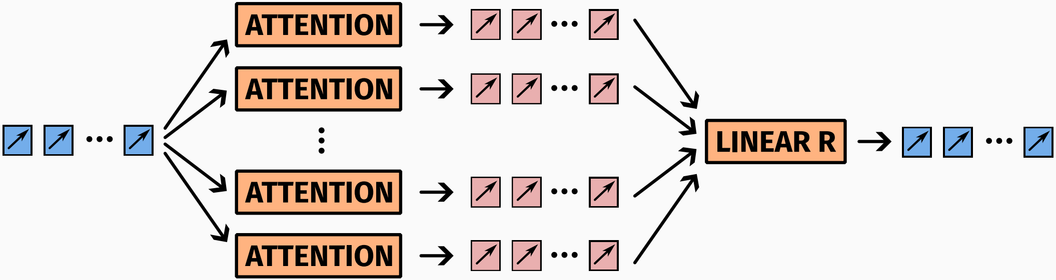
One more detail: each slot can only use information from the past

Morgens stehe ich um halb sieben auf



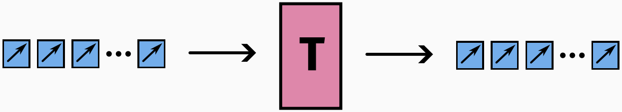
Multihead Attention

Increased parallelism by having more than one attention block happen at the same time.



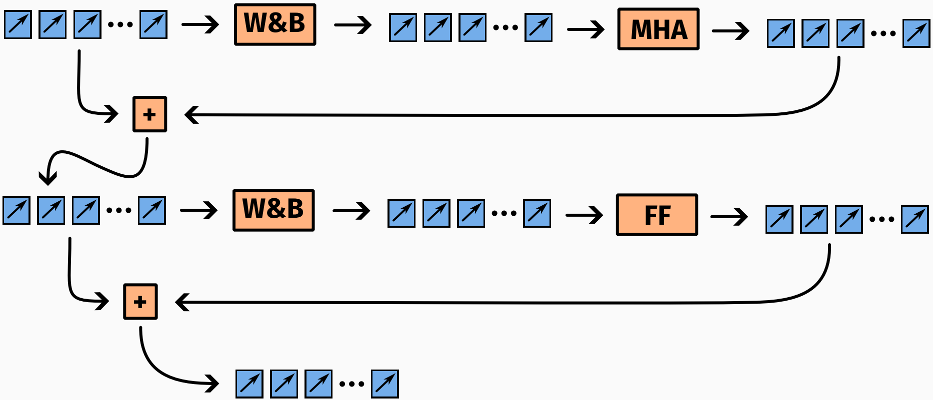
A Single Transformer

Attention is All You Need, Vaswani et. al., 2017



A Single Transformer

Attention is All You Need, Vaswani et. al., 2017



Summary

- An LLM is a map from sequences of tokens to probability distributions over token space.

Summary

- An LLM is a map from sequences of tokens to probability distributions over token space.
- The map is deterministic, but selection of tokens from the distribution can be probabilistic.

Summary

- An LLM is a map from sequences of tokens to probability distributions over token space.
- The map is deterministic, but selection of tokens from the distribution can be probabilistic.
- The entire stream of tokens is reprocessed from scratch to generate the next token.

Summary

- An LLM is a map from sequences of tokens to probability distributions over token space.
- The map is deterministic, but selection of tokens from the distribution can be probabilistic.
- The entire stream of tokens is reprocessed from scratch to generate the next token.
- The inner machinery is made entirely out of familiar mathematical maps:
  - Linear maps
  - Dot products, scaling, vector addition
  - Element-wise activation functions
  - softmax

Thank you!

Parameter Counts

GPT-2: ~125M parameters

1. EMB and OUT linear maps: 40%
2. Feed forward 45%
3. Attention linear maps: 15%

Parameter Counts

GPT-3: ~175B parameters

1. EMB and OUT linear maps: 20%
2. Feed forward 60%
3. Attention linear maps: 20%