Transformers

Learning to pay attention

Introduction to Machine Learning, Spring 2026

Sequence Modelling

Inputs are sequences $\{x_t\}$ and we wish to model them

Given$\{x_1, \cdots, x_{t-1}\}$ can we predict x_t?

Sequence Modelling

Inputs are sequences $\{x_t\}$ and we wish to model them

Given$\{x_1, \cdots, x_{t-1}\}$ can we predict x_t?

Examples:

- Stock prediction
- Next token prediction (language models!)
- Weather Prediction, etc

Sequence Modelling - Challenges

Sequence Modelling - Challenges

Variable length inputs

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Need to learn temporal order

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Need to learn temporal order

Remembering “context”

Sequence Modelling - Challenges

Variable length inputs

What if sequences are very long?

Need to learn temporal order

Remembering “context”

Predictions need to somehow use the entire history

Approach #1: RNNs

Key Idea

Represent the entire context as a hidden variable $\boldsymbol{h} \in \mathbb{R}^d$

Key Idea

Represent the entire context as a hidden variable $\boldsymbol{h} \in \mathbb{R}^d$

Given the hidden variable, generate the next output

Pitfalls

Pitfalls

Forgets history: for a long context, the hidden state contains very little information about the starting tokens

Pitfalls

Forgets history: for a long context, the hidden state contains very little information about the starting tokens

Values and gradients either explode or vanish - unstable

Pitfalls

Forgets history: for a long context, the hidden state contains very little information about the starting tokens

Values and gradients either explode or vanish - unstable

Hard to parallelise - only run sequentially

Increases cost of training and inference

Approach #2: Transformers

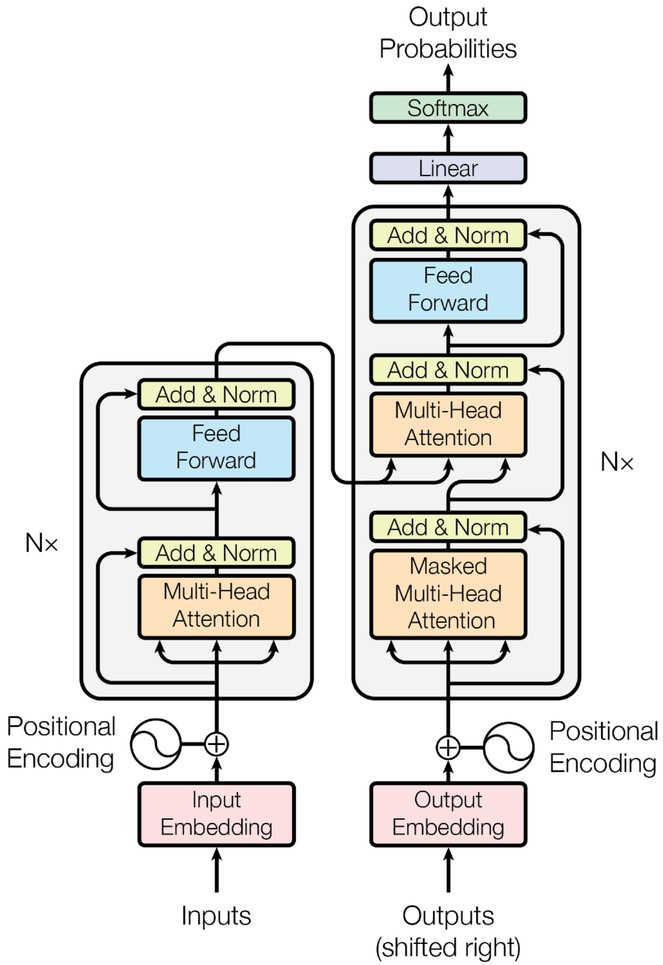
Believe us, attention is all you need

Introduction to Machine Learning, Spring 2026

Vaswani et al. on 12th June 2017

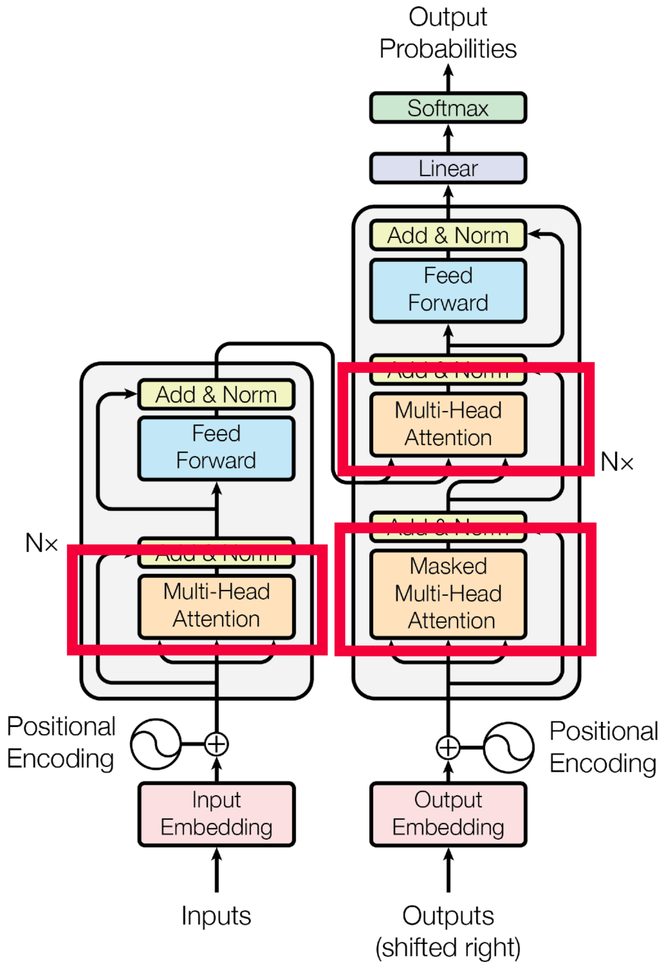
Attention Is All You Need

Vaswani et al. on 12th June 2017



Attention Is All You Need

Vaswani et al. on 12th June 2017



Problem Setup

Problem Setup

Lets consider a next word prediction task -
we have a sequence of words $w_1, w_2, \ldots, w_n$

Problem Setup

Lets consider a next word prediction task -
we have a sequence of words $w_1, w_2, \ldots, w_n$

All $w_i \in \mathbb{R}^d$, some latent space

Problem Setup

Lets consider a next word prediction task -
we have a sequence of words $w_1, w_2, \ldots, w_n$

All $w_i \in \mathbb{R}^d$, some latent space

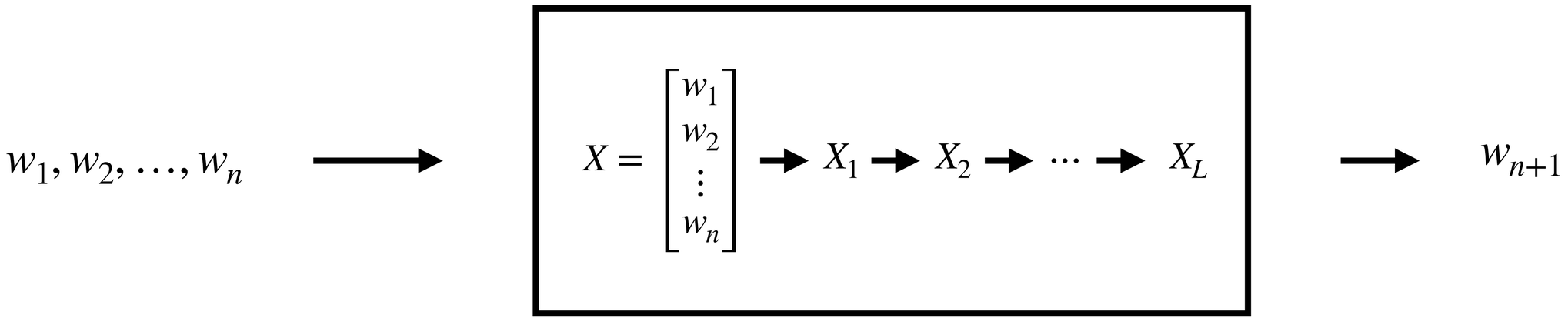
We have a fixed vocabulary set V
For example, all possible words

Problem Setup

Problem Setup

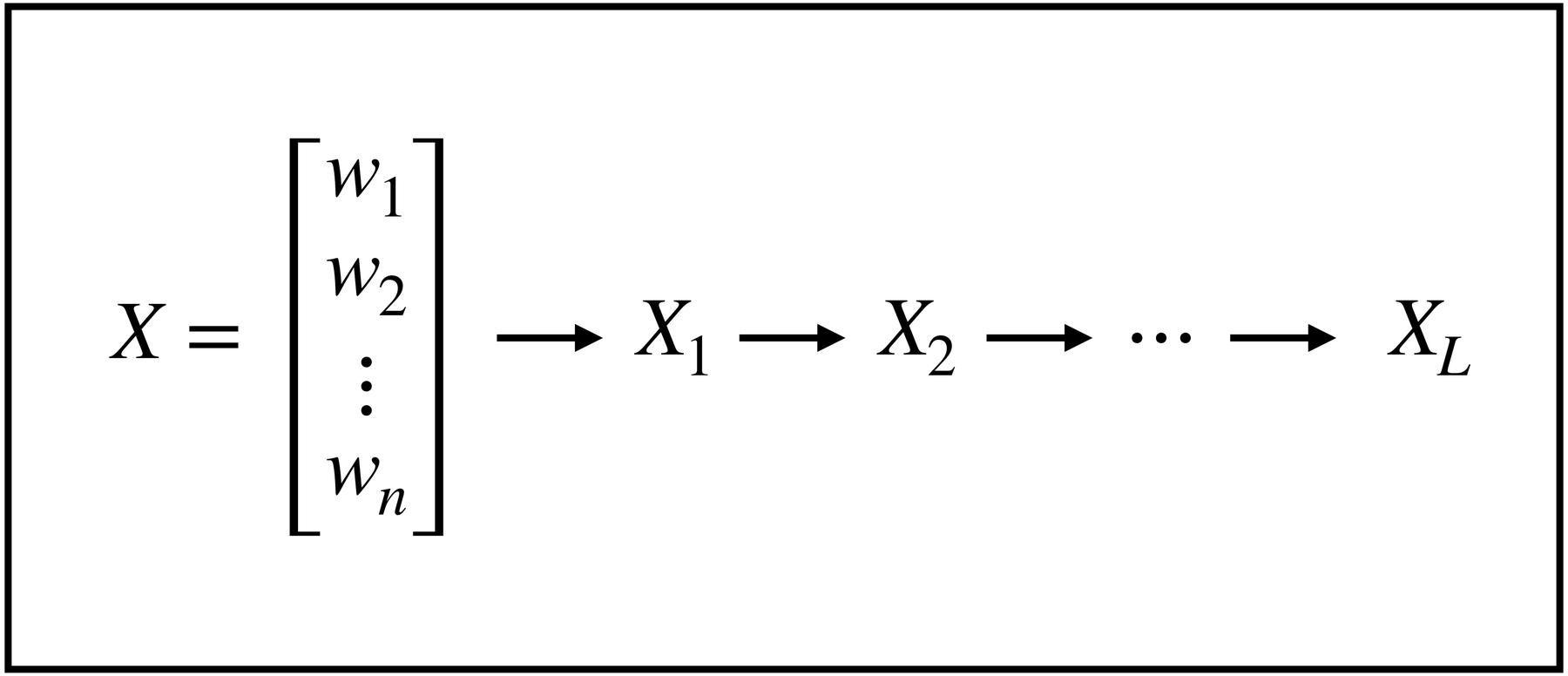
The loss is clear: cross entropy

How transformers work

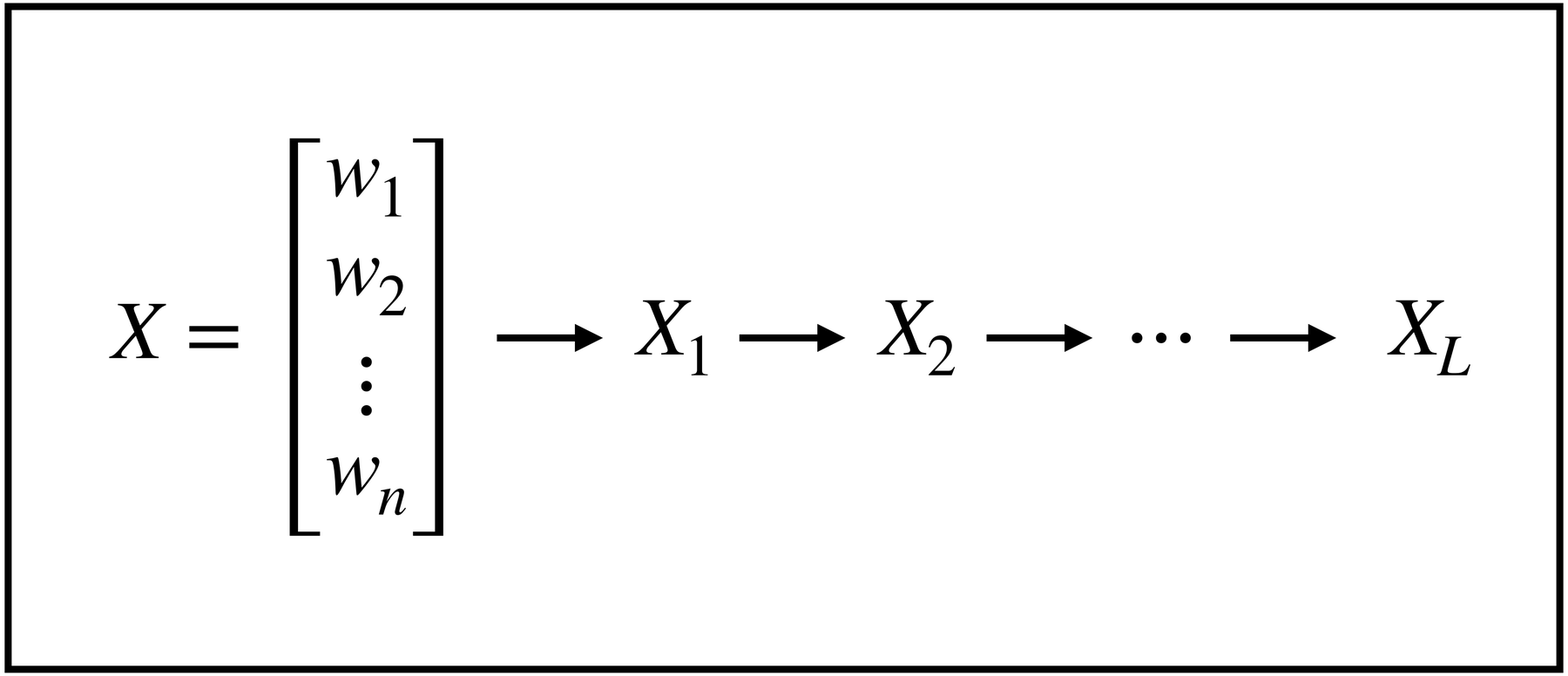


The matrices are ***transformed*** many times

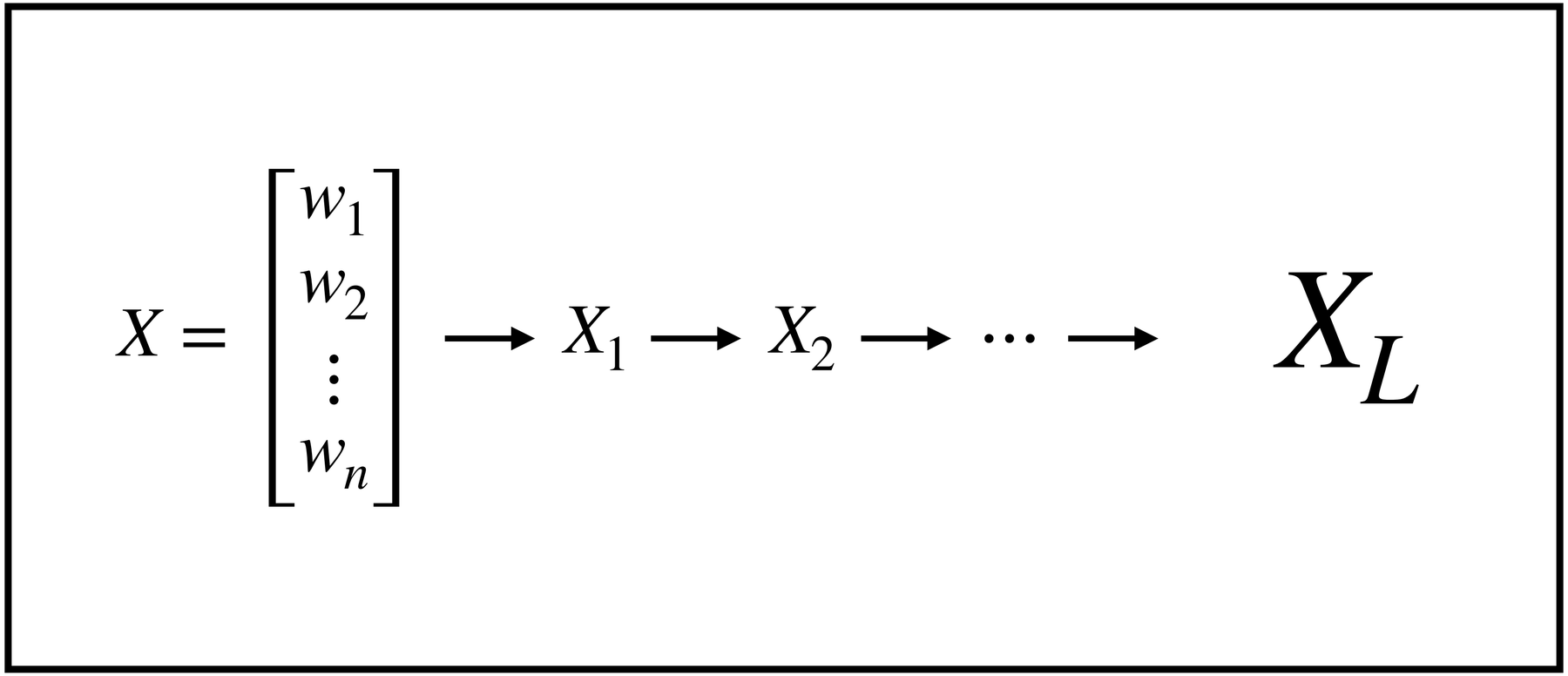
How transformers work



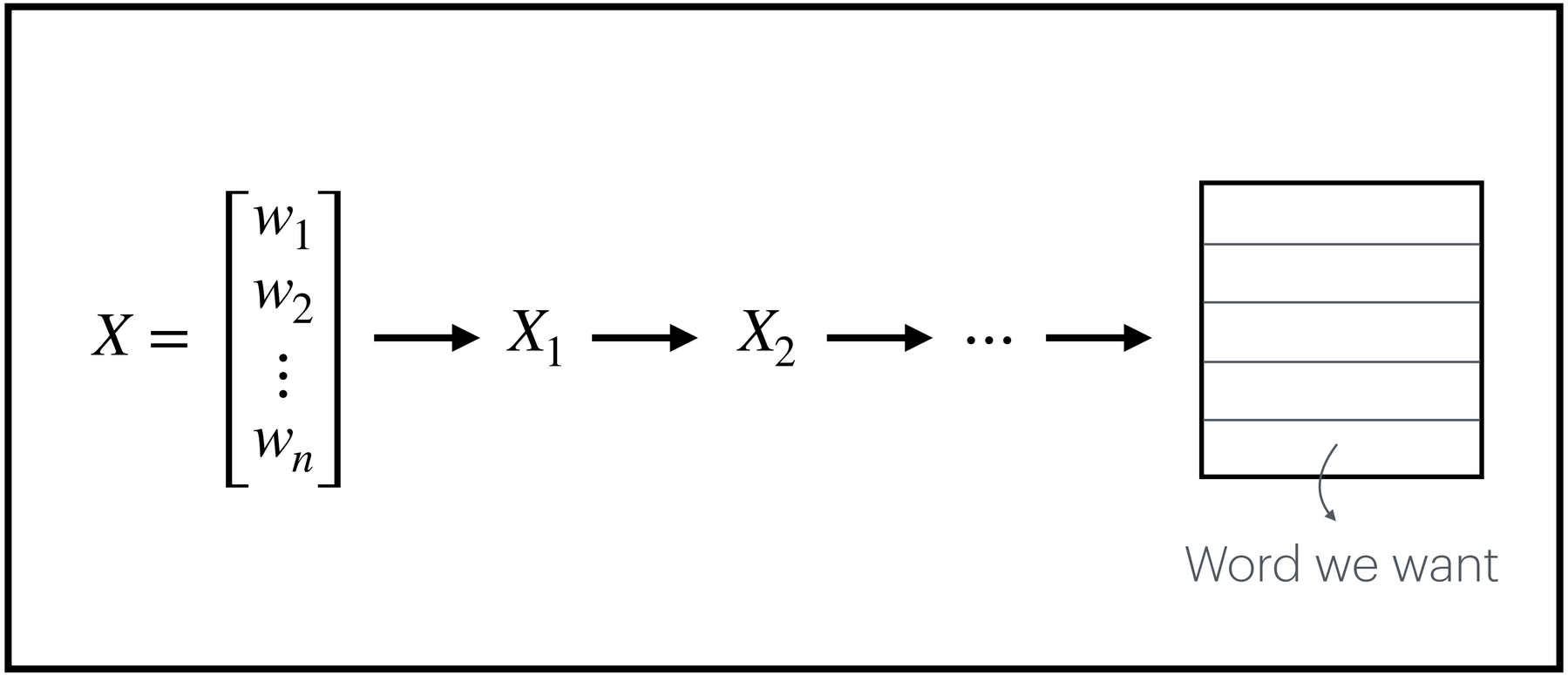
How transformers work



How transformers work



How transformers work



How transformers work

How do we model these transformations in matrices?

The transformation

The transformation

What should a good transformation do?

The transformation

What should a good transformation do?

How can we nicely share information between words?

Before that, some linear algebra

Before that, some linear algebra

Matrix multiplication

Before that, some linear algebra

Matrix multiplication

$$A \in \mathbb{R}^{a \times b}, \quad B \in \mathbb{R}^{c \times b} \Rightarrow AB^T \in \mathbb{R}^{a \times c}$$

$$\left(AB^T\right)_{ij} = a_i \cdot b_j$$

where a_i is the i^{th} row of A and b_j is the j^{th} row of B

Some more linear algebra

Some more linear algebra

Dot Product

Some more linear algebra

Dot Product

$$u \cdot v = u_1 v_1 + \ldots + u_n v_n$$

Some more linear algebra

Dot Product

$$u \cdot v = u_1 v_1 + \ldots + u_n v_n$$

Measures the angle between two vectors

How “aligned” are the vectors

Some more linear algebra

Dot Product

$$u \cdot v = u_1 v_1 + \ldots + u_n v_n$$

Measures the angle between two vectors

How “aligned” are the vectors

$$\left(AB^T\right)_{ij} = a_i \cdot b_j$$

Entries of AB^T measure how close are the rows of A and B

Share the information?

Share the information?

Let $X \in \mathbb{R}^{n \times d}$ be the matrix where the i^{th} row is the embedding w_i

Share the information?

Let $X \in \mathbb{R}^{n \times d}$ be the matrix where the i^{th} row is the embedding w_i

What does XX^T mean?

Share the information?

Let $X \in \mathbb{R}^{n \times d}$ be the matrix where the i^{th} row is the embedding w_i

What does XX^T mean?

$$\left(XX^T\right)_{ij} = x_i \cdot x_j$$

Measures how similar the embeddings are to each other

How relevant are two words to each other?

Finding similar words

$$A = \text{softmax}\left(XX^T\right)$$

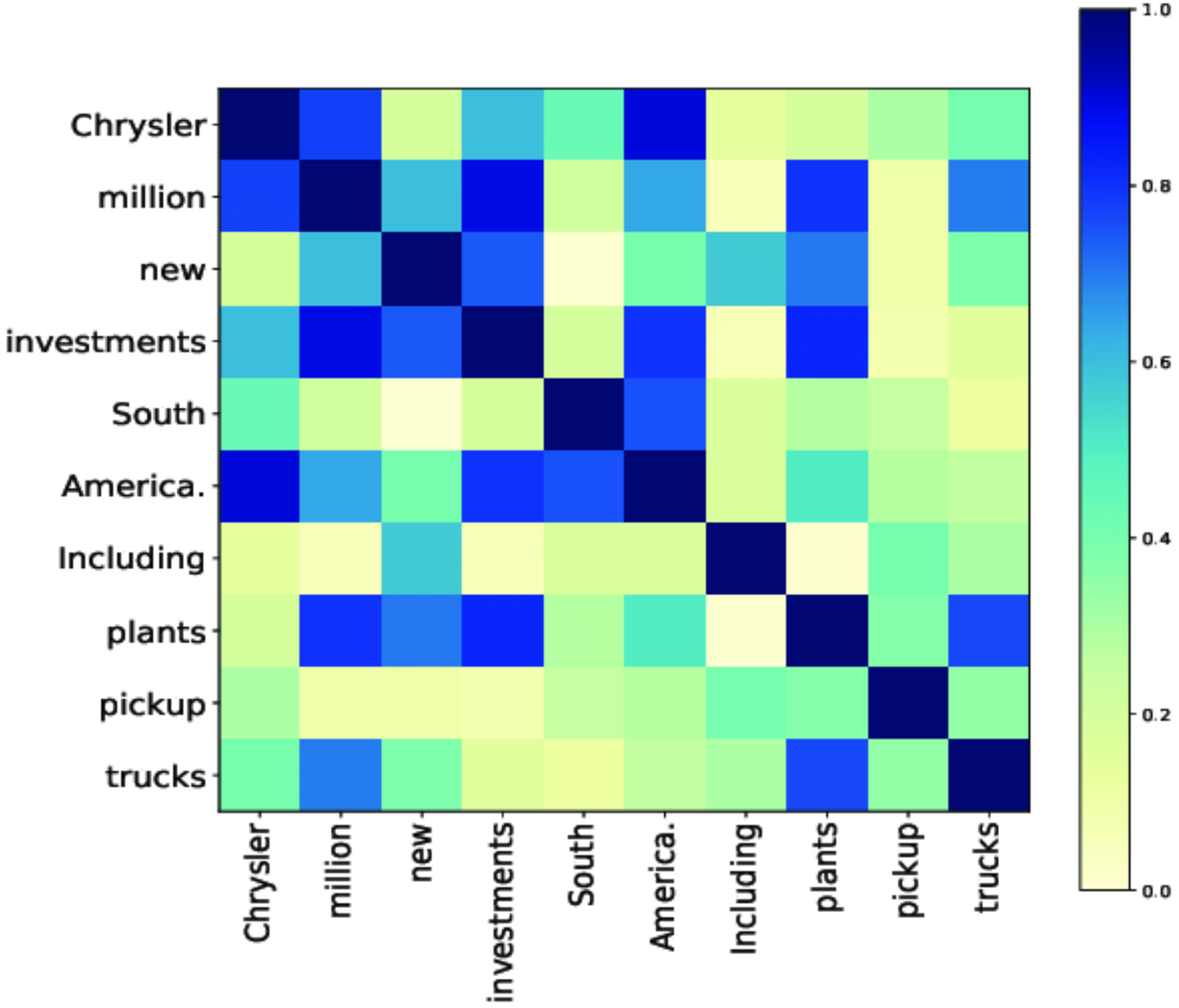
Softmax is taken over rows

Then, A is a matrix where the i^{th} row represents how similar the i^{th} word is to all the other word

An example

$$A = \text{softmax}\left(XX^T\right)$$

Softmax is taken over rows

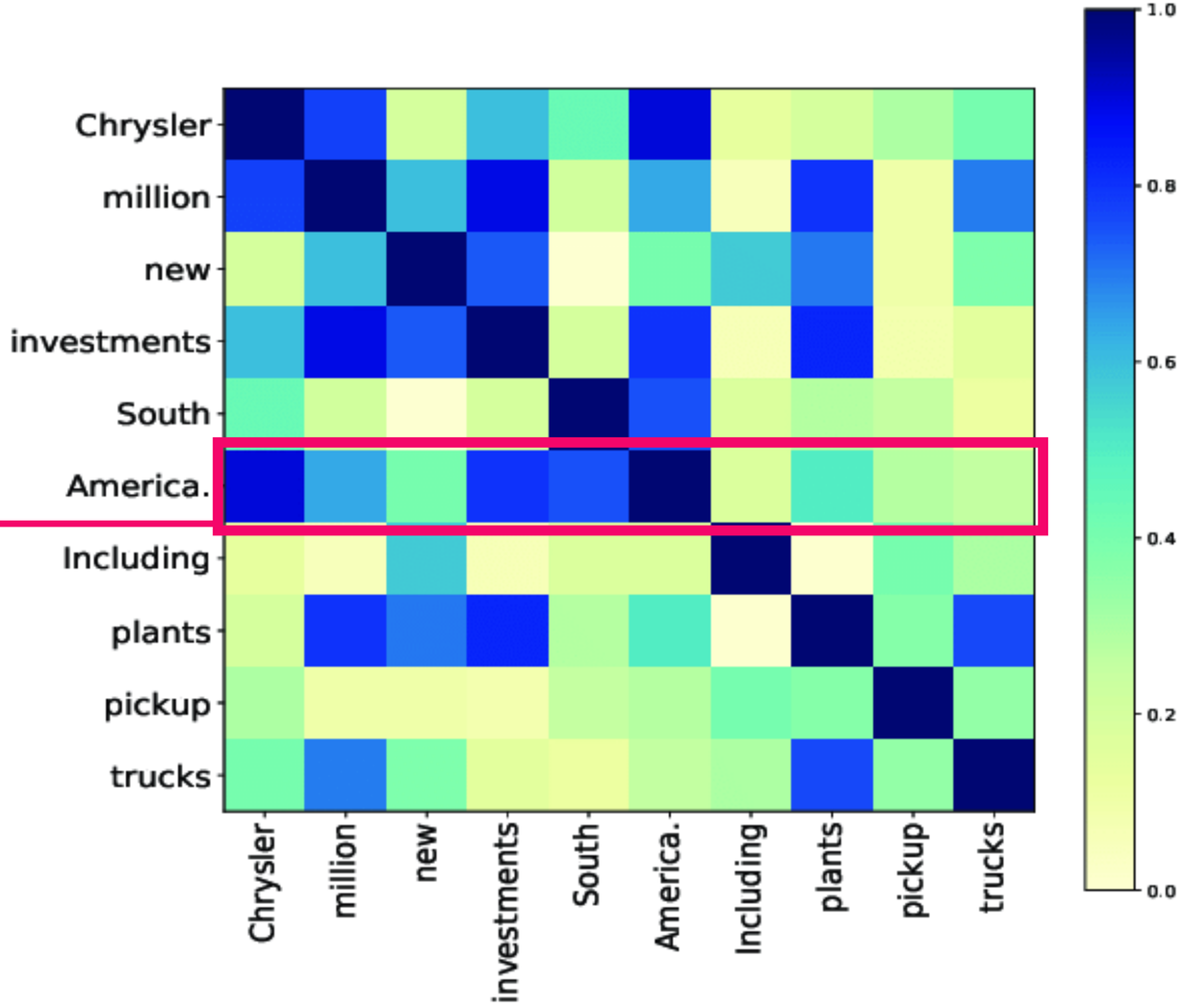


An example

$$A = \text{softmax}\left(XX^T\right)$$

Softmax is taken over rows

The word “America” and “Chrysler” have semantic similarities



The mixing

The mixing

Now that we know how similar two words are, how can we share their information

The mixing

Now that we know how similar two words are, how can we share their information

A naive idea is

$$Y = \text{softmax}\left(XX^T\right) X$$

The mixing

Now that we know how similar two words are, how can we share their information

A naive idea is

$$Y = \text{softmax}\left(XX^T\right) X$$

The i^{th} row is a combination of all the words earlier weighted by how similar they were

Where is the learning?

Where is the learning?

Add learnable weight matrices!

Where is the learning?

Add learnable weight matrices!

$$Y = \text{softmax}\Big((X \cdot W_Q)(X \cdot W_K)^T \Big)(X \cdot W_V)$$

Where is the learning?

Add learnable weight matrices!

$$Y = \text{softmax}\Big((X \cdot W_Q)(X \cdot W_K)^T \Big)(X \cdot W_V)$$

The model *learns* how to *attend* to the embeddings

Finally

Finally

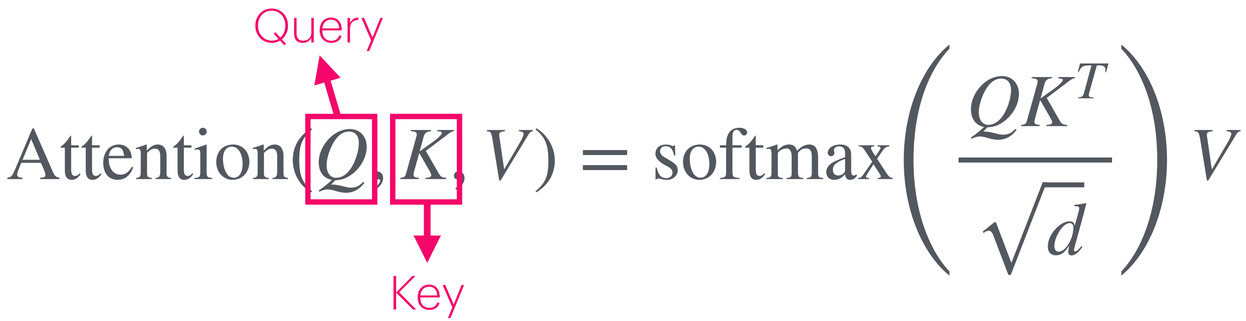
$$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$$

Finally

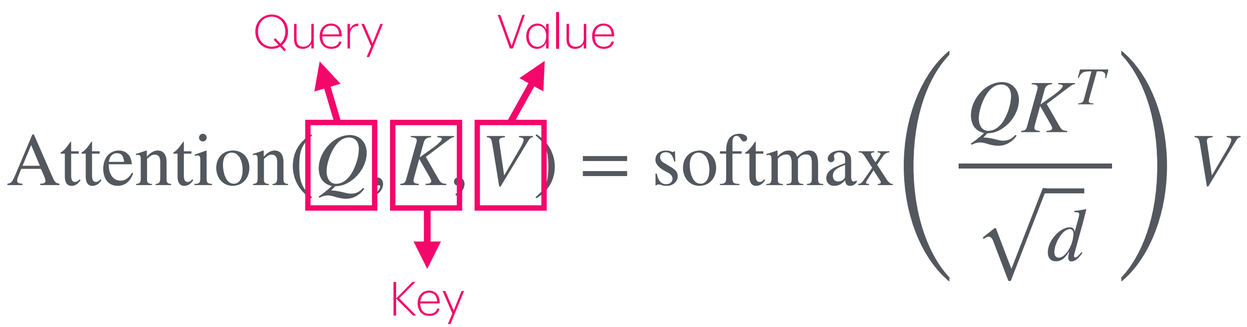
Query

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

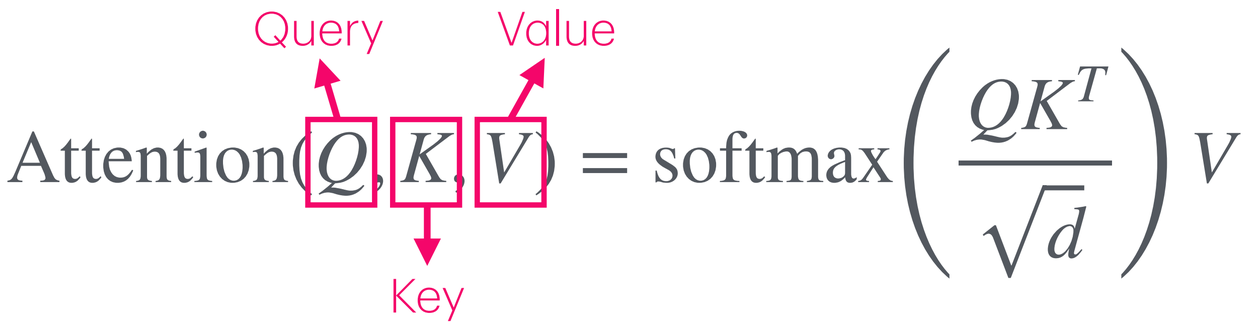
Finally


Query
Key
$\text{Attention}(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d}}\right)V$

Finally

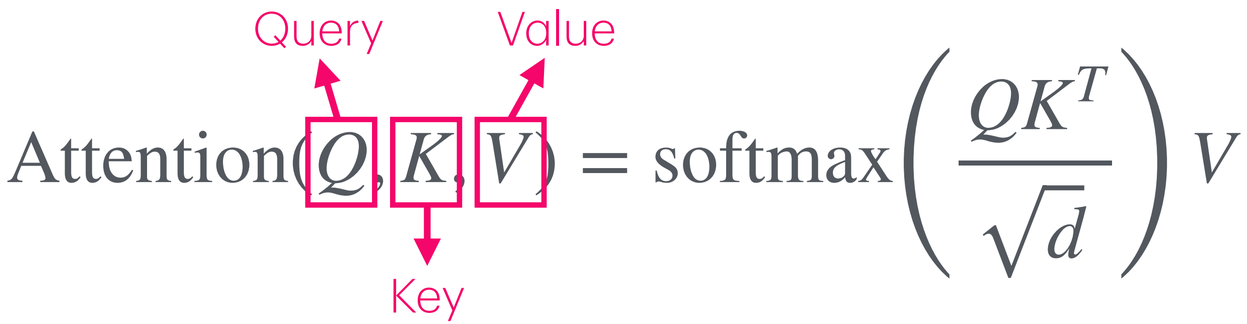


Finally



So the query Q decides how imporant keys K are and the values V are weighed accordingly

Finally



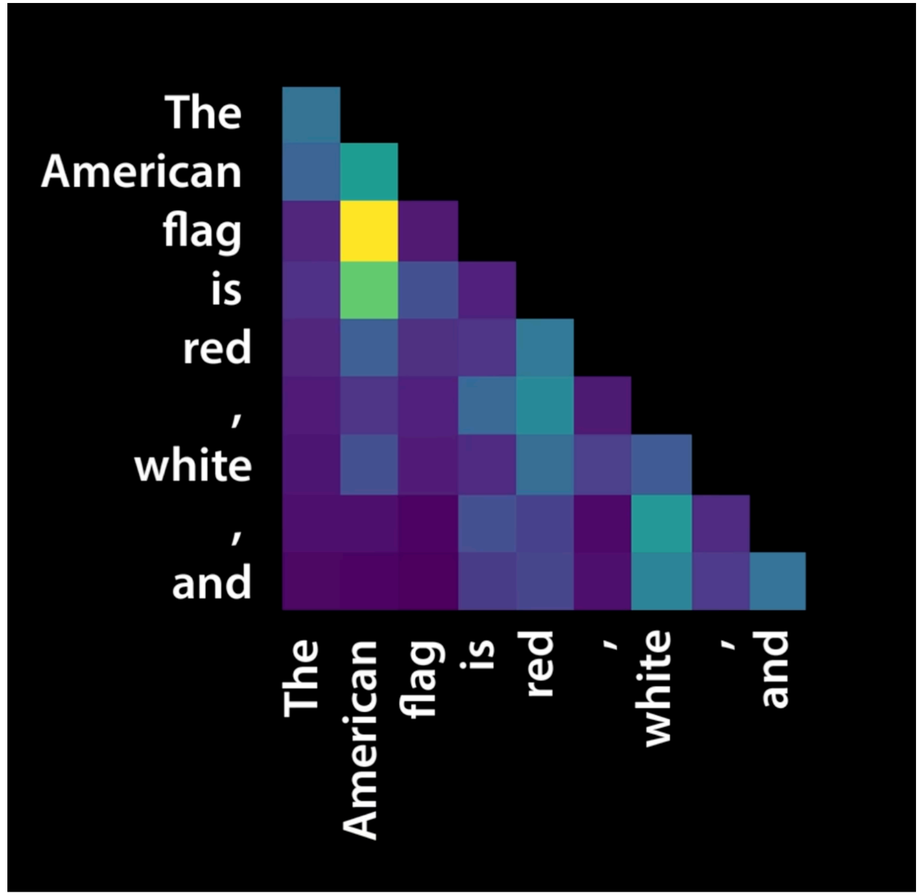
So the query Q decides how imporant keys K are and the values V are weighed accordingly

$$Y = \text{Attention}(XW_Q, XW_K, XW_V)$$

Y is the evolved matrix!

An Example

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

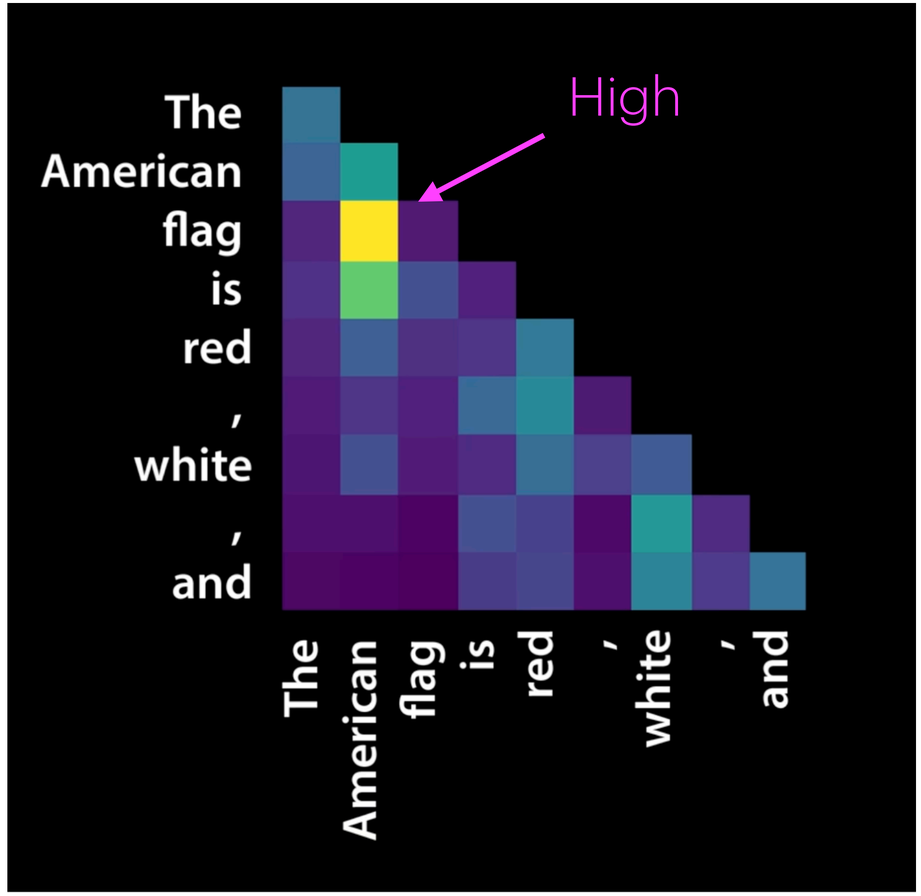


So the model learns that *American* and *flag* are highly related

It can use this to then understand the next word better

An Example

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



So the model learns that *American* and *flag* are highly related

It can use this to then understand the next word better

Where does this fail?

Where does this fail?

We use 3 matrices to capture meaning between tokens

Where does this fail?

We use 3 matrices to capture meaning between tokens

Is this enough?

Where does this fail?

We use 3 matrices to capture meaning between tokens

Is this enough?

Can we break down meaning into different parts?

Where does this fail?

Where does this fail?

Can we break down meaning into different parts?

Where does this fail?

Can we break down meaning into different parts?

One bit learns that American is related to flag

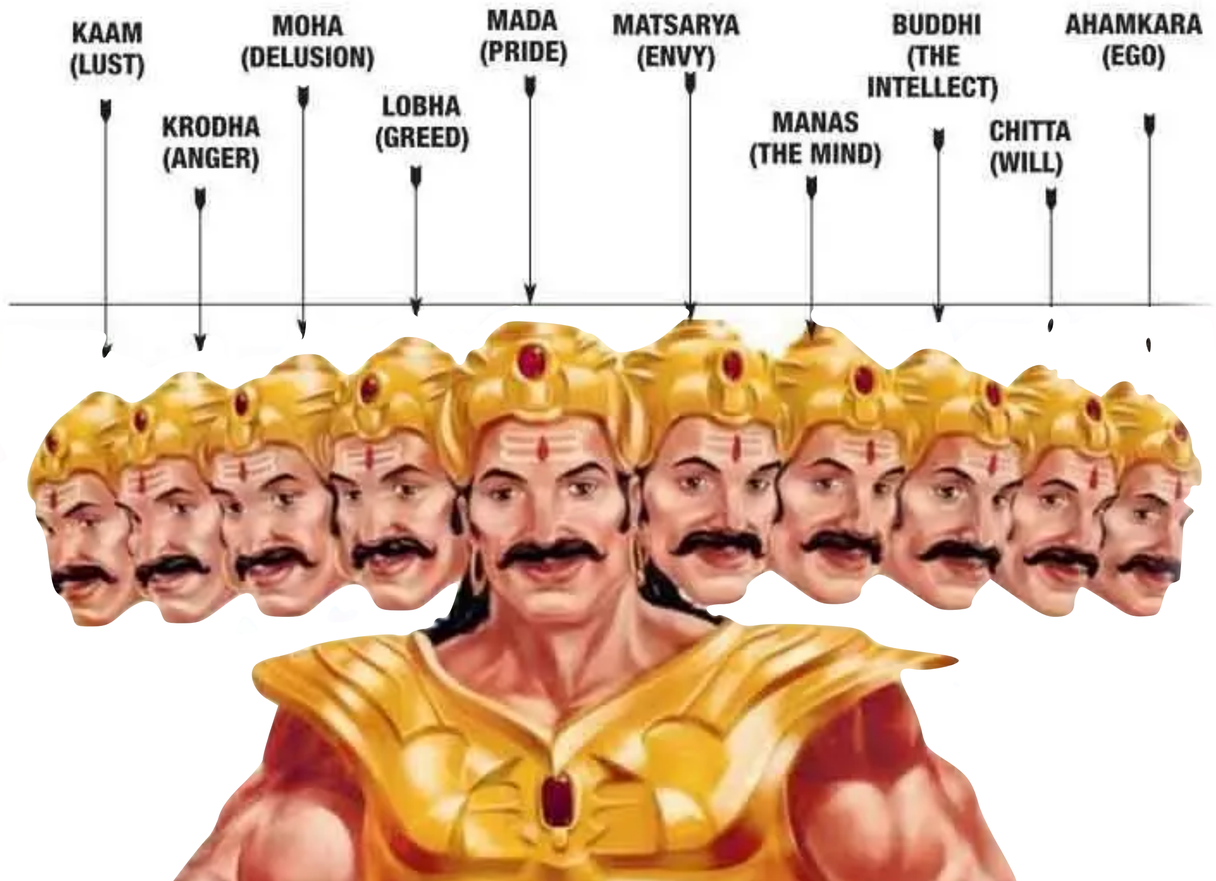
Where does this fail?

Can we break down meaning into different parts?

One bit learns that American is related to flag

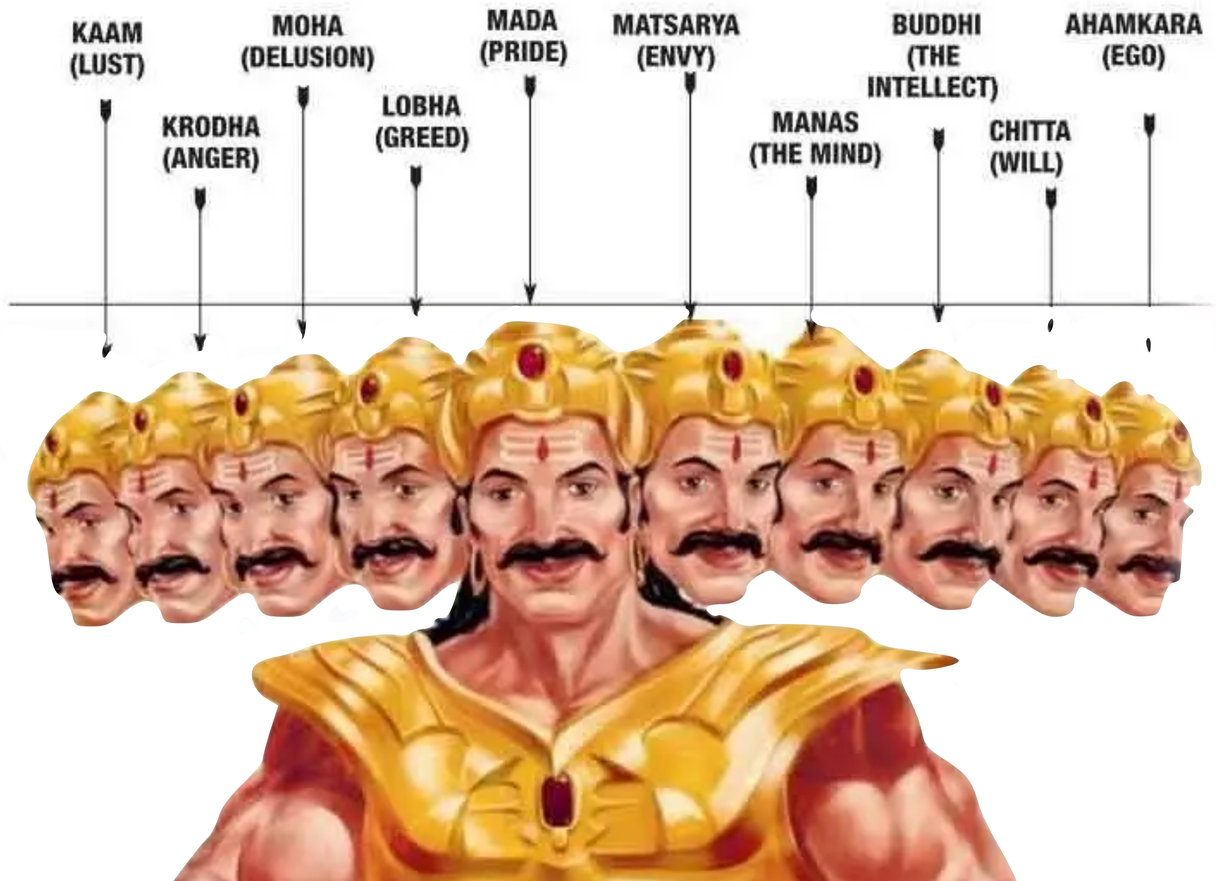
The other bit learns that blue, red, and white are related

Ravana


KAAM (LUST)
KRODHA (ANGER)
MOHA (DELUSION)
LOBHA (GREED)
MADA (PRIDE)
MATSARYA (ENVY)
MANAS (THE MIND)
BUDDHI (THE INTELLECT)
CHITTA (WILL)
AHAMKARA (EGO)

Ravana

India did it first!



Multi Head Attention

Instead of using a single attention head, use many!

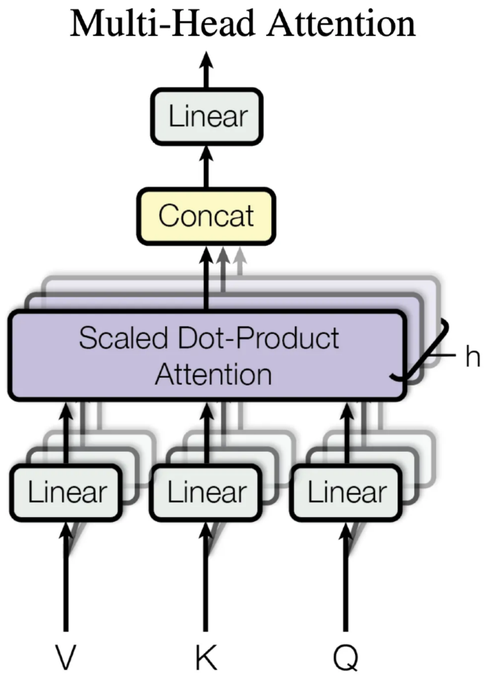
Multi Head Attention

Instead of using a single attention head, use many!

$$X \rightarrow \begin{bmatrix} \text{Attention}(XW_{Q1}, XW_{K1}, XW_{V1}) \\ \text{Attention}(XW_{Q2}, XW_{K2}, XW_{V2}) \\ \vdots \\ \text{Attention}(XW_{Qh}, XW_{Kh}, XW_{Vh}) \end{bmatrix} \rightarrow \begin{bmatrix} Y_1 \\ Y_2 \\ \vdots \\ Y_h \end{bmatrix} = Y \rightarrow YW_O$$

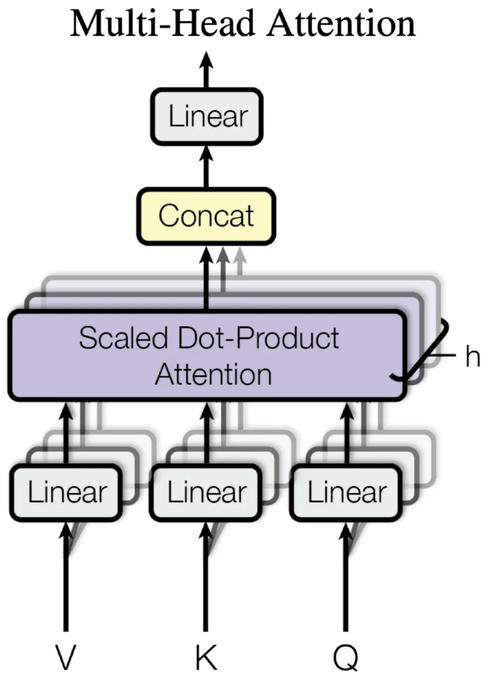
Multi Head Attention

Instead of using a single attention head, use many!



Multi Head Attention

Instead of using a single attention head, use many!



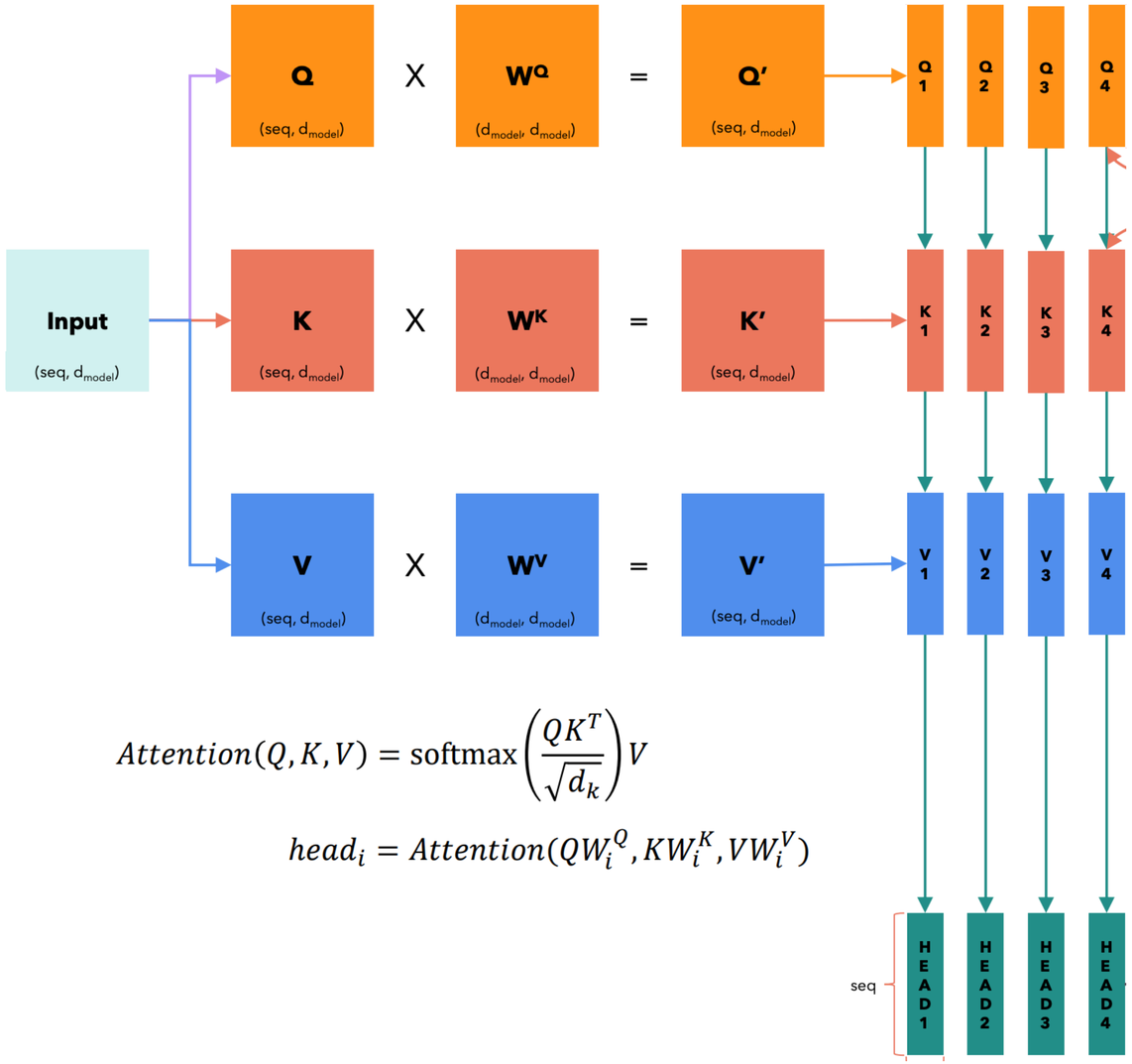
Each head outputs a smaller dimension now

They are concatenated to create the entire output Y

Multi Head Attention

But what are we doing?

Multi Head Attention



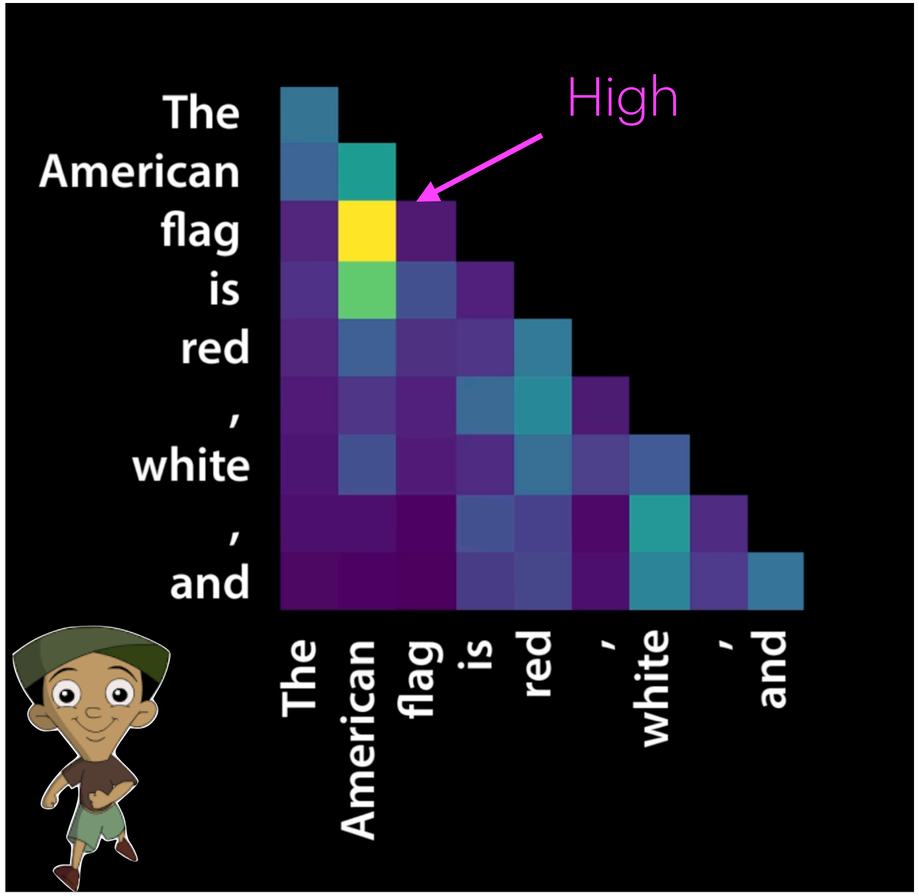
$$Attention(Q, K, V) = \text{softmax}\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

$$head_i = Attention(QW_i^Q, KW_i^K, VW_i^V)$$

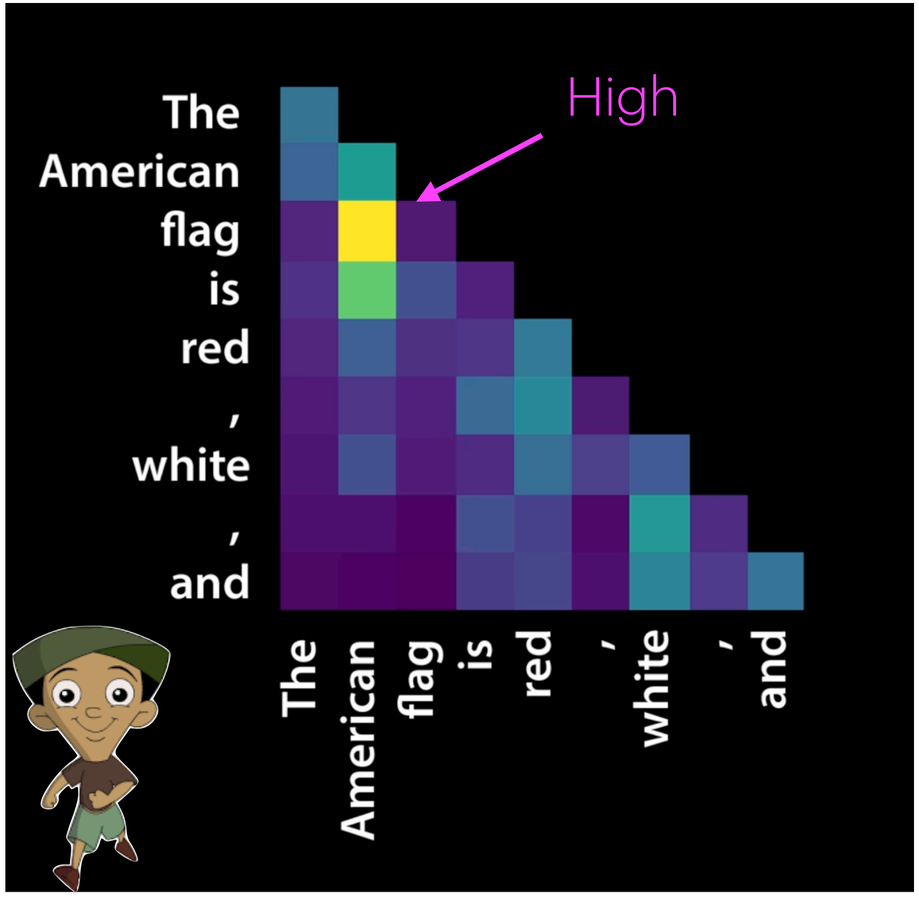
So the size of our K/V/Q matrices remain the same!

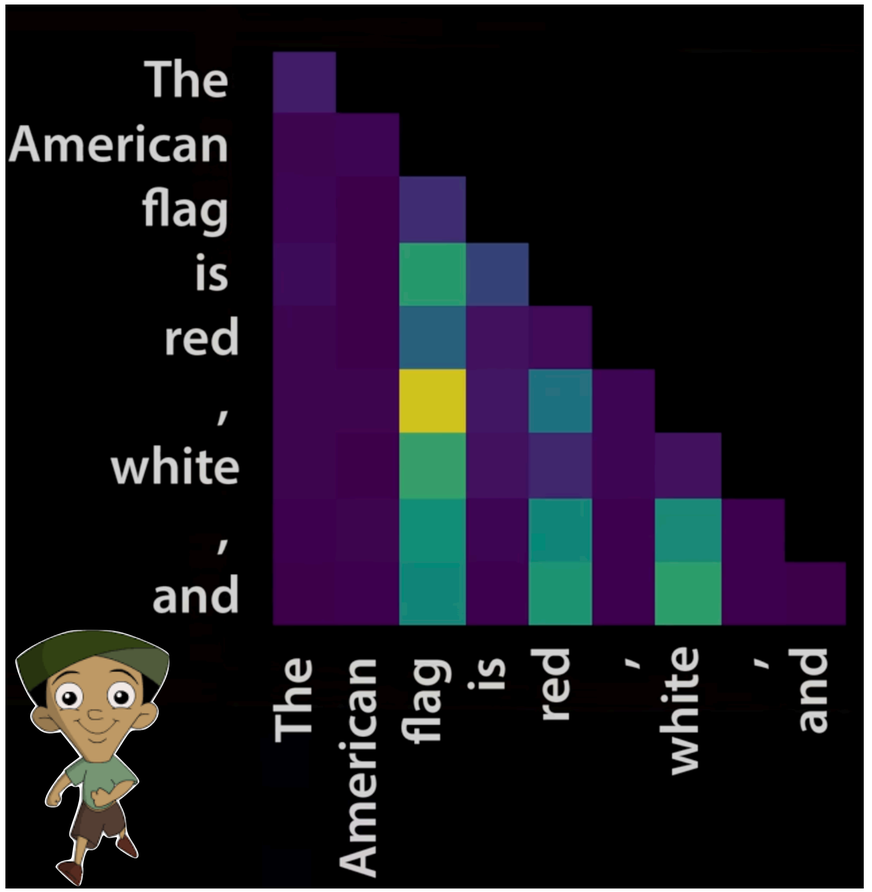
Each head attends to a different dimension of the projected space

MHA: An Example

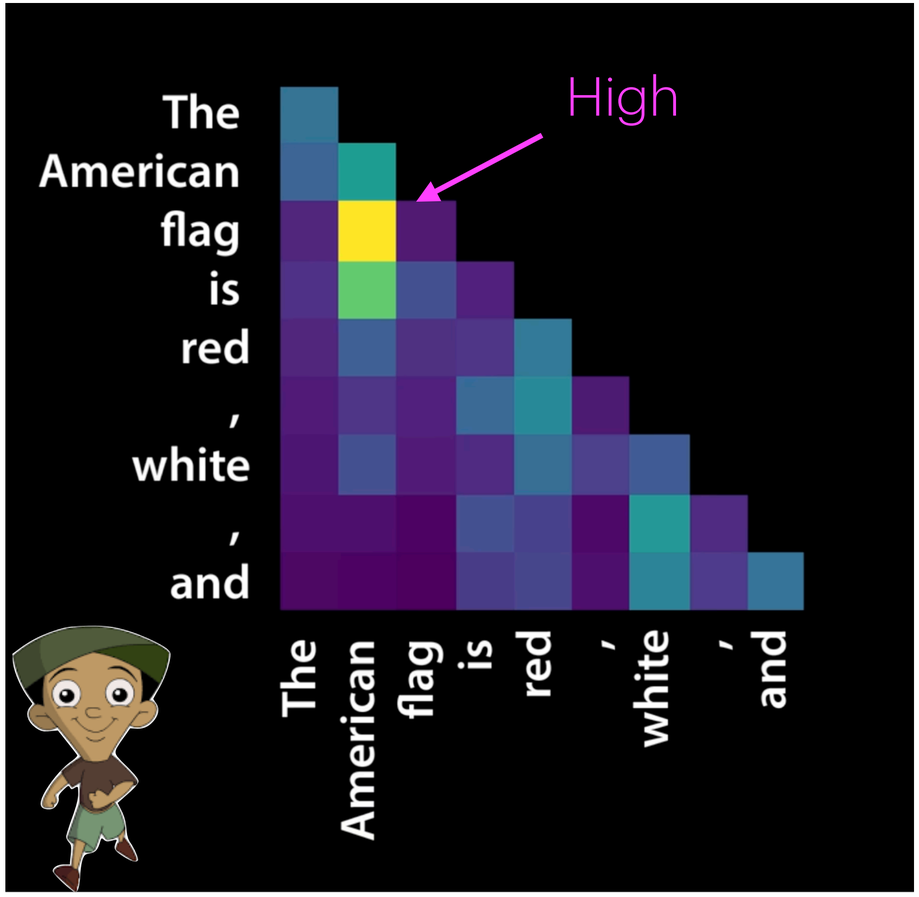


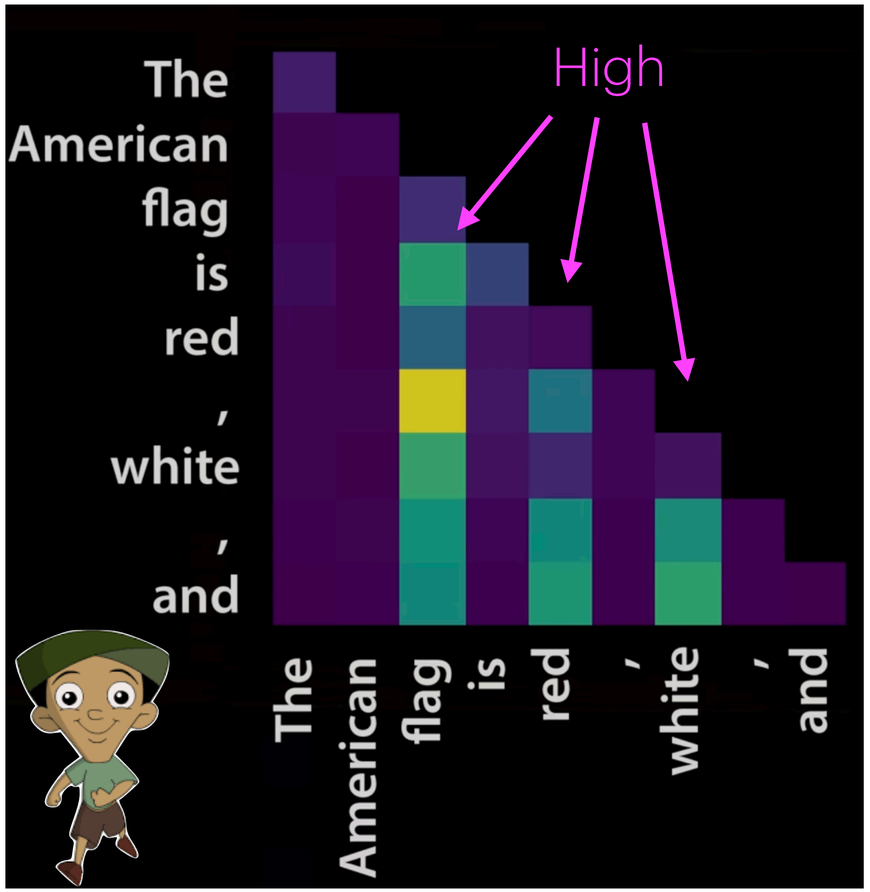
MHA: An Example





MHA: An Example





Zooming out

Zooming out

Let’s get back to the next word prediction problem

Zooming out

Let’s get back to the next word prediction problem

Attention fails. Each token attends to every other token, even the *future*

How would inference work?

What can we do now?

What can we do now?

We want to somehow preserve the causal nature of next token prediction task

What can we do now?

We want to somehow preserve the causal nature of next token prediction task

Can we prevent tokens from attending to tokens that come after it?

Workaround

Workaround

A way to do this would be to give 0 weight to future tokens

Workaround

A way to do this would be to give 0 weight to future tokens

That would be the same as ensuring that all entries above the diagonal in $\mathbf{softmax}\left(QK^T\right)$ are $\mathbf{0}$

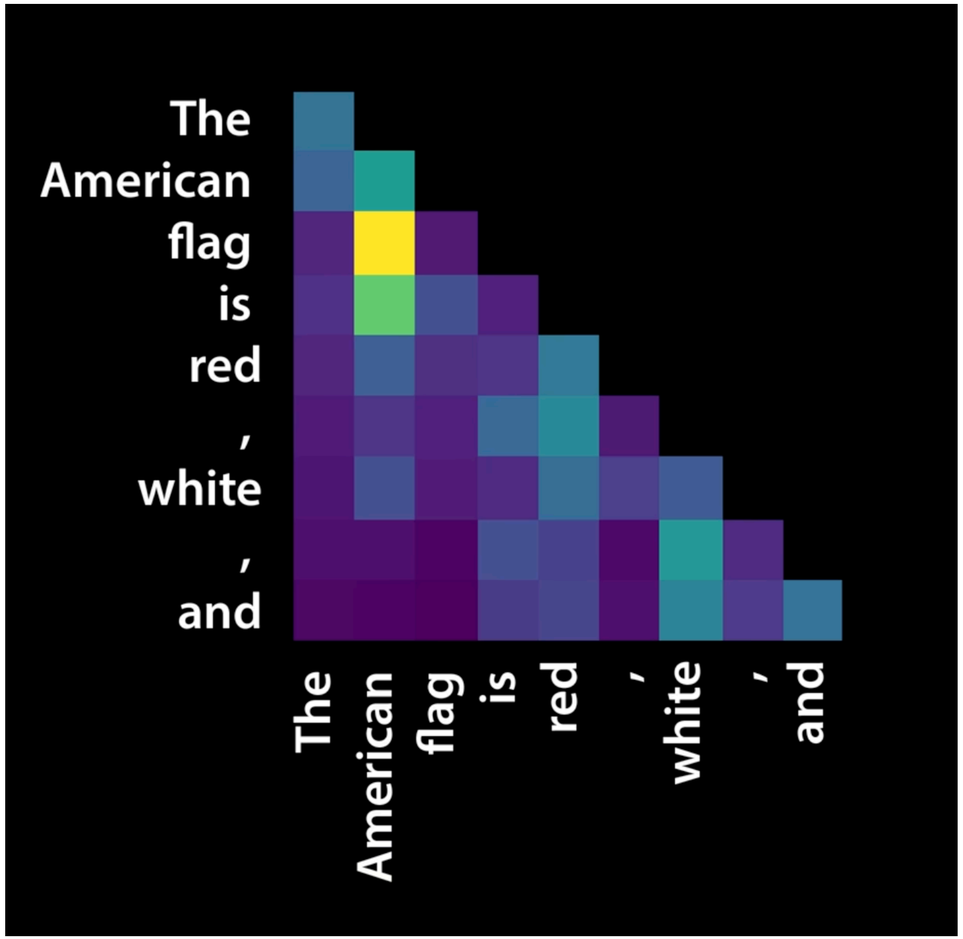
Workaround

A way to do this would be to give 0 weight to future tokens

That would be the same as ensuring that all entries above the diagonal in $\mathbf{softmax}\left(QK^T\right)$ are 0

So we want all entries above the diagonal in QK^T to be $-\infty$

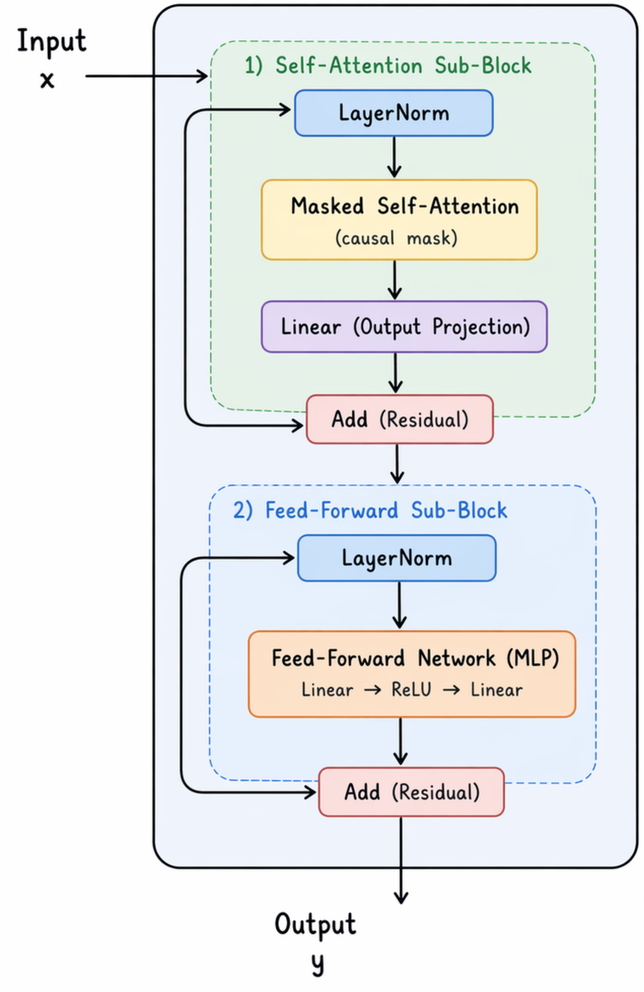
What this looks like



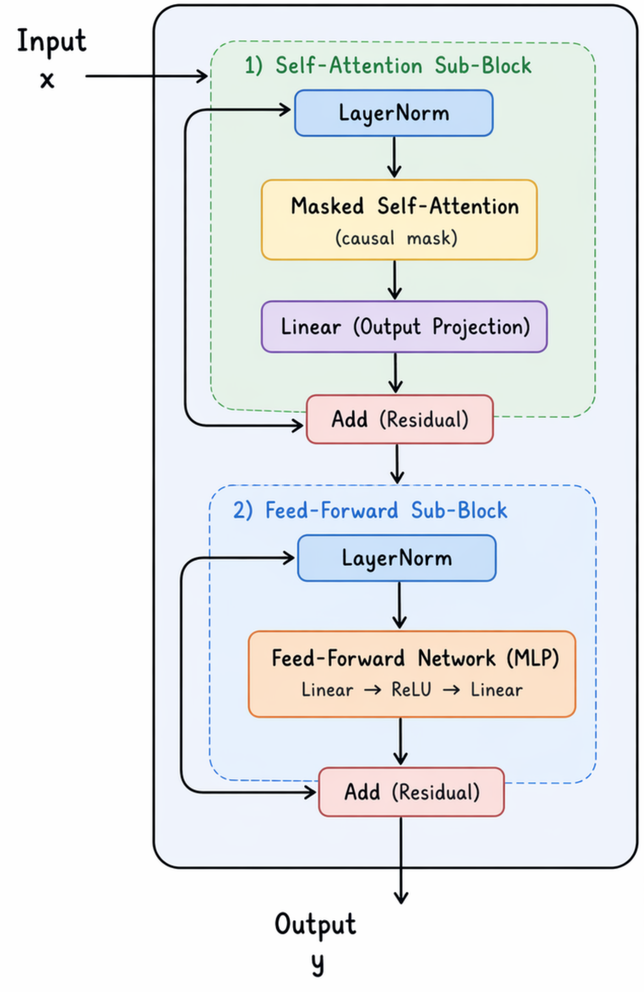
The same example as before!

See how we have a lower triangular matrix

A Transformer Decoder Block

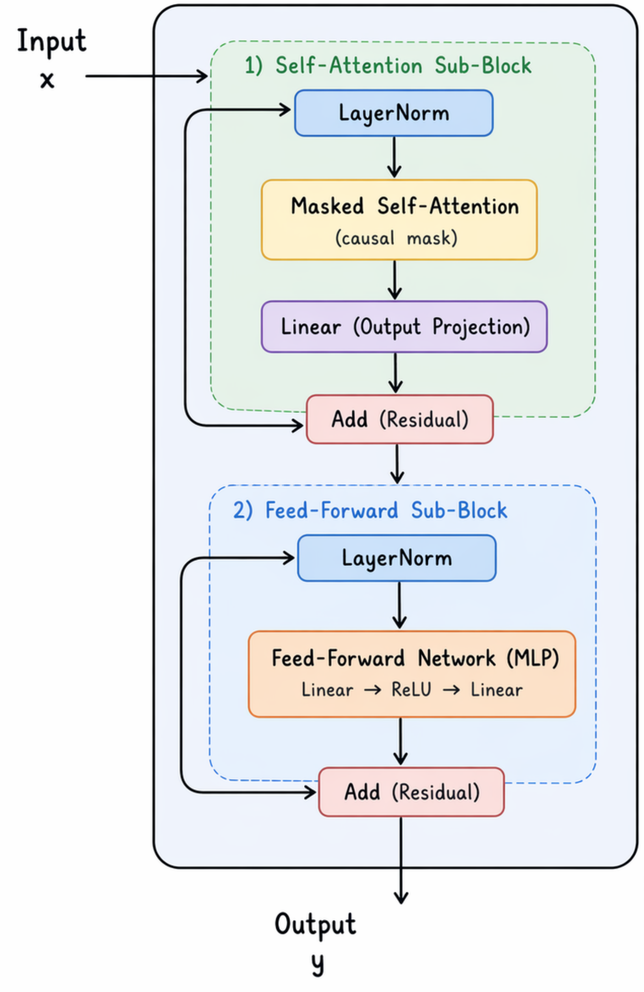


A Transformer Decoder Block



One block *transforms* the data once

A Transformer Decoder Block



One block *transforms* the data once

We stack many of these blocks together to get the overall model

A hidden optimisation

A hidden optimisation

Our dataset has sentences $s = w_1 w_2 \ldots w_n$

A hidden optimisation

Our dataset has sentences $s = w_1 w_2 \ldots w_n$

Look at what $f_\theta(s)$ looks like

A hidden optimisation

Our dataset has sentences $s = w_1 w_2 \ldots w_n$

Look at what $f_\theta(s)$ looks like

$$f_\theta(s) = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix}$$

A hidden optimisation

Our dataset has sentences $s = w_1 w_2 \ldots w_n$

Look at what $f_\theta(s)$ looks like

$$f_\theta(s) = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \leftarrow \text{we output this}$$

A hidden optimisation

Our dataset has sentences $s = w_1 w_2 \ldots w_n$

Look at what $f_\theta(s)$ looks like

$$f_\theta(s) = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \begin{matrix} \leftarrow \text{what about this?} \\ \\ \\ \leftarrow \text{we output this} \end{matrix}$$

A hidden optimisation

Our dataset has sentences $s = w_1 w_2 \ldots w_n$

Look at what $f_\theta(s)$ looks like

$$f_\theta(s) = \begin{bmatrix} E_1 \\ E_2 \\ \vdots \\ E_n \end{bmatrix} \begin{matrix} \\ \\ \\ \leftarrow \text{we output this} \end{matrix}$$

$\leftarrow$ what about this?

Backprop for all n at once!

Some things we did not cover

Some things we did not cover

What the first paper used transformers for - translation

Some things we did not cover

What the first paper used transformers for - translation

Cross attention - a way for two sequences to interact with each other (images with a prompt)

Some things we did not cover

How to learn the embeddings + positional embeddings

What the first paper used transformers for - translation

Cross attention - a way for two sequences to interact with each other (images with a prompt)

What the current LLMs have

What the current LLMs have

Mixture of Experts

What the current LLMs have

Mixture of Experts

Latent Attention

What the current LLMs have

Mixture of Experts

Latent Attention

Multimodality

What the current LLMs have

Mixture of Experts

Latent Attention

Multimodality

Bleeding Edge

Engram

Manifold-Constrained Hyper-Connections, AttnRes