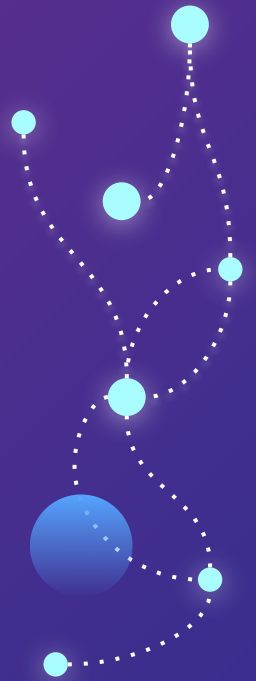
Large Language Models

TABLE OF CONTENTS

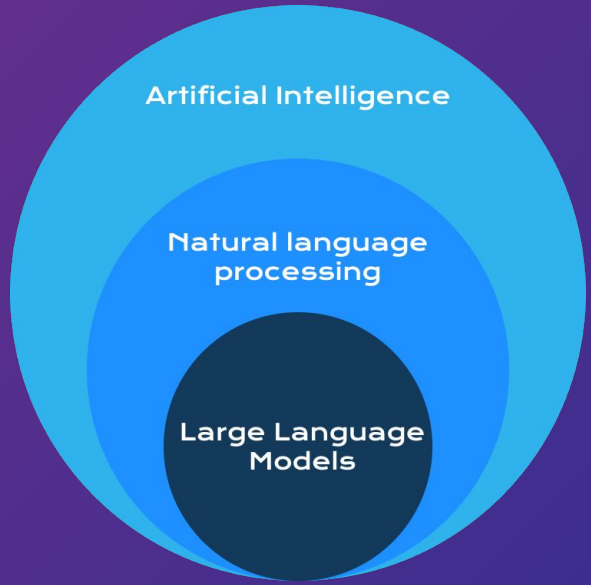


01

LLMs & AI

How does LLMs relate to AI?

- Artificial intelligence - simulating human abilities
- Natural Language Processing (NLP)
  - Computational linguistics
  - Machine learning
- Large Language Model (LLM)
  - Focused on understanding, interpreting and generating human-like language



Why do we care about LLMs?

Conversational agents

Personalized learning

Content creation

Find bugs in code

Language preservation

Translate text

Universal design

Automated customer service

Help with formulation

Daily productivity

Create summaries

NPC dialogue in video games

Analyze text

“Large” Language Model

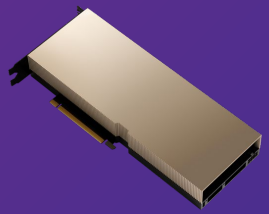
1.760.000.000.000

- estimated number of parameters in GPT-4

Trained on 25.000 NVIDIA A100 GPUs

Trained for 90-100 days

Estimated cost of training: $60-100 million



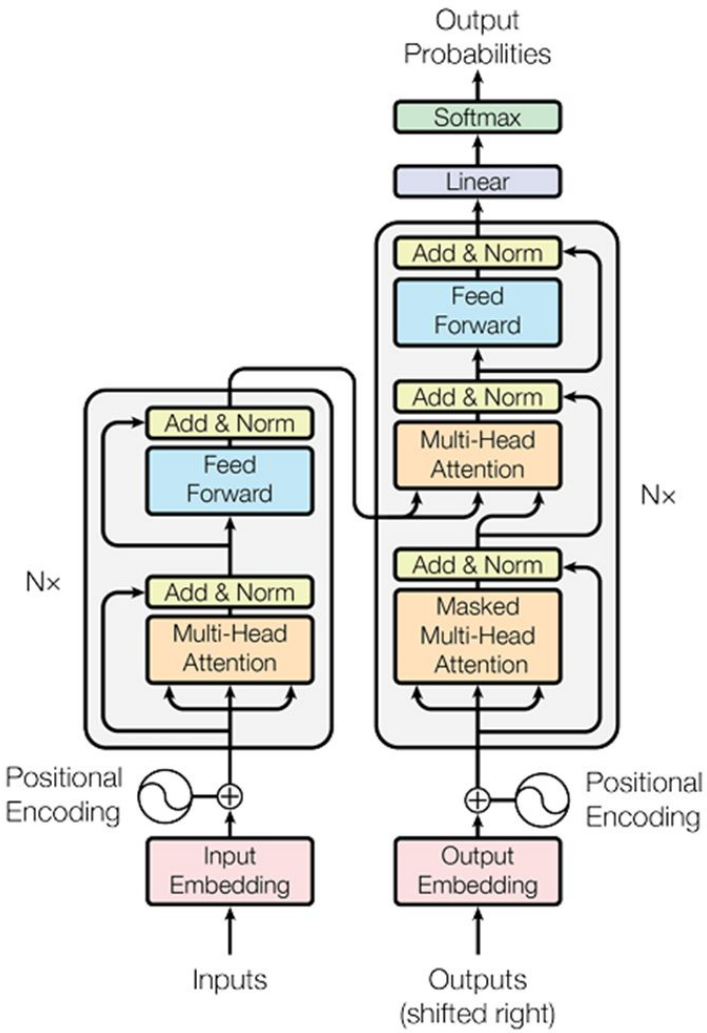
~ $12k each

Training GPT-3 with 175 billion parameters emitted 552 tonnes of CO_2
= Yearly emissions of 123 gasoline cars

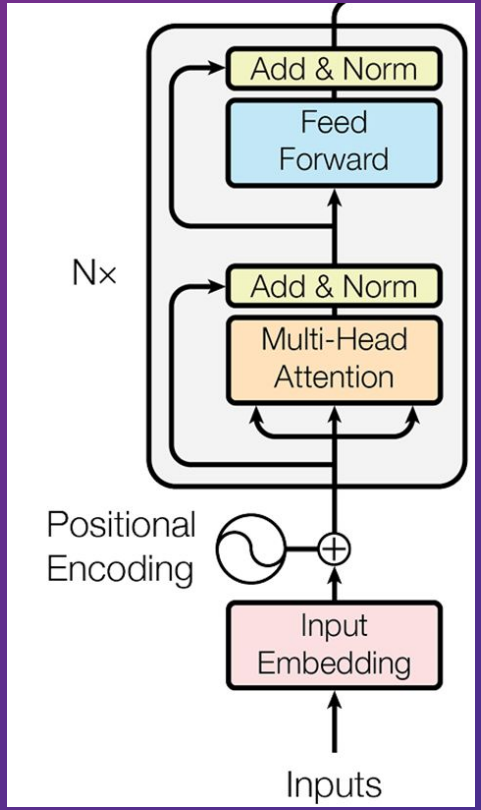
(Source: https://arxiv.org/ftp/arxiv/papers/2104/2104.10350.pdf)

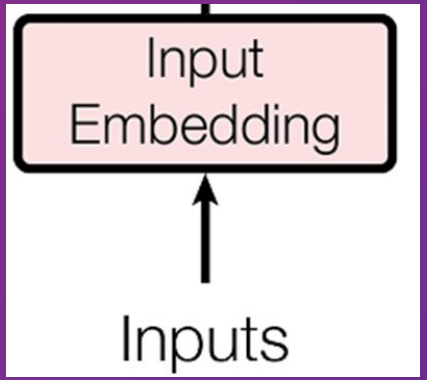
02

Transformers


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Encoder


Add & Norm
Feed Forward
Nx
Add & Norm
Multi-Head Attention
Positional Encoding
Input Embedding
Inputs

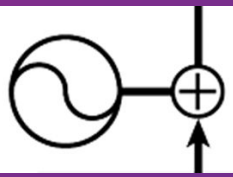


I am a supersimple sample textr

«I am», «a», «super», «simple», «sample», «text», «r»

$$\begin{bmatrix}1.36\\0.78\\\vdots\\0.34\end{bmatrix}, \begin{bmatrix}8.54\\3.33\\\vdots\\1.72\end{bmatrix}, \begin{bmatrix}4.44\\5.55\\\vdots\\6.66\end{bmatrix}, \begin{bmatrix}1\\1\\\vdots\\2\end{bmatrix}, \begin{bmatrix}3\\5\\\vdots\\8\end{bmatrix}, \begin{bmatrix}0.5\\0.25\\\vdots\\0.125\end{bmatrix}, \begin{bmatrix}-1.36\\-0.78\\\vdots\\-0.34\end{bmatrix}$$

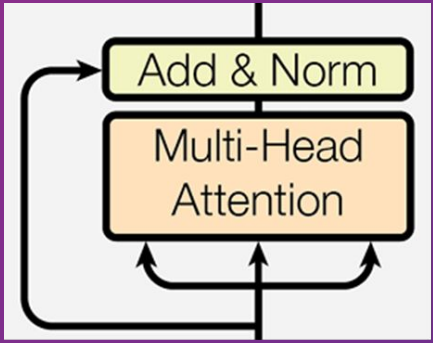
Positional Encoding

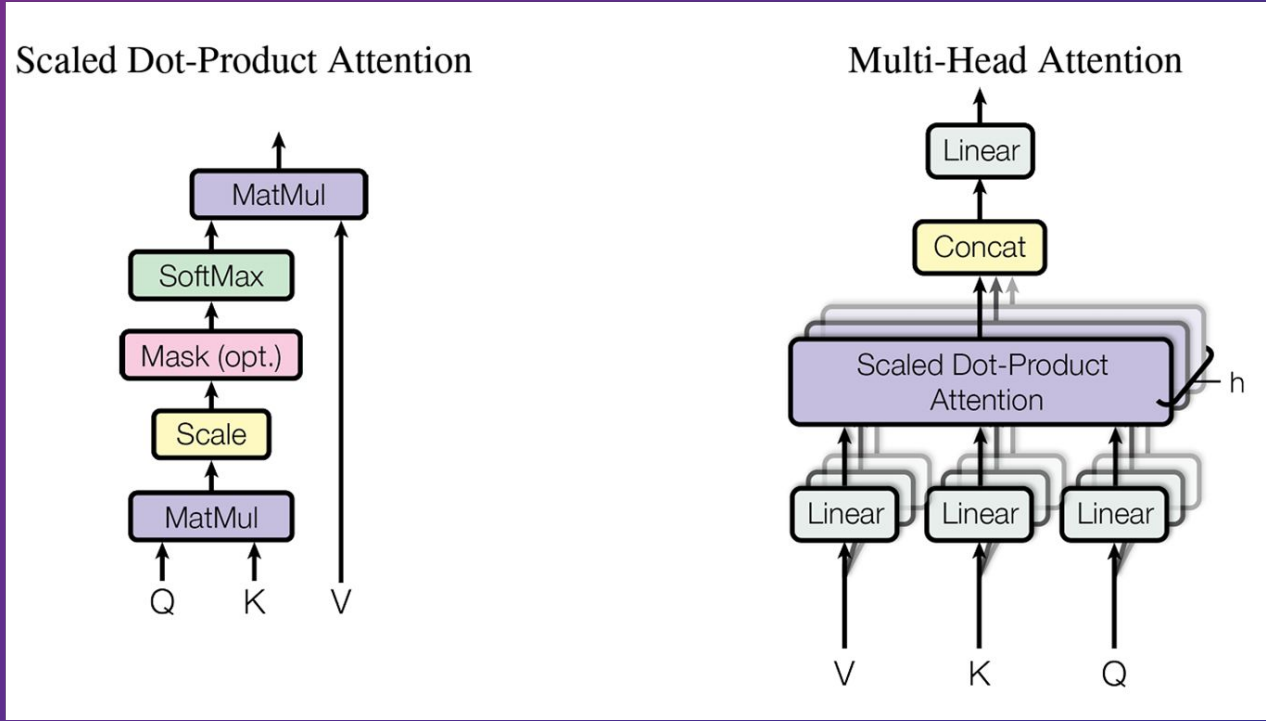


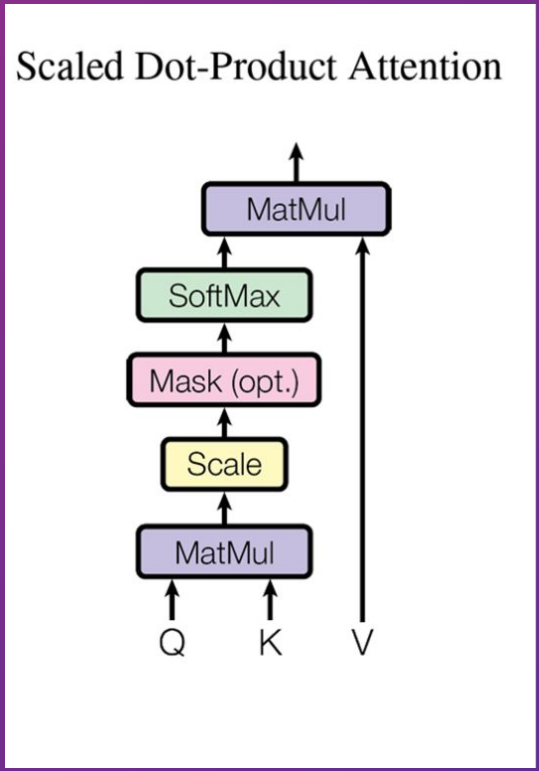
$$PE_{(pos,2i)} = sin(pos/10000^{2i/d_{\text{model}}})$$
$$PE_{(pos,2i+1)} = cos(pos/10000^{2i/d_{\text{model}}})$$

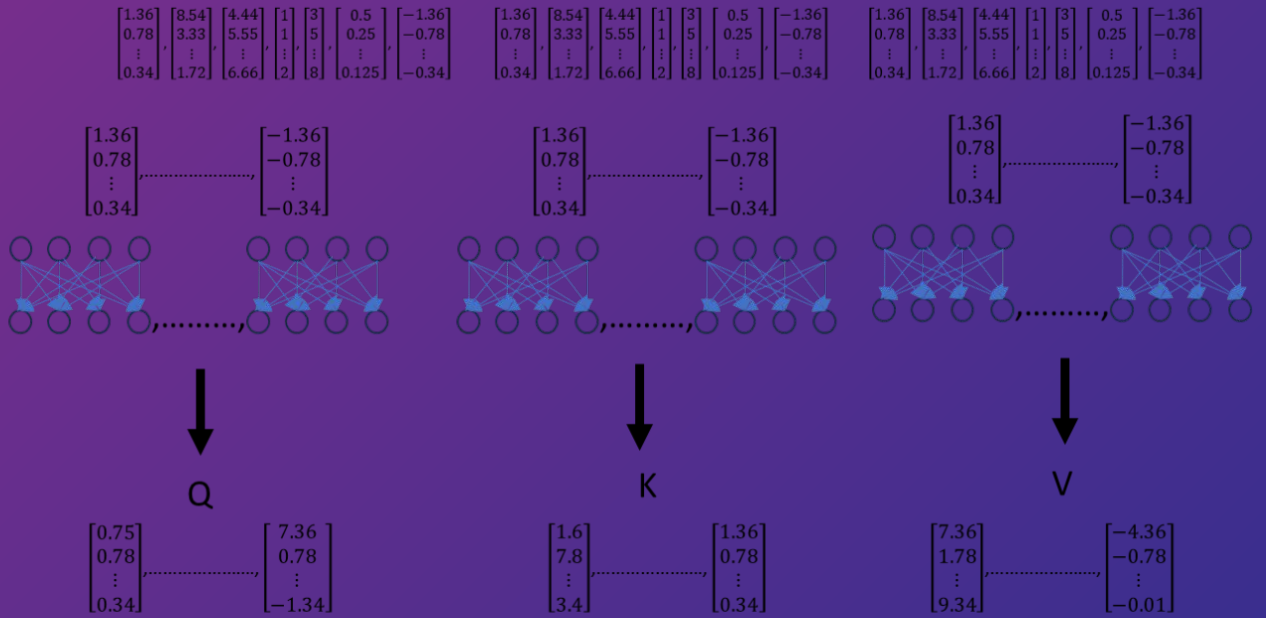
$$\begin{bmatrix}0\\1\\0\\1\end{bmatrix}, \begin{bmatrix}0.84\\0.98\\0.03\\0.99\end{bmatrix}, \begin{bmatrix}0.91\\0.93\\0.06\\0.99\end{bmatrix}, \begin{bmatrix}0.14\\0.86\\0.09\\0.99\end{bmatrix}, \begin{bmatrix}-0.75\\0.76\\0.13\\0.99\end{bmatrix}, \begin{bmatrix}-0.96\\0.25\\0.15\\0.98\end{bmatrix}, \begin{bmatrix}-0.33\\-0.64\\0.18\\0.98\end{bmatrix}$$

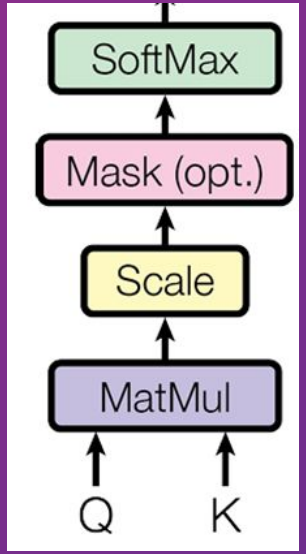
$$\begin{bmatrix}1.36\\0.78\\\vdots\\0.34\end{bmatrix}, \begin{bmatrix}8.54\\3.33\\\vdots\\1.72\end{bmatrix}, \begin{bmatrix}4.44\\5.55\\\vdots\\6.66\end{bmatrix}, \begin{bmatrix}1\\1\\\vdots\\2\end{bmatrix}, \begin{bmatrix}3\\5\\\vdots\\8\end{bmatrix}, \begin{bmatrix}0.5\\0.25\\\vdots\\0.125\end{bmatrix}, \begin{bmatrix}-1.36\\-0.78\\\vdots\\-0.34\end{bmatrix}$$


Add & Norm
Multi-Head Attention


Scaled Dot-Product Attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V
Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q


Scaled Dot-Product Attention
MatMul
SoftMax
Mask (opt.)
Scale
MatMul
Q
K
V


Q
K
V



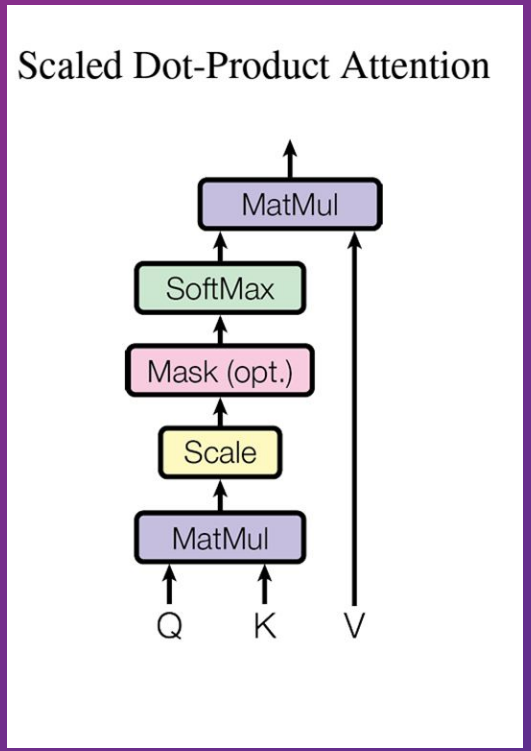
Q*K

$$\begin{bmatrix} 3.01 & \cdots & \cdots & 0.38 \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ 4.42 & \cdots & \cdots & 3.72 \end{bmatrix}$$

Softmax function

$$\begin{bmatrix} 0.78 & \cdots & \cdots & 0.22 \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ 0.58 & \cdots & \cdots & 0.42 \end{bmatrix}$$

Matrix size: NxN, where N is # tokens

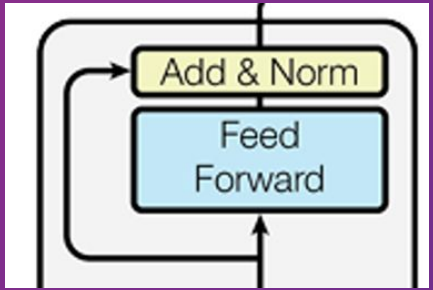


QK

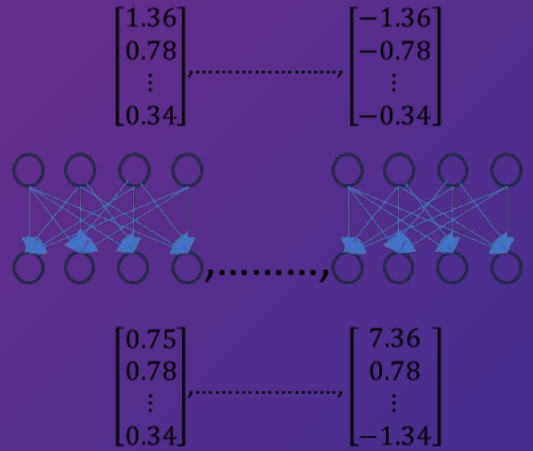
$$\begin{bmatrix} 0.78 & \cdots & \cdots & 0.22 \\ \vdots & \cdots & \cdots & \vdots \\ \vdots & \cdots & \cdots & \vdots \\ 0.58 & \cdots & \cdots & 0.42 \end{bmatrix} * \quad V \quad \begin{bmatrix} 7.36 \\ 1.78 \\ \vdots \\ 9.34 \end{bmatrix}, \cdots\cdots\cdots, \begin{bmatrix} -4.36 \\ -0.78 \\ \vdots \\ -0.01 \end{bmatrix}$$

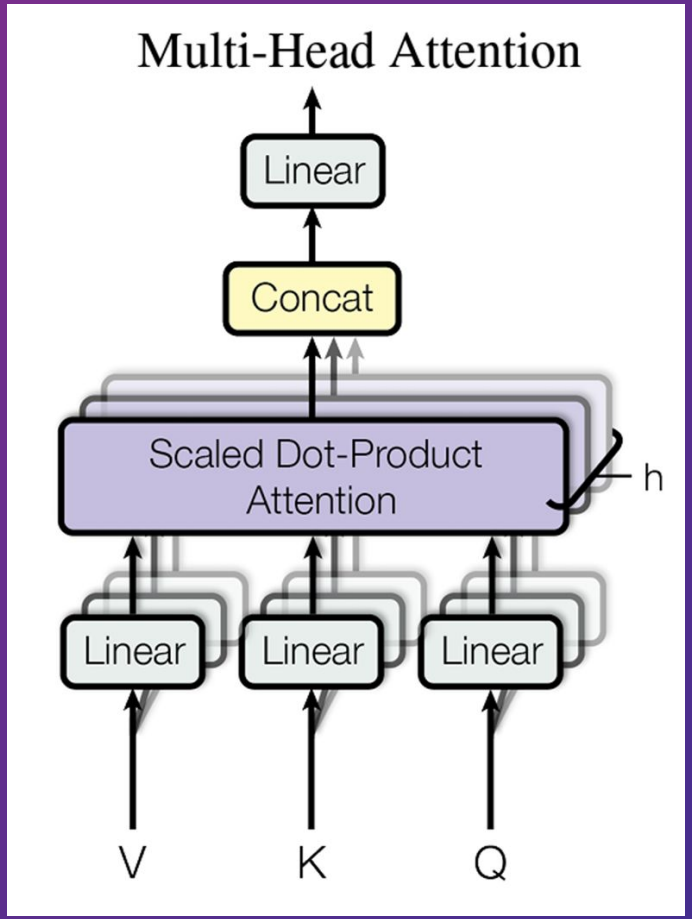
$$\downarrow$$

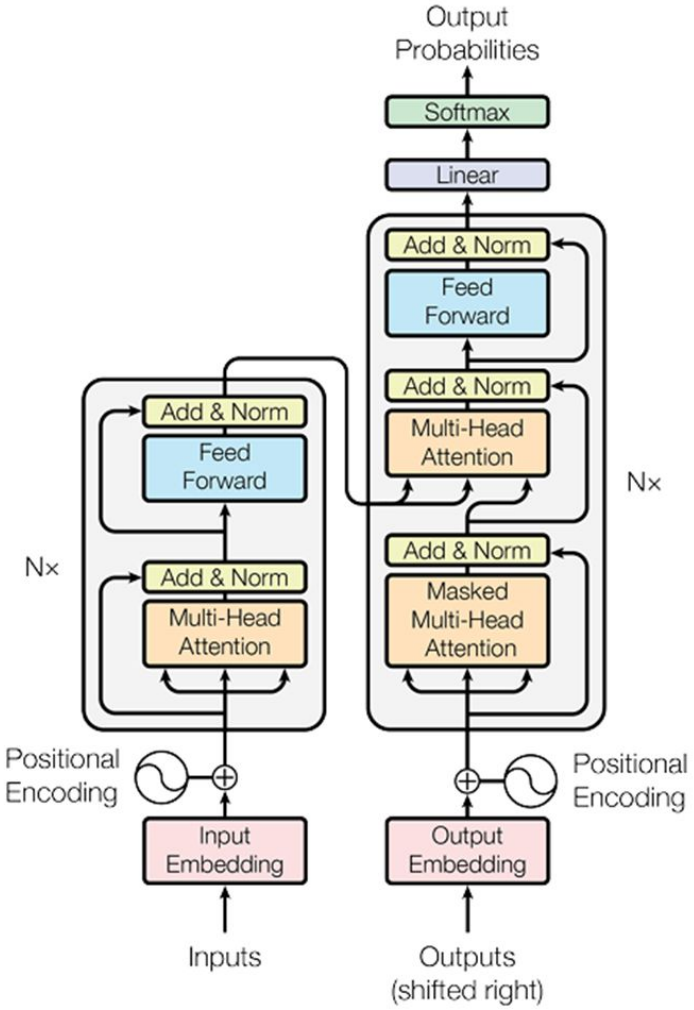
$$\begin{bmatrix} 1.36 \\ 0.78 \\ \vdots \\ 0.34 \end{bmatrix}, \begin{bmatrix} 8.54 \\ 3.33 \\ \vdots \\ 1.72 \end{bmatrix}, \begin{bmatrix} 4.44 \\ 5.55 \\ \vdots \\ 6.66 \end{bmatrix}, \begin{bmatrix} 1 \\ 1 \\ \vdots \\ 2 \end{bmatrix}, \begin{bmatrix} 3 \\ 5 \\ \vdots \\ 8 \end{bmatrix}, \begin{bmatrix} 0.5 \\ 0.25 \\ \vdots \\ 0.125 \end{bmatrix}, \begin{bmatrix} -1.36 \\ -0.78 \\ \vdots \\ -0.34 \end{bmatrix}$$



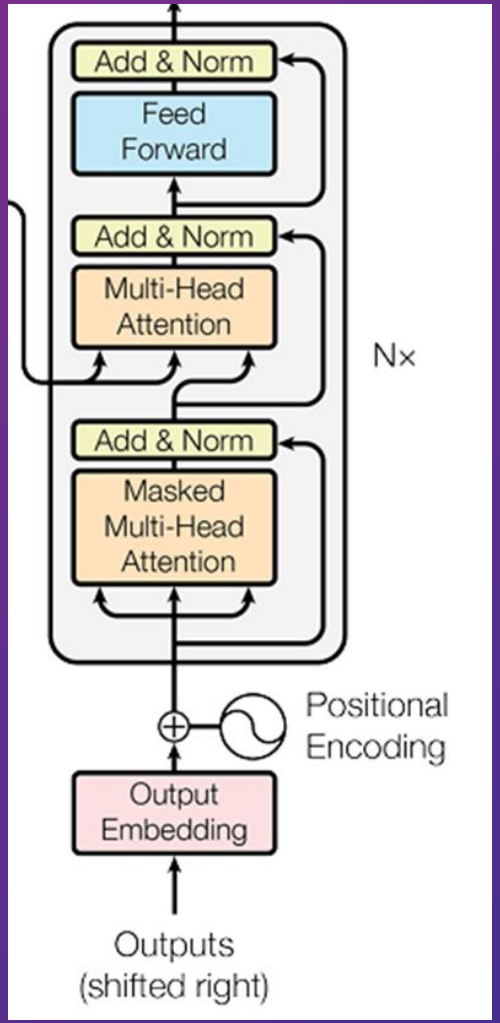
$$\begin{bmatrix}1.36\\0.78\\\vdots\\0.34\end{bmatrix},\begin{bmatrix}8.54\\3.33\\\vdots\\1.72\end{bmatrix},\begin{bmatrix}4.44\\5.55\\\vdots\\6.66\end{bmatrix},\begin{bmatrix}1\\1\\\vdots\\2\end{bmatrix},\begin{bmatrix}3\\5\\\vdots\\8\end{bmatrix},\begin{bmatrix}0.5\\0.25\\\vdots\\0.125\end{bmatrix},\begin{bmatrix}-1.36\\-0.78\\\vdots\\-0.34\end{bmatrix}$$

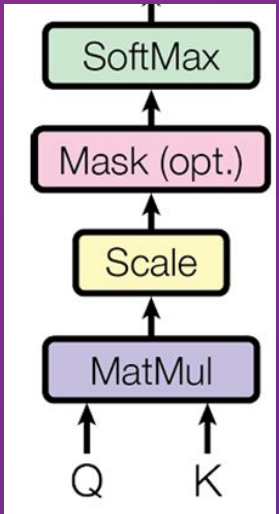


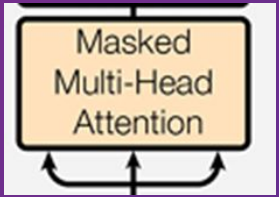

Multi-Head Attention
Linear
Concat
Scaled Dot-Product Attention
h
Linear
Linear
Linear
V
K
Q


Output Probabilities
Softmax
Linear
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Nx
Positional Encoding
Positional Encoding
Input Embedding
Output Embedding
Inputs
Outputs (shifted right)

Decoder


Add & Norm
Feed Forward
Add & Norm
Multi-Head Attention
Add & Norm
Masked Multi-Head Attention
Nx
Positional Encoding
Output Embedding
Outputs (shifted right)





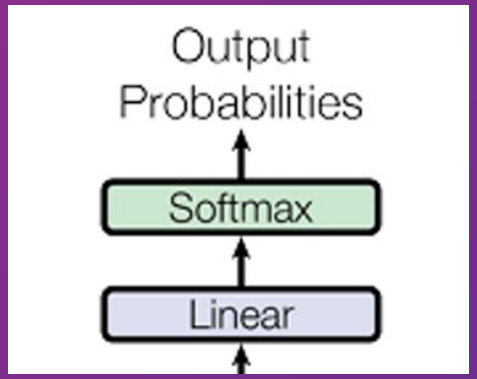
Q*K

$$\begin{bmatrix} 3.01 & -inf & -inf & -inf \\ \vdots & \cdots & -inf & -inf \\ \vdots & \cdots & \cdots & -inf \\ 4.42 & \cdots & \cdots & 3.72 \end{bmatrix}$$

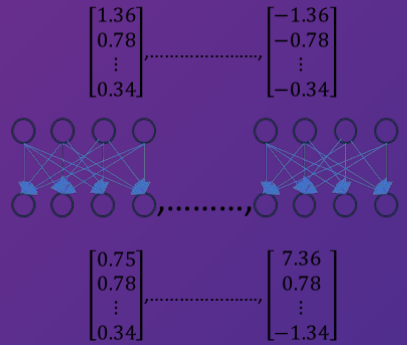
Softmax function

$$\begin{bmatrix} 1 & 0 & 0 & 0 \\ \vdots & \cdots & 0 & 0 \\ \vdots & \cdots & \cdots & 0 \\ 0.58 & \cdots & \cdots & 0.42 \end{bmatrix}$$

Matrix size: NxN, where N is # tokens



$$\begin{bmatrix}1.36\\0.78\\\vdots\\0.34\end{bmatrix},\begin{bmatrix}8.54\\3.33\\\vdots\\1.72\end{bmatrix},\begin{bmatrix}4.44\\5.55\\\vdots\\6.66\end{bmatrix},\begin{bmatrix}1\\1\\\vdots\\2\end{bmatrix},\begin{bmatrix}3\\5\\\vdots\\8\end{bmatrix},\begin{bmatrix}0.5\\0.25\\\vdots\\0.125\end{bmatrix},\begin{bmatrix}-1.36\\-0.78\\\vdots\\-0.34\end{bmatrix}$$



$$\begin{bmatrix}0.32\\0.28\\\vdots\\0.40\end{bmatrix},\begin{bmatrix}0.88\\0.03\\\vdots\\0.09\end{bmatrix},\begin{bmatrix}0.44\\0.44\\\vdots\\0.12\end{bmatrix},\begin{bmatrix}0.1\\0.1\\\vdots\\0.8\end{bmatrix},\begin{bmatrix}0.3\\0.5\\\vdots\\0.2\end{bmatrix},\begin{bmatrix}0.5\\0.25\\\vdots\\0.25\end{bmatrix},\begin{bmatrix}0.36\\0.18\\\vdots\\0.46\end{bmatrix}$$

03

Connect to the API

Steps:

1. Copy the Google Colab Notebook (Link in slack)
2. Run the code cells for Part 0: Setup
3. Run the code cells for Part 1: API Connections

04

BREAK

10 minutes

05

Parameters

Notable Parameters

- Model Name
- Messages
- Temperature
- Max Tokens
- Top P
- Frequency Penalty
- Presence Penalty

Chat Completion

Do Part 2: Understanding Completion Parameters

06

Prompt engineering

Message Roles

System

The system message helps set the behavior of the assistant

Assistant

The messages sent from the llm to respond to the user.

User

The user messages provide requests or comments for the assistant to respond to

Adopting a persona

Bad prompt:
“Explain how to solve quadratic equations.”

Better prompt:
“As a math tutor, could you explain to a high school student who is just starting with algebra how to solve quadratic equations, using simple terms and step-by-step examples”

Include details

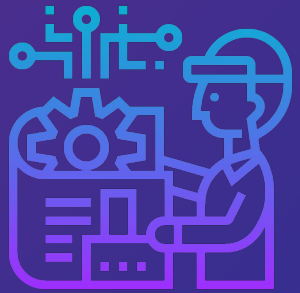
Bad prompt:
“Ice Cream”

Better prompt:
“I want to make Ice Cream, but I am on a diet, help me make a grocery list for creating a healthy Ice Cream from the ground up.”

Specify the steps to complete the task

Bad prompt:
“Help me find a bug in my code.”

Better prompt:
“I'm debugging a web application written in JavaScript that's supposed to fetch user data from an API and display it on the front end. However, the data isn't showing up. Could you outline the steps to take to systematically identify and fix the issue and then execute them on the code?"”



Do Part 3: Prompt Engineering

07

Embeddings

Embeddings

[-0.023302145302295685, 0.003694885177537799, ..., 0.033458139747381 21] = “This is an embedding!”

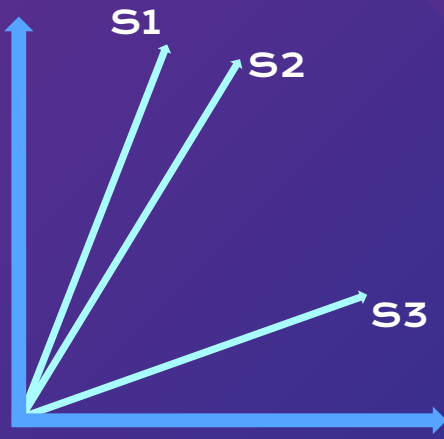
1536

Captures the contextual information in a multidimensional space

Basic idea: Measure the relatedness or similarity of text strings

Sentence 1 (S1): “The bank of the river was flooded after the heavy rain.”
Sentence 2 (S2): “Families picnicked on the river bank, enjoying the sunny day.”

Sentence 3 (S3): “She went to the bank to deposit her paycheck.”



Finding similar embeddings

Does anyone have an idea of how we can find similar embeddings?

Answer: Distance metrics! E.g., cosine distance, dot product or euclidean distance

We're going to use cosine distance:

$$\cos(\theta) = \frac{\mathbf{A} \cdot \mathbf{B}}{\|\mathbf{A}\|\|\mathbf{B}\|} = \frac{\sum_{i=1}^{n} A_i B_i}{\sqrt{\sum_{i=1}^{n} A_i^2} \sqrt{\sum_{i=1}^{n} B_i^2}}$$

Outputs a floating point number between 0 and 1

Why are embeddings useful?

Search

Q: “What is the capital of Norway?”

A: “The capital of Norway is Oslo”

Clustering

Group related data, without having to use explicit tags, e.g., “animal”, “vehicle”, or “city”

Gives more nuance

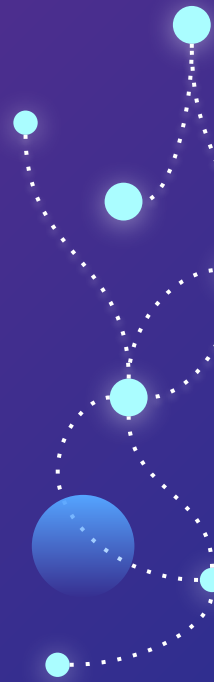
Classification

Group related data automatically and explicitly with tags

Text strings classified with most similar label

Anomaly detection

Detect outliers - things with little relatedness



Do Part 4: Embeddings

08

BREAK

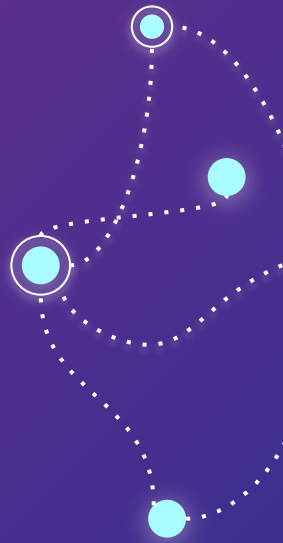
10 minutes

09 Function calling

Do Part 5: Function calling

Summary

- LLMs & AI
- Transformers
- Implementing GPT-3.5
  - Connect to the API
  - Parameters
  - Prompt engineering
- More advanced features
  - Embeddings
  - Function calling



Related projects!

Thank you for today!

Next workshop:
3D-printing & 3D-modelling

CREDITS: This presentation template was created by **Slidesgo**, and includes icons by **Flaticon**, and infographics & images by **Freepik**

Content by: Birk Stoveland, Olav Lorentzen and Sverre Nystad