

How LLM Works - Overview

what are vector embeddings

1706.03762

Understanding Pipeline.ipynb

ChatGPT

app.eraser.io/workspace/Wfxsp0qp9Hp2LgmnnFW

How LLM Works

Document Both Canvas

Share

197%

# Gen Pre-Trained Transformer

# Generative



v:1706.03762v7 [cs.CL] 2 Aug 2023

# Attention Is All You Need

**Ashish Vaswani\***  
Google Brain  
avaswani@google.com

**Noam Shazeer\***  
Google Brain  
noam@google.com

**Niki Parmar\***  
Google Research  
nikip@google.com

**Jakob Uszkoreit\***  
Google Research  
usz@google.com

**Llion Jones\***  
Google Research  
llion@google.com

**Aidan N. Gomez\* †**  
University of Toronto  
aidan@cs.toronto.edu

**Lukasz Kaiser\***  
Google Brain  
lukaszkaiser@google.com

**Illia Polosukhin\* ‡**  
illia.polosukhin@gmail.com

## Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks that include an encoder and a decoder. The best performing models also connect the encoder and decoder through an attention mechanism. We propose a new simple network architecture, the Transformer, based solely on attention mechanisms, dispensing with recurrence and convolutions entirely. Experiments on two machine translation tasks show these models to be superior in quality while being more parallelizable and requiring significantly less time to train. Our model achieves 28.4 BLEU on the WMT 2014 English-to-German translation task, improving over the existing best results, including ensembles by over 2 BLEU. On the WMT 2014 English-to-French translation task



How LLM Works - Overview

what are vector embeddings

1706.03762

Understanding Pipeline.ipynb

ChatGPT

app.eraser.io/workspace/Wfxsp0qp9Hp2LlgmnFW

How LLM Works

Document Both Canvas

⌘ K Share

150%

# Gen Pre-Trained Transformer

# Generative Pre-Trained Transformer



4:53 / 1:00:56

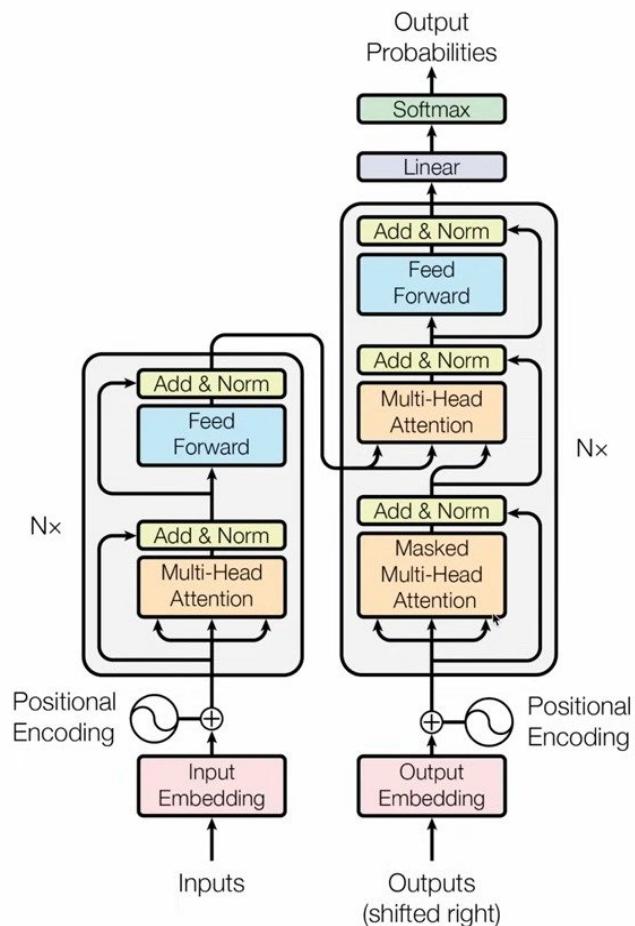


Figure 1: The Transformer - model architecture.



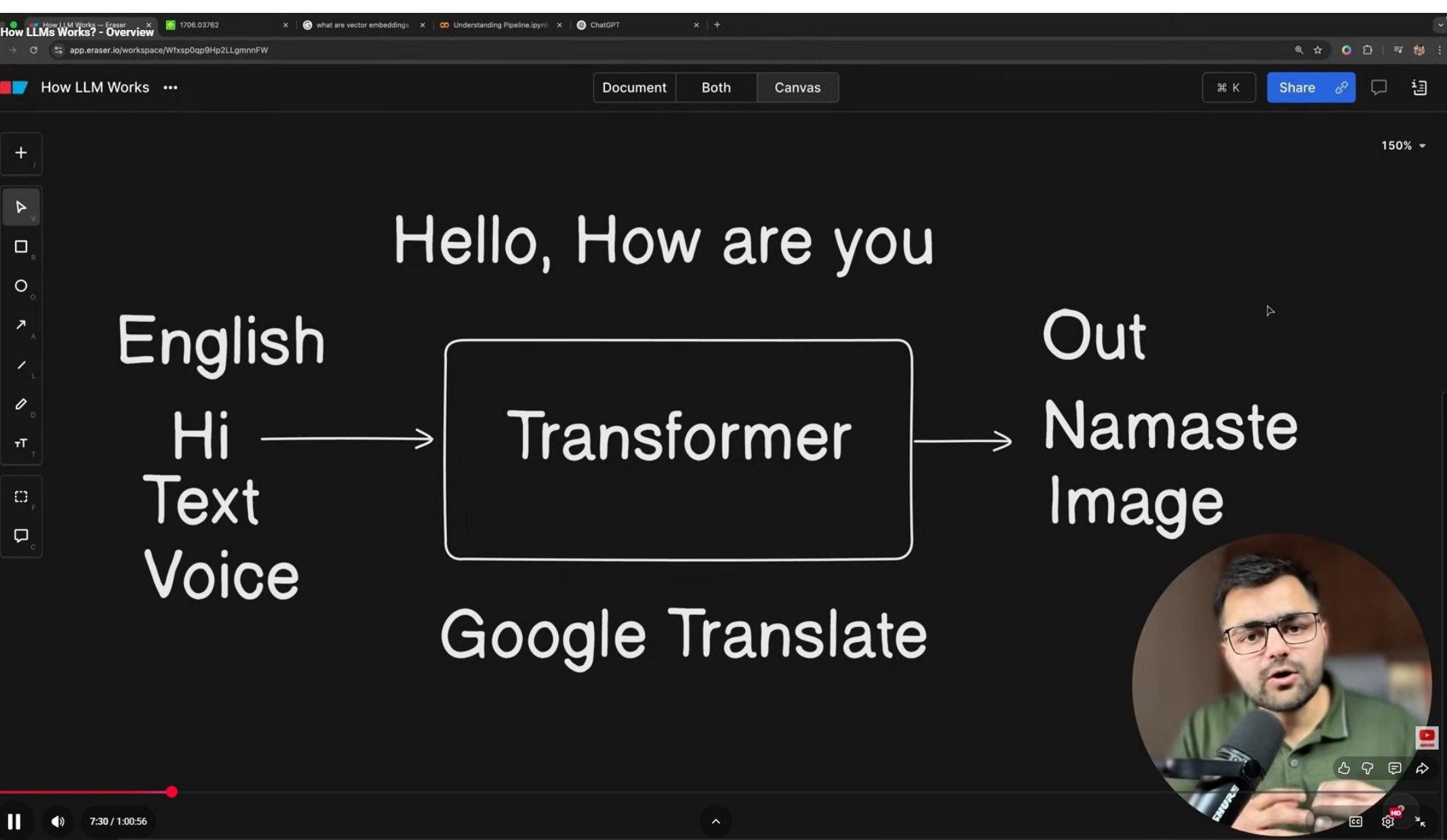
English  
Hi  
Text  
Voice

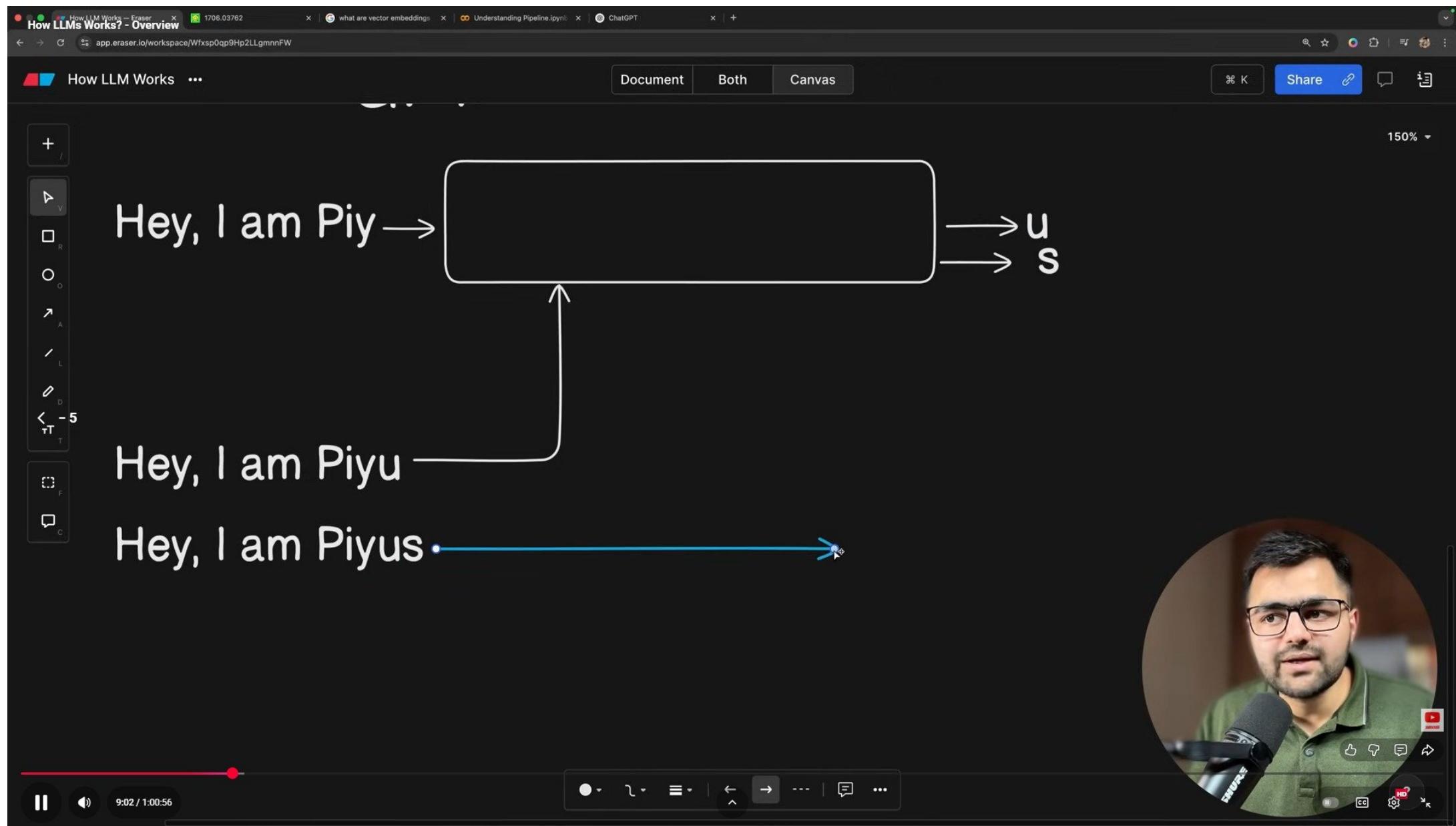
Hello, How are you

Transformer

Out  
Namaste  
Image

Google Translate





How LLM Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x +

app.eraser.io/workspace/Wfxsp0qp9Hp2LlgmnFW

How LLM Works ... Document Both Canvas ⌘ K Share 🔗

150% +

# GPT

The diagram illustrates the GPT architecture. It shows four input sequences: "Hey, I am Piy", "Hey, I am Piyu", "Hey, I am Piyus", and "Hey, I am Piyush". Each sequence is mapped by an arrow to a large central rectangular box representing the embedding space. From this box, two arrows point to the letters "u" and "s", representing the output words "us". A curved arrow also points from the bottom of the embedding box back up towards the input "Hey, I am Piyu".

Hey, I am Piy →

Hey, I am Piyu

Hey, I am Piyus →

Hey, I am Piyush → Khatam

A circular video player in the bottom right corner shows a man with dark hair and glasses, wearing a green polo shirt. He appears to be speaking or presenting. The video player includes standard controls like play/pause, volume, and a progress bar indicating 9:29 / 1:00:56.

Hey, I am PiyushH → Khatam

a-1

b-2

c-3

Step 1: Encoding → Hey There

Tokenize

↓      ↓  
10      36

Dic  
Hey - 10  
There - 36



# Tiktokenerizer

System

You are a helpful assistant



User

Content



Add message

```
<|im_start|>system<|im_sep|>You are a helpful  
assistant<|im_end|><|im_start|>user<|im_sep|><|im_end|>  
<|im_start|>assistant<|im_sep|>
```

Token count

16

```
<|im_start|>system<|im_sep|>  
t<|im_end|><|im_start|>use  
art|assistant<|im_sep|>
```

gpt-4o

Search model or encoder...

## Popular

cl100k\_base

o200k\_base

gpt-4-1106-preview

gpt-3.5-turbo

codellama/CodeLlama-7b-hf

## Open-Source Models

codellama/CodeLlama-70b-hf

meta-llama/Meta-Llama-3-8B

meta-llama/Meta-Llama-3-70B

microsoft/phi-2

google/gemma-7b

deepseek-ai/DeepSeek

Qwen/Qwen2.5-72B

tiiuae/falcon-7b



200264, 17360, 200266, 357  
00265, 200264, 1428, 20026  
00266

Show whitespace



13:30 / 1:00:56



# Tiktokenerizer

User

Hey There



Add message

```
<|im_start|>user<|im_sep|>Hey There<|im_end|>  
<|im_start|>assistant<|im_sep|>
```

gpt-4o

Token count

9

```
<|im_start|>user<|im_sep|>Hey There<|im_end|><|im_start|>assistant<|im_sep|>
```

200264, 1428, 200266, 25216, 3274, 200265, 200267, 781, 200266



14:04 / 1:00:56



Show whitespace



How LLMs Works - Eraser x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 Tiktokenizer x +

colab.research.google.com/drive/1nY5N9G3iWTK2rhj0t5soX12LD3FZ9GyY#scrollTo=1MOHKPxPuHL

CO Understanding Pipeline.ipynb ☆

File Edit View Insert Runtime Tools Help

Commands + Code + Text RAM Disk

0s import os

os.environ["HF\_TOKEN"] = "hf\_WFBwkTvvAKDBxZzSjgW0oTuMBVoloVTASw"

[4] from transformers import AutoTokenizer

tokenizer = AutoTokenizer.from\_pretrained("google/gemma-3-1b-it")  
tokenizer("Hey There!")

tokenizer\_config.json: 100% 1.16M/1.16M [00:00<00:00, 6.19MB/s]

tokenizer.model: 100% 4.69M/4.69M [00:00<00:00, 14.7MB/s]

tokenizer.json: 100% 33.4M/33.4M [00:00<00:00, 80.9MB/s]

added\_tokens.json: 100% 35.0/35.0 [00:00<00:00, 1.60kB/s]

special\_tokens\_map.json: 100% 662/662 [00:00<00:00, 16.9kB/s]

{'input\_ids': [2, 17531, 2085, 236888], 'attention\_mask': [1, 1, 1, 1]}



16:24 / 1:00:56 5s completed at 10:23

Hey, I am PiyushH → Khatam

a-1

b-2

c-3

Dic

Hey - 10

There - 36

Step 1: Encoding → Hey There

Tokenize

↓      ↓  
10      36



How LLMs Works? - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 T5 tokenizer x +

google.com/search?q=what+are+vector+embeddings&lz=1C6OZZY\_enN1135IN1135&oq=what+are+vector+embeddings&gs\_lcrp=EgZjaHJvbWUqDggAEUYJxg7GIAEGloFMg4IABBFGCcYOxiABBIKBTIHCAEQABiABDIMCAIQABgUGlCGIAEMggIAxAACBYYHjICQQABgWGB4yCAgFEAAWFhgeMggIBhAAGBYYHjICAcQABgW...

# what are vector embeddings

All Images Videos Short videos Forums Shopping Web More

## Search Labs | AI Overview

हिं En Listen

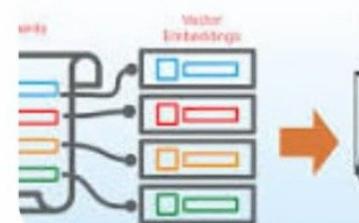
Vector embeddings are numerical representations of data, like words, images, or audio, that capture semantic relationships and meaning, allowing machine learning models to process and compare them efficiently.

What are Vector embeddings and other data into Elastic

What they are:

Vector embeddings are essentially "fingerprints" or numerical representations of data

18:57 / 1:00:56 Show more ▾



Step 1: Encoding → Hey There

10 36

There - 36

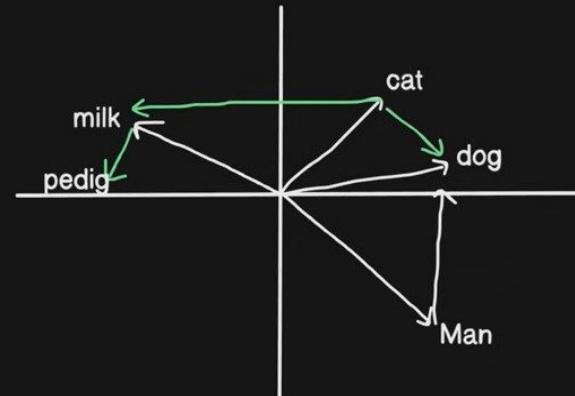
Tokenize

CAT - MILK  
DOG - Pedig

Vector Embeddings

CAT = 20

DOG = 80



Step 1: Encoding → Hey There

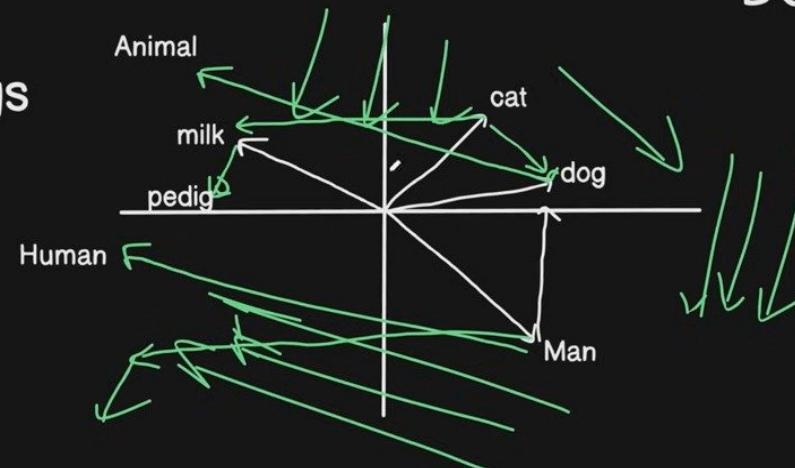
There - 36

Tokenize

Vector Embeddings

CAT = 20

DOG = 80



CAT - MILK  
DOG - Pedig



How LLMs Works - Eraser | 1706.0376 | what are vector embeddings | Understanding Pipeline.ipynb | ChatGPT | T5\_Tiktokener | platform.openai.com/docs/guides/embeddings | +

platform.openai.com/docs/guides/embeddings

Google openai vector embeddings

All Images Videos Short videos Shopping Forums News More Tools

OpenAI https://platform.openai.com/docs/guides/embeddings Vector embeddings

What are embeddings? OpenAI's text embeddings measure the relatedness of text strings. Embeddings are commonly used for: ... An embedding is a vector (list) of ...

Vector embeddings What are embeddings? OpenAI's text embeddings measure the ...

OpenAI Platform Explore resources, tutorials, API docs, and dynamic examples to ...

What are embeddings? An embedding is a vector (list) of floating point numbers. The ...

Here Explore resources, tutorials, API docs, and dynamic examples to ...

Sample use-case Explore resources, tutorials, API docs, and dynamic examples to ...

More results from openai.com »

OpenAI https://openai.com/index/new-embedding-models-a... New embedding models and API updates 25 Jan 2024 — We are launching a new generation of embedding models, new moderation models, new API usage management tools, and soon, lower prior

People also ask : What is the embedding vector dimension in OpenAI? What is vector embedding in AI? Are OpenAI embeddings normalized? क्या OpenAI एमेडिंग सामान्यीकृत है?

DataCamp https://www.datacamp.com/.../Artificial Intelligence Introduction to Text Embeddings with the OpenAI API 9 Jun 2020 Explore our guide on using the OpenAI API for creating text embeddings for their applications in text classification, information retrieval, and semantic ...

Search for apps, settings, and documents

Pinned Edge Outlook (new) Calendar Microsoft Store Photos Settings Calculator Clock Notepad Paint File Explorer Films & TV Recycle Bin Class 16

All >

Recommended x64 Native Tools Command Prompt... Recently added x64\_x86 Cross Tools Command Pro... Recently added x86\_x64 Cross Tools Command Pro... Recently added Gemini\_Generated\_Image\_hukdx4... 2h ago Gemini\_Generated\_Image\_b5fx61b... 2h ago

More >

Rahmat Ali Hide this pane

Select device Android™ iPhone®

Access your mobile device here Keep up with calls, messages, and recent activity here in the Start menu.

Hide this pane

16°C Sunny ENG IN 11:04 07-12-2025

• **Diversity measurement** (where similarity distributions are analyzed)

• **Classification** (where text strings are classified by their most similar label)

An embedding is a vector (list) of floating point numbers. The [distance](#) between two vectors measures their relatedness. Small distances suggest high relatedness and large distances suggest low relatedness.

Visit our [pricing page](#) to learn about embeddings pricing. Requests are billed based on the number of [tokens](#) in the [input](#).

## How to get embeddings

To get an embedding, send your text string to the [embeddings API endpoint](#) along with the embedding model name (e.g., `text-embedding-3-small`):

Example: Getting embeddings

```
1 from openai import OpenAI
2 client = OpenAI()
3
4 response = client.embeddings.create(
5     input="Your text string goes here",
6     model="text-embedding-3-small"
7 )
8
9 print(response.data[0].embedding)
```

The response contains the embedding vector (list of floating point numbers) along with some additional metadata. You can extract the embedding vector, save it in a vector database, and use for many different use cases.

A circular portrait of a man with dark hair, a beard, and glasses, wearing a green polo shirt. He is speaking into a black microphone. In the bottom right corner of the portrait, there is a small red YouTube logo.

for many different use cases.

```
1  {
2      "object": "list",
3      "data": [
4          {
5              "object": "embedding",
6              "index": 0,
7              "embedding": [
8                  -0.006929283495992422,
9                  -0.005336422007530928,
10                 -4.547132266452536e-05,
11                 -0.024047505110502243
12             ],
13         }
14     ],
15     "model": "text-embedding-3-small",
```



Requirement already satisfied: h11<0.15,>=0.13 in /usr/local/lib/python3.11/dist-packages (from httpcore==0.18.1) (from requests==2.28.1)  
Requirement already satisfied: annotated-types>=0.6.0 in /usr/local/lib/python3.11/dist-packages (from pydantic==1.10.2)  
Requirement already satisfied: pydantic-core==2.27.2 in /usr/local/lib/python3.11/dist-packages (from pydantic==1.10.2)

```
from openai import OpenAI
import os

client = OpenAI()

os.environ["OPENAI_API_KEY"] = "sk-proj-JJSk5dBf12JGy_4Giniq3UvbV7Xmbb7ezXRKhE_BDjqWeUY-2EHVGtbMz00CeluMT

response = client.embeddings.create(
    input="Cat loves milk",
    model="text-embedding-3-small"
)

print(response.data[0].embedding)
```



26:20 / 1:00:56

How LLM Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 Tikkontizer x Vector embeddings - OpenAI x + app.eraser.io/workspace/Wfxsp0qp9Hp2LlgmnFW

How LLM Works ... Document Both Canvas ⌘ K Share ↲ ⓘ 149% ▾

+ / ▶ R O A L D T F C

Step 2: Positional Encoding

The diagram illustrates the process of generating a final vector representation for a sentence. It starts with two separate vectors: one for the word "dog" and one for the word "cat". The "dog" vector is shown in green, and the "cat" vector is shown in red. These vectors are added together using a plus sign (+) to produce a combined vector, which is then equated (=) to the final result. The final result is a blue vector that represents both words in their respective positions within the sentence. To the right of the vectors, there is a text box containing the sentence "The dog chased cat" and "The cat chased dog", with the words "dog", "cat", and "dog" highlighted in green, red, and blue respectively, corresponding to the colors of the vectors.

+

=

The dog chased cat  
The cat chased dog

|| 30:18 / 1:00:56 Text Code 19px T ...

A circular video overlay in the bottom right corner shows a man with glasses and a beard, wearing a green shirt, speaking into a black microphone. The video has a play button icon in the bottom right corner. Below the video are several small icons for interacting with the video player.

	Operations		
Self-Attention	$O(n^2 \cdot d)$	$O(1)$	$O(1)$
Recurrent	$O(n \cdot d^2)$	$O(n)$	$O(n)$
Convolutional	$O(k \cdot n \cdot d^2)$	$O(1)$	$O(\log_k(n))$
Self-Attention (restricted)	$O(r \cdot n \cdot d)$	$O(1)$	$O(n/r)$

### 3.5 Positional Encoding

Since our model contains no recurrence and no convolution, in order for the model to make use of the order of the sequence, we must inject some information about the relative or absolute position of the tokens in the sequence. To this end, we add "positional encodings" to the input embeddings at the bottoms of the encoder and decoder stacks. The positional encodings have the same dimension  $d_{\text{model}}$  as the embeddings, so that the two can be summed. There are many choices of positional encodings, learned and fixed [9].

In this work, we use sine and cosine functions of different frequencies:

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

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

where  $pos$  is the position and  $i$  is the dimension. That is, each dimension of the positional encoding corresponds to a sinusoid. The wavelengths form a geometric progression from  $2\pi$  to  $10000 \cdot 2\pi$ . We chose this function because we hypothesized it would allow the model to easily learn to attend to relative positions, since for any fixed offset  $k$ ,  $PE_{pos+k}$  can be represented as a linear function of  $PE_{pos}$ .



How LLMs Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 Tiktokenizer x Vector embeddings - OpenAI x +

app.eraser.io/workspace/Wfxsp0qp9Hp2LLgmnnFW

How LLM Works ... Share

Document Both Canvas 149% ▾

Step 2: Positional Encoding

Self Attention

=

The diagram illustrates the addition of positional encoding to word embeddings. It shows two sets of vertical vectors representing word embeddings. The left set is colored red, and the right set is colored blue. An equals sign (=) is positioned between the two sets. Below the red set, there is a horizontal line with a wavy underline, and below the blue set, there is another horizontal line with a wavy underline. This visualizes how the original word embeddings are combined with learned positional encodings.

32:25 / 1:00:56

Text Code 19px T ...

Like Dislike Comment Share

A circular video feed in the bottom right corner shows a man with dark hair and glasses, wearing a green shirt, speaking into a black microphone. The video has a play button icon in the bottom right corner.

How LLM Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 Tiktokenizer x Vector embeddings - OpenAI x + app.eraser.io/workspace/Wfxsp0qp9Hp2LlgmnFW

How LLM Works ... Document Both Canvas Share ↗ ⓘ 175% ▾

+ / ▶ R O A L D T F C

RNN → [ ] [ ] [ ] [ ]

[ ] [ ] [ ] [ ]

The river bank  
The ICICI Bank+65 >

Self Attention

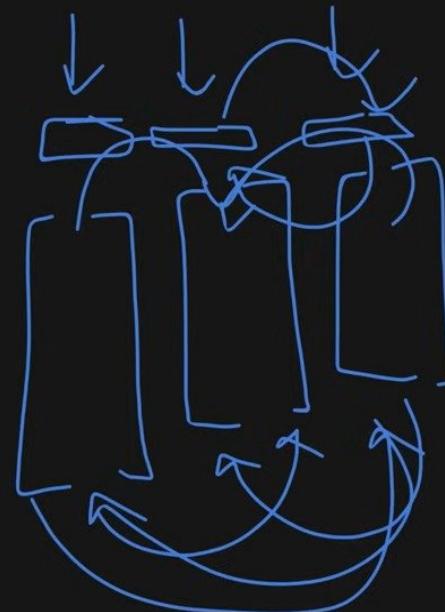
II 🔊 34:42 / 1:00:56 ⏵ 🔍 CC HD

Self Attention

The river bank

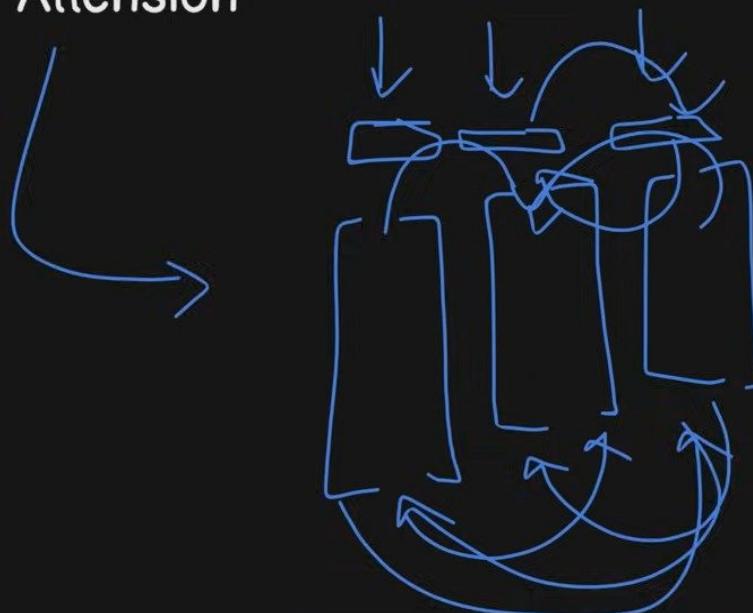
Context of the word is maintained

Relates words / token to each other

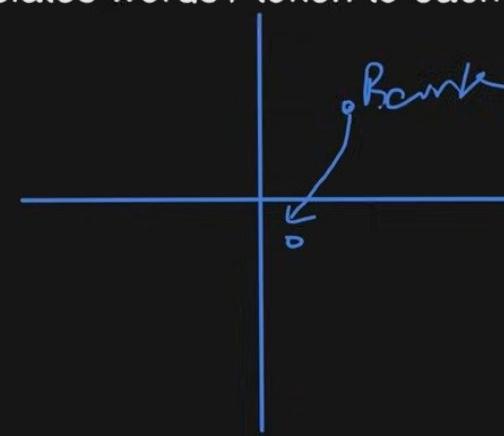


Self Attention

The river bank



Context of the word is maintained  
Relates words / token to each other



37:36 / 1:00:56

Text    Code    15px    T    ...

How LLM Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 Tiktokerizer x Vector embeddings - OpenAI x +

app.eraser.io/workspace/Wfxsp0qp9Hp2LgmnnFW

How LLM Works ... Document Both Canvas Share

147% ▾

Single Head Self Attention

Multi Head Attention Improves the contextual understanding

Dog was in train

Dog was sleeping

Dog was brown in color

+

0:00 40:24 / 1:00:56

The image shows a hand-drawn diagram of a hand with arrows indicating self-attention flow, followed by a video player interface showing a man speaking. The video player includes controls for volume, playback, and a progress bar.

How LLM Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 Tiktokenizer x Vector embeddings - OpenAI x +

app.eraser.io/workspace/Wfxsp0qp9Hp2LgmnnFW

How LLM Works ... Document Both Canvas Share

Dog was brown in color

94% ▾

+ 60 >

Diagram illustrating the LLM processing pipeline:

- The input sentence "Dog was brown in color" is processed by an encoder (represented by a stack of four rectangles).
- The output of the encoder is fed into a transformer architecture (represented by a stack of four rectangles).
- The transformer's output is then processed by a decoder (represented by a stack of three rectangles).
- The final output is a sequence of tokens (represented by green brackets containing symbols like 'u', 'v', 'w', 'x', 'y', 'z', 'A', 'B', 'C', 'D', 'E', 'F', 'G', 'H', 'I', 'J', 'K', 'L', 'M', 'N', 'O', 'P', 'Q', 'R', 'S', 'T', 'U', 'V', 'W', 'X', 'Y', 'Z').
- The tokens are then passed through a "De-Tokenize" layer.
- The final output is the reconstructed sentence: "U

Below the diagram, there is a video player interface showing a man speaking into a microphone. The video player includes controls for play/pause, volume, and a progress bar indicating 42:31 / 1:00:56.

Bottom navigation bar: Create Figure F, ..., HD, CC, Settings, etc.

Training  
Inferencing Phase

Training

Output →

Output → Decoder

Encoder

Hi, How are you?

<start> I am fine

<EOS>  
label



45:43 / 1:00:56

Training  
Inferencing Phase

[ J L J L J L ]

Training

Output → Decoder

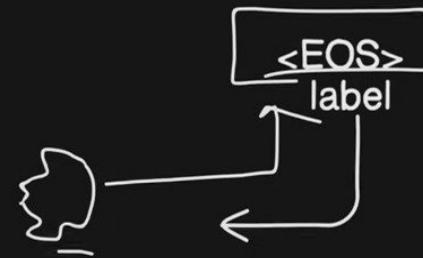
Encoder

Hi, How are you?

<start> I am fine

Output →

Back Prop



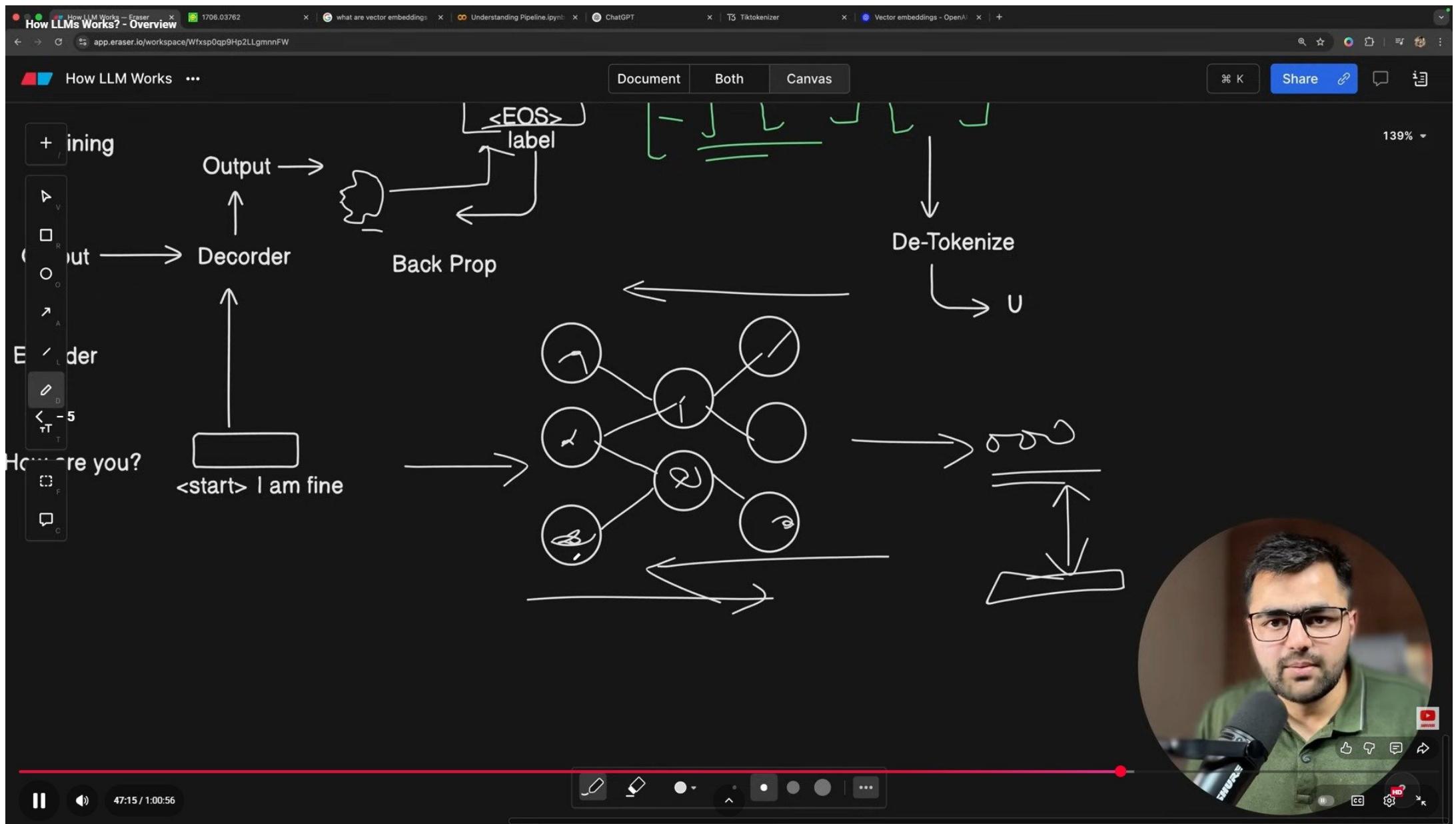
[ J L J L J L ]

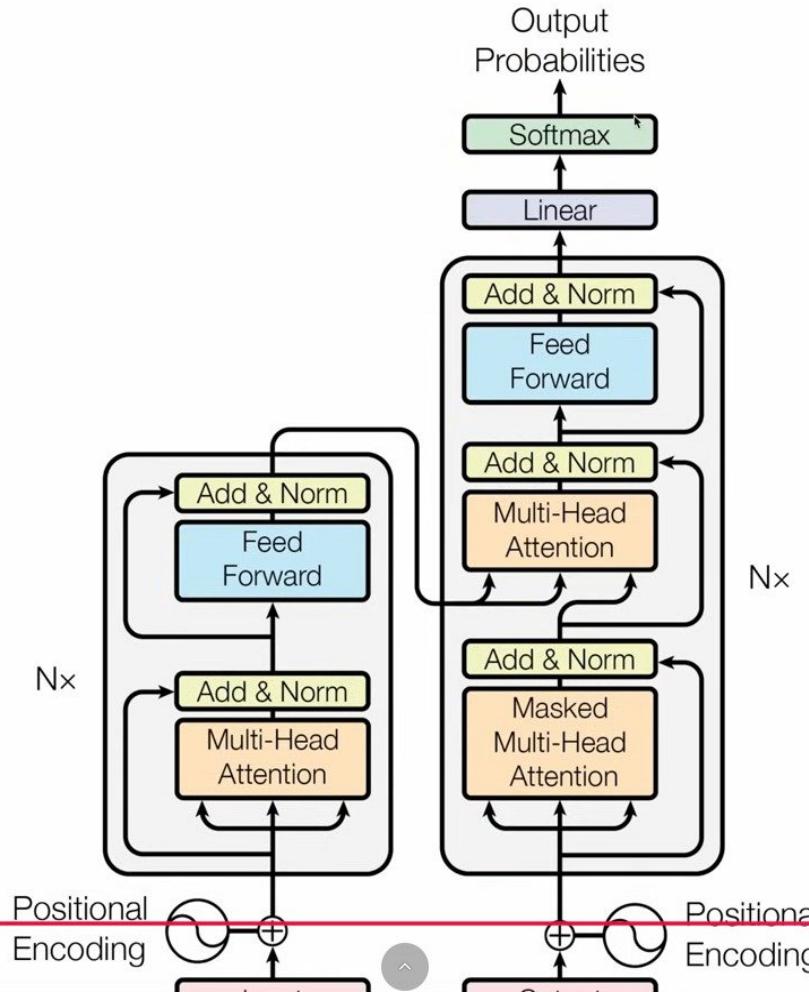
160%

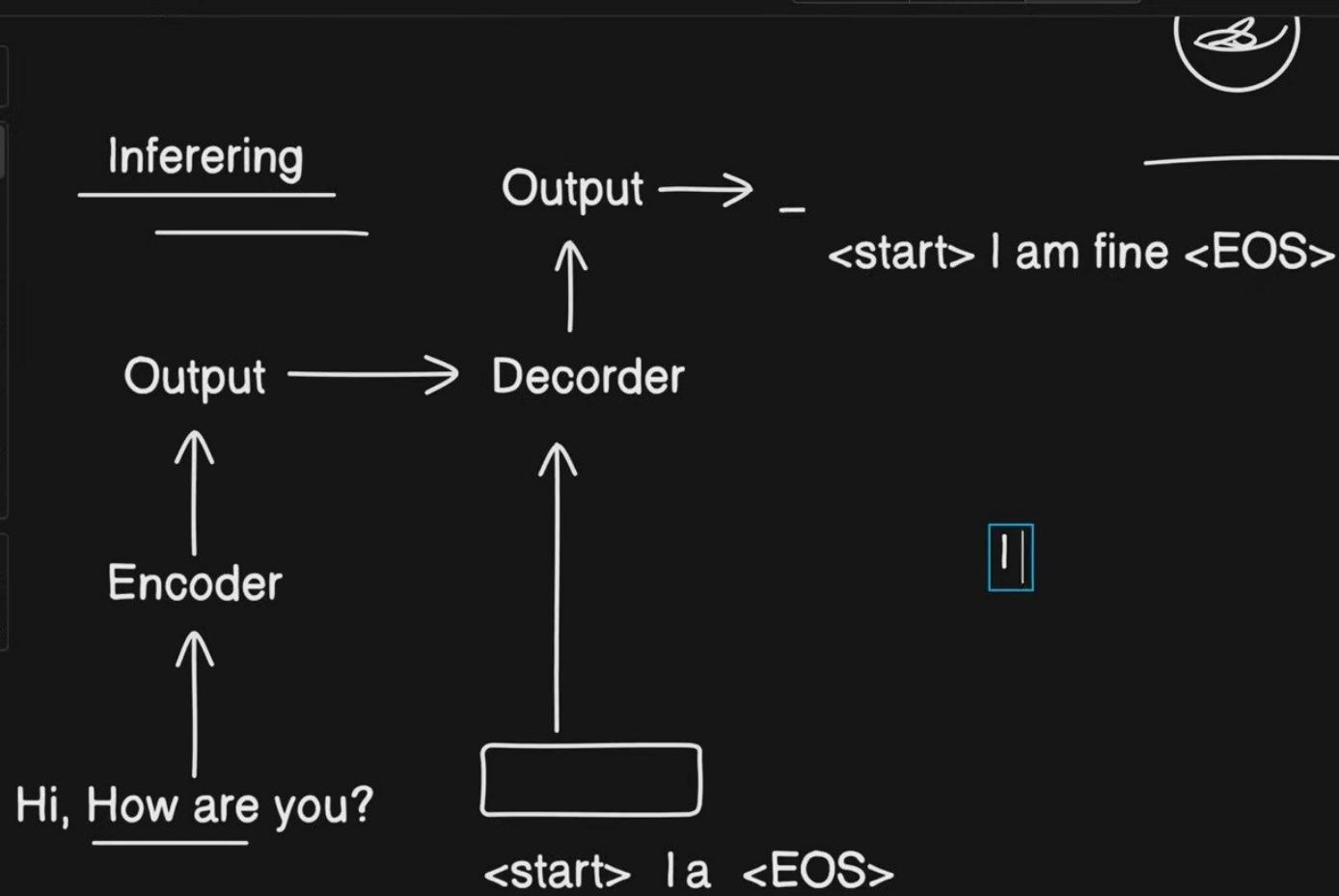
De-To



46:18 / 1:00:56







How LLM Works - Overview x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x ChatGPT x T5 T5 tokenizer x Vector embeddings - OpenAI x +

app.eraser.io/workspace/Wfxsp0qp9Hp2LLgmnnFW

How LLM Works ... Document Both Canvas ⌘ K Share ↗ ⓘ 172% ▾

+ ↗ \_linear → \_  
□ R o o  
→ ↗ recorder A  
' L  
□ D  
□ T T  
□ F  
□ C

<start> I am fine <EOS>

I am f

Linear  
f: 0.9  
g: 0.1 → Softmax 1  
n:  
x:

+ 35 >

<start> I a <EOS>

11 51:12 / 1:00:56

```
How LLMs Works - Fraser x 1706.03762 x what are vector embeddings x Understanding Pipeline.ipynb x Greeting and Assistance | Go x T5 Tiktokenizer x Vector embeddings - OpenAI x + colab.research.google.com/drive/inY5N9G3iWTKZrhjOt5soXl2LD3FZ9GyY#scrollTo=9W_UO7THaQ8t
```

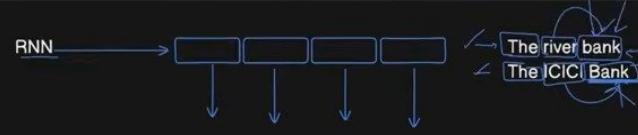
```
→ CausalLMOutputWithPast(loss=None, logits=tensor([[-14.1875, -1.1250, 4.6562, ..., -14.6250, -14.6875, -14.6250], [-17.6250, -0.6992, -3.2500, ..., -18.0000, -18.0000, -17.8750], [-12.1875, -2.9844, -4.9688, ..., -13.4375, -13.4375, -13.3125], ..., [-14.8750, -3.8750, -2.4688, ..., -16.6250, -16.6250, -16.5000], [-15.9375, -3.1250, 1.3047, ..., -18.0000, -18.0000, -17.8750], [-17.0000, -2.3125, -0.4629, ..., -19.5000, -19.5000, -19.2500]]], dtype=torch.bfloat16, grad_fn=<UnsafeViewBackward0>), past_key_values=<transformers.cache_utils.HybridCache object at 0x000001D8A8E8000>, hidden_states=None, attentions=None)
```

13s [18] gen\_out = model.generate(input\_ids=input\_tokens["input\_ids"])  
gen\_out

```
→ tensor([[ 2, 6974, 496, 23181, 3393, 573, 8009, 1156, 4945, 107, 2717, 6719, 107, 2063, 1138, 236779, 34488, 236769, 236781, 236764, 570, 1473, 107, 138, 12234, 174054, 1156, 4945, 3075]]) + 155 >
```

Start coding or generate with AI.

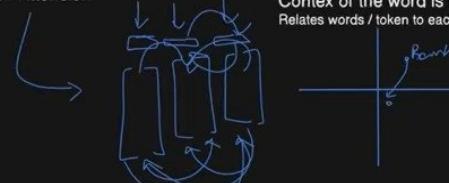




## Self Attention

## The river bar

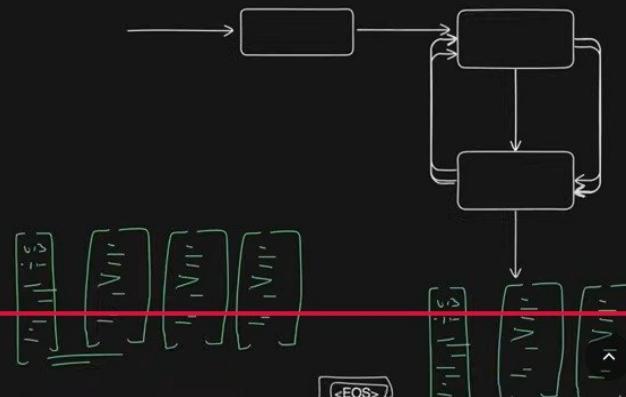
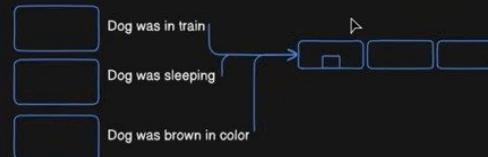
**Context of the word is maintained**  
Relates words / token to each other



Single Head Self Attention

### Multi Head Attention

Improves the contextual understanding



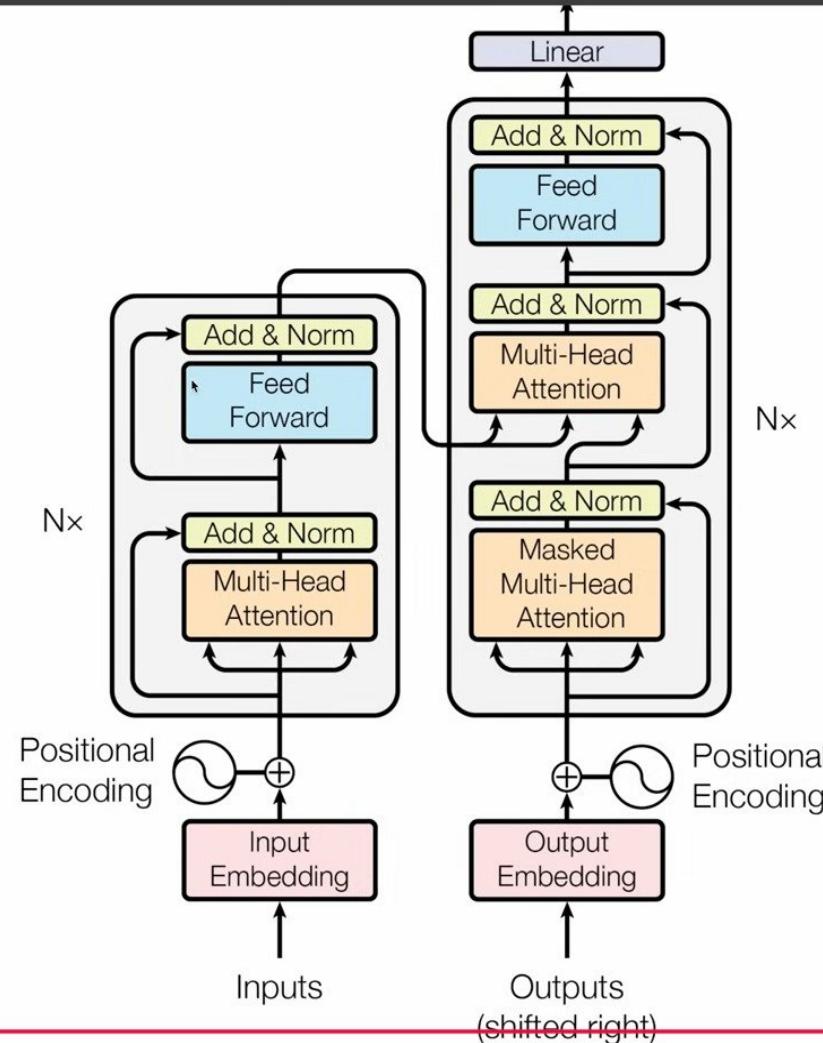


Figure 1: The Transformer model architecture



