KEYVALUE
Let's build your idea

Introduction to Large Language Models (LLMs)

Freshers Training June 2025

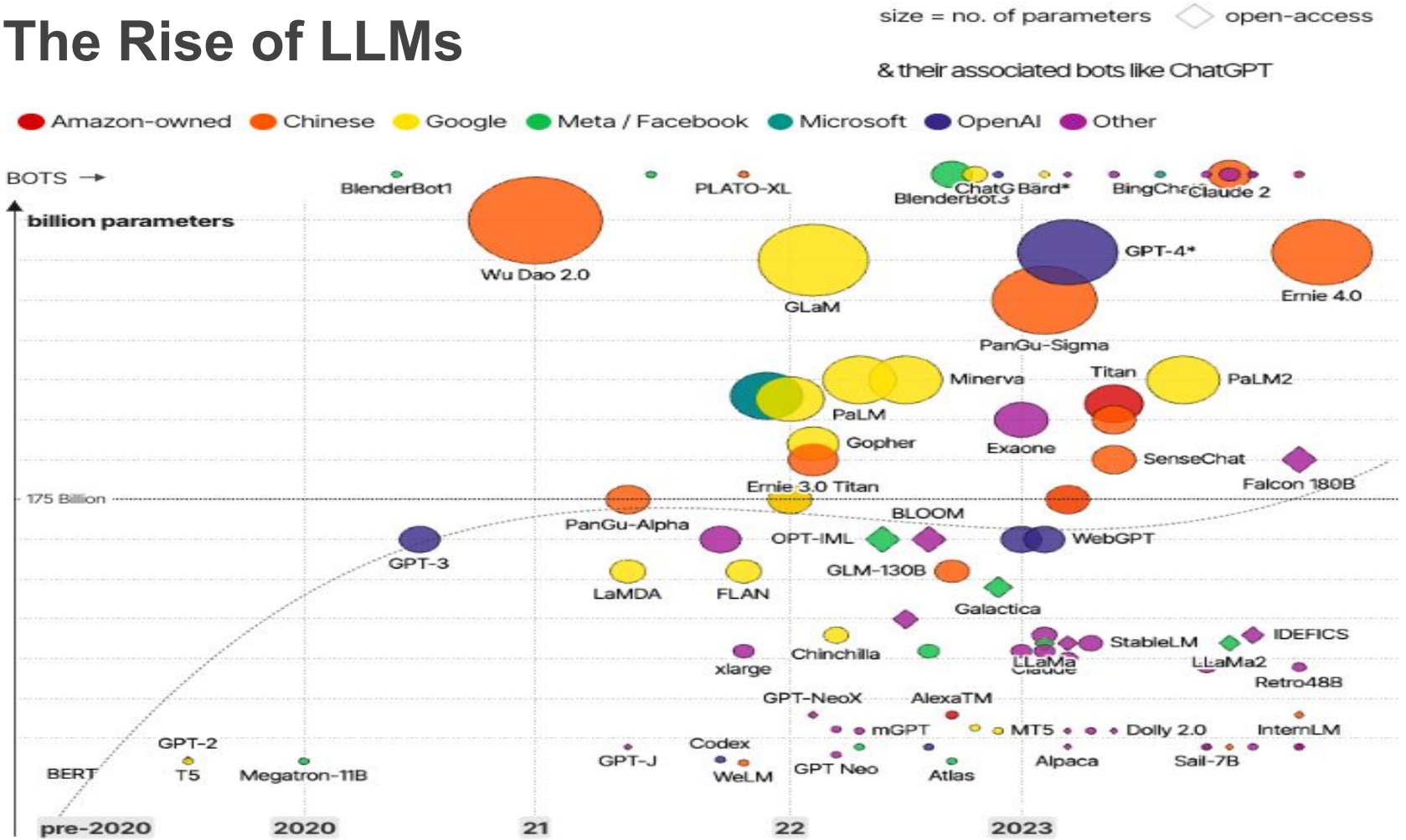
Overview

- Introduction to Large Language Models (LLMs)
- Understanding Transformer Networks
- Word Embeddings: Representing Language as Vectors
- Vector Databases: Storing and Retrieving Embeddings
- Sentence Transformers: Generating Sentence Embeddings
- Practical Applications and Use Cases

What are Large Language Models (LLMs)?

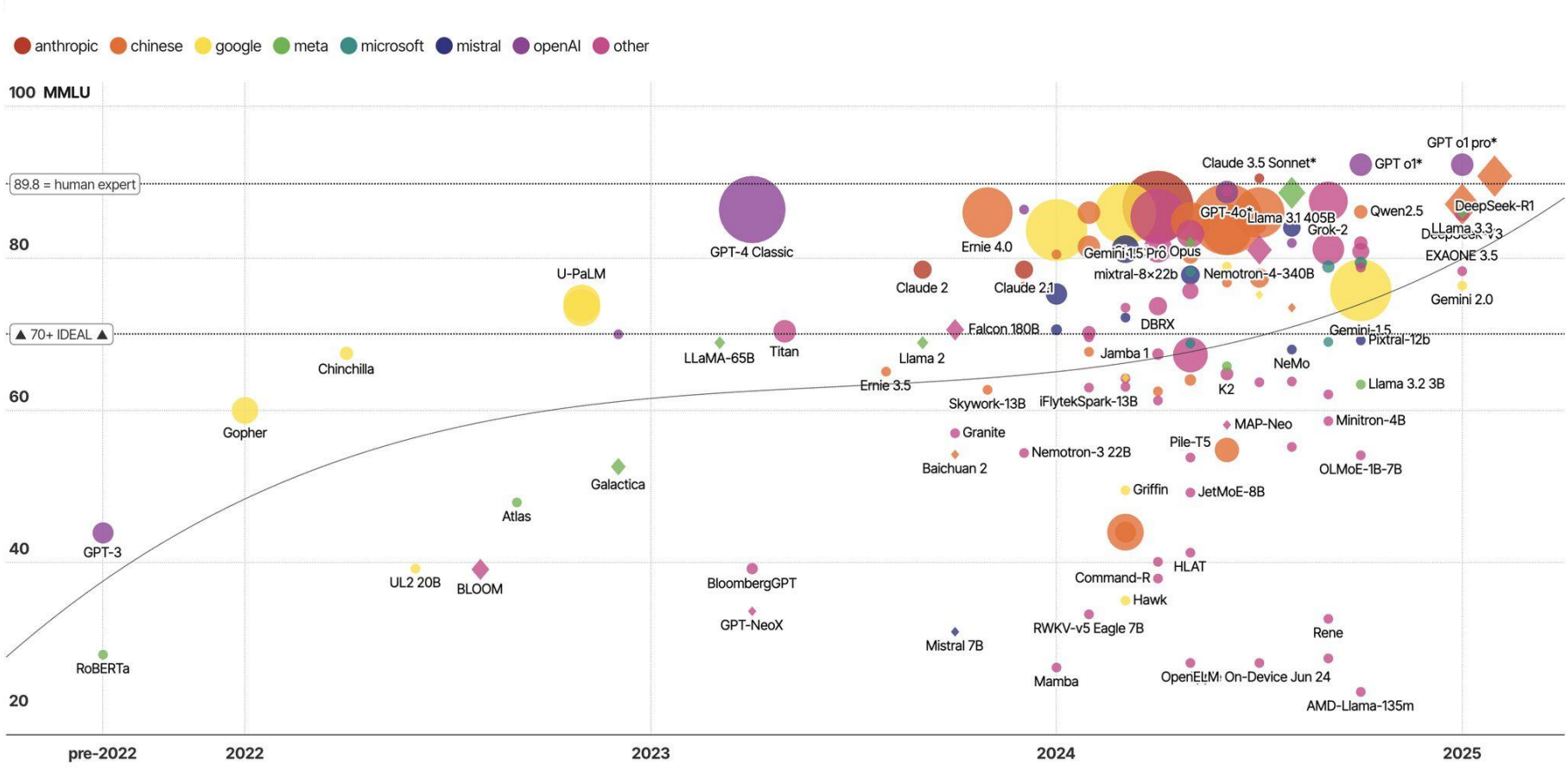
- Deep learning models trained on massive amounts of text data.
- Capable of understanding, generating, and manipulating human language.
- Use **Transformer** architecture for sequence modeling.
- Unlike LSTM, rely on **self-attention** to capture relationships between tokens.
- Examples: GPT-3, BERT, T5, and Llama.

The Rise of LLMs



Major Large Language Models (LLMs)

ranked by capabilities, sized by billion parameters used for training



2025 Releases !

January
- DeepSeek R1
- Qwen 2.5-Max
- o3-mini

February
- Gemini 2.0 Flash
- Gemini 2.0 Pro
- Grok 3
- Grok 3 mini
- Claude 3.7 (Sonnet)
- QwQ-Max (preview)
- GPT-4.5

March
- Gemma 3 (1B • 4B • 12B • 27B)
- Gemini 2.5 Pro (public preview)
- Llama 4 Scout
- Llama 4 Maverick

April
- GPT-4.1
- GPT-4.1-mini
- GPT-4.1-nano
- o3 (full)
- o4-mini
- Gemini 2.5 Flash (preview)
- Gemma 3 QAT (1B • 4B • 12B • 27B)

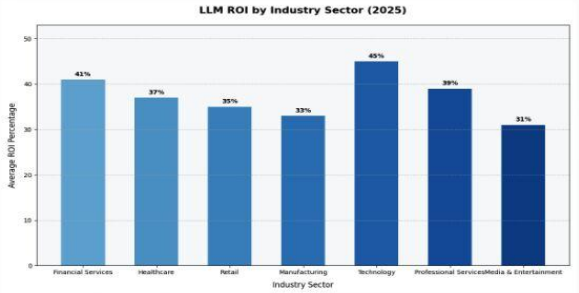
Applications of LLMs

- Text generation (articles, stories, poems).
- Translation.
- Question answering.
- Chatbots and virtual assistants.
- Code generation.
- Summarization.
- Search enhancement.
- Document classification.
- Q&A systems.
- Entity Extraction.

ROI & Strategic Impact

Measuring the business value of LLM implementations

Average ROI by Industry (2025)



Early adopters report 41% average ROI across sectors

Strategic Benefits

Productivity
5-15% marketing productivity boost

Innovation
50% faster product development

Cost Savings
30% lower development costs

Knowledge Access
4x better info discovery

Key Business Applications

Top use cases driving LLM adoption across industries

Customer Support
24/7 assistance with human-like interactions

Content Creation
High-quality content generation at scale

Data Analysis
Natural language interface to complex data

Code Generation
55% faster development cycles

Key Concepts of LLMs

- Parameters: The model's learned values that define its behavior.
- Training Data: The text used to teach the model.
- Training Objectives :
  - *Masked Language Modeling* (BERT): Predict masked tokens.
  - *Causal Language Modeling* (GPT): Predict next token in sequence.
- Pretraining: Model learns from large corpus (unsupervised).
- Fine-tuning: Adapts model to specific task with labeled data.
- Inference: Using the trained model to generate new text.

Popular LLM Types

- Base LLMs: Raw pretrained models with no fine-tuning.

  Eg: GPT, LLaMA, PaLM (base)- Use: Foundation for adaptation

- Instruction-Tuned Models: Trained to follow human instructions.

  Eg: FLAN-T5, Mistral-Instruct, LLaMA-2-Chat- Use: Task completion, zero-shot prompts

- Dialogue-Tuned Models: Optimized for multi-turn conversations.

  Eg: ChatGPT, Claude, Gemini Chat- Use: Chatbots, assistants

- RLHF-Tuned Models: Aligned using Reinforcement Learning from Human Feedback.

  Eg: GPT-4 (ChatGPT), Instruct GPT- Use: Safer, aligned responses

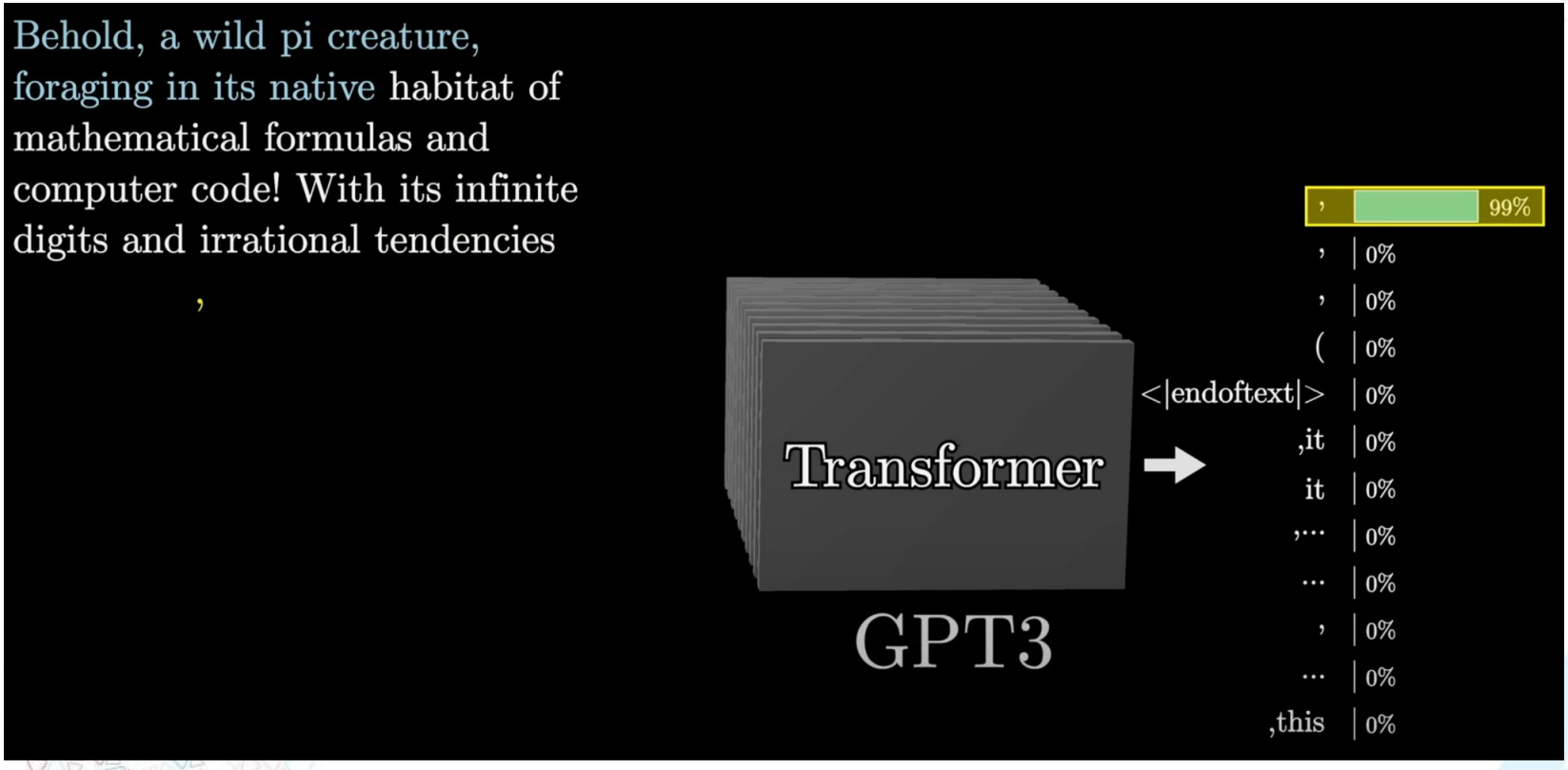
- Domain-Specific LLMs: Adapted for specific fields

  Eg: Med-PaLM, CodeLLaMA, Legal-BERT- Use: Medicine, Law, Programming

Transformer Networks: The Engine of LLMs

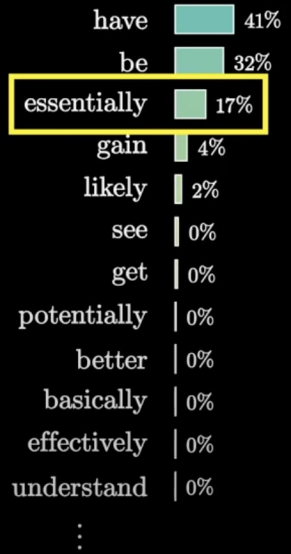
- A type of neural network architecture that revolutionized NLP.
- “Attention is All You Need” - Google Landmark research paper in 2017.
- Key innovation: the "attention mechanism."
- Excels at capturing long-range dependencies in text.
- Stack of encoders and decoders.

Predict, Sample, Repeat



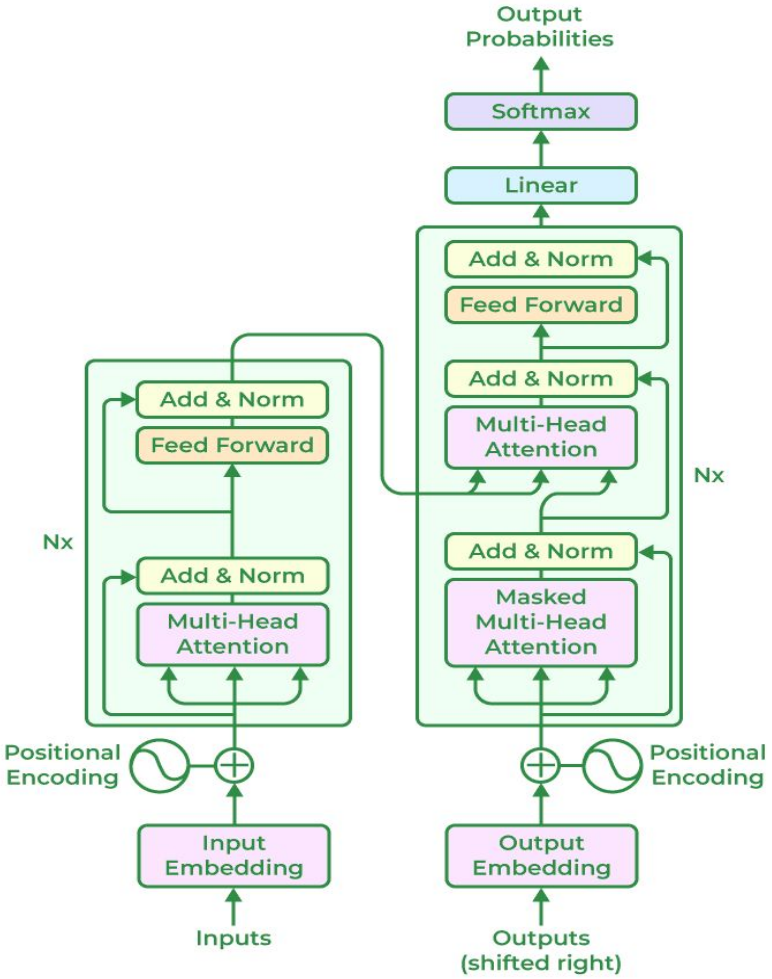
If you could see the underlying probability distributions a large language model uses when generating text, then you

If you could see the underlying probability distributions a large language model uses when generating text, then you would essentially



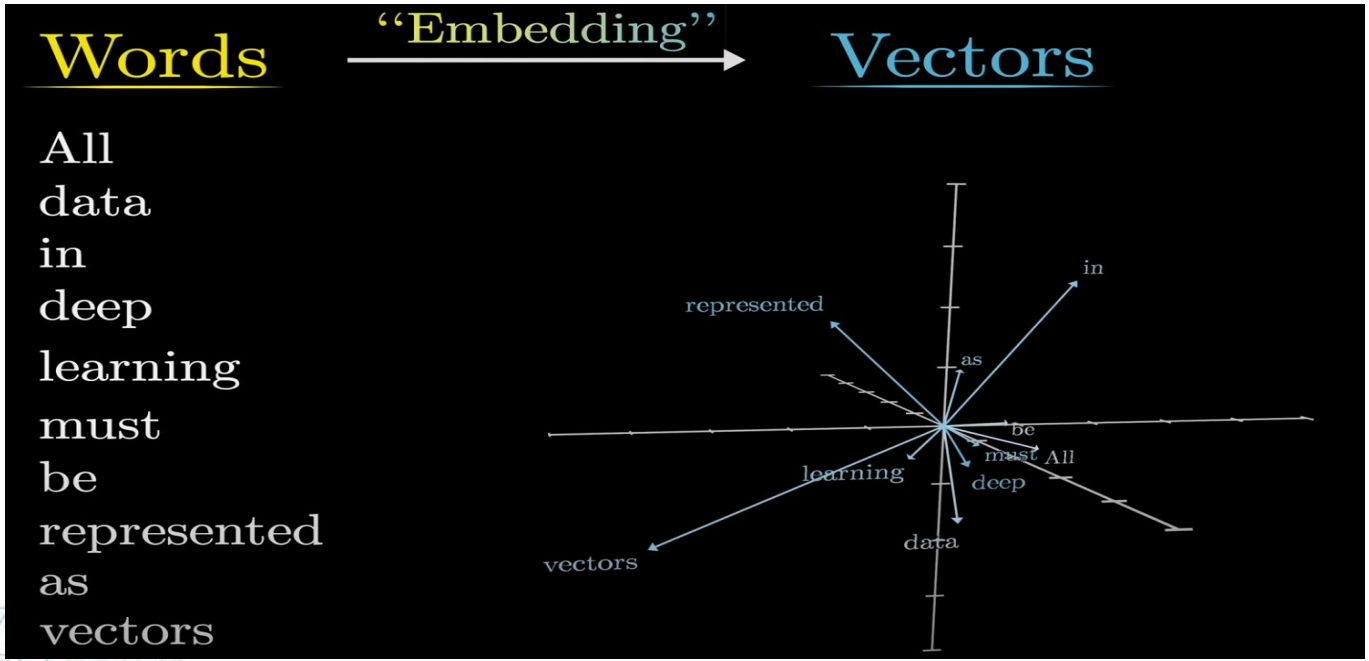
Encoder and Decoder

- **Input Embedding**: Converts tokens into vector representations.
- **Positional Encoding**: Adds information about the position of each token in the sequence.
- **Multi-Head Attention**: Allows the model to focus on different parts of the input sequence simultaneously.
- **Feed Forward Neural Network**: Processes the attention outputs to capture complex patterns.
- **Add & Norm**: Applies residual connections followed by layer normalization.
- **Output Probabilities**: Generates the probability distribution over the vocabulary for the next token prediction.

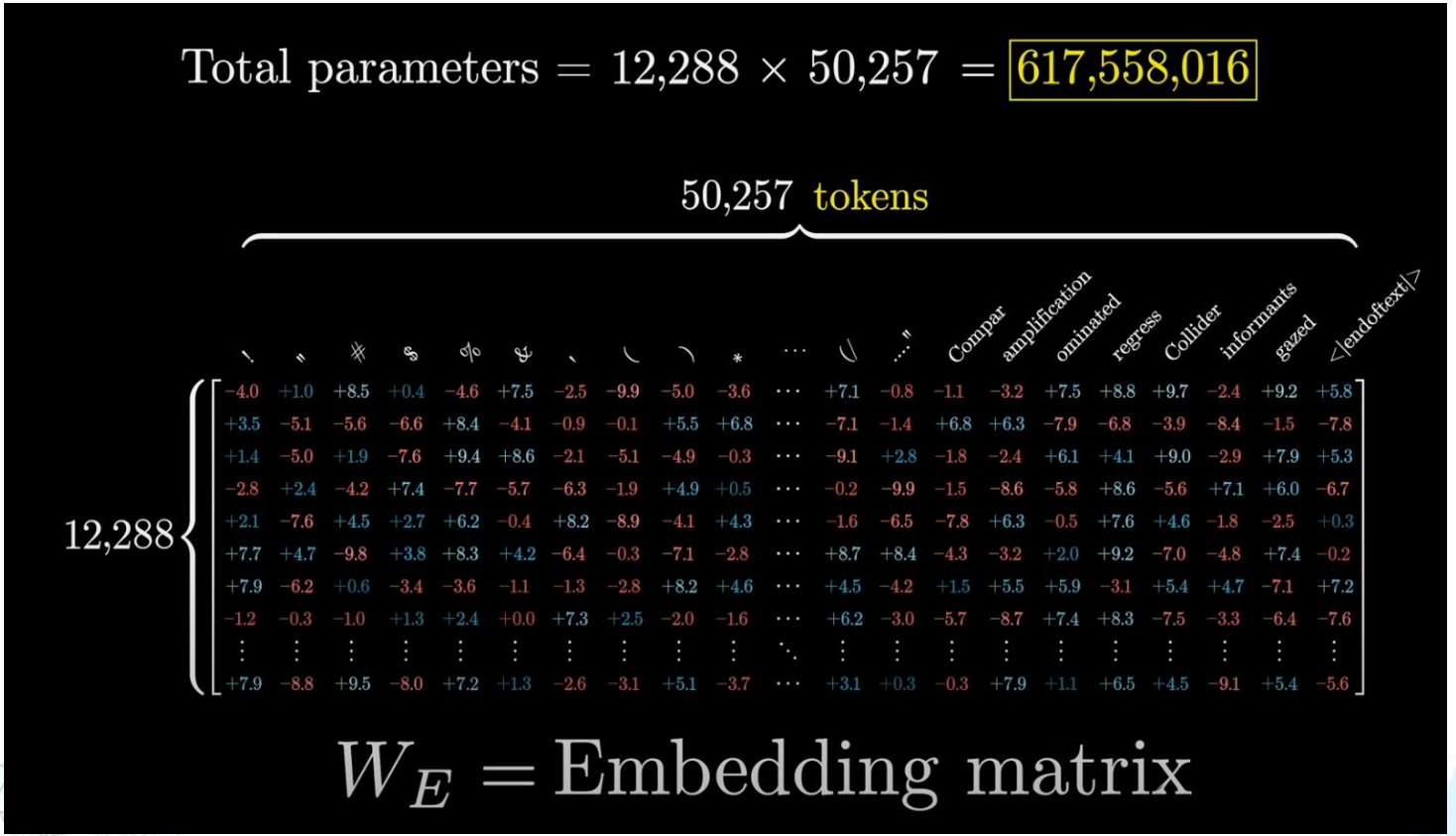


Input Layer

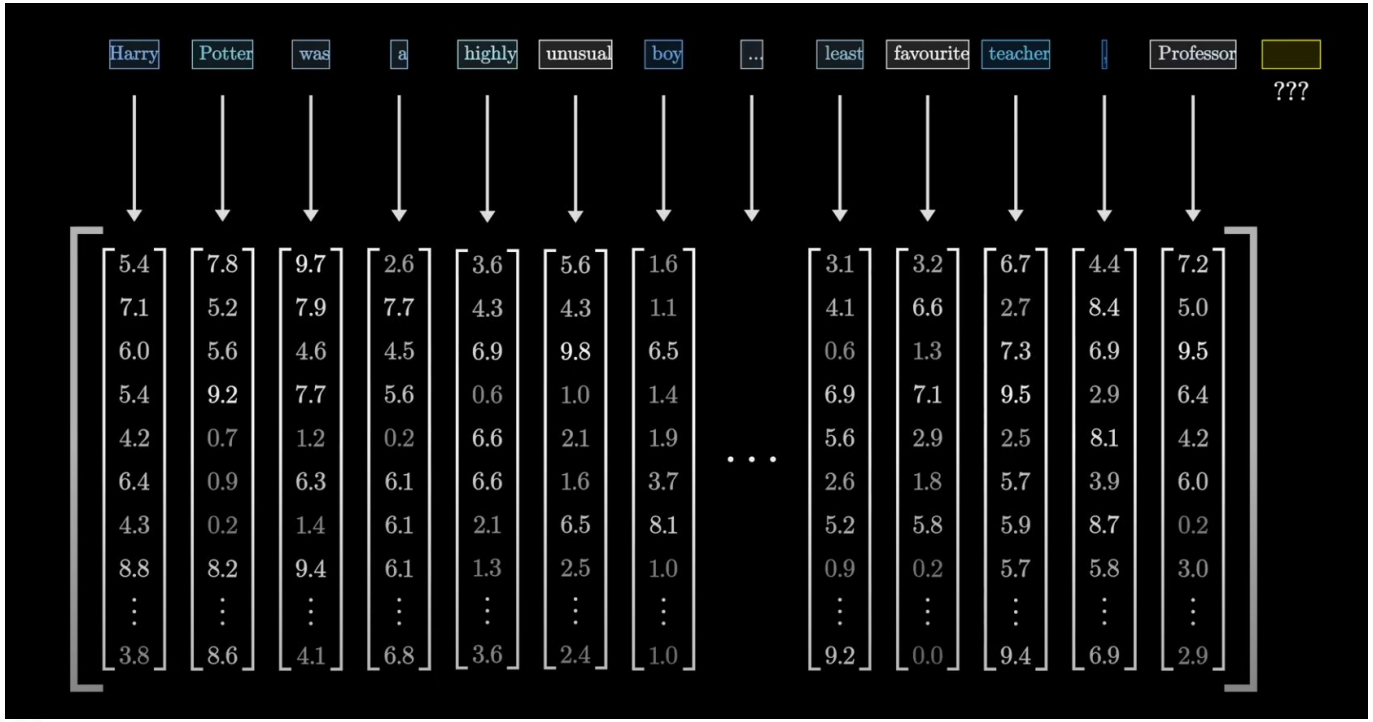
- Once the input split into tokens, they are converted into embeddings.
- Embeddings: Numerical representations of words, phrases, or other units of text in a continuous vector space.



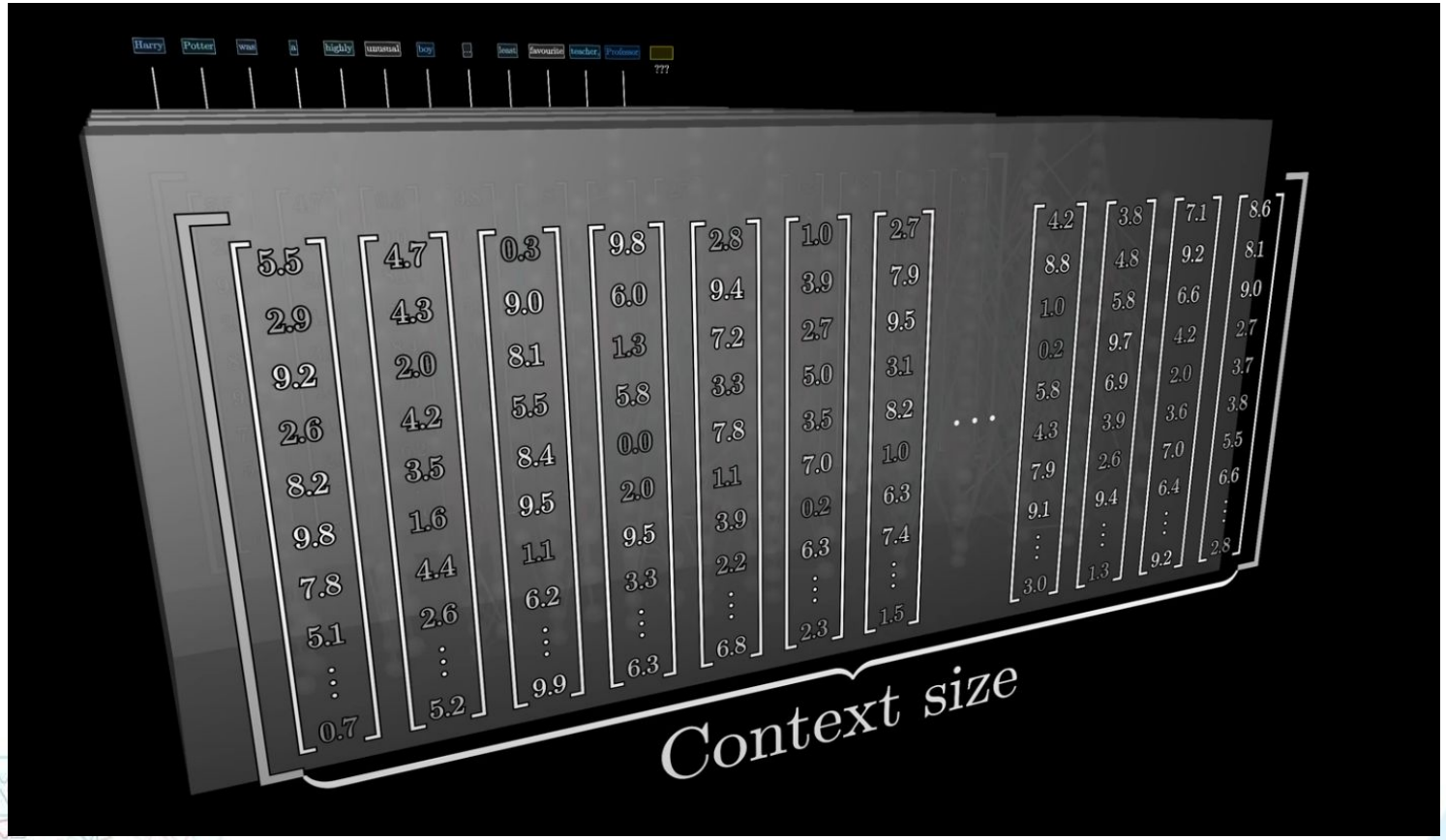
Embedding Matrix



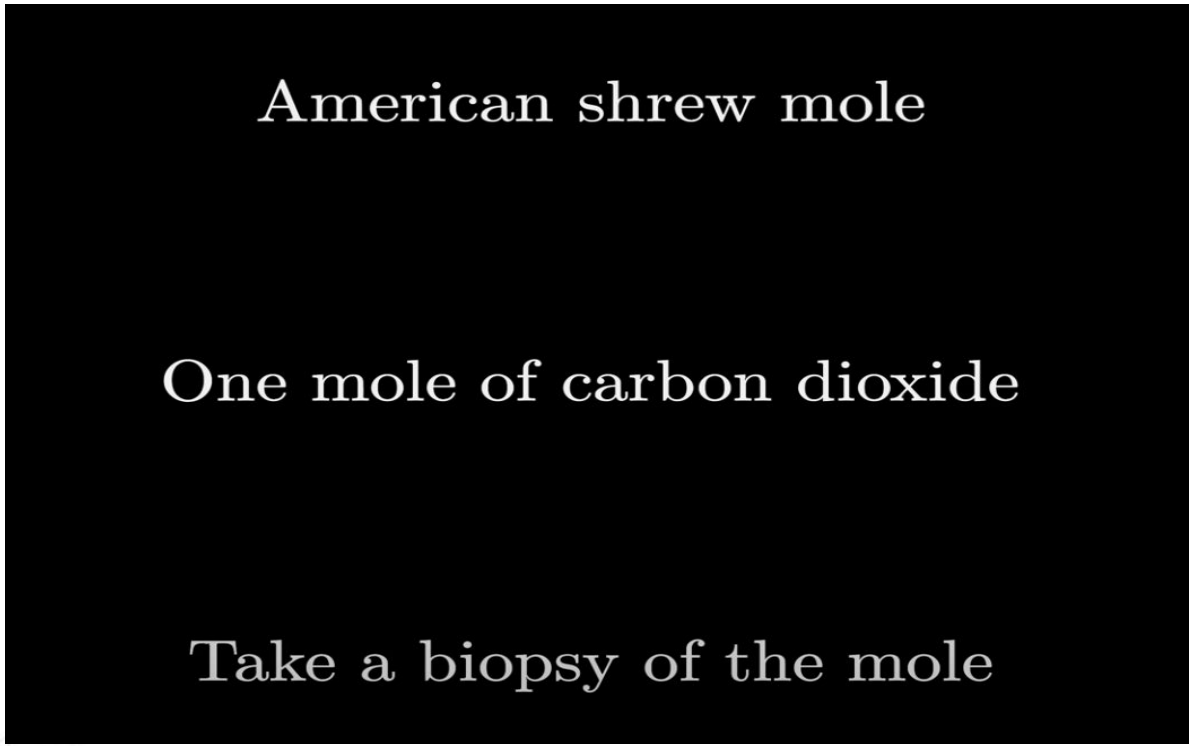
Vector Embeddings

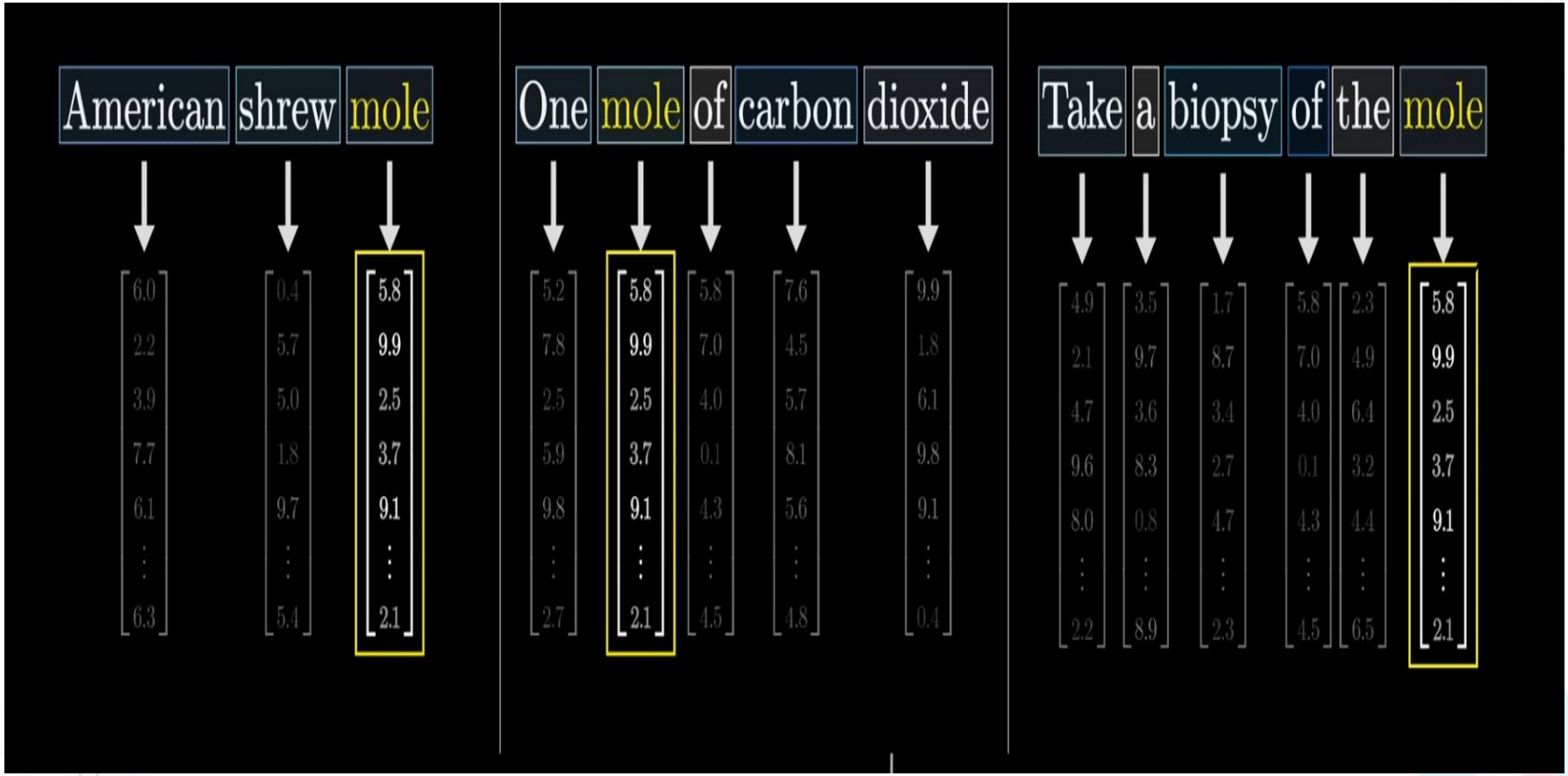


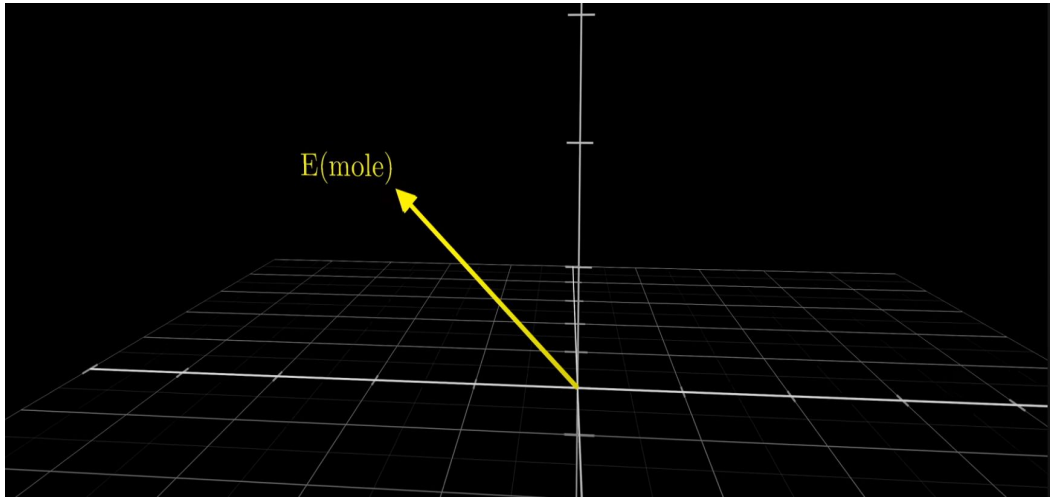
Context Length

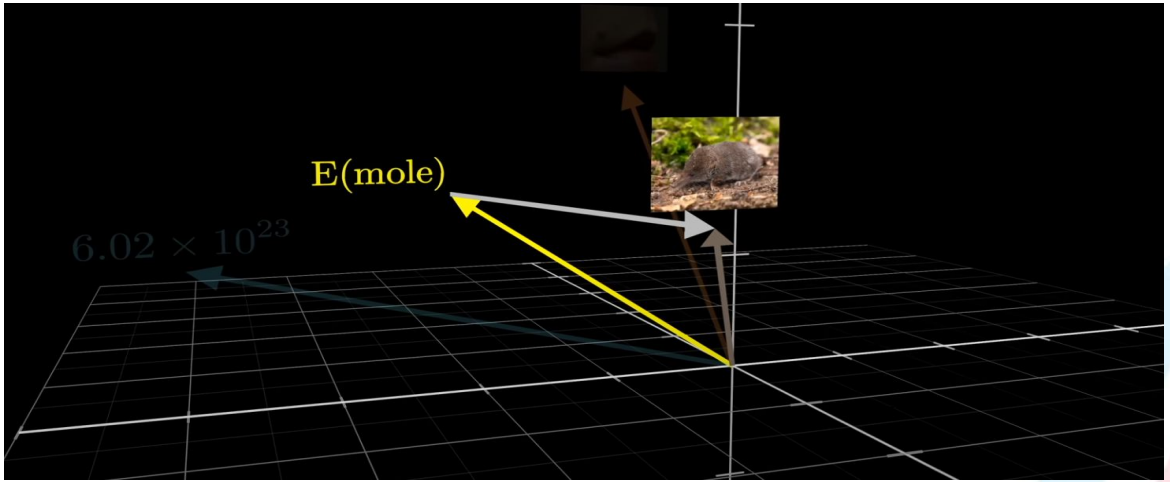


Attention Layer

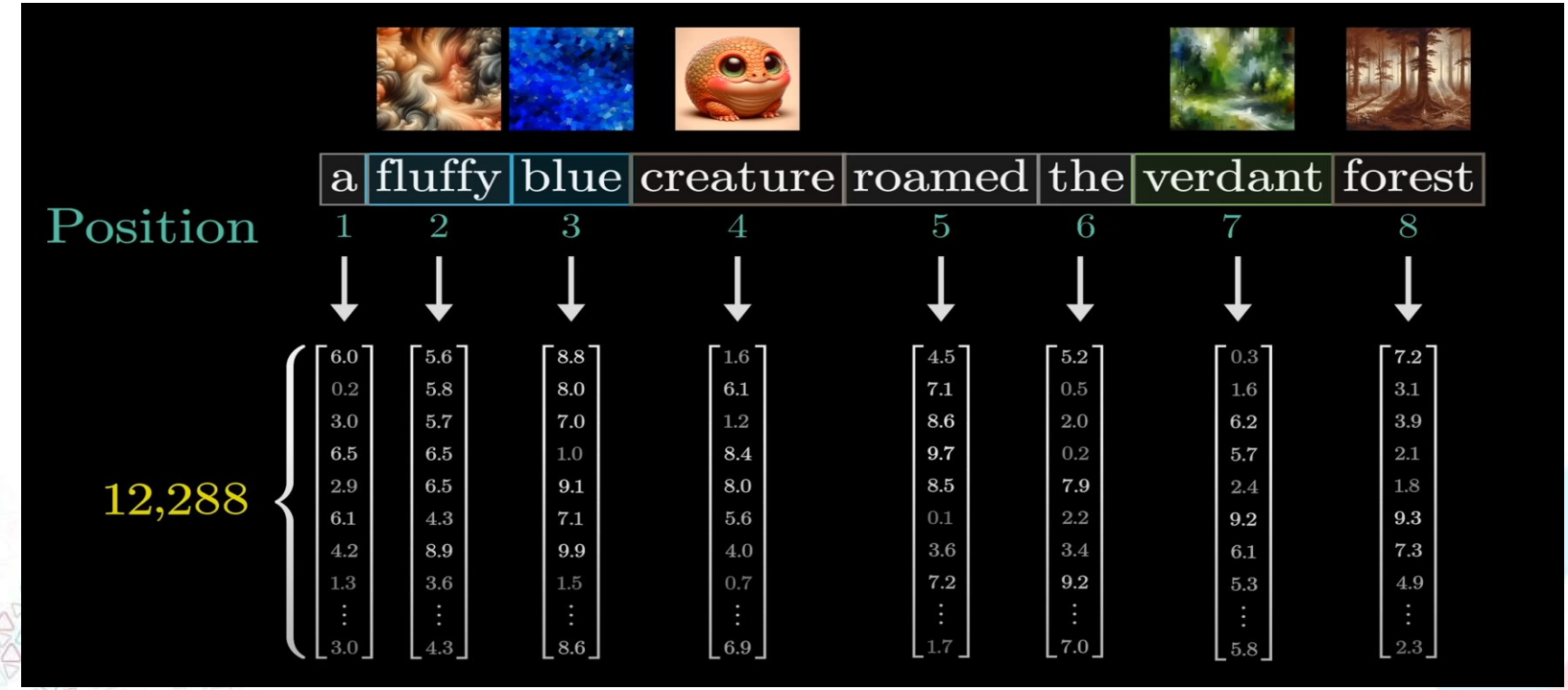


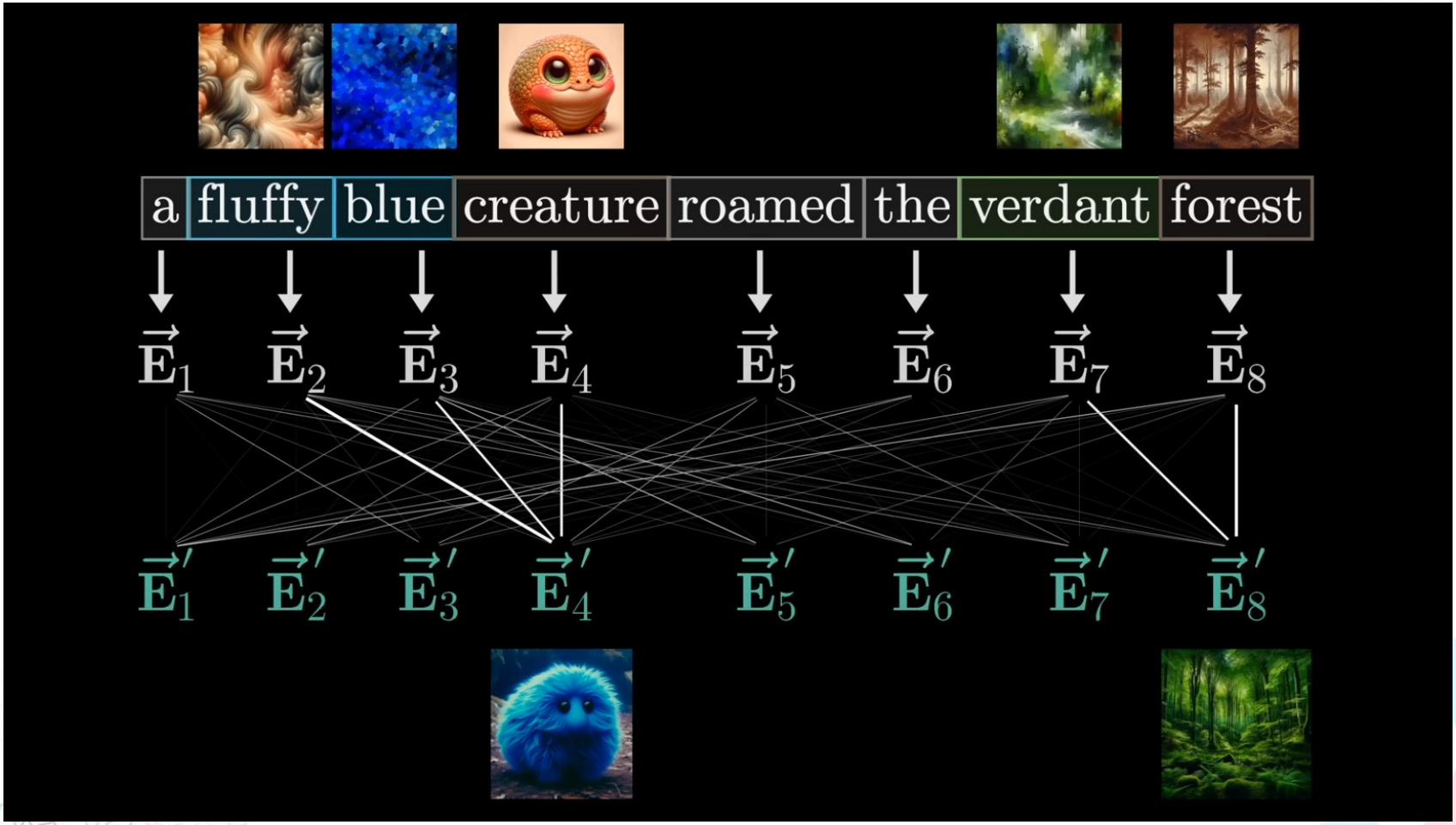

American shrew mole
One mole of carbon dioxide
Take a biopsy of the mole
6.0 2.2 3.9 7.7 6.1 ⋮ 6.3
0.4 5.7 5.0 1.8 9.7 ⋮ 5.4
5.8 9.9 2.5 3.7 9.1 ⋮ 2.1
5.2 7.8 2.5 5.9 9.8 ⋮ 2.7
5.8 9.9 2.5 3.7 9.1 ⋮ 2.1
5.8 7.0 4.0 0.1 4.3 ⋮ 4.5
7.6 4.5 5.7 8.1 5.6 ⋮ 4.8
9.9 1.8 6.1 9.8 9.1 ⋮ 0.4
4.9 2.1 4.7 9.6 8.0 ⋮ 2.2
3.5 9.7 3.6 8.3 0.8 ⋮ 8.9
1.7 8.7 3.4 2.7 4.7 ⋮ 2.3
5.8 7.0 4.0 0.1 4.3 ⋮ 4.5
2.3 4.9 6.4 3.2 4.4 ⋮ 6.5
5.8 9.9 2.5 3.7 9.1 ⋮ 2.1


E(mole)

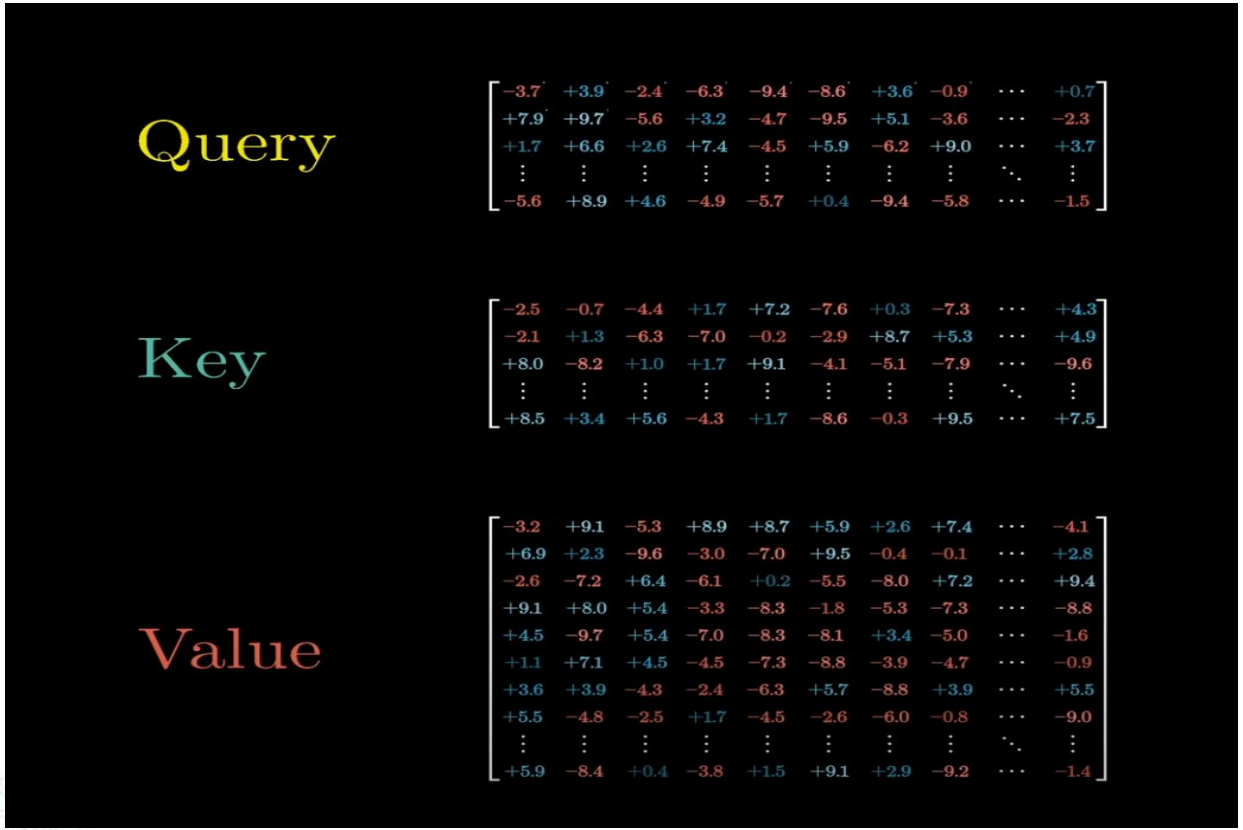

E(mole)
6.02×10^{23}


a fluffy blue creature roamed the verdant forest


a fluffy blue creature roamed the verdant forest
Position
1 2 3 4 5 6 7 8
12,288
6.0 0.2 3.0 6.5 2.9 6.1 4.2 1.3 ⋮ 3.0
5.6 5.8 5.7 6.5 6.5 4.3 8.9 3.6 ⋮ 4.3
8.8 8.0 7.0 1.0 9.1 7.1 9.9 1.5 ⋮ 8.6
1.6 6.1 1.2 8.4 8.0 5.6 4.0 0.7 ⋮ 6.9
4.5 7.1 8.6 9.7 8.5 0.1 3.6 7.2 ⋮ 1.7
5.2 0.5 2.0 0.2 7.9 2.2 3.4 9.2 ⋮ 7.0
0.3 1.6 6.2 5.7 2.4 9.2 6.1 5.3 ⋮ 5.8
7.2 3.1 3.9 2.1 1.8 9.3 7.3 4.9 ⋮ 2.3


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
$\vec{E}_1'$ $\vec{E}_2'$ $\vec{E}_3'$ $\vec{E}_4'$ $\vec{E}_5'$ $\vec{E}_6'$ $\vec{E}_7'$ $\vec{E}_8'$

Query, Key, Value Matrix

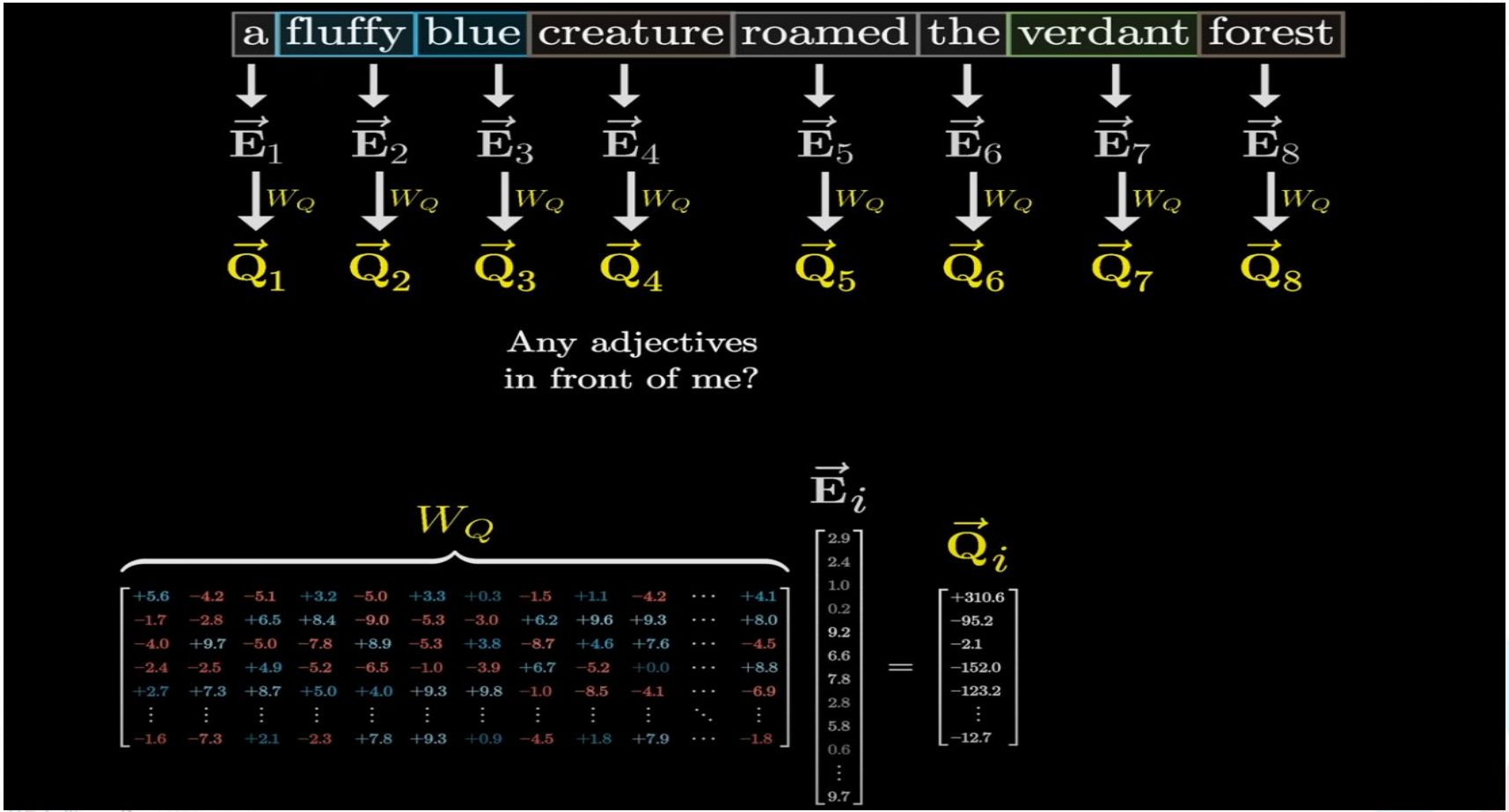


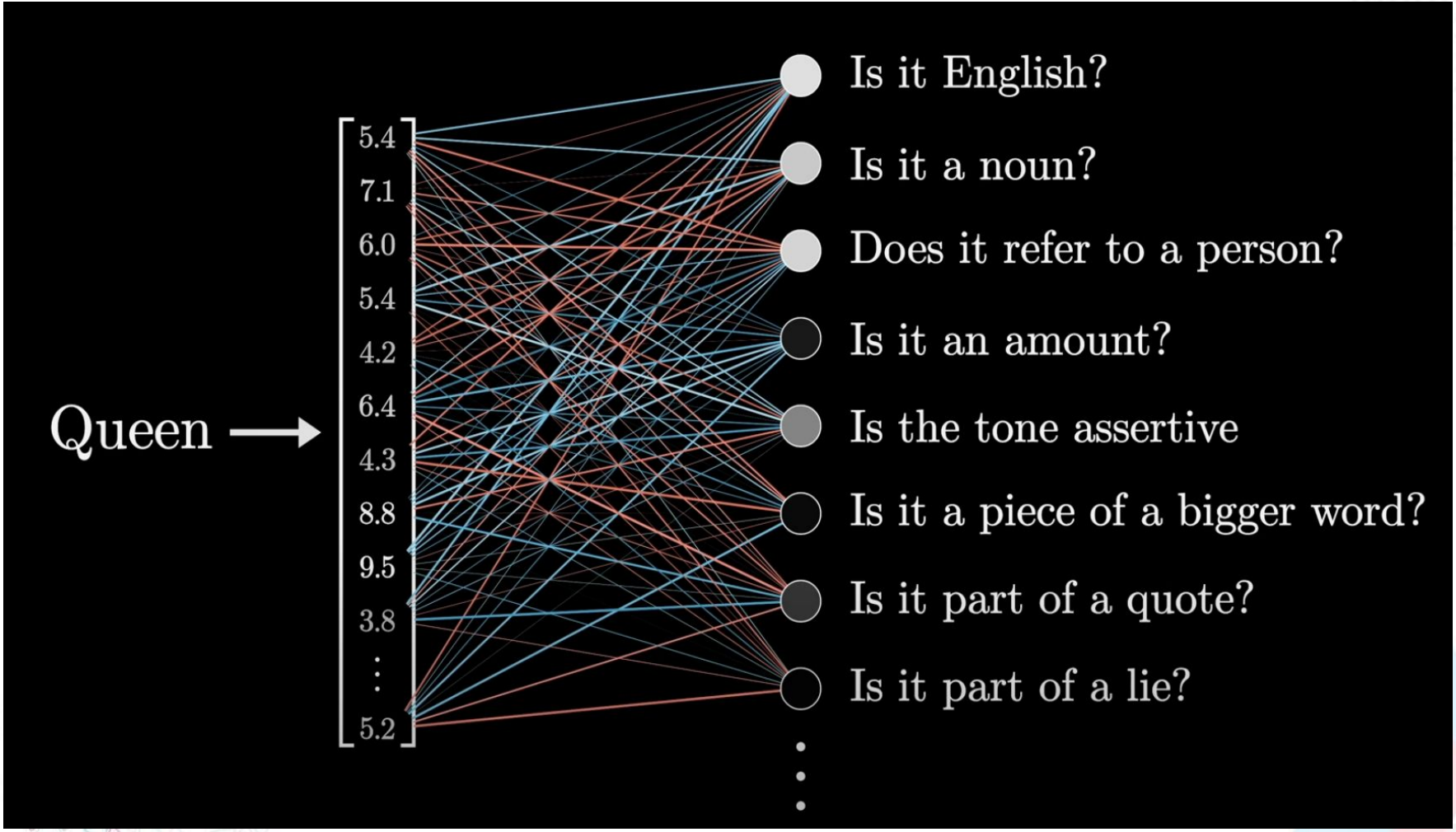
Query, Key, Value Matrix

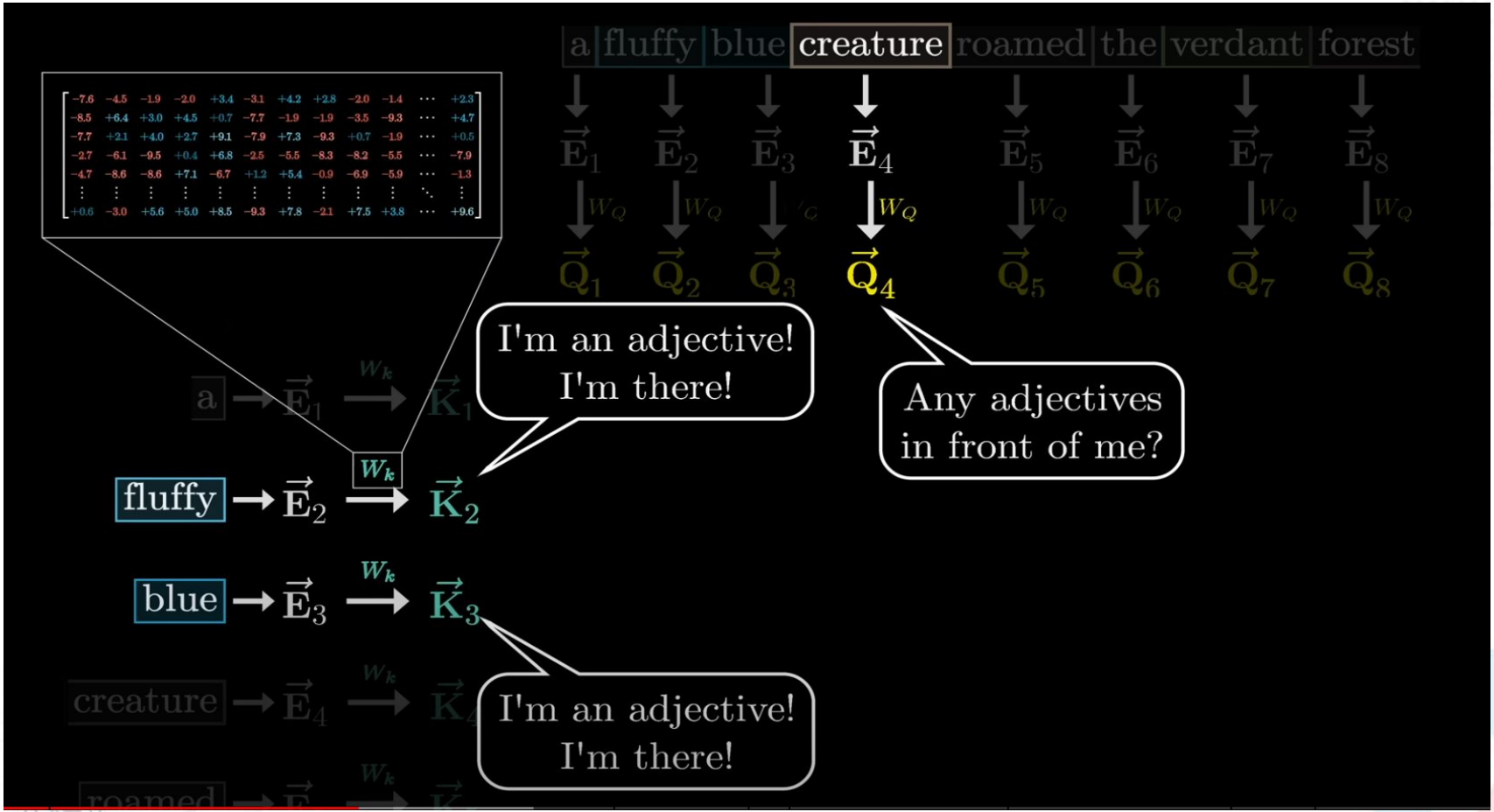
When you process a batch of tokens:

- Input tokens are stacked into a matrix (X).
- Each is multiplied with learnable weight matrices W^Q, W^K, W^V to get:
  - $Q = XW^Q$
  - $K = XW^K$
  - $V = XW^v$

These matrices W^Q, W^K, W^V are trained during model training.These are just single-layer feed-forward networks (dense or linear layers) with no activation function.

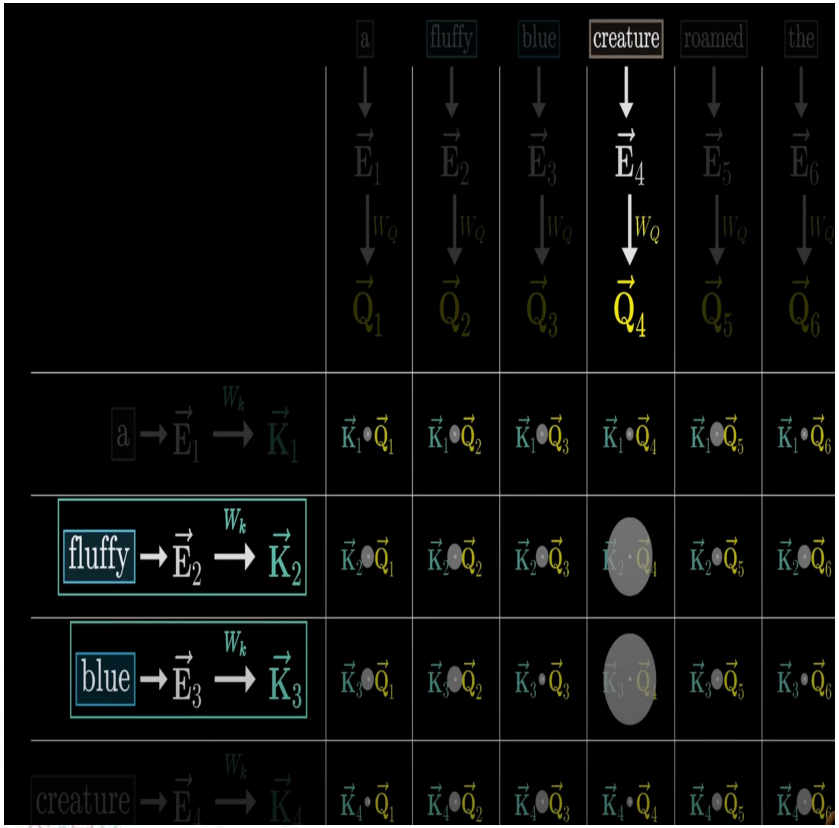

a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
Any adjectives in front of me?
W_Q
$\vec{E}_i$
$\vec{Q}_i$

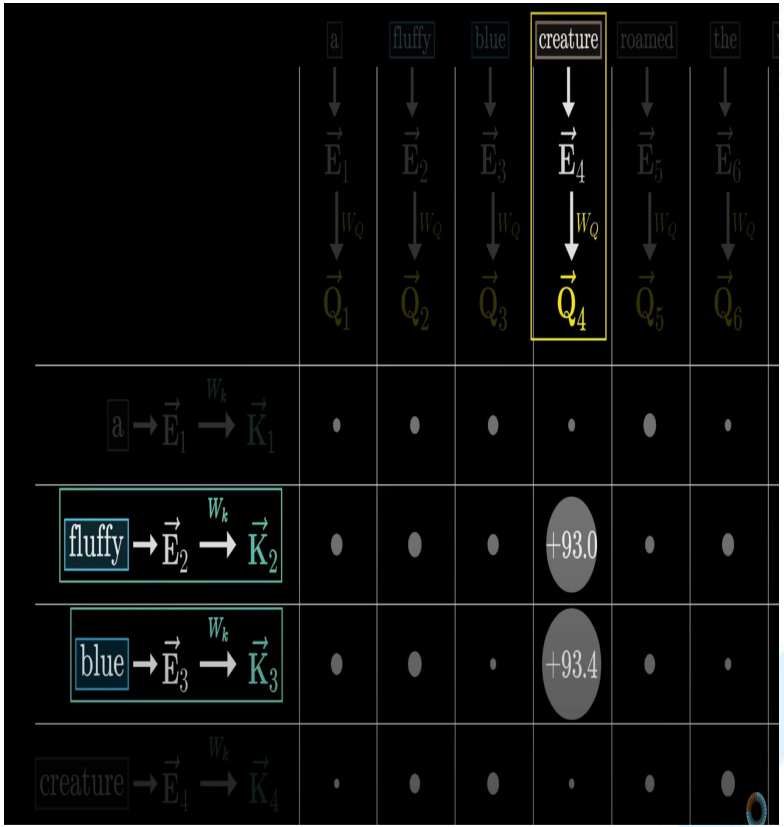

Queen
5.4
7.1
6.0
5.4
4.2
6.4
4.3
8.8
9.5
3.8
5.2
Is it English?
Is it a noun?
Does it refer to a person?
Is it an amount?
Is the tone assertive
Is it a piece of a bigger word?
Is it part of a quote?
Is it part of a lie?


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$ $\vec{Q}_7$ $\vec{Q}_8$
Any adjectives in front of me?
I'm an adjective! I'm there!
W_k
a $\vec{E}_1$ $\vec{K}_1$
fluffy $\vec{E}_2$ $\vec{K}_2$
blue $\vec{E}_3$ $\vec{K}_3$
creature $\vec{E}_4$ $\vec{K}_4$
I'm an adjective! I'm there!

Attention Pattern

	a	fluffy	blue	creature	roamed	the	verdant	forest
	$\vec{E}_1 \xrightarrow{W_Q} \vec{Q}_1$	$\vec{E}_2 \xrightarrow{W_Q} \vec{Q}_2$	$\vec{E}_3 \xrightarrow{W_Q} \vec{Q}_3$	$\vec{E}_4 \xrightarrow{W_Q} \vec{Q}_4$	$\vec{E}_5 \xrightarrow{W_Q} \vec{Q}_5$	$\vec{E}_6 \xrightarrow{W_Q} \vec{Q}_6$	$\vec{E}_7 \xrightarrow{W_Q} \vec{Q}_7$	$\vec{E}_8 \xrightarrow{W_Q} \vec{Q}_8$
a $\rightarrow \vec{E}_1 \xrightarrow{W_K} \vec{K}_1$	$\vec{K}_1 \cdot \vec{Q}_1$	$\vec{K}_1 \cdot \vec{Q}_2$	$\vec{K}_1 \cdot \vec{Q}_3$	$\vec{K}_1 \cdot \vec{Q}_4$	$\vec{K}_1 \cdot \vec{Q}_5$	$\vec{K}_1 \cdot \vec{Q}_6$	$\vec{K}_1 \cdot \vec{Q}_7$	$\vec{K}_1 \cdot \vec{Q}_8$
fluffy $\rightarrow \vec{E}_2 \xrightarrow{W_K} \vec{K}_2$	$\vec{K}_2 \cdot \vec{Q}_1$	$\vec{K}_2 \cdot \vec{Q}_2$	$\vec{K}_2 \cdot \vec{Q}_3$	$\vec{K}_2 \cdot \vec{Q}_4$	$\vec{K}_2 \cdot \vec{Q}_5$	$\vec{K}_2 \cdot \vec{Q}_6$	$\vec{K}_2 \cdot \vec{Q}_7$	$\vec{K}_2 \cdot \vec{Q}_8$
blue $\rightarrow \vec{E}_3 \xrightarrow{W_K} \vec{K}_3$	$\vec{K}_3 \cdot \vec{Q}_1$	$\vec{K}_3 \cdot \vec{Q}_2$	$\vec{K}_3 \cdot \vec{Q}_3$	$\vec{K}_3 \cdot \vec{Q}_4$	$\vec{K}_3 \cdot \vec{Q}_5$	$\vec{K}_3 \cdot \vec{Q}_6$	$\vec{K}_3 \cdot \vec{Q}_7$	$\vec{K}_3 \cdot \vec{Q}_8$
creature $\rightarrow \vec{E}_4 \xrightarrow{W_K} \vec{K}_4$	$\vec{K}_4 \cdot \vec{Q}_1$	$\vec{K}_4 \cdot \vec{Q}_2$	$\vec{K}_4 \cdot \vec{Q}_3$	$\vec{K}_4 \cdot \vec{Q}_4$	$\vec{K}_4 \cdot \vec{Q}_5$	$\vec{K}_4 \cdot \vec{Q}_6$	$\vec{K}_4 \cdot \vec{Q}_7$	$\vec{K}_4 \cdot \vec{Q}_8$
roamed $\rightarrow \vec{E}_5 \xrightarrow{W_K} \vec{K}_5$	$\vec{K}_5 \cdot \vec{Q}_1$	$\vec{K}_5 \cdot \vec{Q}_2$	$\vec{K}_5 \cdot \vec{Q}_3$	$\vec{K}_5 \cdot \vec{Q}_4$	$\vec{K}_5 \cdot \vec{Q}_5$	$\vec{K}_5 \cdot \vec{Q}_6$	$\vec{K}_5 \cdot \vec{Q}_7$	$\vec{K}_5 \cdot \vec{Q}_8$
the $\rightarrow \vec{E}_6 \xrightarrow{W_K} \vec{K}_6$	$\vec{K}_6 \cdot \vec{Q}_1$	$\vec{K}_6 \cdot \vec{Q}_2$	$\vec{K}_6 \cdot \vec{Q}_3$	$\vec{K}_6 \cdot \vec{Q}_4$	$\vec{K}_6 \cdot \vec{Q}_5$	$\vec{K}_6 \cdot \vec{Q}_6$	$\vec{K}_6 \cdot \vec{Q}_7$	$\vec{K}_6 \cdot \vec{Q}_8$
verdant $\rightarrow \vec{E}_7 \xrightarrow{W_K} \vec{K}_7$	$\vec{K}_7 \cdot \vec{Q}_1$	$\vec{K}_7 \cdot \vec{Q}_2$	$\vec{K}_7 \cdot \vec{Q}_3$	$\vec{K}_7 \cdot \vec{Q}_4$	$\vec{K}_7 \cdot \vec{Q}_5$	$\vec{K}_7 \cdot \vec{Q}_6$	$\vec{K}_7 \cdot \vec{Q}_7$	$\vec{K}_7 \cdot \vec{Q}_8$


a
fluffy
blue
creature
roamed
the
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$
a → $\vec{E}_1$ —W_k→ $\vec{K}_1$
fluffy → $\vec{E}_2$ —W_k→ $\vec{K}_2$
blue → $\vec{E}_3$ —W_k→ $\vec{K}_3$
creature → $\vec{E}_4$ —W_k→ $\vec{K}_4$
$\vec{K}_1 \cdot \vec{Q}_1$ $\vec{K}_1 \cdot \vec{Q}_2$ $\vec{K}_1 \cdot \vec{Q}_3$ $\vec{K}_1 \cdot \vec{Q}_4$ $\vec{K}_1 \cdot \vec{Q}_5$ $\vec{K}_1 \cdot \vec{Q}_6$
$\vec{K}_2 \cdot \vec{Q}_1$ $\vec{K}_2 \cdot \vec{Q}_2$ $\vec{K}_2 \cdot \vec{Q}_3$ $\vec{K}_2 \cdot \vec{Q}_4$ $\vec{K}_2 \cdot \vec{Q}_5$ $\vec{K}_2 \cdot \vec{Q}_6$
$\vec{K}_3 \cdot \vec{Q}_1$ $\vec{K}_3 \cdot \vec{Q}_2$ $\vec{K}_3 \cdot \vec{Q}_3$ $\vec{K}_3 \cdot \vec{Q}_4$ $\vec{K}_3 \cdot \vec{Q}_5$ $\vec{K}_3 \cdot \vec{Q}_6$
$\vec{K}_4 \cdot \vec{Q}_1$ $\vec{K}_4 \cdot \vec{Q}_2$ $\vec{K}_4 \cdot \vec{Q}_3$ $\vec{K}_4 \cdot \vec{Q}_4$ $\vec{K}_4 \cdot \vec{Q}_5$ $\vec{K}_4 \cdot \vec{Q}_6$


a
fluffy
blue
creature
roamed
the
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$
W_Q
$\vec{Q}_1$ $\vec{Q}_2$ $\vec{Q}_3$ $\vec{Q}_4$ $\vec{Q}_5$ $\vec{Q}_6$
a → $\vec{E}_1$ —W_k→ $\vec{K}_1$
fluffy → $\vec{E}_2$ —W_k→ $\vec{K}_2$
blue → $\vec{E}_3$ —W_k→ $\vec{K}_3$
creature → $\vec{E}_4$ —W_k→ $\vec{K}_4$
+93.0
+93.4

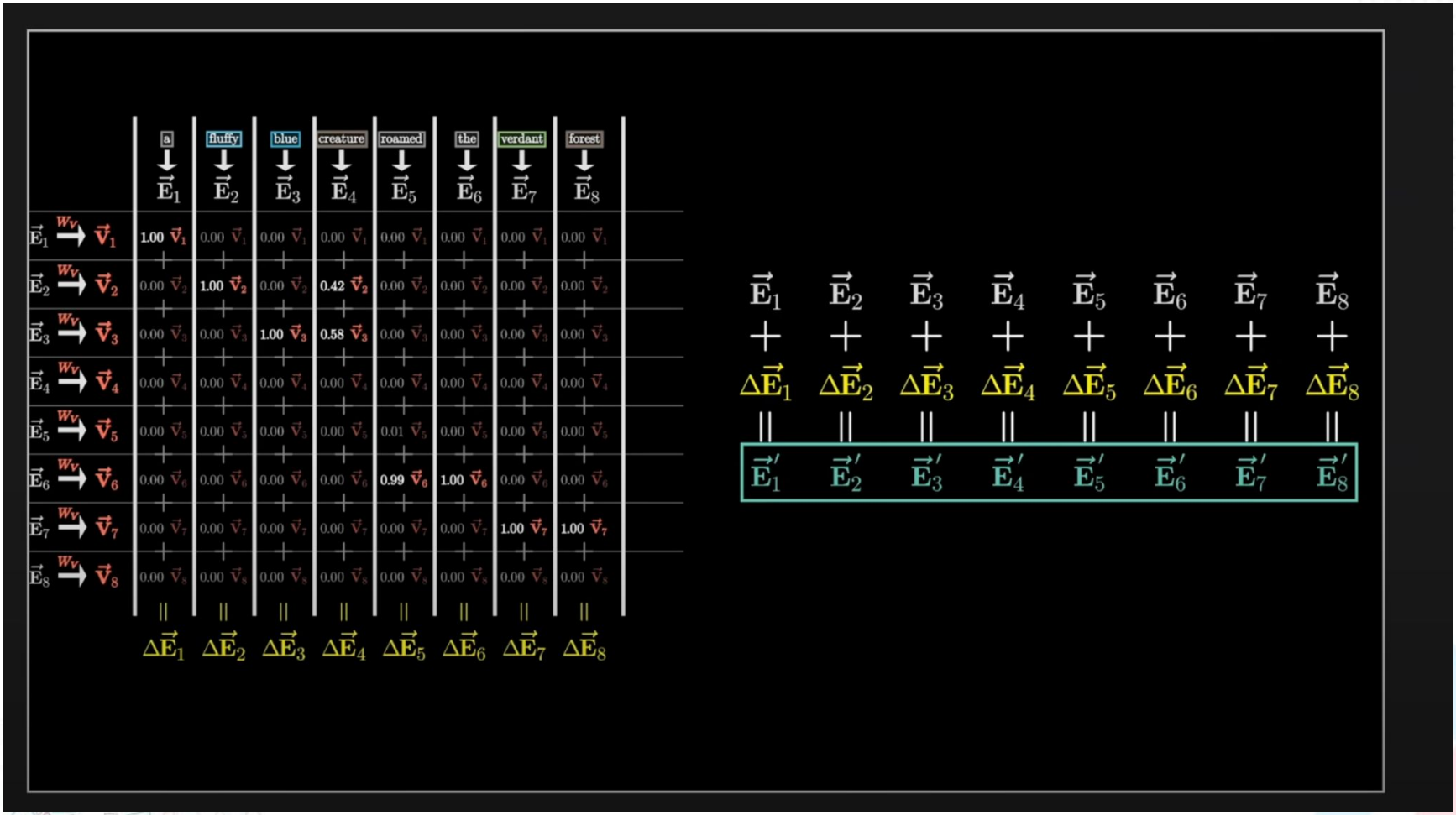
	a	fluffy	blue	creature	roamed	the	verdant	forest
	$\vec{E}_1 \xrightarrow{W_Q} \vec{Q}_1$	$\vec{E}_2 \xrightarrow{W_Q} \vec{Q}_2$	$\vec{E}_3 \xrightarrow{W_Q} \vec{Q}_3$	$\vec{E}_4 \xrightarrow{W_Q} \vec{Q}_4$	$\vec{E}_5 \xrightarrow{W_Q} \vec{Q}_5$	$\vec{E}_6 \xrightarrow{W_Q} \vec{Q}_6$	$\vec{E}_7 \xrightarrow{W_Q} \vec{Q}_7$	$\vec{E}_8 \xrightarrow{W_Q} \vec{Q}_8$
a $\rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	+0.7	−83.7	−24.7	−27.8	−5.2	−89.3	−45.2	−36.1
fluffy $\rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	−73.4	+2.9	−5.4	+93.0	−48.2	−87.3	−49.7	+7.8
blue $\rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	−53.4	−5.7	+1.8	+93.4	−55.6	−56.0	−26.1	−62.1
creature $\rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	−21.5	−29.7	−56.1	+4.9	−32.4	−92.3	−9.5	−28.1
roamed $\rightarrow \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$	−20.1	−40.9	−87.8	−55.4	+0.6	−64.7	−96.7	−18.9
the $\rightarrow \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$	−87.9	−33.3	−22.6	−31.4	+5.5	+0.6	−4.6	−96.8
verdant $\rightarrow \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$	−41.2	−55.5	−42.3	−59.8	−79.0	−97.9	+3.7	+93.8
forest $\rightarrow \vec{E}_8 \xrightarrow{W_k} \vec{K}_8$	−58.9	−75.5	−91.1	−90.6	−75.6	−89.0	−70.8	+4.7

	a	fluffy	blue	creature	roamed	the	verdant	forest
	$\vec{E}_1 \xrightarrow{W_Q} \vec{Q}_1$	$\vec{E}_2 \xrightarrow{W_Q} \vec{Q}_2$	$\vec{E}_3 \xrightarrow{W_Q} \vec{Q}_3$	$\vec{E}_4 \xrightarrow{W_Q} \vec{Q}_4$	$\vec{E}_5 \xrightarrow{W_Q} \vec{Q}_5$	$\vec{E}_6 \xrightarrow{W_Q} \vec{Q}_6$	$\vec{E}_7 \xrightarrow{W_Q} \vec{Q}_7$	$\vec{E}_8 \xrightarrow{W_Q} \vec{Q}_8$
a $\rightarrow \vec{E}_1 \xrightarrow{W_k} \vec{K}_1$	1.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
fluffy $\rightarrow \vec{E}_2 \xrightarrow{W_k} \vec{K}_2$	0.00	1.00	0.00	0.42	0.00	0.00	0.00	0.00
blue $\rightarrow \vec{E}_3 \xrightarrow{W_k} \vec{K}_3$	0.00	0.00	1.00	0.58	0.00	0.00	0.00	0.00
creature $\rightarrow \vec{E}_4 \xrightarrow{W_k} \vec{K}_4$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00
roamed $\rightarrow \vec{E}_5 \xrightarrow{W_k} \vec{K}_5$	0.00	0.00	0.00	0.00	0.01	0.00	0.00	0.00
the $\rightarrow \vec{E}_6 \xrightarrow{W_k} \vec{K}_6$	0.00	0.00	0.00	0.00	0.99	1.00	0.00	0.00
verdant $\rightarrow \vec{E}_7 \xrightarrow{W_k} \vec{K}_7$	0.00	0.00	0.00	0.00	0.00	0.00	1.00	1.00
forest $\rightarrow \vec{E}_8 \xrightarrow{W_k} \vec{K}_8$	0.00	0.00	0.00	0.00	0.00	0.00	0.00	0.00

Value matrix W_V

	a $\vec{E}_1$	fluffy $\vec{E}_2$	blue $\vec{E}_3$	creature $\vec{E}_4$	roamed $\vec{E}_5$	the $\vec{E}_6$	verdant $\vec{E}_7$	forest $\vec{E}_8$
a $\vec{E}_1 \xrightarrow{W_V} \vec{V}_1$				0.00 $\vec{V}_1$				
fluffy $\vec{E}_2 \xrightarrow{W_V} \vec{V}_2$				0.42 $\vec{V}_2$				
blue $\vec{E}_3 \xrightarrow{W_V} \vec{V}_3$				0.58 $\vec{V}_3$				
creature $\vec{E}_4 \xrightarrow{W_V} \vec{V}_4$				0.00 $\vec{V}_4$				
roamed $\vec{E}_5 \xrightarrow{W_V} \vec{V}_5$				0.00 $\vec{V}_5$				
the $\vec{E}_6 \xrightarrow{W_V} \vec{V}_6$				0.00 $\vec{V}_6$				
verdant $\vec{E}_7 \xrightarrow{W_V} \vec{V}_7$				0.00 $\vec{V}_7$				


a
fluffy
blue
creature
roamed
the
verdant
forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
a → $\vec{E}_1$ W_V → $\vec{V}_1$
fluffy → $\vec{E}_2$ W_V → $\vec{V}_2$
blue → $\vec{E}_3$ W_V → $\vec{V}_3$
creature → $\vec{E}_4$ W_V → $\vec{V}_4$
roamed → $\vec{E}_5$ W_V → $\vec{V}_5$
the → $\vec{E}_6$ W_V → $\vec{V}_6$
verdant → $\vec{E}_7$ W_V → $\vec{V}_7$
forest → $\vec{E}_8$ W_V → $\vec{V}_8$
0.00 $\vec{V}_1$
+
0.42 $\vec{V}_2$
+
0.58 $\vec{V}_3$
+
0.00 $\vec{V}_4$
+
0.00 $\vec{V}_5$
+
0.00 $\vec{V}_6$
+
0.00 $\vec{V}_7$
+
0.00 $\vec{V}_8$
=
$\Delta\vec{E}_4$


a fluffy blue creature roamed the verdant forest
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
$\vec{E}_1 \xrightarrow{W_V} \vec{V}_1$ 1.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$ 0.00 $\vec{V}_1$
$\vec{E}_2 \xrightarrow{W_V} \vec{V}_2$ 0.00 $\vec{V}_2$ 1.00 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.42 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.00 $\vec{V}_2$ 0.00 $\vec{V}_2$
$\vec{E}_3 \xrightarrow{W_V} \vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$ 1.00 $\vec{V}_3$ 0.58 $\vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$ 0.00 $\vec{V}_3$
$\vec{E}_4 \xrightarrow{W_V} \vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$ 0.00 $\vec{V}_4$
$\vec{E}_5 \xrightarrow{W_V} \vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.01 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$ 0.00 $\vec{V}_5$
$\vec{E}_6 \xrightarrow{W_V} \vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.99 $\vec{V}_6$ 1.00 $\vec{V}_6$ 0.00 $\vec{V}_6$ 0.00 $\vec{V}_6$
$\vec{E}_7 \xrightarrow{W_V} \vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 0.00 $\vec{V}_7$ 1.00 $\vec{V}_7$ 1.00 $\vec{V}_7$
$\vec{E}_8 \xrightarrow{W_V} \vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$ 0.00 $\vec{V}_8$
$\Delta\vec{E}_1$ $\Delta\vec{E}_2$ $\Delta\vec{E}_3$ $\Delta\vec{E}_4$ $\Delta\vec{E}_5$ $\Delta\vec{E}_6$ $\Delta\vec{E}_7$ $\Delta\vec{E}_8$
$\vec{E}_1$ $\vec{E}_2$ $\vec{E}_3$ $\vec{E}_4$ $\vec{E}_5$ $\vec{E}_6$ $\vec{E}_7$ $\vec{E}_8$
$\Delta\vec{E}_1$ $\Delta\vec{E}_2$ $\Delta\vec{E}_3$ $\Delta\vec{E}_4$ $\Delta\vec{E}_5$ $\Delta\vec{E}_6$ $\Delta\vec{E}_7$ $\Delta\vec{E}_8$
$\vec{E}_1'$ $\vec{E}_2'$ $\vec{E}_3'$ $\vec{E}_4'$ $\vec{E}_5'$ $\vec{E}_6'$ $\vec{E}_7'$ $\vec{E}_8'$

Let’s calculate the Attention Score

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

Q — Query Matrix
K — Key Matrix
V — Value Matrix
d□ — Dimensionality of Key Matrix

Query

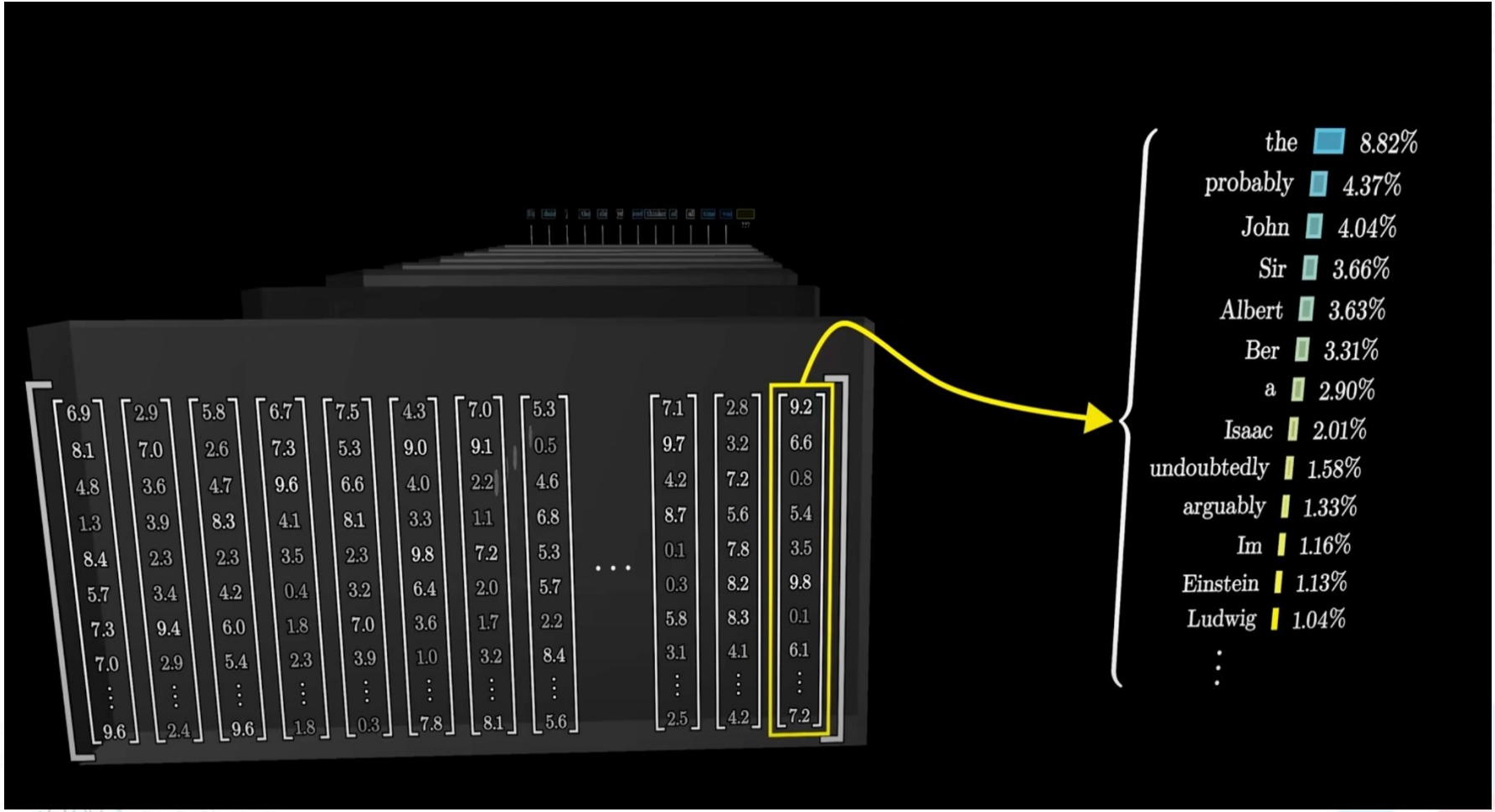
$$128 \times 12{,}288 = 1{,}572{,}864$$

$$128 \begin{cases} \overbrace{\begin{bmatrix} -3.7 & +3.9 & -2.4 & -6.3 & -9.4 & -8.6 & +3.6 & -0.9 & \cdots & +0.7 \\ +7.9 & +9.7 & -5.6 & +3.2 & -4.7 & -9.5 & +5.1 & -3.6 & \cdots & -2.3 \\ +1.7 & +6.6 & +2.6 & +7.4 & -4.5 & +5.9 & -6.2 & +9.0 & \cdots & +3.7 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ -5.6 & +8.9 & +4.6 & -4.9 & -5.7 & +0.4 & -9.4 & -5.8 & \cdots & -1.5 \end{bmatrix}}^{12{,}288} \end{cases}$$

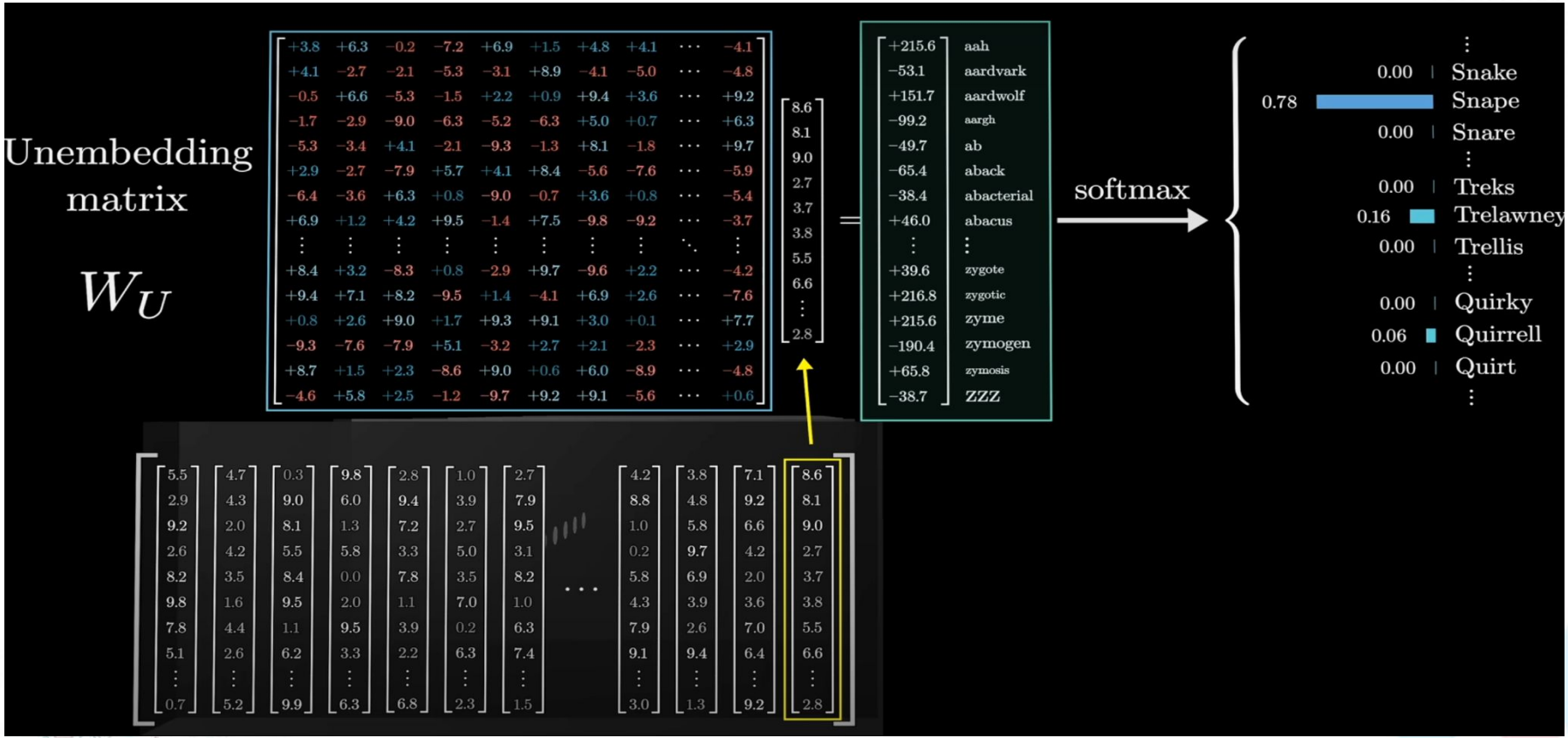
Key

$$128 \times 12{,}288 = 1{,}572{,}864$$

$$128 \begin{cases} \overbrace{\begin{bmatrix} -2.5 & -0.7 & -4.4 & +1.7 & +7.2 & -7.6 & +0.3 & -7.3 & \cdots & +4.3 \\ -2.1 & +1.3 & -6.3 & -7.0 & -0.2 & -2.9 & +8.7 & +5.3 & \cdots & +4.9 \\ +8.0 & -8.2 & +1.0 & +1.7 & +9.1 & -4.1 & -5.1 & -7.9 & \cdots & -9.6 \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \vdots & \ddots & \vdots \\ +8.5 & +3.4 & +5.6 & -4.3 & +1.7 & -8.6 & -0.3 & +9.5 & \cdots & +7.5 \end{bmatrix}}^{12{,}288} \end{cases}$$

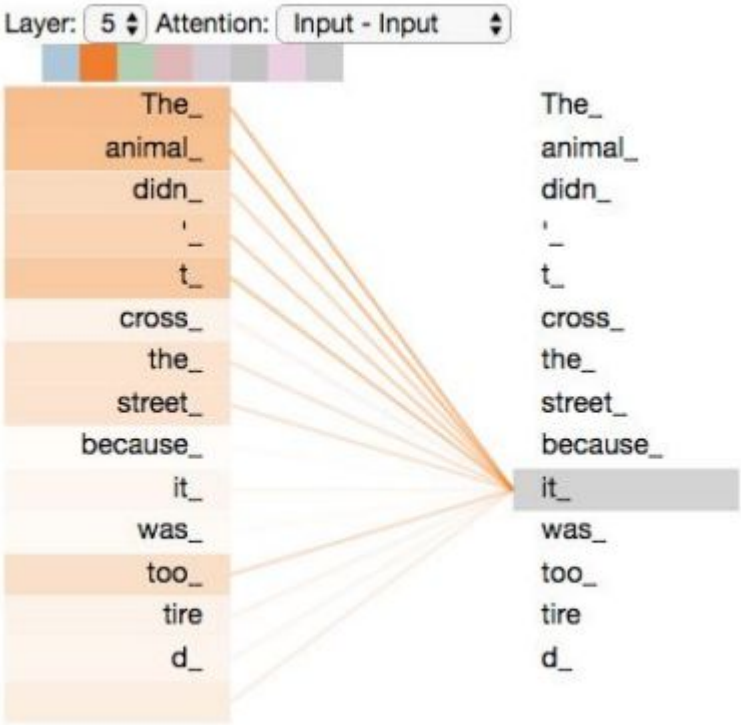

the 8.82%
probably 4.37%
John 4.04%
Sir 3.66%
Albert 3.63%
Ber 3.31%
a 2.90%
Isaac 2.01%
undoubtedly 1.58%
arguably 1.33%
Im 1.16%
Einstein 1.13%
Ludwig 1.04%

Unembedding Matrix



Self-Attention

- Each element attends to every other element
- Attention mechanism relating different positions of a single sequence in order to compute a representation of the same sequence.
- Each element becomes query, key, and value from the input embeddings by multiplying by a weight matrix.



Idea Behind Attention

Goal: Learn (differentiable) how to pick relevant information from input data

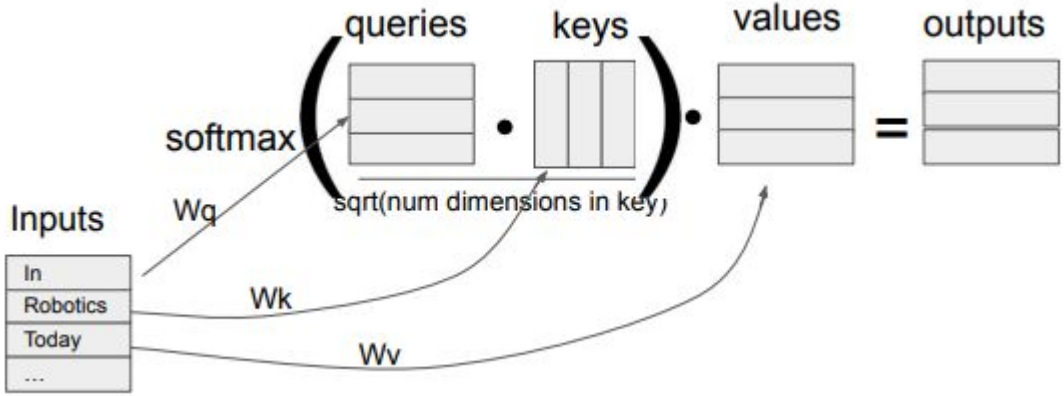
1. Create three vectors from each of the input values (query, key, value)
2. Calculate a score for how much to focus on each part of the input when we encode words at specific positions
3. How?
    a. Idea is to select a value (referenced by a key) relevant to a query (what trying to pull from input)

Scaled Dot Product Attention(sdpa)

- Attention weights: how likely each query matches key (used for weighted sum of values)
- Dividing by d_k makes algorithm easier to train
- then can take loss, backpropagate, update the weights and values
- NOT stepping through a sequence like with a RNN

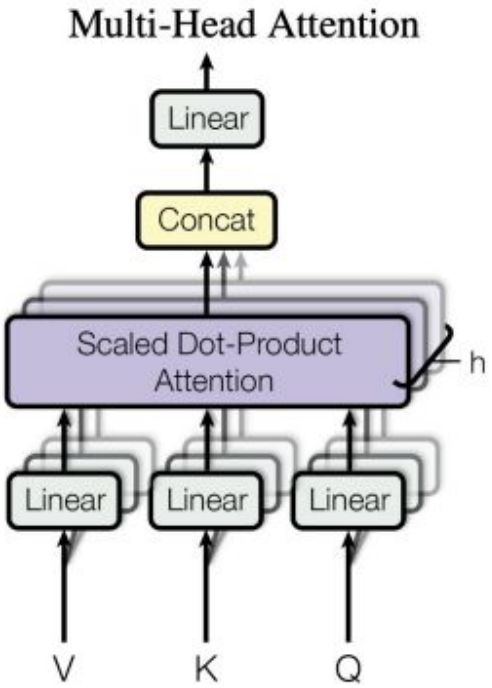
$$\text{Attention}(Q, K, V) = \text{softmax}(\frac{QK^T}{\sqrt{d_k}})V$$

- input consists of queries and keys of dimension dk, and values of dimension d
- Queries packed into matrix Q
- The keys and values are also packed together into matrices K and V .



Multi-Head Attention

- Idea:
  - Stack linear layers (weight matrices without biases) that are independent each for keys, queries, values.
  - Concatenate output of attention heads to form (plus non-linearity) output layer
  - Attention heads are independent of each other
- Why?
  - Allows for model to focus on different positions
  - Gives attention layer multiple “representation subspaces”
  - No longer need to oversaturate one attention mechanism



$$\text{MultiHead}(Q, K, V) = \text{Concat}(\text{head}_1, ..., \text{head}_\text{h})W^O$$

$$\text{where head}_\text{i} = \text{Attention}(QW_i^Q, KW_i^K, VW_i^V)$$

$h = 8$ parallel attention layers, or heads.

Learnable parameter matrices

Evaluating LLMs

- Metrics: Perplexity, BLEU, accuracy, F1.
- Human evaluation for quality and relevance.
- Benchmarks: MMLU, HELM, BigBench.

Prompt Engineering

- Practice of designing input queries to get the best possible output from AI language models like ChatGPT, Gemini etc.
- Since language models don’t “understand” tasks the way humans do, the way you phrase your prompt greatly influences the model’s response quality.
- Enables users to communicate effectively with AI models, leading to more accurate, efficient, and creative outputs

Ineffective prompt	Effective prompt
Tell me about law	In 3 bullet points, summarize the Indian Contract Act, 1872 for a law student.
Write something about technology	Write a 100 word paragraph explaining how blockchain is used in supply chains

Prompt Engineering Techniques

- Zero-shot Prompting: Giving the model a prompt without any examples, relying on its existing knowledge.
- Few-shot Prompting: Providing a few examples within the prompt to guide the model's understanding and desired output style.
- Chain-of-Thought (CoT) Prompting: Encouraging the model to explain its reasoning step-by-step before providing the final answer.
- Role Prompting: Assigning a specific role or persona to the AI to influence its tone and perspective.
- Constraint-based Prompting: Setting specific rules or limitations for the output, such as length, format, or content.

Examples of prompt engineering techniques are given here.

Challenges with LLMs

- Bias and fairness issues.
- Tendency to hallucinate incorrect facts.
- High energy and resource consumption.

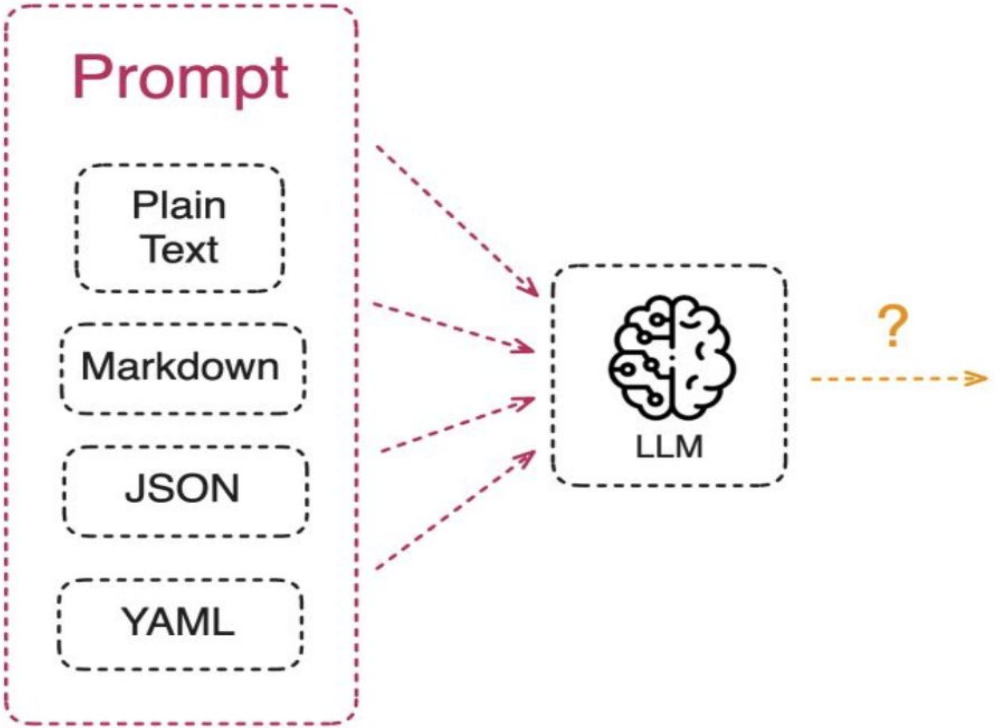
What’s following LLMs?

- Improved reasoning and factual grounding.
- Energy-efficient models and better alignment.
- Deeper integrations with applications.
- Multi-modal LLMs (text, image, audio).
- Mixture-of-Experts (MoE) for scalability.
- Small Language Models (SLMs) for efficiency.

Recap and Questions

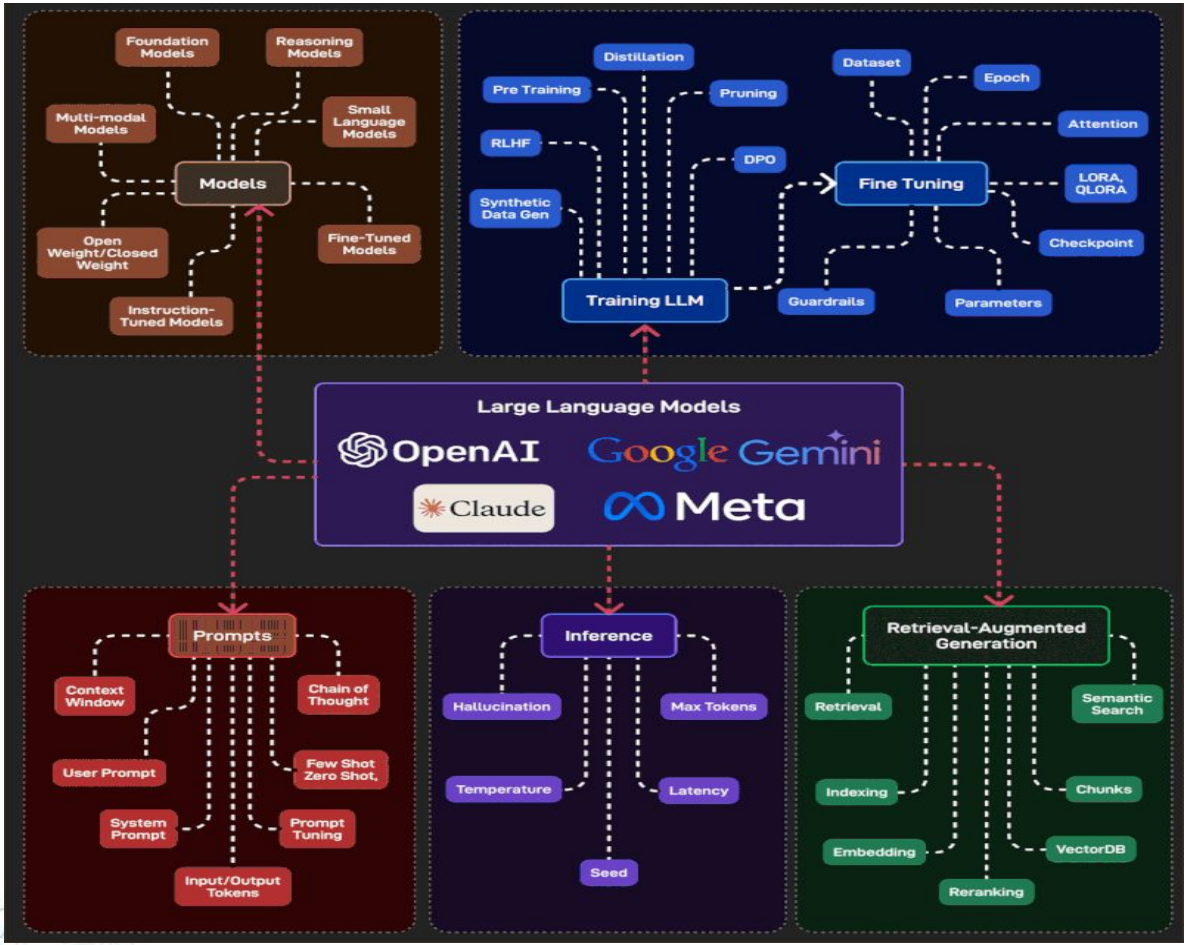
- We covered LLM types, architecture, core techniques, training, tuning, and trends.
- Big players in Generative AI.
- Any questions or clarifications?

A brainstorm exercise !



How does prompt formatting influence LLM performance?

LLM Glossary



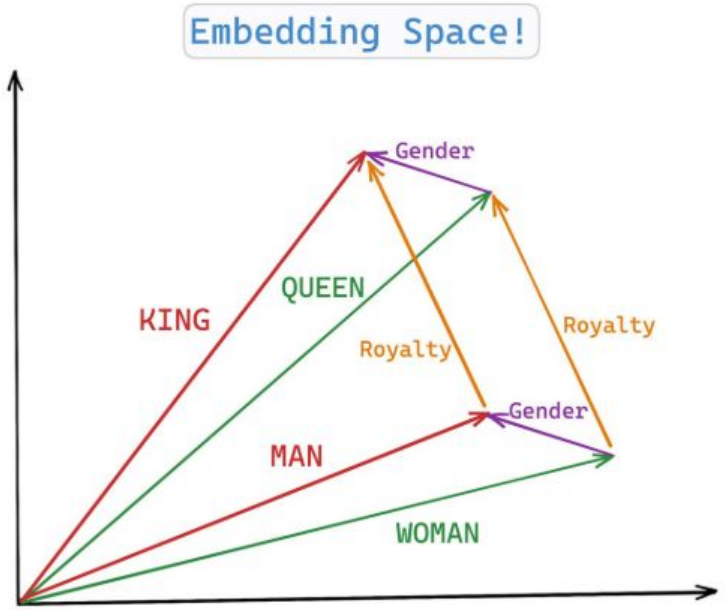
Hugging Face, Ollama

- Link and model
- Transformer

The

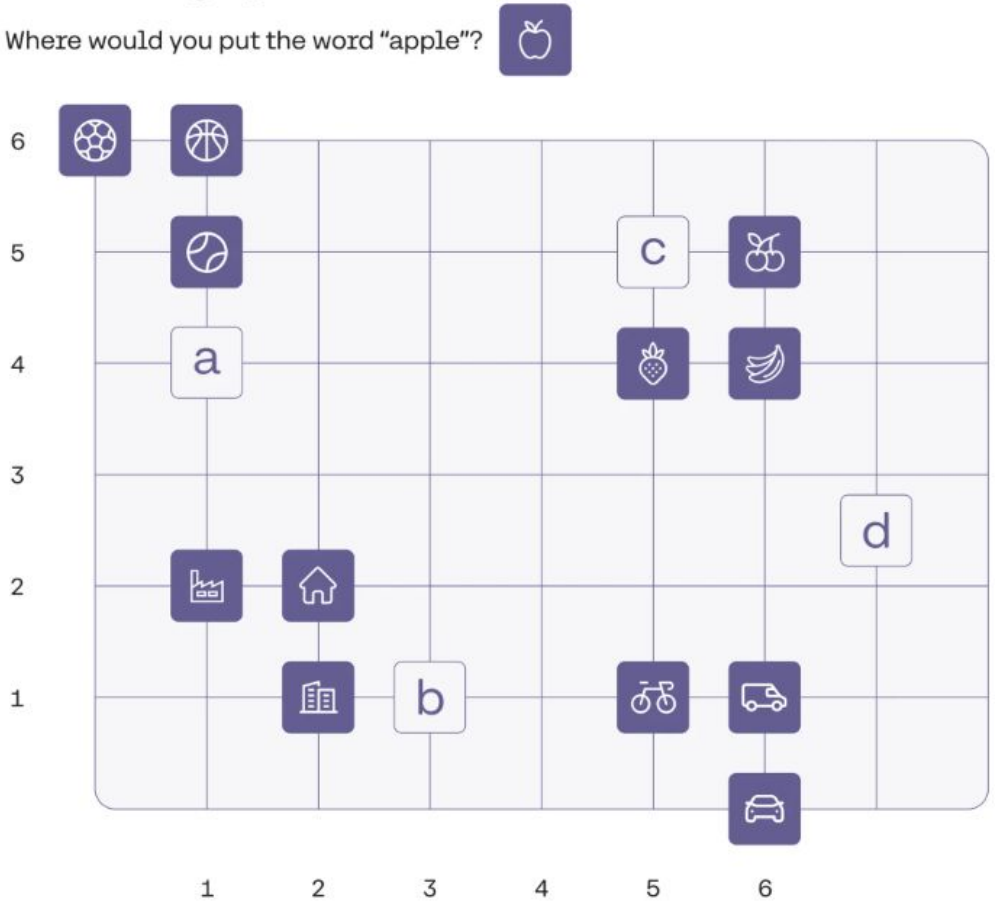
What are Word Embeddings?

- Natural language processing (NLP) models do not work directly with plain text. A numerical representation was necessary.
- Word embedding is a class of techniques where each word is represented as a real-valued vector.
- It represents words in a continuous vector space.
- Embeddings are dense representations in vector space.
- They can be represented in smaller dimensions compared to sparse representations like one-hot encoding.
- Most word embedding methods rely on the distributional hypothesis by Zellig Harris: "Words occurring in similar contexts tend to have similar meanings."
- We can visualize these embeddings using PCA.
- Example: king - man + woman ≈ queen



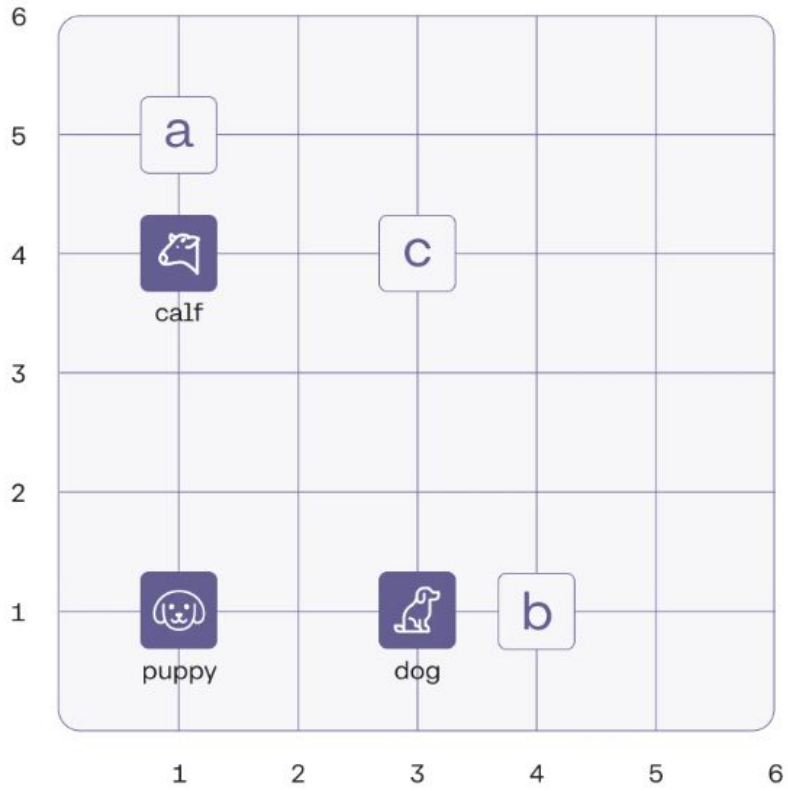
Embeddings Quiz 1:

Where would you put the word "apple"?



Embeddings Quiz 2:

Where would you put the word “cow”?



Popular Embedding Models

- Word2Vec: CBOW and Skip-Gram architectures.
- GloVe: Uses global word co-occurrence matrix.
- FastText: Includes subword information.

Word2Vec

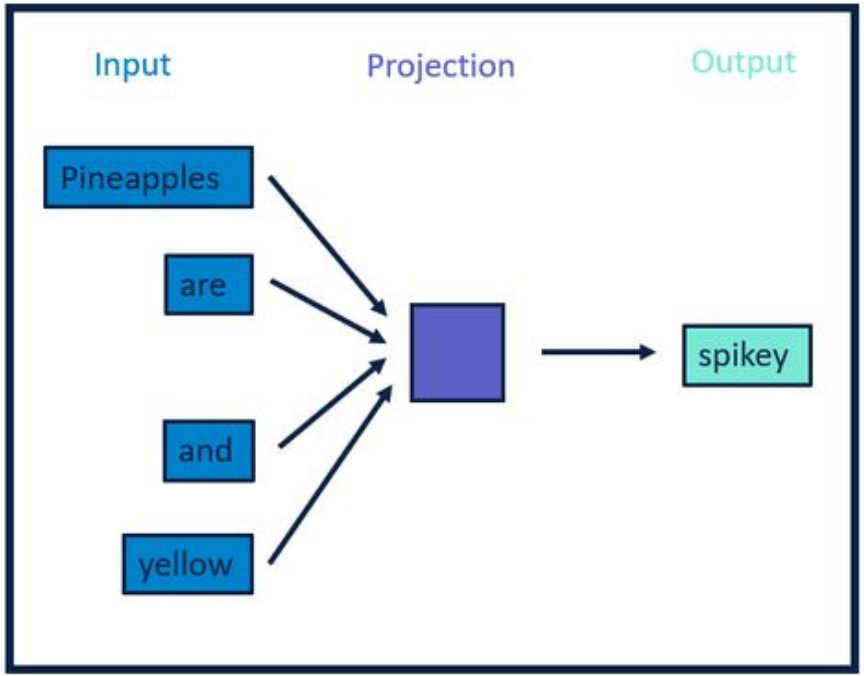
- Word2Vec creates word embeddings based on context similarity.
- Words that share contexts are located close together.

Main Architectures:

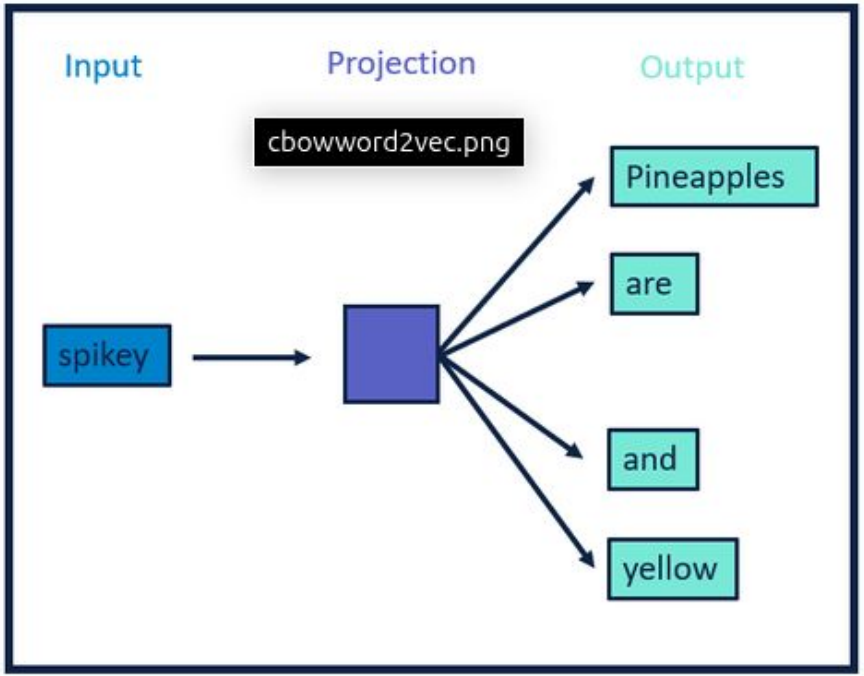
- One-Word Context (Educational Purpose):
  - Predicts a word from just one neighboring word. Rarely used practically.
- Skip-Gram:
  - Given a target word, predicts surrounding words (context).
  - Captures word-context relationships effectively.
- Continuous Bag-of-Words (CBOW):
  - Predicts a target word based on multiple surrounding words.
  - Does not consider order (hence “bag-of-words”).

Properties:

- Embeddings preserve semantic relationships.
- Can capture meaningful analogies and patterns.
- https://remykarem.github.io/word2vec-demo/



CBOW



Skip-gram

GloVe

- **GloVe (Global Vectors)** is a word embedding method based on **matrix factorization**.
- **Two key components**:
  - **Global Matrix Factorization**: Reducing large co-occurrence matrices.
  - **Local Context Window**: Capturing context (like CBOW and Skip-Gram).
- **Key idea**:
  - Constructs a **word-context co-occurrence matrix** from a large text corpus.
  - Optimizes embeddings so the dot product of two vectors equals the log frequency of their co-occurrences.

Example:

- If "cat" and "dog" co-occur 20 times within a certain context window, embeddings are trained so:
  a. $\text{Vector(cat)} \cdot \text{Vector(dog)} = \log(20)$

FastText

FastText – developed by Facebook AI Research.

- **Key advantage:** Handles **out-of-vocabulary (OOV)** and rare words effectively.
- **Morphological awareness:**
  - Breaks words into character-level **n-grams**.
  - Derives embeddings for unknown words based on morphology.
- **Performance comparison:**
  - **FastText**: Superior in capturing morphology and rare words.
  - **Word2Vec/GloVe**: Better for purely semantic relationships.
- **Efficient training and prediction.**

Limitations of Word Embeddings

Static representations

→ One fixed vector per word, regardless of context.

Fails with polysemy

→ Can’t distinguish between different meanings of the same word.

Example: “bank” (river bank vs. bank account)

Context is ignored

→ Meaning doesn’t change based on sentence.

Contextual embeddings (e.g., BERT, GPT) solve this

→ Different vector for each usage.

Transition to Contextual Embeddings

- **Contextual embeddings** generate different vectors for the same word depending on usage.
- Each word’s meaning is shaped by its **surrounding context**.
- **BERT, GPT** and similar models:
  - Provide **token-level embeddings**.
  - Understand word **sense, syntax, and semantics**.
- Enable more accurate NLP tasks: Q&A, sentiment analysis, translation.

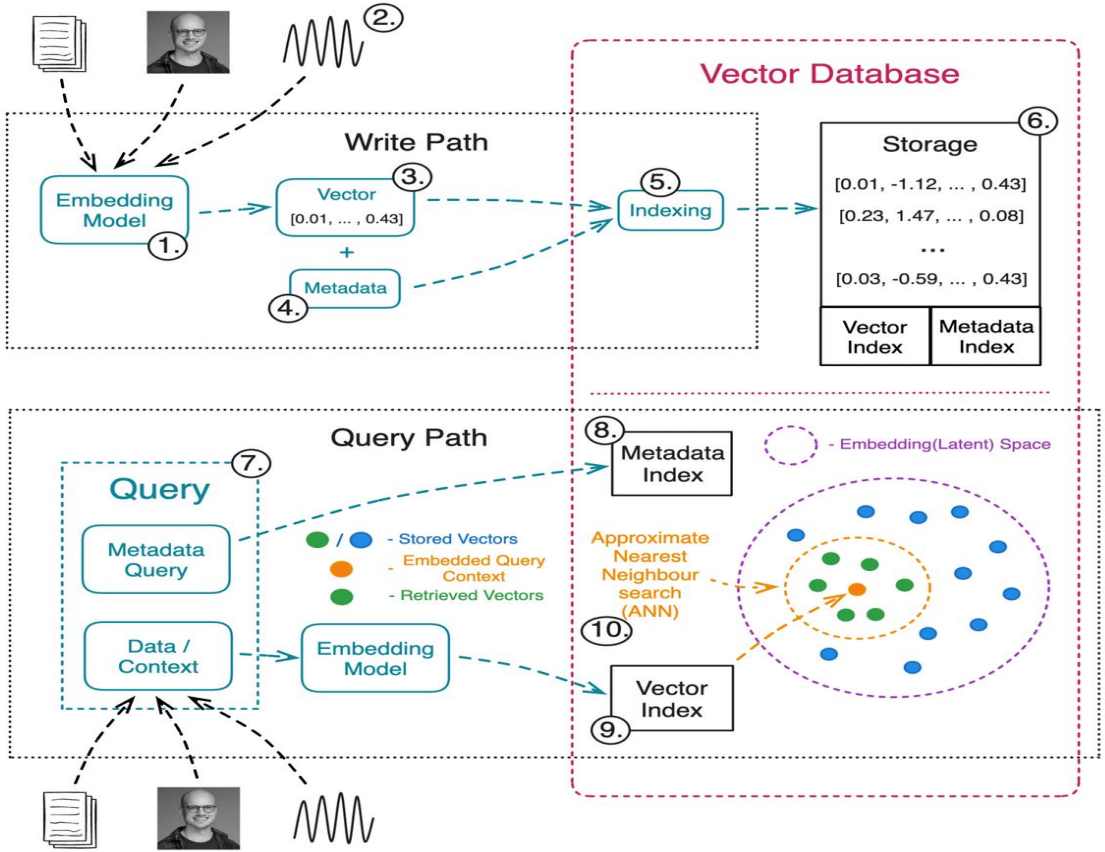
What is a Vector Database?

- Stores and indexes embeddings (vectors).
- Enables fast similarity search using approximate methods.
- Core to Retrieval-Augmented Generation (RAG).
- Similarity metrics: cosine, dot product, Euclidean.
- ANN algorithms: HNSW, IVF, PQ for scalability.

Popular Vector DB Tools

- FAISS (Facebook AI Similarity Search).
- Milvus, ChromaDB, Weaviate, Qdrant, Pinecone.
- Integrated with LLM pipelines.

Work flow of Vector DBs



Use Cases in Practice

- Semantic search over documents.
- Recommendation engines.
- RAG: enhances LLMs with external knowledge.

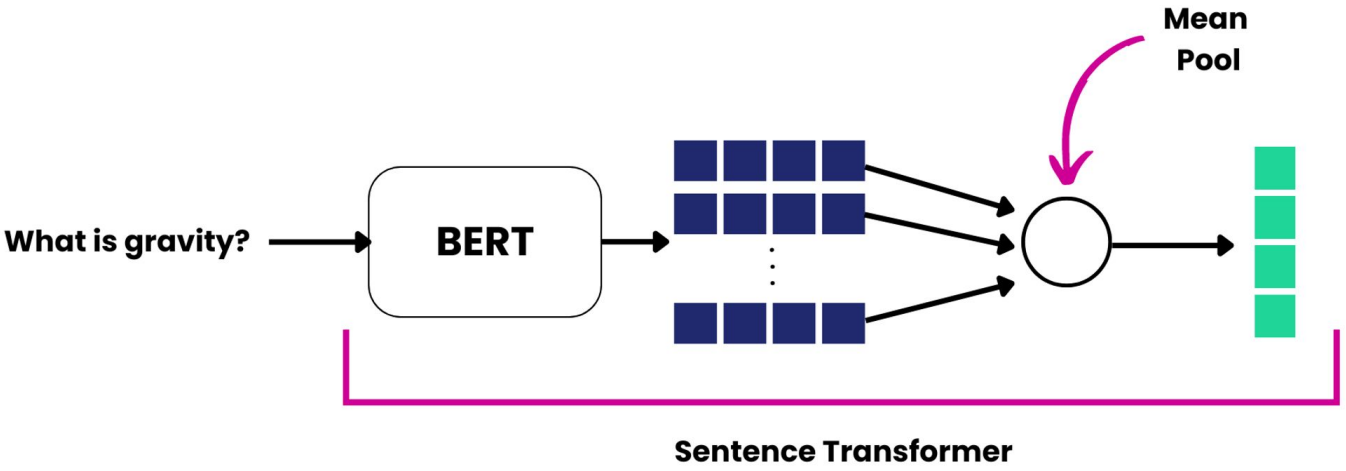
A simple demo of vector databases using ChromaDB is shown here

Recap and doubts

Sentence Transformers

What are Sentence Transformers?

- Models that encode sentences into fixed-size embeddings
- Capture semantic meaning beyond word-level embeddings
- Useful for tasks like semantic search, clustering, and similarity
- Built on top of BERT with pooling layers.



How Sentence-Transformers Wraps Transformers

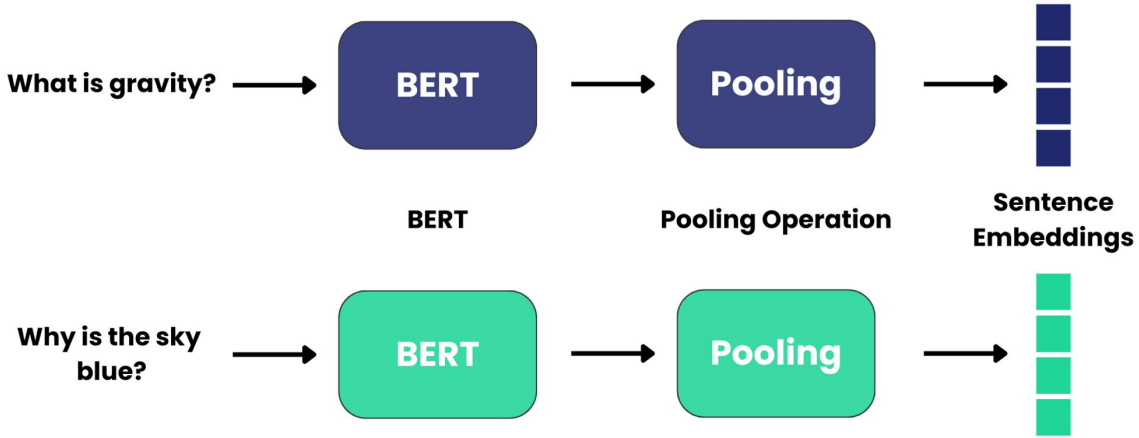
- Initialize with any Hugging Face model (BERT, RoBERTa, DistilBERT, T5, etc.)
- Automatically inserts a Transformer encoder module
- Includes a Mean Pooling layer by default
- Enables out-of-the-box sentence embedding generation
- Both code snippets are identical:

```python
from sentence_transformers import SentenceTransformer

model = SentenceTransformer("bert-base-uncased")
```

```python
from sentence_transformers import models, SentenceTransformer

transformer = models.Transformer("bert-base-uncased")
pooling = models.Pooling(transformer.get_word_embedding_dimension(), pooling_mode="mean")
model = SentenceTransformer(modules=[transformer, pooling])
```

SBERT – Enhanced Sentence Embeddings

- Tokenization & embedding lookup
- Siamese (twin) Transformer encoders
- Mean-pooling + vector concatenation ‘u, v, |u–v|‘
- Optional FFNN head for classification/similarity
- Fine-tuned on sentence-pair tasks (NLI, STS)
- One forward pass for both sentences → efficient



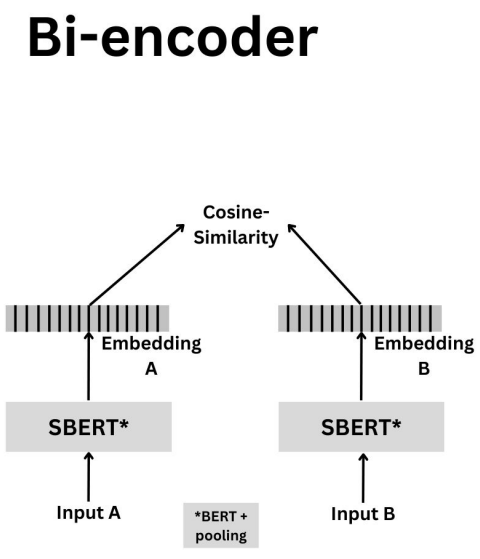
Use Cases of Sentence Transformer

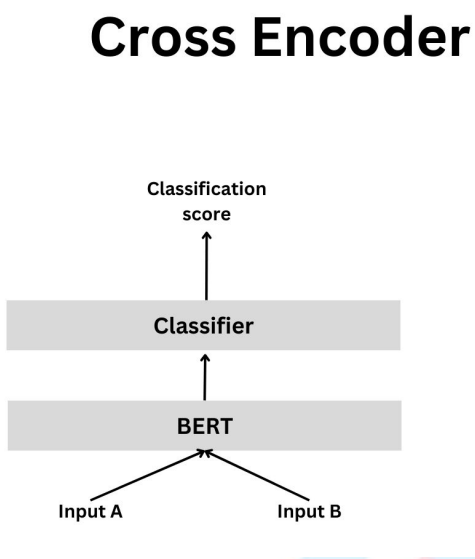
- Semantic Search & Retrieval
- Clustering & Topic Modeling
- Paraphrase & Duplicate Detection
- Question–Answer Matching
- Recommendation Engines
- Cross-lingual Retrieval

An example of sentence embedding is given here

Sentence Transformers vs. Cross-Encoders

- **Architecture**
  - Sentence Transformer: Siamese encoders + pooling
  - Cross-Encoder: Single encoder over concatenated pair
- **Inference Cost**
  - Sentence Transformer: 1 forward-pass per sentence → reuse embeddings
  - Cross-Encoder: 1 forward-pass per sentence pair
- **Speed vs. Accuracy**
  - Sentence Transformer: Fast, scalable; slightly lower pairwise accuracy
  - Cross-Encoder: Slower; higher accuracy on fine-grained tasks
- **Use Cases**
  - Sentence Transformer: Large-scale retrieval, clustering
  - Cross-Encoder: Re-ranking top K candidates, classification





Thank You