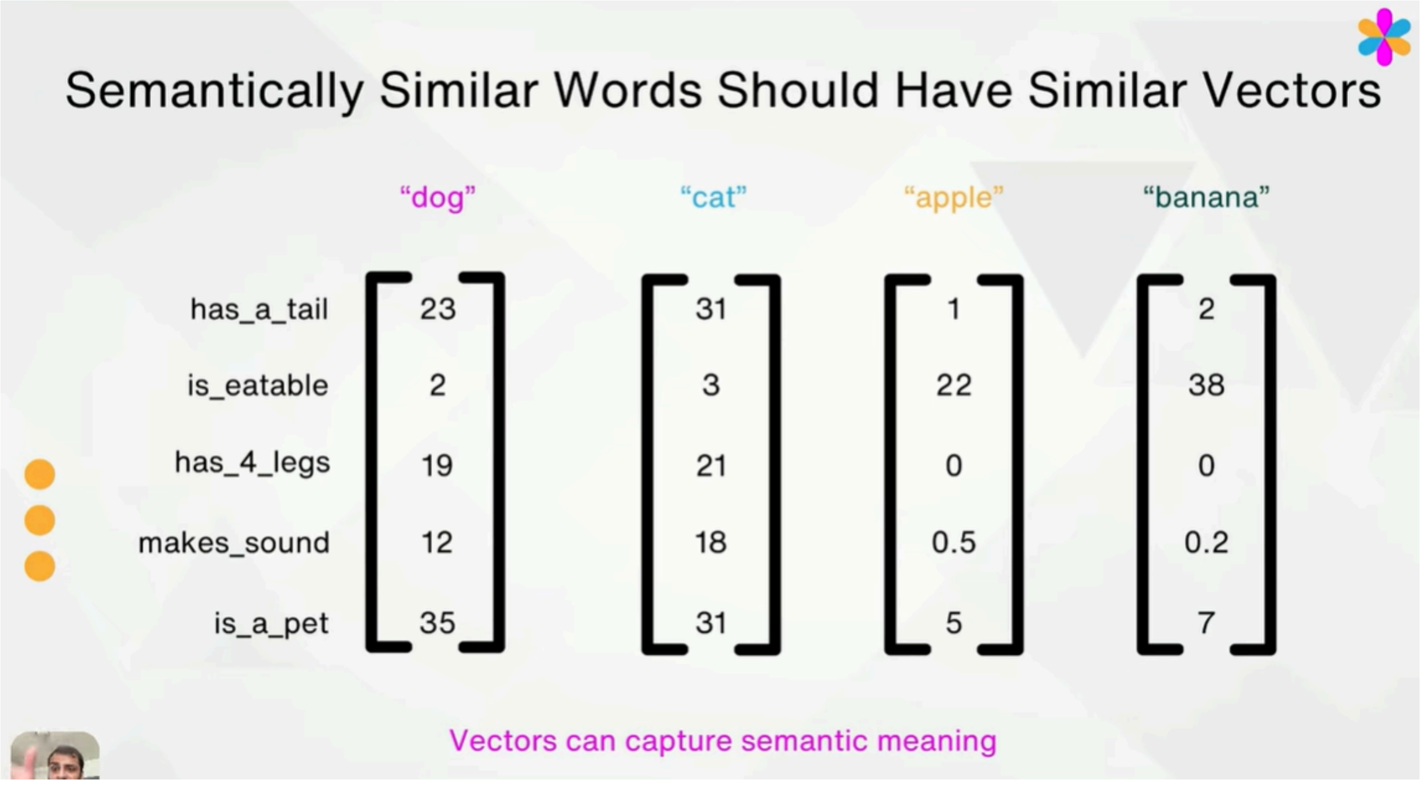
Lecture 10: Token Embeddings

Why not just use the token ids for each word?

→ Because, just randomly assigning token ids cannot capture the semantic meaning of words.



Semantically Similar Words Should Have Similar Vectors

	"dog"	"cat"	"apple"	"banana"
has_a_tail	23	31	1	2
is_eatable	2	3	22	38
has_4_legs	19	21	0	0
makes_sound	12	18	0.5	0.2
is_a_pet	35	31	5	7

Vectors can capture semantic meaning

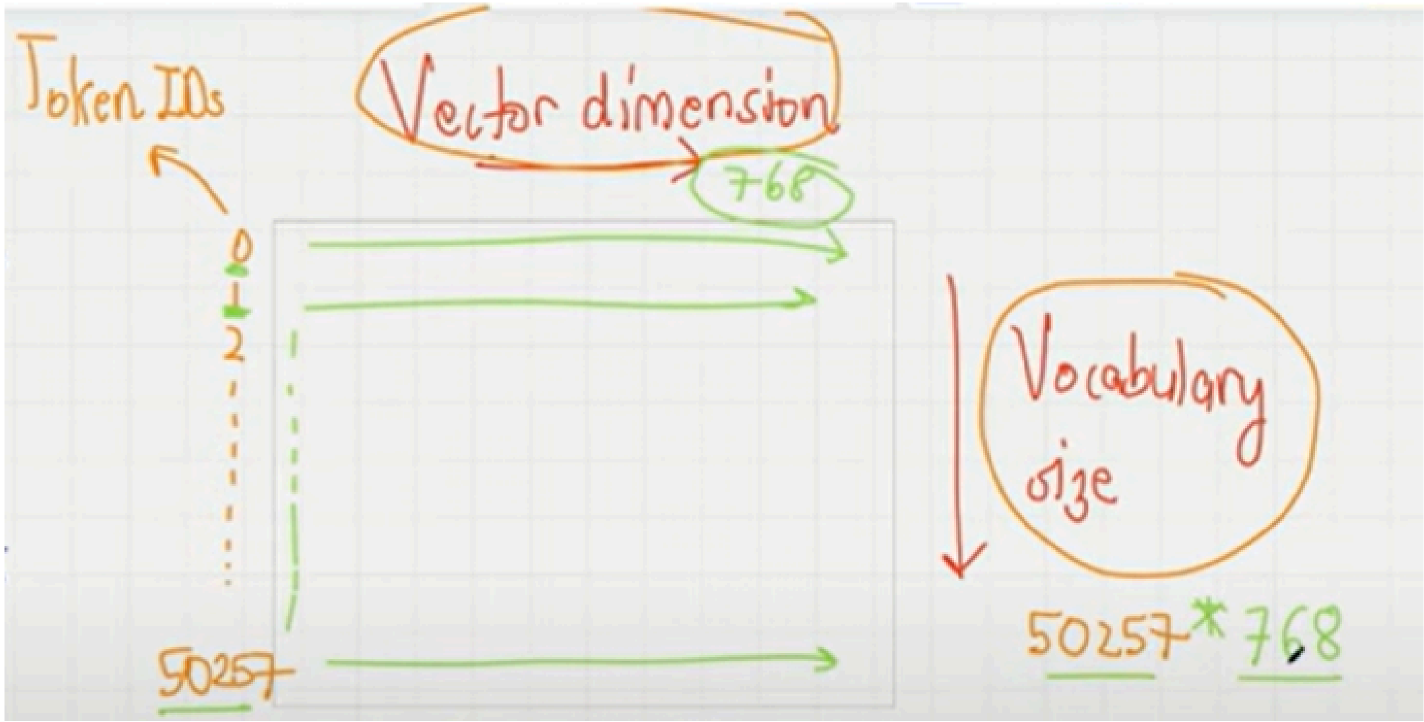
Whatever is higher in dog, is higher in cat. Similarly for the low numbers. So it means dog and cat are similar things.

Whatever is higher in apple, is higher in banana. Similarly for the low numbers. So it means apple and banana are similar things.

These are called as vector embeddings or token embeddings.

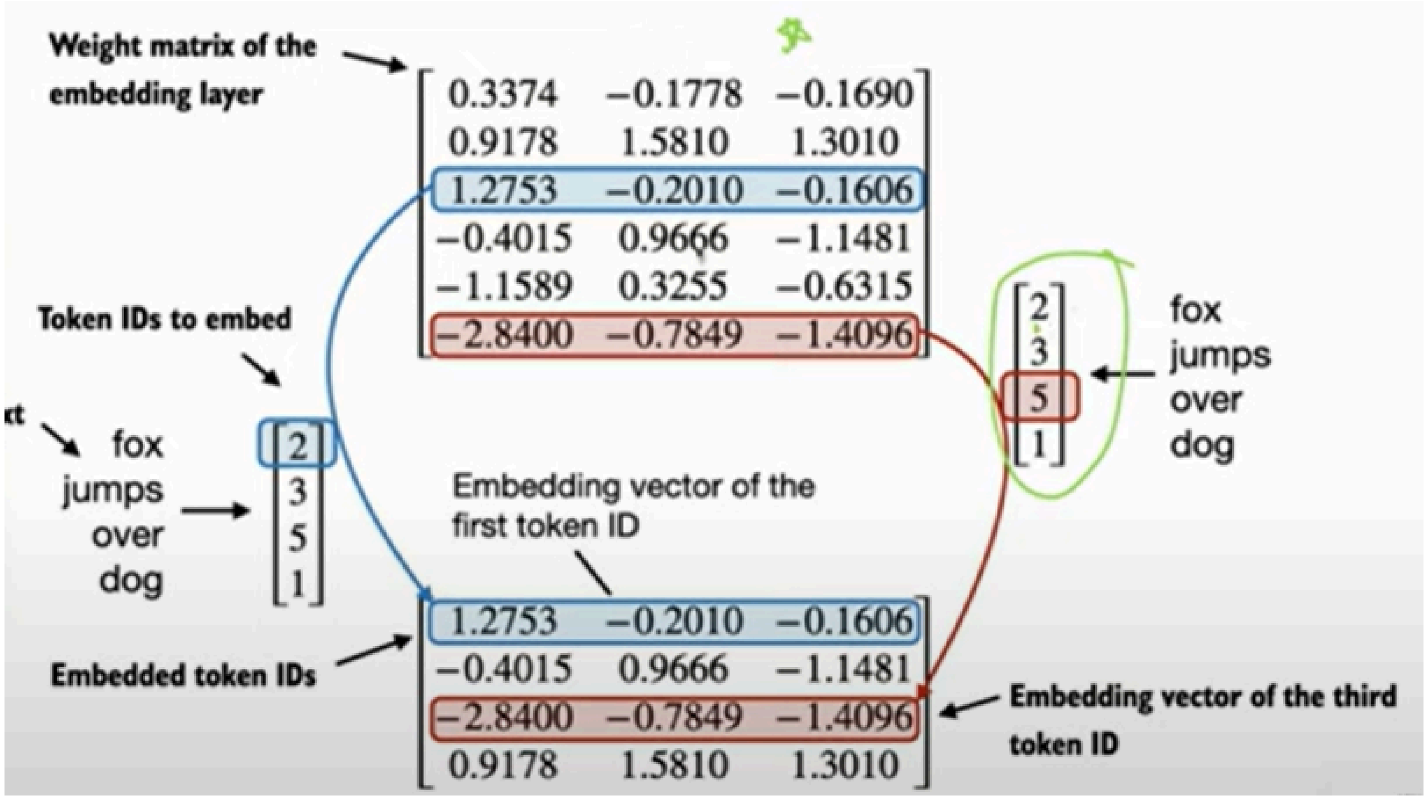
We have to train a neural network to create vector embeddings.

In GPT-2 model:



This is why embedding vector is also called a **lookup table** that stores embeddings of a fixed dictionary and size.

The embedding layer is a **lookup operation** that retrieves rows from the embedding layer weight matrix using a token ID.


Weight matrix of the embedding layer
Token IDs to embed
xt
fox
jumps
over
dog
Embedding vector of the first token ID
Embedded token IDs
fox
jumps
over
dog
Embedding vector of the third token ID

$$\begin{bmatrix} 0.3374 & -0.1778 & -0.1690 \\ 0.9178 & 1.5810 & 1.3010 \\ 1.2753 & -0.2010 & -0.1606 \\ -0.4015 & 0.9666 & -1.1481 \\ -1.1589 & 0.3255 & -0.6315 \\ -2.8400 & -0.7849 & -1.4096 \end{bmatrix}$$

$$\begin{bmatrix} 2 \\ 3 \\ 5 \\ 1 \end{bmatrix}$$

$$\begin{bmatrix} 1.2753 & -0.2010 & -0.1606 \\ -0.4015 & 0.9666 & -1.1481 \\ -2.8400 & -0.7849 & -1.4096 \\ 0.9178 & 1.5810 & 1.3010 \end{bmatrix}$$

Both embedding layer and Neural network layer (linear layer) lead to the same output.

But embedding layer is used instead because it is much more computationally efficient, neural network has so many unnecessary multiplication with zeros.