Understanding Large Language Models

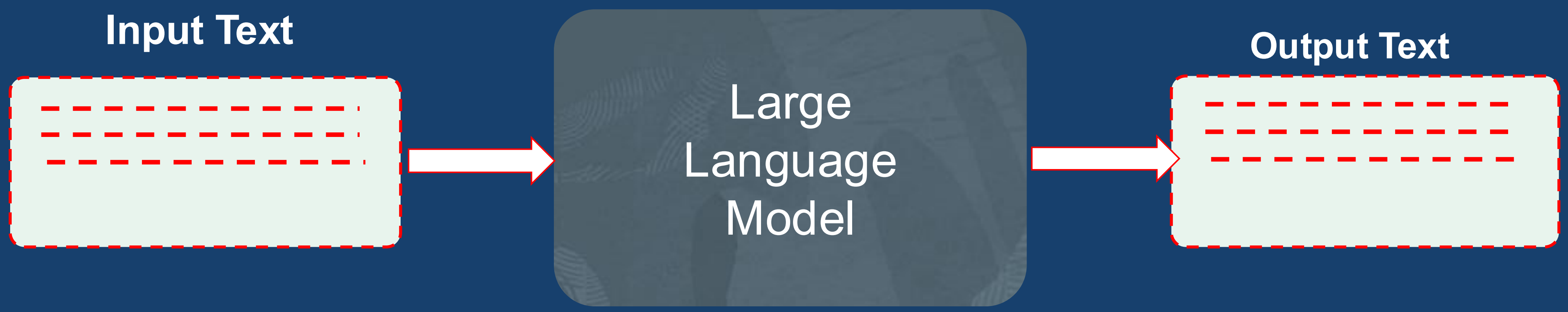
Generative AI and Prompt Engineering

Ram N Sangwan

- Large Language Models
- Transformers, Sequence Models, RNN, Encoder - Decoder
- Embeddings, Tokenization

Large Language Models (LLMs)

- Understand, generate and process human language at massive scale.
- Designed for sequence-to-sequence tasks such as machine translation



Large Language Models Example

1 Translate "How are you" into French.

2 What is the capital of France?

3 Write an essay on French Revolution.

Large Language Model

1 "Comment allez-vous?"

2 The capital of France is Paris.

3 **Title:** The French Revolution: A Turning Point in History
Introduction: The French Revolution, which took place between 1789 and 1799, stands as one of the most influential and transformative events in human history...

LLMs – What you need to Know.

- LLM Architectures
  - What else can LLMs do?
- Prompting and Training
  - How do we affect the distribution over the vocabulary?
- Decoding
  - How do LLMs generate text using these distributions?

Model Size and Parameters

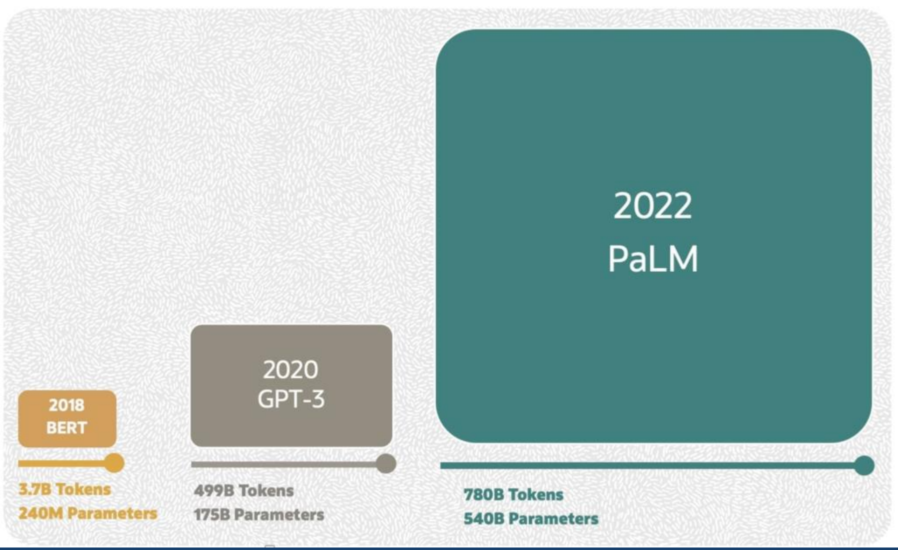
Model Size: Memory required to store the model's parameters.

Parameters:

- Tuneable variables that are adjusted during training to minimize the difference between predicted outputs and actual labels (e.g., correct answers).
- Used to represent relationships between input tokens and output text.
- The goal is to find the optimal values such that the model can accurately predict new, unseen inputs.

Two Types of Parameters

1. **Learned Embeddings** are dense vectors that represent each token in a fixed-dimensional space. Learned embeddings capture semantic relationships between tokens and are used as input features for the model.
2. **Model Weights** - are the adjustable values within the neural network's layers, such as fully connected layers, convolutional layers, or recurrent layers.

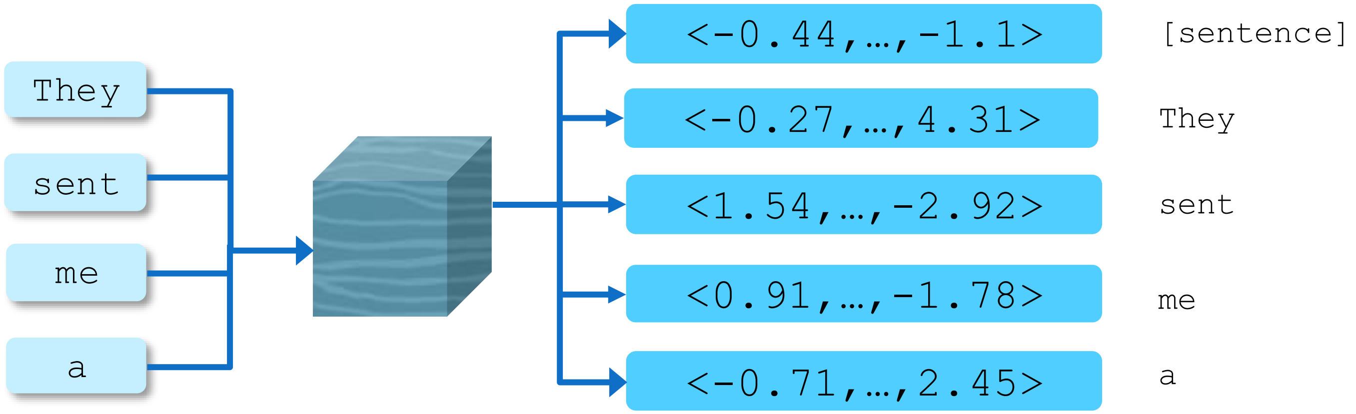


Encoders

Encoder – models that convert a sequence of words to an embedding (vector representation)

Examples

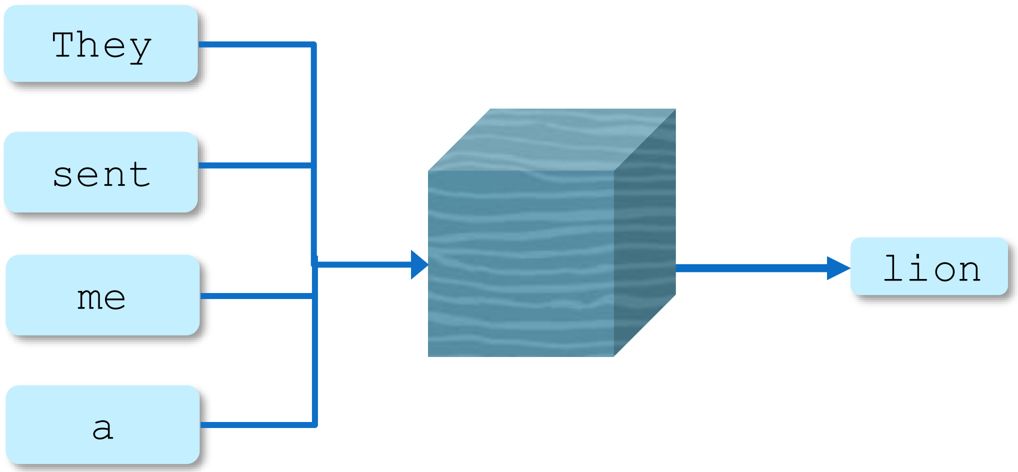
Embed-light, BERT, RoBERTA, DistillBERT, SBERT,…



Primary uses: embedding tokens, sentences, & documents

Decoders

- **Decoder** – models take a sequence of words and output next word



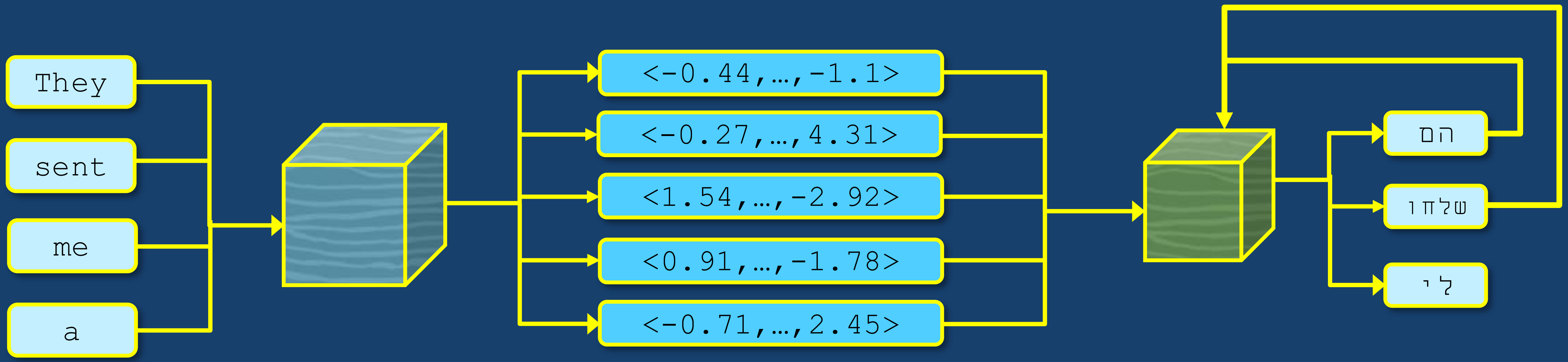
- **Examples**
- *GPT-4, Llama, BLOOM, Falcon, …*

Primary uses: text generation, chat-style models, (including QA, etc…)

Encoders - Decoders

Encoder-decoder - encodes a sequence of words and use the encoding + to output a next word

Examples
T5, UL2, BART,…



Understanding Language for Machines Can be Tricky

Jane is doing the throwing

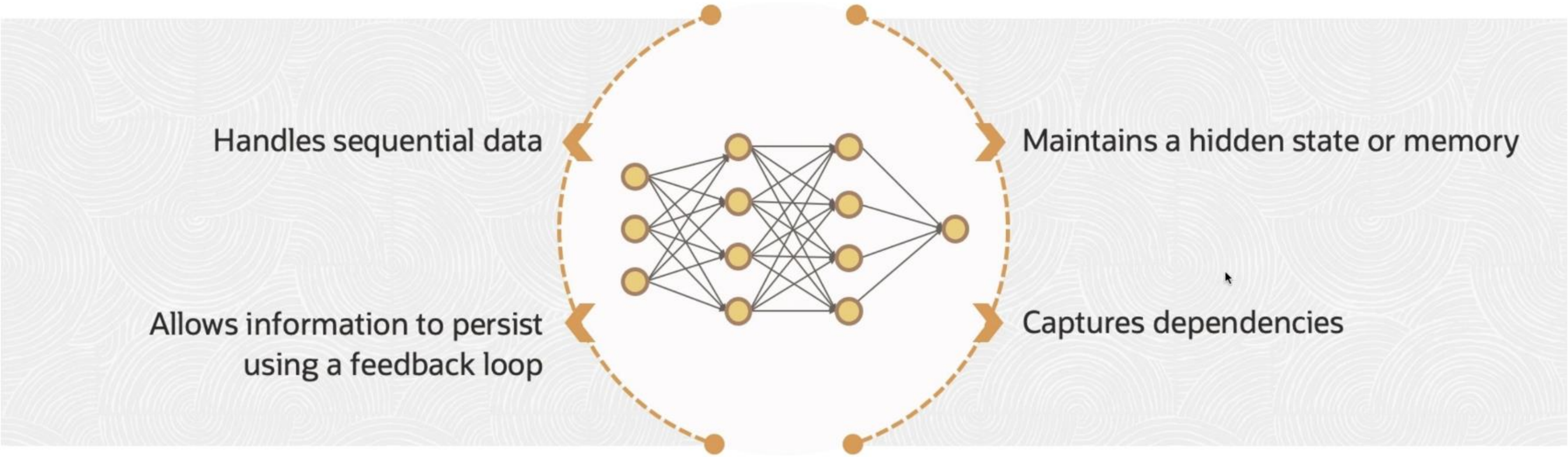
Dog is doing the fetching

It refers to the frisbee

Jane threw the frisbee and her dog fetched it.

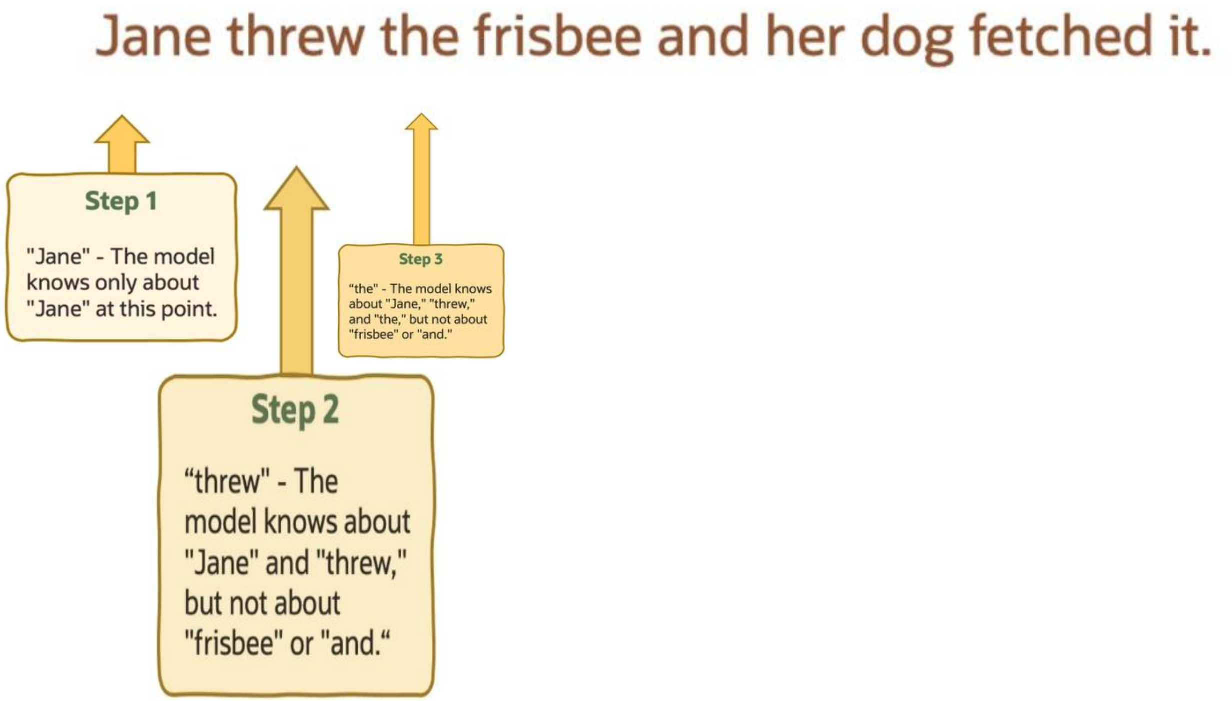
As human, we understand easily that “it” refers to the frisbee. But for a machine, it can be tricky.

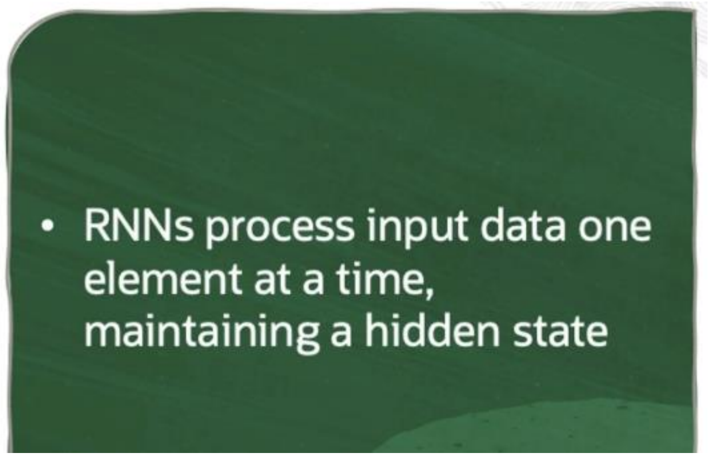
Recurrent Neural Network (RNN)



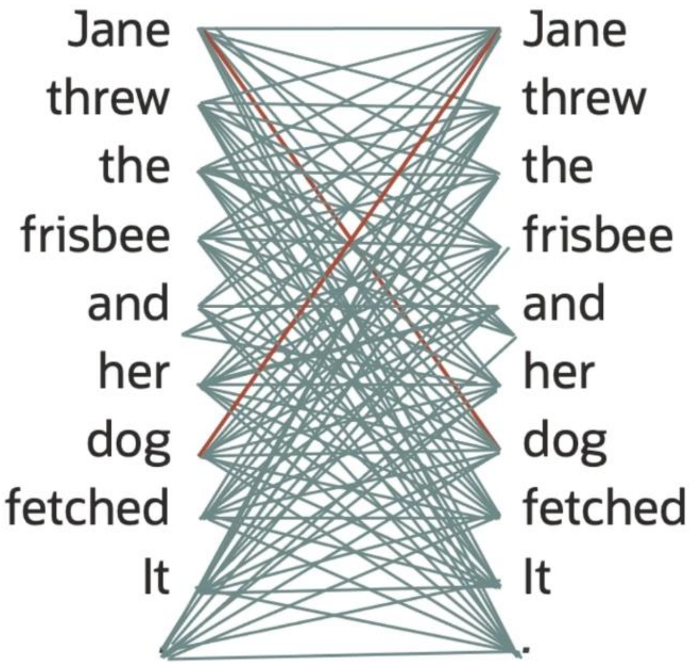
Recurrent Neural Network works on the principle of saving the output layer and feeding this back to the
input in order to predict the output of the layer.

RNNs Struggle with Long-Range Dependencies.



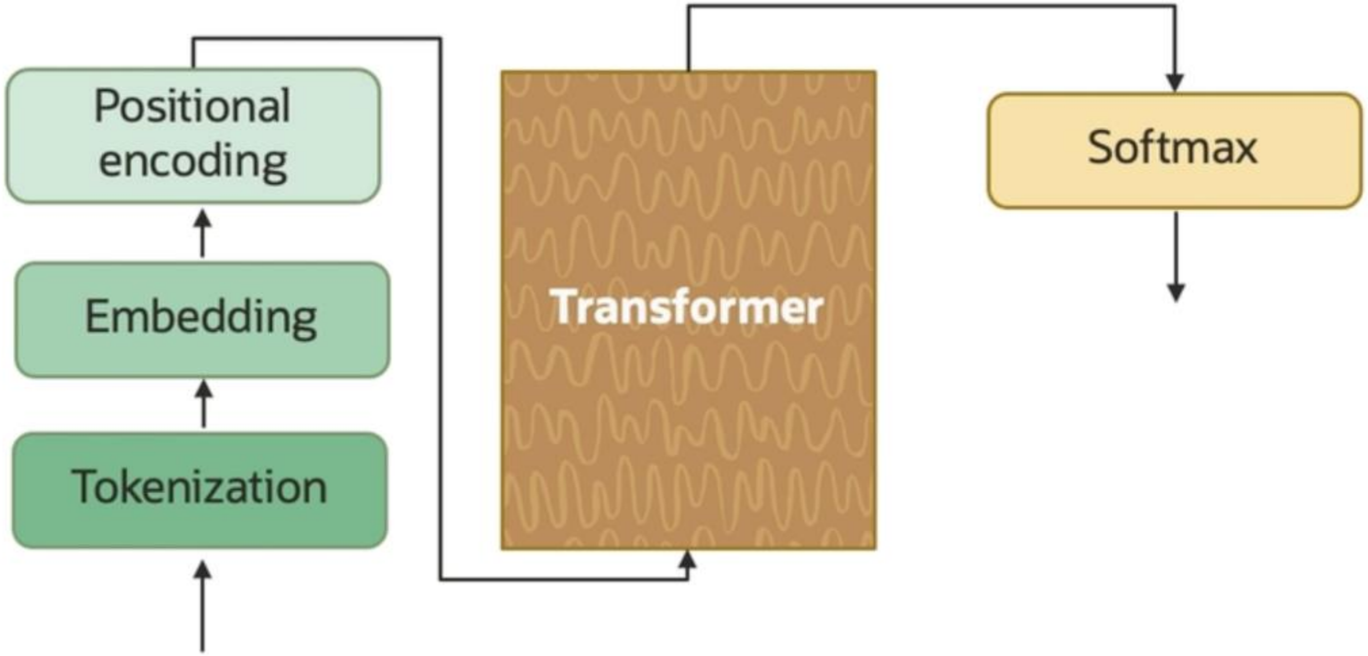


Transformers Understand Sentences as a Whole



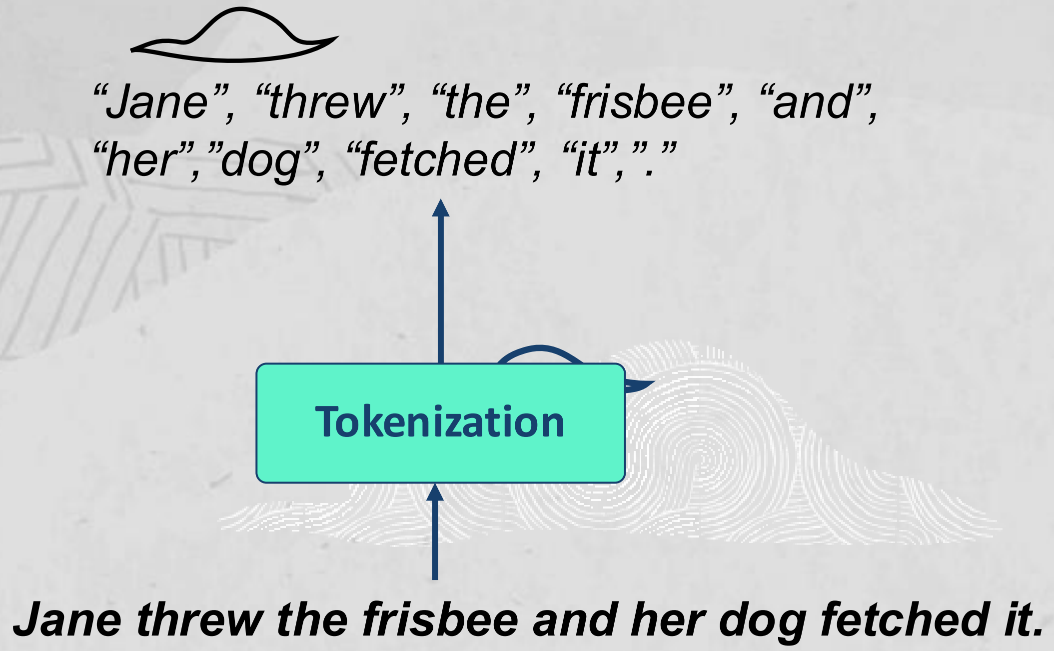
- Can look at all the words in the sentence at the same time and understand how they relate to each other
- Can simultaneously understand the connections between "Jane" and "dog," even though they are far apart in the sentence

Understanding Transformer Architecture



Jane threw the frisbee and her dog fetched it.

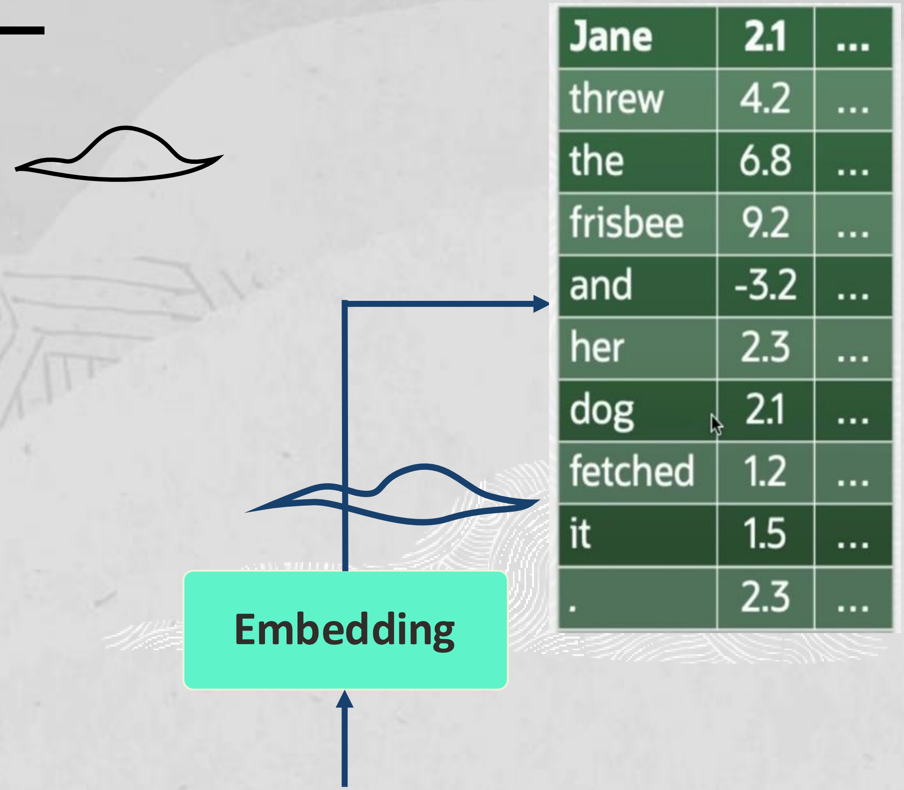
Tokenization



- The Sentence is broken down into smaller pieces called tokens.

- The choice of how to break down the sentence depends on the specific tokenizer used.
- Learn to Create a Tokenizer from Scratch *here*

Embedding



“Jane”, “threw”, “the”, “frisbee”, “and”, “her”,”dog”, “fetched”, “it”,”.”

- Each token is then converted into a numerical form (a vector) that the model can understand.
- These vectors are created in a way that captures the meaning of the word.
- More on Vectors and Embeddings here

Positional Encoding

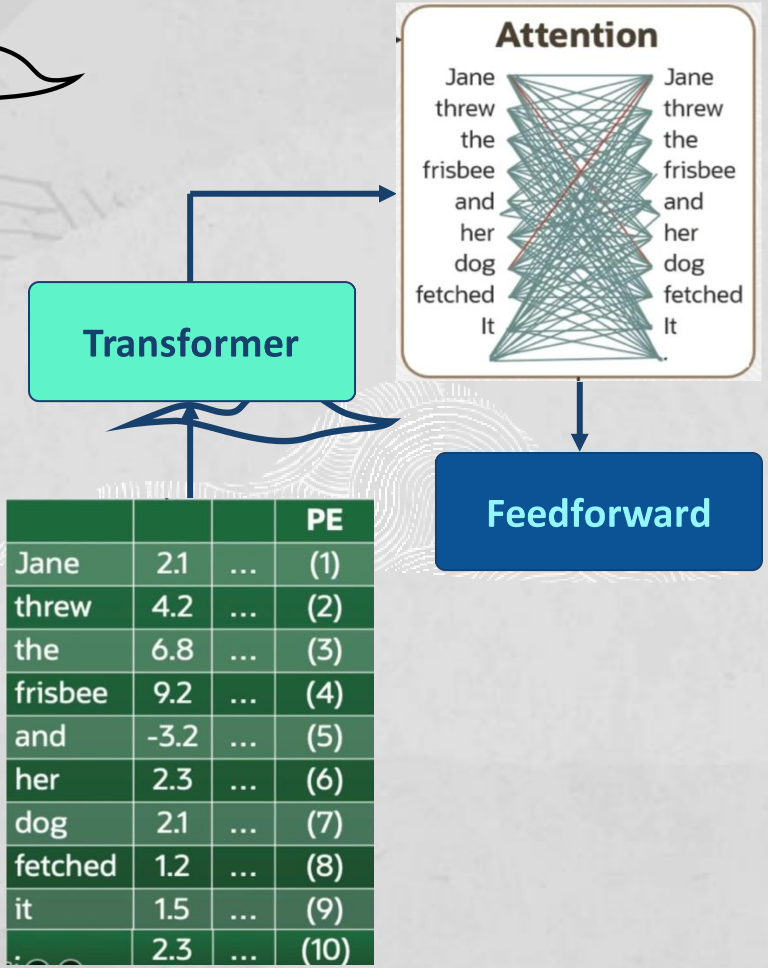
Jane	2.1	...
threw	4.2	...
the	6.8	...
frisbee	9.2	...
and	-3.2	...
her	2.3	...
dog	2.1	...
fetched	1.2	...
it	1.5	...
.	2.3	...

Positional encoding

			PE
Jane	2.1	...	(1)
threw	4.2	...	(2)
the	6.8	...	(3)
frisbee	9.2	...	(4)
and	-3.2	...	(5)
her	2.3	...	(6)
dog	2.1	...	(7)
fetched	1.2	...	(8)
it	1.5	...	(9)
.	2.3	...	(10)

- The Position Encoding layer represents the position of the word.
- The Model needs to know the order of the words in the sentence.
- The model adds information about the position of each word.

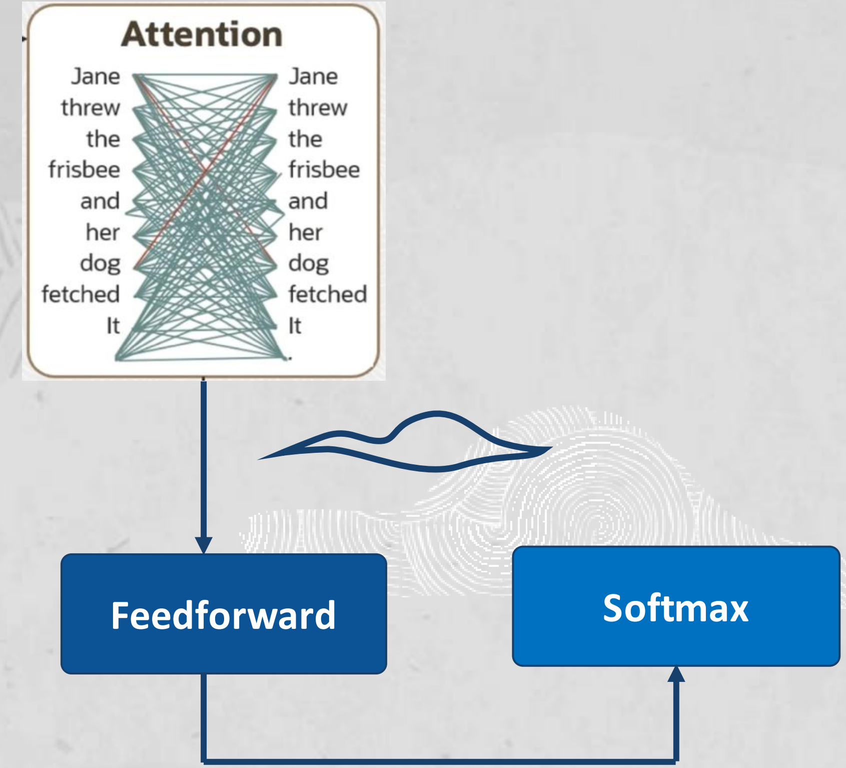
Transformer



			PE
Jane	2.1	...	(1)
threw	4.2	...	(2)
the	6.8	...	(3)
frisbee	9.2	...	(4)
and	-3.2	...	(5)
her	2.3	...	(6)
dog	2.1	...	(7)
fetched	1.2	...	(8)
it	1.5	...	(9)
[illegible]	2.3	...	(10)

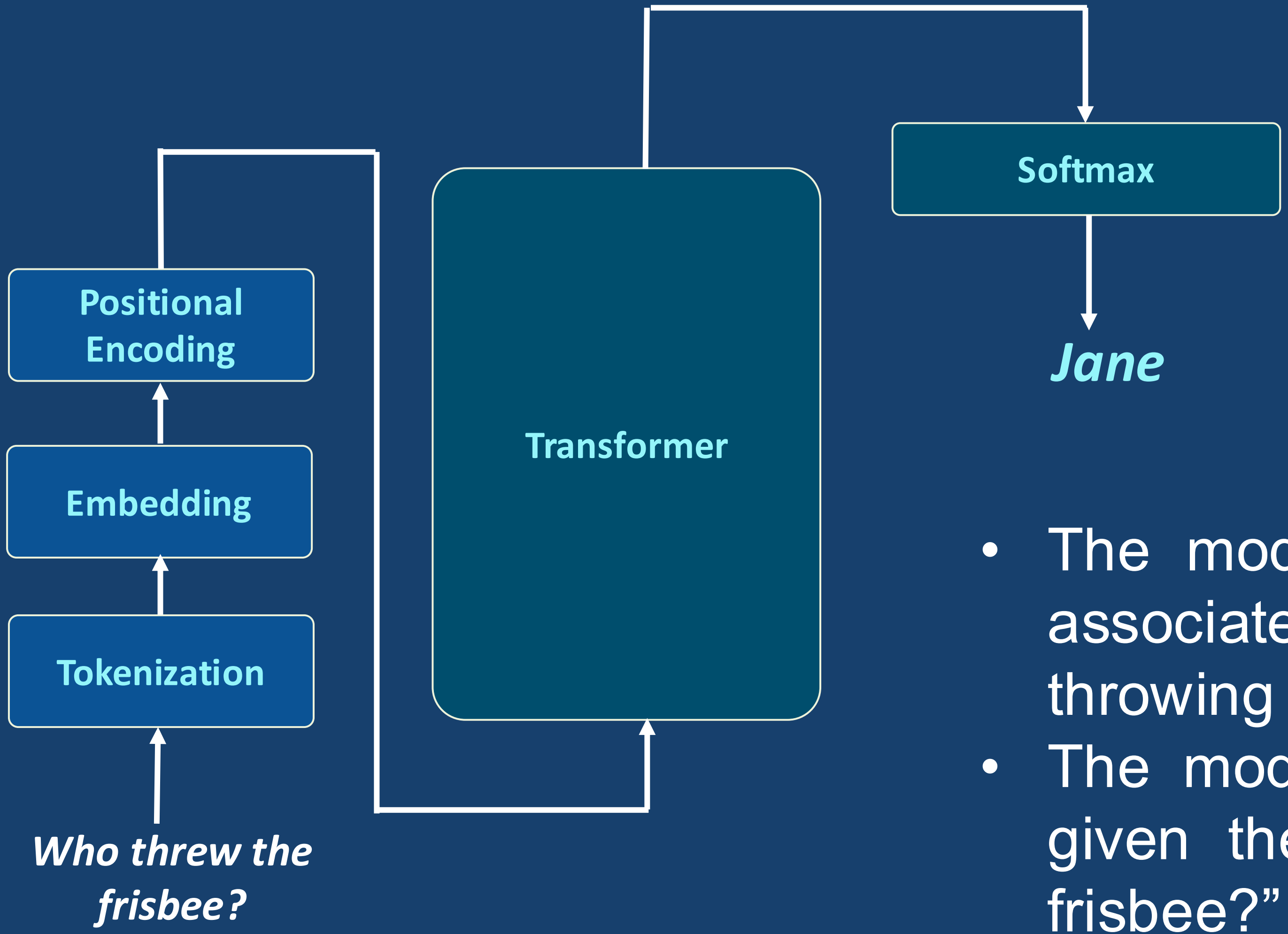
- Attention: Helps understand the context of each word.
- Feedforward: Applies a specific function to each word individually.

Transformer



- The model generates a list of scores for each word in the vocabulary.
- The model uses the Softmax function to turn these scores into probabilities.

Transformer



Output from the transformer's Softmax layer is a probability distribution across the entire dictionary of words.

- The model realizes that "Jane" is associated with the action of throwing the frisbee.
- The model predicts the next word given the context "who threw the frisbee?"

Thank You

- Watch a Video on LLS Explained here