Applied Text Analytics & Natural Language Processing

with Dr. Mahdi Roozbahani
& Wafa Louhichi

Transformers
Introduction (Attention) – Part 3

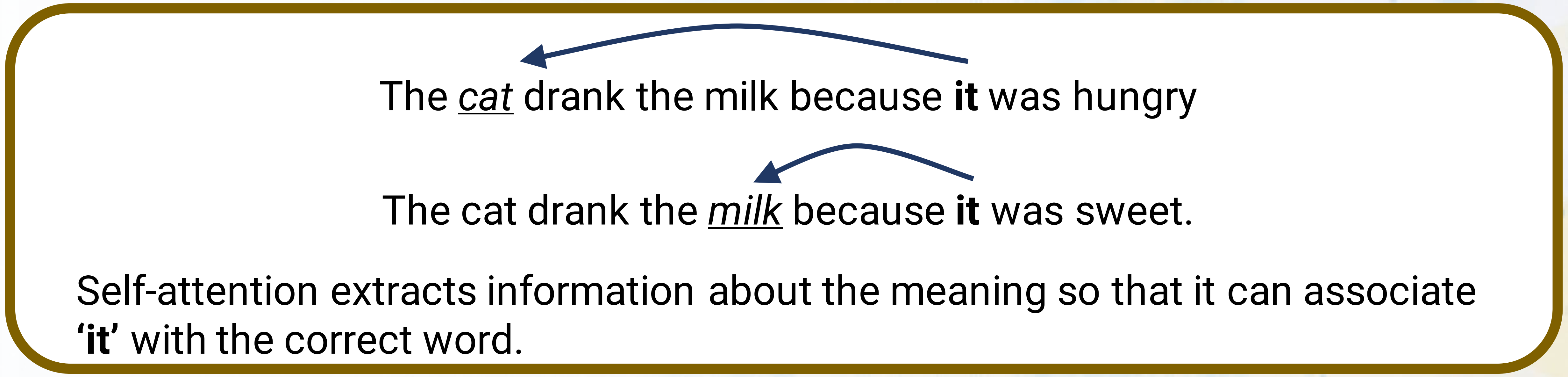
Attention

- Bahdanau et al. (2014) introduced the Attention mechanism to improve the performance of encoder-decoder models for Machine Translation
- Attention allows making predictions about the **output** by paying “attention” to some parts of the **input** sequence (we deal with **2 sequences**; an input and an output sequence)
- Attention mechanism fixed the following issues of sequence-to-sequence models:
  - Dealing with long-range dependencies (Dependency among words separated by multiple sentences) between words in a long sentence

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Self-Attention

- When creating the representation for each word in a sequence, Self-Attention tells the model to pay attention to which words in the same sequence .
- Example: in translating "I like exercising in the morning" , when we want to create a representation for the word "exercising":
  - We'd want to look at the subject "I" because languages usually have different verb forms for different subjects.
  - The other words don't matter that much in the translation of this word.
- Words can have different meanings when placed next to other words.



The *cat* drank the milk because **it** was hungry

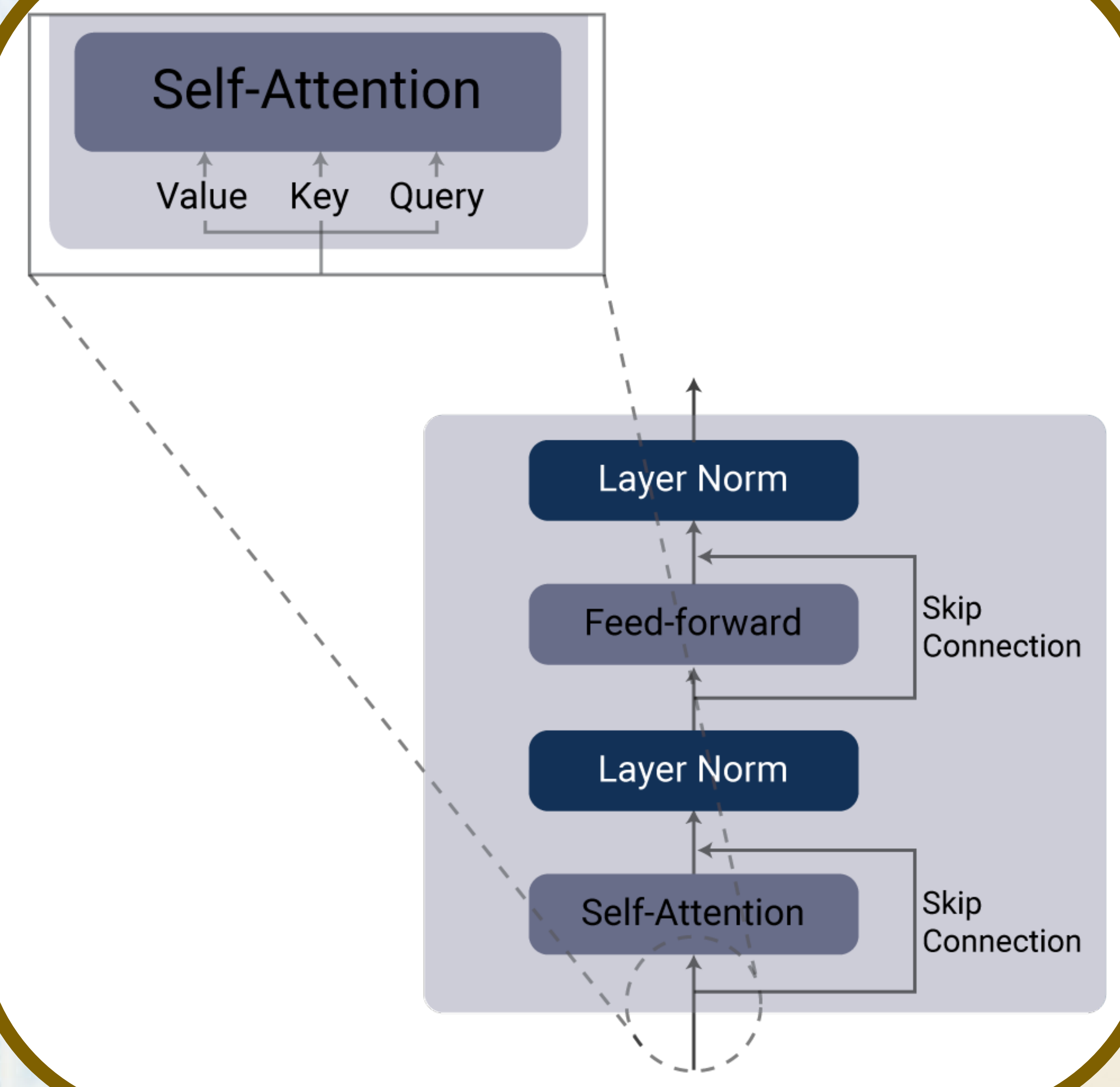
The cat drank the *milk* because **it** was sweet.

Self-attention extracts information about the meaning so that it can associate **'it'** with the correct word.

Bahdanau, Dzmitry, Kyunghyun Cho, and Yoshua Bengio. "Neural machine translation by jointly learning to align and translate." arXiv preprint arXiv:1409.0473 (2014).

Self-Attention

- In the Self-Attention layer (in both encoder and decoder), the input is passed to three parameters: Key, Query and Value.
- The weighted values are computed by "a compatibility function of the query with the corresponding key".
- Key, Query and Value matrices are parameters whose initial weights are small, randomly selected numbers. These parameters change as the model is trained on the dataset.



How to Calculate Attention for Each Input

- Each sequence (sentence) contains several words. The attention score must be calculated for each word.
  - First all the inputs need to be passed through embedding and positional encoding.
  - Key, Query and Value matrices need to be initialized randomly (their values will be optimized as the model is trained)
    - The size of all matrices are the same and depends on the number of words in a sequence and the length of embedding.
  - Each vectorized word is multiplied by Key, Query and Value matrices to create key, query and value representation vector.

How to Calculate Attention for Each Input

- Each sequence (sentence) contains several words. The attention score must be calculated for each word.
  - First all the inputs need to be passed through embedding and positional encoding.
  - Key, Query and Value matrices need to be initialized randomly (their values will be optimized as the model is trained)
    - The size of all matrices are the same and depends on the number of words in a sequence and the length of embedding.
  - Each vectorized word is multiplied by Key, Query and Value matrices to create key, query and value representation vector.

Calculate Attention Score

- This is where we can observe how self-attention is considering other words.
- This is the equation to compute each word’s attention score where d_k is the embedding size, Q is the query matrix, K is the Key matrix and V is the Value matrix.

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

Calculate Attention Score

- This is where we can observe how self-attention is considering other words.
- This is the equation to compute each word’s attention score where d_k is the embedding size, Q is the query matrix, K is the Key matrix and V is the Value matrix.

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

Summary

We learned about transformer models

- Encoder and decoder
- Why using transformer models
- Attention in transformers and why they are needed in such models