

Attention is all you need?

Presented By : Dolev Abuhazira , Dor Azaria.

As part of the seminar specializing in deep learning, we will present the idea behind the famous article “Attention is all you need”

Credit :

Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gómez, Lukasz Kaiser, Illia Polosukhin

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
lukaszkaizer@google.com


Illia Polosukhin*
illia.polosukhin@gmail.com

Abstract

The dominant sequence transduction models are based on complex recurrent or convolutional neural networks in an encoder-decoder configuration. 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, our model establishes a new single-model state-of-the-art BLEU score of 41.0 after training for 3.5 days on eight GPUs, a small fraction of the training costs of the best models from the literature. We show that the Transformer generalizes well to other tasks by applying it successfully to English constituency parsing both with large and limited training data.

Sequence Models

The basic idea of sequence models is very similar to other deep learning model. The main characteristic that we need to ensure is that the model can handle input sequences of different lengths.

- **Input** : Raw sequence data (without feature extraction)
 - **Layers** : Sequence to sequence layers
 - **Output** : Sequence (next token prediction , translation) single vector (classification , regression)
- 

Word Representation

A neural network does not understand words in any language , Therefore we need to convert the words into vectors.

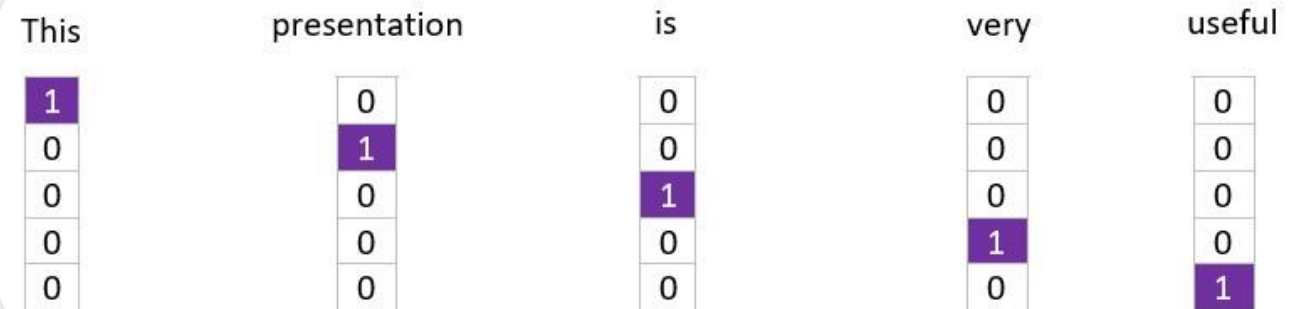
Word embedding is a means of turning texts into numbers , We do this because machine learning algorithms can only understand numbers, not plain texts.

One-Hot Encoding

creates new (binary) columns, indicating the presence of each possible value from the original data.

Disadvantages :

- leads to creating a new dimension for each new word.
- a lot of data is reset, which makes learning difficult.
- tells you nothing of the semantics of the words.

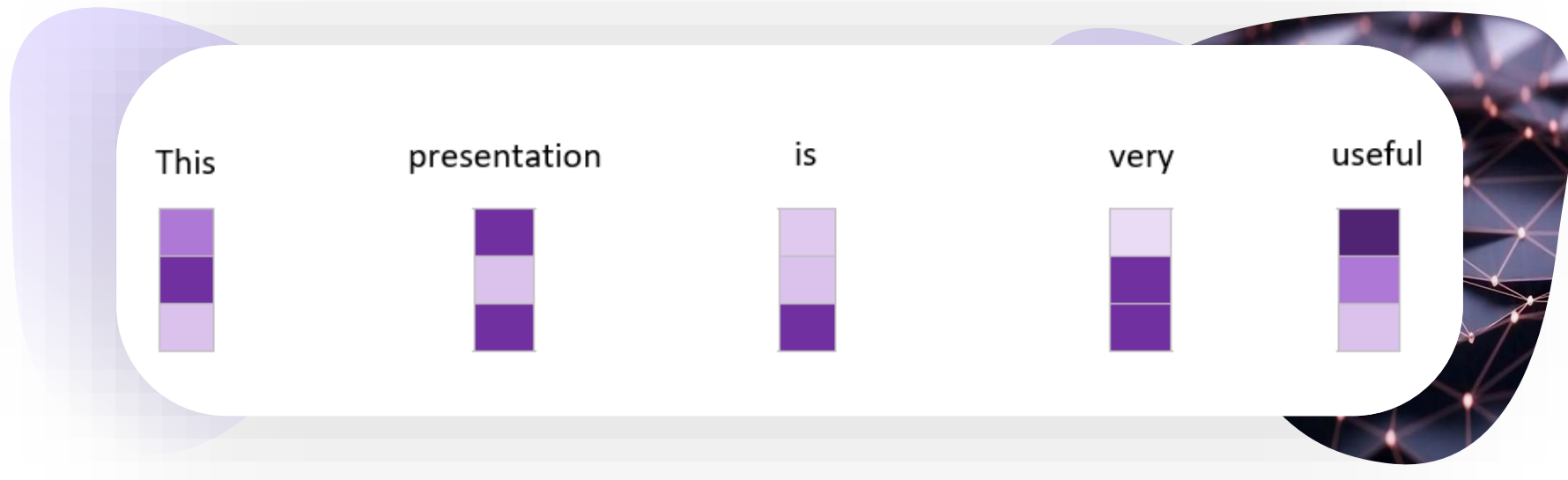


This	presentation	is	very	useful
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

Word Embedding

In general, an embedding is a representation of a symbol (word, character, sentence) in a distributed low-dimensional space of continuous-valued vectors.

The idea here is that you assign each input symbol in your vocabulary a vector of random values. You then translate a symbolic input sequence into a sequence of vectors by mapping the input symbol to their corresponding embedding vectors. The dimensionality of the embedding vectors in the article is 512.



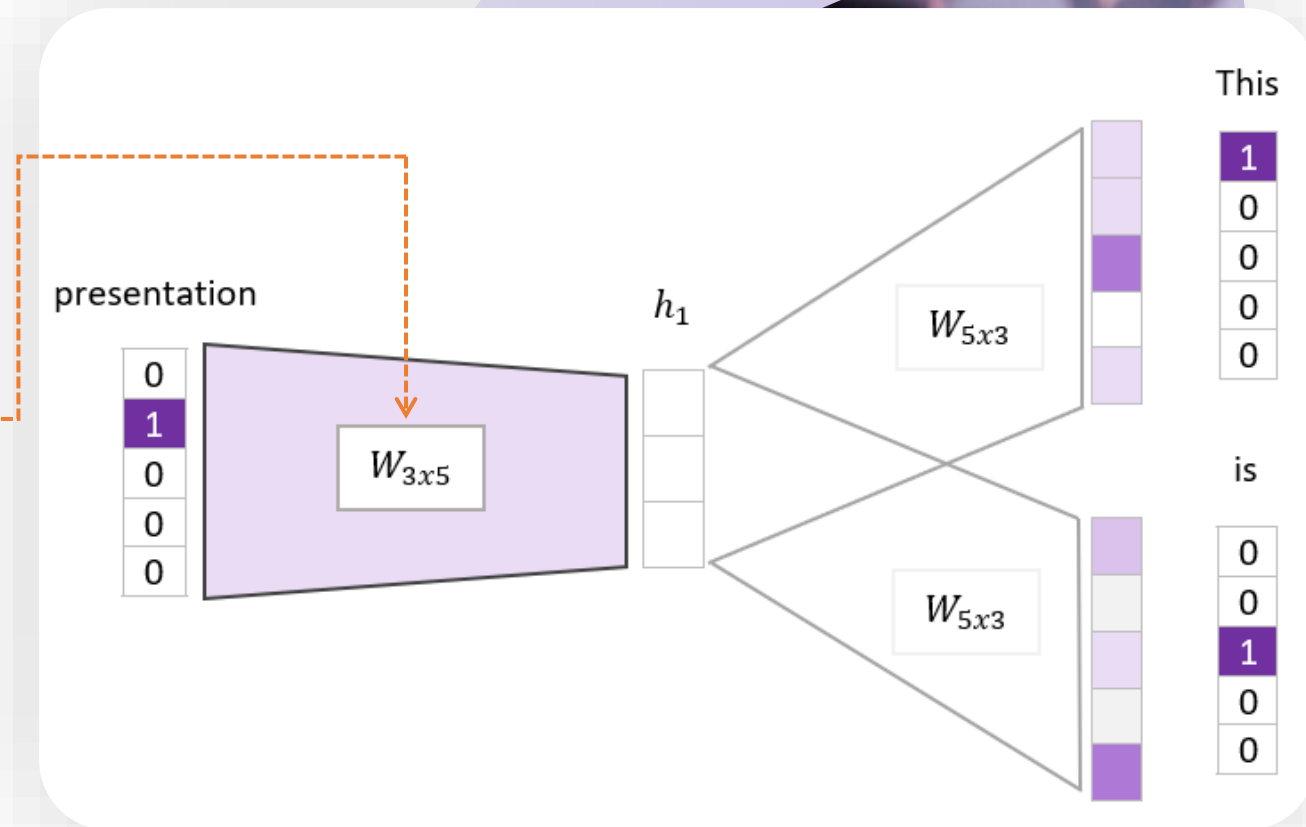
How can we embed the words?

Skip Gram model : predicts words within a certain range before and after the current word in the same sentence.

Given a word we will try to predict its features based on its neighborly words in the sentence.

In fact, we will not be interested the output of the neural network but only in the matrix of weights, the same weights indicate the latent variables.

When we want to extract the resulting representation in the weight matrix, we will multiply the matrix in the One Hot encoding of the same word.



Does the order matter?

Even though she did **not** win the award, she was satisfied.

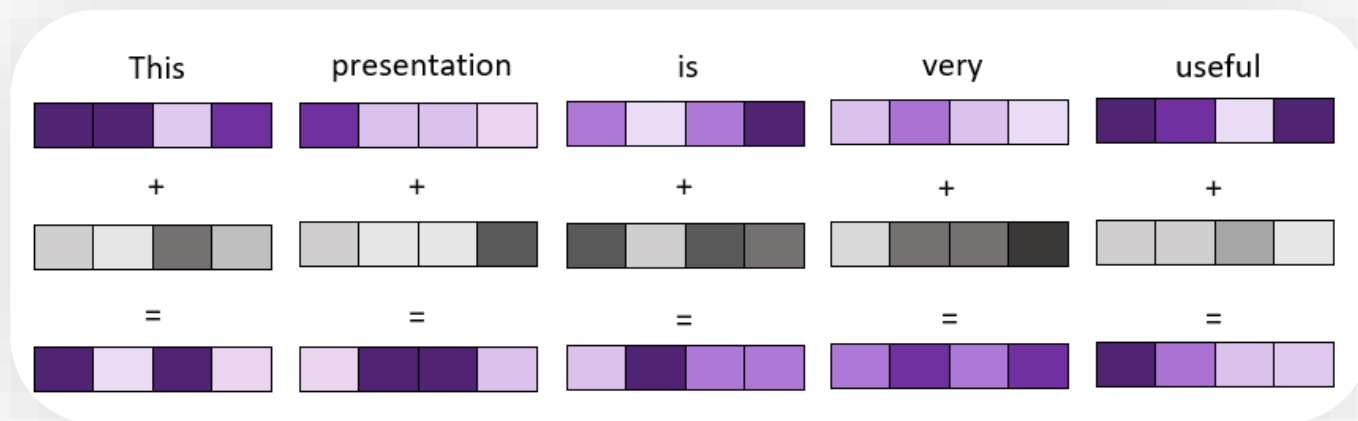
Even though she did win the award, she was **not** satisfied.

- Position and order of words are the essential parts of any language. They define the grammar and thus the actual semantics of a sentence.
- Transformers doesn't have any sense of position/order for each word. Consequently, there's still the need for a way to incorporate the order of the words into our model.
- We want to give the model some sense of the order like add a piece of information to each word about its position in the sentence.

Positional Encoding

Is a vector which are added to the word embedding vector .

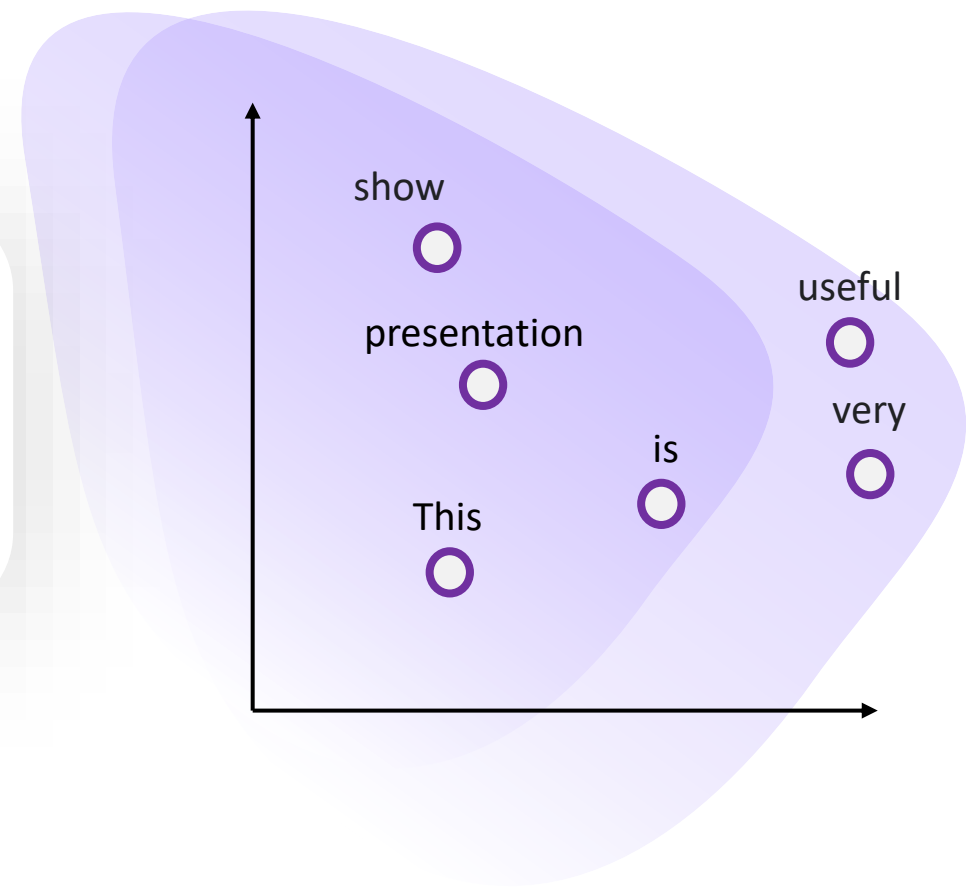
So if the same word appears in a different position, the actual representation will be slightly different, depending on where it appears in the input sentence.



In the paper , the authors came up with the sinusoidal function for the positional encoding :

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

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



Does the context matters?

I **drove** across the **road** to get to the other **bank**

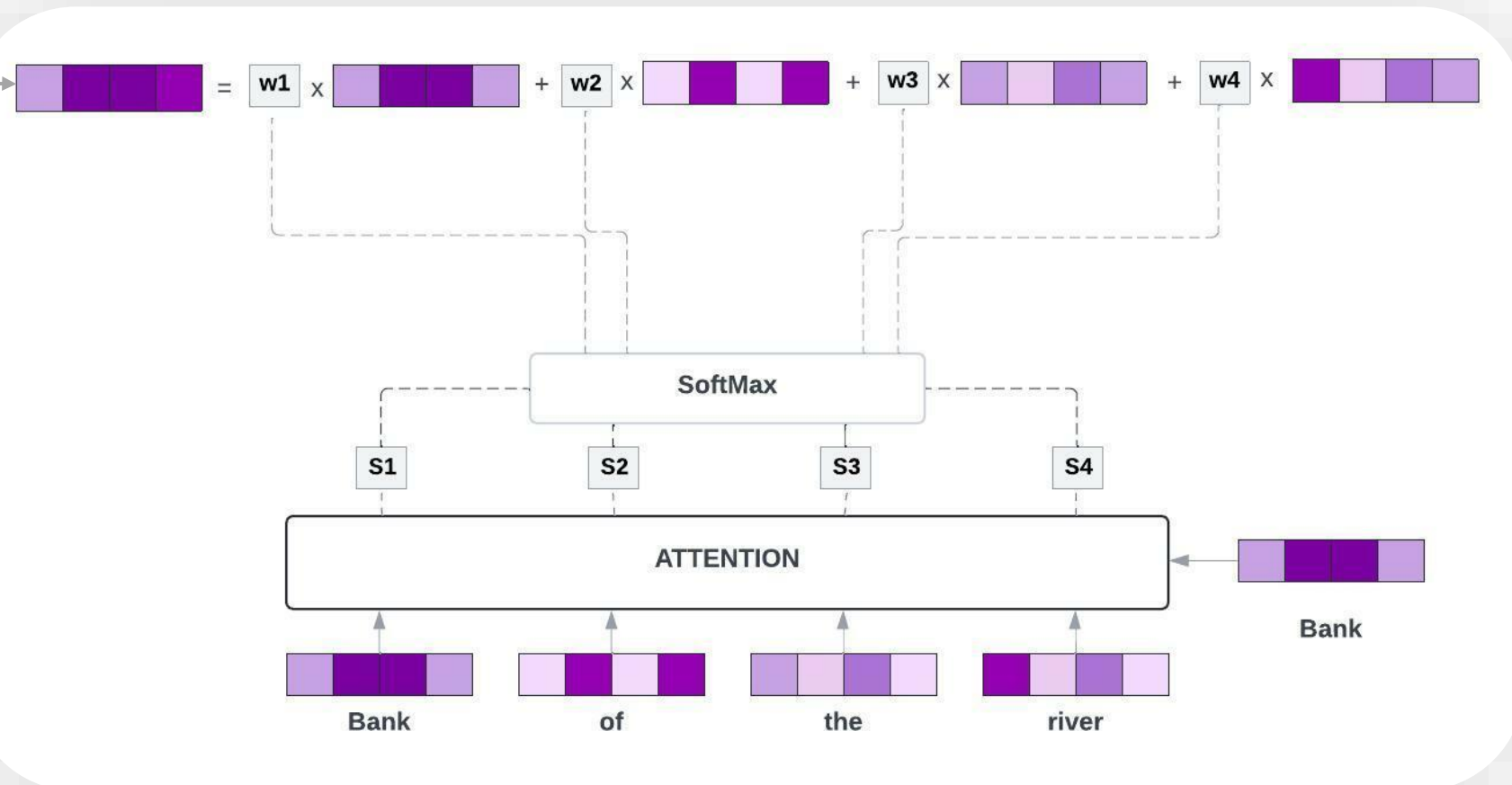
I **swam** across the **river** to get to the other **bank**

Can we built a mechanism that weighs neighboring words to enhance the meaning of the word of interest?

WE NEED TO PAY **ATTENTION**

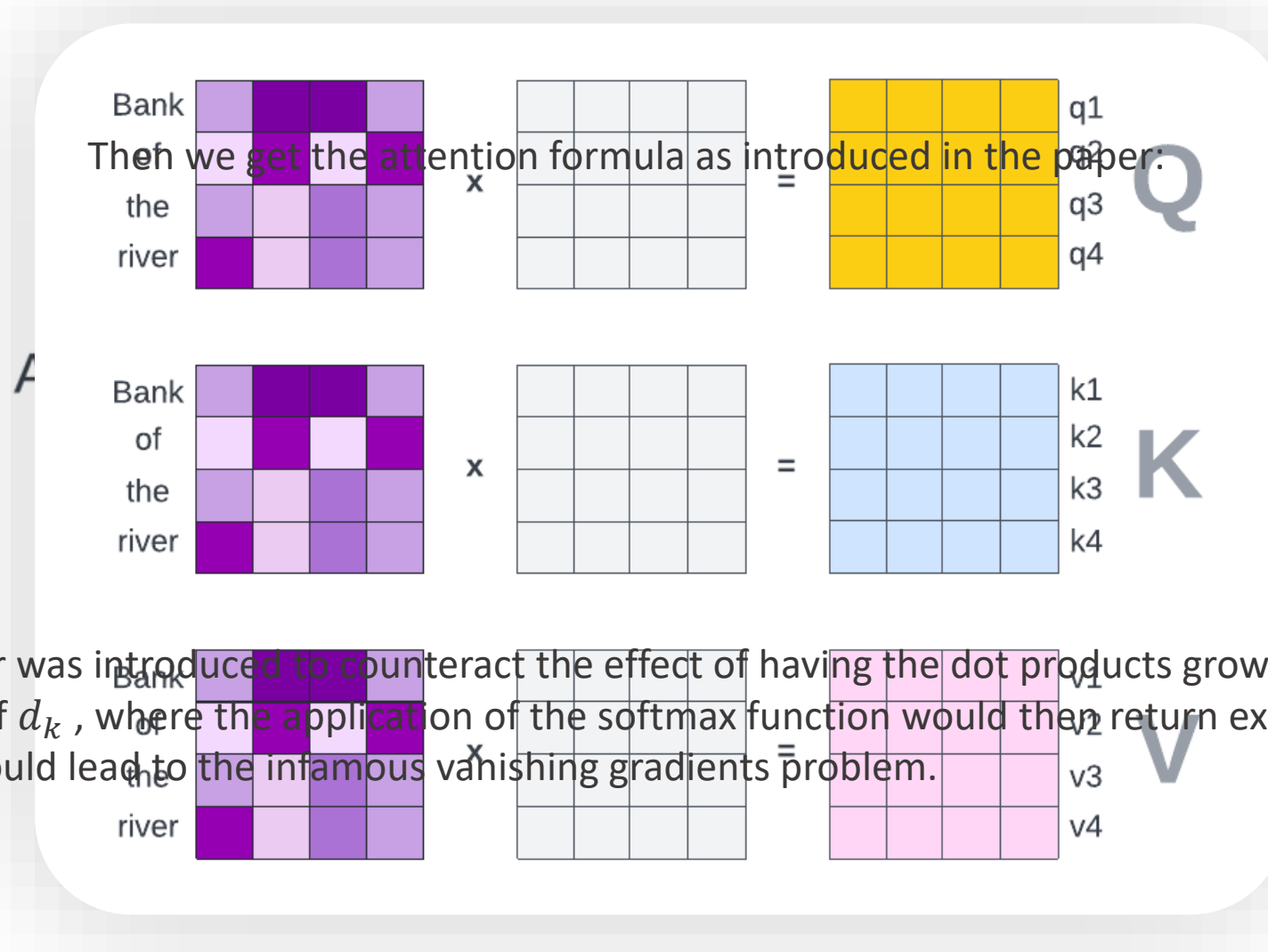
- The idea is basically to add more information to the embedding vector of the different words relative to the other words in the sentence.
- The fundamental operation of any transformer architecture is the **self-attention** operation.
- Self Attention looking for **scores** that represent the similarity between the words , the dot product used to determine the similarity between two vectors.
- If two vectors are close to one another , their dot product is going to be large.

Now , the embedding vector pointing in a different direction which is influence by his neighbors



The previous basic version did not include learning parameters, and this version is not particularly useful for learning a language model

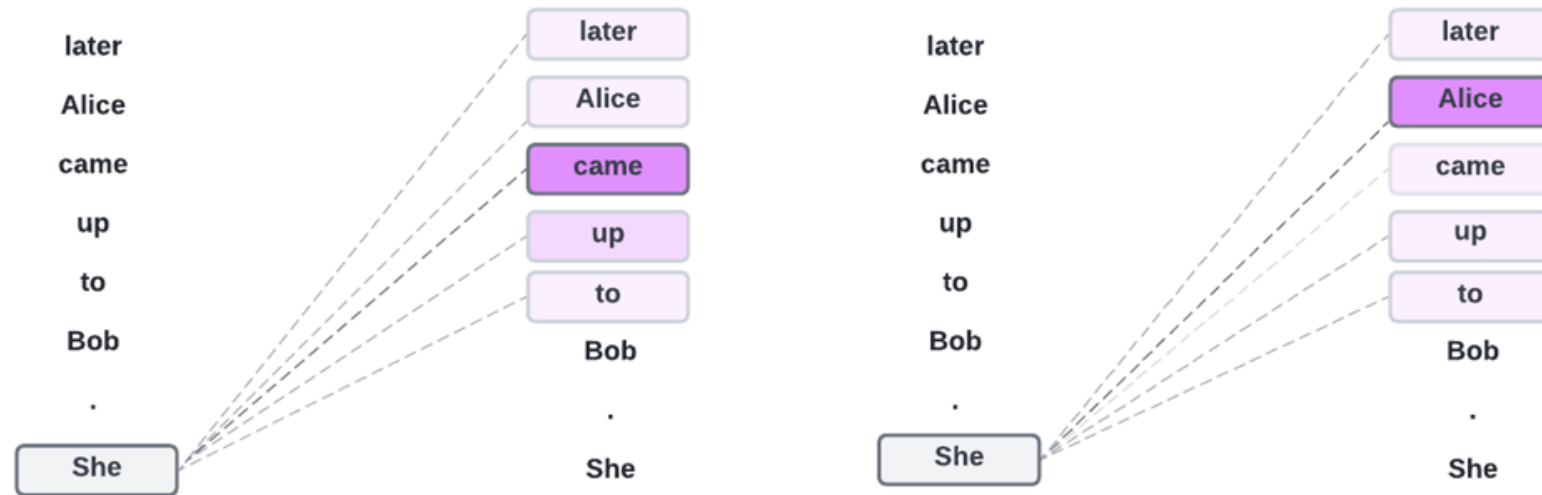
We are now adding 3 trainable weight matrices that are multiplied with the input sequence embeddings



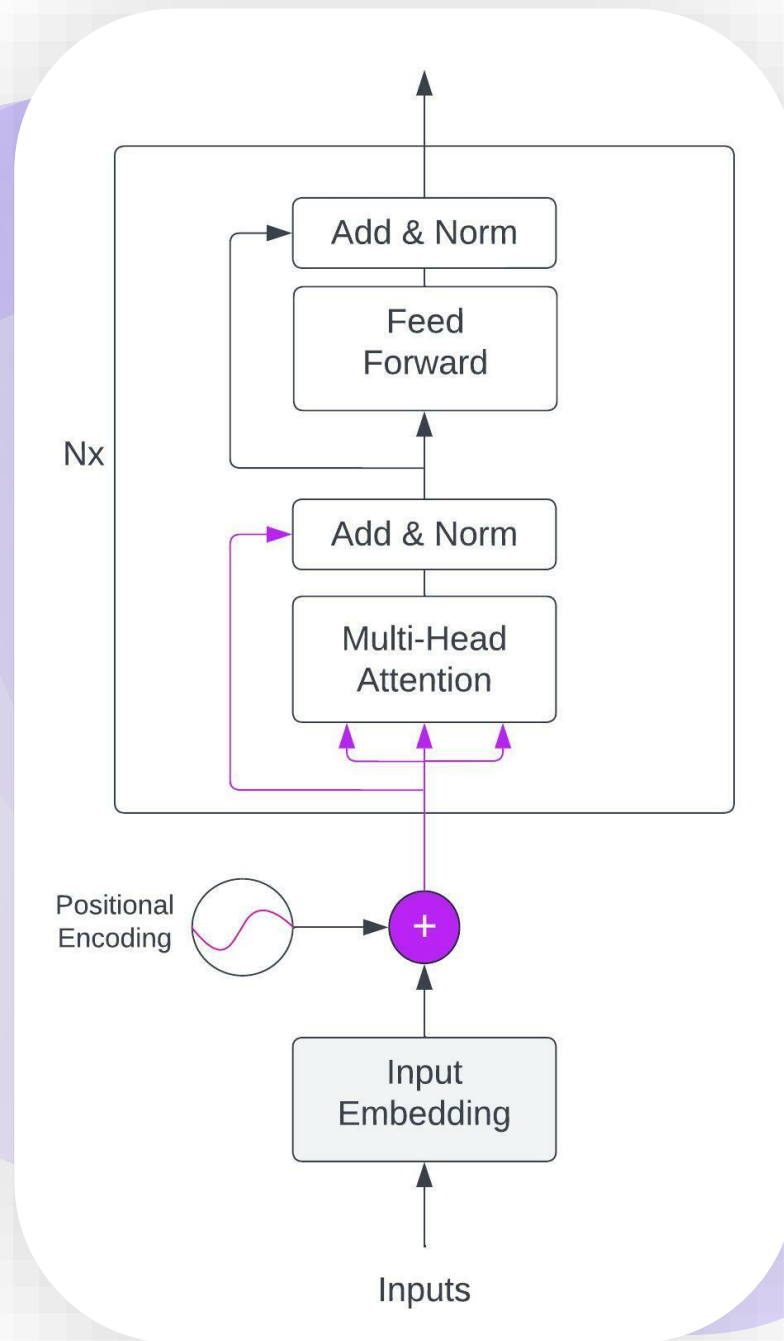
This scaling factor was introduced to counteract the effect of having the dot products grow large in magnitude for large values of d_k , where the application of the softmax function would then return extremely small gradients that would lead to the infamous vanishing gradients problem.

Multi Head Attention

- separate sections of the Embedding can learn different aspects of the meanings of each word, as it relates to other words in the sequence. This allows the Transformer to capture richer interpretations of the sequence.
- It may not be a realistic example, but it might help build intuition. For example, one section may capture the 'gender-ness' (male, female, castration) of a noun, while another may capture the verb that refers the same noun.



Transformers Encoder

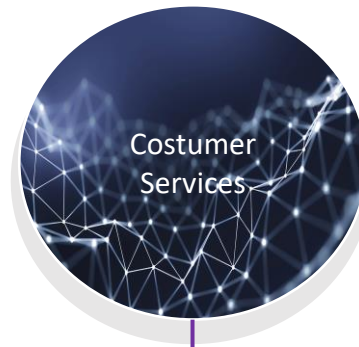


Sentiment Analysis

- Sentiment analysis is the use of natural language processing, text analysis, computational linguistics, and biometrics to systematically identify, extract, quantify, and study affective states and subjective information. Sentiment analysis is widely applied to voice of the customer materials such as reviews and survey responses, online and social media, and healthcare materials for applications that range from marketing to customer service to clinical medicine. (Wikipedia)



It helps to detect and understand the emotions of the people.



Its help understand the opinion of the costumer.



It helps in studying the ratings and reviews given by the customers

BERT pre-trained from unlabeled data extracted from the BooksCorpus with 800M words and English Wikipedia with 2,500M words with the following two tasks :

1. Masked Language Model (MLM)
 1. 15% of the words are masked with [MASK] token.
 2. Predict masked words during training.
 2. Next Sentence Prediction (NSP)
 1. Select sentence B as the sentence after A with 50% probability.
 2. Use [CLS] output to classify B as next/other sentence.
- The output of [CLS] is concluded by all other words in the sentence, so [CLS] contains all information in other words.

