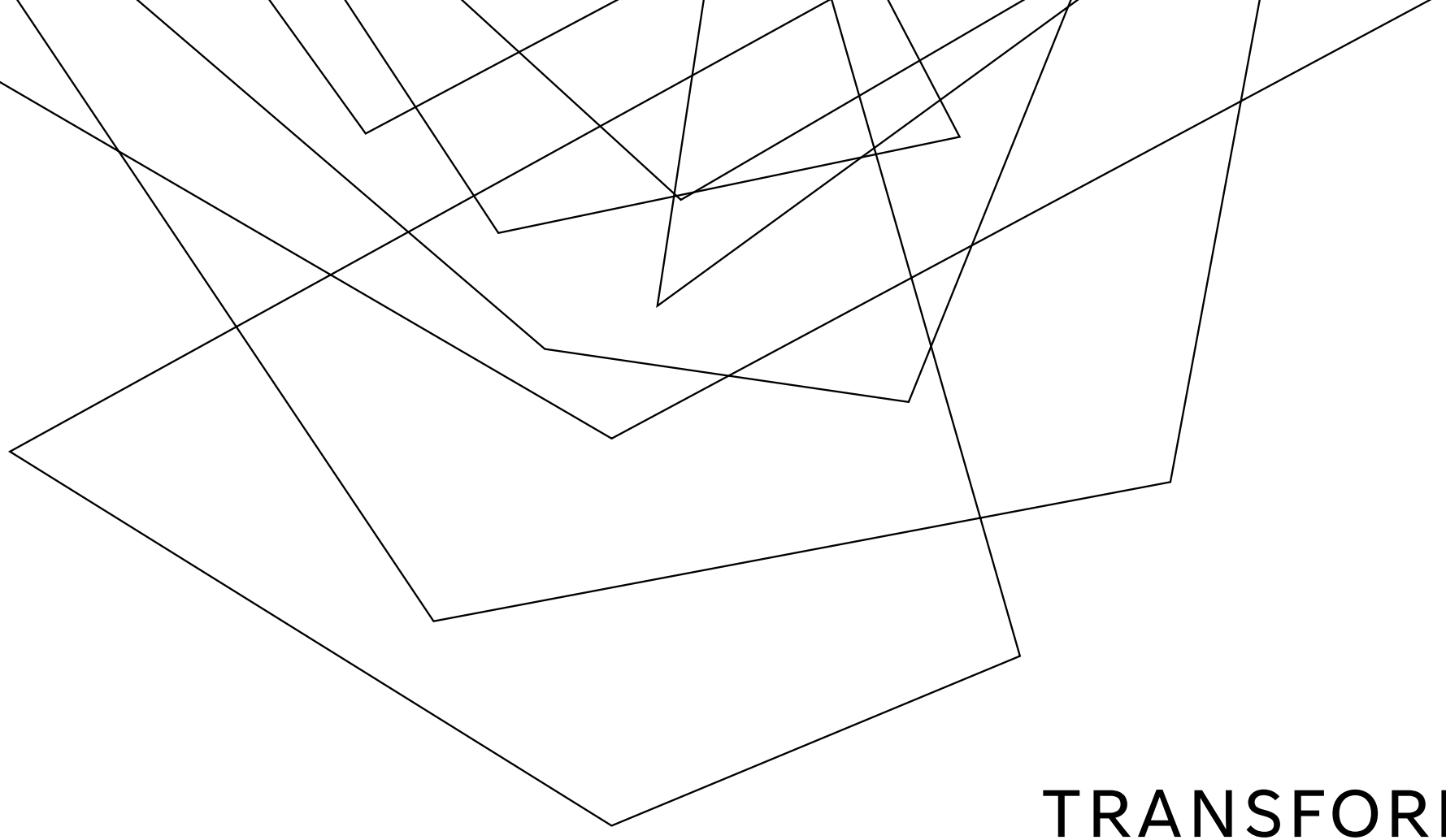


TRANSFORMERS

-MEETU

AGENDA

Word Embeddings

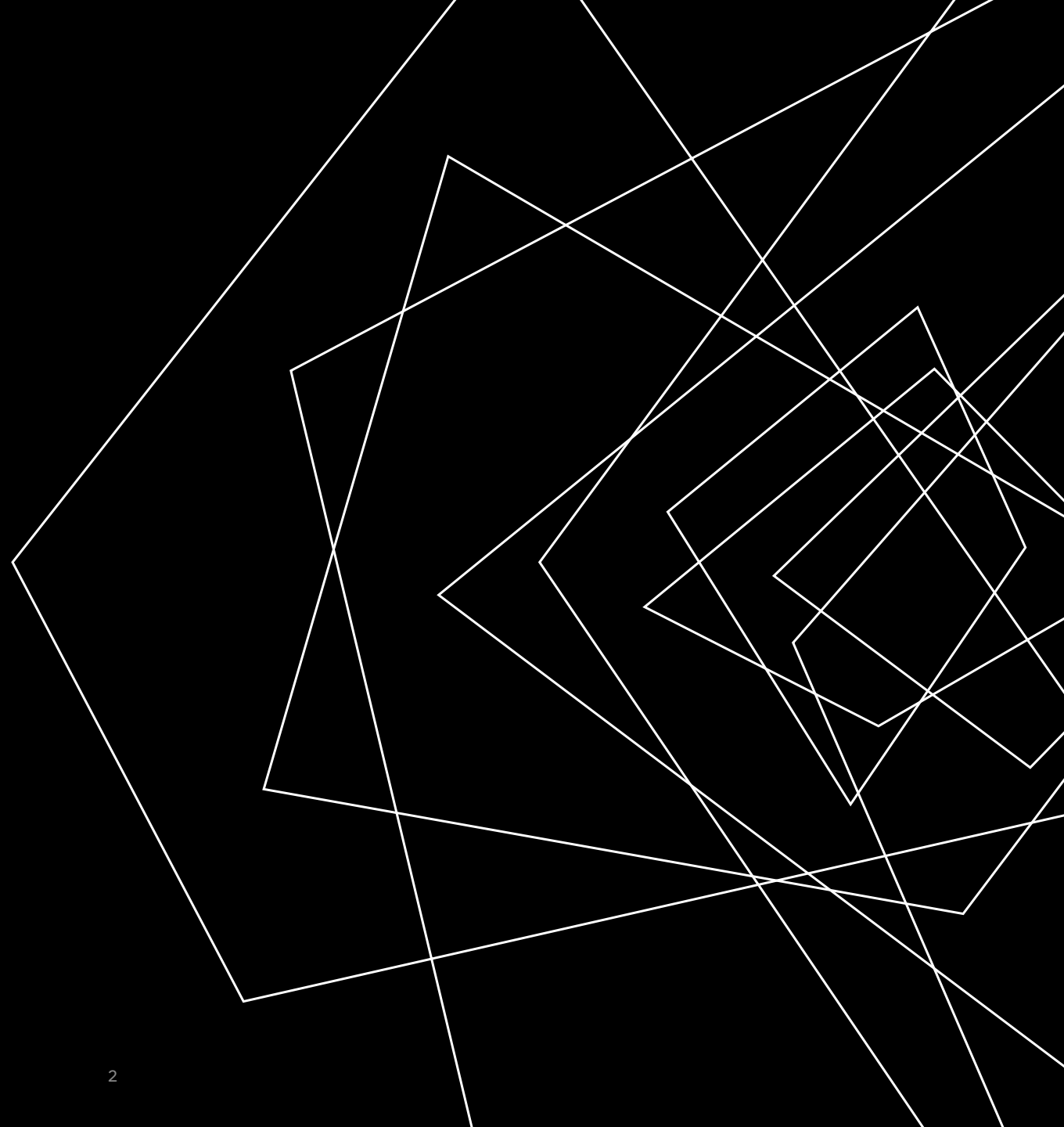
Introduction about Transformer

Understanding important terms –

- Attention
- key-Query- Values
- Positional embeddings
- self Attention

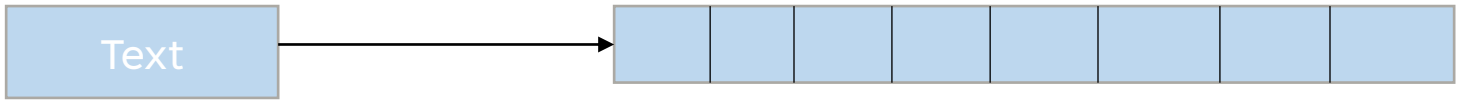
Applications of transformers

Implementation using Hugging Face platform



WORD EMBEDDINGS

In order to feed out text data into a neural network, we need to first encode it, meaning transferring these words to numbers with a fixed-length vector



There are many types of word embeddings such as count vectors, TF-IDF vectors, Word2Vec etc. which are used in neural networks to perform NLP related tasks such as topic modeling, next word prediction, information retrieval etc.

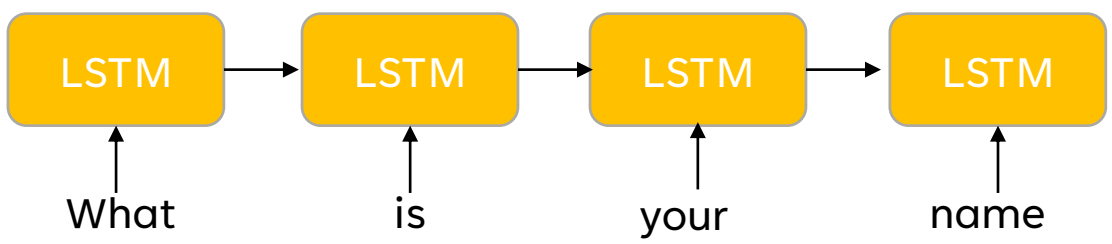
RNNs are used to handle these embeddings, but there are certain drawbacks –

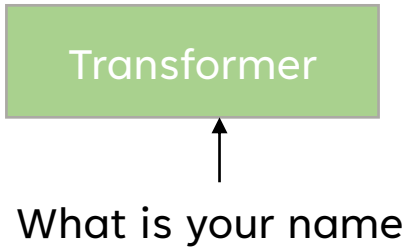
- They cannot retain the contextual meaning especially in long sentences (more than 30 words)
- They are used in sequential manner, so computationally very expensive on complex tasks such as text translation, question-answer text generation.

WHAT IS A TRANSFORMER?

Introduced in 2017, a transformer is a type of neural network or attention-based NN, which processes complex NLP tasks involving long sequences of data in text generation, summarization, generating codes, writing poems etc.

Transformers consider the entire input sequence at once rather than processing it sequentially





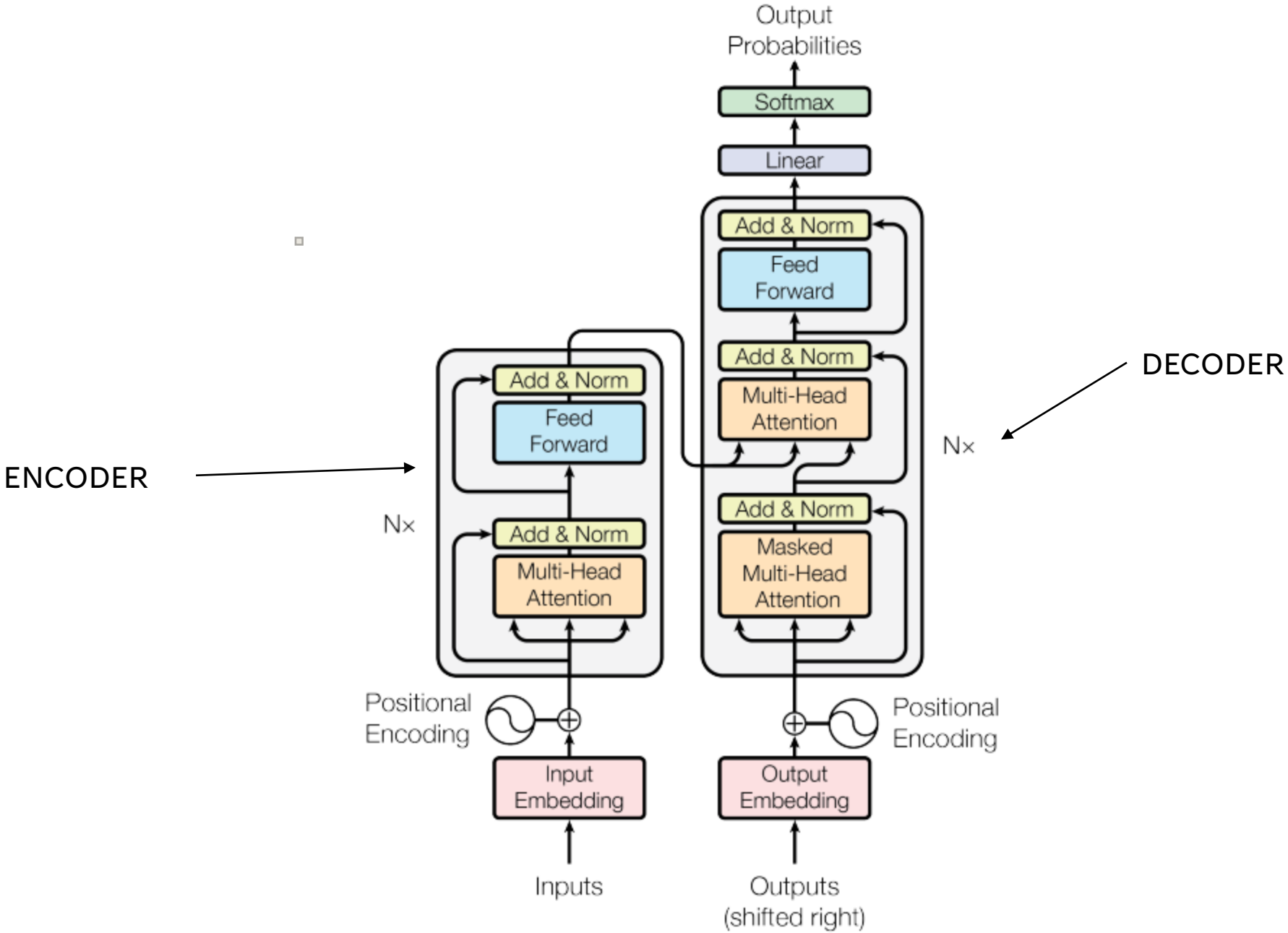


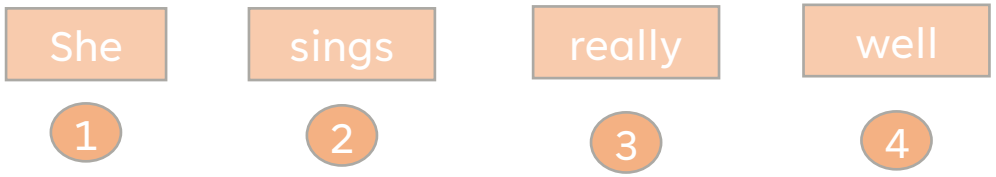
Figure 1: The Transformer - model architecture.

3 KEY FEATURES IN A TRANSFORMER

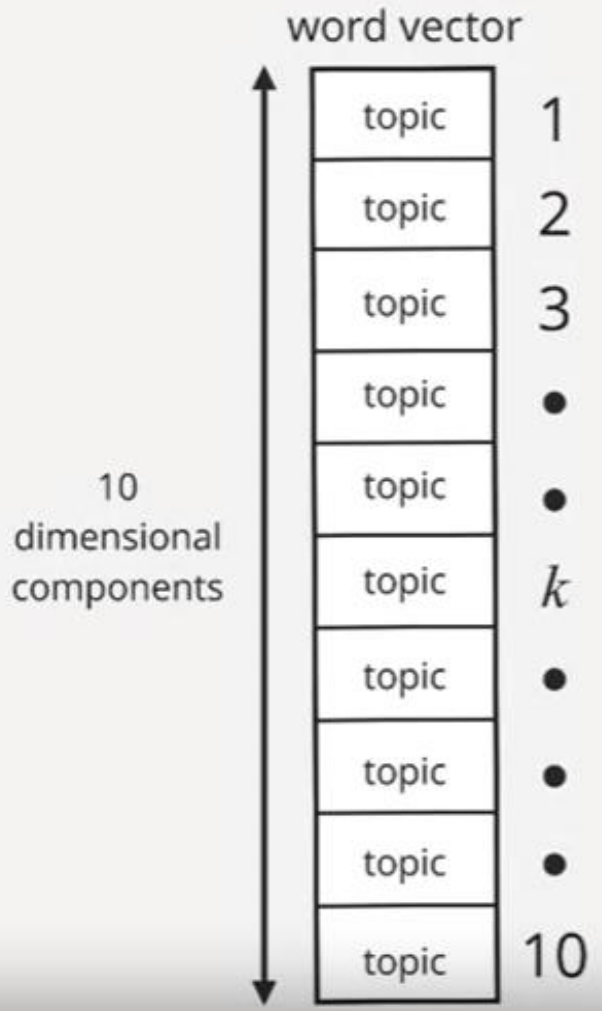
1. It is not sequential in nature. It can work on word embeddings in parallel

2. Attention/Self attention mechanism – To capture the context of a word with respect to surrounding words, attention mechanism is used. In simple terms, it can differentiate the word “bank” (riverbank or financial institute) by considering its surrounding words

3. Positional Encoding – in this a number is assigned to each token in a sentence. So now instead of feeding the tokens, these numbers also are fed into the NN as a part of input text data that helps NN to learn about the word order

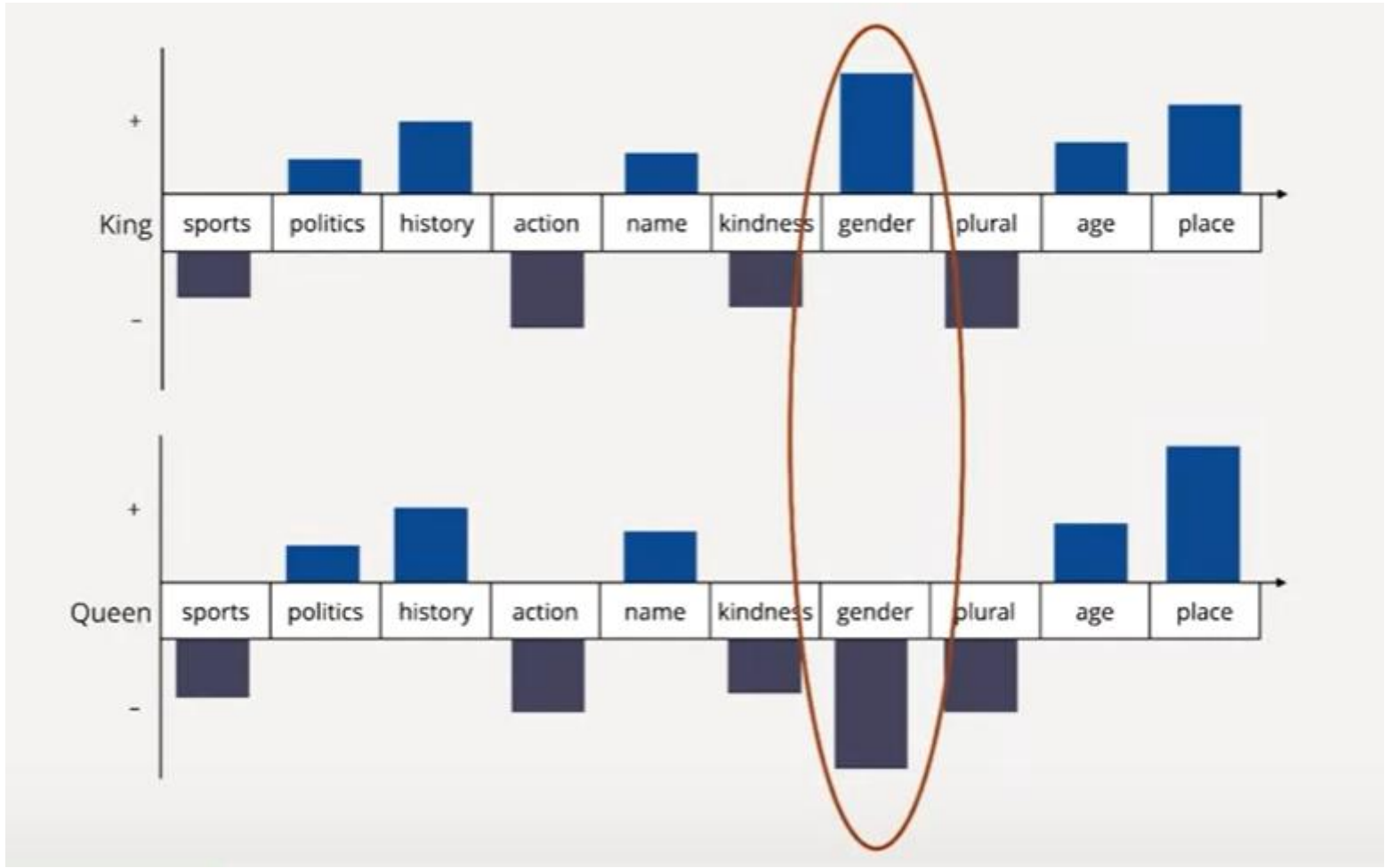


INSIDE THE TRANSFORMER



Conceptualization of Meaning of Word Vectors

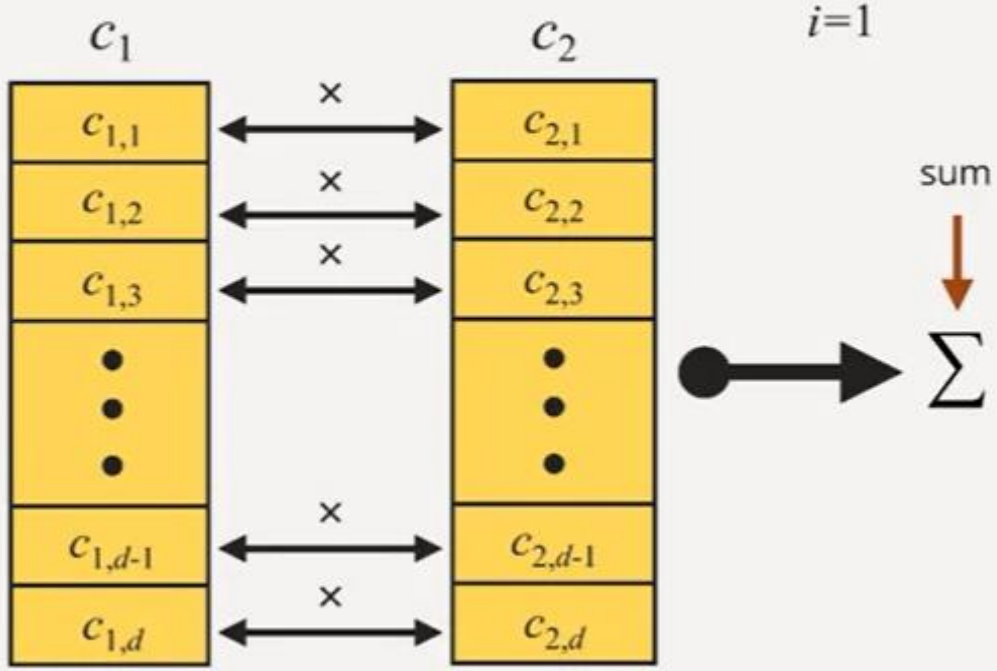
- Each component of the word vector may be viewed as a "topic"
- Each topic represents characteristics of words
- If word is aligned with Topic k, Component k will be a positive number
- If word is not aligned with Topic k, Component k will be a negative number


+
King
sports
politics
history
action
name
kindness
gender
plural
age
place
-
+
Queen
sports
politics
history
action
name
kindness
gender
plural
age
place
-

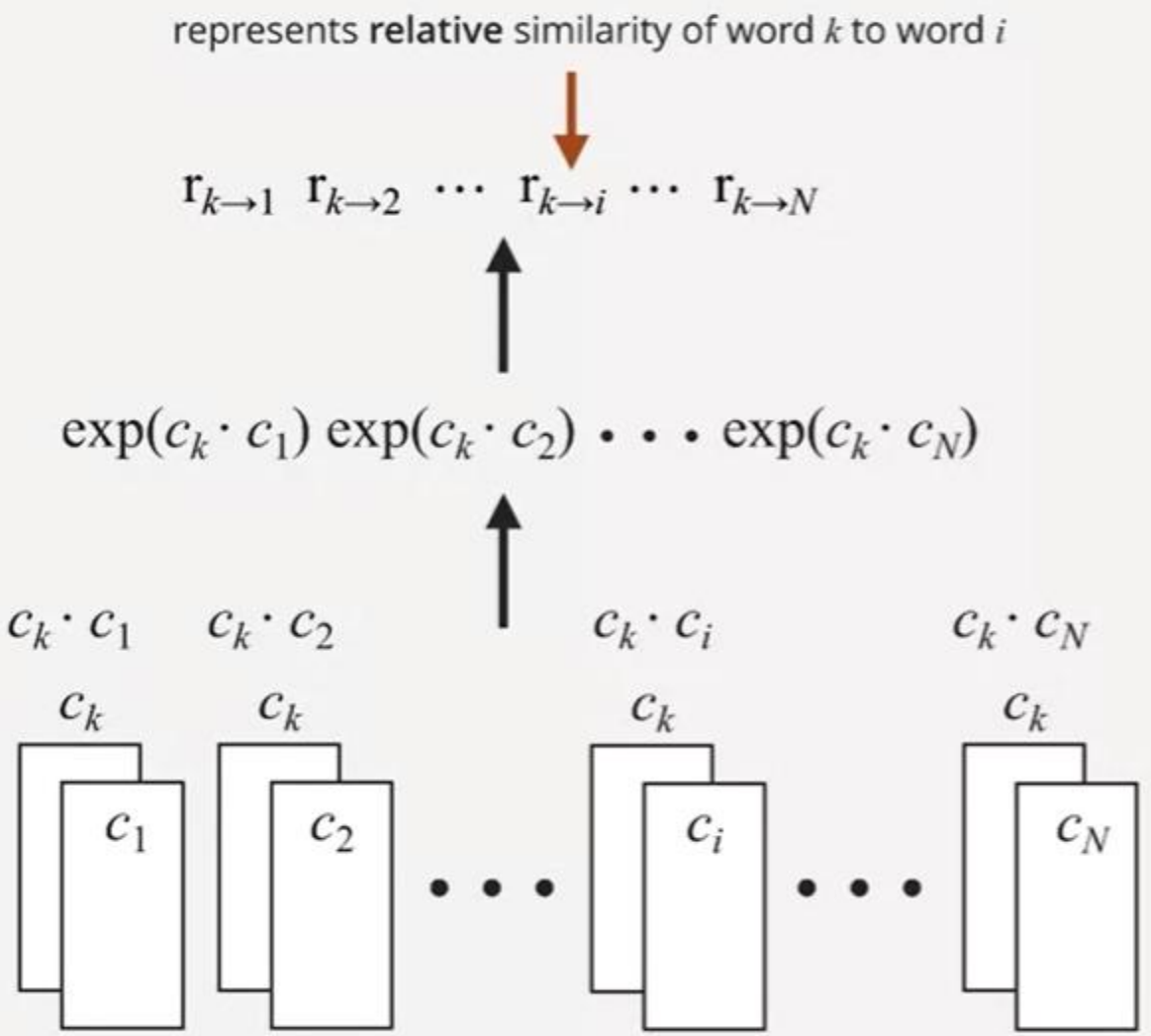
INNER(DOT) PRODUCT OF WORD EMBEDDINGS

$$c_1 \cdot c_2 = (c_{1,1} \times c_{2,1}) + (c_{1,2} \times c_{2,2}) + (c_{1,3} \times c_{2,3}) + \cdots + (c_{1,d} \times c_{2,d}) = \sum_{i=1}^{d} c_{1,i} \times c_{2,i}$$

inner product
(dot product)



RELATIVE SIMILARITY



Quantify How Similar All Words are to Word k

1. Take k^{th} word and perform inner product between each word
2. Exponentiate inner products to make them positive
3. Get a relative representation for the strength of the inner product

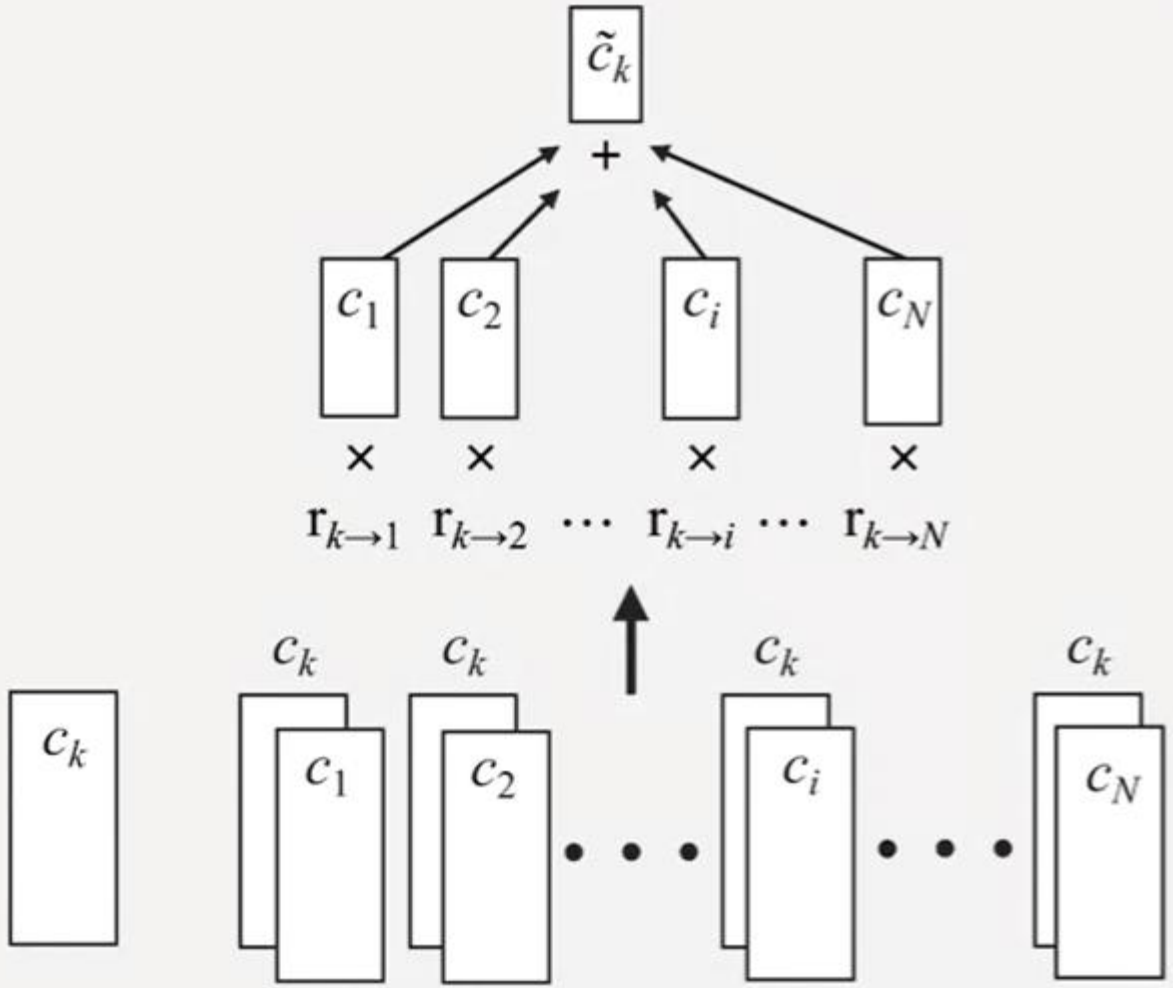
$$r_{k \to i} = \frac{\exp(c_k \cdot c_i)}{\exp(c_k \cdot c_1) + \exp(c_k \cdot c_2) + \exp(c_k \cdot c_3) + \cdots + \exp(c_k \cdot c_N)}$$

simplified representation

$$= \frac{\exp(c_k \cdot c_i)}{\sum_{n=1}^{N} \exp(c_k \cdot c_n)} \geq 0$$

These relative weights are always between 0 and 1
Larger is the value, larger is the degree of correlation between the two words
Until this point, we have not considered “context” of the word

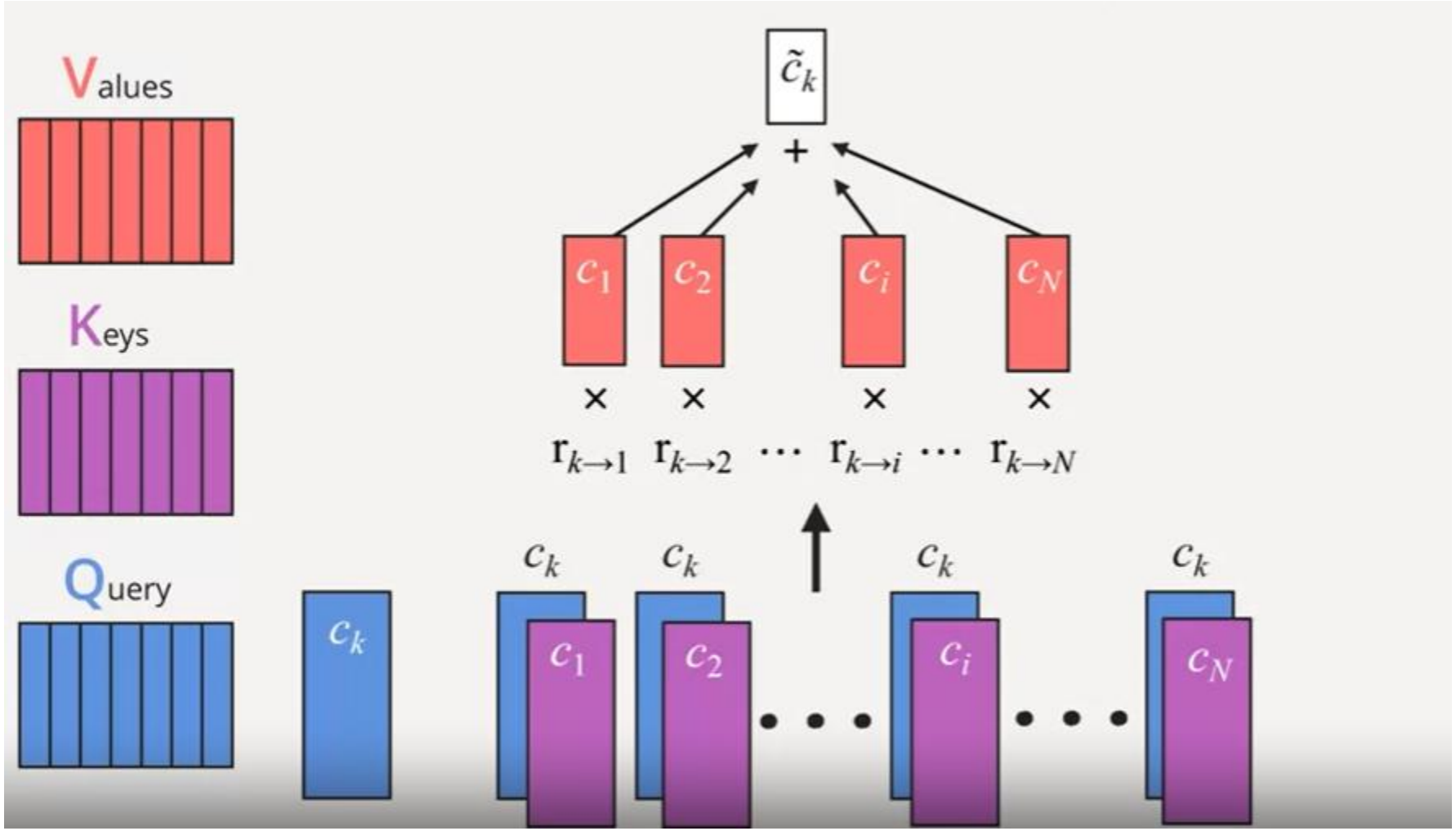
$$\tilde{c}_k = (\mathrm{r}_{k\to 1} \times c_1) + (\mathrm{r}_{k\to 2} \times c_2) + \cdots + (\mathrm{r}_{k\to i} \times c_i) + \cdots + (\mathrm{r}_{k\to N} \times c_N) = \sum_{n=1}^{N} \mathrm{r}_{k\to n} \times c_n$$



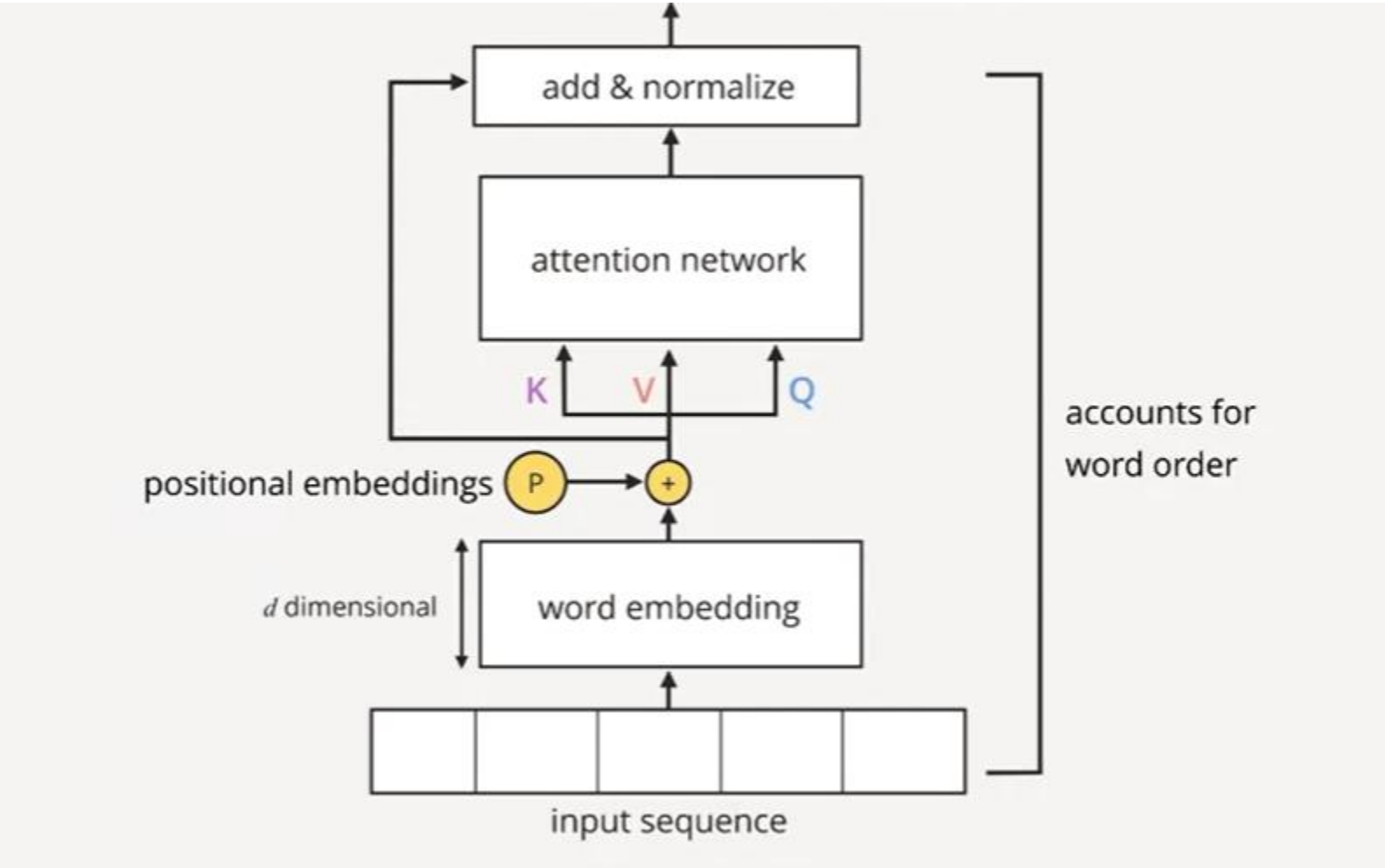
Concept of Attention

- Relational numbers tell us how much attention to pay to corresponding code

QUERIES, KEYS AND VALUES



ADDING POSITIONAL EMBEDDING



APPLICATIONS OF TRANSFORMERS

- **Sentiment analysis:** is a text positive or negative?
- **Text generation (in English):** provide a prompt, and the model will generate what follows
- **Name entity recognition (NER):** in an input sentence, label each word with the entity it represents (person, place, etc.)
- **Question answering:** provide the model with some context and a question, extract the answer from the context
- **Summarization:** generate a summary of a long text
- **Language Translation:** translate a text into another language

IMPLEMENTATION USING HUGGING FACE PLATFORM

Hugging Face Transformers is a platform that provides the community with APIs to access and use SOTA pre-trained models available on its website

It’s a platform with 60K models, 6K datasets and 6K demos, which helps people collaborate in their ML flows

The APIs are based on libraries such as Pytorch, Tensorflow, fastai etc.

The Hugging Face Transformers have **pipelines**, which provide an easy-to-use API through pipeline() method. It encapsulates the overall process of every Natural Language Processing task, such as text cleaning, tokenization, embedding and decoding.

Basic structure of pipeline :

```
from transformers import pipeline
# To use an existing pre-trained model
pipeline("<task-name>", model="<model_name>")
```

REFERENCES:

https://www.datacamp.com/tutorial/an-introduction-to-using-transformers-and-hugging-face

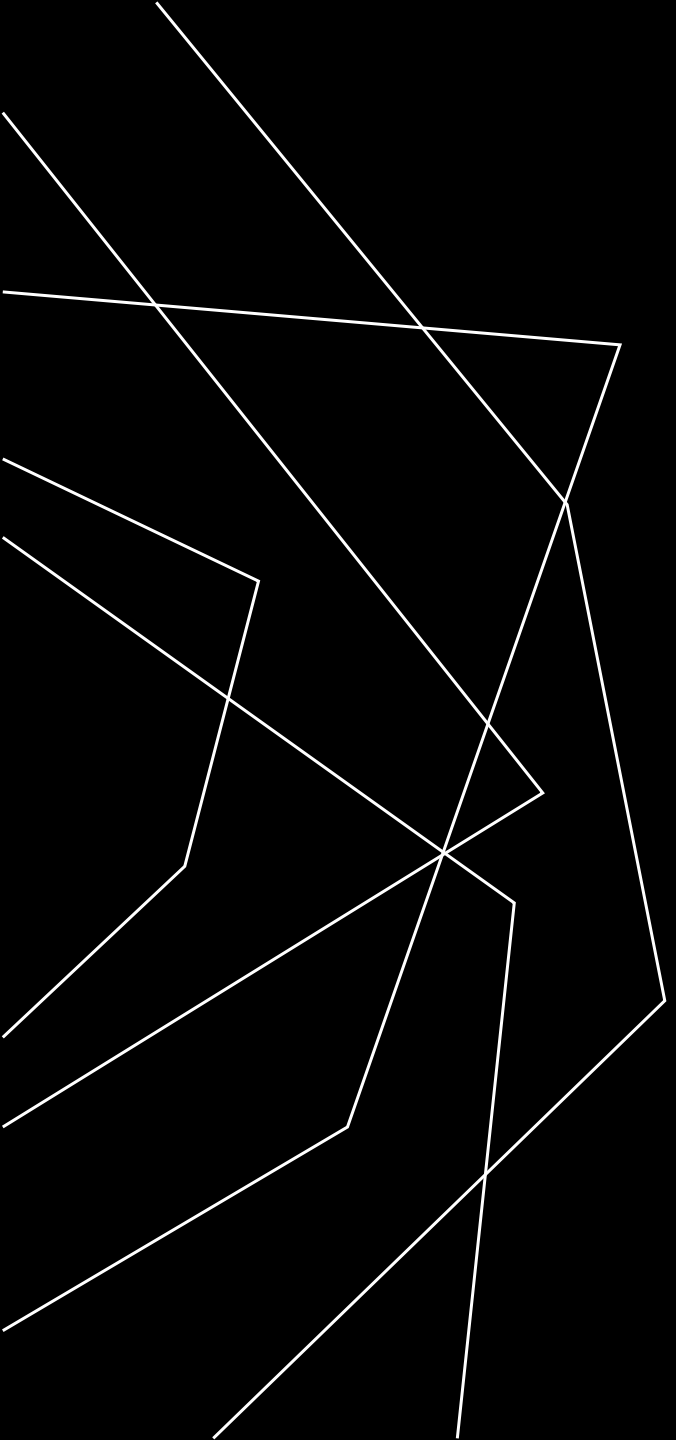
https://jalammar.github.io/illustrated-transformer/

Vaswani, Ashish & Shazeer, Noam & Parmar, Niki & Uszkoreit, Jakob & Jones, Llion & Gomez, Aidan & Kaiser, Lukasz & Polosukhin, Illia, “Attention is all you need” , 2017.

https://www.coursera.org/learn/machine-learning-duke/lecture/AMoyH/self-attention-and-positional-encodings

https://peterbloem.nl/blog/transformers

https://huggingface.co/docs/transformers/autoclass_tutorial



THANK YOU