Llm presentation

Transformer architecture, activation...

Tilian Bourachot

June 16, 2024

Table des matières

Large Language Models (LLMs)

- **Large Language Models (LLMs)** use advanced deep learning techniques to intelligently understand and generate natural language. Their goal is to generate coherent and plausible text based on a set of training data. Their functionality is based on transformer architectures.
- **Language models (LM)**: A model trained to predict the next word in a sequence, given the words that come before it. It is a probabilistic model capable of generating text that follows a certain linguistic style or pattern (e.g., GPT, BERT).

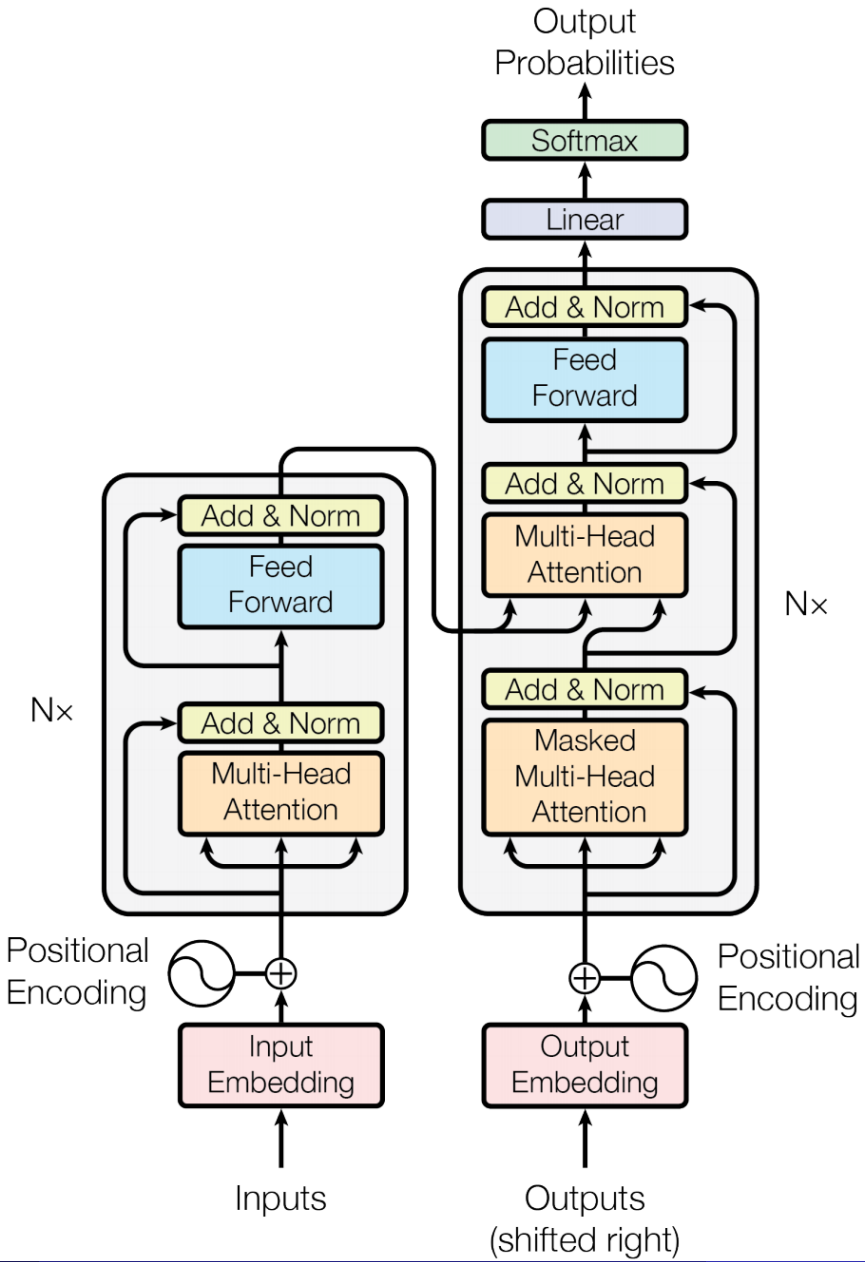
Example Models

- **n-gram models**
- **Hidden Markov Models (HMMs)**
- **Recurrent Neural Networks (RNNs):**
  - RNNs process sequences using internal state (memory), making them suitable for language modeling but struggling with long-range dependencies due to the vanishing gradient problem.
- **Long Short-Term Memory (LSTM):**
  - A specialized RNN designed to capture long-term dependencies using gates that control information flow in and out of the network's memory state.
- **Gated Recurrent Unit (GRU):**
  - A simplified version of LSTM with different gating mechanisms, often faster to train but performance varies by task.

GPT

- LLM as GPT3 or GPT4 are symply LMs that are trained on a very large amount of text and have a very large number of parameters.
- Important advantage : we can train them unsupervised on a large corpus of data and then fine-tune with a limited amount of data for different tasks.

Introduction to Transformers



Introduction to Transformers

1. Enables machines to understand the context of the words in sentences more effectively.
2. Can handle words in parallel (unlike RNNs, which are sequential).
3. Uses attention mechanisms to understand word context within sentences.

Introduction to Transformers

- Pre-trained models (BERT, GPT) are trained on vast amounts of text data.
- They can be fine-tuned for specific tasks, such as:
  - Sentiment analysis
  - Language translation
  - Text summarization

Introduction to Transformers

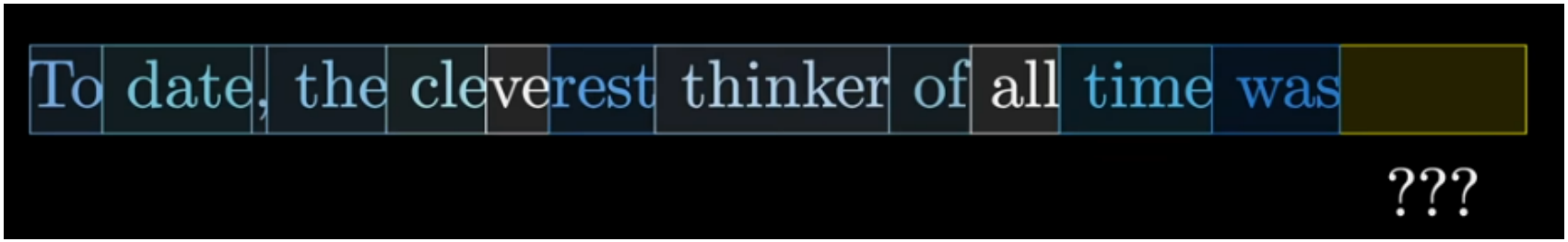
- Specific kind of neural network
- Uses self-attention mechanisms to weigh the relevance of different words in the input when making predictions.
- Suitable for parallel computing (in RNN (LSTM, GRU), the sequence of tokens must be processed sequentially because each ones depends on the previous tokens of the sequence)
- Self attention process allows each token to be computed as a weighted sum of all tokens in the sequence, with the weight determined by the attention mechanism.
- Speed ++

Introduction to Transformers

- The core of ChatGpt is a transformer.
- ChatGPT is based on the GPT version of the transformer. The GPT models are trained to predict the next word in a sequence of words, given all the previous words. They process text from left to right, which makes them well-suited for text generation tasks.
- GPT-3: 175 billion parameters.
- These models are trained using self-supervised learning on a large corpus of text, which means they predict the next word in a sentence (GPT) or a word based on surrounding words (BERT).
- Because they are exposed to a large amount of texts, these models learn grammar, facts about the world, reasoning abilities, and also biases in the data they are trained on.

How does Transformer work ?

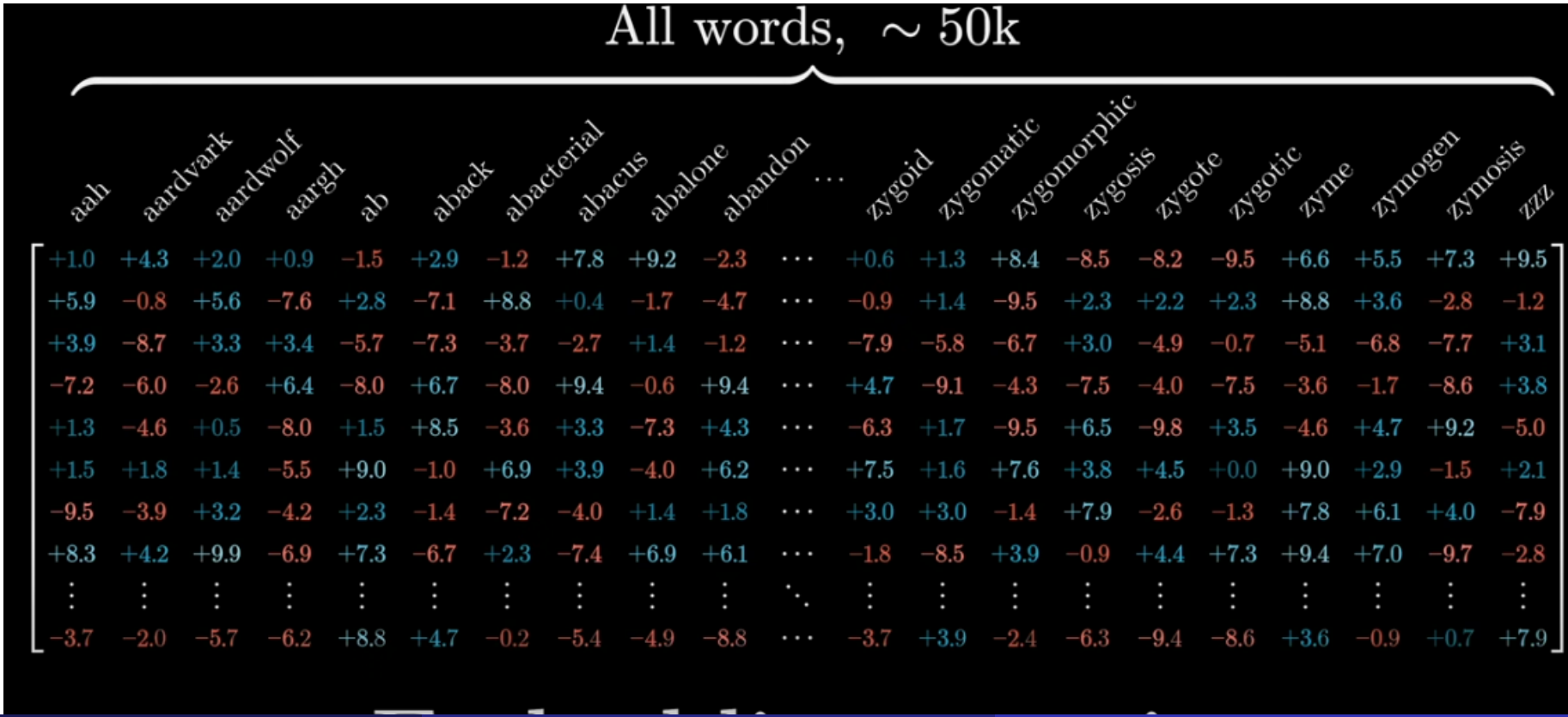
- **First step : Tokenization**



- The input is broken up into little pieces (tokens), words for text, pixcels for picture.

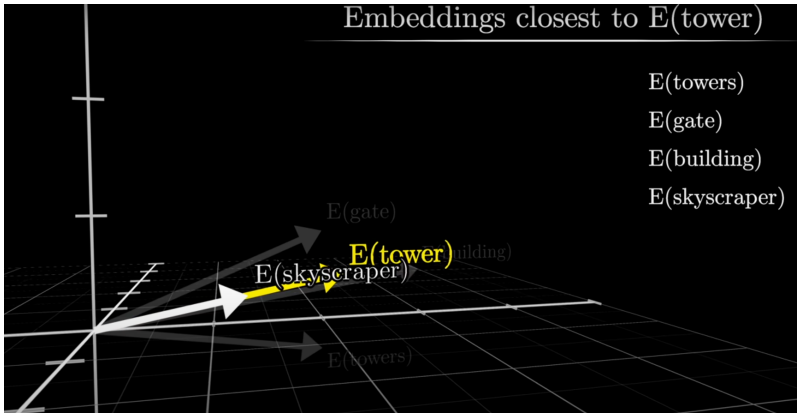
Tokenization

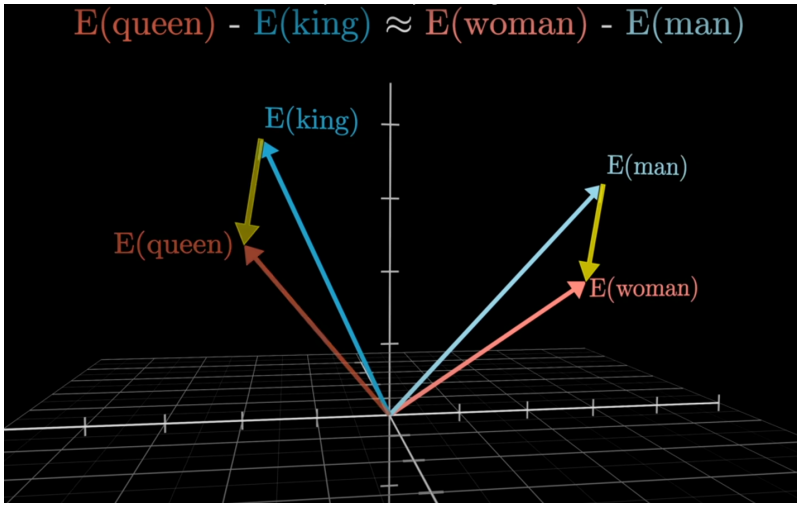
- Embedding matrix has a single columns for each words, wich are represented as vector. Initially, they are set randomly.
- In GPT3, each words have 12,288 coordinates, and total tokens : 50,257 Each words coordinate are called weight. As long the model is trained, the weights are changing, and so the embedding matrix.



Tokenization

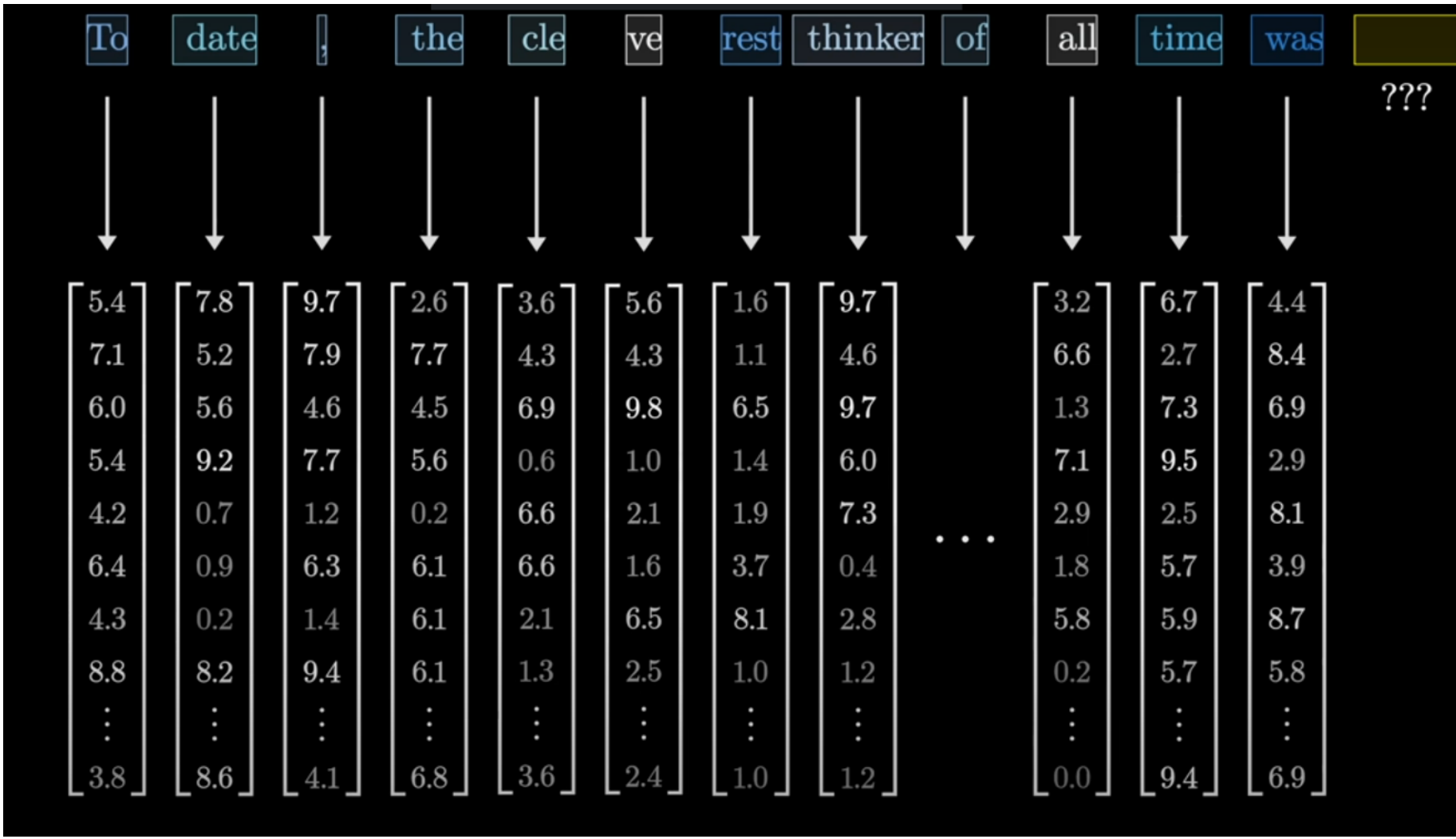
- Words with closest meaning have closest vector





Tokenization

- After several steps of attention, which we will detail next, the sentence is represented by a sequence of vectors.



Tokenization

- To determine the next word, we use the transpose of the embedding matrix, referred to as the unembedding matrix. We compute the matrix product between this matrix and the last vector of the sequence. Applying softmax to the result yields a vector of size 1x50,257 containing the probabilities for each word.

How does Transformer work ?

- The sequence of vectors of the sentence passes through into an operation call attention and who will able the vectors to talk to each other and to pass information to uptade their weights according to the sentence.
- For example : a machin Learning model and a fashion model, model has not the same meaning in both sentences.

How does Attention work ?

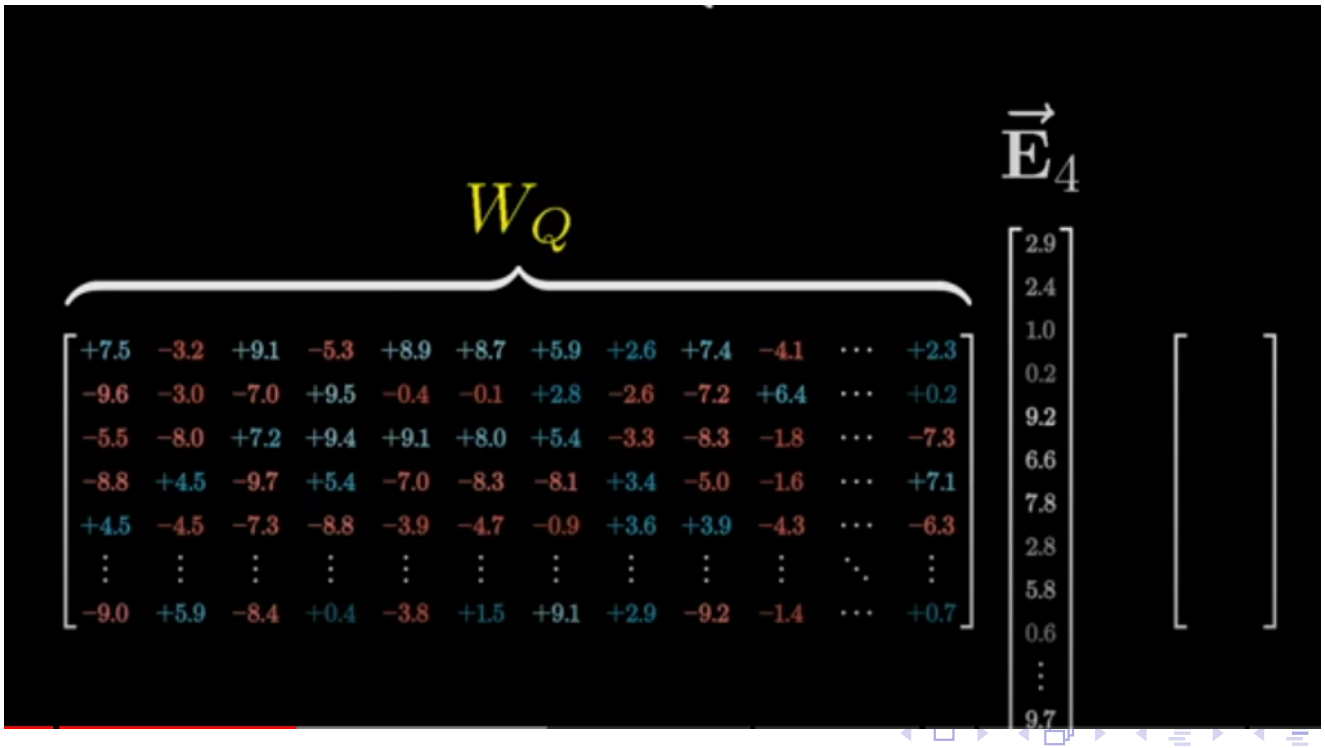
- At the beginning, when a sentence is input, each token is represented by a vector that only corresponds to the word's meaning without context and its position in the sentence. Now, we are going to name each word's vector or embedding $\vec{E}_i$. The goal is to transform each embedding into new ones that inject the meanings from corresponding adjectives, for example.

How does Attention work ?

- Each embedding also has a query vector $\vec{Q}_i$ and a key vector $\vec{K}_i$, which are determined as:

$$\vec{Q}_i = W_q \cdot \vec{E}_i \quad \text{where} \quad W_q \text{ is the query matrix of size } 12,288 \times 128$$

Similarly, the key vector $\vec{K}_i$ is determined using a similar matrix W_k.



How does Attention work ?

- The entries of W_q and W_k are parameters of the model, learned from data.
- Difficult to analyse, somekind encode information into lower dimension space.

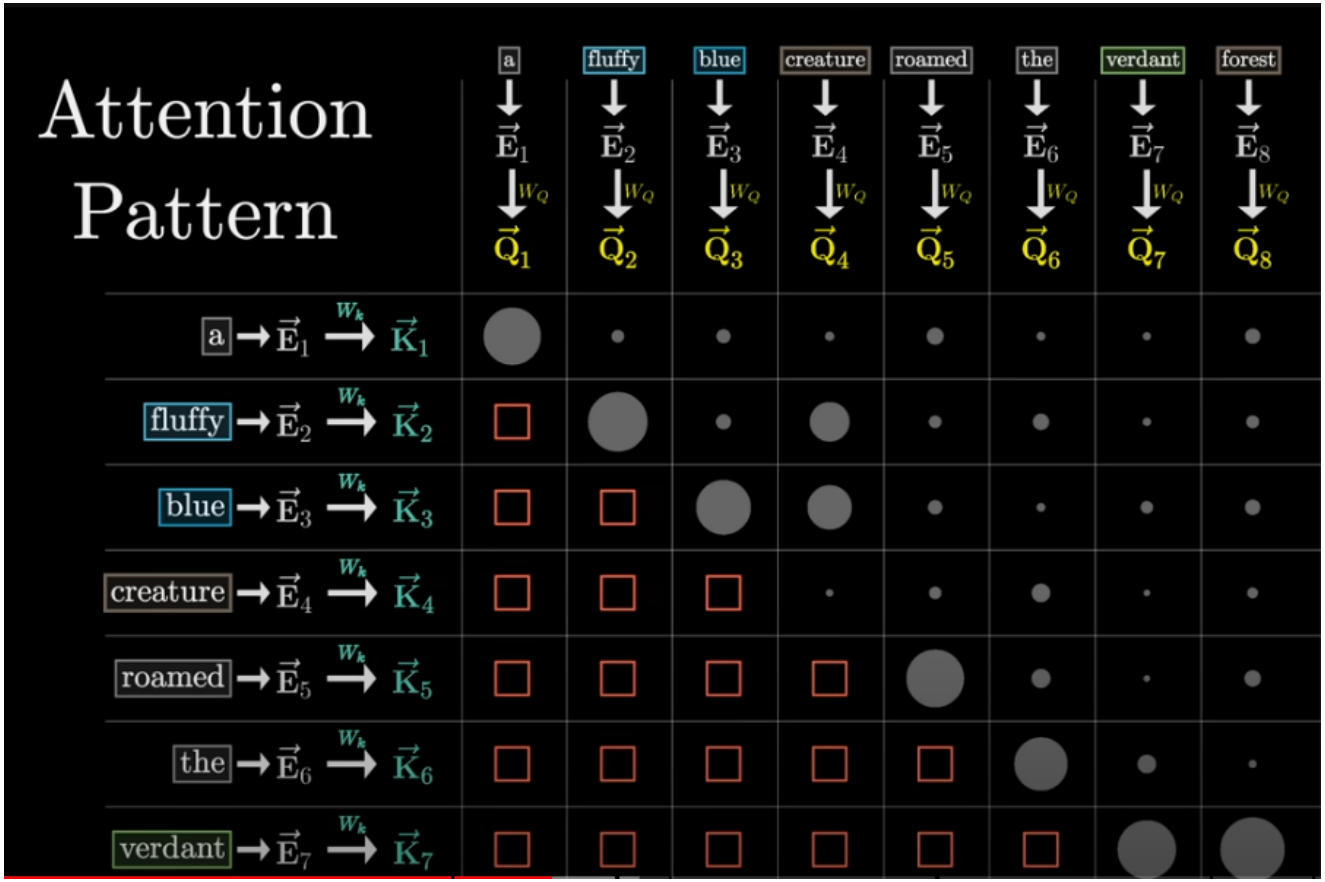
How does Attention work ?

- We can consider that keys are answering queries (for instance $\vec{Q}_i$: "any adjectives in front of me?").
- Keys are matching the queries when $\vec{K}_i$ and $\vec{Q}_j$ are closely aligned, meaning $\vec{E}_i$ has an impact/link with $\vec{E}_j$ (example: adjective).
- To determine if they are matching, we use a dot product: $\vec{K}_i \cdot \vec{Q}_j$ for all i, j.
- If the dot product has a small value, it indicates that they are unrelated.

How does Attention work ?

How does Attention work ?

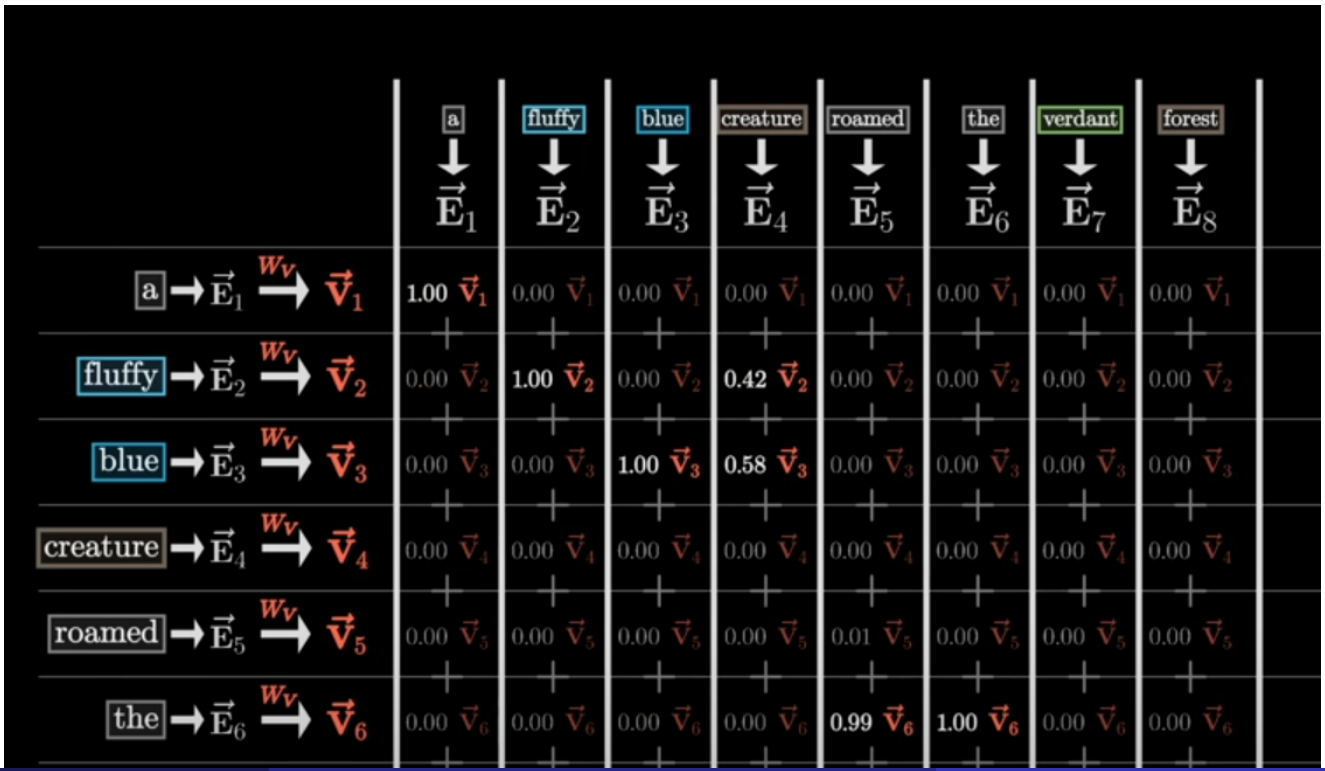
- Attention: We can't allow later words to influence earlier words; otherwise, they can kind of give away the answer of what comes next. So all the values in red become $-\infty$ because with softmax, these elements will become 0 and the column will stay normalized.
- This process is called masking



Value Matrix

- Now, we want to update the embeddings, allowing words to pass information to other words they are relevant to. We use a value matrix W_v that we multiply by the embedding of the first word that is relevant (e.g., adjective). This gives a value vector that we add to the embedding of the second word (e.g., noun).

$$\vec{V}_i = W_v \cdot \vec{E}_i$$



Value Matrix

- and then, $\vec{E}_3 + \Delta\vec{E}_3 = \vec{E}'_3$: a more refined vector encoding a richer contextual meaning. And we have $\vec{E}'_i$ for all i.

Multi head attention

- This is only one head attention. For instance, one head can focus on adjectives, grammar, verbs, etc.
- Multi-head attention is when you run many of these operations in parallel, each with its own key, query, and value matrices: $W_q(i)$, $W_v(i)$, $W_k(i)$.
- GPT-3 uses 96 attention heads.
- This gives the model the capacity to learn many distinct ways that context changes the meaning.

Back to the first draw

Core of the Transformer is Self Attention as in the original paper: "**Attention** is all you need".

arxiv.org/pdf/1706.03762.pdf

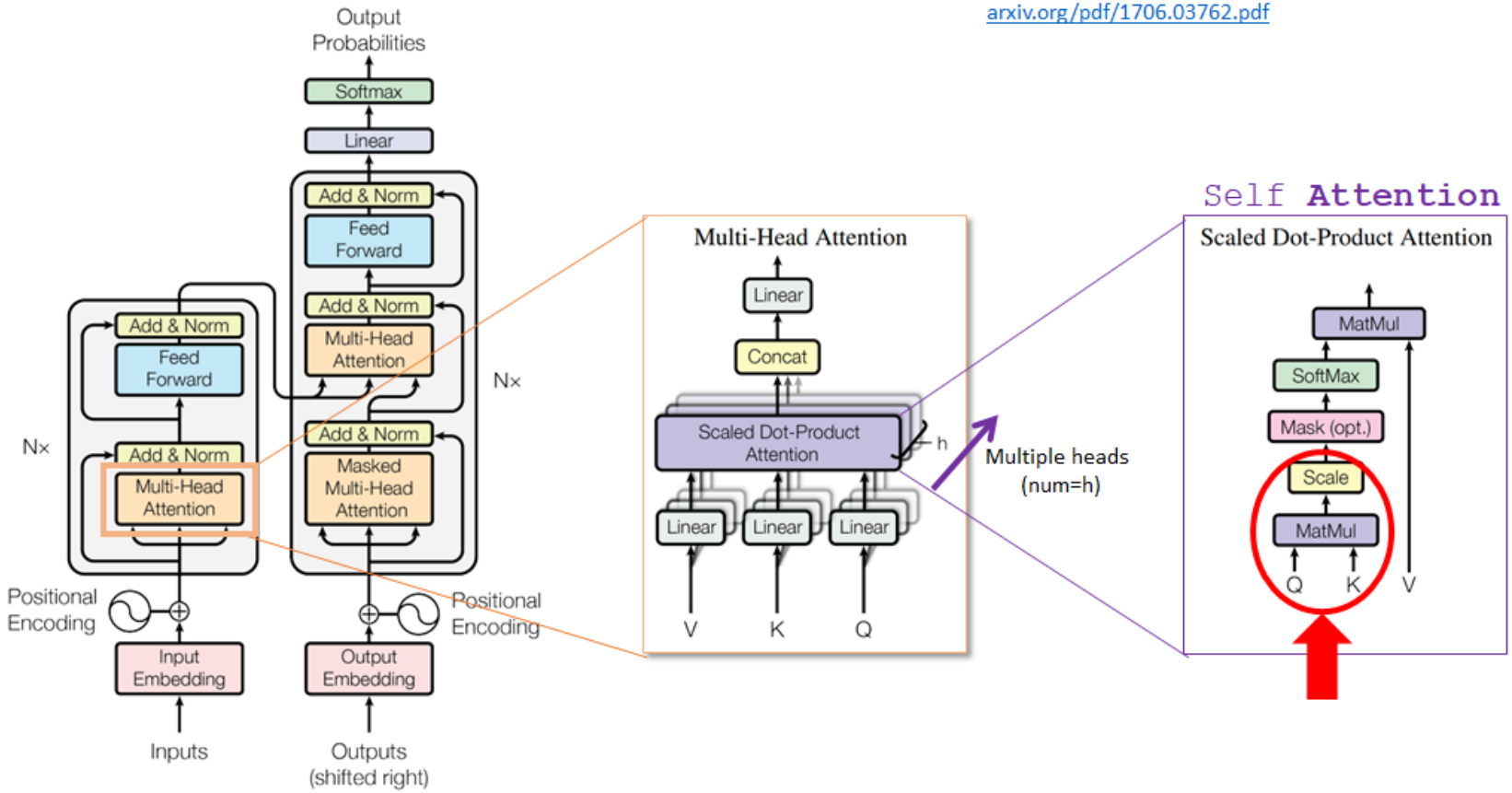
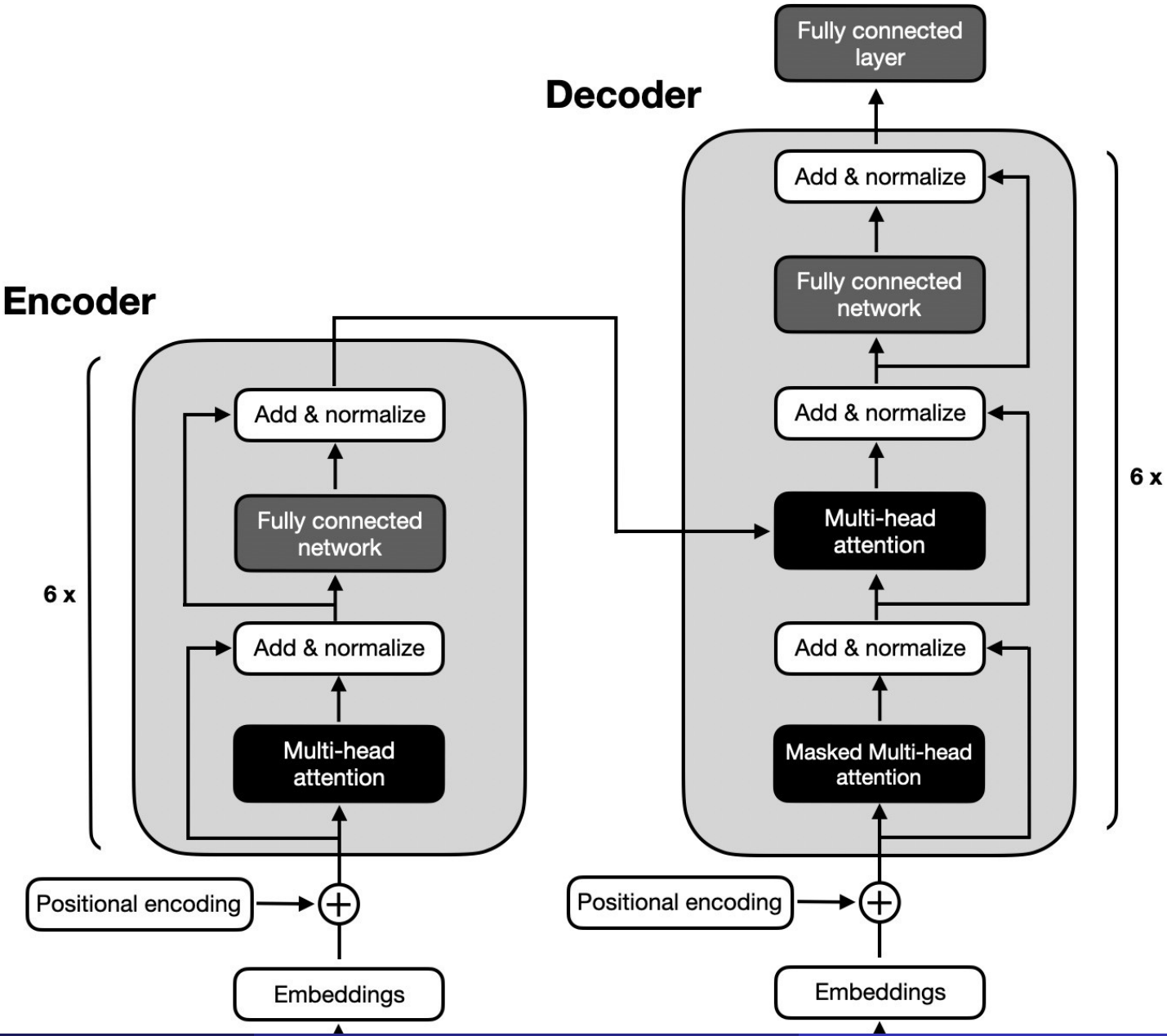


Figure 1: The Transformer - model architecture.

Encoder/ Decoder



Encoder/Decoder

- **Encoder:** Responsible for understanding and extracting the relevant information from the input text. It then outputs a continuous representation (embedding) of the input text that is passed to the decoder.
  - Good for classification, sequence tagging, sentiment analysis
  - Exemple : Bert...
- The masked language (masking random word tokens in the input sequence and then training the model to predict the original masked tokens based on the surrounding context) and next sentence pretraining objectives allow BERT to learn rich contextual representations of the input texts.
- Typically need to be fine-tuned for various downstream tasks like sentiment analysis or specific tasks
- Cannot generate text (only understand text)

Encoder/Decoder

- **Encoder/Decoder:** Use encoder to understand and use decoder to generate output sequence
  - Transformer
  - Good for translation and summarization

Encoder/Decoder

- **The decoder** generates output word by word, and the predictions for position i can depend only on the known outputs at positions less than i.
  - Language Models
  - GPT2,3,4, ChatGPT
  - Phi3, Llama3
- Controlled by prompting (instruction)
- Few-shots $>$ Zero- shots (shots=examples)