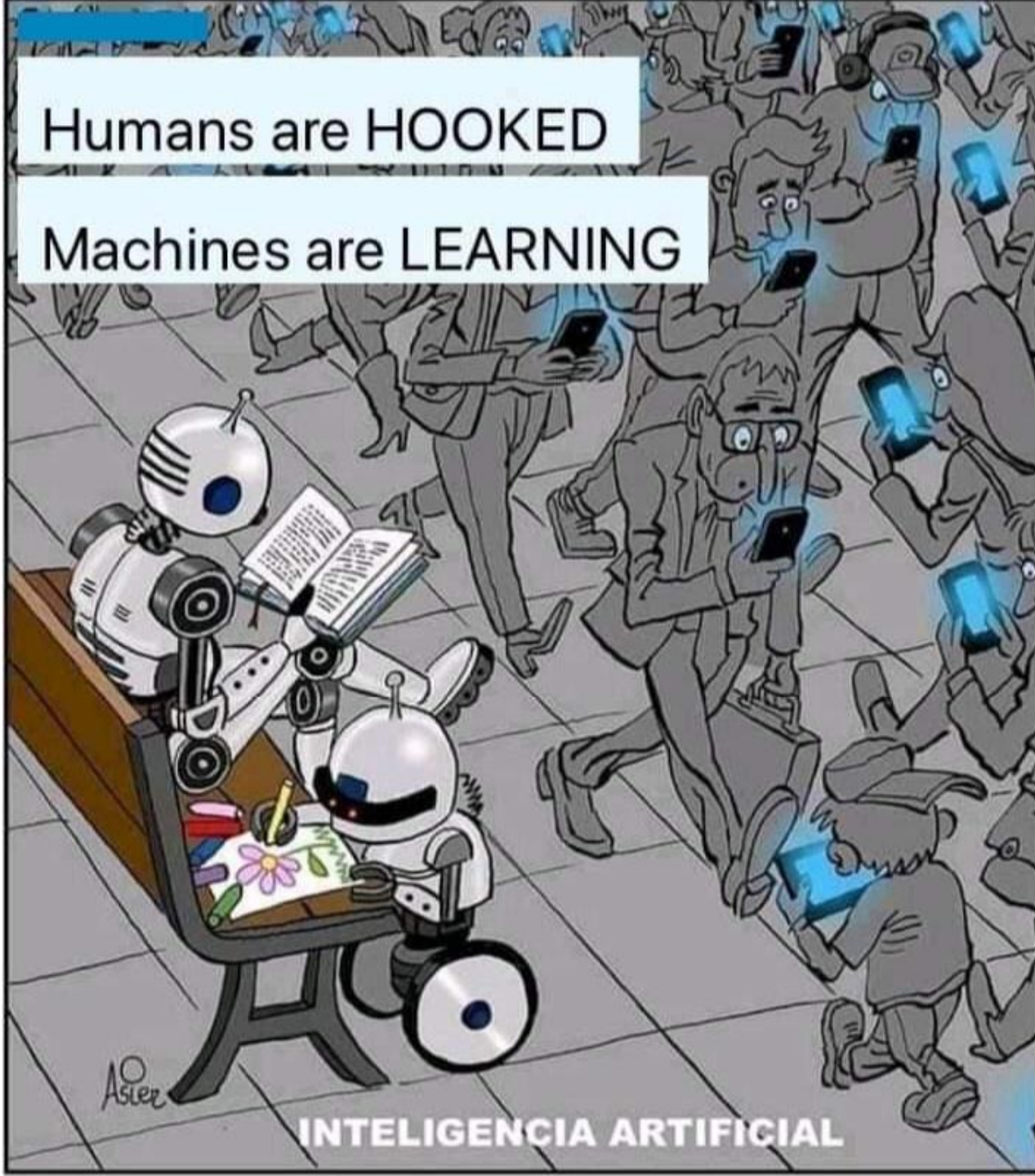


Humans are HOOKED

Machines are LEARNING




Generative AI - LLMs



Naveen Kumar Bhansali
Co-Founder BlitzAI | Adjunct Faculty @
IIM Bangalore





LLMs – What
are they?

Let us ask ChatGPT



Master of Laws

Default (GPT-3.5)

SU

what is llm



The acronym "LLM" can have several meanings depending on the context. Some common interpretations include:



1. **Master of Laws (LL.M):** This is an advanced postgraduate degree in law, typically taken by individuals who already have a Juris Doctor (JD) or a law degree from their home country. An LL.M. program allows lawyers to specialize in a particular area of law or study the legal systems of other countries.
2. **Low-Level Memory:** In the context of computer science and programming, "LLM" might refer to low-level memory management, which involves directly managing computer memory at a low level. This can include tasks like allocating and deallocating memory, managing pointers, and working with data structures.
3. **Limited Liability Company (LLC):** In a business and legal context, "LLM" might also refer to Limited Liability Company. An LLC is a type of business structure that combines the limited liability protection of a corporation with the simplicity and flexibility of a partnership.
4. **London Leasing Market:** In the context of finance and real estate, "LLM" can stand for London Leasing Market, which is a marketplace for leasing commercial property in London.
5. **Land and Leases Management:** In the context of real estate and property management, "LLM" can stand for Land and Leases Management, which involves the management of land and lease agreements.



Large Language Models (LLMs)



LLMs

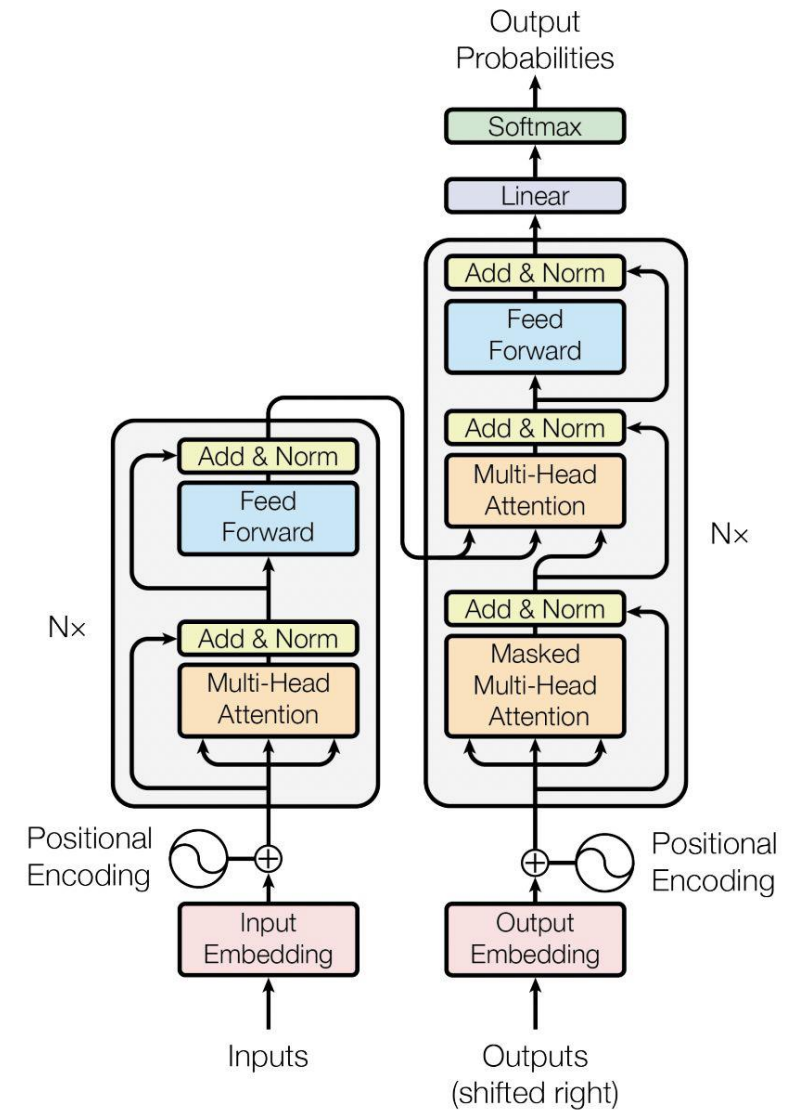
- A Language Model
- Consisting of Neural network with many parameters
 - (typically, billions to trillions of weights or more)
- Trained on large quantities (trillions of words) of unlabelled text.
 - Wikipedia, GitHub, Common Crawl, The Pile etc.
- Using self-supervised learning

LLMs – Based on?

Attention is all you need

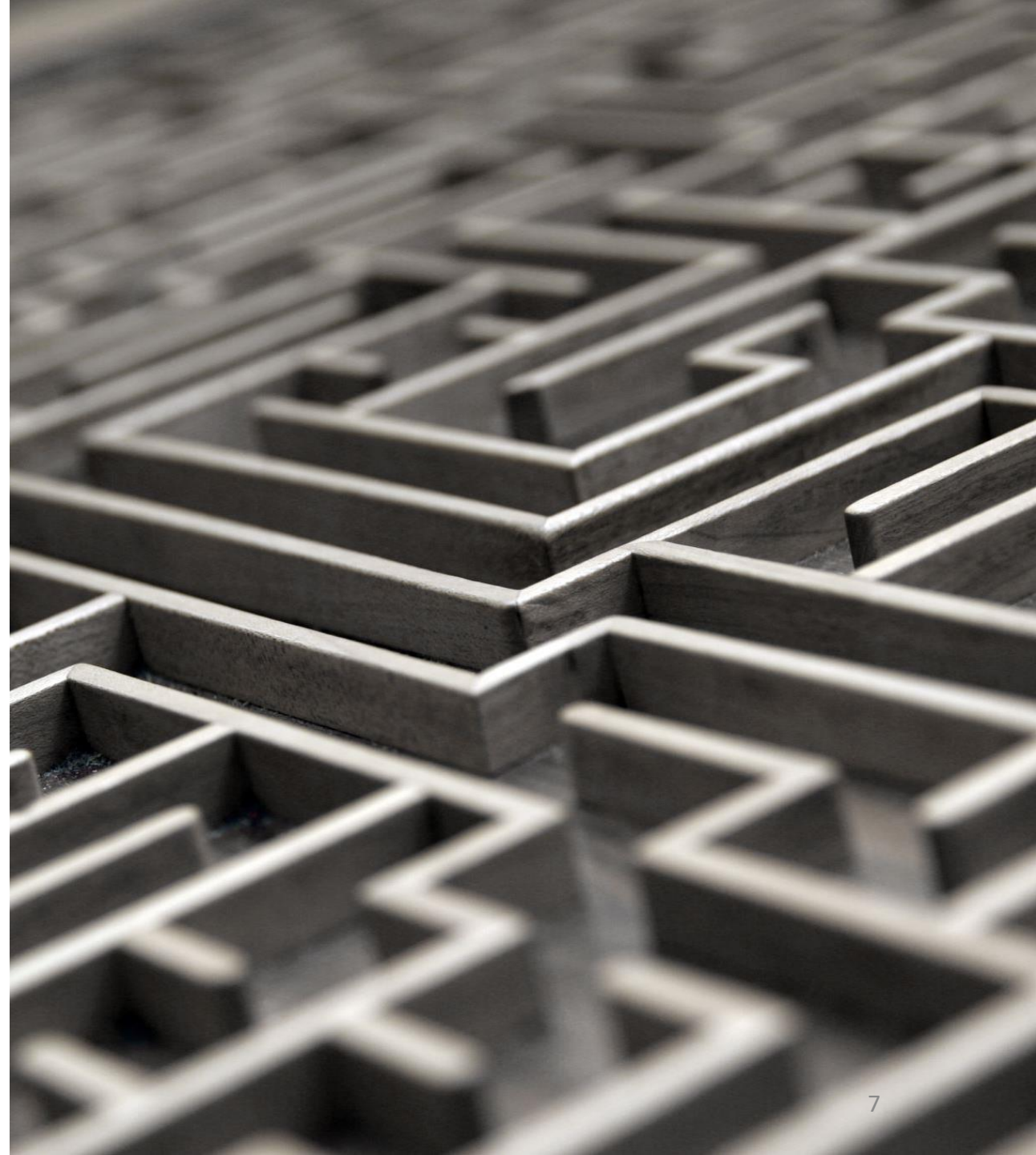
<https://ig.ft.com/generative-ai/>

Transformers architecture



LLMs – Characteristics


- General purpose models
 - Excel at a wide range of tasks as opposed to being trained for one specific task (such as **sentiment analysis, named entity recognition or mathematical reasoning etc**).
- Stateless
- Stochastic



Mode

 Chat 

Model

gpt-3.5-turbo 

Temperature 1



Maximum length 256



Stop sequences

Enter sequence and press Tab

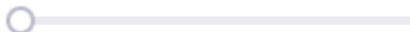
Top P 1



Frequency penalty 0



Presence penalty 0



OpenAI Playground

<https://platform.openai.com/playground>

Parameters





Parameters – Maximum Length

- Maximum length (max_new_tokens):
 - Limits the number of tokens in the generated output.
 - **Tokens** are units of text, which can be as short as one character or as long as one word.
 - For example, the string "ChatGPT is great!" is encoded into six tokens: ["Chat", "G", "PT", " is", " great", "!"].
 - 1 token \sim $\frac{3}{4}$ **words**. 100 tokens \sim 75 words.

Parameter – Top-k Vs. Top-p sampling

Let us say the output of the final layer is as follows:

| Token | Token_Prob |
|-----------|------------|
| Apple | 0.25 |
| Mango | 0.15 |
| Orange | 0.05 |
| Hello | 0.005 |
| Bangalore | 0.001 |

Which word would be selected, if we take

- Greedy Approach?
- Random (-weighted) sampling?

Which approach would you prefer?

Parameter – Top-k Vs. Top-p sampling

Let us say the output of the final layer is as follows:

| Token | Token_Prob |
|-----------|------------|
| Apple | 0.25 |
| Mango | 0.16 |
| Orange | 0.05 |
| Hello | 0.005 |
| Bangalore | 0.001 |

Top-k sampling: select an output from the top-k results after applying random-weighted strategy using the probabilities.

Which word would be selected if $k=3$?

Parameter – Top-k Vs. Top-p sampling

Let us say the output of the final layer is as follows:

| Token | Token_Prob |
|-----------|------------|
| Apple | 0.25 |
| Mango | 0.16 |
| Orange | 0.05 |
| Hello | 0.005 |
| Bangalore | 0.001 |

- **Top-p sampling:** Select an output using the random-weighted strategy with the top-ranked consecutive results by probability and the smallest subset of tokens whose cumulative probability exceeds the threshold p .
- Which word would be selected if $p \geq 0.4$?

Parameter – Temperature

Let us say the output of the final layer is as follows:

Cooler Temperature (< 1)

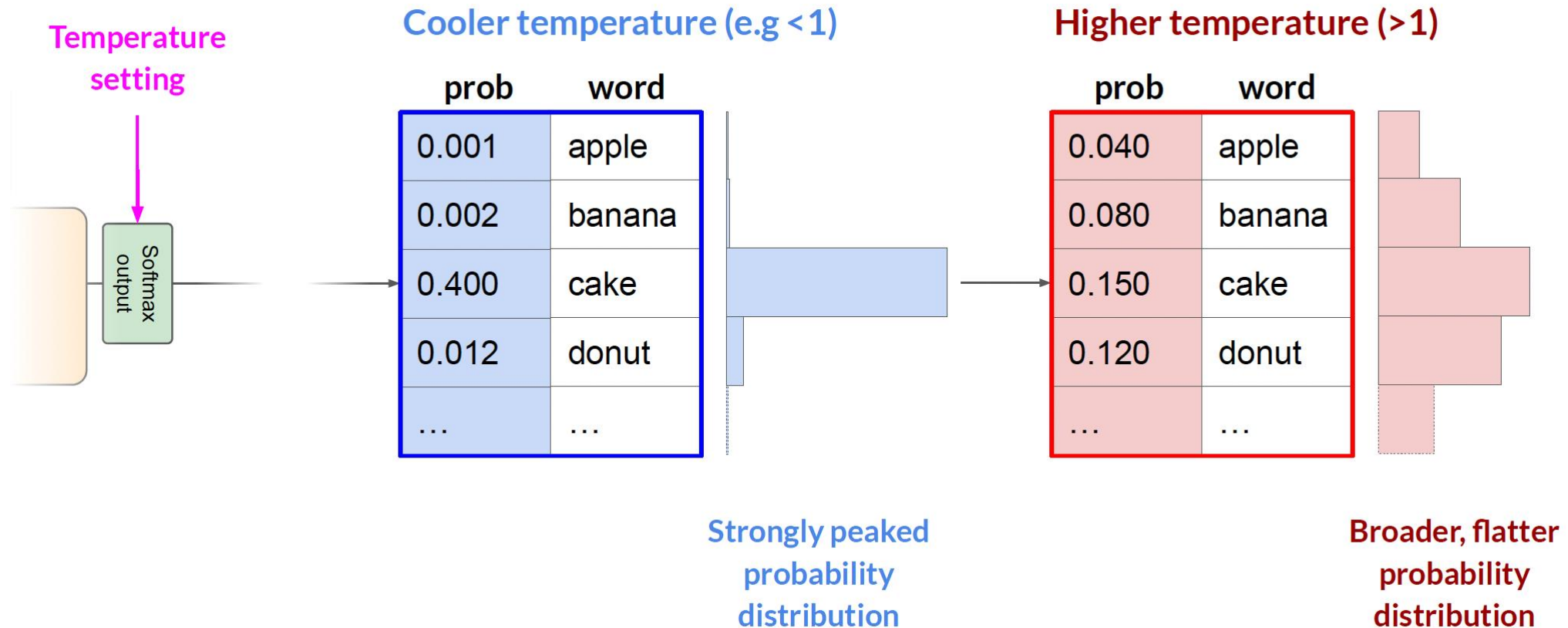
| Token | Token_Prob |
|-----------|------------|
| Orange | 0.01 |
| Mango | 0.1 |
| Apple | 0.54 |
| Juice | 0.005 |
| Bangalore | 0.001 |

Higher Temperature (>1)

| Token | Token_Prob |
|-----------|------------|
| Orange | 0.19 |
| Mango | 0.26 |
| Apple | 0.30 |
| Juice | 0.10 |
| Bangalore | 0.001 |

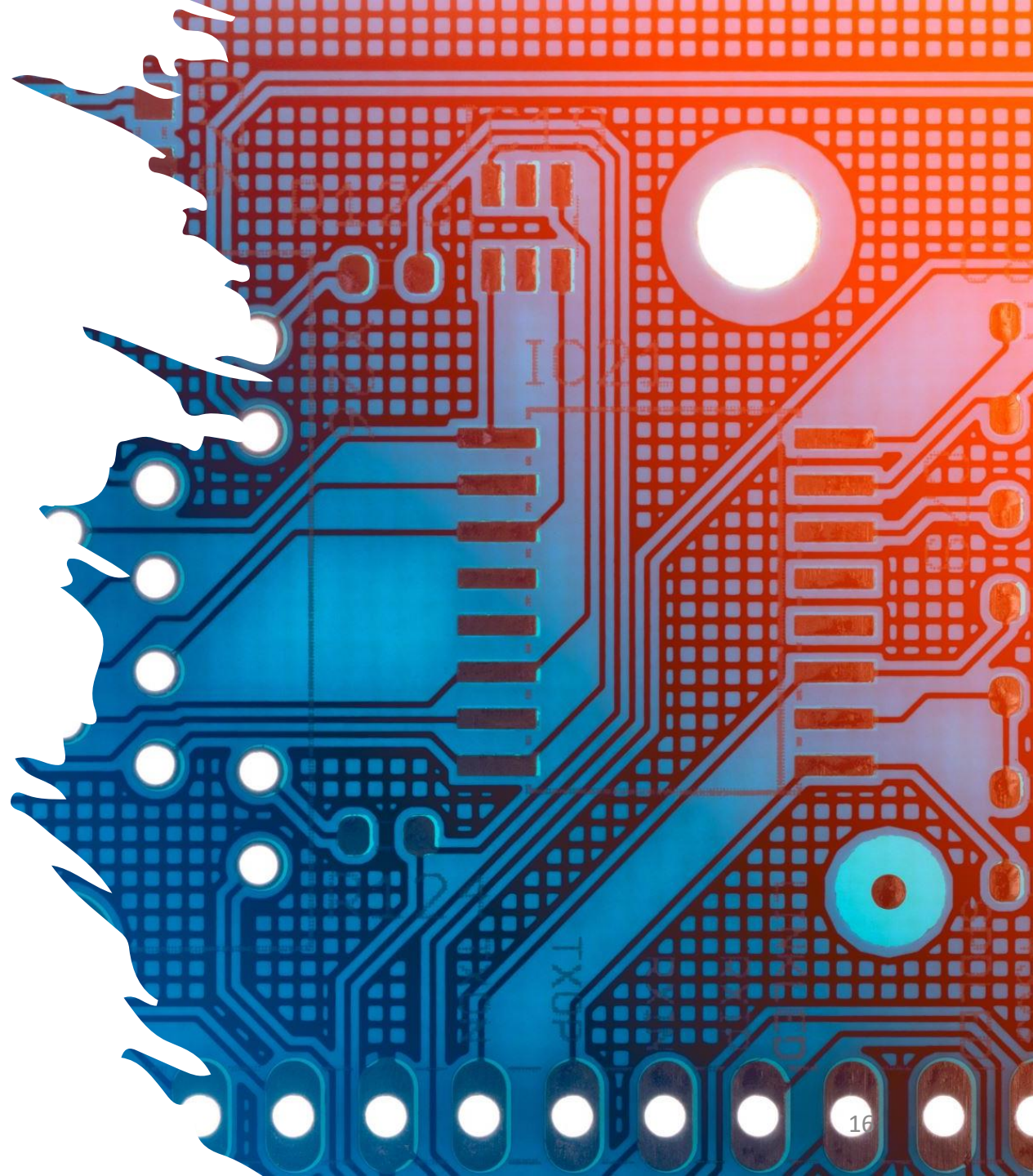
What do you observe?

Generative Config - Temperature



LLMs Settings: Temperature

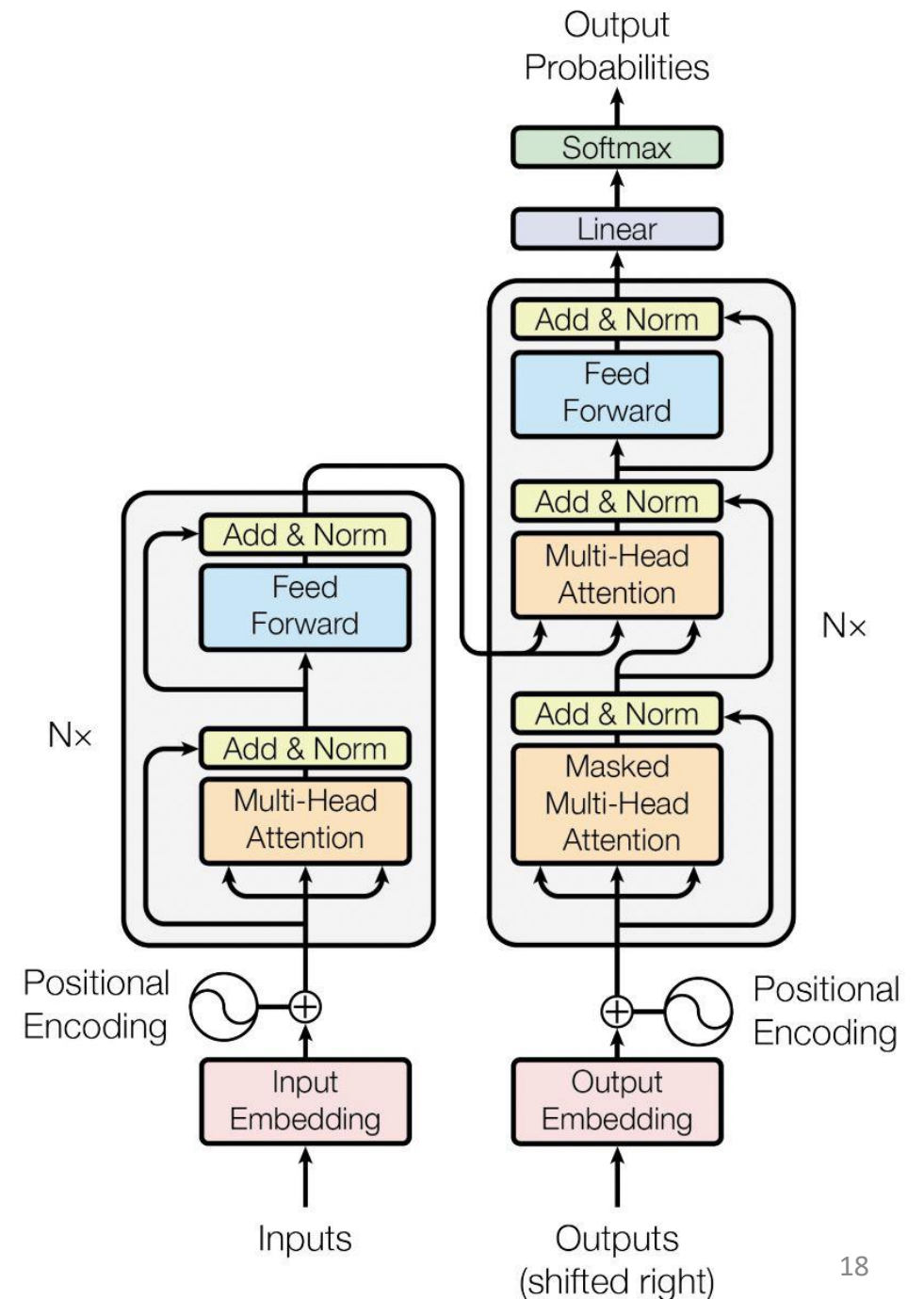
- The lower the temperature, the more deterministic the results in the sense that the highest probable next token is always picked.
 - Tasks like **fact-based QA** to encourage more factual and concise responses
- Increasing temperature could lead to more randomness, which encourages more diverse or creative outputs. You are essentially increasing the weights of the other possible tokens.
 - For **poem generation or other creative tasks**.
- Select a temperature value based on the desired trade-off between coherence and creativity for your specific application.



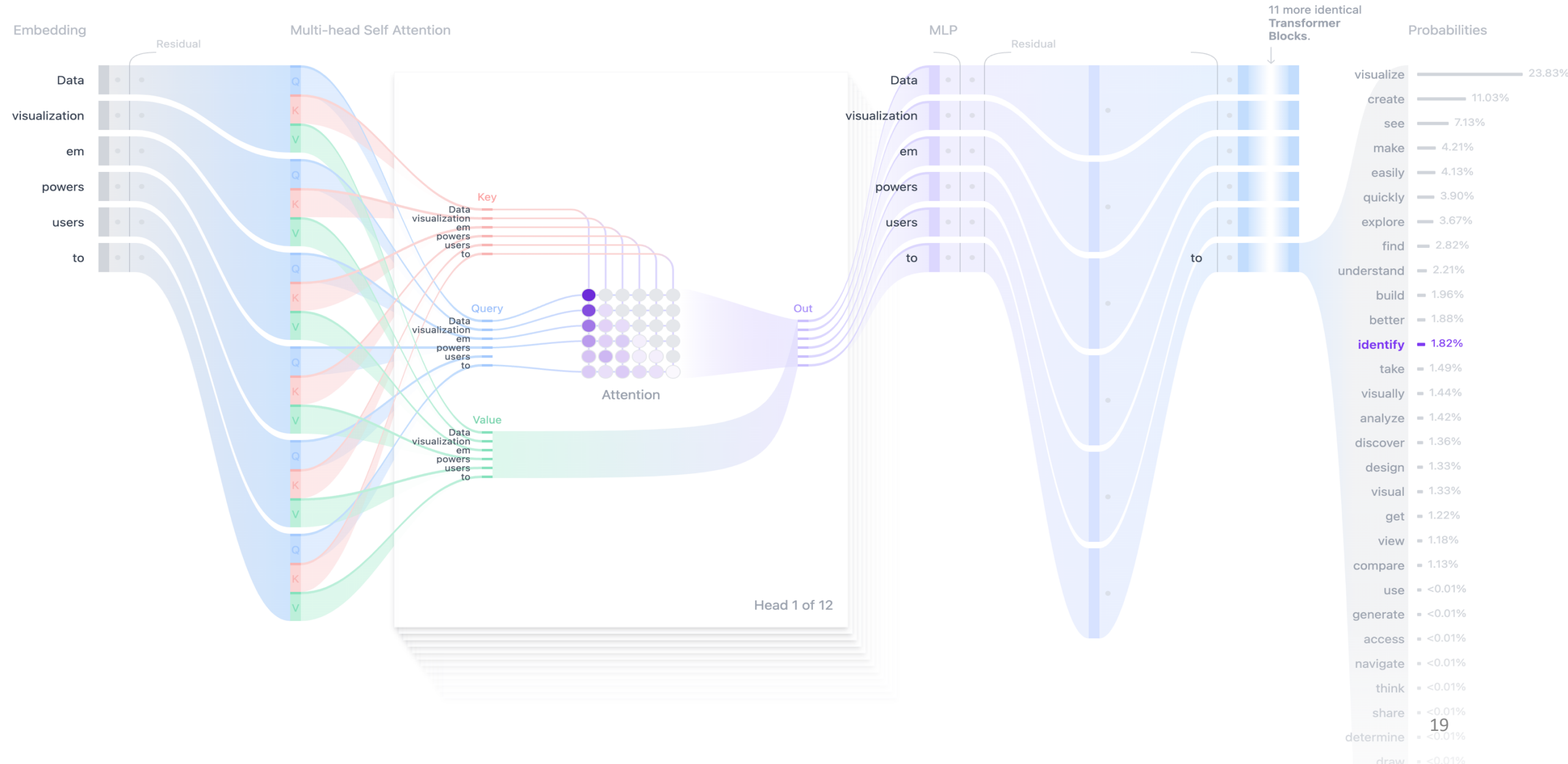
Appendix

Transformer Architecture

- Every text-generative Transformer consists of these three key components:
 - Embedding
 - Transformer Block
 - Output Probabilities



GPT-2 Small: Decoder-only model



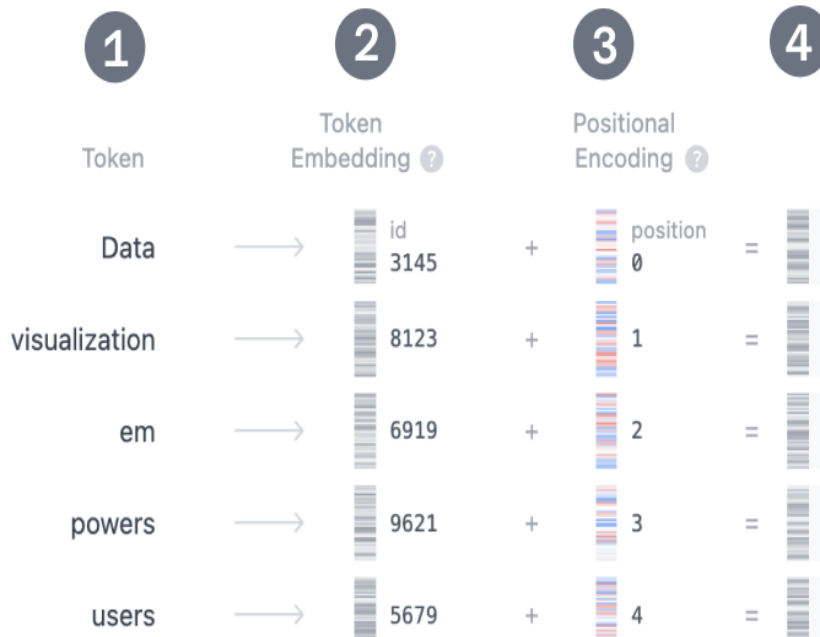


Embedding

- Text input is divided into smaller units called tokens, which can be words or sub-words.
- These tokens are converted into numerical vectors called embeddings, which capture the **semantic meaning** of words.

Embedding

Prompt: Data visualization empowers users to ...



Step 1: Tokenization

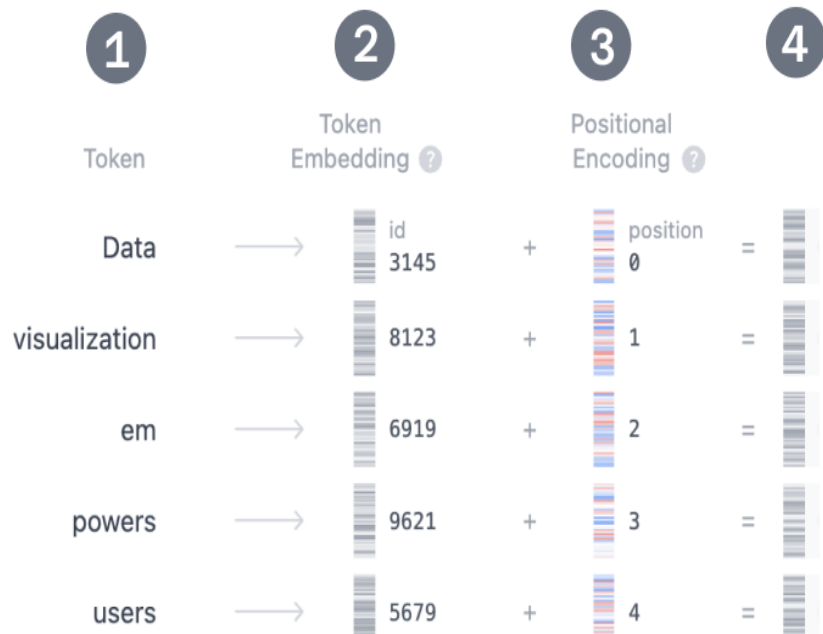
- Tokenization is the process of breaking down the input text into smaller, more manageable pieces called **tokens**. These tokens can be a word or a sub-word.
- The words "Data" and "visualization" correspond to unique tokens, while the word "empowers" is split into two tokens.
- The full vocabulary of tokens is decided before training the model: GPT-2's vocabulary has 50,257 unique tokens.

Step 2. Token Embedding

- GPT-2 Small (124 million parameters) represents each token in the vocabulary as a 768-dimensional vector; the dimension of the vector depends on the model.
- These embedding are stored in a matrix of shape $(50257, 768)$ i.e., 39 Million parameters.

Embedding

Prompt: Data visualization empowers users to . . .



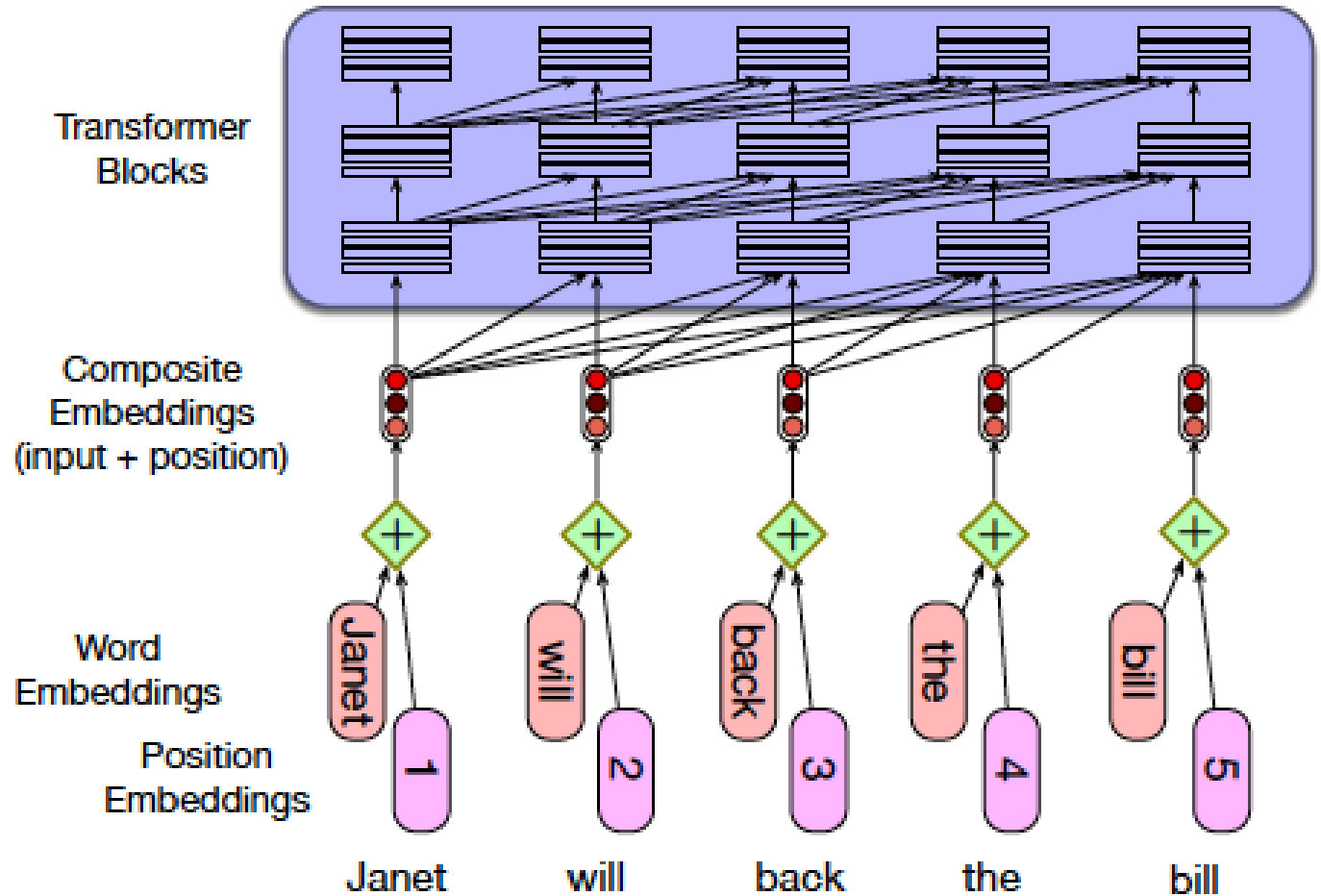
Step 3. Positional Encoding

- The Embedding layer also encodes information about each token's position in the input prompt.
- GPT-2 trains its own positional encoding matrix from scratch, integrating it directly into the training process.
- **Why is position of the word so important?**
 - The dog looked at the boy and ... (barked?)
 - The boy looked at the dog and ... (smiled?)

Step 4. Final Embedding

- Finally, we sum the token and positional encodings to get the final embedding representation.
- This combined representation captures both the semantic meaning of the tokens and their position in the input sequence.

Positional Embedding

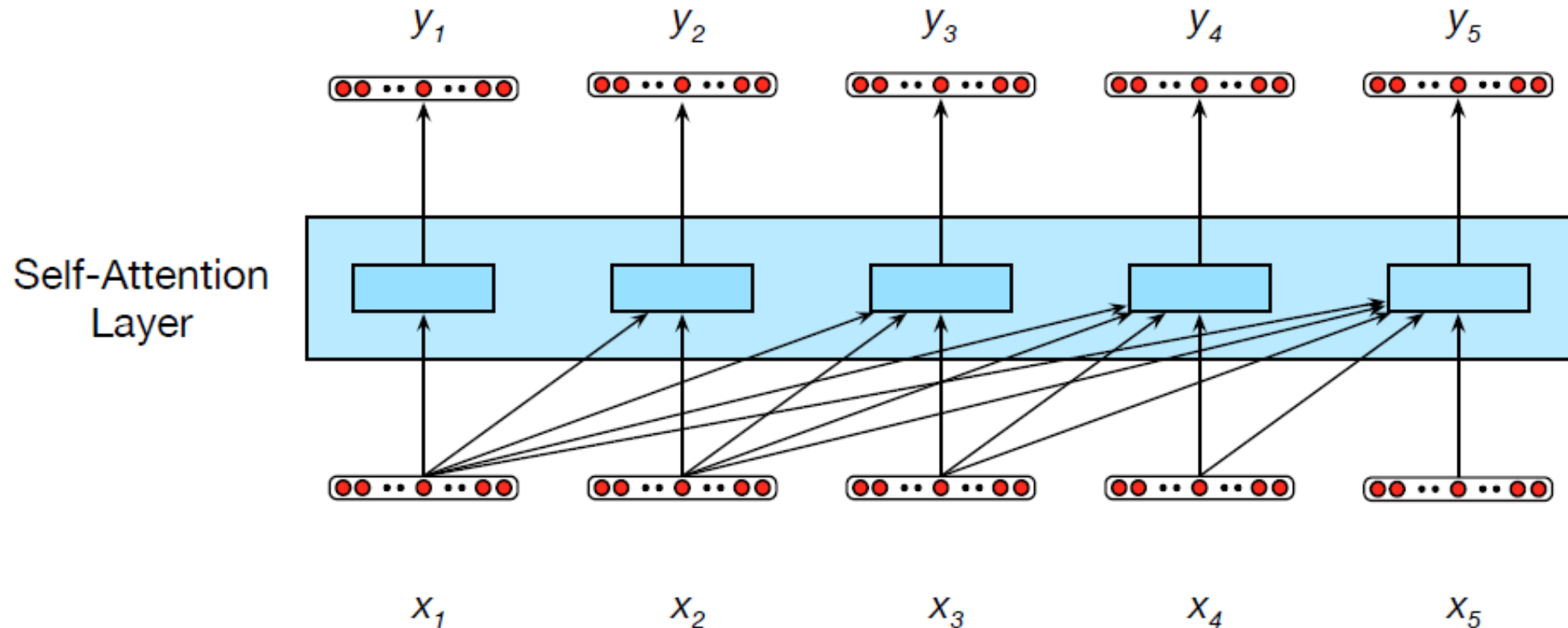


Transformer Block

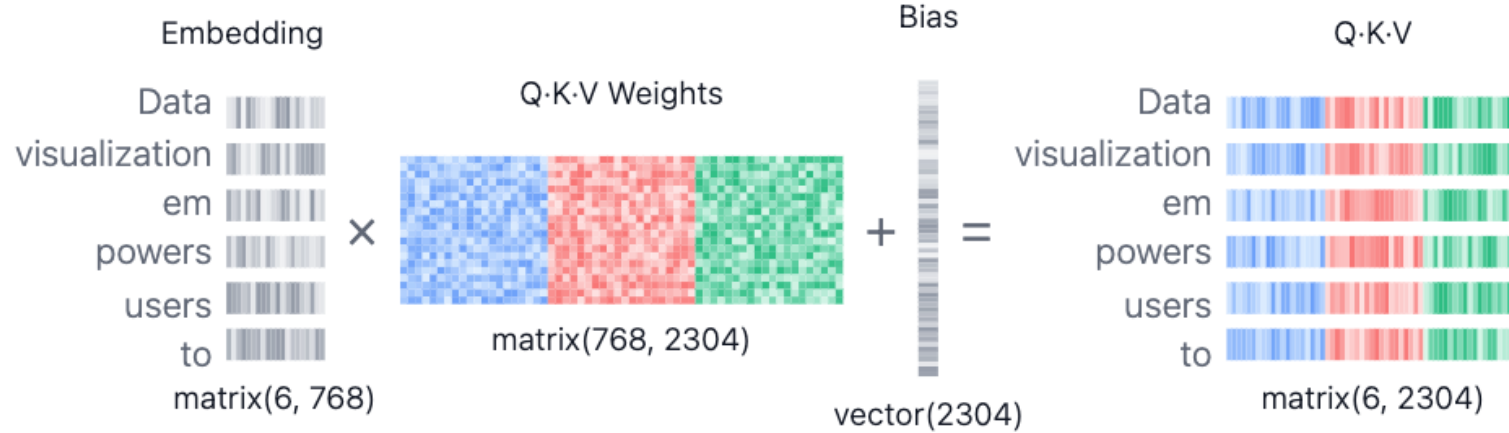
- **Transformer Block** is the fundamental building block of the model that processes and transforms the input data.
- **Each block includes:**
 - **Attention Mechanism**, the core component of the Transformer block.
 - It allows tokens to communicate with other tokens, capturing contextual information and relationships between words.
 - **MLP (Multilayer Perceptron) Layer**, a feed-forward network that operates on each token independently.
 - While the goal of the attention layer is to route information between tokens, the goal of the MLP is to refine each token's representation.

Transformer Block: Self-Attention Layer

Unlike RNNs (Recurrent Neural Networks), the computations at each time step are independent of all the other steps and therefore can be performed in parallel.



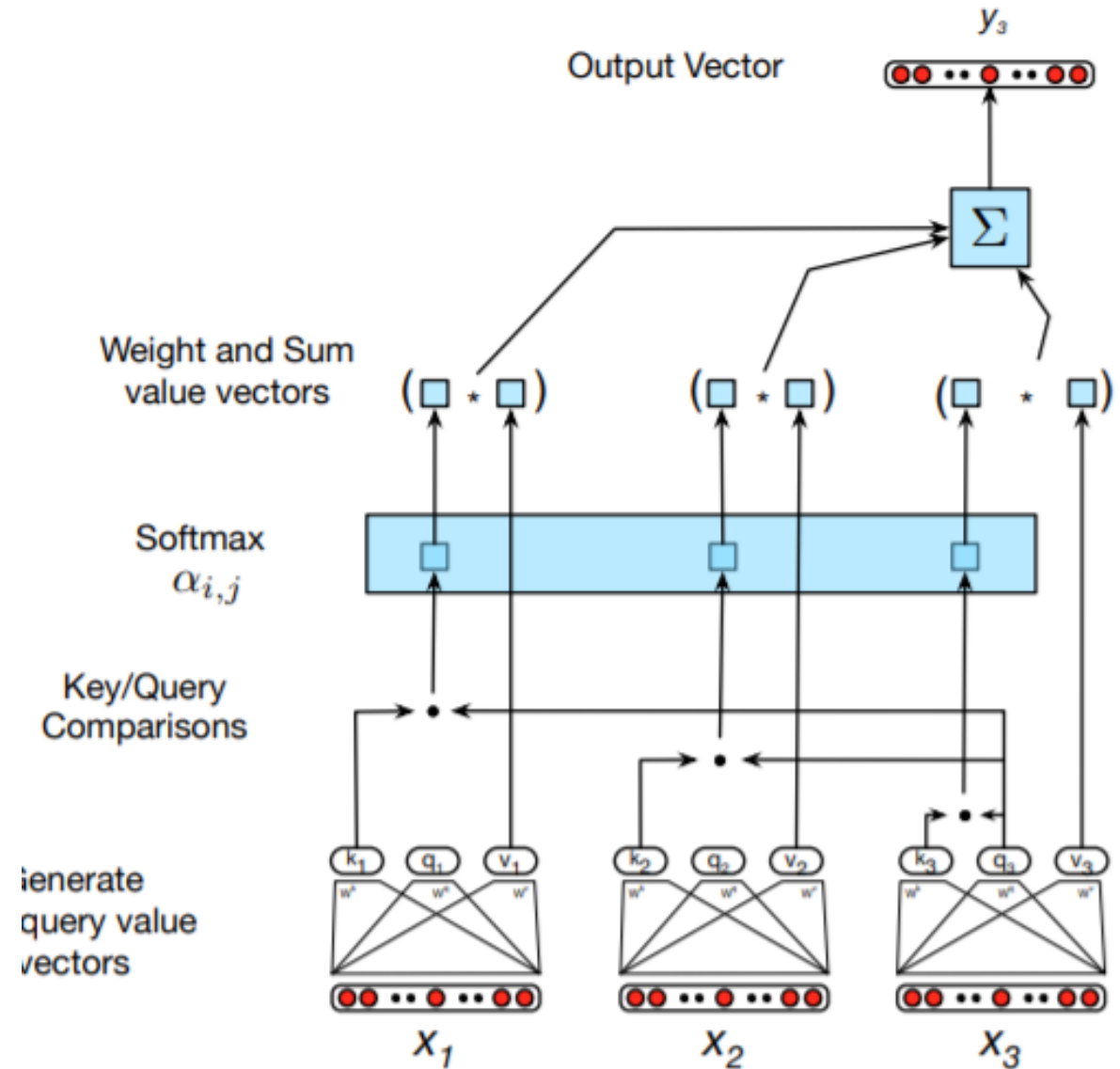
Transformer Block: Self-Attention



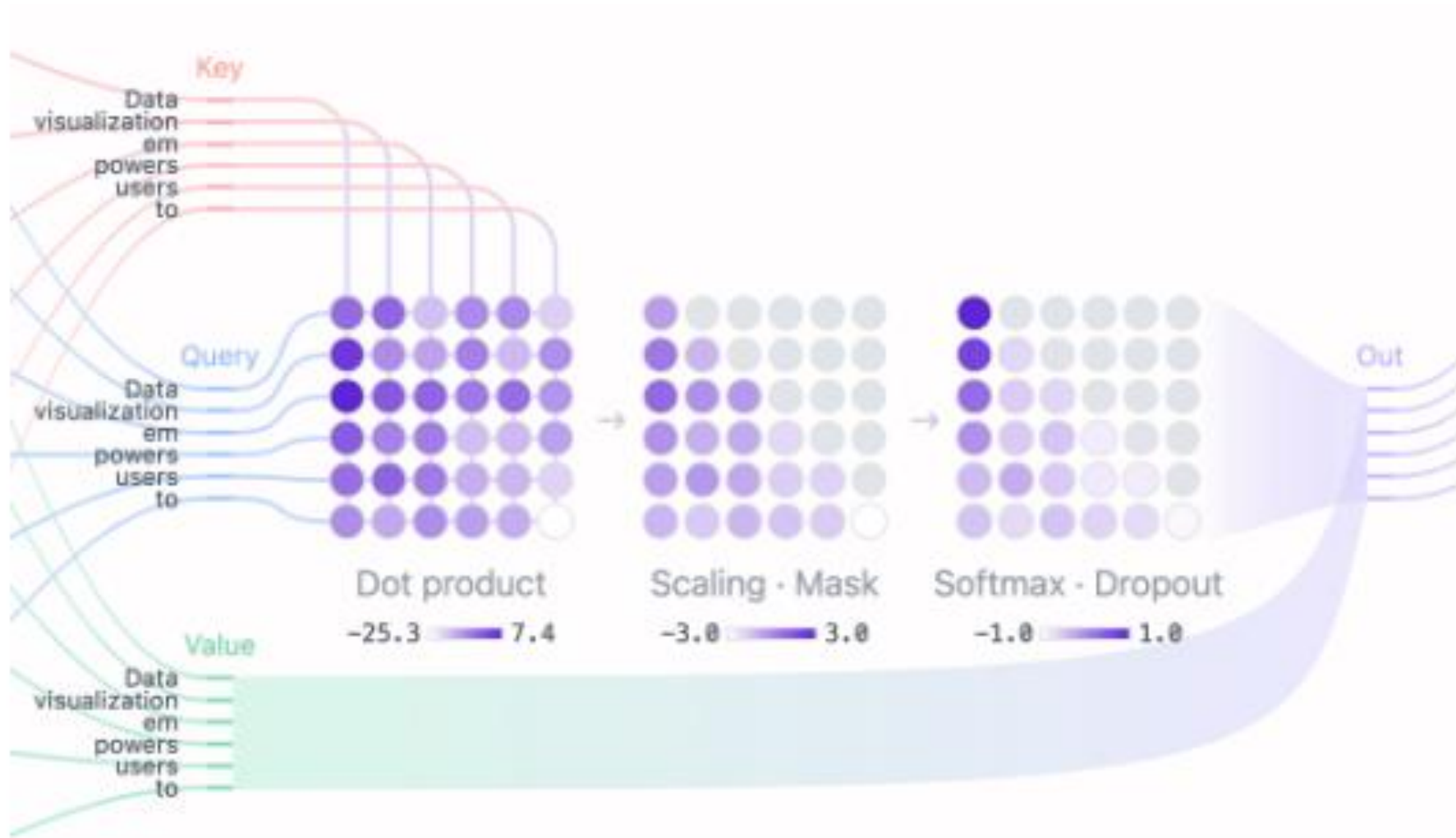
$$\sum_{d=1}^{768} E_{id} \cdot W_{dj} + b_j = QKV_{ij}$$

- Each token's embedding vector is transformed into three vectors: **Query (Q)**, **Key (K)**, and **Value (V)**. These vectors are derived by multiplying the input embedding matrix with learned weight matrices for **Q**, **K**, and **V**.
- Here's a **web search analogy** to help us build some intuition behind these matrices:
 - **Query (Q)** is the search text you type in the search engine bar. This is the token you want to "*find more information about*".
 - **Key (K)** is the title of each web page in the search result window. It represents the possible tokens the query can attend to.
 - **Value (V)** is the actual content of web pages shown. Once we matched the appropriate search term (Query) with the relevant results (Key), we want to get the content (Value) of the most relevant pages.
- By using these QKV values, the model can calculate attention scores, which determine how much focus each token should receive when generating predictions.

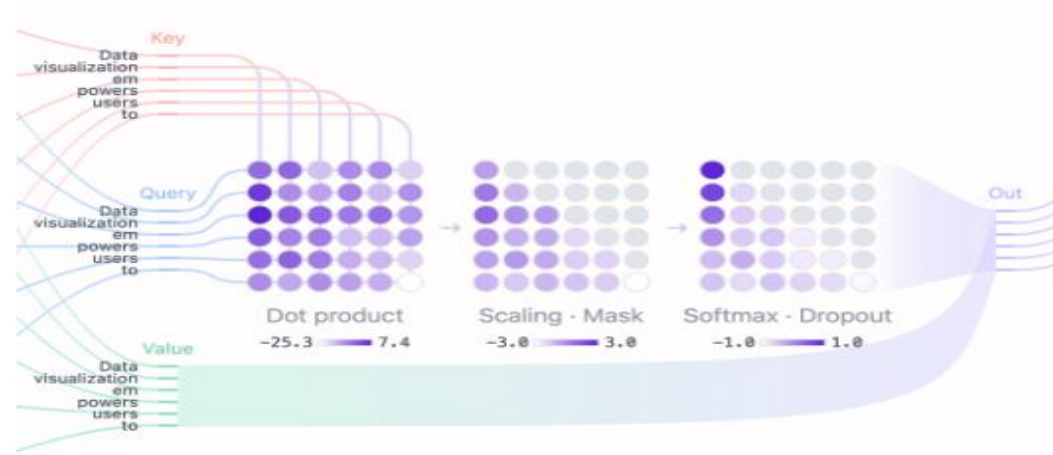
Transformer Block: Self-Attention



Transformer Block: Masked Self-Attention



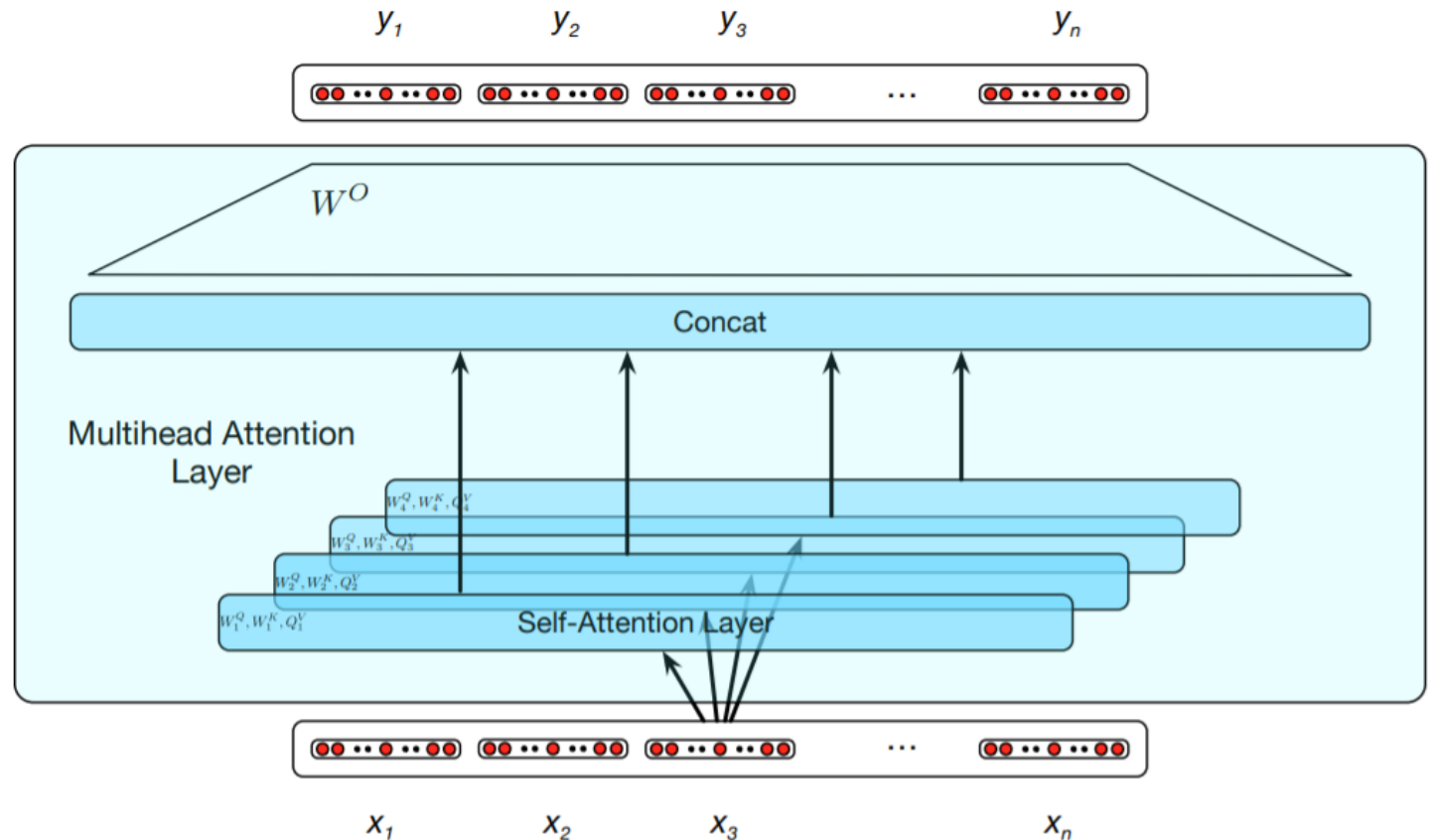
Transformer Block: Masked Self-Attention



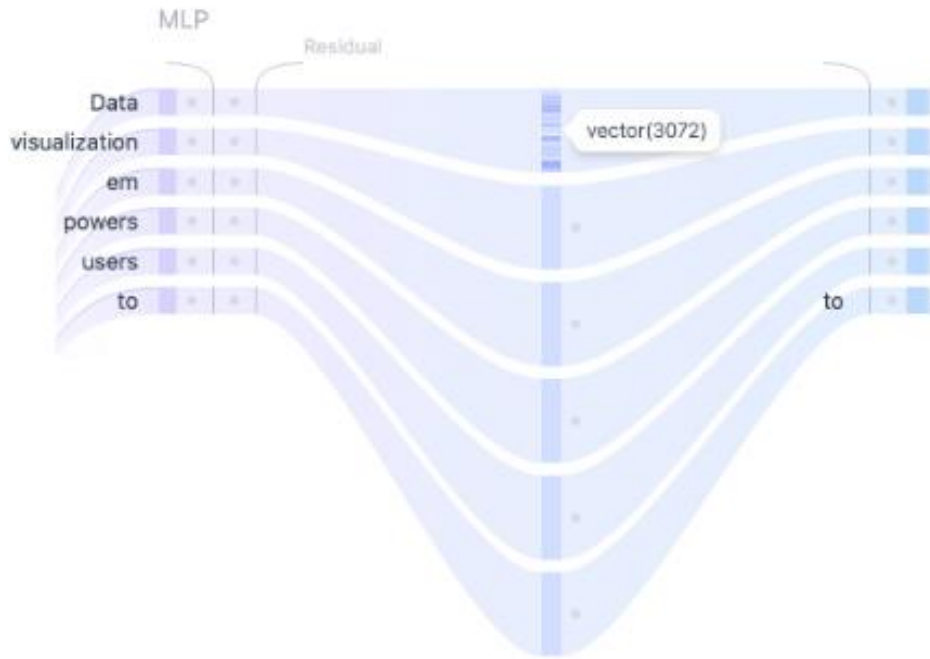
- **Attention Score:** The dot product of Query and Key matrices determines the alignment of each query with each key, producing a square matrix that reflects the relationship between all input tokens.
- **Masking:** A mask is applied to the upper triangle of the attention matrix to prevent the model from accessing future tokens, setting these values to negative infinity.
 - The model needs to learn how to predict the next token without “peeking” into the future.
- **Softmax:** After masking, the attention score is converted into probability by the softmax operation which takes the exponent of each attention score.
 - Each row of the matrix sums up to one and indicates the relevance of every other token to the left of it.
- **Output:** The model uses the masked self-attention scores and multiplies them with the **Value** matrix to get the final output of the self-attention mechanism.

Transformer Block: Multi-head Attention

- GPT-2 has 12 self-attention heads, each capturing different relationships between tokens.
- The outputs of these heads are concatenated and passed through a **linear projection**.




Transformer Block: MLP: Multi-Layer Perceptron



- After the multiple heads of self-attention capture the diverse relationships between the input tokens, the concatenated outputs are passed through the Multilayer Perceptron (MLP) layer to **enhance the model's representational capacity**.
- The MLP block consists of two linear transformations. The first linear transformation increases the dimensionality of the input four-fold from **768 to 3072**.
- The second linear transformation reduces the dimensionality back to the original size of **768**, ensuring that the subsequent layers receive inputs of consistent dimensions.
- Unlike the self-attention mechanism, the MLP processes tokens independently and simply map them from one representation to another.

Output Probabilities



| Tokens | Logits | Exponents | Softmax |
|------------|---------|-----------|---------|
| visualize | -135.91 | 1.00e+0 | 23.83% |
| create | -136.68 | 4.63e-1 | 11.03% |
| see | -137.12 | 2.99e-1 | 7.13% |
| make | -137.65 | 1.77e-1 | 4.21% |
| easily | -137.67 | 1.73e-1 | 4.13% |
| quickly | -137.72 | 1.64e-1 | 3.90% |
| explore | -137.78 | 1.54e-1 | 3.67% |
| find | -138.05 | 1.18e-1 | 2.82% |
| understand | -138.29 | 9.26e-2 | 2.21% |

- After the input has been processed through all Transformer blocks, the output is passed through the final linear layer to **prepare it for token prediction**.
- This layer projects the final representations into a **50,257-dimensional** space, where every token in the vocabulary has a corresponding value called logit (score).
- Any token can be the next word, so this process allows us to **simply rank** these tokens by their **likelihood** of being that next word.
- We then apply the **Softmax** function to convert the logits into a **probability** distribution that sums to one.
- This will allow us to sample the next token based on its likelihood.

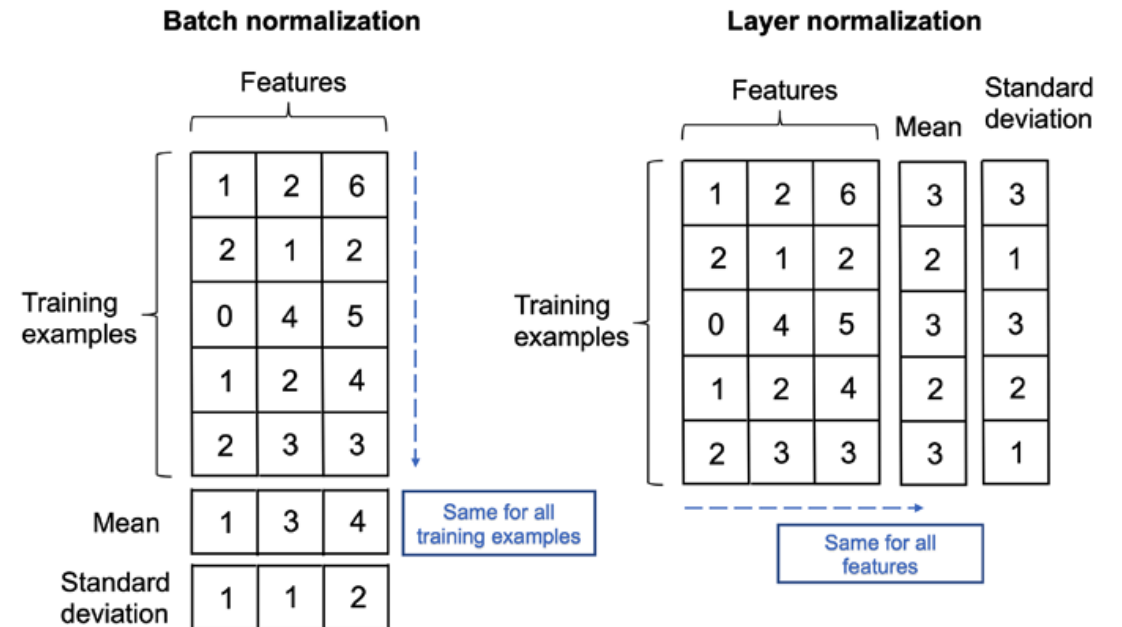
An abstract geometric structure composed of several white and light brown rectangular blocks, some of which are hollow, arranged in a stepped, architectural-like formation on a white surface.

Advanced Architectural Features

- There are several advanced architectural features that enhance the performance of Transformer models.
- While important for the model's overall performance, they are not as important for understanding the core concepts of the architecture.
- **Layer Normalization, Dropout, and Residual Connections** are crucial components in Transformer models, particularly during the training phase.

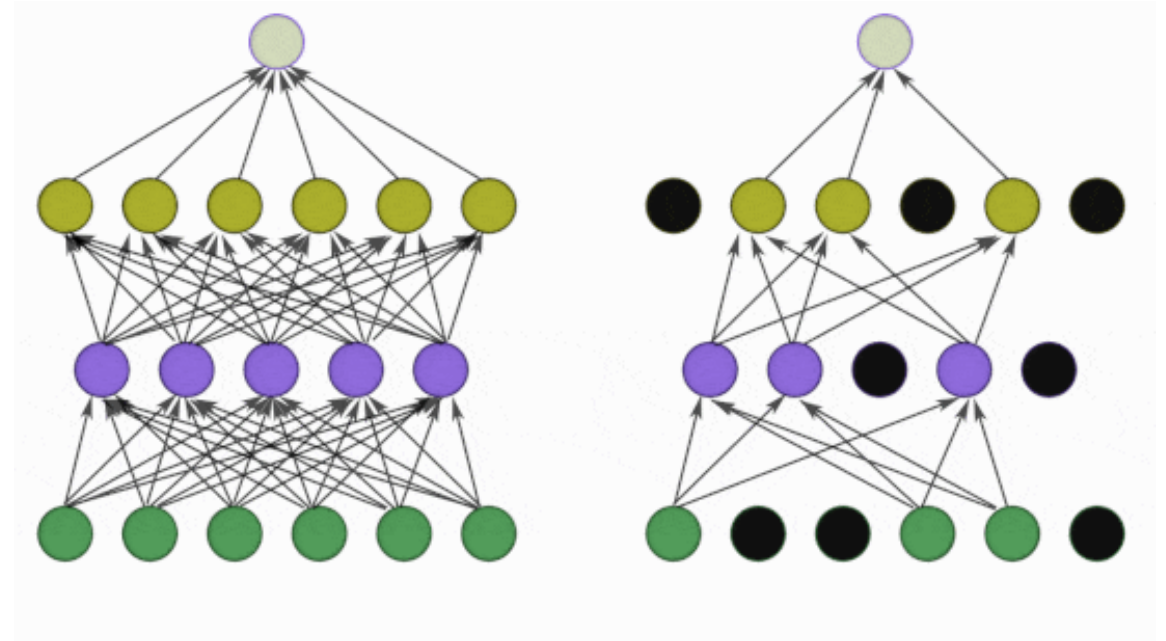
Layer Normalization

- Layer Normalization helps to stabilize the training process and improves convergence.
- It works by normalizing the inputs across the features, ensuring that the mean and variance of the activations are consistent.
- This normalization allows the model to learn more effectively and reducing the sensitivity to the initial weights.
- Layer Normalization is applied twice in each Transformer block, once before the self-attention mechanism and once before the MLP layer.



Dropout

- Dropout is a regularization technique used to prevent overfitting in neural networks by randomly setting a fraction of model weights to zero during training.
- This encourages the model to learn more robust features and reduces dependency on specific neurons, helping the network generalize better to new, unseen data.
- During model inference, dropout is deactivated.
- This essentially means that we are using an ensemble of the trained subnetworks, which leads to a better model performance.



Residual Connections

- Residual connections were first introduced in the ResNet model in 2015.
- This architectural innovation revolutionized deep learning by enabling the training of very deep neural networks.
- Essentially, residual connections are shortcuts that bypass one or more layers, adding the input of a layer to its output.
- This helps mitigate the vanishing gradient problem, making it easier to train deep networks with multiple Transformer blocks stacked on top of each other.
- In GPT-2, residual connections are used twice within each Transformer block: once before the MLP and once after, ensuring that gradients flow more easily, and earlier layers receive sufficient updates during backpropagation.

