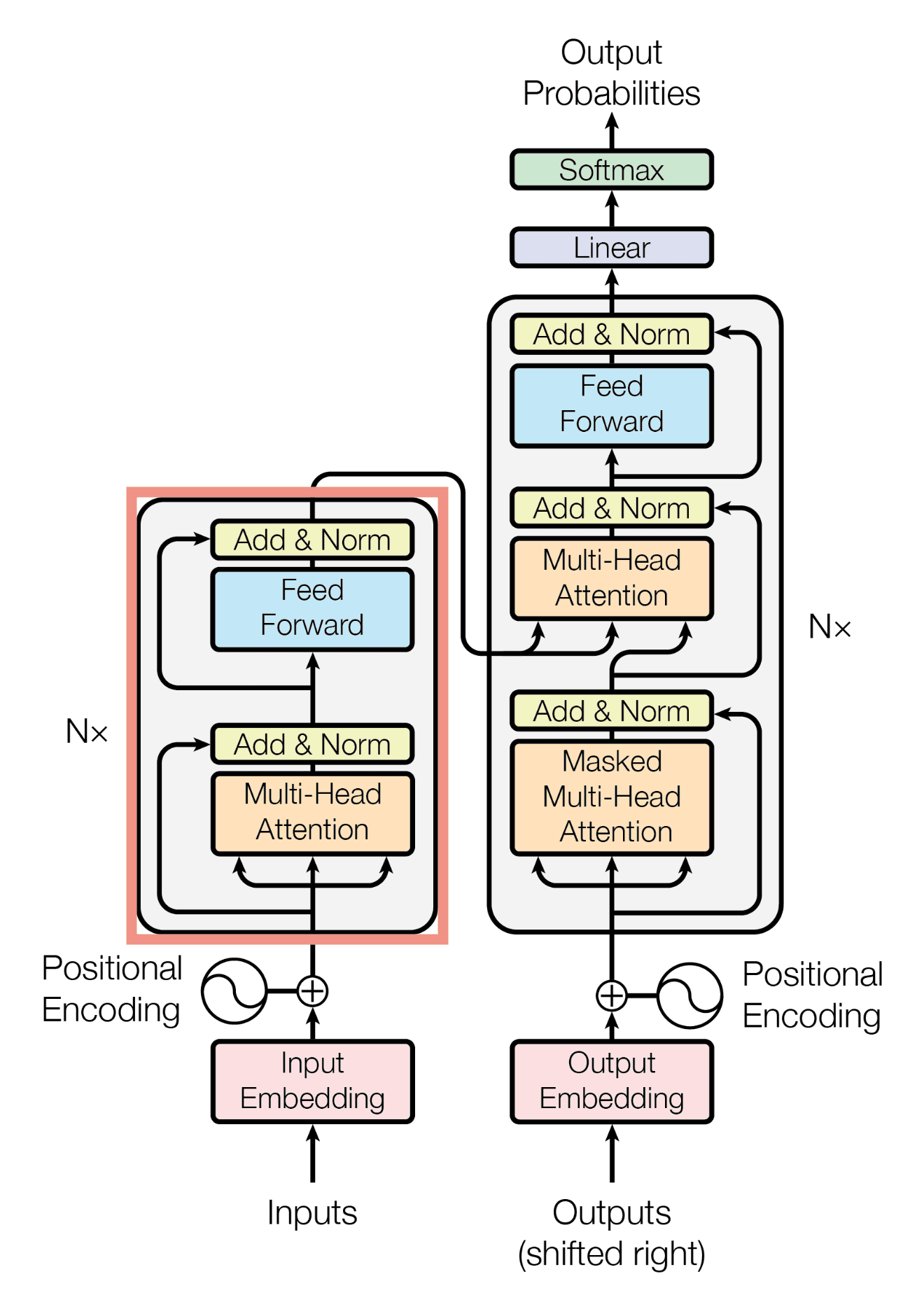
Transformer



**LSTM: Sequence-to-Sequence Processing**

- LSTM (Long Short-Term Memory) is a type of sequence-to-sequence model.

- It processes input data one word at a time, maintaining context over time.

- LSTMs excel at handling sequential tasks but operate step-by-step, which can be slower.

**Transformer: Advancement over LSTM**

- Transformers are an evolved version of LSTMs.

- Unlike LSTMs, Transformers process sequences of words simultaneously, offering higher efficiency.

- They handle data in parallel, making them faster and more suitable for complex tasks.

**Understanding the Transformation Process**

- The Transformer model specializes in transforming one form of data into another.

- For example, when prompted to write Python code to sum 10 numbers, a model like ChatGPT transforms the natural language input into code.

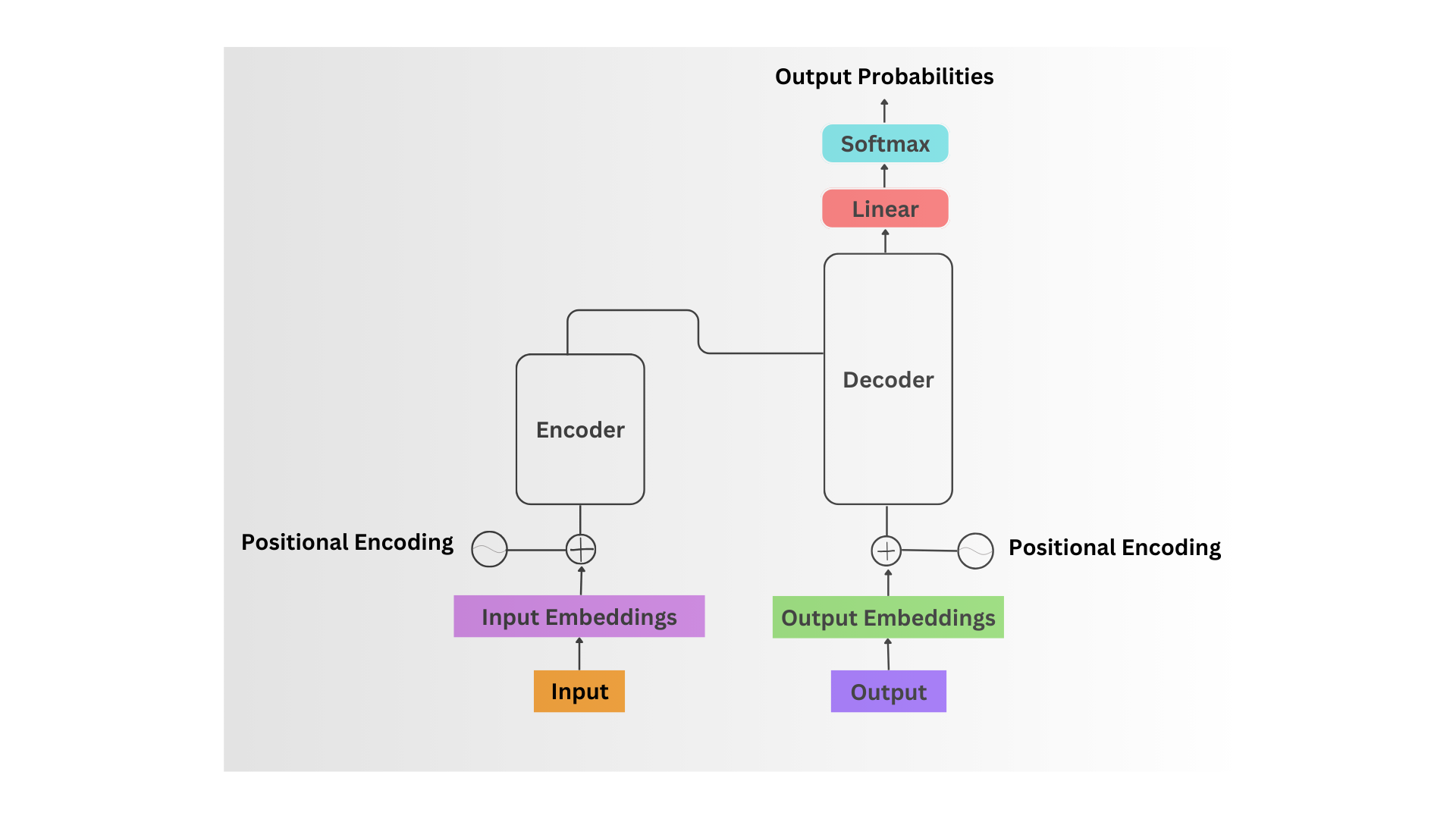
- This process exemplifies how Transformers convert inputs into meaningful outputs.

**Emulating Human-Like Thinking**

- Transformers are designed to think in a way similar to humans.

- Even without explicit instructions, the model can infer the required task, demonstrating a form of "human-like" reasoning.

- This ability to understand and generate relevant outputs is what sets Transformers apart from traditional models like LSTMs.



How Transformers Work

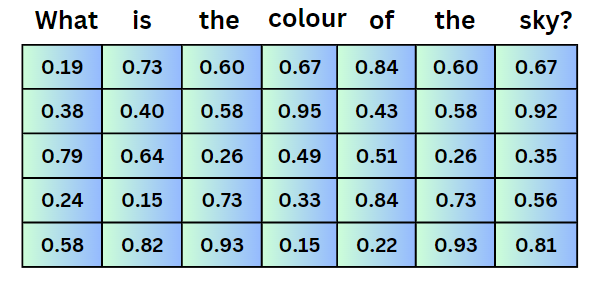
1. **Input** – What is the colour of sky? Output - Blue

In a transformer model, let's explore the process behind converting an input, such as "What is the colour of the sky?" into an output like "Blue," using embeddings and bi-pair transformations.

1. **Input Embeddings**:

When we feed the input into the transformer, it first converts the words into numerical representations called embeddings. These embeddings capture the meaning of words in the context of the sentence. In the original transformer model, the input is typically represented using 512 features, but for simplicity, we will work with 5 features here.

For instance, the phrase "What is the colour of the sky?" will be transformed into a set of numerical values that reflect the relationships and meanings of these words. This conversion is essential as it enables the model to perform mathematical operations on the input.



Note: **Bi-Pair Transformation**

In transformers, attention mechanisms use bi-pair word relationships, where each word is evaluated in relation to every other word in the sentence. This allows the model to determine which words are most relevant to the task of predicting the output. In this case, it helps identify that "colour" and "sky" are crucial for predicting the word "Blue."

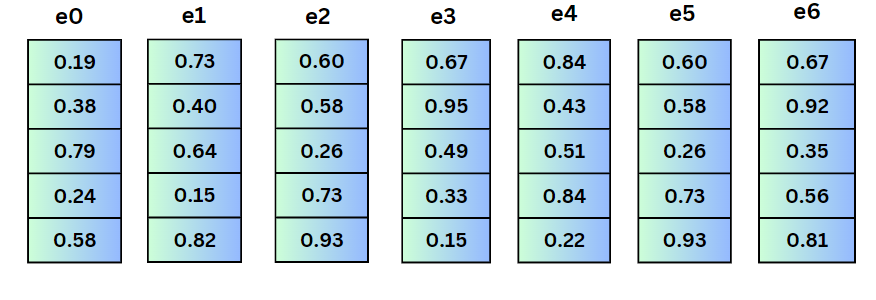
By using embeddings and bi-pair transformations, the transformer can process natural language and deliver accurate predictions.

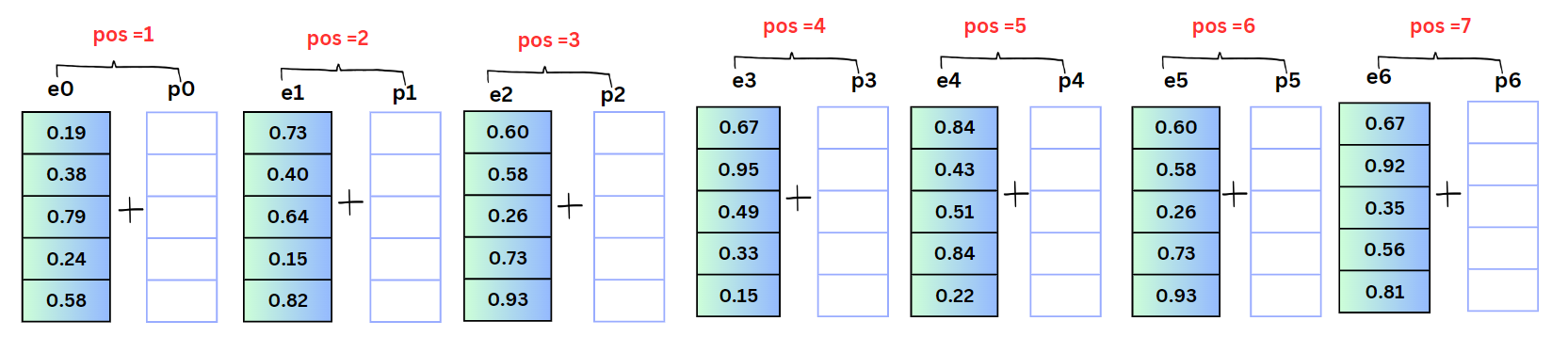
1. **Positional encoding**

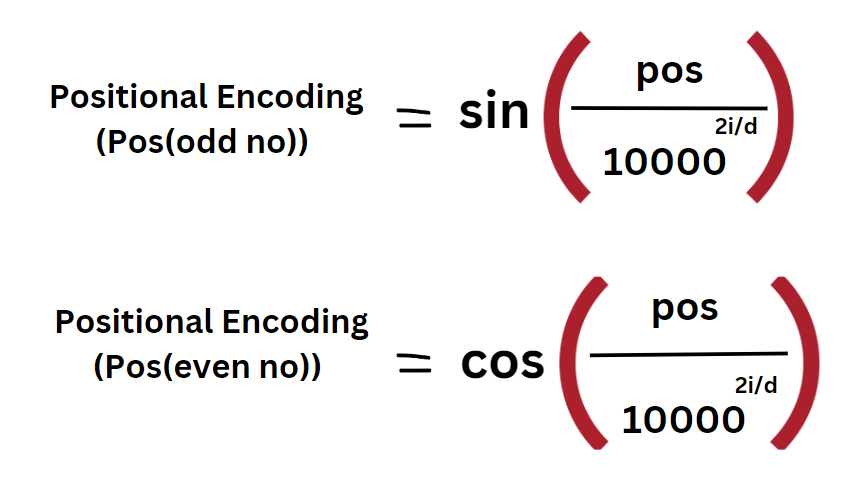
Why Positional Encoding is Important?

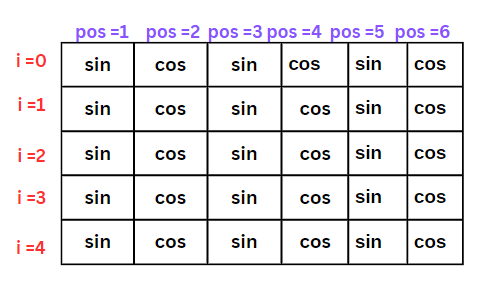
In a transformer model, all the words in a sentence are processed at the same time (in parallel). However, without knowing the order of the words, the model might get confused about which word comes before or after another. This is where positional encoding comes in.

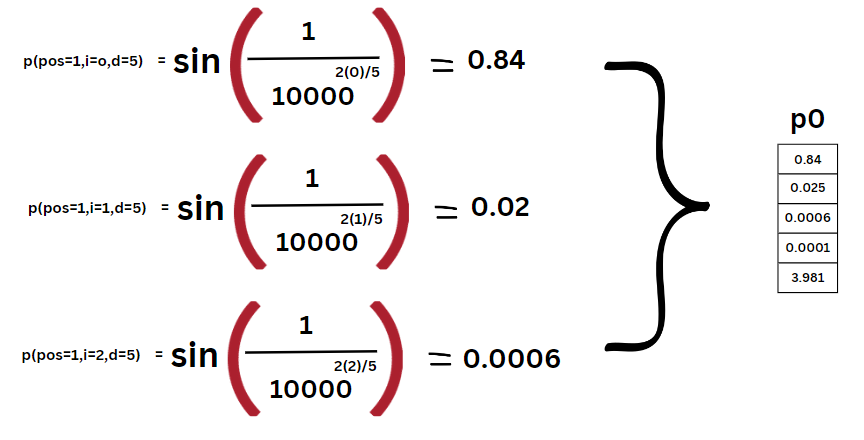
* Purpose of Positional Encoding: Since the model processes all the words together, it needs a way to understand the position of each word in the sentence.
* How It Works: Positional encoding assigns a unique number to each word based on its position in the sentence using sin and cos formula.
* Combining with Input Embeddings: After the positional encoding is added, it gets combined with the input embeddings (the numerical representation of words).
* Final Step: The combined result (positional encoding + input embedding) is then passed to the encoder, allowing the model to understand both the meaning and the order of words in the sentence.

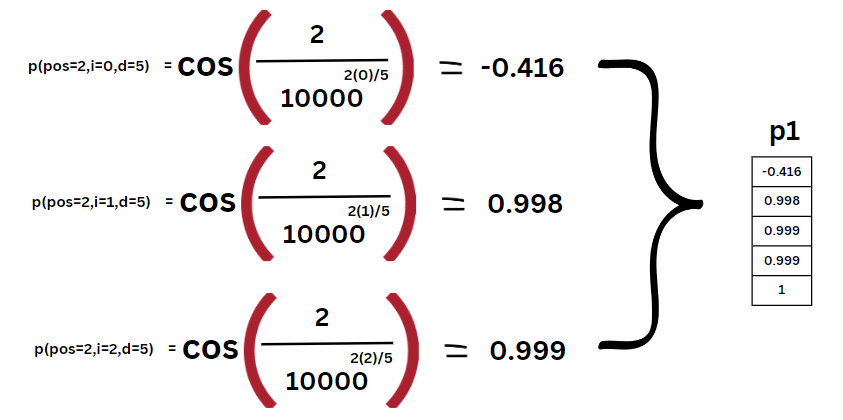


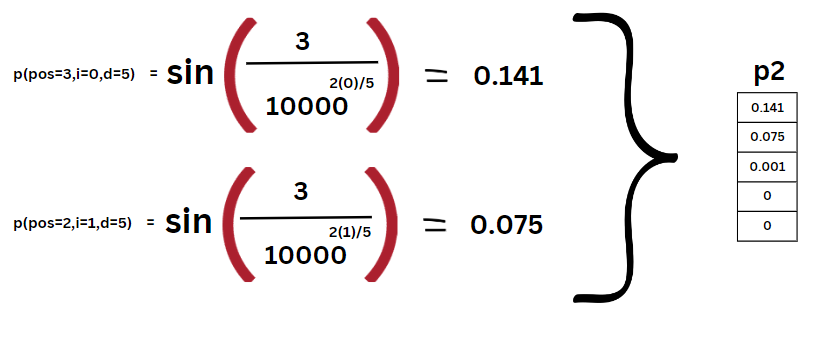
The figure above illustrates the output of the input embeddings. Please note that the values shown in the diagrams are for illustrative purposes only.

Here, we need to calculate p0,p1,…,,pn (i.e., positional encoding). If the position is even, we will use the cosine formula, and if the position is odd, we will use the sine formula, as indicated in the image below.

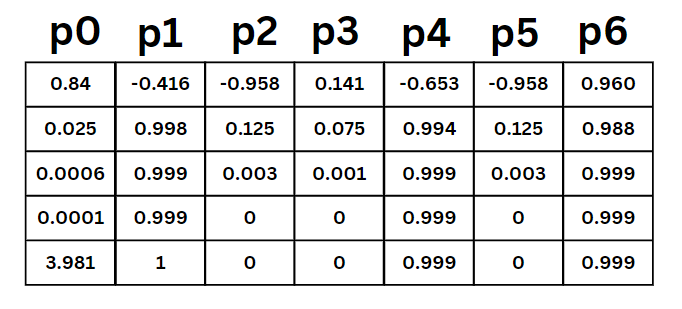


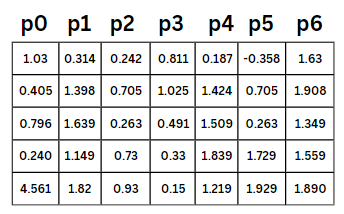
In this context, i represents the word index, D denotes the dimensionality, and Pos indicates the position in the sequence. The table above is provided to help clarify the positional information. Below, you will find sample calculations to illustrate this concept.



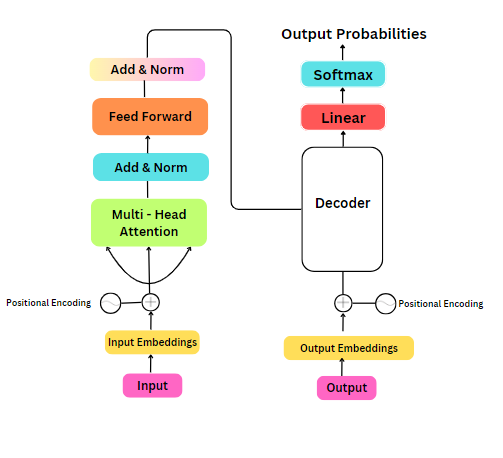


This table presents the final values of the positional encoding.





Multi-Head Attention / Self-Attention

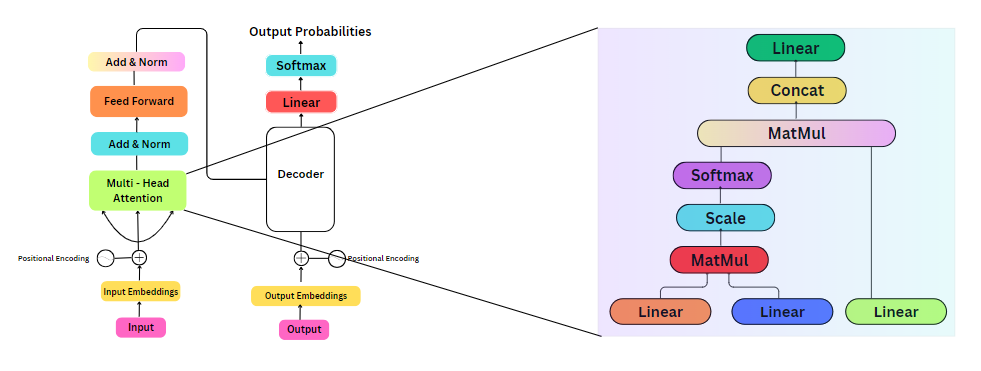


Let's consider an example of the word bass

- Bass (noun) A type of fish: "He caught a large bass in the river."

- Bass (noun) A low pitch in music: "The song has a strong bass line."

Though the spelling is the same, the meanings are different. A model should be able to identify the correct meaning based on the context of the sentence, just as humans do. To achieve this, the self-attention mechanism plays a key role.

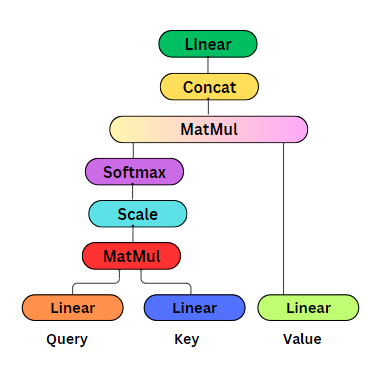


Self-Attention in Context

In a sentence, every word attends to every other word, helping the model understand context and relationships. For example, the word "bass" in "He caught a large bass in the river" would focus on related words like "caught" and "river" to infer that the meaning of "bass" refers to a fish.

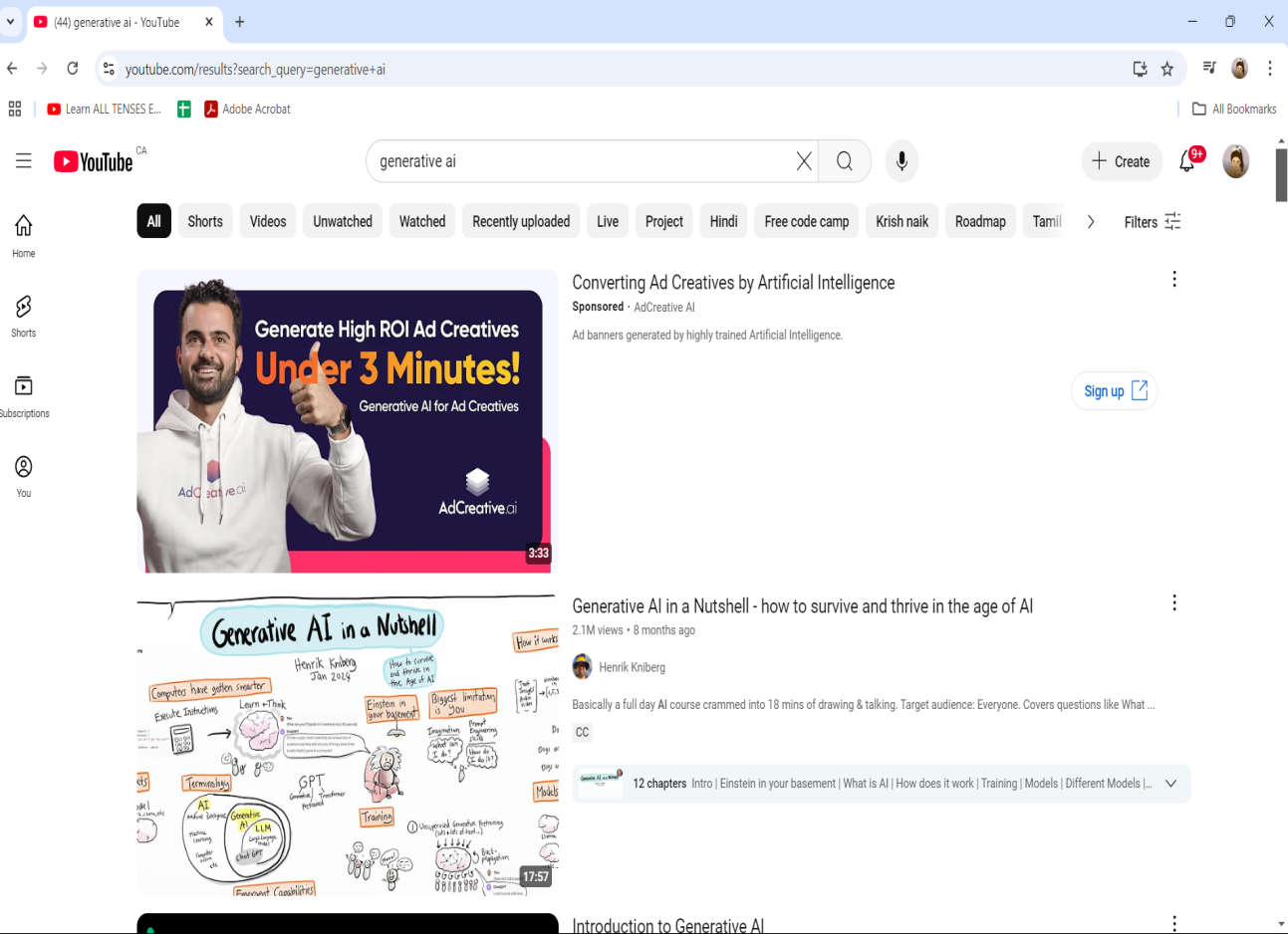
Multi-Head Attention

In multi-head attention the input is processed through queries, keys,and ,values

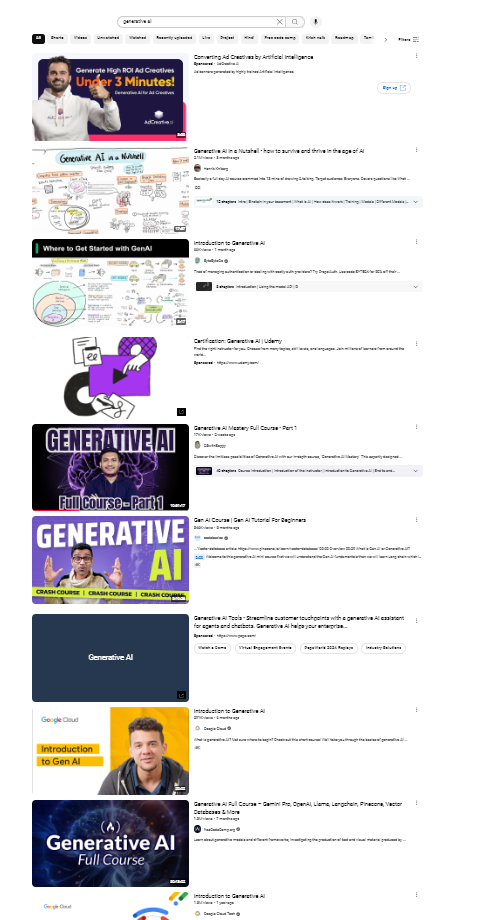


- Query: This is like the user's search query. For example, if you search for a term on YouTube or Chrome, that is the query.

**QUERY**

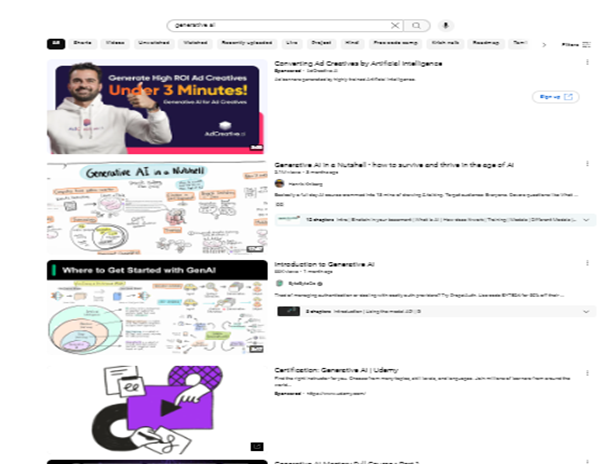


- Key: These are all the available options or content that are being compared with the query. In the YouTube example, it refers to all the videos related to your query.



**KEYS**

- Value: This is the result that is most relevant to the query. The video or content that closely matches the query is treated as the value.



**VALUE**

To find the similarity between the query and the key, the model uses cosine similarity, which measures how close the query and key are in terms of meaning. This helps the model determine which key (content) is most relevant to the query.

**Example:**

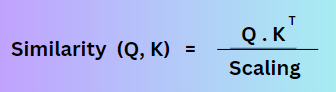
**Sentence**: "He caught a large bass in the river."

**Step-by-Step Breakdown:**

For each word in the sentence, we calculate the similarity (using cosine similarity) with every other word to understand the relationships and context. Here’s how it works:

1. **He → caught**:
   * The model calculates how similar the word "He" is to the word "caught." Since "He" is the subject (a person) and "caught" is the action (verb), there's a strong connection between them. The similarity score will be relatively high.
2. **He → large**:
   * There’s less of a direct connection between "He" and "large" because "large" is describing the bass, not "He." So, the similarity score will be lower.
3. **He → bass**:
   * The similarity between "He" and "bass" might be moderate, as "bass" refers to the object of "He’s" action, but they are still somewhat connected.
4. **Caught → bass**:
   * The model finds a high similarity between "caught" and "bass" because these words are directly related in context: "caught" is the action done to "bass."

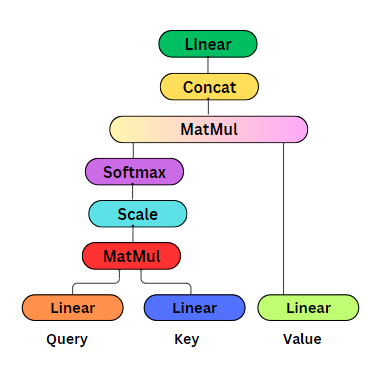
This process continues for every word pair in the sentence (e.g., **"caught → river"**, **"large → bass"**), generating an attention matrix where each word attends to others with varying levels of importance. This matrix helps the model understand which words in the sentence are more important for each word to focus on, allowing the model to disambiguate meanings in context (like distinguishing "bass" the fish from "bass" the musical tone).





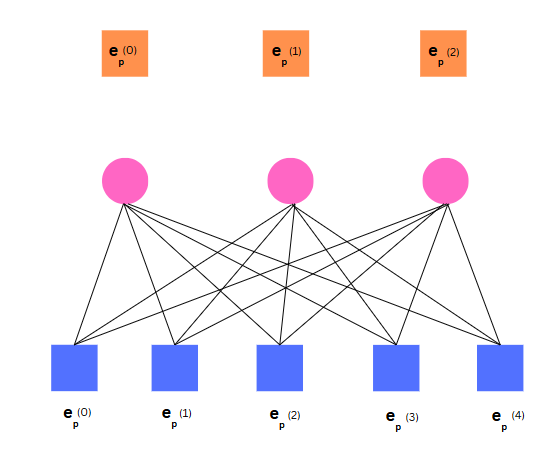
 If the cosine similarity is **1**, the vectors are identical (pointing in the same direction). - Similar

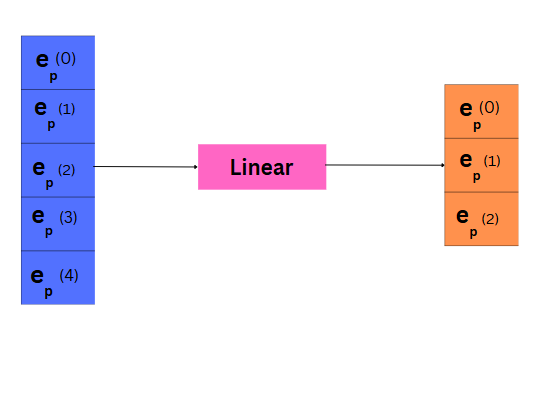
 If the cosine similarity is **-1**, the vectors are completely opposite (pointing in opposite directions). – Dis-similar.



Purpose of linear in the above figure:

Linear is mainly used for redcuing the dimensionality





There is no activstion function in linear here, and it works based on weight.

Mapping input and output can vle possible