**Self-attention in Transformers**

======================================================================

Self-attention is a mechanism which enhances the information content of input embeddings by incorporating contextual information, enabling models to focus on relevant parts of the input for the task at hand.

Self-attention is a mechanism in neural networks that allows models to weigh the importance of different parts of an input sequence when making predictions.

Self-attention, introduced with the Transformer architecture, enables models to capture context and dependencies within a sequence without relying on sequential processing like recurrent neural networks (RNNs).

--------------------------------------------------------------------------------------------------------------

**Why there is a word – Self:**

In the context of self-attention in Transformers, the word "Self" signifies that the attention mechanism is operating within a single input sequence.

Here's a breakdown of what that means:

* **Relating elements within the same sequence**: Instead of the attention mechanism being used to relate one sequence to another (like in traditional encoder-decoder attention where the decoder attends to the encoder's output), self-attention allows each position in the input sequence to attend to all other positions in the **same** sequence.
* **Understanding internal relationships:** The model learns the relationships and dependencies between different elements (e.g., words in a sentence, patches in an image) within that single input. It determines which parts of the sequence are most relevant to other parts when forming a representation of the entire sequence.

--------------------------------------------------------------------------------------------------------------

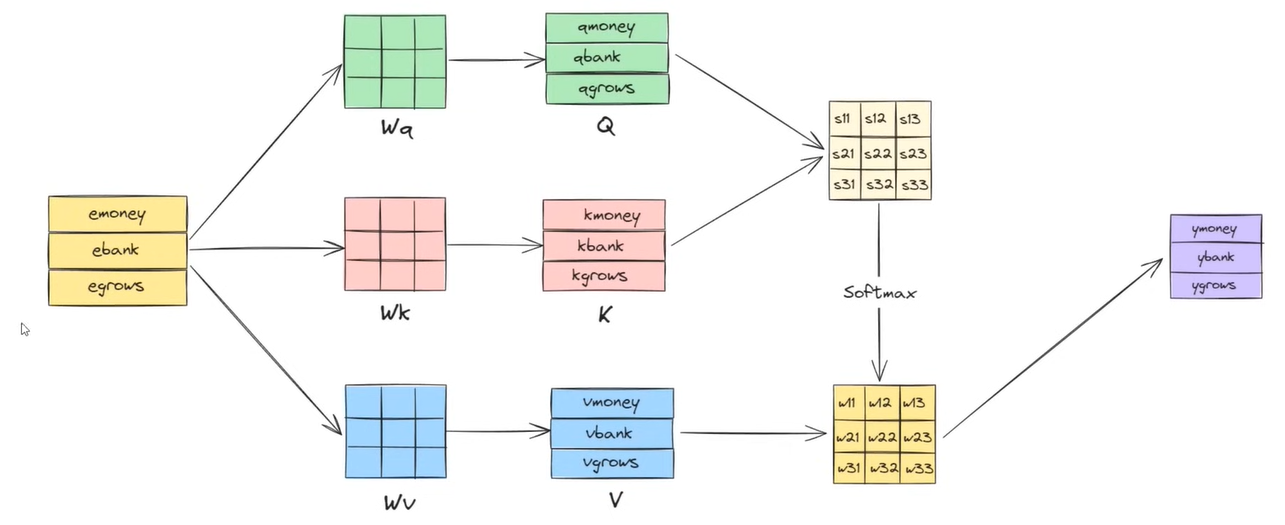
**Key aspects of self-attention:**

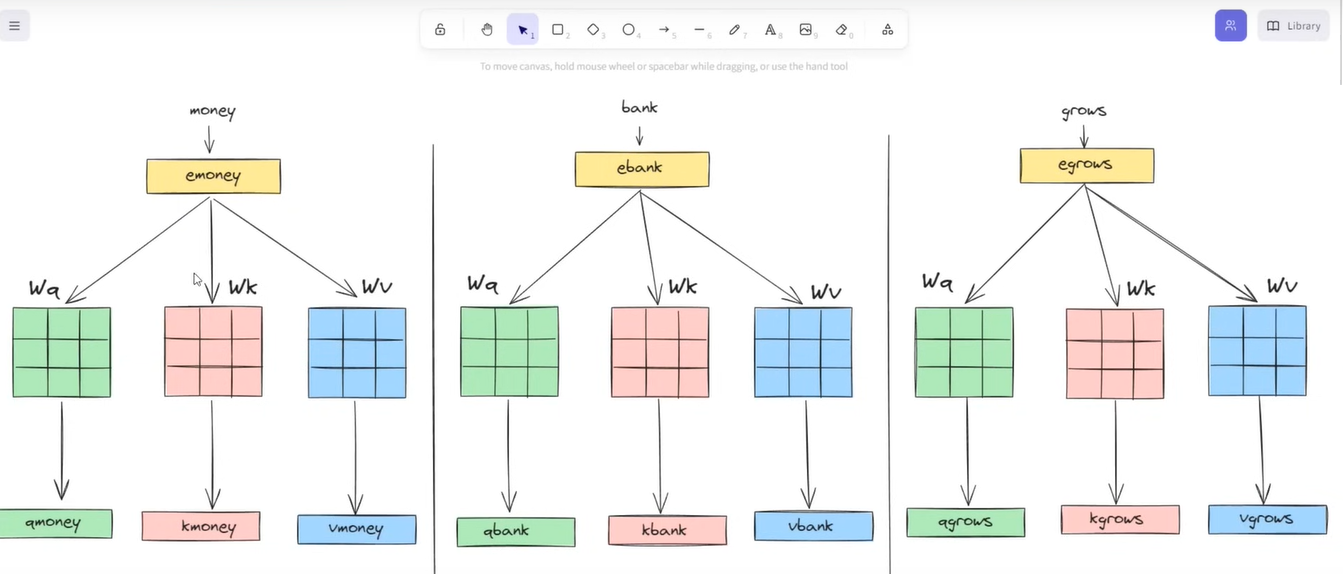
1. **Weighting Importance:** Self-attention assigns weights to different elements in the input sequence, indicating how relevant each element is to others when making predictions.
2. **Contextual Understanding:** By attending to itself, the model can understand relationships between words, tokens, or other input units within the same sequence.
3. **Parallel Processing:** Self-attention allows parallel processing of the input sequence, unlike RNNs which process elements sequentially.
4. **Transformer Foundation:** Self-attention is a fundamental component of Transformer models, which are widely used in NLP tasks and are the foundation for large language models (LLMs).
5. **Enhanced Input Embeddings:** Self-attention enhances the information content of input embeddings by incorporating contextual information, enabling models to focus on relevant parts of the input for the task at hand.
6. **Dependencies and Relationships:** Self-attention helps models capture complex dependencies and relationships between elements within a sequence.
7. **Multiple Heads:** Multi-head attention, an extension of self-attention, enhances a model's ability to capture diverse contextual information by using multiple parallel attention operations.

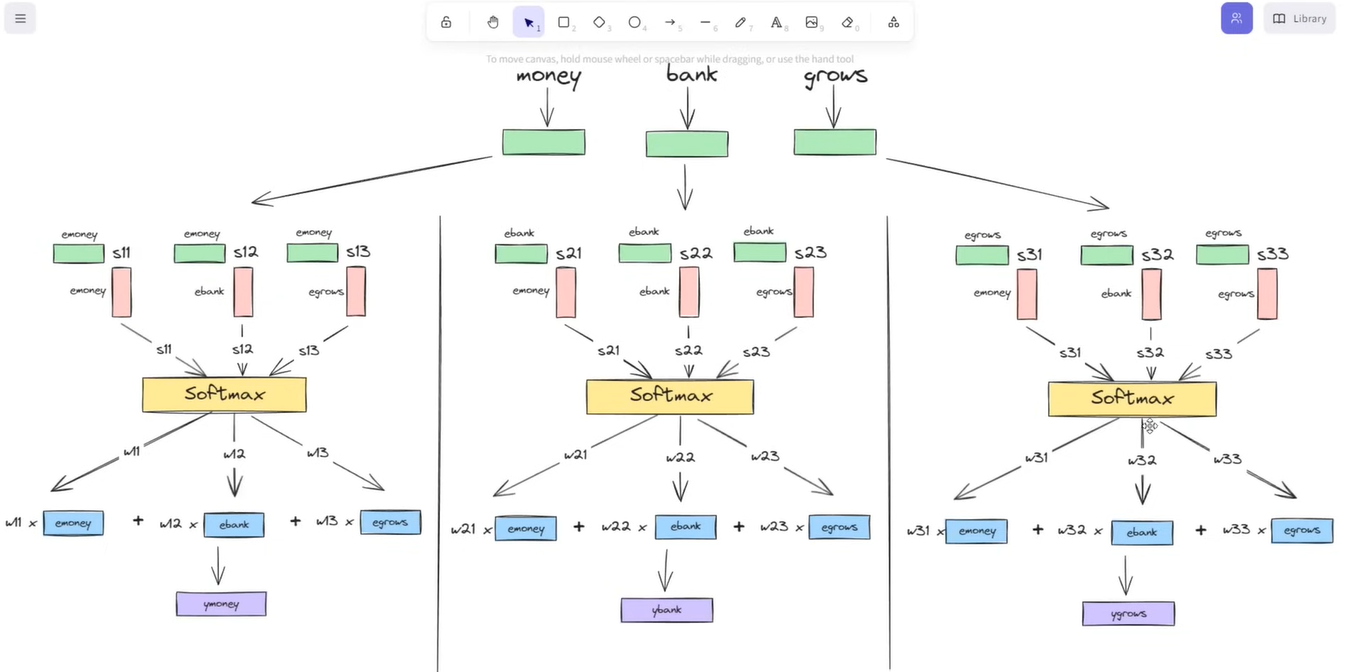
--------------------------------------------------------------------------------------------------------------

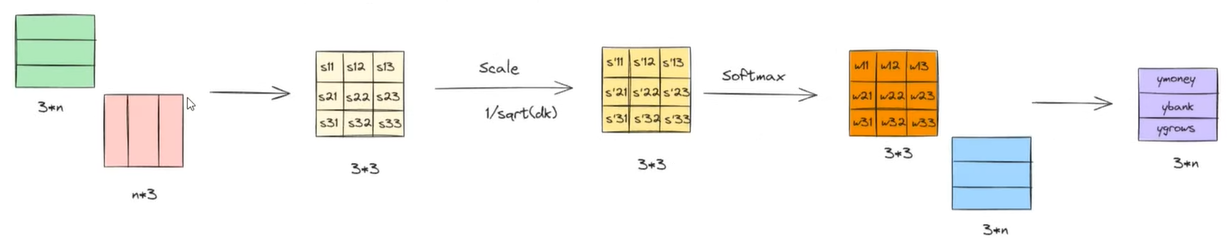
**Calculation of contextual embedding values:**

1. For each word in the input sentence, we generate normal embeddings first.
2. Create three matrices, (as shown below) with the weights of randomly initialized parameters.
3. Create three vectors called as Query, Key and Value by multiplying the embeddings.
4. Get the S matrix (Similarity scores) by multiplying Q and K vectors.
5. Use softmax on Similarity scores and get the W matrix.
6. Multiply W and value matrices to get the final contextual embeddings for each word.
7. All the above steps can happen in parallel, which is main advantage of transformers.

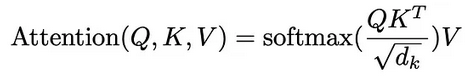








To write in a simple mathematical form:



Here Square root of dk is added – to reduce the variance introduced in the dot products of the above all the vectors. (dk – dimensions of Key matrix).