What is Attention?

w/ Positional Embedding

Weimao Ke
wk@drexel.edu

Example Sequence

“

This is my **mother**.
This is my **father**.

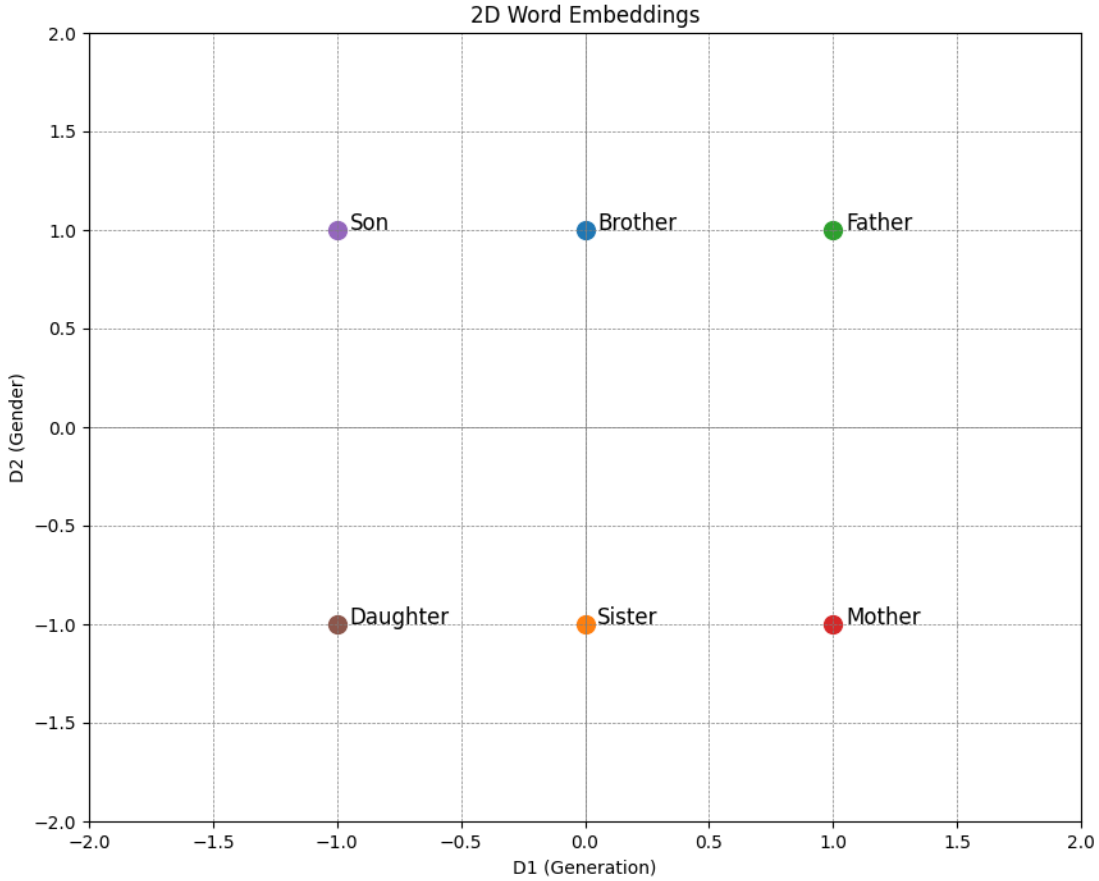
”

Existing Word Embedding

```
Embedding(Father)  = [1,  1]
Embedding(Mother)  = [1, -1]
Embedding(Son)     = [-1, 1]
Embedding(Daughter)= [-1,-1]
```

Visualized Embedding

We'll focus on "Mother", "Father", and "Daughter".



Add Positional Encoding

$$PE(p, 2i) = sin(p/10000^{2i/d})$$
$$PE(p, 2i+1) = cos(p/10000^{2i/d})$$

where:

1. $2i$ and $2i+1$ are even/odd indices of each dimension.
2. d is the size/dimensionality of the embedding.

Q and K

Given each word's (token) position in:

```
1    2  3    4*
This is my mother.
This is my father.
5    6  7    8*
```

We can compute and add the positional encodings.

Mother's Embedding with Positional Encoding:

```
E(Mother) = [1 + sin(4), -1 + cos(4)]
                    = [1 + 0.7568, -1 + -0.6536]
                    = [1.7568, -1.6536]
```

Father's Embedding with Positional Encoding:

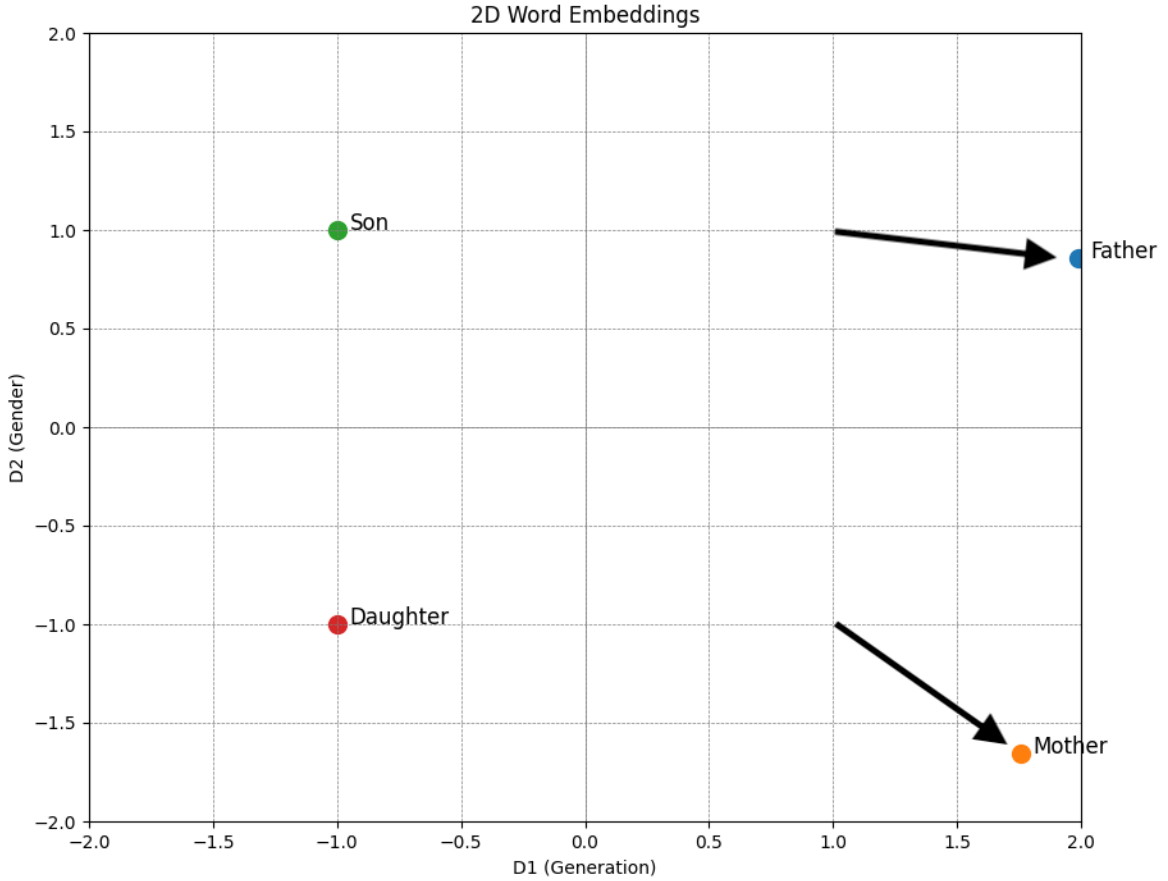
```
E(Father) = [1 + sin(8), 1 + cos(8)]
= [1 + 0.9893, 1 - 0.1455]
= [1.9893, 0.8545]
```

Now, NO positional encoding for Daughter:

```
E(Daughter) = [-1, -1]
```

- "Son" and "Daughter" do not occur in input here.
- Imaginary tokens in the sequence for comparison.
- May occur in other sequences.

Based on the added positional encoding:



The embedding for "Father" and "Mother" are impacted as they appear in different positions of the sequence. Given these new embeddings, we will construct the Q, K, and V vectors using learnable weights. For simplicity, we will propose some hypothetical weights such that the dot product between Q of "mother" and K of "father" is greater than the dot product between Q of "mother" and K of "daughter".

Note

- Q vector is the **query** vector to represent the interest of the token, e.g. "mother".
- K vector is the **key** (self) representation of **another** token, e.g. "father".
- QK^T (dot product) represents how related (relevant or interesting) two tokens are in the context (word+pos):

```
one token (with K) -> another (with Q)
```

Q, K, V Weights (For simplicity):

Given:

$$Q_t = E_t \times W_q$$
$$K_t = E_t \times W_k$$
$$V_t = E_t \times W_v$$

Identity matrix as our hypothetical weight:

$$W_q = W_k = W_v = \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix}$$

Given these weights

Q, K, V vectors are the same as embeddings:

$$Q_{mother} = K_{mother} = V_{mother} = [1.7568, -1.6536]$$

$$Q_{father} = K_{father} = V_{father} = [1.9893, 0.8545]$$

$$Q_{daughter} = K_{daughter} = V_{daughter} = [-1, -1]$$

And Attention! :-)

Now, we compute the attention scores using the dot product:

- $Attention(mother, father) = Q_{mother} \cdot K_{father}^T$
  $= 1.7568 \times 1.9893 + (-1.6536) \times 0.8545 \approx 2.5013$
- $Attention(mother, daughter) = Q_{mother} \cdot K_{daughter}^T$

$$= 1.7568 \times (-1) + (-1.6536) \times (-1) \approx 0.1032$$

- The attention score between "mother" and "father" is significantly larger than that between "mother" and "daughter", which means that,
- given the current Q, K, and V vectors, "father" would indeed have a stronger interest/attention in relation to the token "mother" compared to "daughter".

Normalization and V

The self-attention mechanism in Transformers includes **normalization** by the square root of the dimension of the key vectors and then using the scores to compute a weighted sum of the value vectors.

Given our two-dimensional vectors:

1. **Compute the attention scores**:

   For Mother and Father:

$$Score(mother, father) = \frac{Q_{mother} \cdot K_{father}^T}{\sqrt{2}} \approx \frac{2.5013}{\sqrt{2}} \approx 1.7689$$

   For Mother and Daughter:

$$Score(mother, daughter) = \frac{Q_{mother} \cdot K_{daughter}^T}{\sqrt{2}} \approx \frac{0.1032}{\sqrt{2}} \approx 0.073$$

2. **Compute the attention weights using the softmax function**:

   For Mother and Father:

$$Weight(mother, father) = \frac{exp(1.7689)}{exp(1.7689) + exp(0.073)} \approx 0.95$$

   For Mother and Daughter:

$$Weight(mother, daughter) = \frac{exp(0.073)}{exp(1.7689) + exp(0.073)} \approx 0.05$$

3. **Compute the final attention value**:

$$\text{AttentionValue} = Weight(mother, father) \times V_{father} + Weight(mother, daughter) \times V_{daughter}$$

Attention for "Mother"

Using our V vectors:

$$\approx 0.95 \times [1.9893, 0.8545] + 0.05 \times [-1, -1]$$

$$\approx [1.8948, 0.8113] - [0.05, 0.05]$$

$$\approx [1.8448, 0.7613]$$

So, the final attention value of the "mother" token is the weighted sum of its relations to the other tokens in in the sequence, e.g. "father", "daughter", etc. The relations are captured in the Q and K vectors.

In the case of "Mother"

1. Q ("Mother" as query) shares a strong interest in the K vector of "Father".
2. This, coupled with V (weights for these associations), will help associate tokens in context.
3. Contenxt means: word embedding plus positional encoding with Q, K, V embeddings.

Notes

The result such as $[1.8448, 0.7613]$ is the output of the self-attention mechanism for the word "Mother", based on its relationships to the other words in the sequence (in this case, "Father" and "Daughter").

In the Transformer architecture

1. Each token (or word) is initially represented by its word embedding.
2. A positional encoding (which can be based on sinusoidal functions or learned) is added to this word embedding to give the model information about the position of the word in the sequence. This results in the positionally-encoded embedding, like the ones we computed for "mother" and "father".

3. This positionally-encoded embedding goes through the self-attention mechanism, producing an output like ([1.8448, 0.7613]) for "mother". This output captures both the meaning of the word "mother" and its relationships to other words in the sequence.
4. Every token in the sequence will have a corresponding output from the self-attention mechanism, which is a combination of its own embedding and the relationships it has with other tokens in the sequence.

This self-attention output is then passed through the rest of the Transformer's layers (feed-forward networks, combined self-attention heads/layers, etc.) to produce the final outputs.

References

https://towardsdatascience.com/attention-is-all-you-need-discovering-the-transformer-paper-73e5ff5e0634

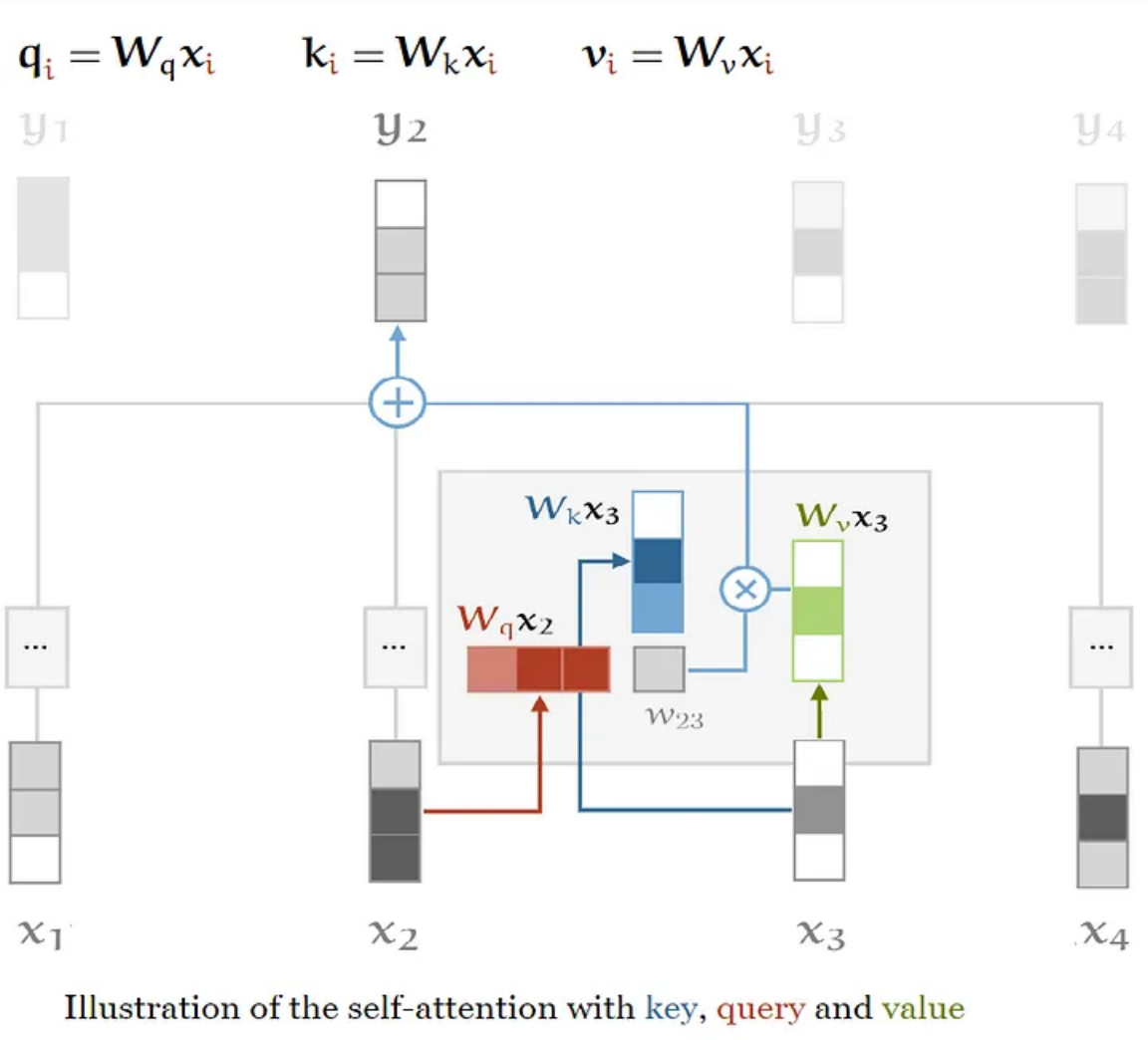


Illustration of the self-attention with key, query and value