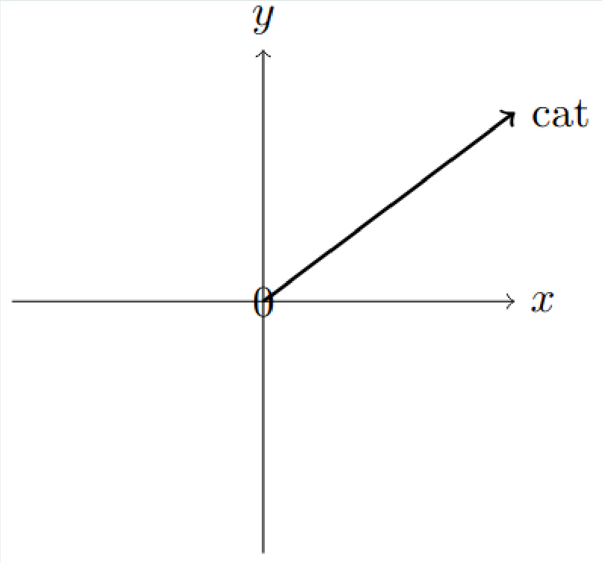
Word Embeddings in high dimensional spaces and Transformers

Vanni Leonardo

Word Embeddings

We want to find a good way to represent words that a computer can understand.

First, we ensure the representation can be used by a machine



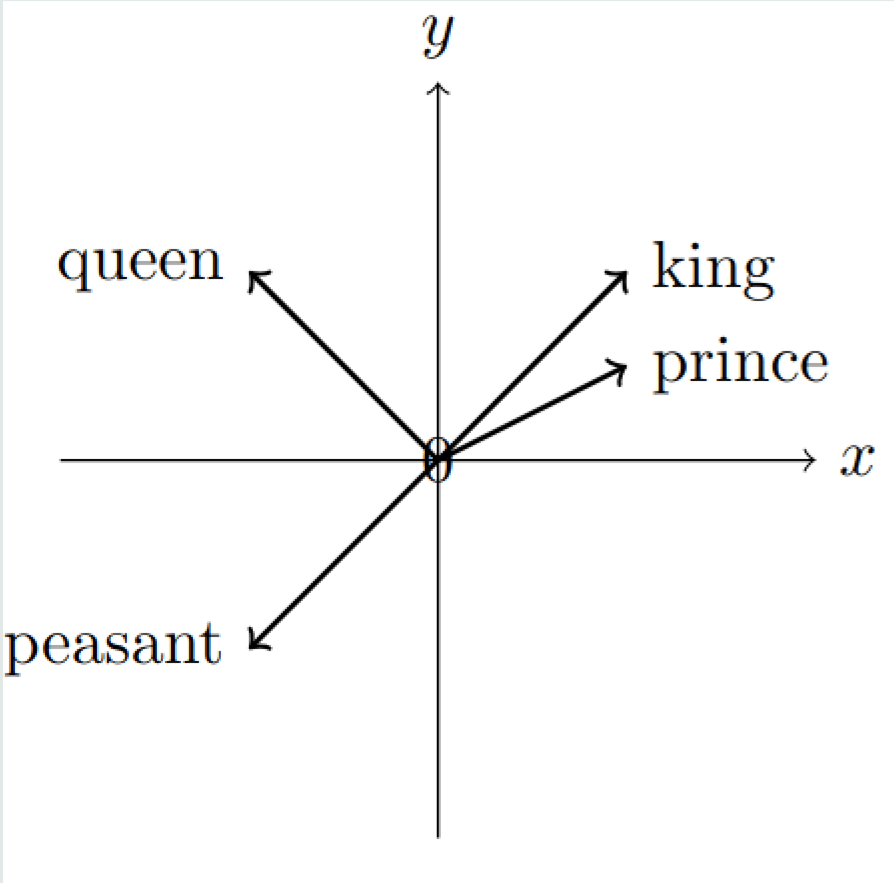
$$Cat = \vec{E} = \begin{bmatrix} x_1 \\ x_2 \\ \dots \\ x_{d-1} \\ x_d \end{bmatrix} \quad s.t\ x_i \in \mathbb{R}\ \forall i, d >> 100$$

$$\text{one-hot encoding is s.t } x_j = 0\ \forall j \neq i$$

$$Let\ \vec{E}_1 \cdot \vec{E}_2 := ||\vec{E}_1||||\vec{E}_2||cos(\theta) \implies \vec{E}_1 \approx \vec{E}_2 \iff \theta \approx 0 \iff \vec{E}_1 \cdot \vec{E}_2\ large$$

Next, we impose conditions to capture relationships between words that are semantically related

Map words to vectors in a vector space using models like Word2Vec

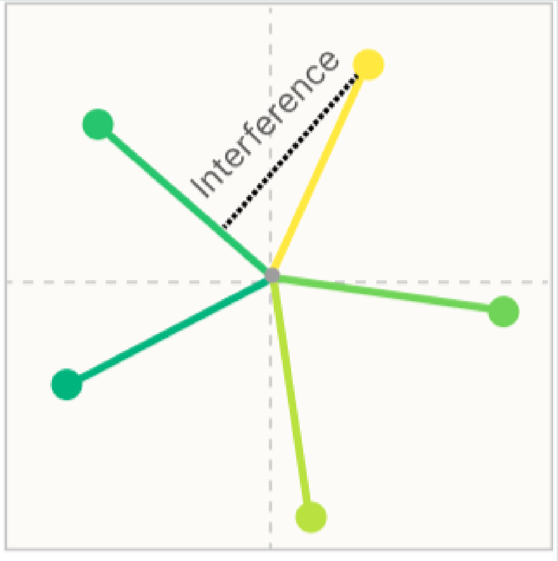


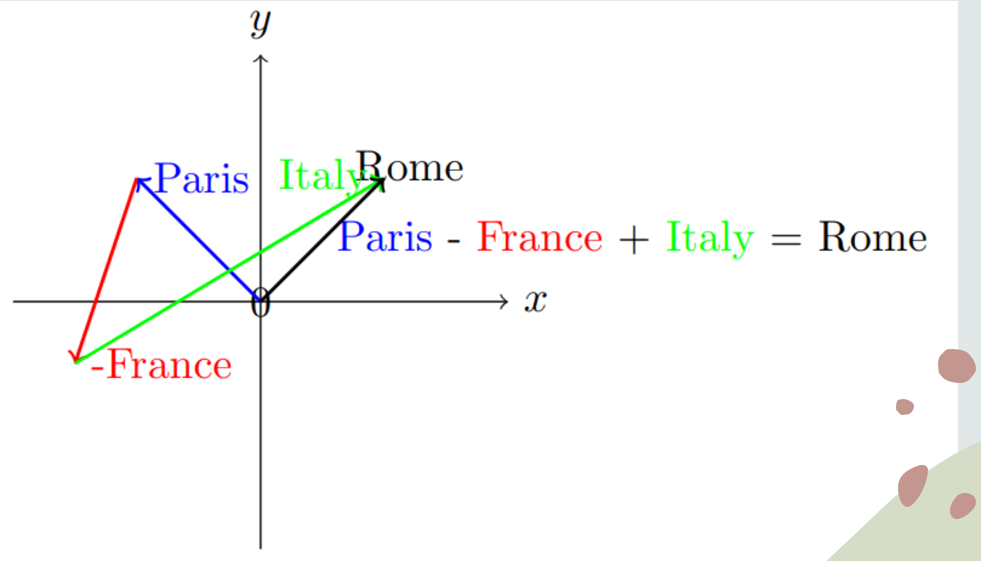
High Dimensional Spaces

Each dimension can now encode some aspect of meaning, like gender, tense or geographical location.

Cosine similarity is perfect

Superposition

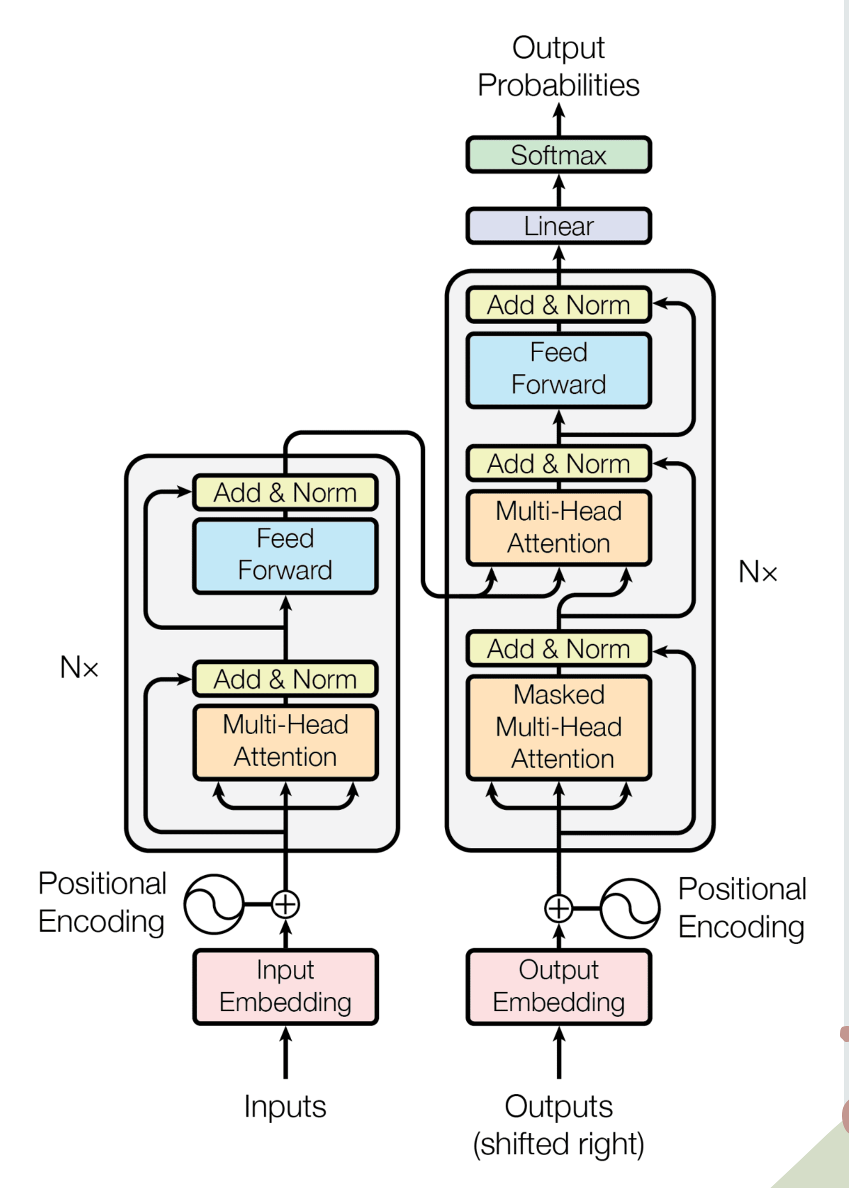




Overview of Transformers

Recurrent Neural Networks (RNNs) and LSTMs process words sequentially, leading to errors in long-term dependencies

Transformers solve this by looking at all the words in a sentence at once



Self-Attention Mechanism

The cat is on the table.

Given a sentence of d words $\vec{E} = [\vec{E}_1, ..., \vec{E}_d]$, calculate Key, Query,

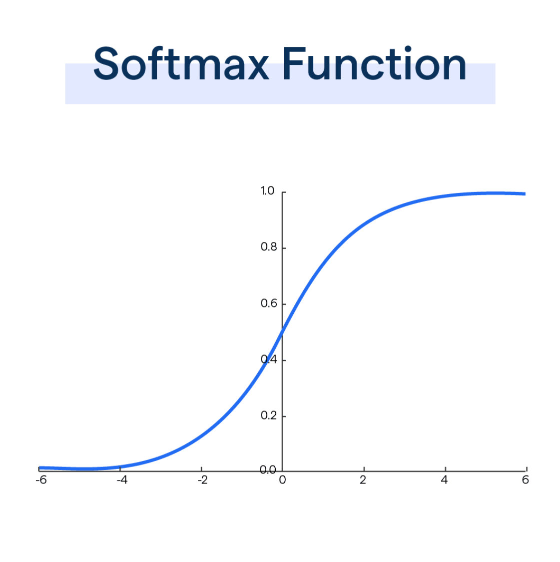
Value Vectors for each embedding with W_Q, W_K, W_V **learned**

$$W_Q \cdot \vec{E}_1 = \text{ what info is } \vec{E}_1 \text{ looking for?}$$

$$W_K \cdot \vec{E}_1 = \text{ what info does } \vec{E}_1 \text{ offer}$$

$$W_V \cdot \vec{E}_1 = \text{ what info does } \vec{E}_1 \text{ contribute when it's relevant to other words}$$

Value Updates



For each word, we ask its query to every (preceding if decoder) word

$$\forall i \ relevance_{ji} = Q_i \cdot K_j \ s.t \ j < i$$

$$\text{Rescale by } \sqrt{d_k} \text{ and softmax : } attention_{ij} = \frac{exp(\vec{Q}_i \cdot \vec{K}_j)}{\sum_k exp(\vec{Q}_i \cdot \vec{K}_k)}$$

$$\text{Update the word to include preceding context } \vec{E}_i^{NEW} = \sum_j attention_{ij} \cdot V_j$$

Self-Attention Algorithm

Algorithm 1 Attention Mechanism (Encoder/Decoder)

Require: $\vec{E} = [\vec{E}_1, ..., \vec{E}_d]$ ▷ Input word embeddings for d words
Require: W_Q, W_K, W_V ▷ Learned weight matrices for Query, Key, and Value

for each word $i = 1$ to d **do**
    $Q_i \leftarrow W_Q \vec{E}_i$ ▷ Compute Query for word i
    $K_i \leftarrow W_K \vec{E}_i$ ▷ Compute Key for word i
    $V_i \leftarrow W_V \vec{E}_i$ ▷ Compute Value for word i
end for
for each word $i = 1$ to d **do**
    for each word $j = 1$ to d **do** ▷ Attend to all words in Encoder
        if Decoder and $j > i$ **then** ▷ Causal mask in Decoder
            $attention_{ij} \leftarrow 0$ ▷ Ignore future words
        else
            $attention_{ij} \leftarrow \frac{\exp(Q_i \cdot K_j / \sqrt{d_k})}{\sum_k \exp(Q_i \cdot K_k / \sqrt{d_k})}$
        end if
    end for
    New $\vec{E}_i \leftarrow \sum_{j=1}^{d} attention_{ij} \cdot V_j$ ▷ Weighted sum of Values
end for
return [New $\vec{E}_1, \ldots,$ New $\vec{E}_d$] ▷ Updated word embeddings

Encoder

Models: BERT and variants

It allows every word in the sequence to attend (gather context) to every other word.

Gather the full context before making a prediction

Use cases: text classification, translation, summarization, named entity recognition.

Decoder

Attends only to previous words, enforcing sequential generation process.

Use cases: text generation, language modeling, question answering

Models: GPT and LLMs in general