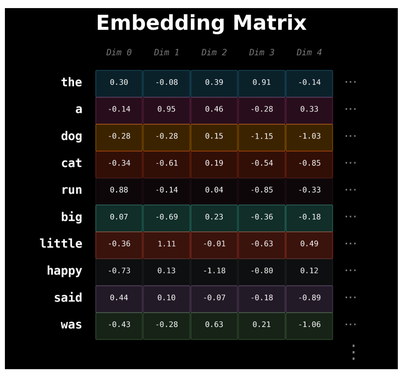
Building a GPT from Scratch

By David Rivard
Advised by Dr. Robert Kipka

The Project

While Generative Pretrained Transformers (GPTs) are widely perceived as true intelligence, the “Artificial” in AI is often overlooked. In reality, a GPT is a numerically optimized conditional probability model. To understand how this model works, we built a Transformer entirely from scratch to run on a laptop with no external deep learning frameworks using only low-level math libraries for matrix multiplication.

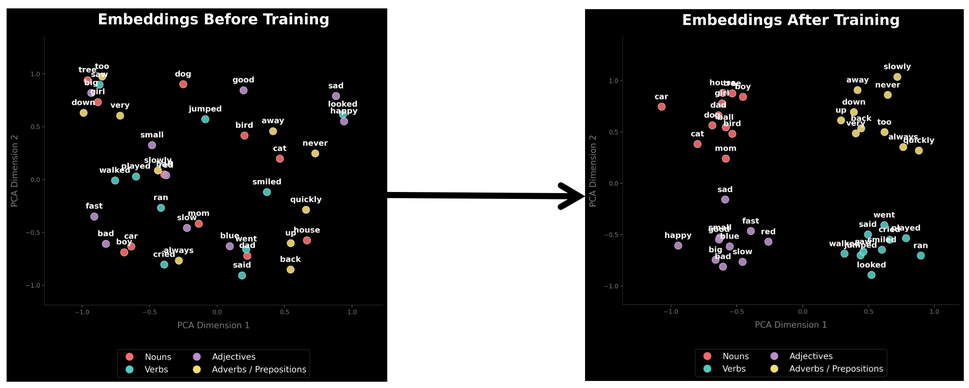


1. The Embeddings

- The model cannot read English. Each word used is stored as a unique vector of real numbers.
- These numbers are initially random and get shaped through training.

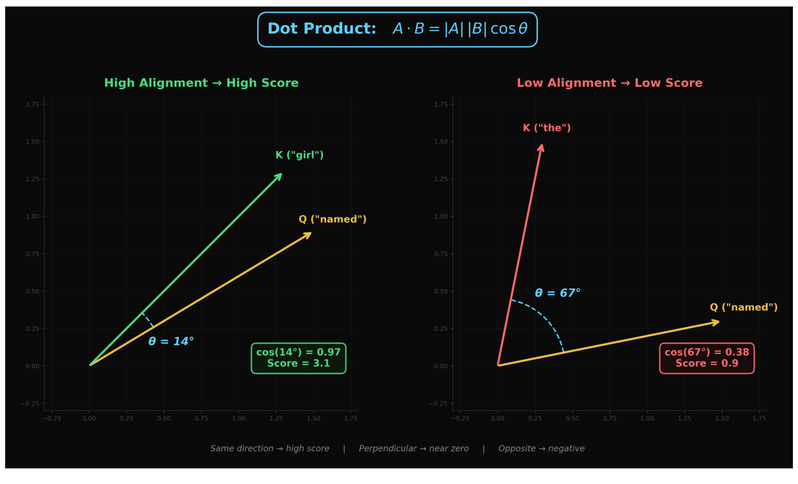
2. Principal Component Analysis (PCA)

- PCA projects these high-dimensional vectors onto a 2D surface, revealing that the model has learned to cluster similar words together geometrically.



3. Dot Product

- Dot product measures similarity between two vectors.
- Aligned vectors → High scores. Misaligned vectors → Low scores.
- This crucial operation appears throughout the model.



Softmax: Converting Scores to Probabilities

Vector B (Word)	Dot Product Score	Exponent e^{Score}	Softmax Probability
The	-1.4	0.25	1.0 %
little	1.1	2.86	11.0 %
girl	2.6	13.46	**51.9 %**
named	2.2	9.39	36.2 %

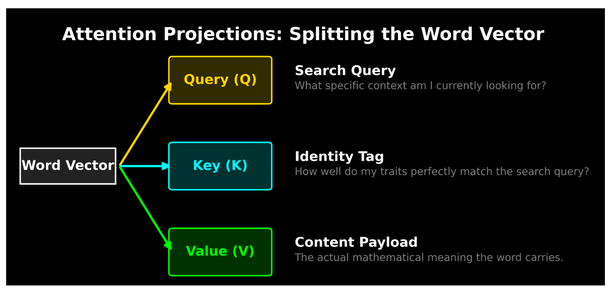
4. Softmax

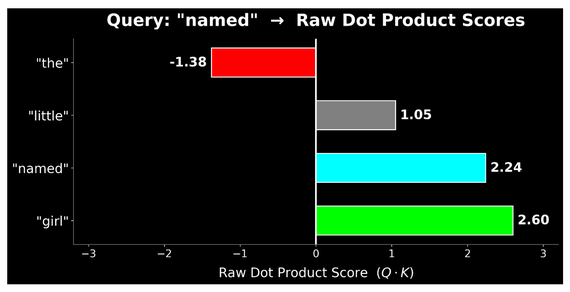
- Softmax converts raw dot-product scores into a probability distribution.
- Formula:

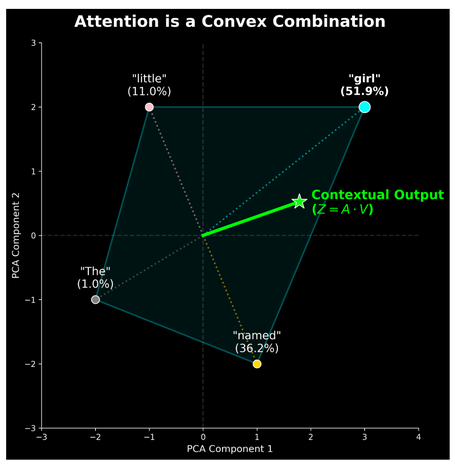
$$Softmax(x_i) = \frac{e^{x_i}}{\sum_j e^{x_j}}$$

5. Attention — Gets the meaning that each of the previous words contributes to predicting the next word.

Example Input Prompt: “The little girl named”



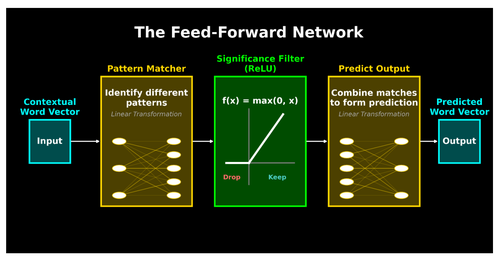




1. Score relevance using **Dot Product** between current word’s **Query** and previous **Keys**.
2. Apply **Softmax** to the scores.
3. Get the **Value** vector of each word.
4. Scale each of these vectors by their scores.
5. Summing up these **Weighted Value** vectors gives us a **Convex Combination.**

(figure on the left)

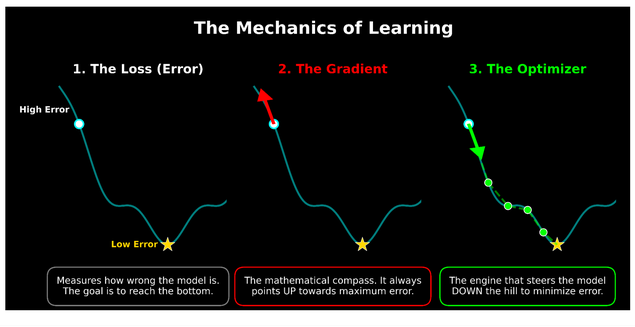
6. Feed-Forward Network (FFN)

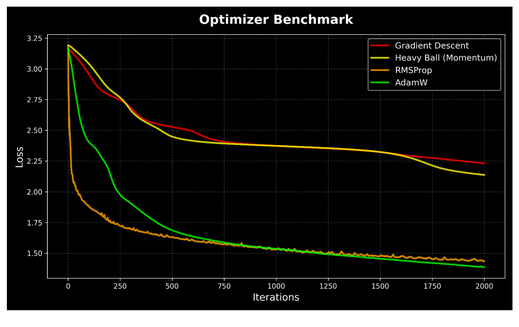


- The FFN learns which word should follow specific combinations of context (**reasoning**).
- We dot product the output vector against the entire embedding matrix and then find the most probable next word with Softmax.

7. Optimizer Benchmark

- Massive FFN layers mathematically drown out tiny attention layers.
- Even Momentum crashes on this size imbalance. AdamW tracks variance, proportionally balancing updates so all layer sizes learn simultaneously.





8. Generated Text (Input prompt of “There was a”)

- **Gradient Descent**: There was a a a . then . . his went for to and there …
- **Heavy Ball**: There was a the big they had the ran . the the the to they …
- **RMSProp**: There was a big tree . it was red . it was lots of fun .
- **AdamW**: There was a bird . the bird had lots of friends and played every day . the bird was happy to see all of his friends so he had a party .