Vector Database [Backbone of RAG]

DataSet: three sentences, each has 3 words (or tokens)

how	are	you

who	are	you

who	am	I

(In practice, a dataset may contain millions or billians of sentences. The max number of tokens may be tens of thousand / billions. For instan Mistral Language model includes 7B parameters.

process

① word Embedding
② Encoding
③ Mean Pooling
④ Indexing

process ①-④

process ①-④

After process (①-④) for all sentences, Now we have Index our dataset in the vector database

⇓

Query: "am I you" → process ①-④ a 2-d query vector

⇓

estimate similarity
Dot Products between query vector and database vectors.
They are all 2-d (transposing the query vector for matrix multiplication)

⇓

Nearest Neighor
Find the largest dot product by linear scan

In practice, because scaning billions of vectors is slow, we use an Approximate Nearest Neighbor (ANN) algorithm like the Hierarchical Navigable Small Worlds (HNSW)

Data: | how | are | you | | who | are | you | | who | am | I |

Query: | am | I | you |

Vocabulary size

Word Embedding

	a	an	the	how	why	who	what	are	is	am	be	was	you	we	I	they	...
↑ #n	0	-1	0	1	0	1	0	0	-1	1	0	0	0	3	1	0	
	2	0	2	0	0	0	-1	1	0	0	0	2	1	0	2	0	
	-1	0	-1	1	2	0	0	1	0	1	-1	0	0	-1	0	3	
	0	1	0	0	1	0	1	0	1	0	1	-2	0	0	0	1	

Data:

1	0	0
0	1	1
1	1	0
0	0	0

1	0	0
0	1	1
0	1	0
0	0	0

1	1	1
0	0	2
0	1	0
0	0	0

Query:

1	1	0
0	2	1
1	0	0
0	0	0

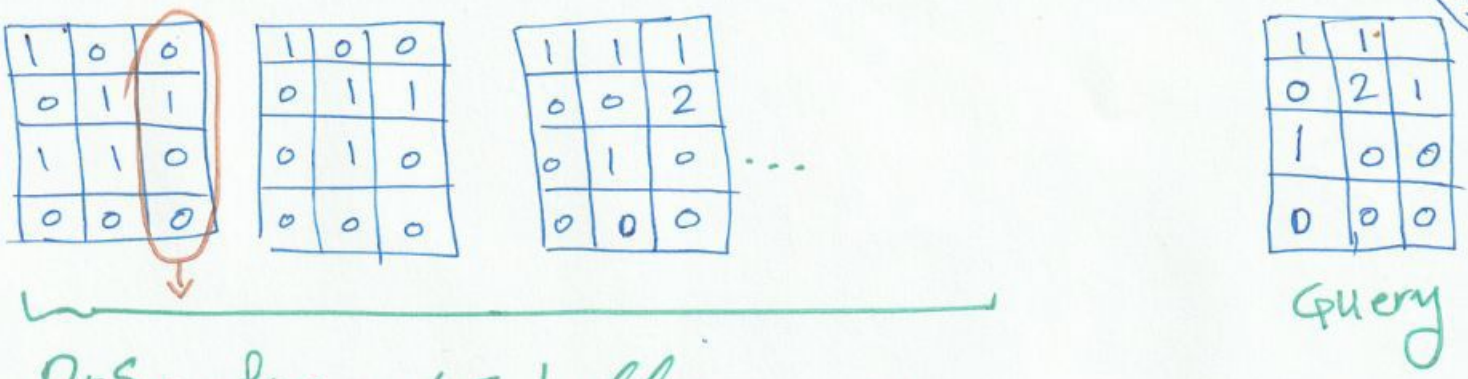
Text Embedding includes ① & ②

① Encoding: Feed the sequence of words embedding to an encoder to obtain a sequence of feature vectors, one per word

② Mean Pooling: Merge the sequence of feature vectors into a vector using "mean pooling" which is to average accross the columns.

---

* In this Example [encoder] is a simple one layer perceptron (linear layer + Relu)
* In practice, the encoder is a transformer or one of its many variants (LLMs)
* The result of text Embedding (or sentence embedding) is a single vector.
* other pooling techniques are: CLS — mean pooling is the most common.



DataSet after word Embedding (page 2)

Query

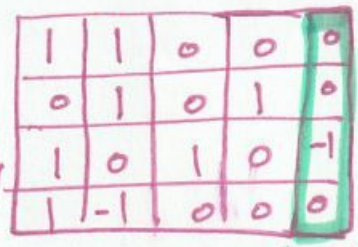
Text Embedding here: a simple one layer perceptron / In practice: LLMs

Transformer-based Architecture

⇓ Encoder

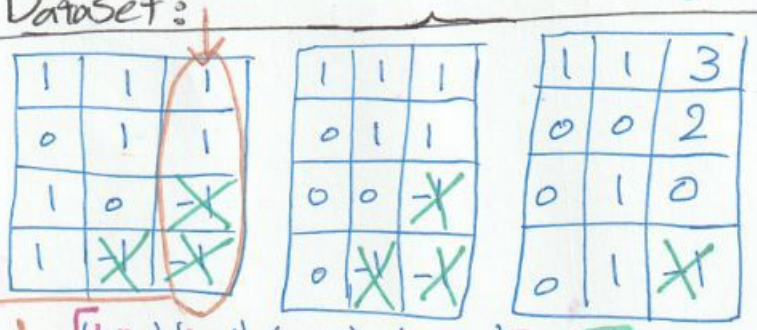
$Z = Wx + b$

$ReLU(Z) = \max\{0, Z\}$

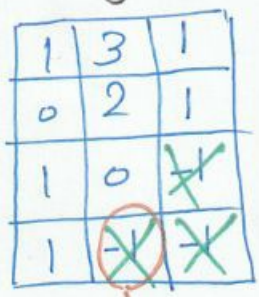


Weight | bias

Dataset:



Query:



$1 = [(1 \times 0) + (1 \times 1) + (0 \times 0) + (0 \times 0)] + 0$

$1 = [(0 \times 0) + (1 \times 1) + (0 \times 0) + (1 \times 0)] + 0$

$-1 = [(1 \times 0) + (0 \times 1) + (1 \times 0) + (0 \times 0)] + -1$

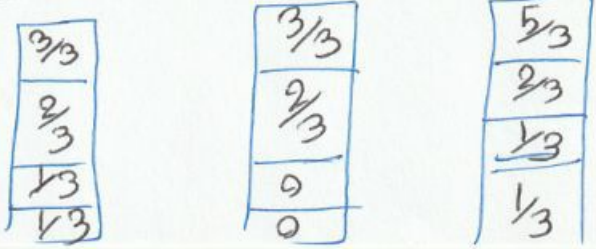
$-1 = [(1 \times 0) + (-1 \times 1) + (0 \times 0) + (0 \times 0)] + 0$

bias

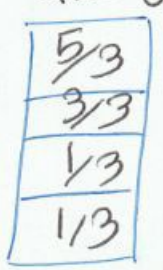
$-1 = [(1 \times 0) + (-1 \times 2) + (0 \times 0) + (0 \times 0)] + 0$

Mean Pooling $\frac{\Sigma}{3}$ over columns

DataSet



Query:



3/3	3/3	5/3
2/3	2/3	2/3
1/3	0	1/3
1/3	0	1/3

Dataset after Mean Pooling & Encoding

Text embedding of DataSet

5/3
3/3
1/3
1/3

Query

Indexing ⇓

projection

1	1	0	0
0	0	1	1

Reducing the text embedding vector dimensions using projection matrix

a short representation of DataSet Sentences which allows faster comparison & retrival

PXV $\begin{cases} \frac{3}{3}+\frac{2}{3} \\ \frac{1}{3}+\frac{1}{3} \end{cases}$

vector Storage

5/3	5/3	7/3
2/3	0	2/3

PXV $\begin{cases} \frac{5}{3}+\frac{3}{3} \\ \frac{1}{3}+\frac{1}{3} \end{cases}$

8/3
2/3

T (Transpose)

8/3	2/3

Dot product (Finding similarity)

$\frac{40}{9}+\frac{4}{9}$ $\quad \frac{40}{9}+0 \quad \frac{56}{9}+\frac{4}{9}$

44/9	40/9	60/9

Retrieval

Performing linear scan to find the largest dot product. This process is slow for billions of values: using ANN / HNSW Algorithms