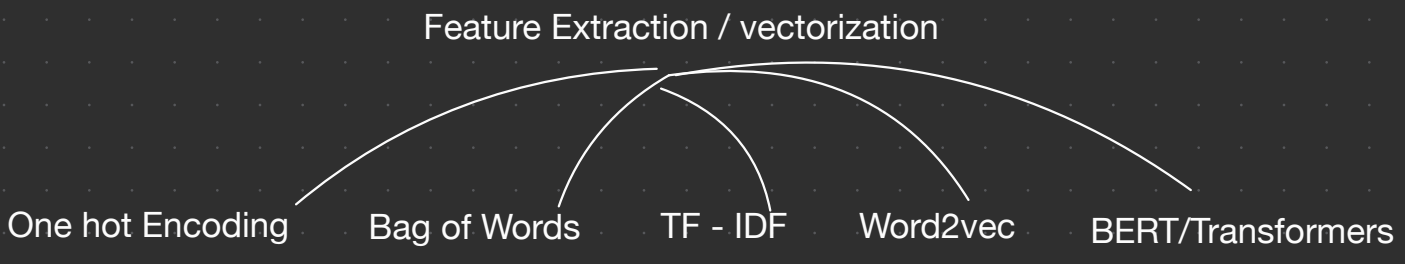


Text is not numbers.
But ML models only understand numbers not feelings, not words, not emojis.
So, we need to convert words into numbers in a smart way that's called feature extraction.

In Machine Learning-based NLP, we mostly use traditional text vectorization techniques like Bag of Words (BoW) and TF-IDF to convert text into numerical format. On the other hand, Deep Learning-based NLP relies on more advanced techniques called embeddings like WorWWord2Vecor example, they treat the words "king" and "queen" as totally unrelated, even though they're semantically connected. On the other hand, Deep Learning-based NLP relies on more advanced techniques called embeddings like Word2Vec, GloVe, and FastText, which convert words into dense vectors that understand relationships and context.

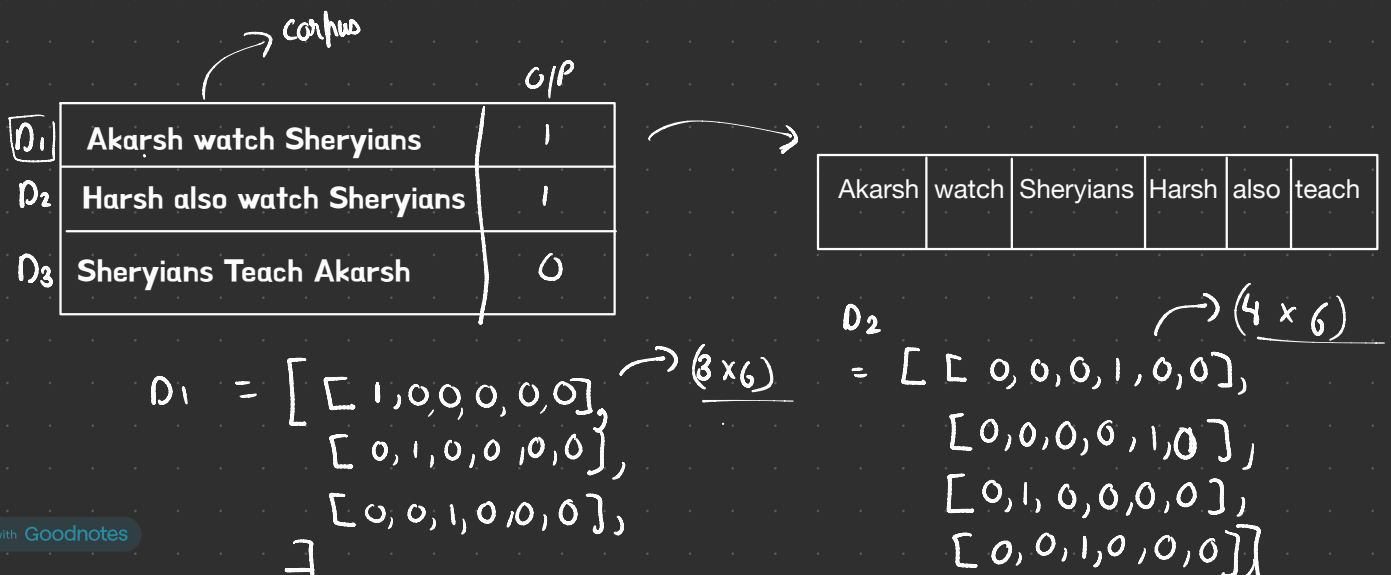
One Hot Encoding

You must know some words before hand.

Corpus - A corpus is just a collection of text that we use in NLP. It can be a bunch of sentenanalyzingts, reviews, or any written content. Think of it like all the text your model will read and learn from. For example, if you're analyzing 100 movie reviews, then those 100 reviews together are called your corpus.

Document - A document in NLP is just a single piece of text inside your dataset. It can be one sentence, one paragraph, or even a full article depends on your project. If your corpus has 100 movie reviews, then each review is called one document.

Vocabulary - A vocabulary in NLP means the list of all unique words present in your corpus. It's like a dictionary your model uses to understand and convert words into numbers. For example, if your dataset has the words "I love pizza" and "I love pasta", then the vocabulary will be: ["I", "love", "pizza", "pasta"]



Pros

→ It is intuitive

→ easy to implement

Cons

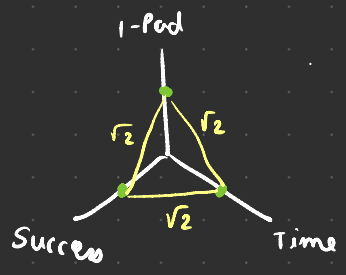
1) Sparsity - too many zero -

2) OOV (Out of vocabulary)

3) Size Difference -

4) Semantic meaning

	Time	Success	1-Pad
Time	1	0	0
Success	0	1	0
1-Pad	0	0	1



Bag of Words

Bag of Words is a technique used to convert text into numbers so that machine learning models can understand it. It creates a list of all the unique words in your dataset (called vocabulary), and then for each sentence, it counts how many times each word appears. It doesn't care about grammar or the order of words only the word frequency matters. That's why it's called a "bag" jumbled, unordered collection of words.

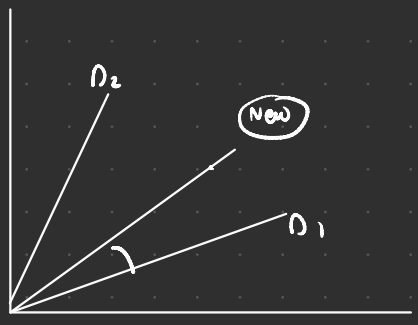
→ Corpus

		O/P
D1	Akarsh watch Sheryians	1
D2	Harsh also watch Sheryians	1
D3	Sheryians Teach sheryians	0

D4 Sheryians is cool

MF = 1

	Akarsh	watch	Sheryians	Harsh	also	teach
D1	1	1	1	0	0	0
D2	0	1	1	1	1	0
D3	0	0	2	0	0	1



Pros

1) Easy to implement
2) Fixed size
3) Work with ML
4) Fast & efficient

Cons

1) Sparse Matrix
2) OOV (Out of vocabulary)
3) Semantic meaning (Better than OHE)
4) Out of order

N-grams

		O/P
D_1	Akarsh watch Sheryians	1
D_2	Harsh also watch Sheryians	1
D_3	Sheryians Teach Akarsh	0

Unigram

Akarsh	watch	Sheryians	Harsh	also	teach

bigram

Akarsh watch, watch Sheryians, harsh also, Sheryians teach, Teach Akarsh

$D_1 \rightarrow$	1	1	0	0	0
$D_2 \rightarrow$	0	1	1	0	0
D_3	0	0	0	1	1

Trigram

Akarsh watch Sheryians, harsh Also watch, watch also sheryians,

D_1 Cricket is very good

D_2 Cricket is not good

Vocab → cricket, is, very, good, not

$D_1 \rightarrow$	1	1	1	1	0
$D_2 \rightarrow$	1	1	0	1	1

5D → 2D are Different

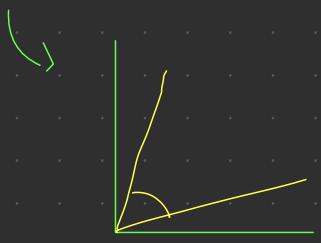
3D are Same

bigram → Cricket is | is very | very good | is not | not good

$D_1 \rightarrow$	1	1	1	0	0
$D_2 \rightarrow$	1	0	0	1	1

→ 5D → 4D are Different

→ 1D is Similar



n gram = can be called as Bag of n grams

Pros	Cons
1) Semantic meaning → A little bit	1) Dimensions increase 2) out of vocabulary

TF-IDF

OHE, BOW, BONGrams

		O/P
D_1	Akarsh watch Sheryians	1
D_2	Harsh also watch Sheryians	1
D_3	Sheryians Teach Akarsh	0

D_I

Akarsh	watch	Sheryians	Harsh	also	teach
$\frac{1}{3} \times \log\left(\frac{3}{2}\right)$ ⇓ 0.7			0	0	0

TF = (Term frequency)

IDF => (Inverse Doc frequency)

TF x IDF

$$TF = \frac{(\text{No. of occurence of Term in Document})}{(\text{Total no. of Terms in Doc})} = \frac{1}{3}, \frac{1}{3}, \frac{1}{3}$$

0, 1

$$IDF = \log_e \frac{(\text{Total no. of Doc in corpus})}{(\text{Number of doc. with Term in them})} = \log_e\left(\frac{3}{2}\right) \Rightarrow \log\left(\frac{3}{1}\right)$$

$$\log_e\left(\frac{3}{3}\right) = 0$$

TF X IDF

Pros

→ Information Retrival
Google search

Cons

1) Sparsity
2) OOV
3) dimensions