PHASE-2

TEXT - TO - VECTORS

Text → Vectors means converting words or sentences into numerical form so, ML/DL understands them.

ONE HOT ENCODING :-

One hot Encoding is a technique to convert categorical text data into numerical vectors for ML models.

What it does :-

→ Each category becomes a binary vector. (0s and 1s)

Example :-

Categories :-

['Red', 'Blue', 'Green']

One-hot vectors :-

Red → [1, 0, 0]

Blue → [0, 1, 0]

Green → [0, 0, 1]

Note :- In NLP, we use $\boxed{\text{Unique Vocabulary}}$.

Unique Vocabulary means the set of distinct words present in a text or corpus.

Example :-

	Text
D1	The food is good
D2	The food is bad

Give, Text, D1 and D2, are the reviews of coustomers here we use unique vocabulary to find Vectors for our NLP model.

So, we will create a unique Vocabulary.

The	food	is	good	bad
1	0	0	0	0
0	1	0	0	0
0	0	1	0	0
0	0	0	1	0
0	0	0	0	1

which means,

The → [1 , 0 , 0 , 0 , 0]
food → [0 , 1 , 0 , 0 , 0
is → [0 , 0 , 1 , 0 , 0]
good → [0 , 0 , 0 , 1 , 0]
bad → [0 , 0 , 0 , 0 , 1]

For D1, The food is good
Our vectors are,

D1 [[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 1, 0]]

For D2, The food is bad
Our vectors are,

D2 [[1, 0, 0, 0, 0]
[0, 1, 0, 0, 0]
[0, 0, 1, 0, 0]
[0, 0, 0, 0, 1]]

ONE-HOT ENCODING

Advantage

→ Easy to implement with Python
[Sklearn, onehotEncoder, pd.get_dummies]

Disadvantages

→ Sparse matrix - Overfitting
(matrix, which most values 0)
[1, 0, 0, 0, 0]

→ ML Aglo. → Fixed size I/P
all i/p arr must be same size

→ No semantic meaning is getting captured
(X does not know the word's meaning,
X different words can have sam meaning)

→ Out of vocabulary
(If new i/p is given its not in vocabulary)

BAG OF WORDS :- (BoW)

Bad of Words (BoW) is a simple text -> vector technique that represents text by word frequency, ignoring grammar and word order.

How BoW works :-

=> Collect Unique vocabulary
=> Collect word occurences
=> Create fixed length vector.

Example :-

Step 1

Text

HE is a good boy
She is a good girl
Boy and girl are good

lower all the words ⟹ stopwords

S1 -> good boy
S2 -> good girl
S3 -> Boy girl good

Note :- Here, we are applying .lower() and after removing stopwords we are getting new S1, S2 and S3

what is happening. { .lower() -> Boy -> boy ; stowards -> is, a, and got removed }

Step-2 =>

Now we will be counting frequency of unique words. {vocabulary}

	Vocabulary	Frequency	
Vocabulary size is 3	good	3	max
	boy	2	
	girl	2	min

Note → Frequency of unique words {vocabulary} is always in decending order, so we can also say that, the word which came maximum times, appeard on top.

Step 3 =>

Vocabulary	Frequency
good	3
boy	2
girl	2

$$\Longrightarrow \begin{array}{cccc} & good & boy & girl \\ S1 & [1 & 1 & 0] \\ S2 & [1 & 0 & 1] \\ S3 & [1 & 1 & 1] \end{array}$$

Creating ~~of~~ Vector of S1, S2, and S3 using vocabulary or Unique vocabulary

Binary BoW	BoW
→ { 1s and 0s }	{ count will get updated based on frequency }
→ Example : "I love love Python"	→ Example :- "I love love Python"
created Vocabulary ["I", "Love", "Python"]	vocabulary ['I', 'Love', 'Python']
Vector will Be [1, 1, 1]	Vector will be [1, 2, 1]
	'2' because of count of 'love'

BAG OF Word (BOW)

Advantages	Disadvantages
-> Simple and Intutive	-> Sparse matrix or array
-> Fixed Size Input doesnt matter because it can handle any size of sentence.	-> Ordering of word is getting changed
	-> Out of vocabulary (OOV)
	-> Semantic meaning is still not captured.

TF-IDF [TERM FREQUENCY - INVERSE DOCUMENT FREQUENCY]

- TF-IDF tells which words are important in a document.

Taking previous Example

S1 → good boy
S2 → good girl
S3 → ~~good~~ boy girl good

Term Frequency :-
How many time a word appears in one document.

$$\text{Term Frequency (TF)} = \frac{\text{No. of rep of words in sentence}}{\text{No. of words in sentence}}$$

So, we are calculating TF for S1, S2 and S3

	S1	S2	S3
good	1/2	1/2	1/3
boy	1/2	0	1/3
girl	0	1/2	1/3

IDF (Inverse Document frequency) :-

IDF tells how important a word is, across all document

- Rare word = More Important
- Common word = Less Important

$$IDF = \log_e\left(\frac{\text{No. of Sentences}}{\text{No. of Sentences containing the word}}\right)$$

Now, calculating IDF for S1, S2 and S3

Words	IDF
good	$\log_e(3/3) = 0$ #
boy	$\log_e(3/2)$
girl	$\log_e(3/2)$

(S1, S2, S3)
no. of sentence good is present in all senteses

** Note :- To find TF-IDF we basically mulply the

$$\boxed{TF\text{-}IDF = TF * IDF}$$

Term-frequency (TF)

	S1	S2	S3
good	1/2	1/2	1/3
boy	1/2	0	1/3
girl	0	1/2	1/3

×

IDF

words	IDF
good	$\log_e(3/3) = 0$
boy	$\log_e(3/2)$
girl	$\log_e(3/2)$

Final TF-IDF

TF-IDF = TF * IDF

	good	boy	girl
Sent 1	$\boxed{0}$	$1/2 * \log_e(3/2)$	0
Sent 2	0	0	$1/2 * \log_e(3/2)$
Sent 3	0	$1/3 * \log_e(3/2)$	$1/3 * \log_e(3/2)$

Here, Mutiplying Sentence 1 (S1)'s TF
with IDF value

good
boy * good
girl boy
girl

which means

=> S1 (good) * IDF (good)
=> 1/2 * 0
=> => 0

TF-IDF

Advantages	Disadvantages
-> Intutive	-> Sparsity still exists
-> Fixed size -> Vocab size	-> Out of vocabulary (OOV)
-> Word Imp is getting captured.	