PHASE - 3

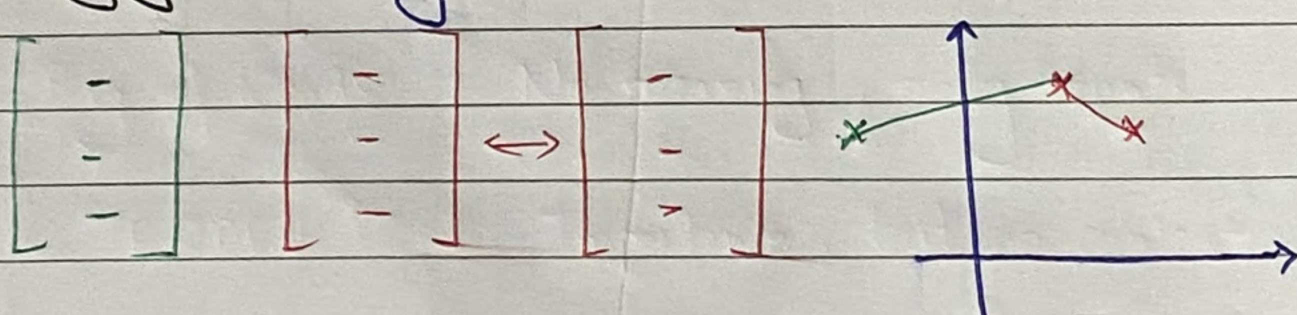
WORD EMBEDDING

Word Embedding means converting words into numbers that keep their meaning.

which means,
Computer dont understand words, but they understand distance b/w number, so

→ words with similar meanings → numbers close together.
→ word with different meanings → numbers far apart

Angry Happy Excited



x → vectors of 'Happy' and 'Excited' word which says they are near to each other more likely to be similar.
x → vector of 'Angry' is far, means it has different meani

Word Embedding Techniques

- Count or Frequency
  1. One Hot Encoding
  2. BoW (Bag of Words)
  3. TF - IDF
- Deep Learning Trained Model
  - Word 2 Vec
    - CBOW [Continuous Bag of Words]
    - Skipgram

WORD2VEC

Word2Vec is a neural-network based technique that learn word association from large corpus of text.

or

Word2Vec is a method that turns words into numbers (vectors) by looking at words around them. it helps computer understand that similar words have similar meanings.

Vocabulary → unique words of Corpus

Important :-

Problem with older methods

Methods	Problems
Bag of Words	No meaning, only counts
TF-IDF	Importance, but no context
One-Hot Encoding	Huge Sparse vector, no similarities

Example :-

King ≠ Queen ≠ Prince

All totally unrelated in BoW / TF-IDF

But,

This does not mean they are unrelated in Word2Vec

It means :

They are different words so they have different vectors,

In code terms :

vec_King != vec_queen

vec_queen != Vec_prince

Each word has its own <u>numeric</u> <u>vector</u> but distance between them matters.

Word vectors are <u>Numbers</u> :

Example :-

King : [0.21, 0.34, -0.12, 0.56, 0.78]

Queen : [0.20, 0.36, -0.10, 0.55, 0.80]

Prince : [0.19, 0.30, -0.15, 0.50, 0.70]

→ Different Vectors

→ but King and Queen are closer than King and Apple.

Why King - Man + Woman ≈ Queen works

because

Word 2 Vec learns relationships not just words

King = Man + Royal
Queen = Woman + Royal

So,

$$\boxed{\text{King} - \text{Man} + \text{Woman} \approx \text{Queen}}$$

How Word2Vec finds 'Queen' from that

Step 1 - Compute the new vector
Step 2 - Compare it with all word vectors
Step 3 - Find the closest vector using cosine similarity

Word2Vec

CBOW
[Continuous Bag of Words]

Skipgram

CBOW (Continuous Bag of Words) :-

CBOW is a Word2Vec model that Predicts a missing word using the surrounding words. using window size.

(Surrounding words → Guess the middle word.)

[Context → target]

Example :-

[My Company is related to] Data Science

$\boxed{\text{Window size} = 5}$

I/P	O/P
[My, company, related, to]	Is
Context	target
[Company, is, to, data]	related

It use one-hot encoding in hidden layer

SKIPGRAM

Skipgram is a word2Vec model that predicts the surrounding context words using target (center) word.

Target -> Context

Example :-

[My company $\boxed{\text{is}}$ related to] data Science.

window size = 5

I/P	O/P
is target	My, Company, related, to Content

When should we apply CBOW or SkipGram

{ Small Dataset → CBOW }
{ Huge Dataset → Skipgram }

How to improve :-

CBoW or Skipgram

→ Increase the training data
→ Increase the window size → vector dimension is also increases.

Average Word2Vec

technique used to represent a sentence or document by taking the avg. of Word2vec vectors of all words in it,

Example :-

Sentence - I love deep learning

Word Vectors :-

I → V1

love → V2

deep → V3

learing → V4

Formula :- If a sentence has 'n' words

$$\text{Sentence Vector} = \frac{V_1 + V_2 + V_3 \cdots V_n}{n}$$

Code :-

```
'''  bash
        pip install gensim
'''

Import gensim
from gensim.models import Word2Vec, KeyedVectors

import gensim.downloader as api

wv = api.load('word2vec-google-news-300')

vec_King = wv['King']

print (vec_King)

print (vec_King.shape) # shape of vector (300,)

wv.most_similar('cricket')
                         # [('cricketing', 0.83722),
                            (cricketers, 0.8165),
                            (Test_cricket, 0.8094)]
wv.most_similar('happy')
                         [('glad', 0.74)
                          ('pleased', 0.663)
                          ('Overjoyed', 0.662)]
```

```
wv.similarity ('hockey', 'sports')
                          # 0.5354

vec = wv['King'] - wv['Man'] + wv['woman']

print(vec)

wv.most_similar([vec])
                  # ('King', 0.8449),
                    ('queen', 0.7300)
                    ('monarch', 0.64)
```

BERT

BERT is a language model that understands the meaning of words by looking at the whole sentence.

which means, The same word can have different meanings in different sentences.

→

It creates context-based word vectors

For Example :-

S1 = "I went to the bank to deposit money"

S2 = "I sat on the bank of the river"

Bert create two different vectors for same word 'bank',

for 'bank' is S1 vector might be - 1108

and for 'bank' in S2 vector might be - 5029

Why BERT is powerfull?

→ Use Transformers

→ Use attention to focus on important words

→ Better than Word2Vec because it understand context.