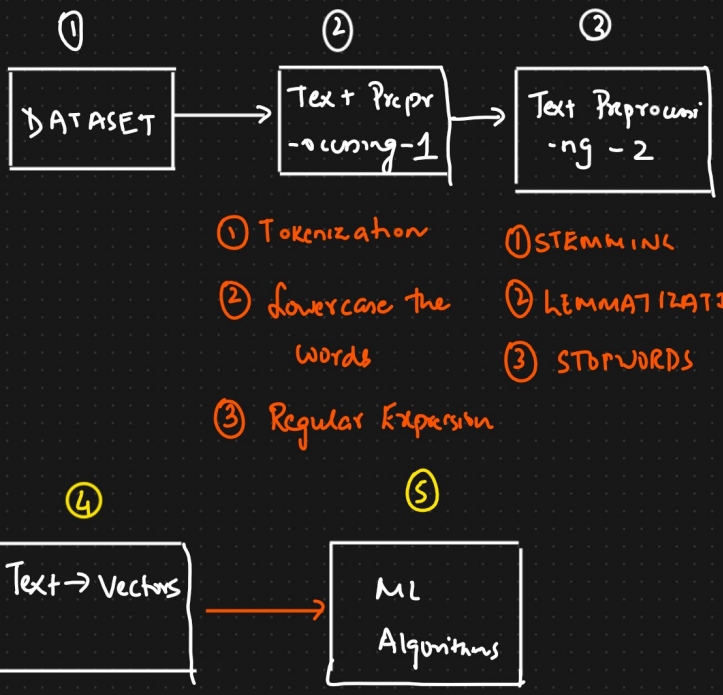
Text Preprocessing → What We have learnt?

Sentiment Analysis

	Text	O/P
D1	The food is good	1
D2	The food is bad	0
D3	Pizza is Amazing	1
D4	Burger is bad	0



D1 → | 1 | 0 | 1 | 1 | → 1

① One Hot Encoded
② Bag of Words (BOW)
③ TF-IDF
④ Word2Vec
⑤ Avg Word2Vec

Gensim

① One Hot Encoding

	Text	O/P
D1	The food is good	1
D2	The food is bad	0
D3	Pizza is Amazing	1

Test [Burger is bad]

Vocabulary size: 7

Vocabulary {unique words}

The	food	is	good	bad	Pizza	Amazing
1	0	0	0	0	0	0
0	1	0	0	0	0	0

D1 4x7
[[1 0 0 0 0 0 0],
[0 1 0 0 0 0 0],
[0 0 1 0 0 0 0],

D2 4x7.
[[1 0 0 0 0 0 0],
[0 1 0 0 0 0 0],
[0 0 1 0 0 0 0],

D3 3x7
[[0 0 0 0 0 1 0],
[0 0 1 0 0 0 0],
[0 0 0 0 0 0 1]]

[0 0 0 1 0 0 0]] [0 0 0 0 1 0 0]]

{50K vocabulary size}

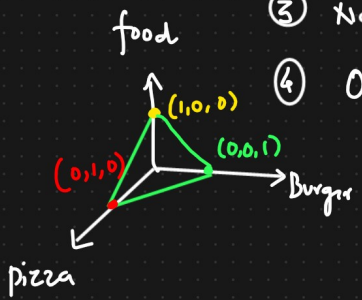
Advantages

① Easy to implement with python
[sklearn OneHotEncoder, pd.get_dummies()]

Disadvantages

① Sparse matrix → Overfitting
② ML Algorithm → Fixed size I/p
③ No semantic meaning is getting captured
④ Out of Vocabulary (OOV).

food	pizza	burger
1	0	0
0	1	0
0	0	1



② Bag of Words

Dataset-

Text	O/P
He is a good boy	1
She is a good girl	1
Boy and girl are good	1

Lower all the words case → Stopwords

S1 → good boy
S2 → good girl good
S3 → Boy girl good school

Test

Vocabulary	frequency
good	3
boy	2
girl	2

	[good	boy	girl]	O/p
S1	[1	1	0]	1
S2	[1	0	1]	1
S3	[1	1	1]	1

Binary BOW and BOW

{1 and 0} {Count will get updated based on frequency}

Advantages

① Simple and Intuitive

② Fixed Sized I/p → ML Algorithms

Disadvantages

① Sparse matrix or array → Overfitting

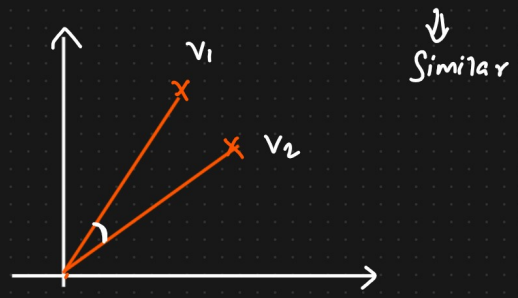
② Ordering of the word is getting changed

③ Out of Vocabulary (OOV).

④ Semantic meaning is still not captured.

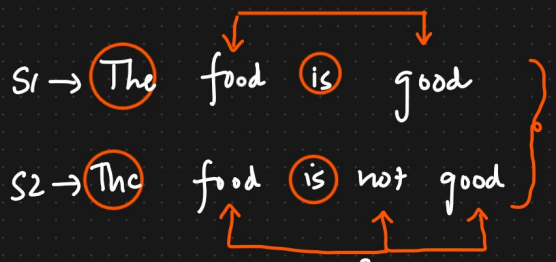
The food is good → [1 1 1 0 1] → v_1

The food is not good → [1 1 1 1 1] → v_2



④ N-grams

Eg: bigrams, trigrams



	food	not	good
→	1	0	1
→	1	1	1

	[food	not	good	food good	food not	not good]
S1	1	0	1	1	0	0
S2	1	1	1	0	1	1

Sklearn → n-grams = (1, 1) → unigrams

= (1, 2) → unigram, bigram

= (1, 3) → unigram, bigram, trigram

= (2, 3) → Bigram, trigram.

④ TF-IDF [Term Frequency – Inverse Document Frequency]

S1 → good boy

S2 → good girl

S3 → boy girl good

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

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

Term Frequency * IDF

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

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

Final TF-IDF

	[good	boy	girl]
Sent 1	0	$\frac{1}{2} \times \log_e(3/2)$	0
Sent 2	0	0	$\frac{1}{2}\log_e(3/2)$
Sent 3	0	$\frac{1}{3}\log_e(3/2)$	$\frac{1}{3}\log_e(3/2)$

Op

Bow

1	1	0
1	0	1
1	1	1

Advantages

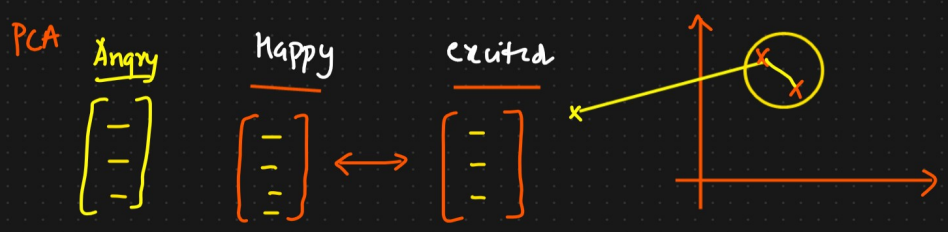
1. Intuitive
2. Fixed Size → Vocab size
3. Word Importance is getting captured

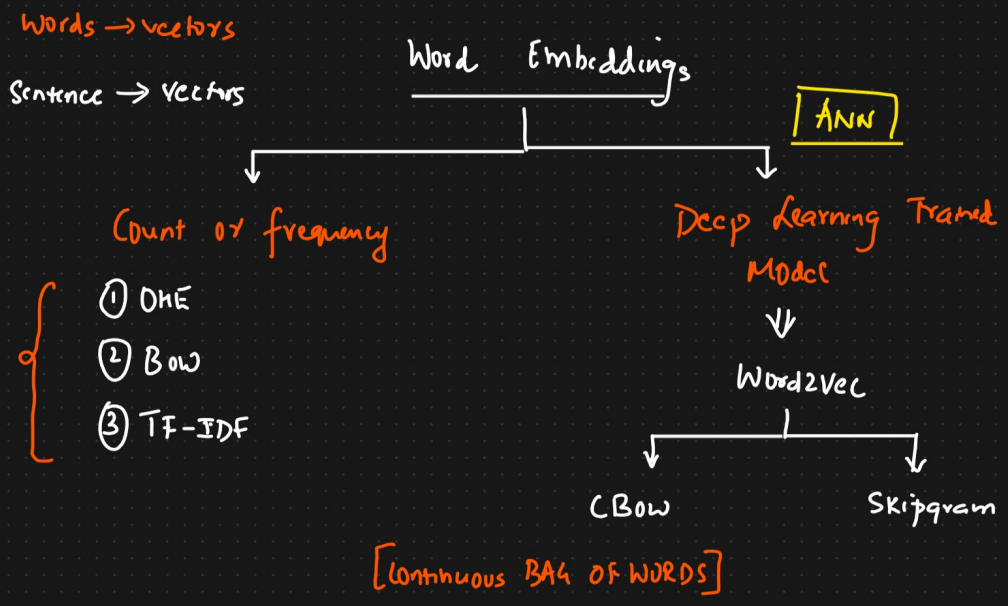
Disadvantages

1. Sparsity still exists
2. OOV

Word Embeddings [Wikipedia]

In natural language processing (NLP), word embedding is a term used for the representation of words for text analysis, typically in the form of a real-valued vector that encodes the meaning of the word such that the words that are closer in the vector space are expected to be similar in meaning.





Word2Vec → Feature Representation

Google

Word2vec is a technique for natural language processing published in 2013. The word2vec algorithm uses a neural network model to learn word associations from a large corpus of text. Once trained, such a model can detect synonymous words or suggest additional words for a partial sentence. As the name implies, word2vec represents each distinct word with a particular list of numbers called a vector.

Vocabulary → Unique words → Corpus.

Feature Representation	Boy	Girl	KING	QUEEN	Apple	Mango
Gender	-1	1	-0.92	+0.93	0.01	0.05
Royal	0.01	0.02	0.95	0.96	-0.02	0.02
Age	0.03	0.02	0.75	0.68	0.95	0.96
Food	-	-	-	-	0.91	0.92
					-	-

300 dimension !

$$[\text{KING} - \text{BOY} + \text{QUEEN} = \text{GIRL}]$$

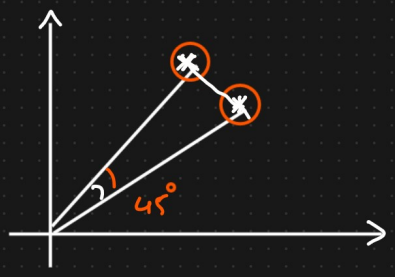
KING [0.95, 0.96] Man [0.95 0.98]

QUEEN [-0.96, 0.95] Women [-0.94, -0.96]

KING - MAN + QUEEN = WOMEN

Cosine Similarity

Distance = 1 − Cosine Similarity

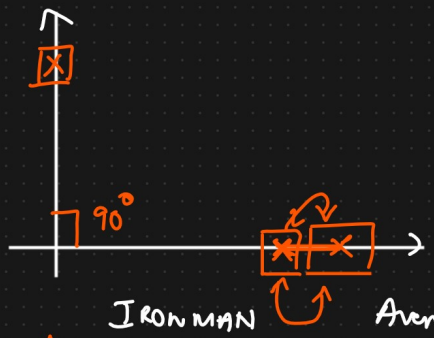


Cosine-Sim = $\cos 45 = \frac{1}{\sqrt{2}} = 0.7071$

Distance = 1 − 0.7071

↳ = 0.29

1 − 1 = 0



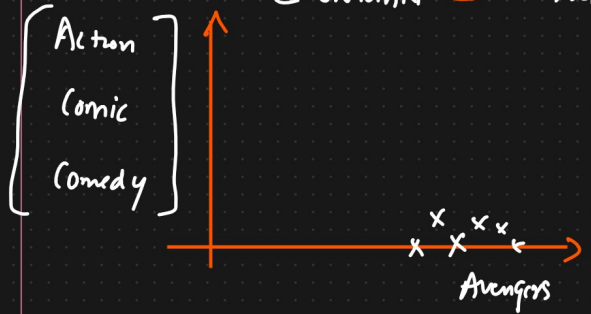
Distance = 1 − 0

= 1

Distance = 1 − cos0

= 1 − 1

= 0//.



ANN, Loss, Optimizers

(#) Word2Vec → CBOW [Continuous Bag of Words]

→ Skipgram.

Pretrained Model

Train A Model For Scratch

① CBOW [Continuous Bag of Words]

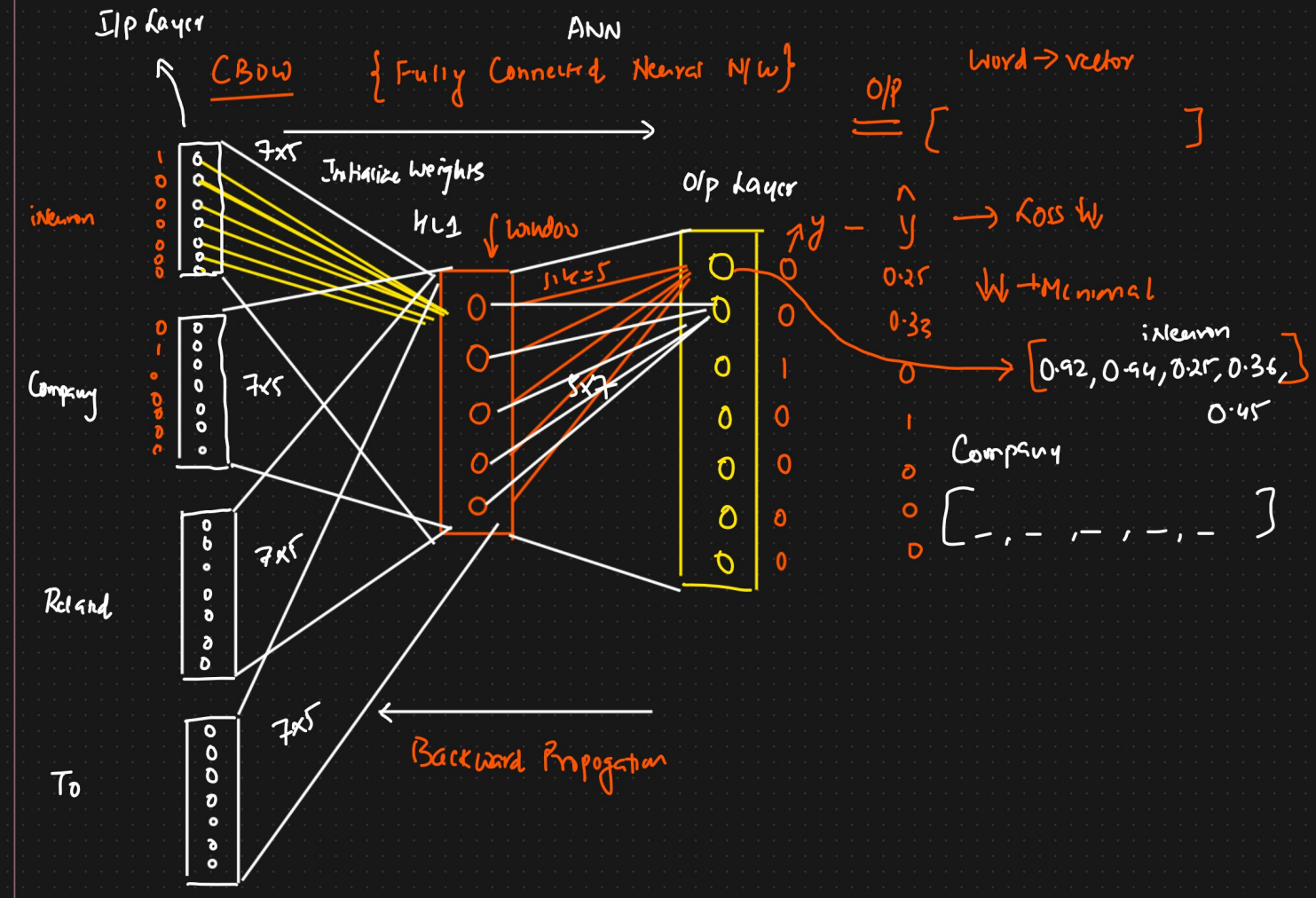
CORPUS ÷ DATASET

[iNeuron Company IS Related To DATA SCIENCE] → Vocabulary

300 dimension — 300 ← Window size = 5 ⇒ Feature Representation [_ _ ___] — OHE

I/P	O/P
→ [iNeuron, Company, Related, To]	IS
→ Company, IS, To, DATA	Related
→ Is, Related, DATA, SCIENCE	To

iNeuron	[1 0 0 0 0 0 0]
Company	[0 1 0 0 0 0 0]
Related	[0 0 0 1 0 0 0]
to	0 0 0 0 1 0 0



② Skipgram – Word2Vec

Window size=5

O/p

→ [Neuron, Company, Related, To]

→ Company, Is, To, DATA

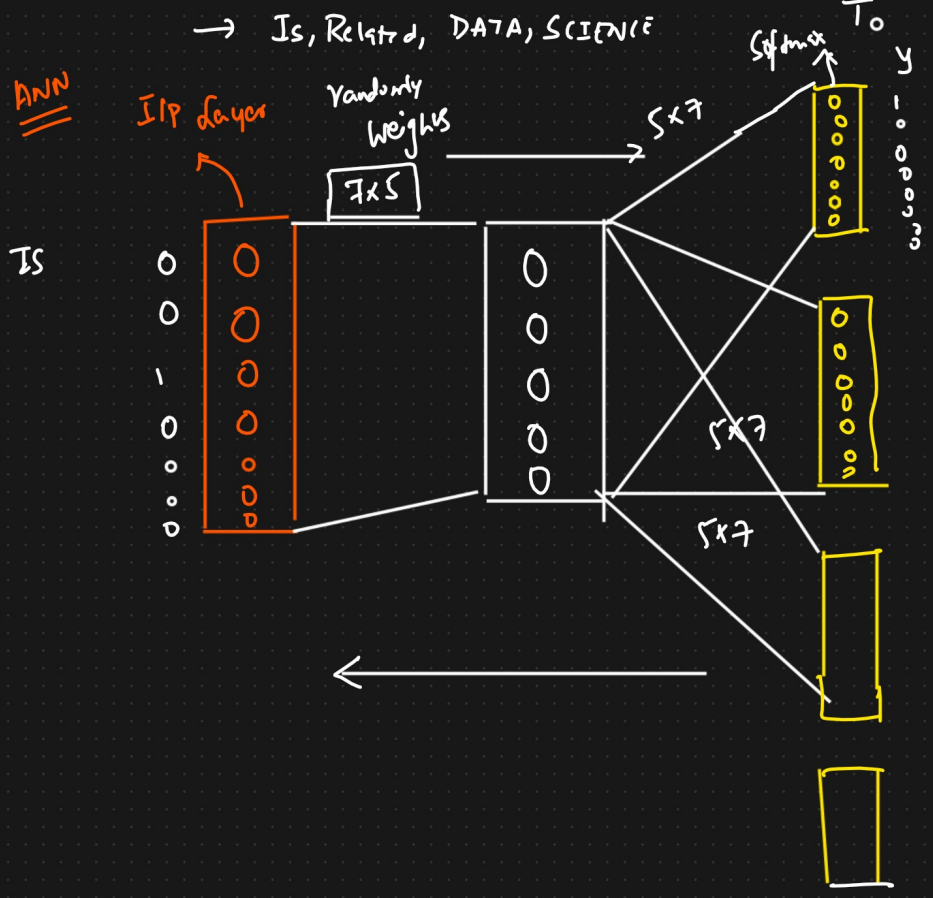
→ Is, Related, DATA, SCIENCE

I/p

Is

Related

To



Improve CBow or Skipgram

① Increasing the Training Data

② Increase the window size - vector dimension is also increasing.

When Should we apply CBow or SkipGram.

- Small Dataset → CBow
- Huge Dataset → SkipGram

Google Word2vec

3 billion words → Google News

feature representation of 300 dimension vectors]

Cricket → [- - - - - - - - - - - - - -].

Gensim

Advantages of Word2Vec

*) Sparse Matrix ⟶ Dense Matrix

*) Semantic Info is getting captured [Honest, good]

*) Vocabulary Size ⟶ Fixed set of dimension vectors
Google Word2Vec [300 dimension]

*) OOV is also solved

⑧ Avg Word2Vec

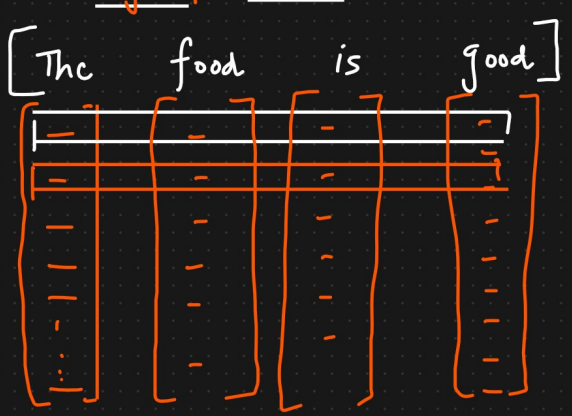
	Text	O/P
D1	The food is good	1
D2	The food is bad	0
D3	Pizza is Amazing	1

Avg word2vec

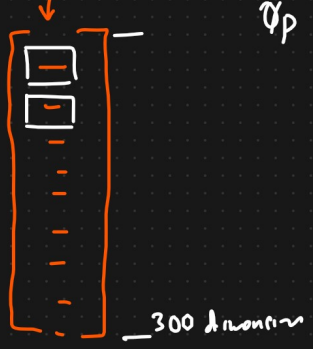
[. . . . . .]
[ ]
[ ]

O/P
1

Google pretrained Word2Vec Model



Avg Word2Vec



⑧ Gensim, Glove

↳ pretrained Google Word2Vec

↳ Train a word2vec from scratch