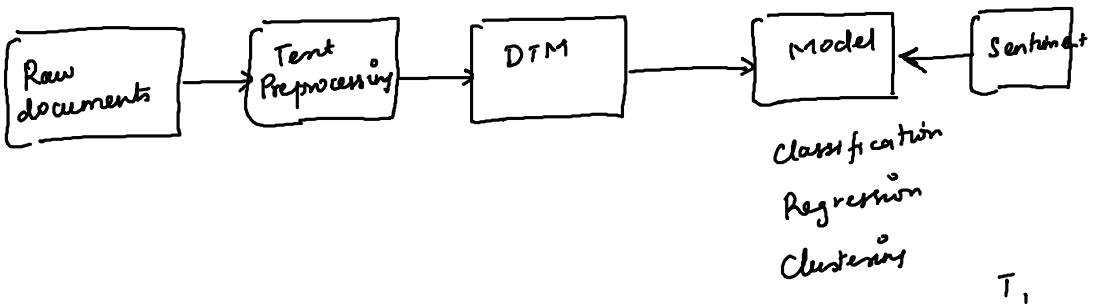



```
<800x2479 sparse matrix of type '<class 'numpy.int64'>'
        with 52937 stored elements in Compressed Sparse Row format>
```

800 × 2479

	T_1	T_2	T_3	T_4	T_5
D1	1	0	0	1	0
D2	0	0	0	1	0
D3	0	0	0	1	0
D4	0	0	1	1	0

(0,0) → 1 . . . (3,4) → 5
(0,2) → 1

```
<40000x25748 sparse matrix of type '<class 'numpy.int64'>'
        with 3230445 stored elements in Compressed Sparse Row format>
```

$$\text{F1 score} = 2 \times \frac{\text{precision} * \text{recall}}{\text{precision} + \text{recall}}$$

$$\text{precision} = \frac{TP}{TP+FP} \qquad \text{recall} = \frac{TP}{TP+FN}$$

Actual	Predicted	Prediction Type	
positive	negative	FN	→ Type II
positive	positive	TP	
negative	positive	FP	→ Type I.

Covid diagnosis

→ positive => has covid ✓
→ negative = does not covid. ✓

Sentiment Analysis – VADER

Valence Aware Dictionary & sEntiment Reasoner

"I hate tea" → "negative" (Label/Sentiment) & 100% (Sentiment score)

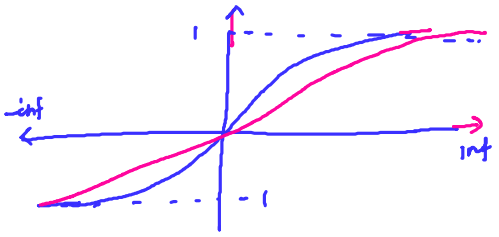
→ ~~I~~ hate tea ⇒ −2.7 ⇒ Total score

hate	tea
Neg	Neu
−2.7	0

→ ~~I~~ hate tea the taste is bad ⇒ −5.2 ⇒ Total score

hate	tea	the	taste	is	bad
−2.7	0	0	0	0	−2.5

Total score (−inf to +inf) → Normalize → −1 to +1 (compound score)



$$\text{Compound score} = \frac{\text{Total score}}{\sqrt{\text{Total score}^2 + \text{alpha}}}, \quad \text{alpha} = 15$$

Total score = −2.7

$$\text{Compound score} = \frac{-2.7}{\sqrt{(-2.7)^2 + 15}} =$$

~~I~~ hate tea and ~~I~~ love coffee

hate	tea	and	love	coffee
(−2.7)	0	0	3.2	0

Total score = 0.5

~~I~~ hate tea

Neg	Neu
−2.7	0
abs(−2.7)+1	(0)+1
3.7	1

⇒ Total score ⇒ 4.7

% of pos score = 0/4.7 = 0%

% of neg score = 3.7/4.7 = 0.78

% of neu score = 1/4.7 = 0.22

	~~I~~	hate	tea	and	~~I~~	love	coffee
Original Score		-2.7	0	0		3.2	0
Abs Score		2.7+1	0+1	0+1		3.2+1	0+1
Final Score		3.7	1	1		4.2	1

% of pos score = 4.2 / 10.9 = 0.385

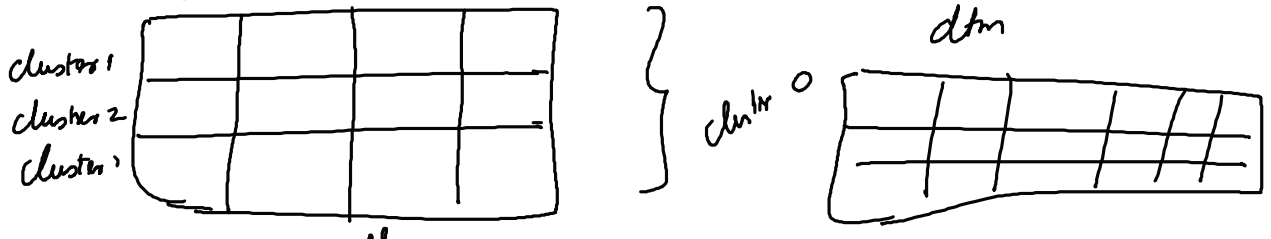
% of neg score = 3.7 / 10.9 = 0.339

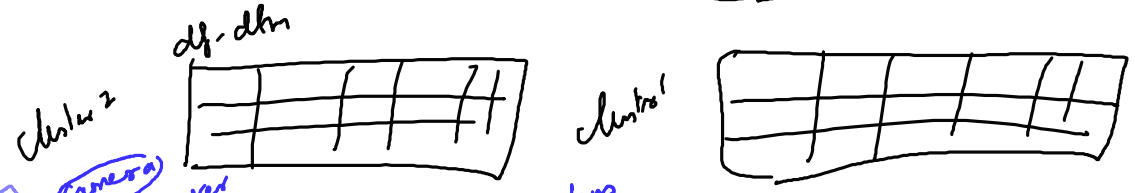
% of neu score = 3 / 10.9 = 0.275

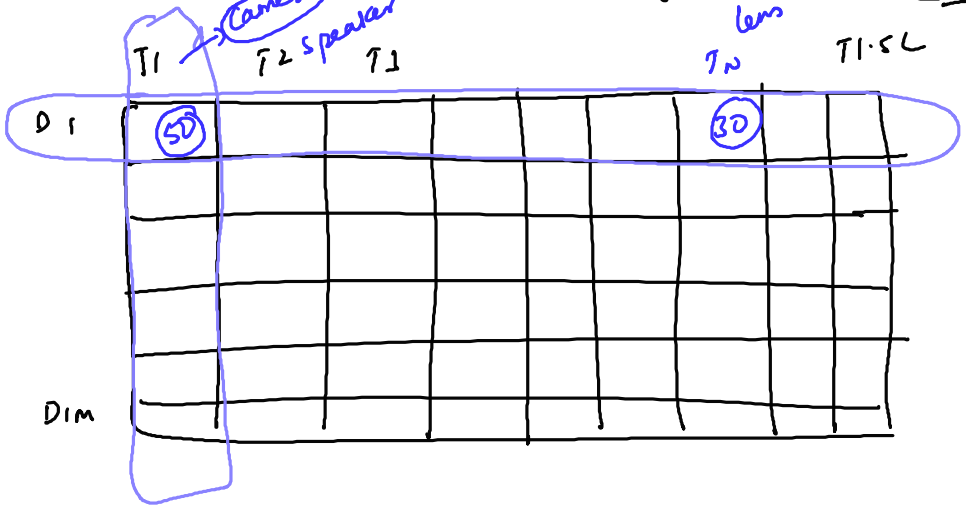
Document clustering

Unsupervised

- Clustering algorithms (K-means)
  - Average balance
- Topic modelling (LSA, LDA)







1M x 1.5L

Document Vector dimension :- No. of columns in DTM ≈ Vocab size

Word vector dimension : No. of rows in DTM ≈ No of document

Word2 vec

Shallow neural network — Relationship b/w words

Doc1 The credit card is not having enough benefit
Doc2 Credit card penalty is very high
...
Doc 1000 I missed my credit card bill this month. Please help.

Bigram

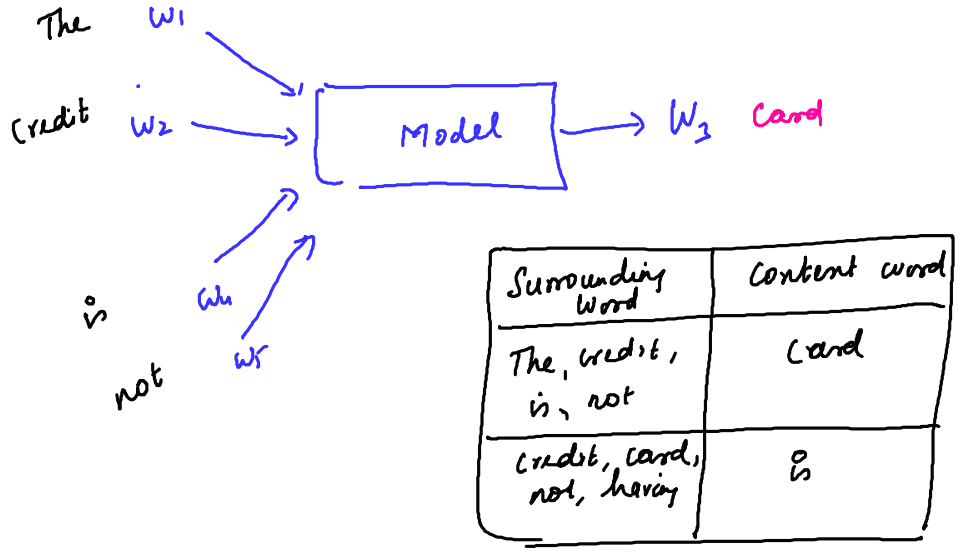
I/p Input word	target Content word
The	Credit
Credit	Card
Card	is
is	not
not	having

Predict the next word

Trigram

2 input word	Content word
The, Credit	Card
Credit, Card	is
Card, is	not
is, not	having
Credit, Card	penalty

The credit card is not having enough benefit



Surrounding Word	Content word
The, credit, is, not	Card
credit, card, not, having	is

Word2 vec

Sparse, high dimension, less content vector representation → Word2vec dense, low dimension, content preserved vector representation of words

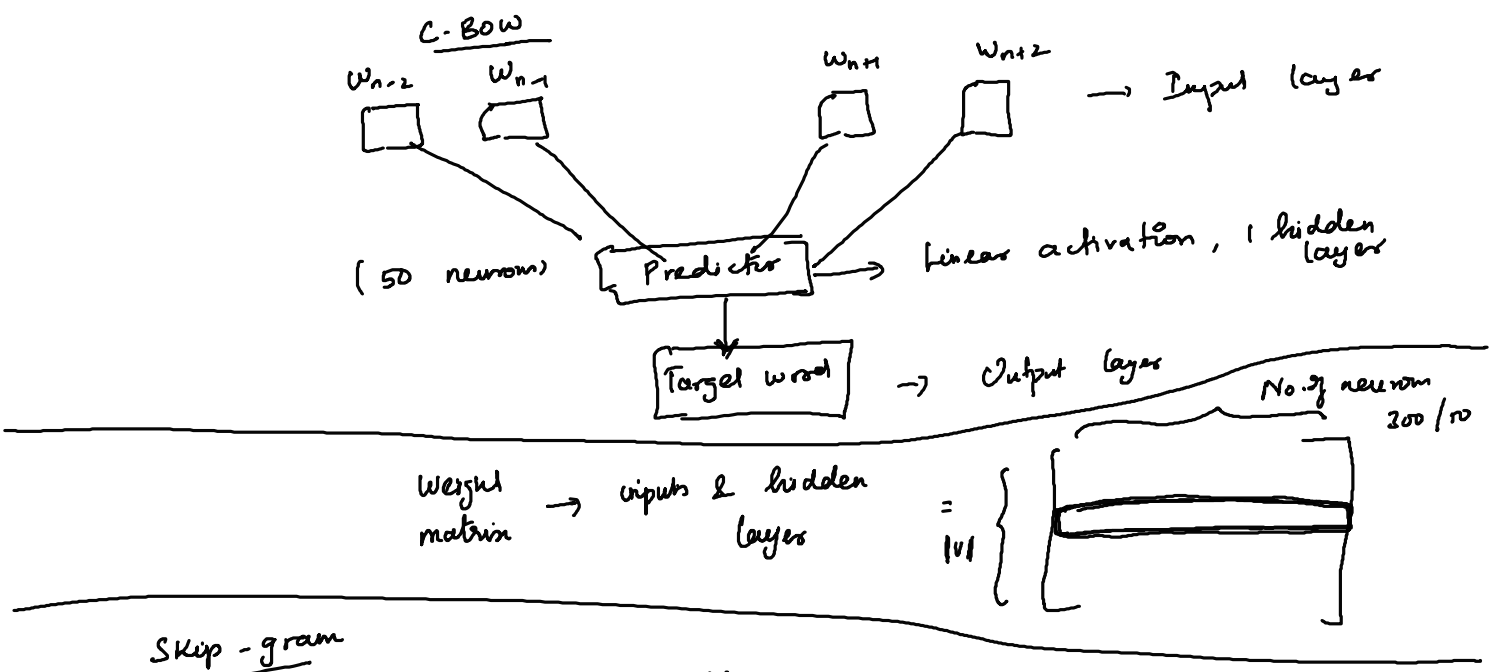
---

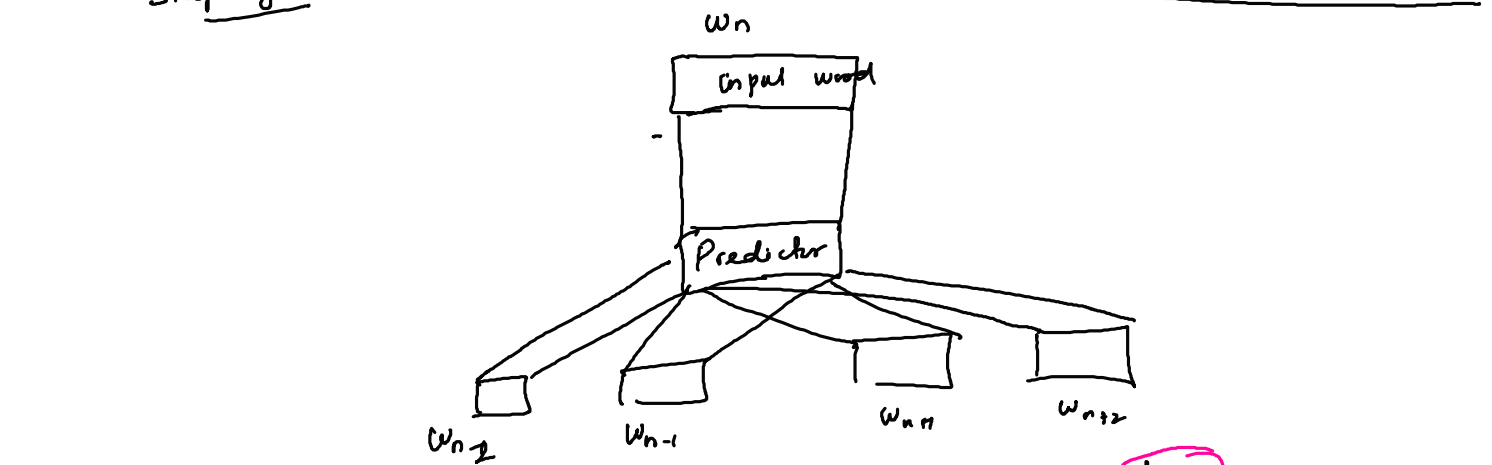
DTM

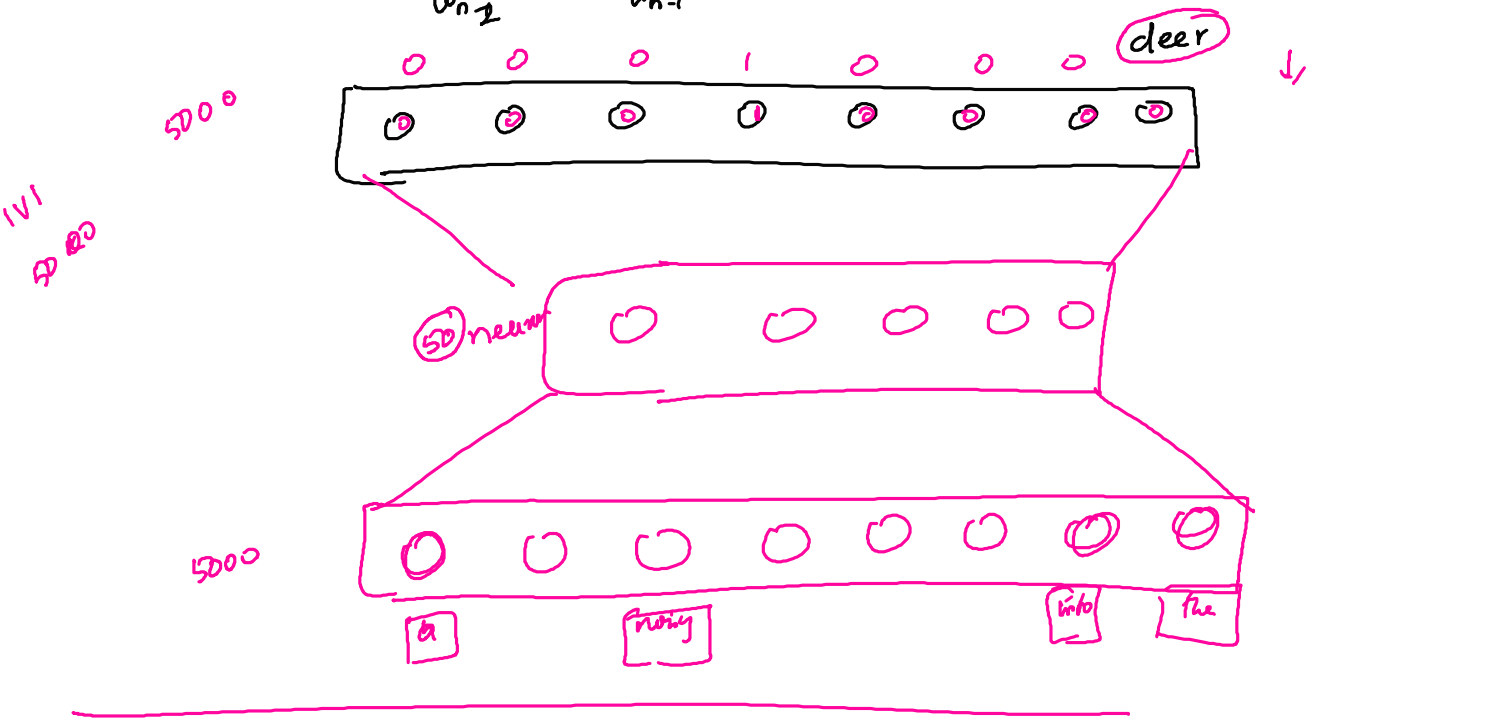
- Sparse
- High dimension
- Less content is used
- Frequency based
- Varies corpus to corpus

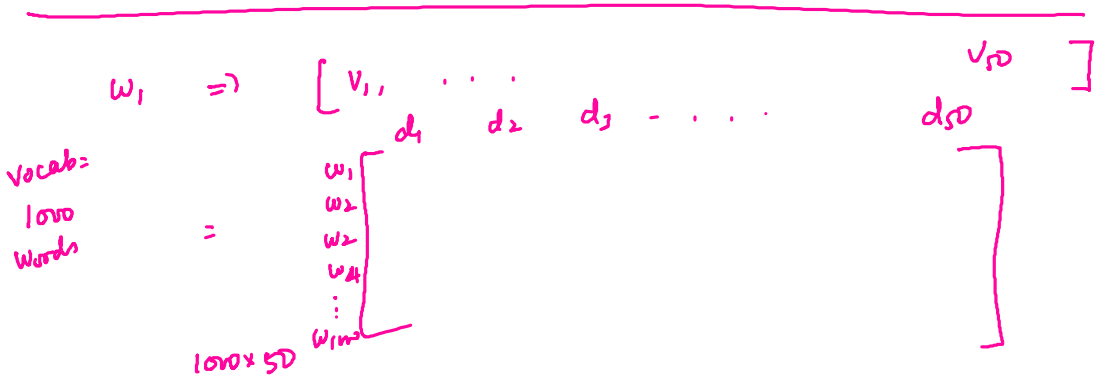
Word Embeddings (word2vec)

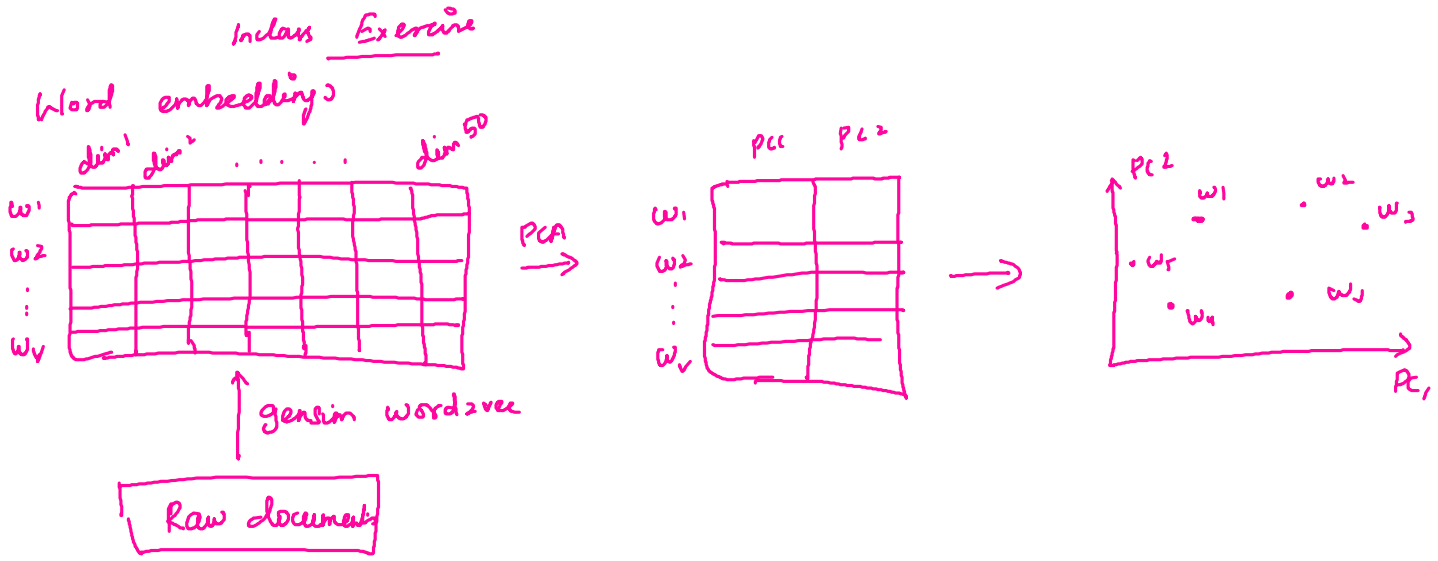
- Dense
- Customized dimension — Low dimension (300)
- More content is used
- Based on shallow neural network
- Pre-trained word embeddings (Trained) or Customized word embedding is also possible


C-BOW
w_{n-2}
w_{n-1}
w_{n+1}
w_{n+2}
→ Input layer
(50 neurons)
Predictor
→ Linear activation, 1 hidden layer
Target word
→ Output layer
No. of neurons
200/10
Weight matrix
→ inputs & hidden layers
=
|V|


Skip-gram
w_n
Input word
Predictor
w_{n-2}
w_{n-1}
w_{n+1}
w_{n+2}


deer
5000
|V|
50
50 neurons
5000
a
rising
into
the


w_1 => [v_1, . . . v_{50}]
d_1 d_2 d_3 . . . . d_{50}
Vocab: 1000 Words =
w_1
w_2
w_3
w_4
w_{1000}
1000 × 50


Inclass Exercise
Word embeddings
dim 1
dim 2
dim 50
w1
w2
wV
PCA
PC1
PC2
w1
w2
wV
PC2
PC1
w1
w2
w3
w4
w5
gensim word2vec
Raw documents