NLP

Saturday, 28 December 2024 6:10 PM

Topics:

1) Corpus – Paragraph
2) Documents - Sentences
3) Vocabulary – unique words
4) words.

Tokenization:

↳ Convert either a paragraph or sentence into tokens.

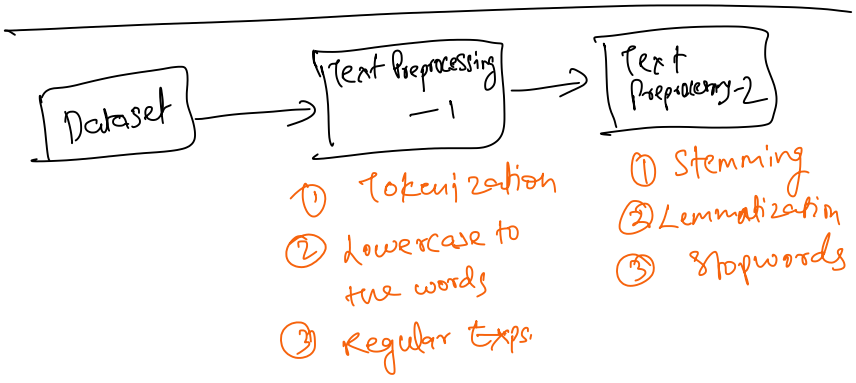
Vocabulary:

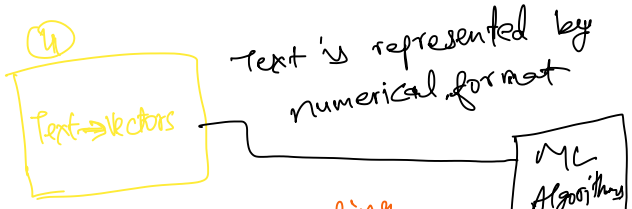
tokens ① I like to drink apple juice.
② My friend likes mango juice.

⇓

total 11 words
unique words: 10 {juice is repeating}

NLTK :-





① One Hot Encoding
② Bag of words (BoW)
③ TF-IDF
④ Word2Vec
⑤ Avg Word2Vec

One Hot Encoding :-

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

Unique vocabularies :→ The, food, is, good, bad, pizza, Amazing

	The	food	is	good	bad	pizza	Amazing
The →	1	0	0	0	0	0	0
food →	0	1	0	0	0	0	0

D1 [[1 0 0 0 0 0 0], → This
[0 1 0 0 0 0 0] → food
[0 0 1 0 0 0 0], → is
[0 0 0 1 0 0 0]] → good
4x7

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

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

→ one hot for each and every word.

Advantages	Disadvantages
① Easy to implement in python. (Sklearn, OneHotEncoder..)	① Creates a sparse matrix → Overfitting → we need fix size for ML algorithm
	② No semantic meaning is captured. (cosine similarity..)
	③ Out of vocabulary. (unique words.)
	④ if we have 50k unique vocabulary size, there will be huge sparse matrix.

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 → stopwords

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

Vocabulary
- good
- boy
- girl

Frequency
3
2
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 & Intuitive
2. fixed size input

Disadvantages:

1. Sparse matrix → overfitting
2. ordering of the words is getting changed.
3. Out of Vocabulary. (OOV)
4. Semantic Meaning still not getting captured.

N Grams:→ Eg. bigrams, trigrams

	food	not	good
S1 → The food is good →	1	0	1
S2 → The food is not good. →	1	1	1

→ ... is both the vector have a ...

The issue ...
little difference although both sentence are opposite to each other.

Bigram - combination of two words.

	food	not	good	food good	food not	not good
S1 →	1	0	1	1	0	0
S2 →	1	1	1	0	1	1

→ Now we have a diff.

Sklearn → n-grams (1,1) → unigrams
(1,2) → unigram bigram
(1,3) → unigram, bigram, trigram
(2,3) → Bigram, trigram.

TF-IDF [Term Frequency - Inverse Document Frequency]

$$\text{term freq} = \frac{\text{No. of rep. of words in sentence}}{\text{No. of words in sentence}}$$

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

Term	freq.		
	S_1	S_2	S_3
good	1/2	1/2	1/2
boy	1/2	0	1/3
girl	0	1/2	1/3

IDF

good $\log_e(3/3)$
boy $\log_e(3/2)$
$\log_e(3/1)$

girl $\log_e(3/2)$

Find TF-IDF

	S_1	S_2	S_3
good	$\frac{1}{2}\times 0$	$\frac{1}{2}\times 0$	$\frac{1}{3}\times\log_e(3/2)$
boy	$\frac{1}{2}\times(\log_e\frac{3}{2})$	0	$\frac{1}{3}\times\log_e(\frac{3}{2})$
girl	0	$\frac{1}{2}\times\log_e(\frac{3}{2})$	

S_1 [0 $\frac{1}{2}\log_e\frac{3}{2}$ 0]

S_2 [0 0 $\frac{1}{2}\log_e\frac{3}{2}$]

S_3 [0 $\frac{1}{3}\log_e\frac{3}{2}$ $\frac{1}{3}\log_e\frac{3}{2}$]

Advantages	Disadvantages.
① Intuitive	① Sparsity exists
② Fixed Size	② OOV.
③* word imp is getting captured. (if word is in all sentences it is considered less important)	

Word Embeddings:-

→ representation of words.

→ in the form of real valued vectors. that encodes the meaning of the words such that words that are closer in vector space are similar in meaning | semantics.

Word Embeddings

- Count or frequency
  - one hot
  - BOW
- Deep Learning Trained models.
  - ↓
  - Word2Vec.

- TF-IDF

- Word...

CBOW

Skipgram

[Continuous BAG of Words]

Word2Vec :- (Published by Google)

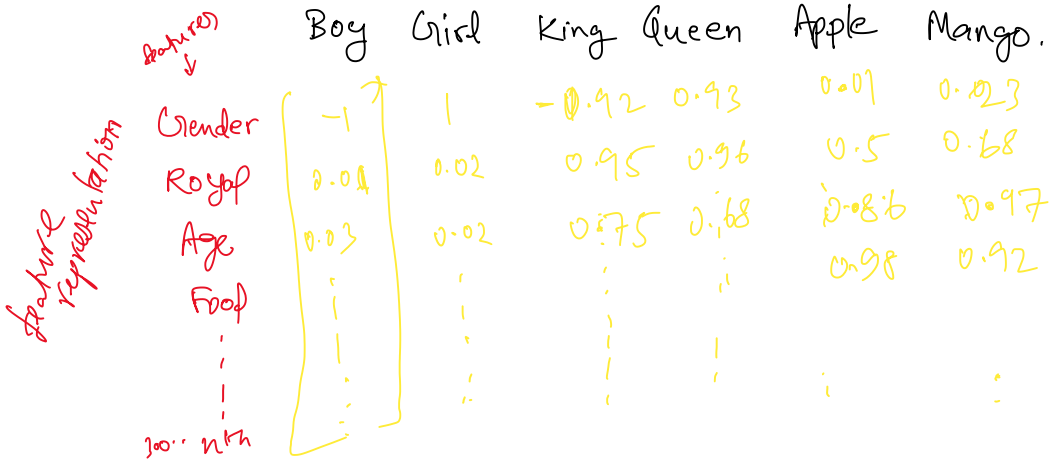
→ Word2Vec is a technique for natural language processing published in 2013 by Google.

→ The word2vec uses a neural network model to learn word associations from a large corpus of a 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.

→ Google trained with 3 billion words.

Vocabulary → unique words in corpus.

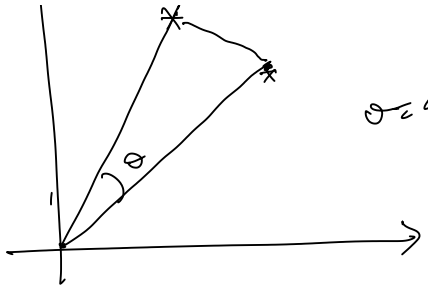


→ Each & every word is represented based on feature representation.

[KING - BOY + QUEEN] = GIRL

Cosine Similarity :-

Distance = 1 - cosine similarity.



$\theta = 45^\circ$ $\quad 1 - \cos\theta = 1 - \cos 45 = 1 - \frac{1}{\sqrt{2}} = 1 - 0.707 = 0.29$

$\theta = 90^\circ \quad 1 - \cos 90 = 1 - 0 = 1$ (different)

$\theta = 0^\circ \quad 1 - \cos 0^\circ = 1 - 1 = 0$ (similar)

\# Word2Vec → CBOW [Continuous Bag of Words]
Word2Vec → skipgram
Word2Vec → Pretrained Model
Word2Vec → Train a model from scratch.

① CBOW

CORPUS:- Dataset

[iNeuron Company is related to Data Science]

↳ central word based on window.

window size = 5

I/P	O/P	(Backward Context & forward context is considered)
→ iNeuron Company, related To	IS	
→ Company IS, To Data	related	
→ IS related, Data Science	To	

CBOW [Fully Connected Neural Nw]

② Skipgram!

O/P	I/P
[iNeuron, Company, Related, To]	Is
[Company, Is, To, Data]	Related

Small Dataset → CBOW

Huge Dataset → Skipgram.

Increase acc. of CBOW or Skipgram.

① Increasing training data.
② Increase window size
→ vector dimension increases.

Google word2vec.

3 billion words → Google News

Feature representation of 300 dim.

cricket = [0. ---- 300dim]

eg. [illegible]

#Advantages of Word2Vec.

1. Dense matrix – earlier we were getting sparse.
2. Semantic info is getting captured
3. Vocab. size → we have fixed set of dimension in word2vec.
4. OOV → every word has some feature representation.