28/03/24

Feature engineering & NLP Algo:

gathered corpus
Pre processing our data
Divide the classes & finds the imp. pt.

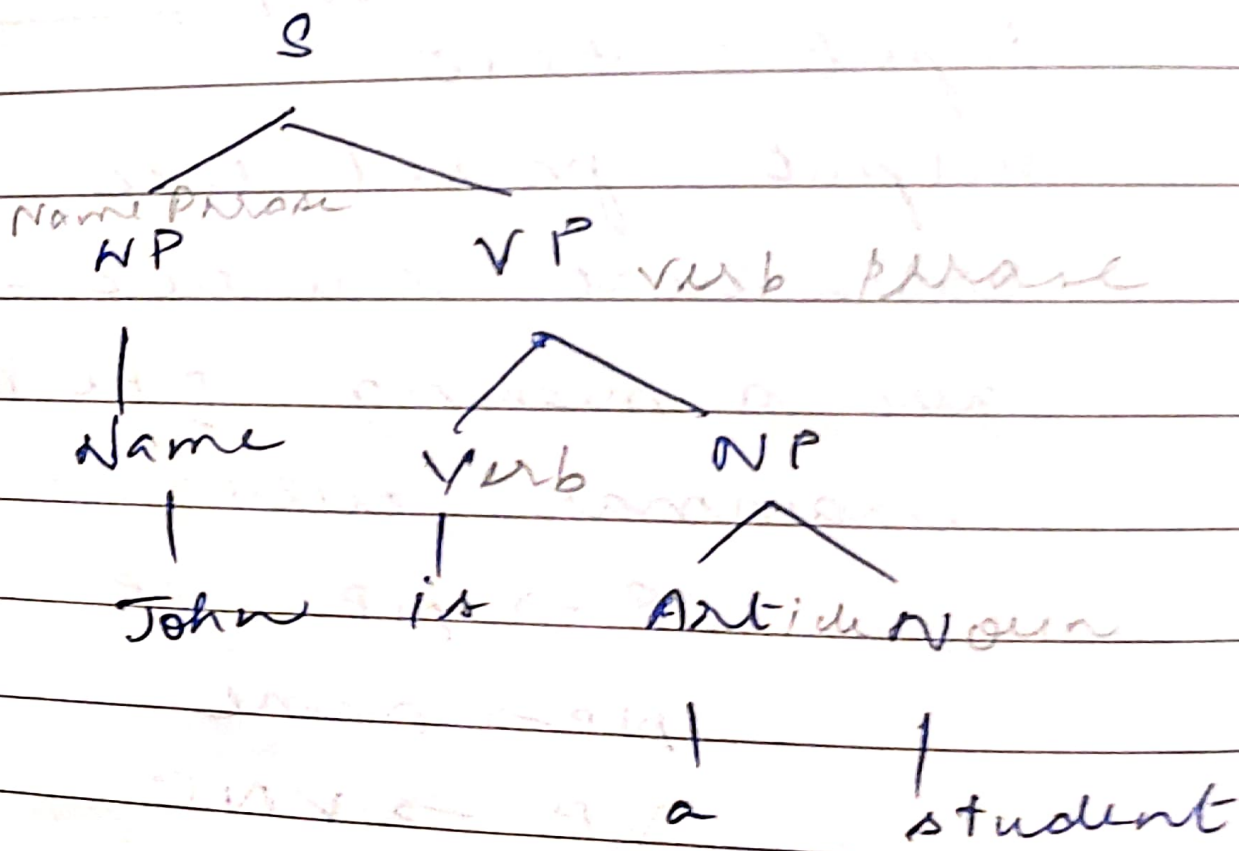
challenges:
- Detect good features
- Feature selection : selected no. of. features
- Manual feature engineering is time consuming
- should have basic knowledge on domain.

7 techniques used in FE for NLP

1. dependency parsing

input : text
output : parsed tree
constituent structure - combining words to give a meaning [He is Tom]
grammar rules:

S → NP VP (S → sentence, NP → Noun Phrase)
NP → Name
VP → V NP
NP → Art N/object


S
NP
VP
Name
Verb
NP
John
is
Article
Noun
a
student

This happens which is followed by dependency parsing

Lexical analysis :

Name → John

VERB → is

⋮

mapping in table

2. Bag-of-words (BOW)

Aim : To verify the syntactical structure of the sentence (need not be grammatically crt.)

Detects ambiguity in the sentence

01.04.24

Used in : Regression based algo.

A (corpus)
- 2 Text document
- d1 : John likes to watch cricket. Chris likes cricket too.
- d2 : John also likes to watch football.

BOW : ["John", "likes", "to", "watch", "cricket", "chris", "too", "also", "football"]

Repetition is avoided.

Picking out the unique words.

"John", "likes", "to", "watch", "cricket", "also", "too"

Freq. of d_1 [1, 1, 1, 1, 1, 1, 1, 0, 0]

d_2 [1, 1, 1, 1, 0, 0, 0, 1, 1]

3] Term Frequency (TF) – Inverse Document Frequency (IDF)

(within the document) ↓ importance

(across the document) ↓ importance

$$TF_{(t)} = \frac{\text{no. of times } t \text{ appeared in } d}{\text{Total no. of terms in } d}$$

$$IDF_{(t)} = \log\left[\frac{\text{Total no. of documents (D)}}{\text{no. of documents with term } t \text{ in it}}\right]$$

$$TF\_IDF = TF * IDF$$

eg: d_1 : This is a car

d_2 : This is a bike

t = this

$TF_{(t)} = \frac{2}{8}$ & $\frac{1}{4}$

$IDF_{(t)} = \log\left(\frac{2}{2}\right) = \log 1 = 0.$

$TF\_IDF = \frac{1}{4} \times 0 = 0$

[t = car]

$TF_{(t)} = \frac{1}{4}$

$IDF_{(t)} = \log\left(\frac{2}{1}\right) = \log 2 = 0.30$

$TF\_IDF = \frac{1}{4} \times 0.3 = 0.25 \times 0.3 = 0.075$

× ——— ×

04-04-21

Text / Word Rep.

① Dependency Parsing
② BOW [0, 1, . . .]
③ TF_IDF [ ]
④ word embedding.

④ Word embedding (converting words to vector) — Word2Vec / Glove

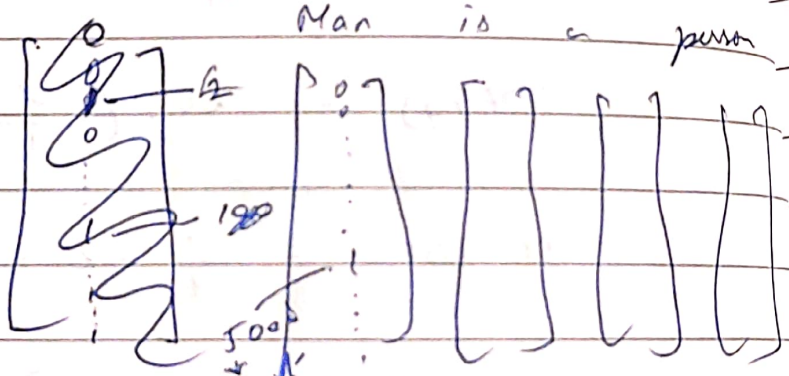
One hot rep.

$|V| = 10,000$

Man is a person

index: [5000] [100] [2] [200]

1 Hot rep:-



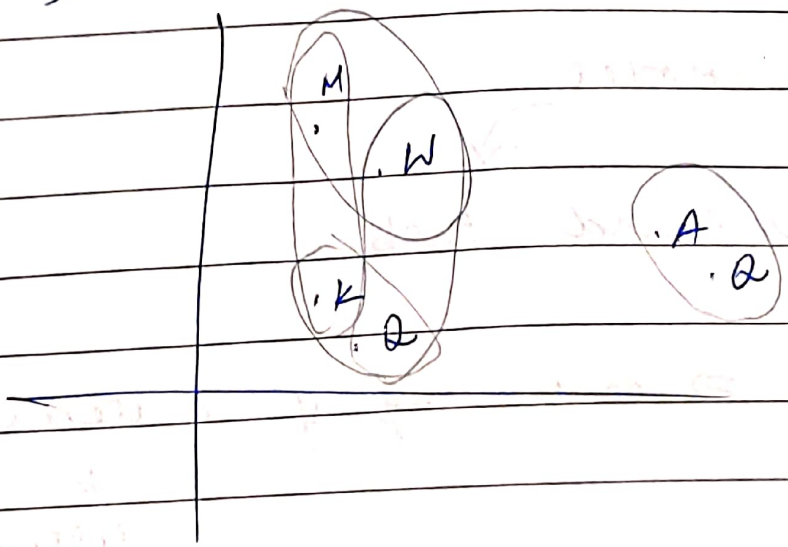
Disadvantage:

- Vector [high dimension]

Now comes Word embedding

Features	Words					
	Man	Woman	King	Queen	Apple	Orange
Gender	-1	1	0.93	0.92	0.0	0.01
Human	0.98	0.97	-1	1	0.0	0
Fruit	0.01	0.0	0.09	0.01	0.1	1
⋮						
300 dimension						

Backend :



steps: word2
- Vector dimension mention (10,000) in our eg.
- One hot rep.
- Out of hot rep
- no. of dimension (no. of features) mention

Word embedding using Keras

① sentences
② 1 Hot Rep. ⇒ Index from Dictionary
③ 1 Hot Rep. ⇒ embedding layer to keras to get Embedding Matrix
④ Emb. Matrix ⇒ ✓ Size = 10,000 Dim = 10

05-04-24

technique

① N-gram Models

↓ ↓ ↘

no. letter / word Prob.

n-gram ⇒ conti. seq. of n items

↓

letter / word / syllables

Types

n=1 ... unigram

n=2 ... bigram

n=3 ... Trigram

⋮

I/P = The boy ate a chocolate

Uni = The boy ate a chocolate

Bi = The boy ate a a chocolate

Tri = The boy ate boy ate a ate a choc.

Application of ngram

→ plagiarism tool
(copying others work)

→ computational biology

Language Model: To predict next words in sentence suggestion

$$P(w) = \frac{count(w)}{N}$$

$$P(A|B) = \frac{P(A \cap B)}{P(B)}$$

Bigram: $P(w_i / w_{i-1}) = \frac{count(w_{i-1}, w_i)}{count(w_{i-1})}$

Trigram: $P(w_i | w_{i-2}, w_{i-1})$

$$= \frac{count(w_{i-2}, w_{i-1}, w_i)}{count(w_{i-2}, w_{i-1})}$$

Eg:

corpus:

The girl bought a choc.
The boy ate a choc.
The girl bought a toy.
The girl played with the toy.

step 1: Build vocabulary.

B·O·W { the, girl, bought, a, choc, boy, ate, toy, played, with }

step 2: predict next words, for all words in vocabulary

I/P : the girl

$$P(\text{the} / \text{the, girl}) = \frac{\text{count(the, girl, the)}}{\text{count(the, girl)}}$$

$$= \frac{0}{3} = 0.$$

$$P(\text{girl} / \text{the, girl}) = \frac{\text{count(the, girl, girl)}}{\text{count(the, girl)}}$$

$$= \frac{0}{3} = 0$$

$$P(\text{bought} / \text{the, girl}) = \frac{\text{count(the, girl, bought)}}{\text{count(the, girl)}}$$

$$= \frac{2}{3} = 0.67 = 67\%$$

$P(a / the, girl) = \frac{0}{3} = 0.$

$P(cheeca. / the, girl) = \frac{0}{3}, 0.$

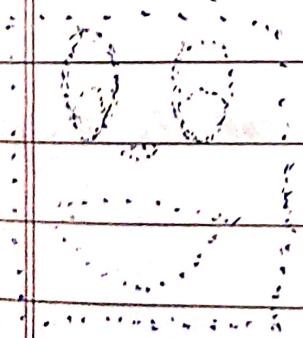
$P(boy / the, girl) = 0$

$P(ate / the, girl) = 0$

$P(toy / the, girl) = 0$

$P(played / the, girl) = \frac{1}{3} = 0.33 = 33\%$

$P(with / the, girl) = 0$



-04-24

⑥ POS Tagging

↓

Parts of speech

POS Tagger ⇒ nltk

Assignment

Word imbedding :

Word 2 vec :

Pre-trained word 2 vec:

- continuous bag of words
'the', 'cat', 'over'
predict → 'jumping'

- skip-gram

How its Trained
- Initialization
- Sliding window
- context Prediction
- Optimization
- Train

Applications:

Advantages of Word2 vec :

- semantic understanding
- Reduced Dimensionality

Word2Vec Implementation :

Prerequisite : gerism

9.05.24

Rule based systems

Components
- rules
- parsers
- linguistic resources (dictionary)

Diff b/w Machine learning + deep learning

Rule-based vs. Machine learning approach
↑
self driving cars

project:
what is ?
Domain ?
data set ?
Next action ?

Rule based architecture

general archi

Practical archi

9/05/24

general architecture of the rule-based system as an expert system