# PHASE-1 - TEXT PREPROCESSING - 2 :

## TEXT PROCESSING (NLP) :-

## STEMMING :-

process of reducing words to their root form.

Example :-

| Original Word | Stem |
|---|---|
| running | run |
| played | play |
| eating | eat |

## # PORTER STEMMER (PorterStemmer)

PorterStemmer is a rule-based stemming algo. in NLTK used to reduce words to their root form by removing suffixes.

Example :-

```
words = ['eating', 'eats', 'eaten', 'writing', 'writes',
         'programming', 'programs', 'history', 'finally',
         'finalized']

from nltk.stem import PorterStemmer

Stemming = PorterStemmer()

for i in words:
    print(i + "----->" + Stemming.stem(i))

# eating ---> eat
  eats ---> eat
  eaten ---> eaten
  writing ---> write
  writes ---> write
  programming ---> program
  programs ---> program
  history ---> histori
  finally ---> final
  finalized ---> final

Sometimes, it changes the meaning

print(stemming.stem('congratulations'))
# congratul

print(stemming.stem('sitting'))
# sit
```

# Regexp Stemmer

means using regular expression to remove common word endings (suffixes) to get the root word.

Example :- Common Suffixes Removed :-
ing, ed, s, ly

# Output

playing ⟶ play
played ⟶ play
players ⟶ player
slowly ⟶ slow

Syntax :-

```
from nltk.stem import RegexpStemmer

reg_stemmer = RegexpStemmer(
    'ing$|s$|e$|able$', min=
```

```
reg_stemmer.stem('eating')   # eat

reg_stemmer.stem('ingeating')   # ingeat
```

But is we not use dollar sign ($) in front so it removes the whole ~~prefix~~

```
RegexpStemmer('ing', min=4)   # eat
```

# SNOWBALL STEMMER

Snowball stemmer is also a rule-based Stemming algo used in NLP. But is better than Porter Stemmer.

Syntax :-

```
from nltk.stem import SnowballStemmer

Snowballstemmerobject = SnowballStemmer('english')
```

Note :- In Snowball Stemmer, we have to provid a language (we are using 'english' here).

```
words = ['eating', 'eats', 'writing', 'writes', 'finally']

for word in words :-
    print (word + " --> " + snowballstemmerobject
                            .stem (word))
```

#
```
eating  -> eat
eats    -> eat
writing -> write
writes  -> write
finally -> write
```

```
print (Stemmig.stem ("fairly")) # fairli
```

Note :- If we use porter stemmer it doesn't give the right (meaningfull) output, all the time

```
print [ Snowoballstemmerobject.stem ("fairly")
                                    # fair
```

Note :- Snowball Stemmer is better than porter Stemmer but it also lacks perfection sometimes.

# LEMMATIZATION :-

(WordNetLemmatizer)

Lemmatization is an NLP technique that reduce words to their base dictionary form (lemma) by considering grammer and meaning.

# More Accurate than stemming
# Produce Meaning full words
# But slower than stemming.

Syntax :-

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

# In WordNetLemmatizer, we provide 2 parameters ('word', pos = 'n')

Where, word => the word you passed

Pos
- pos = 'n' ⟶ noun
- pos = 'v' ⟶ verb
- pos = 'a' ⟶ adjective
- pos = 'r' ⟶ adverb

So basically we are provide 2 parameter inside WordNet Lemmatizer ('word', pos='n')

where word is what we passed and pos is what we need this word to change into.

And pos='n' / # noun is by default

Or we can also write it as

WordNet Lemmatize ('word')

Code :-

```
from nltk.stem import WordNetLemmatizer
lemmatizer = WordNetLemmatizer()
print(lemmatizer.lemmatize("going"))
                    # 'going'
                    pos by default noun
print(lemmatizer.lemmatize("going", pos="v"))
                    # 'go'
                    pos = verb.

words = ['eating', 'eats', 'eaten', 'writing', 'writes',
         'programming', 'programs', 'history',
         'finally', 'finalize']
```

```
for word in words:
    print(word + "-->" + lemmatizer.lemmatize
                         (word, pos='v'))
```

#

eating --> eat
eats --> eat
eaten --> eat
writing --> write
writes --> write
programming --> program
programs --> program
history --> history
finally --> finally
finalized --> finalize

Note :- It provides the best possible and meaningful output but it is just a bit slow from stemming.

# When to use It:

→ NLP tasks needing accuracy (chatbots, QA)
→ When speed is priority → use stemming.