**Experiment No.: 13**

**Name: JESNAMOL THOMAS**

**Roll No:6**

**Batch:MCA B**

**Date:24/11/22**

**Aim**

Nltk preprocess

**Program and Output**

# Setup

!pip install -q wordcloud

import wordcloud

import nltk

nltk.download('stopwords')

nltk.download('wordnet')

nltk.download('punkt')

nltk.download('averaged\_perceptron\_tagger')

import pandas as pd

import matplotlib.pyplot as plt

import io

import unicodedata

import numpy as np

import re

import string

**Output**

[nltk\_data] Downloading package stopwords to /root/nltk\_data...

[nltk\_data] Package stopwords is already up-to-date!

[nltk\_data] Downloading package wordnet to /root/nltk\_data...

[nltk\_data] Package wordnet is already up-to-date!

[nltk\_data] Downloading package punkt to /root/nltk\_data...

[nltk\_data] Package punkt is already up-to-date!

[nltk\_data] Downloading package averaged\_perceptron\_tagger to

[nltk\_data] /root/nltk\_data...

[nltk\_data] Package averaged\_perceptron\_tagger is already up-to-

[nltk\_data] date!

**Program**

from nltk import pos\_tag

from nltk import RegexpParser

text ="learn php from guru99 and make study easy".split()

print("After Split:",text)

tokens\_tag = pos\_tag(text)

print("After Token:",tokens\_tag)

patterns= """mychunk:{<NN.?>\*<VBD.?>\*<JJ.?>\*<CC>?}"""

chunker = RegexpParser(patterns)

print("After Regex:",chunker)

output = chunker.parse(tokens\_tag)

print("After Chunking",output)

**Output**

After Split: ['learn', 'php', 'from', 'guru99', 'and', 'make', 'study', 'easy']

After Token: [('learn', 'JJ'), ('php', 'NN'), ('from', 'IN'), ('guru99', 'NN'), ('and', 'CC'), ('make', 'VB'), ('study', 'NN'), ('easy', 'JJ')]

After Regex: chunk.RegexpParser with 1 stages:

RegexpChunkParser with 1 rules:

<ChunkRule: '<NN.?>\*<VBD.?>\*<JJ.?>\*<CC>?'>

After Chunking (S

(mychunk learn/JJ)

(mychunk php/NN)

from/IN

(mychunk guru99/NN and/CC)

make/VB

(mychunk study/NN easy/JJ))

**Program**

# Constants

# POS (Parts Of Speech) for: nouns, adjectives, verbs and adverbs

DI\_POS\_TYPES = {'NN':'n', 'JJ':'a', 'VB':'v', 'RB':'r'}

POS\_TYPES = list(DI\_POS\_TYPES.keys())

# Constraints on tokens

MIN\_STR\_LEN = 3

RE\_VALID = '[a-zA-Z]'

from collections import Counter

import nltk

text = "Guru99 is one of the best sites to learn WEB, SAP, Ethical Hacking and much more online."

lower\_case = text.lower()

tokens = nltk.word\_tokenize(lower\_case)

tags = nltk.pos\_tag(tokens)

counts = Counter( tag for word,  tag in tags)

print(counts)

**Output**

Counter({'NN': 5, ',': 2, 'VBZ': 1, 'CD': 1, 'IN': 1, 'DT': 1, 'JJS': 1, 'NNS': 1, 'TO': 1, 'VB': 1, 'JJ': 1, 'CC': 1, 'RB': 1, 'JJR': 1, '.': 1})

**Program**

import nltk

text = "Guru99 is a totally new kind of learning experience."

Tokens = nltk.word\_tokenize(text)

output = list(nltk.bigrams(Tokens))

print(output)

**Output**

[('Guru99', 'is', 'a'), ('is', 'a', 'totally'), ('a', 'totally', 'new'), ('totally', 'new', 'kind'), ('new', 'kind', 'of'), ('kind', 'of', 'learning'), ('of', 'learning', 'experience'), ('learning', 'experience', '.')]

**Program**

import nltk

text = "Guru99 is a totally new kind of learning experience."

Tokens = nltk.word\_tokenize(text)

output = list(nltk.trigrams(Tokens))

print(output)

**Output**

[('Guru99', 'is', 'a'), ('is', 'a', 'totally'), ('a', 'totally', 'new'), ('totally', 'new', 'kind'), ('new', 'kind', 'of'), ('kind', 'of', 'learning'), ('of', 'learning', 'experience'), ('learning', 'experience', '.')]

**Program**

import nltk

text = "learn php from guru99"

tokens = nltk.word\_tokenize(text)

print(tokens)

tag = nltk.pos\_tag(tokens)

print(tag)

grammar = "NP: {<DT>?<JJ>\*<NN>}"

cp  =nltk.RegexpParser(grammar)

result = cp.parse(tag)

print(result)

**Output**

['learn', 'php', 'from', 'guru99']

[('learn', 'JJ'), ('php', 'NN'), ('from', 'IN'), ('guru99', 'NN')]

(S (NP learn/JJ php/NN) from/IN (NP guru99/NN))

**Program**

from nltk.corpus import stopwords

print(stopwords.words('english'))

**Output**

['i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', "you're", "you've", "you'll", "you'd", 'your', 'yours', 'yourself', 'yourselves', 'he', 'him', 'his', 'himself', 'she', "she's", 'her', 'hers', 'herself', 'it', "it's", 'its', 'itself', 'they', 'them', 'their', 'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', "that'll", 'these', 'those', 'am', 'is', 'are', 'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an', 'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about', 'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up', 'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when', 'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no', 'nor', 'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', "don't", 'should', "should've", 'now', 'd', 'll', 'm', 'o', 're', 've', 'y', 'ain', 'aren', "aren't", 'couldn', "couldn't", 'didn', "didn't", 'doesn', "doesn't", 'hadn', "hadn't", 'hasn', "hasn't", 'haven', "haven't", 'isn', "isn't", 'ma', 'mightn', "mightn't", 'mustn', "mustn't", 'needn', "needn't", 'shan', "shan't", 'shouldn', "shouldn't", 'wasn', "wasn't", 'weren', "weren't", 'won', "won't", 'wouldn', "wouldn't"]

**Program**

en\_stopwords = stopwords.words('english')

def remove\_stopwords(text):

    result = []

    for token in text:

        if token not in en\_stopwords:

            result.append(token)

    return result

#Test

text = "this is the only solution of that question".split()

remove\_stopwords(text)

**Output**

['solution', 'question']

**Program**

from nltk.stem import PorterStemmer

from nltk.tokenize import word\_tokenize

ps = PorterStemmer()

sentence = "Programmers program with programming languages"

words = word\_tokenize(sentence)

for w in words:

    print(w, " : ", ps.stem(w))

**Output**

Programmers : programm

program : program

with : with

programming : program

languages : languag