NLP PRACTICAL ASSIGNMENT

##Assignment No.01##

#Title:"Text Pre-Processing using NLP operations:perform Tokenization

# Stop word removal,Lemmatization ,Part-of-Speech Tagging use any sample text"

#import libraries

import spacy

# Load the language model

nlp = spacy.load("en\_core\_web\_sm")

# Define the input text with spaces between sentences

about\_text = (

   "India is my country. "

   "Maharashtra is my state."

)

# 1. Tokenization:

about\_doc = nlp(about\_text)

print("1. Tokenization:")

for token in about\_doc:

    print(token, token.idx)

# 2. Stop Words Removal:

about\_doc = nlp(about\_text)

print("\n2. Stop Words Removal:")

print([token for token in about\_doc if not token.is\_stop])

# 3. Lemmatization:

about\_doc = nlp(about\_text)

print("\n3. Lemmatization:")

for token in about\_doc:

    if str(token) != str(token.lemma\_):

        print(f"{str(token):>20} : {str(token.lemma\_)}")

# 4. Part of Speech Tagging:

about\_doc = nlp(about\_text)

print("\n4. Part of Speech Tagging:")

for token in about\_doc:

    print(

        f"""

TOKEN: {str(token)}

=====

TAG: {str(token.tag\_):10} POS: {token.pos\_}

EXPLANATION: {spacy.explain(token.tag\_)}"""

    )

#----------output-------#

"""India 0

is 6

my 9

country 12

. 18

Maharashtra 20

is 32

my 35

state 38

. 43

[India, country, ., Maharashtra, state, .]

is : be

is : be

TOKEN: India

=====

TAG: NNP        POS: PROPN

EXPLANATION: noun, proper singular

TOKEN: is

=====

TAG: VBZ        POS: AUX

EXPLANATION: verb, 3rd person singular present

TOKEN: my

=====

TAG: PRP$       POS: DET

EXPLANATION: pronoun, possessive

TOKEN: country

=====

TAG: NN         POS: NOUN

EXPLANATION: noun, singular or mass

TOKEN: .

=====

TAG: .          POS: PUNCT

EXPLANATION: punctuation mark, sentence closer

TOKEN: Maharashtra

=====

TAG: NNP        POS: PROPN

EXPLANATION: noun, proper singular

TOKEN: is

=====

TAG: VBZ        POS: AUX

EXPLANATION: verb, 3rd person singular present

TOKEN: my

=====

TAG: PRP$       POS: DET

EXPLANATION: pronoun, possessive

TOKEN: state

=====

TAG: NN         POS: NOUN

EXPLANATION: noun, singular or mass

TOKEN: .

=====

TAG: .          POS: PUNCT

EXPLANATION: punctuation mark, sentence closer"""

##Assignment No.02##

#Title:Assignment to implement Bag of Words and TFIDF using Gensim library.

import gensim

from gensim import corpora

from gensim.utils import simple\_preprocess

text2 = ["""I love programming

         Python is my favorite programming language.

         Programming allows me to solve real-world problems."""]

tokens2 = [[item for item in line.split()] for line in text2]

g\_dict2 = corpora.Dictionary(tokens2)

print("The dictionary has: " + str(len(g\_dict2)) + " tokens\n")

print(g\_dict2.token2id)

g\_bow2 = [g\_dict2.doc2bow(token, allow\_update=True) for token in tokens2]

print("Bag of Words : ", g\_bow2)

text3 = ["""I love programming

         Python is my favorite programming language.

         Programming allows me to solve real-world problems."""]

g\_dict3 = corpora.Dictionary([simple\_preprocess(line) for line in text3])

g\_bow3 = [g\_dict3.doc2bow(simple\_preprocess(line)) for line in text3]

print("\nDictionary : ")

for item in g\_bow3:

    print([[g\_dict3[id], freq] for id, freq in item])

##OUTPUT##

'''

The dictionary has: 12 tokens

{'I': 0, 'Python': 1, 'allows': 2, 'favorite': 3, 'is': 4, 'language.': 5, 'love': 6, 'me': 7, 'my': 8, 'programming': 9, 'real-world': 10, 'solve': 11}

Bag of Words :  [[(0, 1), (6, 1), (9, 1)], [(1, 1), (3, 1), (4, 1), (5, 1), (8, 1), (9, 1)], [(2, 1), (7, 1), (10, 1), (11, 1), (9, 1)]]

Dictionary :

[['I', 1], ['love', 1], ['programming', 1]]

[['Python', 1], ['favorite', 1], ['is', 1], ['language', 1], ['my', 1], ['programming', 1]]

[['allows', 1], ['me', 1], ['real-world', 1], ['solve', 1], ['programming', 1]]

'''

###  Assignment No 3 ###

#Assignment Title : Name Entity Recognition in python with spacy

import spacy

# Load the English language model

nlp = spacy.load("en\_core\_web\_sm")

def perform\_ner(text):

    # Process the text using SpaCy

    doc = nlp(text)

    # Extract named entities and their labels

    entities = [(ent.text, ent.label\_) for ent in doc.ents]

    return entities

if \_\_name\_\_ == "\_\_main\_\_":

    # Example text

    text = "Earth is the third planet from the Sun in our solar system and the only known celestial body to support life. With a diverse range of ecosystems, it is home to a vast array of plant and animal species, including humans."

    # Perform Named Entity Recognition

    named\_entities = perform\_ner(text)

    # Print the results

    print("Named Entities:")

    for entity, label in named\_entities:

        print(f"{entity} - {label}")

'''

\*\*\*\*\*\*\*\*\*\*\*\*\*\*    OUTPUT

Named Entities:

Earth - LOC

third - ORDINAL

Sun - ORG

'''

###  Assignment No 4 ###

"""Assignment Title : Implement Bi-gram, Tri-gram word sequence and its count in text input

data using NLTK library"""

from nltk import ngrams

from nltk.util import ngrams

#unigram model

n = 1

sentence = 'Earth is the third planet from the Sun in our solar system and the only known celestial body to support life. With a diverse range of ecosystems, it is home to a vast array of plant and animal species, including humans.'

unigrams = ngrams(sentence.split(), n)

print(f"\n\*\*\*\*\*\*\*\*\*\*\*   UNIGRAM    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

for item in unigrams:

    print(item)

#bigram model

n = 2

sentence = 'Earth is the third planet from the Sun in our solar system and the only known celestial body to support life. With a diverse range of ecosystems, it is home to a vast array of plant and animal species, including humans.'

unigrams = ngrams(sentence.split(), n)

print(f"\n\*\*\*\*\*\*\*\*\*\*\*   BIGRAM    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

for item in unigrams:

    print(item)

#trigram model

n = 3

sentence = 'Earth is the third planet from the Sun in our solar system and the only known celestial body to support life. With a diverse range of ecosystems, it is home to a vast array of plant and animal species, including humans.'

unigrams = ngrams(sentence.split(), n)

print(f"\n\*\*\*\*\*\*\*\*\*\*\*   TRIGRAM    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*")

for item in unigrams:

    print(item)

'''

\*\*\*\*\*\*\*\*\*\*\*\*    OUTPUT    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

\*\*\*\*\*\*\*\*\*\*\*   UNIGRAM    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

('Earth',)

('is',)

('the',)

('third',)

('planet',)

('from',)

('the',)

('Sun',)

('in',)

('our',)

('solar',)

('system',)

('and',)

('the',)

('only',)

('known',)

('celestial',)

('body',)

('to',)

('support',)

('life.',)

('With',)

('a',)

('diverse',)

('range',)

('of',)

('ecosystems,',)

('it',)

('is',)

('home',)

('to',)

('a',)

('vast',)

('array',)

('of',)

('plant',)

('and',)

('animal',)

('species,',)

('including',)

('humans.',)

\*\*\*\*\*\*\*\*\*\*\*   BIGRAM    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

('Earth', 'is')

('is', 'the')

('the', 'third')

('third', 'planet')

('planet', 'from')

('from', 'the')

('the', 'Sun')

('Sun', 'in')

('in', 'our')

('our', 'solar')

('solar', 'system')

('system', 'and')

('and', 'the')

('the', 'only')

('only', 'known')

('known', 'celestial')

('celestial', 'body')

('body', 'to')

('to', 'support')

('support', 'life.')

('life.', 'With')

('With', 'a')

('a', 'diverse')

('diverse', 'range')

('range', 'of')

('of', 'ecosystems,')

('ecosystems,', 'it')

('it', 'is')

('is', 'home')

('home', 'to')

('to', 'a')

('a', 'vast')

('vast', 'array')

('array', 'of')

('of', 'plant')

('plant', 'and')

('and', 'animal')

('animal', 'species,')

('species,', 'including')

('including', 'humans.')

\*\*\*\*\*\*\*\*\*\*\*   TRIGRAM    \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

('Earth', 'is', 'the')

('is', 'the', 'third')

('the', 'third', 'planet')

('third', 'planet', 'from')

('planet', 'from', 'the')

('from', 'the', 'Sun')

('the', 'Sun', 'in')

('Sun', 'in', 'our')

('in', 'our', 'solar')

('our', 'solar', 'system')

('solar', 'system', 'and')

('system', 'and', 'the')

('and', 'the', 'only')

('the', 'only', 'known')

('only', 'known', 'celestial')

('known', 'celestial', 'body')

('celestial', 'body', 'to')

('body', 'to', 'support')

('to', 'support', 'life.')

('support', 'life.', 'With')

('life.', 'With', 'a')

('With', 'a', 'diverse')

('a', 'diverse', 'range')

('diverse', 'range', 'of')

('range', 'of', 'ecosystems,')

('of', 'ecosystems,', 'it')

('ecosystems,', 'it', 'is')

('it', 'is', 'home')

('is', 'home', 'to')

('home', 'to', 'a')

('to', 'a', 'vast')

('a', 'vast', 'array')

('vast', 'array', 'of')

('array', 'of', 'plant')

('of', 'plant', 'and')

('plant', 'and', 'animal')

('and', 'animal', 'species,')

('animal', 'species,', 'including')

('species,', 'including', 'humans.')

'''

###  Assignment No 5 ###

"""Assignment Title :  Implement regular expression function to find URL, IP address, Date,

PAN number in textual data using python libraries"""

import spacy

import re

# Load the spaCy English language model

nlp = spacy.load("en\_core\_web\_sm")

# Define regular expressions

url\_pattern = re.compile(r'https?://\S+|www\.\S+')

ip\_address\_pattern = re.compile(r'\b(?:\d{1,3}\.){3}\d{1,3}\b')

date\_pattern = re.compile(r'\d{4}-\d{2}-\d{2}')

pan\_number\_pattern = re.compile(r'[A-Z]{5}[0-9]{4}[A-Z]')

def extract\_entities(text):

    # Tokenize the text using spaCy

    doc = nlp(text)

    # Find entities using regular expressions

    urls = re.findall(url\_pattern, text)

    ip\_addresses = re.findall(ip\_address\_pattern, text)

    dates = re.findall(date\_pattern, text)

    pan\_numbers = re.findall(pan\_number\_pattern, text)

    # Extract spaCy entities

    entities = [(ent.text, ent.label\_) for ent in doc.ents]

    return {

        'urls': urls,

        'ip\_addresses': ip\_addresses,

        'dates': dates,

        'pan\_numbers': pan\_numbers,

        'spaCy\_entities': entities

    }

# Example usage

text\_data = """

Here is a sample text with a URL: https://www.Sample.com.

Also, an IP address: 192.168.789.102.

The date is 2023-01-01.

A PAN number is BBRPL4574H.

"""

results = extract\_entities(text\_data)

print("URLs:", results['urls'])

print("IP Addresses:", results['ip\_addresses'])

print("Dates:", results['dates'])

print("PAN Numbers:", results['pan\_numbers'])

print("Entities:", results['spaCy\_entities'])

'''

\*\*\*\*\*\*\*\*\*\*\*\*\*\*   OUTPUT     \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

URLs: ['https://www.Sample.com.']

IP Addresses: ['192.168.789.102']

Dates: ['2023-01-01']

PAN Numbers: ['BBRPL4574H']

Entities: [('IP', 'ORG'), ('192.168.789.102', 'CARDINAL'), ('2023-01-01', 'DATE'), ('PAN', 'ORG')]

'''

###  Assignment No 6 ###

"""Assignment Title : : Implement and visualize Dependency Parsing of Textual Input

using Stan- ford CoreNLP and Spacy library"""

import spacy

from spacy import displacy

nlp = spacy.load("en\_core\_web\_sm")

multiline\_text = """

Earth is the third planet from the Sun in our solar system and the only known celestial body to support life.

With a diverse range of ecosystems, it is home to a vast array of plant and animal species, including humans.

Earth's atmosphere, composed mainly of nitrogen and oxygen, sustains life by providing the necessary conditions for biological processes to thrive.

"""

multiline\_doc = nlp(multiline\_text)

for token in multiline\_doc:

    print(

        f"""

TOKEN: {token.text}

=====

{token.tag\_ = }

{token.head.text = }

{token.dep\_ = }"""

    )

displacy.serve(multiline\_doc, style="dep")

'''

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*    OUTPUT      \*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

TOKEN:

=====

token.tag\_ = '\_SP'

token.head.text = 'Earth'

token.dep\_ = 'dep'

TOKEN: Earth

=====

token.tag\_ = 'NNP'

token.head.text = 'is'

token.dep\_ = 'nsubj'

TOKEN: is

=====

token.tag\_ = 'VBZ'

token.head.text = 'is'

token.dep\_ = 'ROOT'

TOKEN: the

=====

token.tag\_ = 'DT'

token.head.text = 'planet'

token.dep\_ = 'det'

TOKEN: third

=====

token.tag\_ = 'JJ'

token.head.text = 'planet'

token.dep\_ = 'amod'

TOKEN: planet

=====

token.tag\_ = 'NN'

token.head.text = 'is'

token.dep\_ = 'attr'

TOKEN: from

=====

token.tag\_ = 'IN'

token.head.text = 'planet'

token.dep\_ = 'prep'

TOKEN: the

=====

token.tag\_ = 'DT'

token.head.text = 'Sun'

token.dep\_ = 'det'

TOKEN: Sun

=====

token.tag\_ = 'NNP'

token.head.text = 'from'

token.dep\_ = 'pobj'

TOKEN: in

=====

token.tag\_ = 'IN'

token.head.text = 'planet'

token.dep\_ = 'prep'

TOKEN: our

=====

token.tag\_ = 'PRP$'

token.head.text = 'system'

token.dep\_ = 'poss'

TOKEN: solar

=====

token.tag\_ = 'JJ'

token.head.text = 'system'

token.dep\_ = 'amod'

TOKEN: system

=====

token.tag\_ = 'NN'

token.head.text = 'in'

token.dep\_ = 'pobj'

TOKEN: and

=====

token.tag\_ = 'CC'

token.head.text = 'planet'

token.dep\_ = 'cc'

TOKEN: the

=====

token.tag\_ = 'DT'

token.head.text = 'body'

token.dep\_ = 'det'

TOKEN: only

=====

token.tag\_ = 'JJ'

token.head.text = 'body'

token.dep\_ = 'amod'

TOKEN: known

=====

token.tag\_ = 'VBN'

token.head.text = 'body'

token.dep\_ = 'amod'

TOKEN: celestial

=====

token.tag\_ = 'JJ'

token.head.text = 'body'

token.dep\_ = 'amod'

TOKEN: body

=====

token.tag\_ = 'NN'

token.head.text = 'planet'

token.dep\_ = 'conj'

TOKEN: to

=====

token.tag\_ = 'TO'

token.head.text = 'support'

token.dep\_ = 'aux'

TOKEN: support

=====

token.tag\_ = 'VB'

token.head.text = 'is'

token.dep\_ = 'advcl'

TOKEN: life

=====

token.tag\_ = 'NN'

token.head.text = 'support'

token.dep\_ = 'dobj'

TOKEN: .

=====

token.tag\_ = '.'

token.head.text = 'is'

token.dep\_ = 'punct'

TOKEN:

=====

token.tag\_ = '\_SP'

token.head.text = '.'

token.dep\_ = 'dep'

TOKEN: With

=====

token.tag\_ = 'IN'

token.head.text = 'is'

token.dep\_ = 'prep'

TOKEN: a

=====

token.tag\_ = 'DT'

token.head.text = 'range'

token.dep\_ = 'det'

TOKEN: diverse

=====

token.tag\_ = 'JJ'

token.head.text = 'range'

token.dep\_ = 'amod'

TOKEN: range

=====

token.tag\_ = 'NN'

token.head.text = 'With'

token.dep\_ = 'pobj'

TOKEN: of

=====

token.tag\_ = 'IN'

token.head.text = 'range'

token.dep\_ = 'prep'

TOKEN: ecosystems

=====

token.tag\_ = 'NNS'

token.head.text = 'of'

token.dep\_ = 'pobj'

TOKEN: ,

=====

token.tag\_ = ','

token.head.text = 'is'

token.dep\_ = 'punct'

TOKEN: it

=====

token.tag\_ = 'PRP'

token.head.text = 'is'

token.dep\_ = 'nsubj'

TOKEN: is

=====

token.tag\_ = 'VBZ'

token.head.text = 'is'

token.dep\_ = 'ROOT'

TOKEN: home

=====

token.tag\_ = 'RB'

token.head.text = 'is'

token.dep\_ = 'advmod'

TOKEN: to

=====

token.tag\_ = 'IN'

token.head.text = 'home'

token.dep\_ = 'prep'

TOKEN: a

=====

token.tag\_ = 'DT'

token.head.text = 'array'

token.dep\_ = 'det'

TOKEN: vast

=====

token.tag\_ = 'JJ'

token.head.text = 'array'

token.dep\_ = 'amod'

TOKEN: array

=====

token.tag\_ = 'NN'

token.head.text = 'to'

token.dep\_ = 'pobj'

TOKEN: of

=====

token.tag\_ = 'IN'

token.head.text = 'array'

token.dep\_ = 'prep'

TOKEN: plant

=====

token.tag\_ = 'NN'

token.head.text = 'species'

token.dep\_ = 'nmod'

TOKEN: and

=====

token.tag\_ = 'CC'

token.head.text = 'plant'

token.dep\_ = 'cc'

TOKEN: animal

=====

token.tag\_ = 'NN'

token.head.text = 'plant'

token.dep\_ = 'conj'

TOKEN: species

=====

token.tag\_ = 'NNS'

token.head.text = 'of'

token.dep\_ = 'pobj'

TOKEN: ,

=====

token.tag\_ = ','

token.head.text = 'species'

token.dep\_ = 'punct'

TOKEN: including

=====

token.tag\_ = 'VBG'

token.head.text = 'species'

token.dep\_ = 'prep'

TOKEN: humans

=====

token.tag\_ = 'NNS'

token.head.text = 'including'

token.dep\_ = 'pobj'

TOKEN: .

=====

token.tag\_ = '.'

token.head.text = 'is'

token.dep\_ = 'punct'

TOKEN:

=====

token.tag\_ = '\_SP'

token.head.text = '.'

token.dep\_ = 'dep'

TOKEN: Earth

=====

token.tag\_ = 'NNP'

token.head.text = 'atmosphere'

token.dep\_ = 'poss'

TOKEN: 's

=====

token.tag\_ = 'POS'

token.head.text = 'Earth'

token.dep\_ = 'case'

TOKEN: atmosphere

=====

token.tag\_ = 'NN'

token.head.text = 'sustains'

token.dep\_ = 'nsubj'

TOKEN: ,

=====

token.tag\_ = ','

token.head.text = 'atmosphere'

token.dep\_ = 'punct'

TOKEN: composed

=====

token.tag\_ = 'VBN'

token.head.text = 'atmosphere'

token.dep\_ = 'acl'

TOKEN: mainly

=====

token.tag\_ = 'RB'

token.head.text = 'of'

token.dep\_ = 'advmod'

TOKEN: of

=====

token.tag\_ = 'IN'

token.head.text = 'composed'

token.dep\_ = 'prep'

TOKEN: nitrogen

=====

token.tag\_ = 'NN'

token.head.text = 'of'

token.dep\_ = 'pobj'

TOKEN: and

=====

token.tag\_ = 'CC'

token.head.text = 'nitrogen'

token.dep\_ = 'cc'

TOKEN: oxygen

=====

token.tag\_ = 'NN'

token.head.text = 'nitrogen'

token.dep\_ = 'conj'

TOKEN: ,

=====

token.tag\_ = ','

token.head.text = 'atmosphere'

token.dep\_ = 'punct'

TOKEN: sustains

=====

token.tag\_ = 'VBZ'

token.head.text = 'sustains'

token.dep\_ = 'ROOT'

TOKEN: life

=====

token.tag\_ = 'NN'

token.head.text = 'sustains'

token.dep\_ = 'dobj'

TOKEN: by

=====

token.tag\_ = 'IN'

token.head.text = 'sustains'

token.dep\_ = 'prep'

TOKEN: providing

=====

token.tag\_ = 'VBG'

token.head.text = 'by'

token.dep\_ = 'pcomp'

TOKEN: the

=====

token.tag\_ = 'DT'

token.head.text = 'conditions'

token.dep\_ = 'det'

TOKEN: necessary

=====

token.tag\_ = 'JJ'

token.head.text = 'conditions'

token.dep\_ = 'amod'

TOKEN: conditions

=====

token.tag\_ = 'NNS'

token.head.text = 'providing'

token.dep\_ = 'dobj'

TOKEN: for

=====

token.tag\_ = 'IN'

token.head.text = 'conditions'

token.dep\_ = 'prep'

TOKEN: biological

=====

token.tag\_ = 'JJ'

token.head.text = 'processes'

token.dep\_ = 'amod'

TOKEN: processes

=====

token.tag\_ = 'NNS'

token.head.text = 'for'

token.dep\_ = 'pobj'

TOKEN: to

=====

token.tag\_ = 'TO'

token.head.text = 'thrive'

token.dep\_ = 'aux'

TOKEN: thrive

=====

token.tag\_ = 'VB'

token.head.text = 'conditions'

token.dep\_ = 'relcl'

TOKEN: .

=====

token.tag\_ = '.'

token.head.text = 'sustains'

token.dep\_ = 'punct'

TOKEN:

=====

token.tag\_ = '\_SP'

token.head.text = '.'

token.dep\_ = 'dep'

'''