Name : J Viswaksena

RollNo : AM.EN.U4AIE21035

21AIE314 lab1

** Assignment1**

Write a paragraph about any large language models and save this as llm.text file.

Read and display this file with Python program.

Extract sentences and words from this file.

If there are any stop-words, remove them.

Identify any 10 most frequently used tokens.

Identify the stems and lemma for those words.

Check for any words that are not lemmatised.

Count number of stop words and non stop-words.

Translate any ten of these words into your native language.

If there are any spelling mistakes, remove the identified ones.

Write a paragraph about any large language models and save this as llm.text file.

```
from google.colab import files
uploaded = files.upload()
```

```
Choose files llm.txt
• llm.txt(text/plain) - 1222 bytes, last modified: 28/02/2024 - 100% done
Saving llm.txt to llm.txt
```

Read and display this file with Python program.

```
f = open('llm.txt','r',encoding='utf8',errors = 'ignore')
lines = f.read()
print(lines)
```

```
LL.M., short for Master of Laws, stands as an eminent postgraduate degree targeted at individuals with a foundational ba
The primary impetus behind pursuing an LL.M. lies in the opportunity it presents for specialization beyond the scope cov
A hallmark of LL.M. programs is their flexiility and diversity, with many universities offering a wide array of speciali
```

Extract sentences from this file.

```
import nltk
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
True
```

```
# Sentence tokenization
tokens = nltk.sent_tokenize(lines)
print(len(tokens))
```

```
12
```

Extract words from this file.

```
# Word Tokenization
tokens = nltk.word_tokenize(lines)
print(len(tokens))
```

```
203
```

Count number of stop words and non stop-words.

```
non_stopwords = []
stop_word = []

for word2 in tokens:
  if word2 in stop_words:
    stop_word.append(word2)

for word in tokens:
  if word not in stop_words:
    # print(word)
    non_stopwords.append(word)

print("Stop Words: ",len(stop_words))
print("Non Stop Words: ",len(non_stopwords))
```

```
Stop Words:  183
Non Stop Words:  119
```

If there are any stop-words, remove them.

```
nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
True
```

```
from nltk.corpus import stopwords
stop_words = stopwords.words('english')
print(len(stop_words))
```

```
179
```

```
stop_words.extend([',',';','also','.'])
```

```
newlist = []
for word in tokens:
  if word not in stop_words:
    # print(word)
    newlist.append(word)
print(newlist)
print(*newlist)
```

```
['LL.M.', 'short', 'Master', 'Laws', 'stands', 'eminent', 'postgraduate', 'degree', 'targeted', 'individuals', 'foundati
LL.M. short Master Laws stands eminent postgraduate degree targeted individuals foundational background law substantial
```

Identify any 10 most frequently used tokens.

```
from nltk.probability import FreqDist
fdist = FreqDist(newlist)
print(fdist)
```

```
<FreqDist with 104 samples and 119 outcomes>
```

```
fdist.most_common(10)
```

```
[('law', 6),
 ('legal', 4),
 ('LL.M', 4),
 ('degree', 2),
 ('international', 2),
 ('The', 2),
 ('specialization', 2),
 ('LL.M.', 1),
```

```
 ('short', 1),
 ('Master', 1)]
```

```
fdist.max()
```

```
'law'
```

Identify the stems for those words.

```
nltk.download('wordnet')
```

```
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
True
```

```
import nltk
from nltk.stem import PorterStemmer
from nltk.stem import LancasterStemmer
```

```
porter = PorterStemmer()
lancaster=LancasterStemmer()
```

```
print("--------------------Porter Stemmer--------------------")
for word in newlist:
  print(porter.stem(word))
# #proide a word to be stemmed
# print("Porter Stemmer")

# print(porter.stem(newlist))
# print("Lancaster Stemmer")
# print(lancaster.stem(newlist))
```

```
      crimin
      justic
      's
      like
      ll.m
      program
      craft
      meet
      obiect

print("--------------------Lancaster Stemmer--------------------")
for word in newlist:
  print(lancaster.stem(word))
```

```
      intern
      arbit
      the
      ll.m
      curricul
      entail
      comprehend
      expl
      leg
      the
      principl
      pract
      apply
      pertin
      chos
      field
      study
      a
      hallmark
      ll.m
      program
      flexiil
      divers
      many
      univers
      off
      wid
      array
      spec
      opt
      thi
      flex
      en
      stud
      tail
      sudy
      align
      profess
      aspir
      person
      interest
      wheth
      on
      seek
      car
      corp
      law
      environ
      law
      crimin
      just
      's
      lik
      ll.m
      program
      craft
      meet
      object
```

Identify the lemma for those words.

```
from nltk.stem import WordNetLemmatizer

lemmatizer = WordNetLemmatizer()
```

```
lemmatized_words = []

for word in newlist:
    lemma = lemmatizer.lemmatize(word)
    print(word, "->", lemma)
    lemmatized_words.append(lemma)

print('Lemmatized words:', lemmatized_words)
```

```
arbitration -> arbitration
The -> The
LL.M -> LL.M
curriculum -> curriculum
entails -> entail
comprehensive -> comprehensive
exploration -> exploration
legal -> legal
theories -> theory
principles -> principle
practical -> practical
applications -> application
pertinent -> pertinent
chosen -> chosen
field -> field
study -> study
A -> A
hallmark -> hallmark
LL.M -> LL.M
programs -> program
flexiility -> flexiility
diversity -> diversity
many -> many
universities -> university
offering -> offering
wide -> wide
array -> array
specialization -> specialization
options -> option
This -> This
flexibility -> flexibility
enables -> enables
students -> student
tailor -> tailor
sudies -> sudies
alignment -> alignment
professional -> professional
aspirations -> aspiration
personal -> personal
interests -> interest
Whether -> Whether
one -> one
seeks -> seek
career -> career
corporate -> corporate
law -> law
environmental -> environmental
law -> law
criminal -> criminal
justice -> justice
's -> 's
likely -> likely
LL.M -> LL.M
program -> program
crafted -> crafted
meet -> meet
objectives -> objective
Lemmatized words: ['LL.M.', 'short', 'Master', 'Laws', 'stand', 'eminent', 'postgraduate', 'degree', 'targeted', 'indivi
```

Check for any words that are not lemmatised.

```
non_lemmatized_words = []

for word, lemma in zip(newlist, lemmatized_words):
    if word != lemma:
        non_lemmatized_words.append(word)

print("Non-lemmatized words:", non_lemmatized_words)
```

```
Non-lemmatized words: ['stands', 'individuals', 'serves', 'professionals', 'domains', 'systems', 'lies', 'presents', 'sp
```

Translate any ten of these words into your native language.

```
pip install translate
```

```
Requirement already satisfied: translate in /usr/local/lib/python3.10/dist-packages (3.6.1)
Requirement already satisfied: click in /usr/local/lib/python3.10/dist-packages (from translate) (8.1.7)
Requirement already satisfied: lxml in /usr/local/lib/python3.10/dist-packages (from translate) (4.9.4)
Requirement already satisfied: requests in /usr/local/lib/python3.10/dist-packages (from translate) (2.31.0)
Requirement already satisfied: libretranslatepy==2.1.1 in /usr/local/lib/python3.10/dist-packages (from translate) (2.1.
Requirement already satisfied: charset-normalizer<4,>=2 in /usr/local/lib/python3.10/dist-packages (from requests->trans
Requirement already satisfied: idna<4,>=2.5 in /usr/local/lib/python3.10/dist-packages (from requests->translate) (3.6)
Requirement already satisfied: urllib3<3,>=1.21.1 in /usr/local/lib/python3.10/dist-packages (from requests->translate)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.10/dist-packages (from requests->translate)
```

```
from translate import Translator

translator= Translator(to_lang="te")

for j in newlist[:10]:
  translation = translator.translate(j)
  print(j,'-->',translation)
```

```
LL.M. --> LL.M.
short --> పొట్టి
Master --> మాస్టర్
Laws --> చట్టాలు
stands --> స్టాండ్  లు
eminent --> ప్రముఖ
postgraduate --> పోస్ట్ గ్రాడ్యుయేట్
degree --> patta

targeted --> లక్ష్యంగా
individuals --> వ్యక్తులు
```

If there are any spelling mistakes, remove the identified ones.

```
pip install pyspellchecker
```

```
Requirement already satisfied: pyspellchecker in /usr/local/lib/python3.10/dist-packages (0.8.1)
```

```
from hashlib import new
from spellchecker import SpellChecker

spell = SpellChecker()
c=0
print("Before Spell Check: ",len(newlist))
for k in newlist:
  # misspelled = spell.unknown(k)
  # print(misspelled)
  # print(spell.correction(misspelled))
  # # Get a list of `likely` options
  # print(spell.candidates(misspelled))
  if spell.correction(k) != k:
    c+=1
    print(k)
    newlist.remove(k)
print('Number of misspelled words',c)
print('After Spell Check: ',len(newlist))
```

```
Before Spell Check:  119
LL.M.
legl
LL.M
coverd
LL.M
LL.M
flexiility
sudies
's
LL.M
Number of misspelled words 10
After Spell Check:  109
```

```
Start coding or generate with AI.
```