#text preprocessing methods

import nltk

from nltk.tokenize import word\_tokenize

tokens=word\_tokenize("Hello this is assignment of data science")

nltk.download('stopwords')

print(tokens)

#to remove stopwords from the above sentence

from nltk.corpus import stopwords

stop\_words=set(stopwords.words('english'))

tokens = [w for w in tokens if not w in stop\_words] #words writte in brackets are a short version of 2 for loops

print(tokens)

#example for stemming

from nltk.stem import PorterStemmer

nltk.download("punkt")

# Initialize Python porter stemmer

ps = PorterStemmer()

# Example inflections to reduce

example\_words = ["program","programming","programer","programs","programmed"]

# Perform stemming

print("{0:20}{1:20}".format("--Word--","--Stem--"))

for word in example\_words:

print("{0:20}{1:20}".format(word, ps.stem(word)))

#example for lemmatization

from nltk.stem import WordNetLemmatizer

nltk.download("wordnet")

nltk.download("omw-1.4")

# Initialize wordnet lemmatizer

wnl = WordNetLemmatizer()

# Example inflections to reduce

example\_words = ["program","programming","programer","programs","programmed"]

# Perform lemmatization

print("{0:20}{1:20}".format("--Word--","--Lemma--"))

for word in example\_words:

print("{0:20}{1:20}".format(word, wnl.lemmatize(word, pos="v")))

#tfidf for a sample

import pandas as pd

import sklearn as sk

import math

first\_sentence = "He is an intelligent boy"

second\_sentence = "Everyone is not intelligent"

third\_sentence="She is an intelligent girl "

first\_sentence = first\_sentence.split(" ")

second\_sentence = second\_sentence.split(" ")

third\_sentence = third\_sentence.split(" ")

total= set(first\_sentence).union(set(second\_sentence)).union(set(third\_sentence))

print(total)

#counting total words and putting them in dataframe

wordDictA = dict.fromkeys(total, 0)

wordDictB = dict.fromkeys(total, 0)

wordDictC=dict.fromkeys(total, 0)

for word in first\_sentence:

wordDictA[word]+=1

for word in second\_sentence:

wordDictB[word]+=1

for word in third\_sentence:

wordDictC[word]+=1

pd.DataFrame([wordDictA, wordDictB,wordDictC])

#calculatin TF function

def computeTF(wordDict, doc):

tfDict = {}

corpusCount = len(doc)

for word, count in wordDict.items():

tfDict[word] = count/float(corpusCount)

return(tfDict)#running our sentences through the tf function:

tfFirst = computeTF(wordDictA, first\_sentence)

tfSecond = computeTF(wordDictB, second\_sentence)

tfThird = computeTF(wordDictC, third\_sentence)#Converting to dataframe for visualization

tf = pd.DataFrame([tfFirst, tfSecond,tfThird])

print(tf)

stop\_words = set(stopwords.words('english'))

filtered\_sentence = [w for w in wordDictA if not w in stop\_words]

print(filtered\_sentence)

def computeIDF(docList):

idfDict = {}

N = len(docList)

idfDict = dict.fromkeys(docList[0].keys(), 0)

for word, val in idfDict.items():

idfDict[word] = math.log10(N / (float(val) + 1))

return(idfDict)#inputing our sentences in the log file

idfs = computeIDF([wordDictA, wordDictB,wordDictC])

print(idfs)

def computeTFIDF(tfBow, idfs):

tfidf = {}

for word, val in tfBow.items():

tfidf[word] = val\*idfs[word]

return(tfidf)

#running our two sentences through the IDF:

idfFirst = computeTFIDF(tfFirst, idfs)

idfSecond = computeTFIDF(tfSecond, idfs)

idfThird = computeTFIDF(tfThird, idfs)

#putting it in a dataframe

idf= pd.DataFrame([idfFirst, idfSecond,idfThird])

print(idf)