#CPSC475 Dr.DePalma Fall 2016 asgn5

#N-Gram Generator

#Sebastian Vargas

#id: avargas

#

#To run on linux, go to your terminal then go to the directory in which

# this program is stored. Then type "python asgn5.py" in the command line

#This should execute the program

import random

from random import \*

import nltk

from nltk.corpus import brown

from collections import Counter

def main():

news = brown.sents(categories='editorial')

#news1 is a list of all the individual words

news1 = brown.words(categories='editorial')

#news2 is a tokenized list of news

news2 = [len(news)]

#news3 makes a list of all the individual words including <s> </s>

news3 = [len(news1)+1]

#Initializes dictionaries and strings

#

#

unigram = {}

unigramString = []

bigram = {}

bigramString = []

trigram = {}

trigramString = []

quadgram = {}

quadgramString = []

#Tokenizing the corpus

Tokenize(news, news2)

#Puts words into a list

makeNewList(news2,news3)

#Produces Unigrams

countUnigram = countAllWords(news3)

setUpUnigram(countUnigram, news3,unigram)

printUnigrams(news3,unigram,unigramString)

#Produces Bigrams

bigramArray = find\_ngrams(news3, 2)

countBigram = countAllBigrams(bigramArray)

setUpBigram(countBigram, bigramArray,bigram)

printBigrams(bigramArray,bigram,bigramString)

#Produces Trigrams

trigramArray = find\_ngrams(news3, 3)

countTrigram = countAllTrigrams(trigramArray)

setUpTrigram(countTrigram, trigramArray,trigram)

printTrigrams(trigramArray,trigram,trigramString)

#Produces Quadgrams

quadgramArray = find\_ngrams(news3, 4)

countQuadgram = countAllQuadgrams(quadgramArray)

setUpQuadgram(countQuadgram, quadgramArray,quadgram)

printQuadgrams(quadgramArray,quadgram,quadgramString)

#Prints Unigrams

def printUnigrams(news3,unigram,unigramString):

print("Unigrams: \n")

print("1: ")

unigramGenerator(news3,unigram,unigramString)

printSentences(unigramString)

del unigramString[:]

print("2: ")

unigramGenerator(news3,unigram,unigramString)

printSentences(unigramString)

del unigramString[:]

print("3: ")

unigramGenerator(news3,unigram,unigramString)

printSentences(unigramString)

del unigramString[:]

print("4: ")

unigramGenerator(news3,unigram,unigramString)

printSentences(unigramString)

del unigramString[:]

print("5: ")

unigramGenerator(news3,unigram,unigramString)

printSentences(unigramString)

print("\n")

#Prints Bigrams

def printBigrams(bigramArray,bigram,bigramString):

print("Bigrams: \n")

print("1: ")

bigramGenerator(bigramArray,bigram,bigramString)

printSentences(bigramString)

del bigramString[:]

print("2: ")

bigramGenerator(bigramArray,bigram,bigramString)

printSentences(bigramString)

del bigramString[:]

print("3: ")

bigramGenerator(bigramArray,bigram,bigramString)

printSentences(bigramString)

del bigramString[:]

print("4: ")

bigramGenerator(bigramArray,bigram,bigramString)

printSentences(bigramString)

del bigramString[:]

print("5: ")

bigramGenerator(bigramArray,bigram,bigramString)

printSentences(bigramString)

del bigramString[:]

print("\n")

#Prints trigrams

def printTrigrams(trigramArray,trigram,trigramString):

print("Trigrams: \n")

print("1: ")

trigramGenerator(trigramArray,trigram,trigramString)

printSentences(trigramString)

del trigramString[:]

print("2: ")

trigramGenerator(trigramArray,trigram,trigramString)

printSentences(trigramString)

del trigramString[:]

print("3: ")

trigramGenerator(trigramArray,trigram,trigramString)

printSentences(trigramString)

del trigramString[:]

print("4: ")

trigramGenerator(trigramArray,trigram,trigramString)

printSentences(trigramString)

del trigramString[:]

print("5: ")

trigramGenerator(trigramArray,trigram,trigramString)

printSentences(trigramString)

del trigramString[:]

print("\n")

#Prints qudgrams

def printQuadgrams(quadgramArray,quadgram,quadgramString):

print("Quadgrams: \n")

print("1: ")

quadgramGenerator(quadgramArray,quadgram,quadgramString)

printSentences(quadgramString)

del quadgramString[:]

print("2: ")

quadgramGenerator(quadgramArray,quadgram,quadgramString)

printSentences(quadgramString)

del quadgramString[:]

print("3: ")

quadgramGenerator(quadgramArray,quadgram,quadgramString)

printSentences(quadgramString)

del quadgramString[:]

print("4: ")

quadgramGenerator(quadgramArray,quadgram,quadgramString)

printSentences(quadgramString)

del quadgramString[:]

print("5: ")

quadgramGenerator(quadgramArray,quadgram,quadgramString)

printSentences(quadgramString)

del quadgramString[:]

print("\n")

###############################

# Tokenize(news)

# takes a list of lists and goes through it, adding

# <s> to the start and </s> to the end

# while removing all periods

def Tokenize(news, news2):

for i in range(len(news)):

news[i].insert(0,u"<s>")

news[i].append(u"</s>")

while "." in news[i]: news[i].remove(".")

news2.append(news[i])

news2.pop(0)

################################

# makeNewList(news2, news3)

# takes a concantenated corpus view and

# inserts it into a list

#

def makeNewList(news2, news3):

for i in range(len(news2)-1):

for w in news2[i]:

news3.append(w)

news3.pop(0)

################################

# counts all the words for a unigram

#

def countAllWords(news3):

return Counter(news3)

################################

# counts the number of bigrams

#

def countAllBigrams(bigramArray):

return Counter(bigramArray)

################################

# counts the number of trigrams

#

def countAllTrigrams(trigramArray):

return Counter(trigramArray)

################################

# counts the number of quadgrams

#

def countAllQuadgrams(quadgramArray):

return Counter(quadgramArray)

################################

# setUpUnigram stores all the unigrams

# into a dictionary unigram with the value

# as the word and the key is the relative frequency

#

def setUpUnigram(count, news3,unigram):

counter = 0

for w in count:

counter = counter + count[w]

unigram[counter] = w

################################

# unigramGenerator produces a string

# of 8 random unigrams

#

#

def unigramGenerator(unigramArray,unigram,unigramString):

for x in range(8):

x = randint(1,len(unigramArray))

while x not in unigram:

x = x - 1

unigramString.append(unigram[x].encode("ascii"))

################################

# setUpBigram stores all the bigrams

# into a dictionary bigram with the value

# as the bigram and the key is the relative frequency

#

def setUpBigram(bigramCount, bigramArray,bigram):

counter = 0

for w in bigramCount:

counter = counter + bigramCount[w]

bigram[counter] = w

###############################

# generates a string of random bigrams that starts

# with a bigram starting with "<s>" and ends with

# a bigram containing "</s>"

#

def bigramGenerator(bigramArray,bigram,bigramString):

x = randint(1,len(bigramArray))

while x not in bigram or bigram[x][0] != "<s>":

x = randint(1,len(bigramArray))

while x not in bigram:

x = x - 1

bigramString.append(bigram[x][0].encode("ascii"))

bigramString.append(bigram[x][1].encode("ascii"))

while bigram[x][1] != "</s>":

x = randint(1,len(bigramArray))

while x not in bigram or bigram[x][0]== "<s>":

x = x - 1

bigramString.append(bigram[x][0].encode("ascii"))

bigramString.append(bigram[x][1].encode("ascii"))

################################

# setUpTrigram stores all the trigrams

# into a dictionary trigram with the value

# as the trigram and the key is the relative frequency

#

def setUpTrigram(trigramCount, trigramArray, trigram):

counter = 0

for w in trigramCount:

#print[w]

counter = counter + trigramCount[w]

trigram[counter] = w

###############################

# generates a string of random trigrams that starts

# with a trigram starting with "<s>" and ends with

# a trigram containing "</s>"

#

def trigramGenerator(trigramArray,trigram,trigramString):

x = randint(1,len(trigramArray))

while x not in trigram or trigram[x][0] != "<s>":

x = randint(1,len(trigramArray))

while x not in trigram:

x = x - 1

trigramString.append(trigram[x][0].encode("ascii"))

trigramString.append(trigram[x][1].encode("ascii"))

trigramString.append(trigram[x][2].encode("ascii"))

while trigram[x][2] != "</s>" and trigram[x][1] != "</s>":

x = randint(1,len(trigramArray))

while x not in trigram or "<s>" in trigram[x]:

#trigram[x][0] == "<s>" or trigram[x][1] == "<s>" or trigram[x][2] == "<s>":

x = x - 1

trigramString.append(trigram[x][0].encode("ascii"))

trigramString.append(trigram[x][1].encode("ascii"))

trigramString.append(trigram[x][2].encode("ascii"))

################################

# setUpQuadgram stores all the quadgrams

# into a dictionary quadgram with the value

# as the quadgram and the key is the relative frequency

#

def setUpQuadgram(quadgramCount, quadgramArray, quadgram):

counter = 0

for w in quadgramCount:

counter = counter + quadgramCount[w]

quadgram[counter] = w

###############################

# generates a string of random quadgrams that starts

# with a quadgram starting with "<s>" and ends with

# a quadgram containing "</s>"

#

def quadgramGenerator(quadgramArray,quadgram,quadgramString):

x = randint(1,len(quadgramArray))

while x not in quadgram or quadgram[x][0] != "<s>":

x = randint(1,len(quadgramArray))

while x not in quadgram:

x = x - 1

quadgramString.append(quadgram[x][0].encode("ascii"))

quadgramString.append(quadgram[x][1].encode("ascii"))

quadgramString.append(quadgram[x][2].encode("ascii"))

quadgramString.append(quadgram[x][3].encode("ascii"))

while quadgram[x][2] != "</s>" and quadgram[x][1] != "</s>" and quadgram[x][3] != "</s>":

x = randint(1,len(quadgramArray))

while x not in quadgram or "<s>" in quadgram[x]:

#quadgram[x][0] == "<s>" or quadgram[x][1] == "<s>"

#or quadgram[x][2] == "<s>" or quadgram[x][3] == "<s>":

x = x - 1

quadgramString.append(quadgram[x][0].encode("ascii"))

quadgramString.append(quadgram[x][1].encode("ascii"))

quadgramString.append(quadgram[x][2].encode("ascii"))

quadgramString.append(quadgram[x][3].encode("ascii"))

##########################

# prints a list as sentence,

# replacing </s> with periods and removing <s>

#

def printSentences(string):

string = [x for x in string if x != "<s>"]

string = ["." if x=="</s>" else x for x in string]

print(' '.join(string))

########################################

# creates a list of n-grams

#

def find\_ngrams(input\_list, n):

return zip(\*[input\_list[i:] for i in range(n)])

main()