import nltk

from nltk import word\_tokenize

from nltk.util import ngrams

from collections import Counter

from collections import Counter

import string

# Read in the Lyrics

f = open("lyrics.txt", 'r',encoding="utf8")

lines = []

for lyric in f:

lyric = lyric.lower()

lyric = '<BOL><BOL>'lyric + '<EOL>'

lines.extend(lyric.split('\n'))

print(lines)

#Make n grams using nltk

lyrics = ' '.join(lines)

token = nltk.word\_tokenize(lyrics)

bigrams = list(ngrams(token, 2))

trigrams = list(ngrams(token,3))

numTrigrams = (Counter(trigrams))

numBigrams = (Counter(bigrams))

# Make a dictionary containing the probabilities

# For example, given the two word phrase 'new york',

# what is the probability for the word 'city' to come next

# it will stored as

# {'new york': ['city', .5], ['state', .5]

# 'rose is': ['pink', .333], ['yellow', .333], ['red'], .333}

def makeDictionary(bigrams, trigrams, numBigrams, numTrigrams):

dictionary = {}

for i in range(len(bigrams)):

thirdWordList = []

for j in range(len(trigrams)):

k = 0

if(bigrams[i][k] == trigrams[j][k] and bigrams[i][k+1] == trigrams[j][k+1]):

bigramCount = numBigrams[bigrams[i]]

trigramCount = numTrigrams[trigrams[j]]

prob = float(trigramCount/bigramCount)

thirdWordTuple = (trigrams[j][k+2], prob)

if(len(thirdWordList) == 0):

thirdWordList.append(thirdWordTuple)

elif (len(thirdWordList) > 0 and isDuplicate(thirdWordList, trigrams[j][k+2]) == False):

thirdWordList.append(thirdWordTuple)

dictionary[bigrams[i]] = thirdWordList

return dictionary

# helper function for make dictionary

# checks if a value already exists in the dictionary

def isDuplicate(thirdWordList, thirdWord):

duplicate = False;

for i in range(len(thirdWordList)):

j = 0

if thirdWordList[i][j] == thirdWord:

duplicate = True;

return duplicate

dictionary = makeDictionary(bigrams, trigrams, numBigrams, numTrigrams)

#print("dictionary")

#for key, value in dictionary.items():

# print(key, value)