#!/usr/bin/python

# -\*- encoding: utf-8 -\*-

#Zack Larsen, CSC 594 (NLP) HW #2

from \_\_future\_\_ import division

import re, pprint, nltk, sys, datetime, os, collections

#from sklearn import \*

#from nltk.probability import \*

from nltk import bigrams, trigrams, word\_tokenize, sent\_tokenize

from nltk.corpus import inaugural

#print inaugural.fileids()

washington = inaugural.raw('1789-Washington.txt')

jefferson = inaugural.raw('1805-Jefferson.txt')

text = sys.stdin.read()

#print text     #or, print text[:100]

######### Number of sentences ##########

try:

    sent\_tokenize\_list = sent\_tokenize(text)

except:

    sent\_tokenize\_list = sent\_tokenize(text.decode('utf-8'))

#print sent\_tokenize\_list[:1]

num\_sentences = len(sent\_tokenize\_list)

############### Tokenize ################

tokens = []

for sentence in sent\_tokenize\_list:

    token\_list=word\_tokenize(sentence)

    tokens.append(token\_list)

#print tokens[:4]

#######Add in the <s> and <\s> to the sentence list########

for sentence in tokens:

    sentence.insert(0,'<s>')

    sentence.append('</s>')

#print tokens[:4]

############### Number of types ##########

counts = {}

for sentence in tokens:

    for token in sentence:

        counts[token] = counts.get(token,0)+1

total = sum(counts.values())

#print total

############## Print the count of each token ###############

sorted\_counts = sorted(counts.items(), key=lambda x: x[1])    # ,reverse = True)

#for i in range(0,len(sorted\_counts)):

#    print sorted\_counts[i][0], sorted\_counts[i][1]

token\_list = []

for sentence in tokens:

    for token in sentence:

        token\_list.append(token)

#token\_list

#########Construct unigram model with LaPlace smoothing##########

def unigram(tokens):

    model = collections.defaultdict(lambda: 0.01)

    for f in tokens:

        try:

            model[f] += 1

        except KeyError:

            model[f] = 1

            continue

    for word in model:

        model[word] = model[word]/float(len(model))

    return model

unigrams = unigram(token\_list)

#print unigrams.values()

#############Generate N-grams##############

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

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

bigrams = find\_ngrams(token\_list,2)

#print bigrams[:60]

bigramcounts = {}

for bigram in bigrams:

    bigramcounts[bigram] = bigramcounts.get(bigram,0)+1

bigramtotal = sum(bigramcounts.values())

#print bigramcounts   #.values()

#print bigramtotal

#############Store bigram probabilities in a dictionary#############

bigram\_probs = {}

for bigram in bigrams:

    bigram\_probs[bigram] = bigramcounts[bigram]/bigramtotal

print bigram\_probs

#print unigrams

############ Store the unigram and bigram probabilities persistently in languagemodel.db #########

import shelve

bigrams = bigram\_probs

d = shelve.open('bigrams.db')

d['bigrams'] = bigrams

d.close()

d = shelve.open('unigrams.db')

d['unigrams'] = unigrams

d.close()