"""

Cody Valle and Jinous Esmaeili

Natural Language Processing Assignment 5: N-Grams of the Brown Corpus

To use:

You must have the Brown corpus downloaded from NLTK already. Run the code 'nltk.download('brown')' to have it available.

This program takes no arguments. Simply run the file.

NB: Creating a unigram sentence takes an excessively long time to complete. Reason unknown, but attempting to fix. That’s the reason the ‘print make\_ngram\_sentence(unigrams, unistarts)’ line in main is commented.

"""

import decimal

from decimal import Decimal

import random

"""

Returns a random number between 0(inclusive) and 1(exclusive) with currently set decimal precision

Code from: http://stackoverflow.com/questions/31595594/python-random-float-with-limited-decimal-digits

"""

def random\_float():

return Decimal(round(random.uniform(0, 1), decimal.getcontext().prec))

"""

Chooses a random N-Gram based on frequency

"""

def random\_ngram(grams):

# Get a random number between 0 and 1, with desired precision

percent = random\_float()

for key,val in grams.iteritems():

# Count down the probability until less than zero

percent -= val

if percent < 0:

return key

return None # Should never run

"""

Makes an N-Gram sentence that starts with '<s>' and ends with '</s>'

"""

def ngram\_sentence(grams, starts):

ret = ''

# Get the start of the sentence

choice = random.choice(starts)

# Append the gram to the return string

for word in choice:

ret += word + ' '

# Keep choosing until we get an end of sentence

choice = ['']

while choice[-1] != '</s>':

choice = random\_ngram(grams)

# Append the gram to the return string

for word in choice:

ret += word + ' '

return ret

"""

Makes a list of N-Grams from the passed in words

"""

def make\_ngrams(words, n):

grams = {}

# Create a gram starting at every word

for index in range(len(words) - n):

# Create the n-length gram

gram\_list = []

for offset in range(n):

gram\_list.append(words[index + offset])

#Convert to tuple to make it hashable

gram\_tuple = tuple(gram\_list)

grams[gram\_tuple] = grams.get(gram\_tuple,0) + 1 # Add and/or increment

# Calculate the probabilities

count = Decimal(len(grams))

for key,val in grams.iteritems():

grams[key] = Decimal(val) / count

# Get the starts of sentences

starts = []

for key,\_ in grams.iteritems():

if key[0] == '<s>':

starts.append(key)

return grams, starts

"""

Flattens a 2-dimensional array into a 1-dimensional array

"""

def flatten(table):

ret = []

for row in table:

ret += [element for element in row]

return ret

"""

Gets the cleaned sentences from the Brown corpus.

"""

def getSentences():

import re

import nltk

from nltk.corpus import brown

# Get the Brown corpus

sentences\_u = brown.sents(categories='editorial')

sentences = [] # Will hold the list of cleaned sentences

pat = re.compile('[A-Za-z]') # Matches strings with only letters

for sentence\_u in sentences\_u: # Clean every sentence

sentence = ['<s>'] # The start of every sentence

for word\_u in sentence\_u:

word = word\_u.encode('ascii','ignore') # Encode to ASCII

if pat.match(word): # Make sure it's just a word

sentence.append(word)

# We have t make sure we have words in this sentence

if sentence[-1] != '<s>':

sentences.append(sentence + ['</s>'])

return sentences

"""

The main function.

"""

def main():

decimal.getcontext().prec = 56 # Double the default precision

# Get the sentences from the Brown corpus

sentences = getSentences()

words = flatten(sentences) # Flatten them into a single list

# Get the N-Grams

unigrams, unistarts = make\_ngrams(words, 1)

bigrams, bistarts = make\_ngrams(words, 2)

trigrams, tristarts = make\_ngrams(words, 3)

quadgrams, quadstarts = make\_ngrams(words, 4)

# Make N-Gram sentences

#print 'Unigram Sentence:'

#print ngram\_sentence(unigrams, unistarts), ''

print 'Bigram Sentence:'

print ngram\_sentence(bigrams, bistarts), ''

print 'Trigram Sentence:'

print ngram\_sentence(trigrams, tristarts),''

print 'Quadgram Sentence:'

print ngram\_sentence(quadgrams, quadstarts), ''

if \_\_name\_\_ == '\_\_main\_\_':

main()

Sample Output:

Unigram Sentence:

<s> and it if as China's It is exhibit of decided to abolish the <s> First <s> The </s> <s> Even in of radioactive and acted the tax on that in The Mr. Hodges which has Hymn A who had million tons to how and West thyroid is <s> Undergraduates ball club city departments great issues it </s>

Bigram Sentence:

<s> Lucius who suggested to the lax municipal calls the do you is apparently cases later the effects many another of the most by and France the University going to one of soldier but propel the their allies our soft-heartedness at the of Chicago </s> <s> Less than </s> <s> corruptible journalists state in Bridges tunnels becomes intelligible <s> The </s> <s> Union would month's vacation great only In a controversial a and then be at a large the House on a citizen was live in Island towns put these might go and the community not </s> <s> and patience objects of Reports that these solutions </s> <s> it attack poetry and approach might sort of has been be able of the average American West Berlin's it has </s> <s> that this appropriate in secessionist president </s> <s> <s> The questions are cold war </s> <s> pressure a new translation Revolution Conservative the political ills </s>

Trigram Sentence:

<s> TASS from his childhood in a town such more benches and <s> Or certain and agreement by Russo-American honeymoon of in the </s>

Quadgram Sentence:

<s> If birds don't was the symbol of a day </s> <s> O.E.C.D. I.M.F. and others a politician by other critics has been calling reports on the schools and county committeemen CDC taxpayers have been impressed </s> <s> Many of he has been dead this optimism </s> <s> Communist listeners and readers </s> <s> The Senate's </s> <s> The charges bison they will never had in fact been the diagnosis at once Cowessett areas of Warwick tax cut </s> <s> </s> <s> Somewhere somebody around it only emphasized October although it's still will not have contaminated control over zoning franchises prominent Americans are now series called Adventures In The editorial concerned legislative bears </s> <s> And job for life </s>