

Student Name <- Rajashekar Mudigonda

CS585 Spring 2023 Programming Assignment #01

Due: Sunday, February 11, 2024, 11:59 PM CST

Points: 150

Part A [50 pts]:

Use Python's NLTK package along with the corpora:




- Brown,
- Reuters,

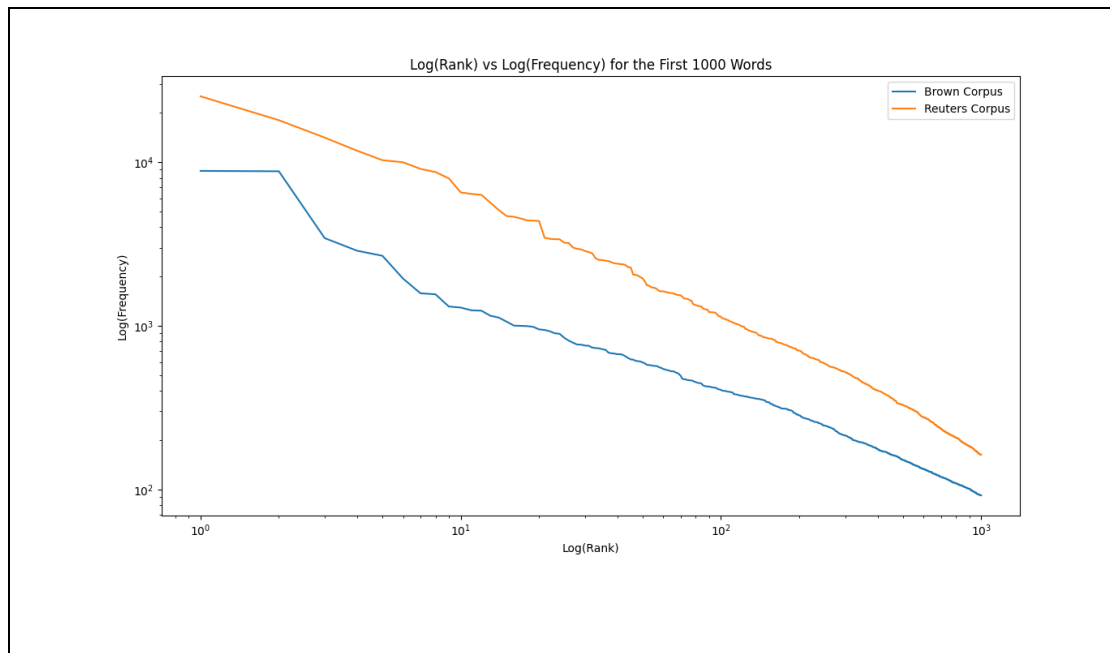
to:

- 1) [10 pts] obtain the word frequency distribution (after removing all stop words; use the `stopwords` corpora for that purpose) for BOTH corpora,
- 2) [10 pts] display a top ten (ranks 1 through 10) words for BOTH corpora on screen (also place them in the table below)

Top 10 words	
Brown	Reuters
' '	said
"	mln
--	vs
one	dlrs
would	000
said	1
could	pct
time	it
two	cts
may	2

- 3) [15 pts] generate **log(rank) vs log(frequency) plots** for the first 1000 (ranks 1 through 1000) words for BOTH corpora (you can use the matplotlib package or some other plotting package / tool). Place BOTH plots in the table below.

log(rank) vs log(frequency) plots	
Brown	Reuters
	
Did you observe anything interesting when comparing all plots? Write your comments below:	
	



4) [15 pts] use frequency counts obtained earlier to calculate the unigram occurrence probability for the TWO (“technical” and not technical) words. Use lowercasing first! **Display all relevant counts and probability on screen for BOTH corpora (also: enter final values in the table below).** It can be zero for some words.

“technical” / seldom used in casual conversation word (for example “adiabatic”)	
Brown	Reuters
algorithm – 1.7877925946055147e-06 protocol - 5.363377783816544e-06	stock - 0.00013295213917080564 inflation - 4.852267852949111e-06
Non- technical / casual / daily-use word (for example “dinner”)	
Brown	Reuters
House - 0.0007151170378422059 Tree - 0.00010011638529790881	chair - 6.404993565892828e-05 banana - 3.881814282359289e-06

Part B [50 pts]:

Use Python's NLTK package along with the Brown corpus for the following tasks:

1. [1 pts] Ask the user to enter a sentence S from a keyboard.
2. [1 pts] Apply lowercasing to S.
3. [45 pts] Calculate $P(S)$ assuming a 2-gram language model (**assume that probability of any bigram starting or ending a sentence is 0.25**)
4. [3 pts] Display the sentence S, list all the individual bigrams and their probabilities, and the final probability $P(S)$ on screen. It is fine if it is zero.

Part C [50 pts]:

Use Python's NLTK package along with the Brown corpus (after removing all stop words; use the `stopwords` corpora for that purpose) for the following tasks:

1. [1 pts] Start by asking the user for initial word/token W1. Apply lowercasing to W1 (and all future entries). If the word is NOT in the corpus offer two options:
 - a. ask again
 - b. QUIT
2. [45 pts] Assuming a 2-gram language model, a menu with TOP 3 "most likely to follow W1" words (according to the W1, NEXT WORD probability estimate). For example, if the user started with W1 = "good", the following could be displayed (**NOTE: I made up this selection and corresponding probability estimates**):

good ...

Which word should follow:

- 1) morning $P(\text{good morning}) = 0.25$
- 2) evening $P(\text{good evening}) = 0.15$
- 3) afternoon $P(\text{good afternoon}) = 0.14$
- 4) QUIT

Repeat (and add subsequent word choices to the "sentence") until user selects (4) and QUITs.

If the user picks a number other than 1,2,3, and 4, **assume user choice is (1).**