Student Name <- replace with your name

CS585 Spring 2023 Programming Assignment #01

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

Points: **150**

Instructions:

1. Place **all your deliverables (as described below) into a single ZIP** file named:

`LastName_FirstName_CS585_Programming01.zip`

2. Submit it to Blackboard Assignments section before the due date. **No late submissions will be accepted**.

Objectives:

1. (50 points) Perform basic word frequency distribution analysis for a text corpus.
2. (50 points) Calculate probability of a sentence.
3. (50 points) Language Model word prediction.

Deliverables:

Your submission should include:

- **Make sure your code is sufficiently commented! NO Jupyter Notebook files!**
- **Part A**: Python code file(s). Your py file should be named:

  `cs585_P01A_AXXXXXXXX.py`

  where AXXXXXXXX is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.
- **Part B**: Python code file(s). Your py file should be named:

  `cs585_P01B_AXXXXXXXX.py`

  where AXXXXXXXX is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.
- **Part C**: Python code file(s). Your py file should be named:

  `cs585_P01C_AXXXXXXXX.py`

  where AXXXXXXXX is your IIT A number (this is REQUIRED!). If your solution uses multiple files, makes sure that the main (the one that will be run to solve the problem) is named that way and others include your IIT A number in their names as well.

- this document with your results and conclusions. You should rename it to:

`LastName_FirstName_cs585_Programming01.doc or pdf`

MS WORD or PDF formats only, please.

Part A [50 pts]:

Use Python's NLTK package along with the corpora:

- Brown,
- Reuters,

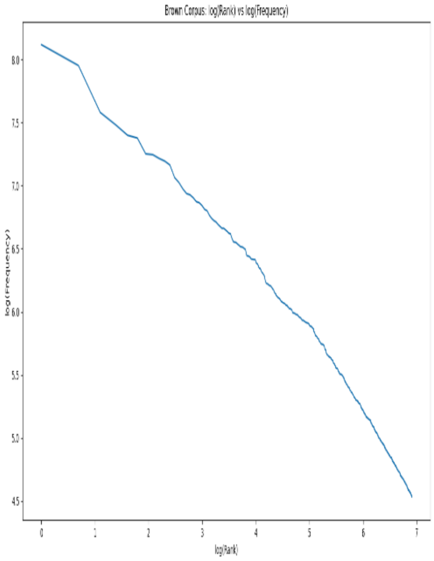
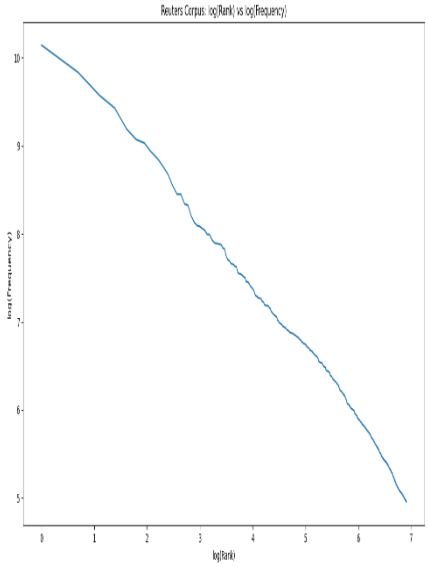
to:

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**		
Top Ten Words in the Brown Corpus:			Top Ten Words in the Reuters Corpus:		
Rank	Word	Frequency	Rank	Word	Frequency
1	one	3357	1	said	25383
2	would	2843	2	mln	18623
3	said	1961	3	vs	14341
4	could	1777	4	dlrs	12417
5	new	1635	5	pct	9810
6	time	1600	6	lt	8696
7	two	1412	7	cts	8361
8	may	1402	8	year	7529
9	first	1361	9	net	6989
10	man	1332	10	u	6392

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.

<table>
<tr><th colspan="2">log(rank) vs log(frequency) plots</th></tr>
<tr><th>Brown</th><th>Reuters</th></tr>
<tr><td>
</td><td>
</td></tr>
<tr><td colspan="2"><b>Did you observe anything interesting when comparing all plots? Write your comments below:</b></td></tr>
<tr><td colspan="2">It has been noted that the Reuters Corpus has higher frequencies than the Brown Corpus, as the maximum value of log (Frequency) is 10, compared to 8 in the Brown Corpus.</td></tr>
</table>

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.

<table>
<tr><th colspan="2">"technical" / seldom used in casual conversation word (for example "adiabatic"</th></tr>
<tr><th>Brown</th><th>Reuters</th></tr>
</table>

Count: **120** Probability: **0.000233**	Count: **95** Probability: **0.000109**
Non- technical / casual / daily-use word (for example "dinner")	
Brown	**Reuters**
Count: **0** Probability: **0.000000**	Count: **0** Probability: **0.000000**

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(good morning) = 0.25
   2) evening P(good evening) = 0.15
   3) afternoon P(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)**.