Amir Stephens

**CS 481 Spring 2023 Programming Assignment #01**

Due: **Monday, February, 11:59 PM CST**

Points: **100**

**Instructions:**

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

LastName\_FirstName\_CS481\_Programming01.zip

1. 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.

**Deliverables:**

Your submission should include:

* **Make sure your code is sufficiently commented!**
* **Part A**: Python code file(s). Your py file should be named:

cs481\_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:

cs481\_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.

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

LastName\_FirstName\_CS481\_Programming01.doc

**Part A [50 pts]:**

Use Python’s NLTK package along with the corpora:

* Brown,
* Reuters,
* corpus of your choice per the Google Docs spreadsheet (https://docs.google.com/spreadsheets/d/1HQVKyVom7p4fz--rPB4fbxFwEkJbVc0LxtGdES5RAWg/edit?usp=sharing) [Note: if, for some reason, you are not getting any interesting results with that corpus, you can change to another (as long as it is different from Brown and Reuters)]

to:

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

|  |  |  |
| --- | --- | --- |
| **Top 10 words** | | |
| **Brown** | **Reuters** | **Your choice: Gutenberg** |
| (',', 58334)  ('.', 49346)  ('``', 8837)  ("''", 8789)  (';', 5566)  ('?', 4693)  ('--', 3432)  ('one', 3292)  ('would', 2714)  (')', 2466) | ('.', 94687)  (',', 72360)  ('said', 25383)  ('mln', 18623)  ('vs', 14341)  ('-', 13705)  ('dlrs', 12417)  ("'", 11272)  ('000', 10277)  ('1', 9977) | (',', 186091)  ('.', 73746)  (':', 47406)  (';', 27329)  ("'", 19873)  ('"', 15422)  ('shall', 11682)  ('said', 9429)  ('unto', 9010)  ('-', 8850) |

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

|  |  |
| --- | --- |
| **log(rank) vs log(frequency) plots** |  |
| **Brown** |  |
| **Reuters** |  |
| **Your choice: Gutenberg** |  |
| **Did you observe anything interesting when comparing all plots? Write your comments below:** | |
| The plots all have a similar shape when viewed. This have a roughly linear shape however the words at the start seem to plaetu. A noticible point was the Brown corpus. There are two plaetu’s before large drops in word usage. These three corpus show that Ziph’s law holds with the frequency of words being inversely propertional to their rank. | |
|  | |

1. **[15 pts]** use frequency counts obtained earlier to calculate the unigram occurrence probability for the TWO (“technical” and not technical) words you chose for the Google Docs spreadsheet earlier this month. Use lowercasing first! **Display all relevant counts and probability on screen for ALL THREE corpora (also: enter final values in the table below)**. It can be zero for some words.

|  |  |  |
| --- | --- | --- |
| **“technical” word** | | |
| **Brown** | **Reuters** | **Your choice** |
| 1.033420829630242e-05 | 9.878546180169574e-06 | 1.5257782136417542e-06 |
| **Non- technical word** | | |
| **Brown** | **Reuters** | **Your choice** |
| 0.00047106766150645197 | 0.00013771855557059934 | 0.0002597637408725086 |

**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.