Vandan Vaishya

**CS585 Spring 2023 Programming Assignment #01**

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

Points: **150**

**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.

**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** |
| 1. ',' 🡪 58334 2. '.' 🡪 49346 3. '``' 🡪 8837 4. "''" 🡪 8789 5. ';' 🡪 5566 6. '?' 🡪 4693 7. '--' 🡪 3432 8. 'one' 🡪 3292 9. 'would' 🡪 2714 10. ')' 🡪 2466 | 1. '.' 🡪 94687 2. ',' 🡪 72360 3. 'said' 🡪 25383 4. 'mln' 🡪 18623 5. 'vs' 🡪 14341 6. '-' 🡪 13705 7. 'dlrs' 🡪 12417 8. "'" 🡪 11272 9. '000' 🡪 10277 10. '1' 🡪 9977 |

1. **[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:** | |
| On a log(rank) vs log(frequency) plot, if Zipf's law holds, we could typically see a roughly straight, descending line. This indicates that there is a predictable relationship between the rank of a word and its frequency, highlighting the regularity with which certain patterns of word use occur in human language. The slope of this line in such a plot is often close to -1, reflecting the inverse relationship between rank and frequency as stated in Zipf's law i.e. the frequency of any word is inversely proportional to its rank. This implies that the second most common word occurs about half as often as the most common word, the third most common word occurs about one-third as often, and so on. | |

1. **[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** |
| Economy🡪 0.00011513299318092057 | Economy 🡪 0.0004908020068348724 |
| **Non- technical / casual / daily-use word (for example “dinner”)** | |
| **Brown** | **Reuters** |
| House 🡪 0.0008613113793661273 | House 🡪 0.0003872672839759863 |

**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:
   1. **ask again**
   2. **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)**.