CODE DOCUMENTATIONS

Homework1 | asj170430 | Aadish Joshi

Highly recommend Jupyter Notebook

**Problem 2:**

**Methods:**

1)

*def preprocess (arr[strings])*

*returns String*

string data consists of the padded start <s> and end tag </s>. This will be useful in segregating the sentences with the help of start tags.

2)

*def unigrams (String)*

*returns array[Strings]*

String argument is split into character array and returned.

3)

*def bigrams (Strings)*

*returns List of array[Strings]*

Firstly, the String argument is split and then joined into list data structure with 2 words in a sub-list element.

**Terminologies:**

unigramSplit = unigram split of the data

unigramCounter = counter object of unigramSplit to count all the unique variables and its counts

unigramCountsDict = dictionary of unigram counts

uniMatrix = numpy matrix of unigram counts

bigramSplit = bigram split of the data

bigramCounter = counter object of bigramSplit to count all the unique variables and its counts

bigramCountsDict = dictionary of bigram counts

bigramMatrix = numpy matrix of bigram counts

probabilities = dictionary of bigram counts without smoothing

TestProb = total probability of test string

bigramTestSplit = bigram split of the test object

Vocabulary = V = unique words in bigramCounts

LaplaceianProbs = dictionary of probabilities with add 1 smoothing

reconProbs = dictionary of reconstituted probabilities

TestLapProb – total Laplacian smoothed probability of test String

**Problem 3:**

**Methods:**

Efficient implementation of the term-context word count

1)

*def find\_term\_context\_count(paragraph, context, term):*

*return count*

finds the sentences in paragraph. If context in the sentence, and term is in between the window of 5 words on the left and 5 words on the right, count is updated. By default the count will be zero.

2)

*def pmi\_matrix(TermContextMatrix, N):*

*return PMIMatrix*

return PMI Matrix.

3)

*def ppmi\_matrix(PMIMatrix):*

*return PPMIMatrix*

return PPMI Matrix.

4)

*def pmi\_matrix\_smoothed(TermContextMatrix\_smoothed, N):*

*return PMIMatrix\_smoothed*

return PMI add2 smoothed Matrix

5)

*def ppmi\_matrix\_smoothed(PMIMatrix\_smoothed):*

*return PPMIMatrix\_smoothed*

return PPMI add2 smoothed Matrix

**Terminologies:**

Paragraph = read data from the inputforbigrams.txt file

chairman\_said = appearance of word said in context of chairman

chairman\_of = appearance of word of in context of chairman

chairman\_board = appearance of word board in context of chairman

company\_said = appearance of word said in context of company

company \_of = appearance of word of in context of company

company \_board = appearance of word board in context of company

sales\_said = appearance of word said in context of sales

sales\_of = appearance of word of in context of sales

sales\_board = appearance of word board in context of sales

economy\_said = appearance of word said in context of economy

economy \_of = appearance of word of in context of economy

economy \_board = appearance of word board in context of economy

TCM = Total Term Context Matrix

N = Total addition of the term

P\_context = horizontal sum

P\_information = Vertical sum

TermContextMatrix = Term context matrix for [[chairman, company], [said,of,board]]

TermContextMatrix\_smoothed = add 2 smoothed matrix for [[chairman, company], [said,of,board]]