/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*

Name: Aadish Joshi

Email: asj170430@utdallas.edu

Number of problems solved: 4

\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

**Problem 1:**

*Q: Write regular expressions for the following three languages. By “word”, we mean an alphabetic string separated from other words by whitespace, any relevant punctuation, line breaks, and so forth.*

*1. Language 1: the set of all strings with two consecutive repeated words (e.g., “Humbert Humbert” and “the the” but not “the bug” or “the big bug”); (1 point)*

*2. Language 2: all strings that start at the beginning of the line with an integer and that end at the end of the line with a word; (2 points)*

*3. Language 3: all strings that have both the word “grotto” and the word “raven” in them (but not, e.g., words like grottos that merely contain the word grotto); (2 points)*

Regex 1: \b(\w+)\s+\1\b

**Explanation:** \w+ means one more more occurrence of word and numeric. Also \1 signifies repeating by 1 so that the bug and the big bug will not be selected.

Regex 2: ^[\d].\*[a-zA-Z]$

**Explanation:**

We want string to start with an integer and end with a word. \w cannot be used at the end as it will characterize words and numbers. So we specifically use **[a-zA-Z]** at the end. **^** and **$** signify start and end of the sentence. **.\*** is 0 or more occurrence of anything in between.

Regex 3: .\*(\bgrotto\b.\*\braven\b|\braven\b.\*\bgrotto\b).\*

**Explanation:**

**|** signify OR statement. Hence anything with confined boundary \b grotto \b and \braven\b or anything between raven and grotto will be selected.

**Problem 2:**

*Question:*

*1. Write a program to compute the bigrams for any given input.* ***TOTAL: 5 points***

*Apply your program to compute the bigrams you need for sentences S1 and S2.*

*2. Construct automatically (by the program) the tables with (a) the bigram counts (5 points) and the (b) bigram probabilities for the language model without smoothing. (5 points)* ***TOTAL: 10 points***

*3. Construct automatically (by the program): (i) the Laplace-smoothed count tables; (5 points) (ii) the Laplace-smoothed probability tables (5 points); and (iii) the corresponding re-constituted counts (5 points)* ***TOTAL: 15 points***

*4. Compute the total probabilities for each sentence S1 and S2, when (a) using the bigram model without smoothing; (5 points) and (b) when using the bigram model Laplace-smoothed. (5 points)* ***TOTAL: 10 points***

**Total Probability for the sentence s1:**

**Without smoothing:** 3.623421529813562e-13

**With add 1 smoothing:** 1.8887153744687491e-28

**Total Probability of the sentence s2:**

**Without smoothing:** 1.0309230980059047e-13

**With add 1 smoothing:** 3.2652026877830996e-33

**Programming Language:** Python3

**Software tool used:** Jupyter Notebook (Anaconda Navigator)

**Primary data structure used:** Dictionary

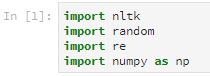
**Supporting file:**

Create following files (Already included in submitted code)

1) "inputforbigrams.txt" containing corpus of the problem 2

2) "testbigramInput.txt" containing lines for the test input.

**Programming imports:**

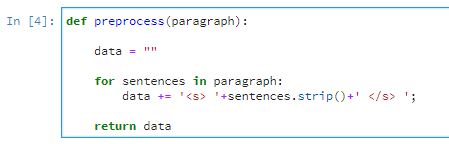


**Code description:**

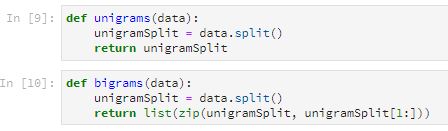
1) Firstly, the data is read through "inputforbigrams.txt" file for training corpus model.



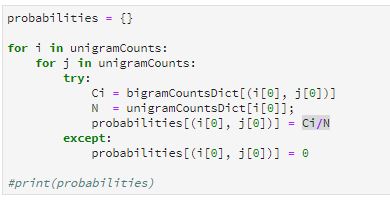
2) It is pre-processed by adding <s> and </s>syntaxes at the start and the end of every sentence in the paragraph. No removal of punctuations as they are considered as a individual token in bigrams.



3) Individual count for unigram and then bigrams is calculated



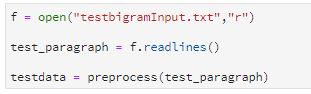
4) Individual probability is calculated as per (Bigram count / Unigram count)

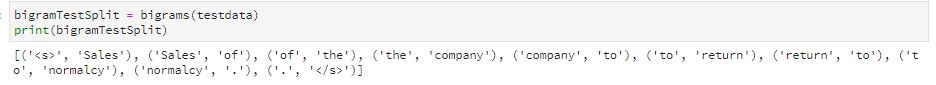


5) Bigram count is calculated by adding Laplace smoothing as well by adding 1 to individual token count.



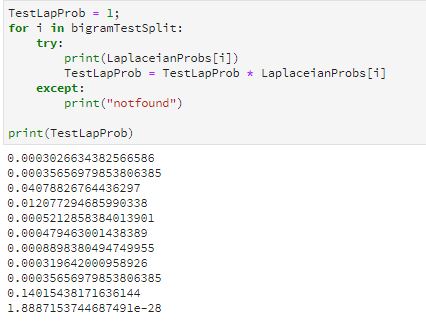
5) Testing data is again split in bigrams.





6) Referring the dictionary probability is calculated for each sentence.





**Problem 3:**

*Considering the same corpus as in Problem 2, write a program to compute the Positive Pointwise Mutual Information (PPMI) of the following words and present the results in a term-context matrix. The context of a context-word is the “window” of words consisting of (i) 5 words to the left of the context-word; (ii) the context-word; and (iii) 5 words to the right of the context-word. IF there are fewer then 5 words to the right or the left of the context-word in the same sentence, the context will be padded with “NIL”. Compute the PPMI and populate the term-context matrix for:*

*a. The word “chairman” for the context-word “said”;*

*b. The word “chairman” in the context-word “of”;*

*c. The word “company” in the context-word “board”;*

*d. The word “company” in the context-word “said”.*

***TOTAL: 10 points***

*2. Use add-2 smoothing for the same words and contexts as in (1) and present the result in a term-context matrix.* ***TOTAL: 5 points***

*3. Find which words are more similar among: [chairman, company], [company, sales] or [company, economy] when considering only the contexts provided by the context-words “said”, “of”, and “board”? Explain why.* ***TOTAL: 5 points***

*4. Using the pre-trained Glove embeddings available at: https://nlp.stanford.edu/projects/glove/*

*which words are more similar among: [chairman, company], [company, sales] or [company, economy]? Explain why.* ***TOTAL: 5 points***

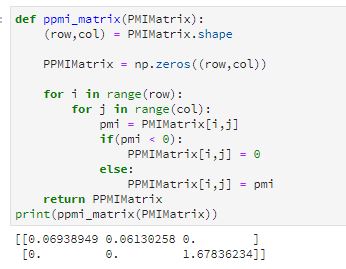
1) PPMI

[Chairman, said] = 0.06938

[Chairman, of] = 0.06130258

[company, board] = 1.678362

[company, said] = 0.0



2)

**add-2 smoothing**

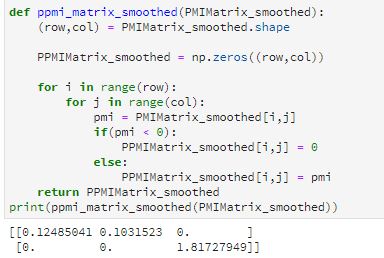
**Result:**

[Chairman, said] = 0.12485041

[Chairman, of] = 0.1031523

[company, board] = 1.81727949

[company, said] = 0.0



3)

**Results:**

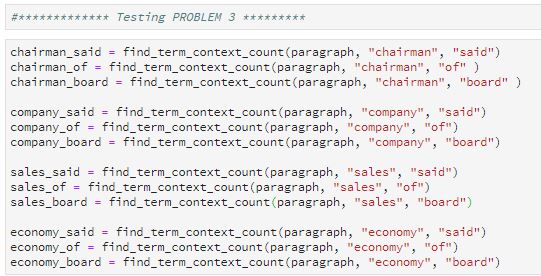
Chairman, company : 0

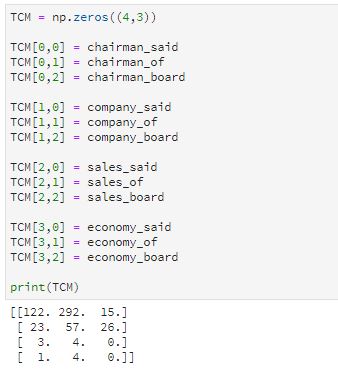
Company, sales: 0.3473

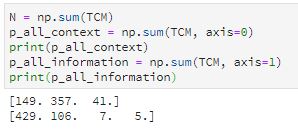
Company, economy: 0.4841

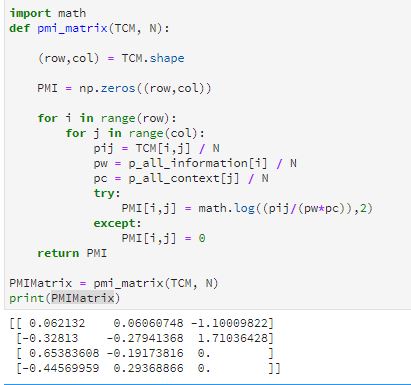
**Explanation:**

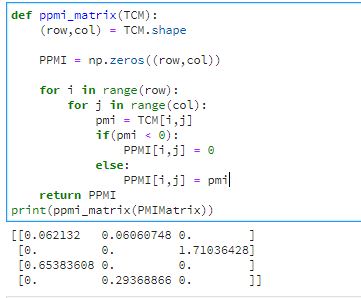
We see that company economy appears to be more similar. The reason for their similarity is two words occurred approximately equal number of times in terms of context words. Hence their word vectors appear to be close as word vectors are calculated based on count of occurrence.











4)

**Word similarity using GloVe**

**Results:**

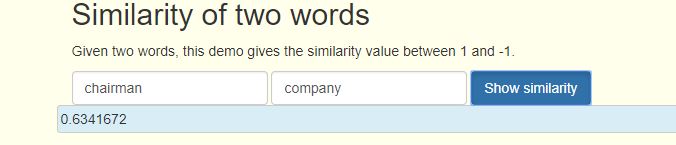
Chairman, company : 0.63416

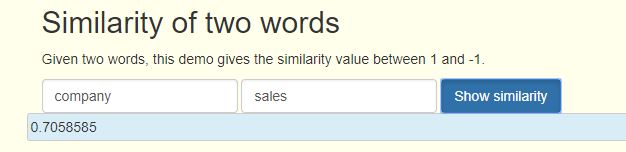
Company, sales: 0.7058585

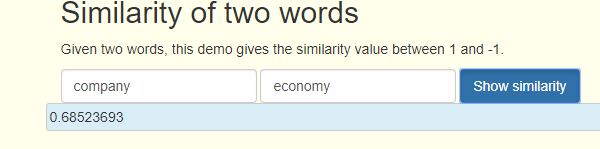
Company, economy: 0.6852

**Explanation :**

Using glove through online website, the similarity of company to sales appear to be more. This similarity is **corpus specific**. Hence in greater terms where in corpus, if sales occurred approximately equal to the company in some corpus, those words will be rated near similar.







Codes:

**Programming Language:** Python3

**Software tool used:** Jupyter Notebook (Anaconda Navigator)

**Primary data structure used:** Dictionary

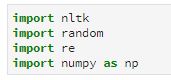
**Supporting file:**

Create following files (Already included in submitted code)

1) "inputforbigrams.txt" containing corpus of the problem 2

2) "testbigramInput.txt" containing lines for the test input.

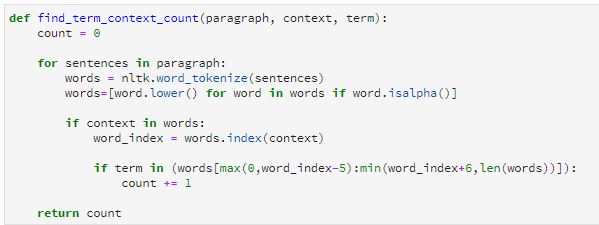
**Programming imports:**



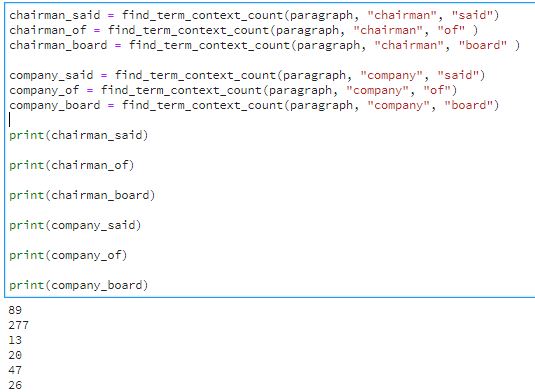
**Code description:**

1) We do not need to pre-process the sentences in the paragraph as we have to find the count of each term word in its context.

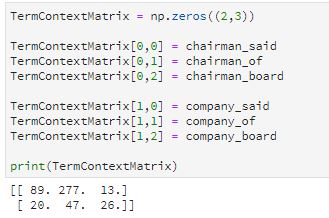
2) I have written find\_term\_context\_count which returns count of that term in the window of 5 words before and 5 words after the terms. As the tokens like punctuations do not add value to the term context matrix, I removed to punctuations and other tokens using ‘nltk.word\_tokenizer’.



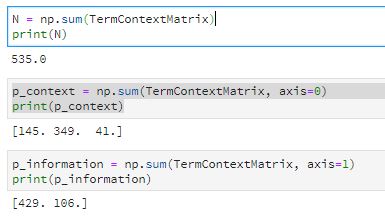
3) Then count of word chairman in context of “said”, [chairman, of] , [chairman, board], [company, said], [company, of], [company, board] are calculated with the help of function.



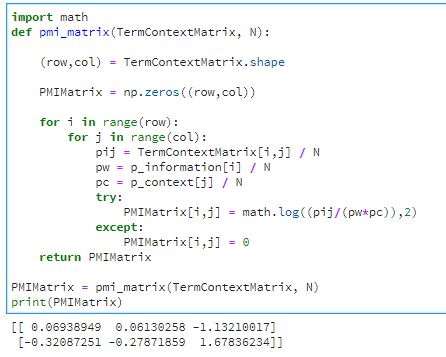
4) A term context matrix has been created and filled in with the individual values.



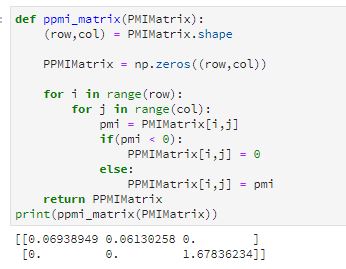
5) total N count has been calculated



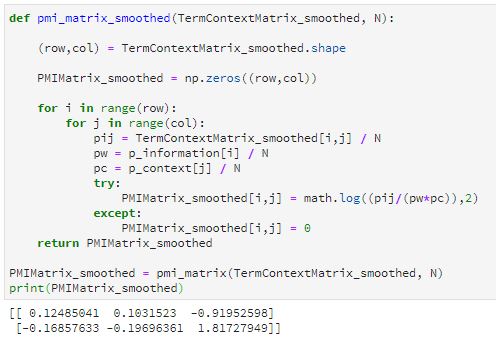
6) PMIMatrix[I,j] = Log2(Count / N)/(( (p(context)/N) \* ((P(information)/N))

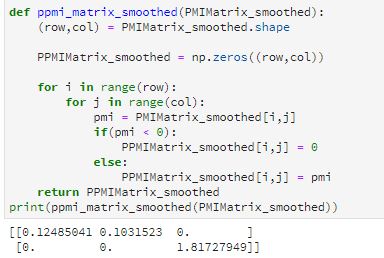


7) all values less than 0 are resulted into zero in PPMI matrix



10) add 2 smoothing has been performed by adding 2 in every term and PMI matrix is calculated.





Problem 4:

*1. Create the Hidden Markov Model (HMM) and show (a) the transition probabilities and (b) observation likelihoods in each state that will be reached by sentences S1 and S2 after 3 time-steps. Present only the transition and observation likelihoods in the states reached after three steps.* ***TOTAL: 5 points***

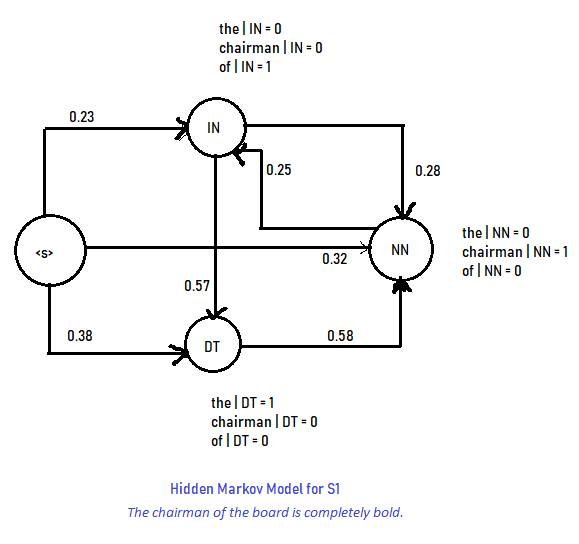
*2. Create the Viterbi table for each sentence and populate it entirely.* ***TOTAL: 10 points***

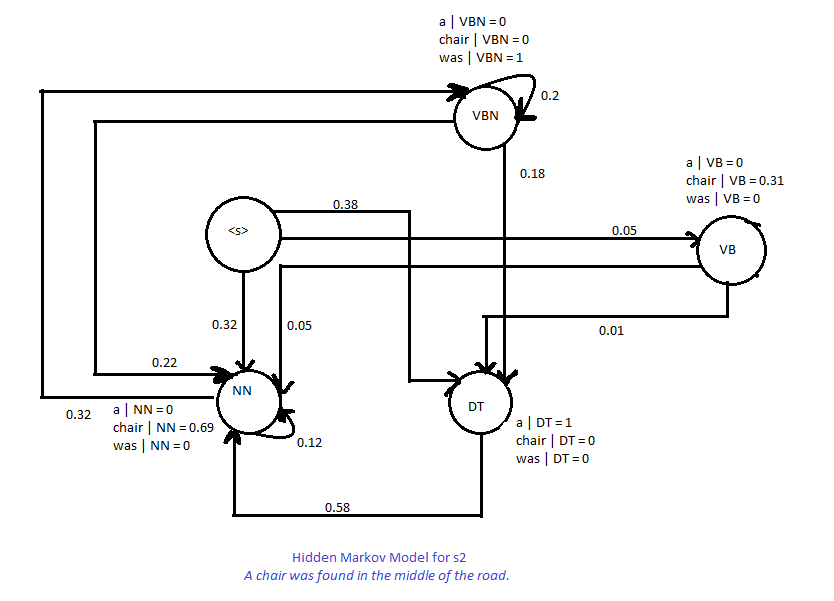
*3. What is the probability of assigning the tag sequence for each of the sentences.?* ***TOTAL: 10 points***

*4. Execute Stanford POS-tagger (available from: https://nlp.stanford.edu/static/software/tagger.shtml)*

*on both sentences. Which POS tags were assigned by the Stanford POS-tagger more accurately? Explain.* ***TOTAL: 5 points***

1)

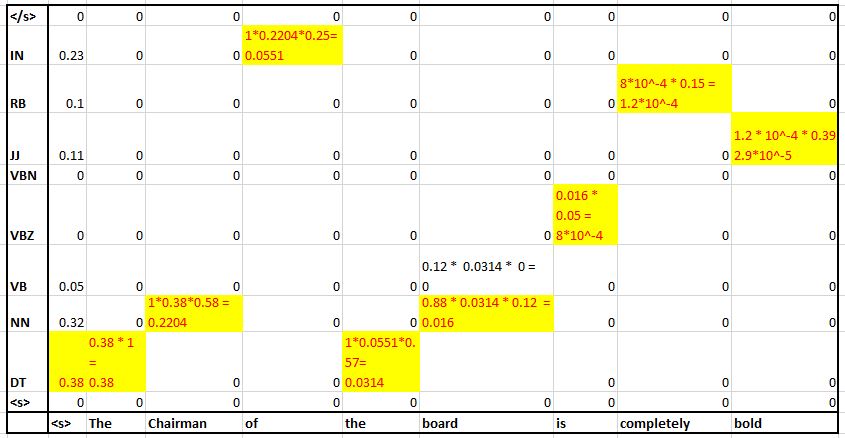




2)

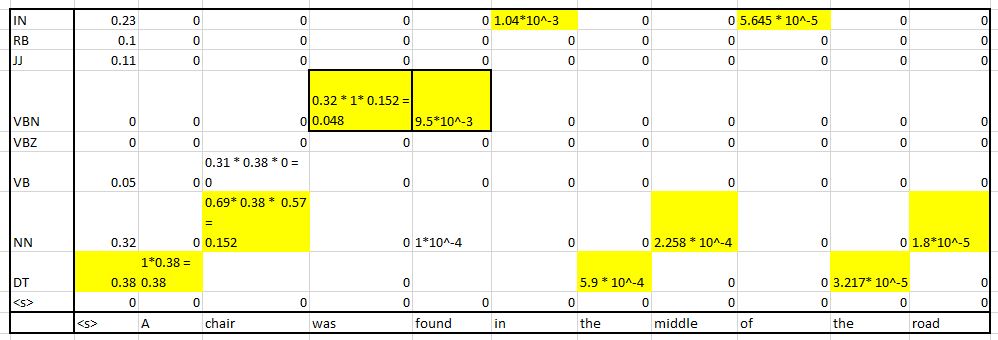
Viterbi table for first sentence: The chairman of the board is completely bold.

**The/DT chairman/NN of/IN the/DT board/NN is/VBZ completely/RB bold/JJ**



Viterbi table for: A chair was found in the middle of the road.

**A/DT chair/NN was/VBN found/VBN in/IN the/DT middle/NN of/IN the/DT road/NN**



3)

|  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- |
| The | Chairman | Of | The | Board | Is | Completely | bold |
| DT | NN | IN | DT | NN | VBZ | RB | JJ |

**The probability of the sentence: 2.9 \* 10^-5**

|  |  |  |  |  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- | --- | --- | --- | --- |
| A | Chair | Was | Found | In | The | Middle | Of | The | road |
| DT | NN | VBN | VBN | IN | DT | NN | IN | DT | NN |

**The probability of the sentence: 1.8 \* 10^-5**

4)

**Explanation:**

**Stanford POS appears to be more accurate**. Because of the sentence S2 ‘was’ is tagged as VBD. However with Viterbi POS, tags it as VBN. ( I used online parse <nlp.stanford.edu:8080/parser/index.jsp>



