EEE586: Statistical Foundations of Natural Language Processing Assignment 3

Atakan Topcu, *Bilkent University*

***Abstract*—Collocations are a crucial aspect of Natural Language Processing (NLP) that involves identifying the word combinations in a language. Learning and identifying collocations not only make machines communicate better with humans but also enables them to do abstractions as collocations provide insights into the structure of a language. Utilization of this property can be used in applications such as text classification, information retrieval, and machine translation. This homework assignment focuses on identifying collocations in the form of bigrams from a corpus of six novels by Fyodor Dostoevsky and analyzing their statistical significance using three hypothesis testing methods: Student's t-test, Pearson's chi-square test, and the likelihood-ratio test.**

***Index Terms*—Natural Language Processing, Collocations, Statistical Methods**

I. Introduction

In this study, I used the corpus extracted from Fyodor Dostoevsky. The main part of the assignment will be explained in the Methods & Results section. To begin with, I will divide the section into three subsections where I will explain each part part by part. Then, I will discuss the techniques used to classify collocations briefly. Finally, I will summarize the implementation details.

The Discussions & Conclusions section will provide an interpretation of the experimental findings. A comparison will be made between the expected and actual experimental results. The methodology of the study will also be reviewed. Lastly, a brief and concise summary of the project's main idea will be presented.

Overall, I have four main tasks to investigate:

- Preprocess the given corpus.
- Generate bigrams with variable window size.
- Utilize statistical methods for classification.

II. Methods & Results

Here, the main tasks for the assignment are further discussed. The dataset used in this study is given as a text file.

Part 1: Corpus Preprocessing

Firstly, I have downloaded the "Fyodor Dostoevsky Processed.txt" from Moodle. After checking the version of the nltk library (the nltk version is 3.8.1), I have tokenized the text. It took me approximately a minute to both tokenize and get the POS (part-of-speech) tags of the tokens. Furthermore, I have utilized the lemmatizer file given in Moodle to lemmatize the verbs and adverbs.

After the initial preprocessing step, I created two bigrams with a window size of 1 and 3 respectively. Also, I have created a unigram list so that it will be easier to implement functions that classify the collocations. For these three sets, I have eliminated all the bigrams or unigrams that don't contain POS tags NOUN-NOUN or ADJ-NOUN (NOUN or ADJ in the case of unigram). Then, I proceed to eliminate all the bigrams or unigrams that contain the stop words as well as any punctuation marks. Finally, I have eliminated bigrams/unigrams that occurred less than 10.

Part 2: Finding the Collocations

In this part, I have written functions to calculate student t-test, chi-square test, and likelihood ratio test. The concepts of these functions will be further discussed in Part 3. After computing each binomial probability for the likelihood ratio test, I substitute values of zero with the smallest positive number in Python, which is approximately $5 * 10^{-325}$. This is done to ensure binomial probabilities are computable. Then, the logarithm of each binomial

	word1	word2	counts
17254	pyotr	stepanovitch	427
20396	stepan	trofimovitch	412
23195	varvara	petrovna	331
15141	old	man	289
7435	fyodor	pavlovitch	246

Fig. 1. Example dataframe for bigram with window size 1.

probability was taken, and I performed subsequent computations with them.

To ease the operations I have converted bigrams and unigram to panda dataframe. An example dataframe for bigram of window size 1 is given in Fig. 1. Using a for loop, I have iteratively selected pairs of words.

After passing the bigrams to three functions, the resulting dataframe can be seen in Fig. 2.

Part 3: Explaining the Statistical Tests

The formula for the student t-test is:

$$t = \frac{\overline{x}_1 - \overline{x}_2}{s_p \sqrt{\frac{1}{N}}}$$

where $\overline{x}_1$ and $\overline{x}_2$ are the means of the two samples, s_p is the standard deviation, and N is the sample size, and t is the t-value. In our case, $\overline{x}_1$ is the mean of the bigram, and $\overline{x}_2$ is the mean of the null hypothesis (i.e., word1 and word2 are independent). Since there are too many dimensions of freedom, I assumed the dimension of freedom was infinite.

The formula for Pearson's chi-square test is:

$$\chi^2 = \sum_{i=1}^{n} \sum_{j=1}^{m} \frac{(O_{ij} - E_{ij})^2}{E_{ij}}$$

	word1	word2	counts	t-score	expected	chi-square	likelihood_ratio_score
17254	pyotr	stepanovitch	427	20.596348	1.397501	129615.257394	2049.102849
20396	stepan	trofimovitch	412	20.255550	0.857240	197188.981543	1998.069682
23195	varvara	petrovna	331	18.152786	0.739011	147592.189178	1785.934891
15141	old	man	289	16.193509	13.710351	5527.531163	1303.475446
7435	fyodor	pavlovitch	246	15.654434	0.469802	128320.075280	1650.990245

Fig. 2. Example dataframe after all the functions for bigram with window size 1.

where n is the number of rows, m is the number of columns, O_{ij} is the observed frequency in cell (i, j), and E_{ij} is the expected frequency in cell (i, j).

The expected frequency E_{ij} is calculated as:

$$E_{ij} = \frac{(R_i \times C_j)}{N}$$

where R_i is the sum of the ith row, C_j is the sum of the jth column, and N is the total sample size.

The degrees of freedom for the chi-square distribution are:

$$df = (n-1) \times (m-1)$$

where in our case, df is 1.

The formula for the likelihood ratio test is:

$$G^2 = -2 \cdot \log \frac{L(H_1)}{L(H_0)}$$

where G^2 is the likelihood ratio score, $L(H_1)/L(H_0)$ is the likelihood ratio, and $\log$ is the natural logarithm.

The likelihood ratio is calculated by comparing the likelihood of the null hypothesis H_0 and the alternative hypothesis H_1. The null hypothesis assumes that the two words in the bigram occur independently of each other, while the alternative hypothesis assumes that the two words occur together more frequently than expected by chance. The likelihood ratio is the ratio of the likelihood of the alternative hypothesis to the likelihood of the null hypothesis. The given equation follows a chi-square distribution with df=1.

III. Discussions & Conclusions

All in all, the main idea of this investigation was to explore and understand collocations in NLP settings and how to classify them using classical methods. In the end, I learned how to utilize collocations along with using classical approaches in NLP. Furthermore, by coming up with a new way to represent a document, I have learned how to utilize POS.

In [192…

```
import nltk
nltk.download('wordnet')
nltk.download('punkt')
nltk.download('averaged_perceptron_tagger')
nltk.download('universal_tagset')
import time
import numpy as np
import custom_lemmatizer
import pandas as pd
from scipy.stats import binom
import math
from scipy.stats import chi2
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\ataka\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\ataka\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\ataka\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
[nltk_data] Downloading package universal_tagset to
[nltk_data]     C:\Users\ataka\AppData\Roaming\nltk_data...
[nltk_data]   Package universal_tagset is already up-to-date!
```

In [2]:

```
print('The nltk version is {}.'.format(nltk.__version__)) # check version
```

```
The nltk version is 3.8.1.
```

Part 1

In [2]:

```
with open('Fyodor Dostoyevski Processed.txt', 'r') as file:
    CorpusData = file.read()
words = nltk.word_tokenize(CorpusData)
tagged_Corpus = nltk.pos_tag(words, tagset="universal")
```

In [264…

```
#what is the total number of words in the corpus?
total_words = len(words)
print('The total number of words in the corpus is {}.'.format(total_words))
```

```
The total number of words in the corpus is 1425758.
```

In [256…

```
lemmatizer = custom_lemmatizer.custom_lemmatizer()
lemmaized_tokens = []

start_time = time.time()
for t in tagged_Corpus:

    lemmaized_tokens.append((lemmatizer.lemmatize(t), t[1]))

print("%s seconds lemmatization time!" % (time.time() - start_time))
```

```
1.989999771118164 seconds lemmatization time!
```

In [269…

```
that_count = 0
for t in lemmaized_tokens:
    if t[0] == 'that':
```

```
        that_count += 1
print('The count of "that" after lemmatization is {}.'.format(that_count))

that_count = 0
for t in lemmaized_tokens:
    if t[0] == 'the':
        that_count += 1
print('The count of "the" after lemmatization is {}.'.format(that_count))

that_count = 0
for t in lemmaized_tokens:
    if t[0] == 'abject':
        that_count += 1
print('The count of "abject" after lemmatization is {}.'.format(that_count))

that_count = 0
for t in lemmaized_tokens:
    if t[0] == 'london':
        that_count += 1
print('The count of "london" after lemmatization is {}.'.format(that_count))
that_count = 0
for t in lemmaized_tokens:
    if t[0] == '.':
        that_count += 1
print('The count of "." after lemmatization is {}.'.format(that_count))
```

```
The count of "that" after lemmatization is 19429.
The count of "the" after lemmatization is 48392.
The count of "abject" after lemmatization is 21.
The count of "london" after lemmatization is 2.
The count of "." after lemmatization is 51738.
```

In [272…

```
import time

start_time = time.time()

bigrams_size1 = [(lemmaized_tokens[i], lemmaized_tokens[i + 1]) for i in range(len(lemmai

bigrams_size2 = [(lemmaized_tokens[i], lemmaized_tokens[i + j]) for i in range(len(lemmai
bigrams_size2 += [(lemmaized_tokens[-3], lemmaized_tokens[-2]), (lemmaized_tokens[-3], le
             (lemmaized_tokens[-2], lemmaized_tokens[-1])]

print(f"{time.time() - start_time} seconds bigram creation time!")
print(f"Size of bigrams_size1: {len(bigrams_size1)}")
print(f"Size of bigrams_size2: {len(bigrams_size2)}")
```

```
1.3456499576568604 seconds bigram creation time!
Size of bigrams_size1: 1425757
Size of bigrams_size2: 4277268
```

In [275…

```
bigram_count = 0
for i in range(len(bigrams_size1)):
    if bigrams_size1[i][0][0] == 'magnificent' and bigrams_size1[i][1][0] == 'capital':
        bigram_count += 1
print('The count of ("magnificent","capital") in windows of size 1 is {}.'.format(bigram_c


bigram_count = 0
for i in range(len(bigrams_size2)):
    if bigrams_size2[i][0][0] == 'bright' and bigrams_size2[i][1][0] == 'fire':
        bigram_count += 1
print('The count of ("bright","fire") in windows of size 1 is {}.'.format(bigram_count))
```

```
The count of ("magnificent","capital") in windows of size 1 is 1.
```

```
The count of ("bright","fire") in windows of size 1 is 1.
```

In [276…

```python
#Eliminate all bigrams except those with POS tags NOUN-NOUN or ADJ-NOUN.
unigrams=[b for b in lemmaized_tokens if b[1] == 'NOUN' or b[1] == 'ADJ']
bigrams_size1 = [b for b in bigrams_size1 if (b[0][1] == 'NOUN' and b[1][1] == 'NOUN') or
bigrams_size2 = [b for b in bigrams_size2 if (b[0][1] == 'NOUN' and b[1][1] == 'NOUN') or

#Eliminate bigrams that include stopwords
stopwords = ["i", "me", "my", "myself", "we", "our", "ours", "ourselves", "you", "your", '
#Eliminate bigrams that include stopwords
unigrams = [b for b in unigrams if b[0] not in stopwords]
bigrams_size1 = [b for b in bigrams_size1 if b[0][0] not in stopwords and b[1][0] not in s
bigrams_size2 = [b for b in bigrams_size2 if b[0][0] not in stopwords and b[1][0] not in s
#Eliminate bigrams including any punctuation marks. (Hint: You can use the isalpha() funct

unigrams = [b for b in unigrams if b[0].isalpha()]
bigrams_size1 = [b for b in bigrams_size1 if b[0][0].isalpha() and b[1][0].isalpha()]
bigrams_size2 = [b for b in bigrams_size2 if b[0][0].isalpha() and b[1][0].isalpha()]

#remove POS tags from bigrams
bigrams_size1 = [(b[0][0], b[1][0]) for b in bigrams_size1]
bigrams_size2 = [(b[0][0], b[1][0]) for b in bigrams_size2]


#create panda dataframe for unigrams with single column
unigrams = pd.DataFrame(unigrams, columns=['words', 'POS'])
#convert bigrams to a pandas dataframe
bigrams_size1 = pd.DataFrame(bigrams_size1, columns=['word1', 'word2'])
bigrams_size2 = pd.DataFrame(bigrams_size2, columns=['word1', 'word2'])

#count the words in uni-grams. Reorder from most frequent to least frequent.
unigrams = unigrams.groupby(['words']).size().reset_index(name='counts')
unigrams = unigrams.sort_values(by=['counts'], ascending=False)
unigrams = unigrams[unigrams['counts'] >= 10]
#Eliminate bigrams that occur less than 10 times.
bigrams_size1 = bigrams_size1.groupby(['word1', 'word2']).size().reset_index(name='counts'
bigrams_size1 = bigrams_size1.sort_values(by=['counts'], ascending=False)
bigrams_size1 = bigrams_size1[bigrams_size1['counts'] >= 10]
bigrams_size2 = bigrams_size2.groupby(['word1', 'word2']).size().reset_index(name='counts'
bigrams_size2 = bigrams_size2.sort_values(by=['counts'], ascending=False)
bigrams_size2 = bigrams_size2[bigrams_size2['counts'] >= 10]
```

In [281…

```python
bigram_count = 0
for i in range(len(bigrams_size1)):
    if bigrams_size1.iloc[i][0] == 'mr.' and bigrams_size1.iloc[i][1] == 'skimpole':
        bigram_count += 1
print('The count of ("mr.","skimpole") in windows of size 1 is {}.'.format(bigram_count))


bigram_count = 0
for i in range(len(bigrams_size2)):
    if bigrams_size2.iloc[i][0] == 'spontaneous' and bigrams_size2.iloc[i][1] == 'combusti
        bigram_count += 1
print('The count of ("spontaneous.","combustion") in windows of size 3 is {}.'.format(bigr
```

```
The count of ("mr.","skimpole") in windows of size 1 is 0.
The count of ("spontaneous.","combustion") in windows of size 3 is 0.
```

Part 2

In [300…

```python
import pandas as pd
import numpy as np
```

```python
def student_t_test(bigram_counts, unigrams, alpha=0.005):
    unigram_total = unigrams['counts'].sum()
    for row in bigram_counts.iterrows():

        H0 = unigrams[unigrams['words'] == row[1]['word1']]['counts'].values[0] * unigrams
        MLE = row[1]['counts'] / unigram_total

        t = (MLE - H0) / np.sqrt(MLE / unigram_total)
        #save the t value in the bigram_counts dataframe
        bigram_counts.loc[row[0], 't-score'] = t

    return bigram_counts

bigrams_size1_t_test = student_t_test(bigrams_size1, unigrams)
bigrams_size2_t_test = student_t_test(bigrams_size2, unigrams)
```

In [301…

```python
bigrams_size1_t_test = bigrams_size1_t_test.sort_values(by=['t-score'], ascending=False)
print('The top 20 bigrams in windows of size 1 are:')
print(bigrams_size1_t_test.head(20))
```

```
The top 20 bigrams in windows of size 1 are:
           word1          word2  counts    t-score
17254      pyotr     stepanovitch     427  20.596348
20396     stepan     trofimovitch     412  20.255550
23195    varvara         petrovna     331  18.152786
15141        old              man     289  16.193509
7435      fyodor       pavlovitch     246  15.654434
10934   katerina         ivanovna     229  15.092046
24386      young              man     232  14.716428
14260   nastasia       philipovna     215  14.638661
14724    nikolay  vsyevolodovitch     198  14.041475
15247        old            woman     208  14.031270
8321       great             deal     164  12.724990
17253      pyotr       petrovitch     143  11.882136
12244  lizabetha       prokofievna     136  11.652786
12567       long             time     143  11.458258
4401      dmitri      fyodorovitch     126  11.185020
14637       next              day     117  10.608096
15127        old             lady     119  10.573032
6678       first             time     130  10.421459
5310      evgenie       pavlovitch     106  10.266672
24450      yulia       mihailovna     105  10.238928
```

In [302…

```python
#find the occurance of word1 and word2 in the unigrams dataframe
bigrams_size1_t_test['word1_count'] = bigrams_size1_t_test.apply(lambda row: unigrams[unig
bigrams_size1_t_test['word2_count'] = bigrams_size1_t_test.apply(lambda row: unigrams[unig
print('The top 20 bigrams in windows of size 1 are:')
print(bigrams_size1_t_test.head(20))
```

```
The top 20 bigrams in windows of size 1 are:
           word1          word2  counts    t-score  word1_count  word2_count
17254      pyotr     stepanovitch     427  20.596348          701          502
20396     stepan     trofimovitch     412  20.255550          430          502
23195    varvara         petrovna     331  18.152786          379          491
15141        old              man     289  16.193509         1356         2546
7435      fyodor       pavlovitch     246  15.654434          260          455
10934   katerina         ivanovna     229  15.092046          253          613
24386      young              man     232  14.716428          776         2546
14260   nastasia       philipovna     215  14.638661          362          247
14724    nikolay  vsyevolodovitch     198  14.041475          354          298
15247        old            woman     208  14.031270         1356         1047
8321       great             deal     164  12.724990         1202          218
17253      pyotr       petrovitch     143  11.882136          701          327
```

```
12244  lizabetha      prokofievna     136  11.652786         153         175
12567       long             time     143  11.458258         574        2623
4401      dmitri     fyodorovitch     126  11.185020         354         319
14637       next              day     117  10.608096         332        1711
15127        old             lady     119  10.573032        1356         680
6678       first             time     130  10.421459        1073        2623
5310     evgenie       pavlovitch     106  10.266672         165         455
24450      yulia       mihailovna     105  10.238928         115         180
```

In [303…

```python
#do the same for bigrams in windows of size 3
bigrams_size2_t_test = bigrams_size2_t_test.sort_values(by=['t-score'], ascending=False)
bigrams_size2_t_test['word1_count'] = bigrams_size2_t_test.apply(lambda row: unigrams[uniq
bigrams_size2_t_test['word2_count'] = bigrams_size2_t_test.apply(lambda row: unigrams[uniq
print('The top 20 bigrams in windows of size 3 are:')
print(bigrams_size2_t_test.head(20))
```

```
The top 20 bigrams in windows of size 3 are:
           word1            word2  counts    t-score  word1_count  word2_count
58985      pyotr     stepanovitch     427  20.596348          701          502
70667     stepan     trofimovitch     412  20.255550          430          502
79235    varvara         petrovna     331  18.152786          379          491
51320        old              man     290  16.224287         1356         2546
26366     fyodor       pavlovitch     246  15.654434          260          455
37238   katerina         ivanovna     229  15.092046          253          613
83930      young              man     234  14.784147          776         2546
48499   nastasia       philipovna     215  14.638661          362          247
51481        old            woman     215  14.278359         1356         1047
49751    nikolay  vsyevolodovitch     198  14.041475          354          298
28920      great             deal     165  12.764221         1202          218
50655          o            clock     157  12.518405          194          188
29888         ha               ha     144  11.980779          241          241
58974      pyotr       petrovitch     143  11.882136          701          327
41331  lizabetha      prokofievna     136  11.652786          153          175
42014       long             time     147  11.631202          574         2623
16696     dmitri     fyodorovitch     126  11.185020          354          319
49366       next              day     119  10.701915          332         1711
51301        old             lady     119  10.573032         1356          680
75032   thousand           rouble     112  10.496407          425          543
```

In [304…

```python
def pearson_chi_test(bigram_counts, unigrams, alpha=0.005):
    unigram_total = unigrams['counts'].sum()
    bigram_counts['expected'] = bigram_counts.apply(lambda row:
                                        unigrams[unigrams['words'] == row['word1']]['count
                                        unigrams[unigrams['words'] == row['word2']]['count
                                        unigram_total, axis=1)

    bigram_counts['chi-square'] = np.power((bigram_counts['counts'] - bigram_counts['expec


    return bigram_counts
```

In [305…

```python
bigrams_size1_chi = pearson_chi_test(bigrams_size1_t_test, unigrams)
bigrams_size2_chi = pearson_chi_test(bigrams_size2_t_test, unigrams)
```

In [308…

```python
#reorder the bigrams by chi-square value
bigrams_size1_chi = bigrams_size1_chi.sort_values(by=['chi-square'], ascending=False)
bigrams_size2_chi = bigrams_size2_chi.sort_values(by=['chi-square'], ascending=False)
```

In [309…

```python
print('The top 20 bigrams in windows of size 1 are:')
print(bigrams_size1_chi.head(20))
```

```
The top 20 bigrams in windows of size 1 are:
            word1          word2  counts     t-score  word1_count  word2_count  \
13538      mihail    makarovitch      20    4.471726           22           21
18163      rodion    romanovitch      80    8.940855           95           81
11554         lef  nicolaievitch      33    5.743619           35           39
22084      trifon   borissovitch      35    5.915033           40           39
20396      stepan   trofimovitch     412   20.255550          430          502
819       avdotya      romanovna      86    9.269403           92          107
14711     nikodim        fomitch      19    4.358449           19           26
7480      gavrila ardalionovitch      49    6.998099           50           67
10662     ippolit   kirillovitch      31    5.566789           38           36
12244   lizabetha    prokofievna     136   11.652786          153          175
18803      semyon   yakovlevitch      33    5.743437           44           37
13277     mavriky   nikolaevitch      96    9.792194          112          127
23195     varvara        petrovna     331   18.152786          379          491
24450       yulia      mihailovna     105   10.238928          115          180
484        andrey    antonovitch      67    8.181137          110           79
14260    nastasia     philipovna     215   14.638661          362          247
17254       pyotr   stepanovitch     427   20.596348          701          502
7435       fyodor      pavlovitch     246   15.654434          260          455
16915    printing          press      10    3.162015           11           19
3532        daria     alexeyevna      14    3.741212           20           21

       expected     chi-square
13538  0.001835  217975.586250
18163  0.030559  209270.985725
11554  0.005421  200826.981245
22084  0.006195  197663.852349
20396  0.857240  197188.981543
819    0.039093  189016.577086
14711  0.001962  183975.540423
7480   0.013304  180376.941065
10662  0.005433  176829.444029
12244  0.106331  173675.473950
18803  0.006465  168373.141600
13277  0.056487  162959.246589
23195  0.739011  147592.189178
24450  0.082205  133905.212640
484    0.034510  129942.687215
14260  0.355088  129749.351666
17254  1.397501  129615.257394
7435   0.469802  128320.075280
16915  0.000830  120462.297481
3532   0.001668  117482.401668
```

In [310…

```python
print('The top 20 bigrams in windows of size 3 are:')
print(bigrams_size2_chi.head(20))
```

```
The top 20 bigrams in windows of size 3 are:
            word1          word2  counts     t-score  word1_count  word2_count  \
45464      mihail    makarovitch      20    4.471726           22           21
62183      rodion    romanovitch      80    8.940855           95           81
39460         lef  nicolaievitch      33    5.743619           35           39
76729      trifon   borissovitch      35    5.915033           40           39
70667      stepan   trofimovitch     412   20.255550          430          502
3791      avdotya      romanovna      86    9.269403           92          107
49655     nikodim        fomitch      19    4.358449           19           26
26705     gavrila ardalionovitch      49    6.998099           50           67
36251     ippolit   kirillovitch      31    5.566789           38           36
41331   lizabetha    prokofievna     136   11.652786          153          175
82319        wisp            tow      14    3.741352           18           16
50655           o          clock     157   12.518405          194          188
64615      semyon   yakovlevitch      33    5.743437           44           37
44546     mavriky   nikolaevitch      96    9.792194          112          127
```

```
79235    varvara        petrovna     331  18.152786         379           491
84079      yulia      mihailovna     105  10.238928         115           180
2102      andrey     antonovitch      67   8.181137         110            79
48499   nastasia      philipovna     215  14.638661         362           247
58985      pyotr    stepanovitch     427  20.596348         701           502
26366     fyodor       pavlovitch    246  15.654434         260           455

       expected      chi-square
45464  0.001835  217975.586250
62183  0.030559  209270.985725
39460  0.005421  200826.981245
76729  0.006195  197663.852349
70667  0.857240  197188.981543
3791   0.039093  189016.577086
49655  0.001962  183975.540423
26705  0.013304  180376.941065
36251  0.005433  176829.444029
41331  0.106331  173675.473950
82319  0.001144  171341.334477
50655  0.144841  169866.430868
64615  0.006465  168373.141600
44546  0.056487  162959.246589
79235  0.739011  147592.189178
84079  0.082205  133905.212640
2102   0.034510  129942.687215
48499  0.355088  129749.351666
58985  1.397501  129615.257394
26366  0.469802  128320.075280
```

In [314…

```python
def likelihood_ratio_test(bigram_counts, unigrams, alpha=0.05):
    # Compute total counts
    unigram_total = unigrams['counts'].sum()
    bigram_total = bigram_counts['counts'].sum()

    for idx, row in bigram_counts.iterrows():
        # Extract counts and calculate probabilities for hypothesis tests
        c_12, c_1, c_2 = row['counts'], unigrams.loc[unigrams['words'] == row['word1'], 'c
        N = unigram_total
        H1_p = c_2 / N
        H2_p1 = c_12 / c_1
        H2_p2 = (c_2 - c_12) / (N - c_1)

        # Calculate the terms in the likelihood ratio test
        H1_term_1 = binom.pmf(c_12, c_1, H1_p)
        H1_term_2 = binom.pmf(c_2 - c_12, N - c_1, H1_p)
        H2_term_1 = binom.pmf(c_12, c_1, H2_p1)
        H2_term_2 = binom.pmf(c_2 - c_12, N - c_1, H2_p2)

        # Check for values of zero and use a small value instead
        H1_term_1 = H1_term_1 if H1_term_1 != 0 else math.ulp(0.0)
        H1_term_2 = H1_term_2 if H1_term_2 != 0 else math.ulp(0.0)
        H2_term_1 = H2_term_1 if H2_term_1 != 0 else math.ulp(0.0)
        H2_term_2 = H2_term_2 if H2_term_2 != 0 else math.ulp(0.0)

        # Calculate the likelihood ratio and the corresponding score
        log_L_H1 = math.log(H1_term_1) + math.log(H1_term_2)
        log_L_H2 = math.log(H2_term_1) + math.log(H2_term_2)
        log_likelihood_ratio = log_L_H1 - log_L_H2
        likelihood_ratio_score = -2 * log_likelihood_ratio

        # Store the likelihood ratio score in the bigram_counts dataframe
        bigram_counts.loc[idx, 'likelihood_ratio_score'] = likelihood_ratio_score

    return bigram_counts
```

In [315…

```
bigrams_size1_likelihood=likelihood_ratio_test(bigrams_size1_chi, unigrams)
bigrams_size2_likelihood=likelihood_ratio_test(bigrams_size2_chi, unigrams)
```

In [316…

```
#reorder according to likelihood ratio score
bigrams_size1_likelihood=bigrams_size1_likelihood.sort_values(by='likelihood_ratio_score',
bigrams_size2_likelihood=bigrams_size2_likelihood.sort_values(by='likelihood_ratio_score',
```

In [317…

```
print('The top 20 bigrams in windows of size 1 are:')
print(bigrams_size1_likelihood.head(20))
```

```
The top 20 bigrams in windows of size 1 are:
           word1            word2  counts    t-score  word1_count  \
17254      pyotr     stepanovitch     427  20.596348          701
20396     stepan      trofimovitch     412  20.255550          430
23195    varvara         petrovna     331  18.152786          379
14260   nastasia        philipovna     215  14.638661          362
14724    nikolay  vsyevolodovitch     198  14.041475          354
7435      fyodor        pavlovitch     246  15.654434          260
12244  lizabetha       prokofievna     136  11.652786          153
10934   katerina          ivanovna     229  15.092046          253
8321       great             deal     164  12.724990         1202
24450      yulia       mihailovna     105  10.238928          115
13277    mavriky      nikolaevitch      96   9.792194          112
819      avdotya        romanovna      86   9.269403           92
18163     rodion      romanovitch      80   8.940855           95
15141        old              man     289  16.193509         1356
4401      dmitri      fyodorovitch     126  11.185020          354
17253      pyotr        petrovitch     143  11.882136          701
24386      young              man     232  14.716428          776
15247        old            woman     208  14.031270         1356
5310     evgenie        pavlovitch     106  10.266672          165
17197  pulcheria     alexandrovna      84   9.154759          117

       word2_count   expected     chi-square  likelihood_ratio_score
17254          502   1.397501  129615.257394             2049.102849
20396          502   0.857240  197188.981543             1998.069682
23195          491   0.739011  147592.189178             1785.934891
14260          247   0.355088  129749.351666             1781.342890
14724          298   0.418938   93183.845170             1659.786223
7435           455   0.469802  128320.075280             1650.990245
12244          175   0.106331  173675.473950             1639.129566
10934          613   0.615902   84687.683108             1582.494087
8321           218   1.040618   25519.214304             1532.790675
24450          180   0.082205  133905.212640             1531.818564
13277          127   0.056487  162959.246589             1470.378026
819            107   0.039093  189016.577086             1394.560562
18163           81   0.030559  209270.985725             1355.030643
15141         2546  13.710351    5527.531163             1303.475446
4401           319   0.448461   35149.541283             1278.534223
17253          327   0.910325   22178.327775             1266.757179
24386         2546   7.846041    6403.865899             1216.645321
15247         1047   5.638153    7263.073218             1170.477602
5310           455   0.298144   37474.807474             1150.846670
17197          205   0.095251   73909.945908             1096.267876
```

In [319…

```
print('The top 20 bigrams in windows of size 3 are:')
print(bigrams_size2_likelihood.head(20))
```

```
The top 20 bigrams in windows of size 3 are:
           word1            word2  counts    t-score  word1_count  \
58985      pyotr     stepanovitch     427  20.596348          701
```

```
70667     stepan     trofimovitch    412  20.255550          430
79235    varvara         petrovna    331  18.152786          379
48499   nastasia       philipovna    215  14.638661          362
50655          o            clock    157  12.518405          194
49751    nikolay  vsyevolodovitch    198  14.041475          354
26366     fyodor       pavlovitch    246  15.654434          260
41331  lizabetha      prokofievna    136  11.652786          153
29888         ha               ha    144  11.980779          241
37238   katerina         ivanovna    229  15.092046          253
28920      great             deal    165  12.764221         1202
84079      yulia       mihailovna    105  10.238928          115
44546    mavriky     nikolaevitch     96   9.792194          112
3791     avdotya        romanovna     86   9.269403           92
62183     rodion      romanovitch     80   8.940855           95
51320        old              man    290  16.224287         1356
16696     dmitri     fyodorovitch    126  11.185020          354
58974      pyotr       petrovitch    143  11.882136          701
83930      young              man    234  14.784147          776
51481        old            woman    215  14.278359         1356

       word2_count   expected     chi-square  likelihood_ratio_score
58985          502   1.397501  129615.257394             2049.102849
70667          502   0.857240  197188.981543             1998.069682
79235          491   0.739011  147592.189178             1785.934891
48499          247   0.355088  129749.351666             1781.342890
50655          188   0.144841  169866.430868             1685.741999
49751          298   0.418938   93183.845170             1659.786223
26366          455   0.469802  128320.075280             1650.990245
41331          175   0.106331  173675.473950             1639.129566
29888          241   0.230656   89612.381962             1594.230357
37238          613   0.615902   84687.683108             1582.494087
28920          218   1.040618   25833.372512             1546.010868
84079          180   0.082205  133905.212640             1531.818564
44546          127   0.056487  162959.246589             1470.378026
3791           107   0.039093  189016.577086             1394.560562
62183           81   0.030559  209270.985725             1355.030643
51320         2546  13.710351    5567.762030             1310.268284
16696          319   0.448461   35149.541283             1278.534223
58974          327   0.910325   22178.327775             1266.757179
83930         2546   7.846041    6518.651914             1231.972141
51481         1047   5.638153    7774.245214             1226.633161
```

Part 3

In [320…

```
bigrams_size1_likelihood[(bigrams_size1_likelihood['word1']=='head') & (bigrams_size1_like
```

Out[320…

	word1	word2	counts	t-score	word1_count	word2_count	expected	chi-square	likelihood_ratio_score
8917	head	clerk	22	4.598527	798	136	0.430995	1079.413746	134.120965

In [321…

```
bigrams_size1_likelihood[(bigrams_size1_likelihood['word1']=='great') & (bigrams_size1_lik
```

Out[321…

	word1	word2	counts	t-score	word1_count	word2_count	expected	chi-square	likelihood_ratio_score
8462	great	man	18	1.378086	1202	2546	12.153276	2.812755	2.488797

In [262…

```
import dataframe_image as dfi
dfi.export(bigrams_size1_likelihood.head(),"mytableFinal.png")
```