Berkay ALTINTAŞ

22002709

EEE 486 Statistical Foundations of Natural Language Processing Assignment 1 Report

Part 1: Corpus Preprocessing

In this part, I have downloaded the given corpus, tokenized the text, found the part-of-speech (POS) tags of the tokens, which was useful to find the lemma for each token and filter the collocation candidates throughtout the task, and lemmatized the tokens. Before starting to apply any of these steps, the all characters in the corpus were converted into lower case.Moreover, these steps were done in the order they were stated and the corresponding results after each step could be seen below.

```
['i',
 '_lady',
 'susan',
 'vernon',
 'to',
 'mr.',
 'vernon._',
 'langford',
 ',',
 'dec.',
 'my',
 'dear',
 'brother',
 ',',
 '—i',
 'can',
 'no',
 'longer',
 'refuse',
 'myself',
 'the',
 'pleasure',
 'of',
 'profiting',
 'by',
...
 'charles',
 'vernon',
 'is',
 'my',
 ...]
```

Figure 1 *After Corpus Tokenization*

```
[('i', 'NOUN'),
 ('_lady', 'VERB'),
 ('susan', 'ADJ'),
 ('vernon', 'NOUN'),
 ('to', 'PRT'),
 ('mr.', 'VERB'),
 ('vernon._', 'NOUN'),
 ('langford', 'NOUN'),
 (',', '.'),
 ('dec.', 'VERB'),
 ('my', 'PRON'),
 ('dear', 'NOUN'),
 ('brother', 'NOUN'),
 (',', '.'),
 ('—i', 'NOUN'),
 ('can', 'VERB'),
 ('no', 'ADV'),
 ('longer', 'ADV'),
 ('refuse', 'VERB'),
 ('myself', 'PRON'),
 ('the', 'DET'),
 ('pleasure', 'NOUN'),
 ('of', 'ADP'),
 ('profiting', 'VERB'),
 ('by', 'ADP'),
...
 ('charles', 'NOUN'),
 ('vernon', 'NOUN'),
 ('is', 'VERB'),
 ('my', 'PRON'),
 ...]
```

Figure 2 *After POS Tagging*

As it could be seen in Figure 1, this kind of tokenization does not eliminate the punctuations which would be handled later in the task. In the Figure 2, it can be seen that the tokens were labelled with their POS tags. In this task, universal tagset, which includes 'VERB', 'NOUN', 'PRON', 'ADJ', 'ADV', 'ADP', 'CONJ', 'DET', 'NUM', 'PRT', 'X', '.', was used. After these steps, lemmatization was applied to tokens regarding the custom_lemmatizer.py file given in assignment file. The lemmatized tokens can be seen in Figure 3. For example, it could be observed that the word 'is' was converted into 'be' after lemmatization regarding the Figure 2 and Figure 3.

```
[('i', 'NOUN'),
 ('_lady', 'VERB'),
 ('susan', 'ADJ'),
 ('vernon', 'NOUN'),
 ('to', 'PRT'),
 ('mr.', 'VERB'),
 ('vernon._', 'NOUN'),
 ('langford', 'NOUN'),
 (',', '.'),
 ('dec.', 'VERB'),
 ('my', 'PRON'),
 ('dear', 'NOUN'),
 ('brother', 'NOUN'),
 (',', '.'),
 ('—i', 'NOUN'),
 ('can', 'VERB'),
 ('no', 'ADV'),
 ('longer', 'ADV'),
 ('refuse', 'VERB'),
 ('myself', 'PRON'),
 ('the', 'DET'),
 ('pleasure', 'NOUN'),
 ('of', 'ADP'),
 ('profit', 'VERB'),
 ('by', 'ADP'),
...
 ('charles', 'NOUN'),
 ('vernon', 'NOUN'),
 ('be', 'VERB'),
 ('my', 'PRON'),
 ...]
```

Figure 3 Tokens After Lemmatization

After these steps, the lemmatized tokens were used to find the bigram counts for window size 1 and for window size 3 regarding the criterias given in the assignment. To be more clear, it was asked to eliminate the bigrams except those with POS tags NOUN-NOUN and ADJ-NOUN, bigrams that include stopwords, bigrams that include punctuation marks and bigrams occuring less than 10 times. Thus, the bigrams, which have window size 1 and satisfy these criteria, could be seen in Figure 4. Moreover, the bigrams, which have window size 3 and satisfy these crteria, could be seen in Figure 5.

```
{('sir', 'thomas'): 242,
 ('miss', 'crawford'): 214,
 ('great', 'deal'): 151,
 ('young', 'man'): 150,
 ('young', 'lady'): 135,
 ('mr', 'elliot'): 132,
 ('lady', 'bertram'): 120,
 ('lady', 'russell'): 118,
 ('captain', 'wentworth'): 115,
 ('sir', 'walter'): 111,
 ('lady', 'catherine'): 104,
 ('sir', 'john'): 91,
 ('colonel', 'brandon'): 88,
 ('lady', 'middleton'): 86,
 ('miss', 'tilney'): 71,
 ('miss', 'bingley'): 71,
 ('next', 'morning'): 61,
 ('miss', 'bennet'): 61,
 ('young', 'people'): 58,
 ('next', 'day'): 57,
 ('young', 'woman'): 56,
 ('young', 'men'): 52,
 ('miss', 'morland'): 51,
 ('henry', 'crawford'): 49,
 ('mr', 'smith'): 48,
...
 ('many', 'day'): 10,
 ('good', 'time'): 10,
 ('strong', 'feeling'): 10,
 ('dear', 'jane'): 10,
 ('edward', 'ferrars'): 10}
```

Figure 4 Collocation Candidates with Window Size 1

```
{('sir', 'thomas'): 242,
 ('miss', 'crawford'): 214,
 ('great', 'deal'): 151,
 ('young', 'man'): 151,
 ('young', 'lady'): 135,
 ('mr', 'elliot'): 132,
 ('lady', 'bertram'): 120,
 ('lady', 'russell'): 118,
 ('captain', 'wentworth'): 118,
 ('sir', 'walter'): 111,
 ('lady', 'catherine'): 104,
 ('sir', 'john'): 91,
 ('colonel', 'brandon'): 88,
 ('lady', 'middleton'): 86,
 ('miss', 'tilney'): 71,
 ('miss', 'bingley'): 71,
 ('miss', 'bennet'): 64,
 ('next', 'morning'): 61,
 ('young', 'people'): 59,
 ('young', 'woman'): 59,
 ('next', 'day'): 57,
 ('young', 'men'): 52,
 ('miss', 'morland'): 51,
 ('henry', 'crawford'): 49,
 ('mr', 'smith'): 49,
...
 ('good', 'woman'): 10,
 ('fifty', 'pound'): 10,
 ('high', 'opinion'): 10,
 ('catherine', 'bourgh'): 10,
 ('edward', 'ferrars'): 10}
```

Figure 5 Collocation Candidates with Window Size 3

Part 2: Finding the Collocations

Student's t test

For a given corpus with total of N words, the probability of any bigram could be calculated as follows

$$P(w1, w2) = \frac{count(w1, w2)}{N}$$

where w1 represent the first word of the bigram and w2 the second. Moreover, the probability of any word can be calculated in a similar way like as follows,

$$P(w1) = \frac{count(w1)}{N}$$

$$P(w2) = \frac{count(w2)}{N}$$

The student's t test can be calculated as follows,

$$t = \frac{\bar{X} - \mu}{\sqrt{\frac{S^2}{N}}}$$

where $\bar{X}$ represents sample mean, μ represents the population mean, S^2 represents the sample variance and N represents the sample size N.

Under the assumption of w1 and w2 occur independently, which represents the null hypothesis, the P(w1, w2) can be calculated as follows,

$$\mu \ = P(w1, w2) \ = \ P(w1). P(w2)$$

Moreover, using the Bernoulli Trial principle, the sample variance can be approximated like as follows,

$$var(P(w1, w2)) \ = P(w1, w2) \, . (1 - P(w1, w2))$$

As a result, t test can be calculated like as follows,

$$t = \frac{P(w_1, w_2) - \mu}{\sqrt{\frac{\mathrm{Var}\left(P(w_1, w_2)\right)}{N}}}$$

In coding part, the N was chosen as the number of lemmatized words. For the collocations having window of size 1 it kept as N. However, for the collocations having window of size 3, it used as 3*N since the number of bigrams have tripled. Regarding this and the t test concept, the top 20 collocation candidates sorted by their t scores can be seen in Figure 6 and Figure 7 for window of size 1 and window of size 3 respectively.

	Bigram	T-Score	c(w1w2)	c(w1)	c(w2)
0	(sir, thomas)	15.534467	242	785	335
1	(miss, crawford)	14.550307	214	1318	615
2	(great, deal)	12.263945	151	1005	215
3	(young, man)	12.196974	150	685	636
4	(young, lady)	11.529326	135	685	1057
5	(mr, elliot)	11.472741	132	479	288
6	(lady, bertram)	10.917462	120	1057	270
7	(lady, russell)	10.842880	118	1057	147
8	(captain, wentworth)	10.714699	115	341	216
9	(sir, walter)	10.521421	111	785	139
10	(lady, catherine)	10.104307	104	1057	626
11	(sir, john)	9.514974	91	785	209
12	(colonel, brandon)	9.375672	88	265	140
13	(lady, middleton)	9.254444	86	1057	119
14	(miss, tilney)	8.377604	71	1318	215
15	(miss, bingley)	8.357100	71	1318	305
16	(next, morning)	7.792387	61	269	363
17	(miss, bennet)	7.728505	61	1318	334
18	(young, people)	7.580462	58	685	272
19	(next, day)	7.508892	57	269	795

Figure 6 Top collocation candidates, student's t-test (window size1)

	Bigram	T-Score	c(w1w2)	c(w1)	c(w2)
0	(sir, thomas)	15.549056	242	785	335
1	(miss, crawford)	14.602598	214	1318	615
2	(great, deal)	12.280119	151	1005	215
3	(young, man)	12.271444	151	685	636
4	(young, lady)	11.589077	135	685	1057
5	(mr, elliot)	11.483664	132	479	288
6	(lady, bertram)	10.942122	120	1057	270
7	(captain, wentworth)	10.859800	118	341	216
8	(lady, russell)	10.856147	118	1057	147
9	(sir, walter)	10.530910	111	785	139
10	(lady, catherine)	10.166797	104	1057	626
11	(sir, john)	9.531253	91	785	209
12	(colonel, brandon)	9.379112	88	265	140
13	(lady, middleton)	9.267227	86	1057	119
14	(miss, tilney)	8.409968	71	1318	215
15	(miss, bingley)	8.403134	71	1318	305
16	(miss, bennet)	7.973410	64	1318	334
17	(next, morning)	7.804296	61	269	363
18	(young, people)	7.669480	59	685	272
19	(young, woman)	7.663288	59	685	415

Figure 7 Top collocation candidates, student's t-test (window size3)

Chi-Squared Test

The contingency table created regarding the bigrams I have can be seen below,

	word w2	**Not word w2**	**Total (Row)**
word w1	O_{11} = c(w1,w2)	O_{12} = c(w1) - c(w1,w2)	c(w1)
not word w1	O_{21} = c(w2) - c(w1, w2)	O_{22} = N – (c(w1) +c(w2) + c(w1, w2))	N – c(w1)
Total (Column)	C(w2)	N – c(w2)	N

Table 1: Contingency Table for Bigrams

where O_{11} represents the bigram count, O_{12} represents the occurrences of w1 without w2, O_{21} represents the occurrences of w2 without w1 and O_{22} represents the occurrences of bigrams which do not have w1 and w2. Moreover, N represent the number of lemmatized tokens.

Under the null hypothesis which indicates the independency between w1 and w2, the expected counts of the events can be calculated as follows,

$$E_{i,j} = \frac{(\text{Total of row i}) \times (\text{Total of column j})}{N}$$

$$E_{11} = \frac{count(w1).count(w2)}{N}$$

$$E_{12} = \frac{count(w1).(N - count(w2))}{N}$$

$$E_{21} = \frac{(N - count(w1)).count(w2)}{N}$$

$$E_{22} = \frac{(N - count(w1)).(N - count(w2))}{N}$$

Regarding the observed and expected counts of the events, the Chi-Square statistic can be calculated like as follows,

$$\chi^2 = \sum \frac{\left(O_{ij} - E_{ij}\right)^2}{E_{ij}}$$

$$\chi^2 = \frac{(O_{11} - E_{11})^2}{E_{11}} + \frac{(O_{12} - E_{21})^2}{E_{12}} + \frac{(O_{21} - E_{21})^2}{E_{21}} + \frac{(O_{22} - E_{22})^2}{E_{22}}$$

If the chi-squared statistic is high, it means that there is a strong relationship between w1 and w2.

In coding part, for the collocations having window of size 1, N is kept same. However, for the collocations having window of size 3, N was used as 3*N since there the bigram counts tripled. The top 20 candidate collocations could be seen in Figure 8 and Figure 9 for window of size 1 and window of size 3 respectively.

	Bigram	Chi-Square Score	c(w1w2)	c(w1)	c(w2)
0	(thornton, lacey)	469979.208427	16	22	17
1	(sir, thomas)	152673.775053	242	785	335
2	(colonel, brandon)	143228.466218	88	265	140
3	(captain, wentworth)	123152.862102	115	341	216
4	(mrs, clay)	102021.953523	35	123	67
5	(mr, elliot)	86556.014982	132	479	288
6	(sir, walter)	77413.550446	111	785	139
7	(great, deal)	72281.077855	151	1005	215
8	(kellynch, hall)	64625.631212	20	72	59
9	(lady, russell)	61401.870217	118	1057	147
10	(berkeley, street)	48322.682720	15	17	188
11	(lady, middleton)	40270.062572	86	1057	119
12	(milsom, street)	38562.070959	13	16	188
13	(harley, street)	38562.070959	13	16	188
14	(captain, benwick)	38493.197459	35	341	64
15	(miss, crawford)	38474.325414	214	1318	615
16	(human, nature)	36147.109563	18	41	150
17	(young, man)	35229.301762	150	685	636
18	(pulteney, street)	35047.710505	12	15	188
19	(sir, john)	34524.493774	91	785	209

Figure 8 Top collocation candidates, chi-squared statistics (window size 1)

	Bigram	Chi-Square Score	c(w1w2)	c(w1)	c(w2)
0	(thornton, lacey)	1409948.235251	16	22	17
1	(o, clock)	947549.555802	32	42	53
2	(sir, thomas)	458490.708207	242	785	335
3	(colonel, brandon)	429868.354651	88	265	140
4	(captain, wentworth)	389266.040522	118	341	216
5	(mrs, clay)	306149.378271	35	123	67
6	(mr, elliot)	260002.323009	132	479	288
7	(sir, walter)	232476.025993	111	785	139
8	(great, deal)	217189.826336	151	1005	215
9	(lover, vow)	204337.936494	10	56	18
10	(kellynch, hall)	193932.222331	20	72	59
11	(lady, russell)	184461.871835	118	1057	147
12	(berkeley, street)	144999.184812	15	17	188
13	(lady, middleton)	121015.900778	86	1057	119
14	(miss, crawford)	116059.972679	214	1318	615
15	(milsom, street)	115715.290497	13	16	188
16	(harley, street)	115715.290497	13	16	188
17	(captain, benwick)	115574.120713	35	341	64
18	(human, nature)	108493.200423	18	41	150
19	(young, man)	107573.116971	151	685	636

Figure 9 Top collocation candidates, chi-squared statistics (window size 3)

Likelihood Ratio Test

In the Likelihood Ratio Test (LRT), I have introduced two different hypotheses which are null hypothesis and alternative hypothesis. In the null hypothesis, hypothesis says that w1 and w2 occur independently. Thus, in any bigram, the probability of having w2 after given w1 could be formulated like as follows,

$$P(w2) = \frac{count(w2)}{N}$$

where N represents the number of lemmatized tokens. On the other hand, the alternative hypothesis states that there is a dependency between w1 and w2 which means that for any bigram, the probability of having w2 could be formulated with given w1. Thus this hypothesis could be formulated like as follows,

$$P(w1) = \frac{count(w1,\ w2)}{count(w1)}$$

$$P(w2) = \frac{count(w2)\ -\ count(w1, w2)}{N\ -\ count(w1)}$$

If the log-likelihood of these hypotheses are taken, then the log-likelihood functions of the null hypothesis and alternative hypothesis could be written respectively like as follows,

$$L_{H_0} = log\binom{c_1}{c_{12}} + c_{12}\log p + (c_1 - c_{12})\log(1-p) + \log\binom{N-c_1}{c_2-c_{12}} + (c_2 - c_{12})\log p + \big(N - c_1 - (c_2 - c_{12})\big)\log(1-p)$$

and

$$L_{H_1} = log\binom{c_1}{c_{12}} + c_{12}\log p_1 + (c_1 - c_{12})\log(1-p_1) + \log\binom{N-c_1}{c_2-c_{12}} + (c_2 - c_{12})\log p_2 + \big(N - c_1 - (c_2 - c_{12})\big)\log(1-p_2)$$

where c1 represents count(w1), c2 represents count(w2) and c12 represents count(w1, w2).

As a results, the log-likelihood ratio can be formulated like as follows,

$$Log - likelihood\ Ratio = \ -2(L_{H_0}\ -\ L_{H_1})$$

In coding part, for the collocations having window of size 1, N was kept same. However, for the collocatons having window of size 3, N was used as 3*N since the bigram counts tripled.

The top 20 candidate collocations could be seen in Figure 10 and Figure 11 for window of size 1 and window of size 3 respectively.

	Bigram	LLR Score	c(w1w2)	c(w1)	c(w2)
0	(sir, thomas)	2966.575259	242	785	335
1	(miss, crawford)	1920.644184	214	1318	615
2	(great, deal)	1733.315818	151	1005	215
3	(mr, elliot)	1561.860969	132	479	288
4	(captain, wentworth)	1495.346361	115	341	216
5	(young, man)	1414.479744	150	685	636
6	(lady, russell)	1396.166374	118	1057	147
7	(sir, walter)	1380.659628	111	785	139
8	(colonel, brandon)	1231.754358	88	265	140
9	(lady, bertram)	1197.922420	120	1057	270
10	(young, lady)	1088.122688	135	685	1057
11	(lady, middleton)	980.694598	86	1057	119
12	(sir, john)	957.811652	91	785	209
13	(lady, catherine)	796.071980	104	1057	626
14	(next, morning)	643.549900	61	269	363
15	(miss, tilney)	619.948137	71	1318	215
16	(mr, smith)	568.289148	48	479	97
17	(miss, bingley)	562.043685	71	1318	305
18	(young, people)	525.108035	58	685	272
19	(mrs, clay)	522.240335	35	123	67

Figure 10 Top collocation candidates, chi-squared statistics (window size 3)

	Bigram	LLR Score	c(w1w2)	c(w1)	c(w2)
0	(sir, thomas)	3498.262308	242	785	335
1	(miss, crawford)	2390.033419	214	1318	615
2	(great, deal)	2065.012608	151	1005	215
3	(mr, elliot)	1851.806335	132	479	288
4	(captain, wentworth)	1803.686524	118	341	216
5	(young, man)	1757.749776	151	685	636
6	(lady, russell)	1655.399453	118	1057	147
7	(sir, walter)	1624.526840	111	785	139
8	(lady, bertram)	1461.330068	120	1057	270
9	(colonel, brandon)	1425.099760	88	265	140
10	(young, lady)	1383.779305	135	685	1057
11	(lady, middleton)	1169.600798	86	1057	119
12	(sir, john)	1157.607951	91	785	209
13	(lady, catherine)	1023.626236	104	1057	626
14	(next, morning)	777.462156	61	269	363
15	(miss, tilney)	775.606757	71	1318	215
16	(miss, bingley)	717.483992	71	1318	305
17	(mr, smith)	690.663016	49	479	97
18	(young, people)	665.895405	59	685	272
19	(o, clock)	655.354581	32	42	53

Figure 11 Top collocation candidates, chi-squared statistics (window size 3)

Part 3: Explaining the Statistical Tests

As introduced in Part 2, in Student's t test, we check whether the occurence of the bigrams is more than expected or not. In Chi-Square test, we measure the strength of the association between words in bigram. In likelihood ratio, we measure the word dependence by comparing the null and alternative hypothesis. The formulas used for these tests could be seen below. For detailed analysis and derivation check Part 2.

For Student's the test, we have

$$t = \frac{P(w_1, w_2) - \mu}{\sqrt{\frac{\mathrm{Var}\left(P(w_1, w_2)\right)}{N}}}$$

For chi-squared test, we have

$$\chi^2 = \sum \frac{\left(O_{ij} - E_{ij}\right)^2}{E_{ij}}$$

For Likelihood Ratio test, we have

$$Log - likelihood\ Ratio = -2(L_{H_0} - L_{H_1})$$

The statistics for the bigrams 'good wish' and 'high spirit' can be seen in Figure 12. By using these statistics and the formulas above, each test could be applied.

```
{'good': 1198, 'wish': 817, 'high': 161, 'spirit': 358}
{('high', 'spirit'): 16, ('good', 'wish'): 11}
```

Figure 12 Some Statistics for Asked Bigrams in Part 3

Then, for significance of α = 0.005, I have checked whether the bigrams are collocations or not using the one-tailed t table and chi-square table. To be more clear, to find the correct number in the table, the degree of freedom is used. For one-tailed t table, since we had a long text and huge numbers of tokens, the degree of freedom was taken as infinity which gives 2.576 for α = 0.005. For Chi-Square table, if we look at the Table 1, we could observe that the degree of freedom is 1 due to the following formula,

$$df = (rows - 1)(columns - 1)$$

Thus, we have 7.897 for α = 0.005 and degree of freedom 1.

Therefore, the results for bigrams 'good wish' and 'high spirit' could be seen in Figure 13.

	Bigram	T-Score	Chi-Square	LLR	c(w1w2)	c(w1)	c(w2)	T-Collocation	Chi-Collocation	LLR-Collocation
0	(good, wish)	2.886847	64.497822	25.995148	11	1198	817	True	True	True
1	(high, spirit)	3.979060	3019.989973	138.521542	16	161	358	True	True	True

Figure 13 Different Test Results for Asked Bigrams in Part 3

As it could be seen in Figure 13, each bigram gives True for each test which means that the asked bigrams are collocations.

APPENDIX

```
pip install numpy pandas nltk

import numpy as np

import pandas as pd

import nltk

from nltk.tokenize import word_tokenize

from nltk.stem import WordNetLemmatizer

from nltk.corpus import wordnet, stopwords

import math

nltk.download('all')

with open('Jane Austen Processed.txt', 'r', encoding = 'utf-8') as file:

   corpus = file.read().lower()

tokens = word_tokenize(corpus)

tokens

print(len(tokens))

pos_tags = nltk.pos_tag(tokens, tagset = 'universal')

pos_tags

class custom_lemmatizer:


   tag_dict = {"ADJ": wordnet.ADJ,

            "NOUN": wordnet.NOUN,

            'VERB': wordnet.VERB,

            'ADV': wordnet.ADV}


   lemmatizer = WordNetLemmatizer()


   def lemmatize(self, word_pos_tuple):

      word = word_pos_tuple[0]

      pos_tag = word_pos_tuple[1]

      if pos_tag in self.tag_dict:

         return self.lemmatizer.lemmatize(word, self.tag_dict[pos_tag]).lower()
```

```
        else:
            return word.lower()
lemmatizer = custom_lemmatizer()
lemmatized_tokens = []
for token, pos in pos_tags:
    lemma = lemmatizer.lemmatize((token, pos))
    lemmatized_tokens.append((lemma, pos))

lemmatized_tokens

lemmatized_words = [word for word, pos in lemmatized_tokens]

targets = ['that', 'the', 'london', 'honor', '.']
targets_counts = [lemmatized_words.count(target) for target in targets]

print(targets_counts)

def filtering_bigrams(lemmatized_tokens, window_size):

    filtered_bigrams = []

    for i in range(len(lemmatized_tokens) - 1):  # Iterate over words
        for j in range(i + 1, min(i + 1 + window_size, len(lemmatized_tokens))):  # Vary second word by
window size
            word1, pos1 = lemmatized_tokens[i]   # First word & POS
            word2, pos2 = lemmatized_tokens[j]   # Second word & POS

            # Apply NOUN-NOUN or ADJ-NOUN filtering condition
            if (pos1 == 'NOUN' and pos2 == 'NOUN') or (pos1 == 'ADJ' and pos2 == 'NOUN'):
                filtered_bigrams.append((word1, word2))
```

```
    return filtered_bigrams

def clean_stopwords_bigram(bigrams):

    custom_stopwords = [
        'i', 'me', 'my', 'myself', 'we', 'our', 'ours', 'ourselves', 'you', 'your', 'yours', 'yourself', 'yourselves',
        'he', 'him', 'his', 'himself', 'she', 'her', 'hers', 'herself', 'it', 'its', 'itself', 'they', 'them', 'their',
        'theirs', 'themselves', 'what', 'which', 'who', 'whom', 'this', 'that', 'these', 'those', 'am', 'is', 'are',
        'was', 'were', 'be', 'been', 'being', 'have', 'has', 'had', 'having', 'do', 'does', 'did', 'doing', 'a', 'an',
        'the', 'and', 'but', 'if', 'or', 'because', 'as', 'until', 'while', 'of', 'at', 'by', 'for', 'with', 'about',
        'against', 'between', 'into', 'through', 'during', 'before', 'after', 'above', 'below', 'to', 'from', 'up',
        'down', 'in', 'out', 'on', 'off', 'over', 'under', 'again', 'further', 'then', 'once', 'here', 'there', 'when',
        'where', 'why', 'how', 'all', 'any', 'both', 'each', 'few', 'more', 'most', 'other', 'some', 'such', 'no',
'nor',
        'not', 'only', 'own', 'same', 'so', 'than', 'too', 'very', 's', 't', 'can', 'will', 'just', 'don', 'should', 'now']

    stop_words = set(custom_stopwords)

    cleaned_stopword_bigrams = []
    for bigram in bigrams:
        if (bigram[0] not in stop_words) and (bigram[1] not in stop_words):
            cleaned_stopword_bigrams.append(bigram)

    return cleaned_stopword_bigrams

def clean_punctuations_bigram(bigrams):
    cleaned_punct_bigrams = []
    for token_1, token_2 in bigrams:
        if token_1.isalpha() == True and token_2.isalpha() == True:
            cleaned_punct_bigrams.append((token_1, token_2))
```

```
    return cleaned_punct_bigrams

def count_bigrams(bigrams):

    bigrams_counts = {}
    for bigram in bigrams:
        if bigram in bigrams_counts:
            bigrams_counts[bigram] += 1
        else:
            bigrams_counts[bigram] = 1

    sorted_bigram_counts = sorted(bigrams_counts.items(), key = lambda x: x[1], reverse = True)
    return sorted_bigram_counts

def thresholding_bigrams(bigrams_counts, threshold):
    bigrams_freq = {}

    for bigram, freq in dict(bigrams_counts).items():
        if freq >= threshold:
            bigrams_freq[bigram] = freq

    return bigrams_freq

pos_filtered_bigrams_ws1 = filtering_bigrams(lemmatized_tokens, window_size = 1)
stopwords_cleaned_bigrams_ws1 = clean_stopwords_bigram(pos_filtered_bigrams_ws1)
punct_cleaned_bigrams_ws1 = clean_punctuations_bigram(stopwords_cleaned_bigrams_ws1)
count_cleaned_bigrams_ws1 = count_bigrams(punct_cleaned_bigrams_ws1)
final_bigrams_ws1 = thresholding_bigrams(count_cleaned_bigrams_ws1 , threshold = 10)

final_bigrams_ws1
target_bigram = ('mr.','skimpole')
```

```
count = (final_bigrams_ws1.get(target_bigram, 0))

count

pos_filtered_bigrams_ws3 = filtering_bigrams(lemmatized_tokens, window_size = 3)

stopwords_cleaned_bigrams_ws3 = clean_stopwords_bigram(pos_filtered_bigrams_ws3)

punct_cleaned_bigrams_ws3 = clean_punctuations_bigram(stopwords_cleaned_bigrams_ws3)

count_cleaned_bigrams_ws3 = count_bigrams(punct_cleaned_bigrams_ws3)

final_bigrams_ws3 = thresholding_bigrams(count_cleaned_bigrams_ws3 , threshold = 10)


final_bigrams_ws3

target_bigram = ('large','fortune')

count = (final_bigrams_ws3.get(target_bigram, 0))

count


lemmatized_words = [word for word, pos in lemmatized_tokens]


targets = ['sir', 'thomas', 'miss', 'crawford', 'great', 'deal','young', 'man', 'young', 'lady', 'mr', 'elliot', 
'lady', 'bertram',

        'lady', 'russell', 'captain', 'wentworth', 'sir', 'walter', 'lady','catherine', 'sir', 'john', 'colonel', 
'brandon','lady',

        'middleton', 'miss', 'tilney','miss', 
'bingley','next','morning','miss','bennet','young','people','next','day']

targets_counts = [lemmatized_words.count(target) for target in targets]


print(targets_counts)


N = len(lemmatized_tokens)

N

word_frequencies = {}

for word, pos in lemmatized_tokens:

  word_frequencies[word] = word_frequencies.get(word, 0) + 1


word_frequencies
```

```python
def compute_t_score(c_w1_w2, c_w1, c_w2, N):

    p_w1_w2 = c_w1_w2 / N
    p_w1 = c_w1 / N
    p_w2 = c_w2 / N

    mu = p_w1 * p_w2

    variance = p_w1_w2 * (1 - p_w1_w2)

    if variance == 0:
        return 0

    t_score = (p_w1_w2 - mu) / math.sqrt(variance / N)
    return t_score

def calculate_t_scores(bigram_counts, word_frequencies, N):

    t_scores = []

    for (w1, w2), c_w1_w2 in bigram_counts.items():
        c_w1 = word_frequencies.get(w1, 1)
        c_w2 = word_frequencies.get(w2, 1)

        t_score = compute_t_score(c_w1_w2, c_w1, c_w2, N)
        t_scores.append(((w1, w2), t_score, c_w1_w2, c_w1, c_w2))



    t_scores.sort(key=lambda x: x[1], reverse=True)
```

```
    return t_scores[:20]

N = len(lemmatized_tokens)

t_score_results_ws1 = calculate_t_scores(final_bigrams_ws1, word_frequencies, N)

table_1 = pd.DataFrame(t_score_results_ws1, columns=["Bigram", "T-Score", "c(w1w2)", "c(w1)", "c(w2)"])
table_1

def compute_chi_square(c_w1_w2, c_w1, c_w2, N):

    O_11 = c_w1_w2
    O_12 = c_w1 - c_w1_w2
    O_21 = c_w2 - c_w1_w2
    O_22 = N - (O_11 + O_12 + O_21)


    E_11 = (c_w1 * c_w2) / N
    E_12 = (c_w1 * (N - c_w2)) / N
    E_21 = ((N - c_w1) * c_w2) / N
    E_22 = ((N - c_w1) * (N - c_w2)) / N

    if min(E_11, E_12, E_21, E_22) == 0:
        return 0


    chi_square = ((O_11 - E_11) ** 2) / E_11 + ((O_12 - E_12) ** 2) / E_12 \
                 + ((O_21 - E_21) ** 2) / E_21 + ((O_22 - E_22) ** 2) / E_22

    return chi_square
```

```
def calculate_chi_square_scores(bigram_counts, word_frequencies, N):

    chi_square_scores = []

    for (w1, w2), c_w1_w2 in bigram_counts.items():
        c_w1 = word_frequencies.get(w1, 1)
        c_w2 = word_frequencies.get(w2, 1)

        chi_square_score = compute_chi_square(c_w1_w2, c_w1, c_w2, N)
        chi_square_scores.append(((w1, w2), chi_square_score, c_w1_w2, c_w1, c_w2))


    chi_square_scores.sort(key=lambda x: x[1], reverse=True)

    return chi_square_scores[:20]

N = len(lemmatized_tokens)

chi_square_results_ws1 = calculate_chi_square_scores(final_bigrams_ws1, word_frequencies, N)
table_2 = pd.DataFrame(chi_square_results_ws1, columns=["Bigram", "Chi-Square Score",
"c(w1w2)", "c(w1)", "c(w2)"])
table_2
import math
import pandas as pd

def compute_likelihood_ratio(c_w1_w2, c_w1, c_w2, N):

    p = c_w2 / N
```

```
    p1 = c_w1_w2 / c_w1 if c_w1 > 0 else 1e-10
    p2 = (c_w2 - c_w1_w2) / (N - c_w1) if (N - c_w1) > 0 else 1e-10


    def log_likelihood(k, n, p):
        if k == 0 or n == 0:
            return 0
        return k * math.log(p) + (n - k) * math.log(1 - p)

    L_H0 = log_likelihood(c_w1_w2, c_w1, p) + log_likelihood(c_w2 - c_w1_w2, N - c_w1, p)
    L_H1 = log_likelihood(c_w1_w2, c_w1, p1) + log_likelihood(c_w2 - c_w1_w2, N - c_w1, p2)


    return -2 * (L_H0 - L_H1)

def calculate_likelihood_ratios(bigram_counts, word_frequencies, N):

    llr_scores = []

    for (w1, w2), c_w1_w2 in bigram_counts.items():
        c_w1 = word_frequencies.get(w1, 1)
        c_w2 = word_frequencies.get(w2, 1)

        llr_score = compute_likelihood_ratio(c_w1_w2, c_w1, c_w2, N)
        llr_scores.append(((w1, w2), llr_score, c_w1_w2, c_w1, c_w2))


    llr_scores.sort(key=lambda x: x[1], reverse=True)

    return llr_scores[:20]
```

```
llr_results_ws1 = calculate_likelihood_ratios(final_bigrams_ws1, word_frequencies, N)


table_3 = pd.DataFrame(llr_results_ws1, columns=["Bigram", "LLR Score", "c(w1w2)", "c(w1)", "c(w2)"])

table_3

N = len(lemmatized_tokens)


t_score_results_ws3 = calculate_t_scores(final_bigrams_ws3, word_frequencies, 3*N)

table_4 = pd.DataFrame(t_score_results_ws3, columns=["Bigram", "T-Score", "c(w1w2)", "c(w1)", "c(w2)"])

table_4

N = len(lemmatized_tokens)


chi_square_results_ws3 = calculate_chi_square_scores(final_bigrams_ws3, word_frequencies, 3*N)

pd.set_option('display.float_format', lambda x: '%.6f' % x)

table_5 = pd.DataFrame(chi_square_results_ws3, columns=["Bigram", "Chi-Square Score", "c(w1w2)", "c(w1)", "c(w2)"])

table_5


llr_results_ws3 = calculate_likelihood_ratios(final_bigrams_ws3, word_frequencies, 3*N)


table_6 = pd.DataFrame(llr_results_ws3, columns=["Bigram", "LLR Score", "c(w1w2)", "c(w1)", "c(w2)"])

table_6

bigram_counts = {

   ("good", "wish"): final_bigrams_ws1.get(("good", "wish"), 0),

   ("high", "spirit"): final_bigrams_ws1.get(("high", "spirit"), 0)

}


word_frequencies_filtered = {
```

```
    "good": word_frequencies.get("good", 0),
    "wish": word_frequencies.get("wish", 0),
    "high": word_frequencies.get("high", 0),
    "spirit": word_frequencies.get("spirit", 0)
}


target_bigrams = {("good", "wish"), ("high", "spirit")}

bigram_counts_filtered = {bigram: count for bigram, count in final_bigrams_ws1.items() if bigram in target_bigrams}

word_frequencies_filtered = {word: word_frequencies.get(word, 1) for bigram in target_bigrams for word in bigram}

print(word_frequencies_filtered)
print(bigram_counts_filtered)

target_bigrams = [("good", "wish"), ("high", "spirit")]

bigram_counts_filtered = {bigram: count for bigram, count in final_bigrams_ws1.items() if bigram in target_bigrams}

word_frequencies_filtered = {word: word_frequencies.get(word, 1) for bigram in target_bigrams for word in bigram}

N = len(lemmatized_tokens)

results = []
for bigram in target_bigrams:
    w1, w2 = bigram
    c_w1_w2 = bigram_counts_filtered.get(bigram, 0)
```

```
    c_w1 = word_frequencies_filtered.get(w1, 1)
    c_w2 = word_frequencies_filtered.get(w2, 1)

    t_score = compute_t_score(c_w1_w2, c_w1, c_w2, N)
    chi_square = compute_chi_square(c_w1_w2, c_w1, c_w2, N)
    llr = compute_likelihood_ratio(c_w1_w2, c_w1, c_w2, N)
    results.append((bigram, t_score, chi_square, llr, c_w1_w2, c_w1, c_w2))

df_scores = pd.DataFrame(results, columns=["Bigram", "T-Score", "Chi-Square", "LLR", "c(w1w2)",
"c(w1)", "c(w2)"])

t_threshold = 2.576
chi_threshold = 7.879
llr_threshold = 7.879

# Decision Making for Collocation
df_scores["T-Collocation"] = df_scores["T-Score"] > t_threshold
df_scores["Chi-Collocation"] = df_scores["Chi-Square"] > chi_threshold
df_scores["LLR-Collocation"] = df_scores["LLR"] > llr_threshold

df_scores
```