**NLP Homework 1**

**Comparing Corpora with Corpus Statistics**

Student: Sharat Sripada (vssripad@syr.edu)

### Introduction

#### 1.1 Choosing the data

Options:

1. Choose existing large documents from NLTK or from the Gutenberg collection on the web, or
2. Collect your own data, by using your own documents or collecting data from other sources. Combine the text from these sources to make two documents for the corpora for the first task. Describe the method that you used to define and collect the data, including the difference between the documents. Note any limitations to the method or the text that you were able to find. Do preprocessing to get the text in a suitable format for processing and describe what you did.

I chose option (a) and decided to use the Gutenberg corpus available in the NLTK library. Specifically, was interested to explore Shakespeare’s work and found the following documents/corpus:

Graphical user interface, text, application, email

Description automatically generated

As seen in the code excerpt above, I chose Caesar and Hamlet.

Although both plays were written in about same period (published in 1599), I intend to study the nuances around Shakespearean literature using the techniques taught with corpus statistics.

### Processing the corpus

Using the NLP and NLTK libraries we will attempt to answer the following questions:

* list the top 50 words by frequency (normalized by the length of the document)
* list the top 50 bigrams by frequencies, and
* list the top 50 bigrams by their Mutual Information scores (using min frequency 5)

#### 2.1 List the top 50 words by frequency (normalized by the length of the document)

As a first step, we would tokenize the corpus.

Tokenizing essentially is to translate prose sentences to smaller items constituting words, punctuation-marks etc. and contain it in a Python data-structure called a list - which is then iterable, can be indexed and therefore easily filtered or munged.

This step is repeated for both corpora and the *FreqDist* utility is used to mine for frequencies of the words (sorted in descending order)

A picture containing text

Description automatically generated

Output:

Most common - top(50) from Caeser corpus:

(',', 2204)

('.', 1275)

('I', 530)

('the', 502)

(':', 499)

('and', 409)

('to', 370)

('you', 342)

('of', 336)

('?', 296)

('not', 255)

('is', 247)

('a', 232)

('And', 218)

('in', 204)

('that', 200)

Most common - top(50) from Hamlet corpus:

(',', 2892)

('.', 1879)

('the', 860)

('and', 605)

('of', 576)

('to', 574)

(':', 566)

('I', 550)

('you', 479)

('?', 459)

('a', 435)

('my', 435)

('it', 354)

('is', 349)

('in', 347)

('Ham', 337)

('not', 313)

(';', 298)

('his', 266)

('And', 257)

('that', 256)

('your', 233)

('this', 231)

The output shows an excerpt of the top-50 words from both the documents.

Notice several punctuation marks (like ?, ‘s, &, ‘re etc.) and words that do not necessarily give much information of the corpus (like the/The, of, by, is etc.).

We will use the following methods or techniques to bring out more insights:

* converting all the words to lower-case
* filtering to remove:
  + punctuation marks
  + non-alphabetical words
  + stop-words – using the stopword corpus provided with the NLTK library and appending a custom list of literature

Graphical user interface, text, application, email

Description automatically generated

Graphical user interface, text, application

Description automatically generated with medium confidence

Correspondingly, the word frequencies (normalized to total words) are captured below:

Normalized by word-frequency - top(50) from Caesar corpus

caesar: 0.01783033033033033

brutus: 0.015108858858858858

bru: 0.014358108108108109

haue: 0.013795045045045045

shall: 0.01173048048048048

thou: 0.010792042042042042

cassi: 0.01004129129129129

cassius: 0.007976726726726727

antony: 0.007038288288288288

come: 0.006944444444444444

let: 0.006662912912912913

good: 0.006569069069069069

Normalized by word-frequency - top(50) from Hamlet corpus

ham: 0.02215938979484482

lord: 0.013874276696475538

haue: 0.011507101525512887

king: 0.011309836927932667

shall: 0.00703577064702788

come: 0.006838506049447659

let: 0.006838506049447659

thou: 0.006838506049447659

hamlet: 0.006575486586007364

good: 0.006443976854287217

hor: 0.0062467122567069966

thy: 0.005917937927406628

enter: 0.00558916359810626

#### List the top 50 bigrams by frequencies

Next using the technique of word collocations, we will specifically collect frequencies for bigrams (two-words grouped together) using the corresponding *NLTK Collocations* packages:

Graphical user interface, text, application

Description automatically generated

An excerpt of the output is shown below:

Bigram measures for top-50 words - Caesar corpus

('let', 'vs') 0.0015015015015015015

('wee', 'l') 0.0014076576576576576

('mark', 'antony') 0.00121996996996997

('marke', 'antony') 0.0011261261261261261

('lord', 'bru') 0.0010322822822822822

('thou', 'art') 0.0010322822822822822

('brutus', 'cassius') 0.0009384384384384384

('art', 'thou') 0.0008445945945945946

('caesar', 'caes') 0.0008445945945945946

('caesar', 'shall') 0.0008445945945945946

('enter', 'brutus') 0.0008445945945945946

('noble', 'brutus') 0.0008445945945945946

('thou', 'hast') 0.0008445945945945946

Bigram measures for top-50 words - Hamlet corpus

('lord', 'ham') 0.004668595476065229

('good', 'lord') 0.0015123619147816938

('enter', 'king') 0.0009863229879011047

('hamlet', 'ham') 0.0009863229879011047

('wee', 'l') 0.0008548132561809574

('hor', 'lord') 0.0007890583903208837

('haue', 'seene') 0.0007233035244608101

('lord', 'hamlet') 0.0007233035244608101

('enter', 'hamlet') 0.0006575486586007364

('exeunt', 'enter') 0.0006575486586007364

('ham', 'oh') 0.0006575486586007364

('ham', 'sir') 0.0006575486586007364

('haue', 'heard') 0.0005917937927406628

#### 2.3 List the top 50 bigrams by their Mutual Information scores (using min frequency 5)

Finally, we will use the Point mutual information (PMI) technique with a minimum frequency 5 to retrieve top-50 words from both corpora.

PMI is defined as follows:

*PMI for bigram (word-1, word-2) = log P(word1, word2) / P(word1) . P(word2)*

PMI measures how much more likely words can co-occur than if they were independent. However, this also leads to instances where mis-spelled words can yield mis-leadingly highly PMI (since the individual words themselves need not be present). It is therefore essential to use this method with a min frequency – here defined as 5:

Graphical user interface, text, application

Description automatically generated

Correspondingly, the output is as below:

PMI for Caesar corpus:

(('ides', 'march'), 9.794415866350107)

(('wee', 'l'), 9.19880612142944)

(('caius', 'ligarius'), 8.83094174237522)

(('metellus', 'cymber'), 8.83094174237522)

(('mine', 'owne'), 7.782443224684029)

(('fell', 'downe'), 7.592782005180457)

(('mark', 'antony'), 7.043644472658871)

(('messala', 'messa'), 6.809522758740314)

(('marke', 'antony'), 6.413594082409176)

(('luc', 'sir'), 6.370389583844007)

(('good', 'morrow'), 6.34320475451778)

(('thou', 'hast'), 5.896458395511594)

(('honourable', 'men'), 5.893951539901021)

(('haue', 'seene'), 5.764668522956054)

(('haue', 'beene'), 5.694279195064656)

(('caius', 'cassius'), 5.591475807679831)

PMI for Hamlet corpus:

(('rosincrance', 'guildensterne'), 9.153003278818518)

(('wee', 'l'), 8.686091939181125)

(('sit', 'downe'), 8.514031193394823)

(('horatio', 'marcellus'), 7.57061472176119)

(('set', 'downe'), 7.540026401927767)

(('fathers', 'death'), 7.400689720318875)

(('dost', 'thou'), 6.540026401927767)

(('wilt', 'thou'), 6.514031193394823)

(('heauen', 'earth'), 6.395888734055058)

(('exeunt', 'enter'), 6.281518019341201)

(('enter', 'polonius'), 6.161223785623486)

(('mine', 'owne'), 6.063877388345535)

(('thou', 'art'), 5.999458020565063)

Answers to specific questions:

1. Briefly state why you chose the processing options that you did.

Running the initial frequency measures proved to be inaccurate and insignificant. It became important to run the corpus through a few filters:

* + - remove stop-words (build a custom stop-words list and append it to the original set)
    - convert the corpus to lower-case so we don’t necessarily count words twice
    - this apart, I also filtered out non-alphabet characters mostly comprising punctuation marks

1. Are there any problems with the word or bigram lists that you found? Could you get a better list of bigrams?

While I had already filtered out for most of the redundant stop-words and punctuation marks, one way to improve the bigrams is to remove the low frequency words using *apply\_freq\_filter()* method. However, this did not yield any different results for me.

1. How are the top 50 bigrams by frequency different from the top 50 bigrams scored by Mutual Information?

Most of the word-pairs are common between the two methods. However, owing to the fundamentally different nature of how they operate the ranking results are stacked differently.

Below are the top-50 words for Caesar’s corpus shown side-by-side for comparison:

|  |  |
| --- | --- |
| **Bigrams for Caesar corpus** | **PMI for Caesar corpus** |
| ('let', 'vs') 0.0015015015015015015  ('wee', 'l') 0.0014076576576576576  ('mark', 'antony') 0.00121996996996997  ('marke', 'antony') 0.0011261261261261261  ('lord', 'bru') 0.0010322822822822822  ('thou', 'art') 0.0010322822822822822  ('brutus', 'cassius') 0.0009384384384384384  ('art', 'thou') 0.0008445945945945946  ('caesar', 'caes') 0.0008445945945945946  ('caesar', 'shall') 0.0008445945945945946  .  . | (('ides', 'march'), 9.794415866350107)  (('wee', 'l'), 9.19880612142944)  (('caius', 'ligarius'), 8.83094174237522)  (('metellus', 'cymber'), 8.83094174237522)  (('mine', 'owne'), 7.782443224684029)  (('fell', 'downe'), 7.592782005180457)  (('mark', 'antony'), 7.043644472658871)  (('messala', 'messa'), 6.809522758740314)  (('marke', 'antony'), 6.413594082409176)  (('luc', 'sir'), 6.370389583844007)  (('good', 'morrow'), 6.34320475451778)  .  . |

1. If you modify the stop word list, or expand the methods of filtering, describe that here.

Yes, I appended to the existing list of stop-words, the following:

'could','would','might','must','need','sha','wo','y',"'s","'d","'ll","'t","'m","'re","'ve", "n't"

The code below shows the final list of stop-words:

A picture containing text

Description automatically generated

#### Conclusion

The reason for picking these two Shakespearean corpora was to understand the nuances in literature given both plays were mostly tragic by genre and written or published at the same time in 1599.

Initial analysis showed that size of the Hamlet corpus was ~1.5x of Caesar and this therefore corroborates our knowing of Hamlet being one of Shakespeare's longest written plays at the time.

Further, apart from expectedly different character names that appear together, following are some of the common bigrams across the corpora:

|  |  |  |
| --- | --- | --- |
| **Bigrams** | **PMI for Caesar** | **PMI for Hamlet** |
| ('thou', 'art') | 5.34946374498946) | 5.999458020565063) |
| ('wee', 'l') | 9.19880612142944 | 8.686091939181125 |
| ('mine', 'owne') | 7.782443224684029 | 6.063877388345535 |
| ('haue', 'seene') | 5.764668522956054 | 5.5788352285661595 |
| ('let', 'vs') | 5.371650252439011 | 4.0687206830021765 |
| ('haue', 'heard') | 5.331709115679947 | 5.803901784200933 |
| ('exeunt', 'enter') | 5.241874843321327 | 6.281518019341201 |

This constitutes ~14% of all known bigrams with frequency filter > 5 found by PMI. Goes to show that Shakespeare liked to use a lot of variation in his writing.

Next researching about the genre of the two plays (which were mostly tragic), I learnt that Hamlet had a comical under-tone to the writing style. Curious, I looked through the list of top-50 words (by count/frequency and words normalized with total word counts) but did not see any visible words or patterns. Perhaps, at a later point in the course we should be able to apply techniques like sentiment-analysis to bring out this aspect.