AI Lab 5: NLP (Text Processing)

Introduction:

In this lab, we will be taking a text file, 'shakespeare.txt' containing the collection of writings from William Shakespeare. We will be tokenizing the text file and calculate probability of occurrence of terms, conditional probabilities, probabilities of dependent or independent occurrences and to make some predictions.

Procedure:

In the program, initially we read the text file, the we lowercase it and then we split it by spaces. The unnecessary symbols like '!. @' were replaced by a blank. Then we create a list of all the words.
We then create a dictionary whose key is the term in the list and value of the frequency of that term in the list.
We then sort this dictionary and perform operations to print tuples with their frequencies.
Furthermore we create a bigram, similar to a dictionary with key as the term and the subsequent adjacent term in list and their frequency as their repetitions. Similar operations were performed on the bigram dictionary as well. (Since in a dictionary, the keys are indexes, it will be fast to search for values, so dictionary data structure was implemented to store the tokenized text.)
We now calculate the probability of certain words by getting their frequency from the dictionary and dividing them by the total number of words in out list.
We also calculate independent and dependent probabilities of happening or certain words in a sequence.
Finally, to predict what word will follow current sentence, chain conditional probability was calculated using Markov's assumption.
(i.e $P(A,B,C,D) = P(A)*P(B|A)*P(C|B)*P(D|C)$). Here D is our prediction and A, B, C are our previous words. We take the word D as the word with the highest probability among all other possible candidates.
Doing so, the prediction was not accurate, so a new probability i.e probability of the new word appearing after the words in sentence was calculated as. (i.e. $P(A,B,C,D) = P(A) * P(D | B,C) * P(D| A, B, C)* P(D | C)$). For this, a python package "nltk" was implemented to create a fourgram and trigram to calculate $P(D | B, C)$ and $P(D|A, B, C)$.

Implementation:

This lab was implemented in Python 3.5.2.
The total number of words : 812803

Part A:

1. **A table containing 20 most frequent words. The table contains three columns: rank, word and frequency.**

   - The top 20 frequent words are:
   - [(1, 'the', 26851), (2, 'and', 24077), (3, 'i', 20535), (4, 'to', 18561), (5, 'of', 16013), (6, 'you', 13856), (7, 'a', 13840), (8, 'my', 12282), (9, 'that', 10761), (10, 'in', 10537), (11, 'is', 9152), (12, 'not', 8462), (13, 'me', 7758), (14, 'it', 7752), (15, 'for', 7584), (16, 'with', 7148), (17, 'be', 6852), (18, 'your', 6755), (19, 'this', 6608), (20, 'his', 6535)]

2. **A table, containing list of bottom frequencies. The table contains three columns: frequency, word count and example words. You are supposed to print word counts for frequencies 10 to 1. The rows in this table show how many words have frequency 10,9,8...1 with example of some of the words.**

   - The bottom words are:
   - [(10, 321, ['relate', 'antique', 'channel']), (9, 391, ['te', 'wretchedness', 'lily']), (8, 462, ['helenus', 'orsinos', 'exact']), (7, 571, ['via', 'advocate', 'clothe']), (6, 778, ['redeemd', 'unusual', 'outrageous']), (5, 978, ['gord', 'dreaded', 'engaged']), (4, 1374, ['lapwing', 'observed', 'discomfited']), (3, 2031, ['presageth', 'mountebank', 'snug']), (2, 3773, ['leven', 'overplus', 'ensconce']), (1, 11503, ['marl', 'purgative', 'darting'])]

3. **A table containing 20 most frequent word-pairs (bigrams). The table contains three columns: rank, word pair and frequency.**

- The top 20 frequent bigrams are:
- [(1, ('i', 'am'), 1858), (2, ('my', 'lord'), 1685), (3, ('i', 'have'), 1628), (4, ('in', 'the'), 1584), (5, ('i', 'will'), 1582), (6, ('to', 'the'), 1518), (7, ('of', 'the'), 1380), (8, ('it', 'is'), 1087), (9, ('to', 'be'), 968), (10, ('that', 'i'), 935), (11, ('i', 'do'), 829), (12, ('and', 'i'), 736), (13, ('and', 'the'), 728), (14, ('you', 'are'), 724), (15, ('of', 'my'), 696), (16, ('is', 'the'), 692), (17, ('i', 'would'), 674), (18, ('the', 'king'), 664), (19, ('he', 'is'), 658), (20, ('you', 'have'), 652)]

Part B:

With the frequency counts of the word at our hand we calculate some basic probability estimates.

1. Calculate the relative frequency (probability estimate) of the words:
(a) “the" (b) “become" (d) “brave" (e) “treason"
[Note: P(the) = count(the) / N . Here, count(the) is the frequency of “the" and “N" is the total word count.]

- Probability of word "the" is: 0.033035065077269644
- Probability of word "become" is: 0.00017716470042556436
- Probability of word "brave" is: 0.00019315873588065006
- Probability of word "treason" is: 0.00011318855860522168

2. Calculate the following word conditional probabilities:
(a) P(court | The) (b) P(word | his) (c) P(qualities | rare) (d) P(men | young)
[Read P(B | A) as “the probability with which word B follows word A". Note: P(B | A) = count(A;B)| count(A)]

- Probability of "court | the" is: 0.0001316432148995513
- Probability of "word | his" is: 2.2145587553195545e-05
- Probability of "qualities | rare" is: 1.230310419621975e-06
- Probability of "men | young" is: 1.1072793776597773e-05

3. Calculate the probability:
(a) P(have, sent) (b) P(will, look, upon) (c) P(I, am, no, baby) (d) P(wherefore, art, thou, Romeo) Hint à use the chain rule (multiplication rule):

- Probability of "have, sent" is: 2.702343854723852e-07
- Probability of "will, look, upon"is: 6.532667708756251e-12
- Probability of "I, am, no , baby " is: 4.021212806084287e-15
- Probability of "wherefore, art, thou, romeo" is: 2.8662635320974777e-19

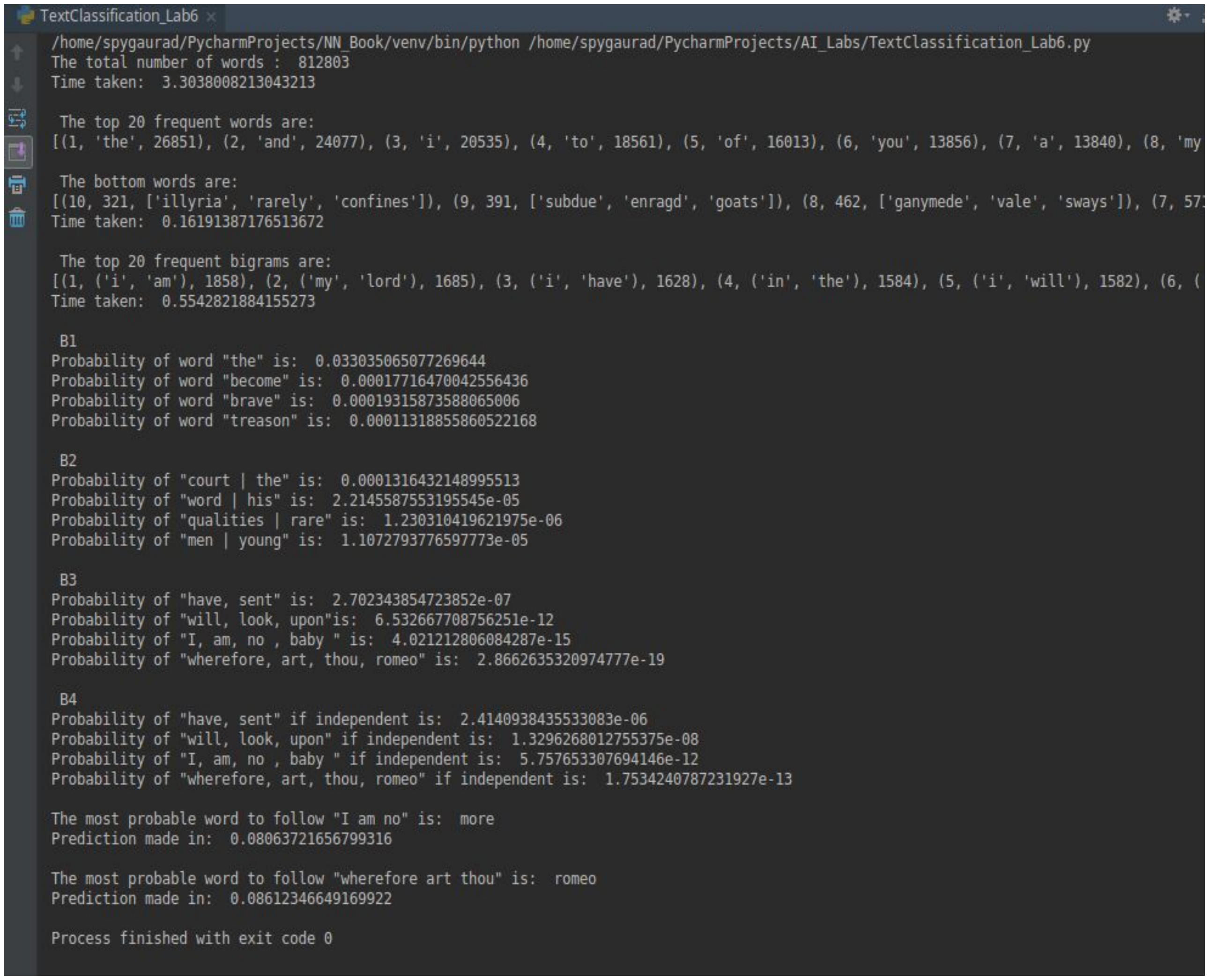
4. Calculate probabilities in Q3 assuming each word is independent of other words (independence assumption).

- Probability of "have, sent" if independent is: 2.4140938435533083e-06
- Probability of "will, look, upon" if independent is: 1.3296268012755375e-08
- Probability of "I, am, no , baby " if independent is: 5.757653307694146e-12
- Probability of "wherefore, art, thou, romeo" if independent is: 1.7534240787231927e-13

5. Find the most probable word to follow this sequence of words:
(a) I am no (b) wherefore art thou

The most probable word to follow "**I am no**" is: **more**
The most probable word to follow "**wherefore art thou**" is: **romeo**

Outputs



```
/home/spygaurad/PycharmProjects/NN_Book/venv/bin/python /home/spygaurad/PycharmProjects/AI_Labs/TextClassification_Lab6.py
The total number of words :  812803
Time taken:  3.3038008213043213

 The top 20 frequent words are:
[(1, 'the', 26851), (2, 'and', 24077), (3, 'i', 20535), (4, 'to', 18561), (5, 'of', 16013), (6, 'you', 13856), (7, 'a', 13840), (8, 'my

 The bottom words are:
[(10, 321, ['illyria', 'rarely', 'confines']), (9, 391, ['subdue', 'enragd', 'goats']), (8, 462, ['ganymede', 'vale', 'sways']), (7, 57
Time taken:  0.16191387176513672

 The top 20 frequent bigrams are:
[(1, ('i', 'am'), 1858), (2, ('my', 'lord'), 1685), (3, ('i', 'have'), 1628), (4, ('in', 'the'), 1584), (5, ('i', 'will'), 1582), (6, (
Time taken:  0.5542821884155273

 B1
Probability of word "the" is:  0.033035065077269644
Probability of word "become" is:  0.00017716470042556436
Probability of word "brave" is:  0.00019315873588065006
Probability of word "treason" is:  0.00011318855860522168

 B2
Probability of "court | the" is:  0.0001316432148995513
Probability of "word | his" is:  2.2145587553195545e-05
Probability of "qualities | rare" is:  1.230310419621975e-06
Probability of "men | young" is:  1.1072793776597773e-05

 B3
Probability of "have, sent" is:  2.702343854723852e-07
Probability of "will, look, upon"is:  6.532667708756251e-12
Probability of "I, am, no , baby " is:  4.021212806084287e-15
Probability of "wherefore, art, thou, romeo" is:  2.8662635320974777e-19

 B4
Probability of "have, sent" if independent is:  2.4140938435533083e-06
Probability of "will, look, upon" if independent is:  1.3296268012755375e-08
Probability of "I, am, no , baby " if independent is:  5.757653307694146e-12
Probability of "wherefore, art, thou, romeo" if independent is:  1.7534240787231927e-13

The most probable word to follow "I am no" is:  more
Prediction made in:  0.08063721656799316

The most probable word to follow "wherefore art thou" is:  romeo
Prediction made in:  0.08612346649169922

Process finished with exit code 0
```

Conclusion

Text Processing, is an important step in performing Natural Language Processing. Text Processing is done to clean a text file, to tokenize the words in text to extract information or knowledge that can be used to predict/estimate certain related values.

Source Code : https://github.com/spygaurad/NLP_Text_Processing_Python