Readme file

Importing Libraries:

1. **First we import libraries such as nltk**

```
import nltk
nltk.download('stopwords')
nltk.download('punkt')
```

2. **Then I import some more libraries like panda and json etc.**

```
import re
import os
import pathlib
import json
import string
import pandas as pd
import numpy as np
from nltk.tokenize import word_tokenize
from nltk.tokenize import RegexpTokenizer
from nltk.stem import WordNetLemmatizer
from nltk.stem import PorterStemmer
from nltk.corpus import stopwords
```

Loading files:

Then I load training set :

```
path = "/content/drive/MyDrive/nlp_data_ass_2/train (2).txt"
data = pd.read_table(path)
data
```

Preprocessing steps:

Assumption: I put the “$” symbol in starting of each line and the “#” symbol at the end of each line.

Then I lower the whole data set line by using s.lower()

Methodology + Assumptions made for each part :

1) **Finding Unigrams :**
   a) **I use `make_unigram(sentence)` to make the unigram of all the vocabulary:**
      i) **Take dictionary `unigram_count = {}`**
      ii) **Apply and filling elements of dictionary :**

```
unigram_count = {}
bigram_count={}
for i in range(line):
  sentence  = dt.loc[i][0]
  sentence = "$ "+sentence+" #"
  # sentence  = dt[i]
  sentence = lower(sentence)
  bigra = make_bigram(sentence)
  # print(bigra)
  unigra = make_unigram(sentence)
  # print(unigra)

  for j in unigra:
    if j in unigram_count:
      unigram_count[j]=[unigram_count[j][0]+1,0]
    else:
      unigram_count[j]=[1,0]

  for j in bigra:
    if j in bigram_count:
      bigram_count[j]=[bigram_count[j][0]+1,0]
    else:
      bigram_count[j]=[1,0]
```

      iii) **I calculate all unigrams with the help of a dictionary and also stores the pair count and probability as the value:**

```
V = len(unigram_count)
print("V =",V)
###################### finding unigram counts &
probabilities
u_count = 0
for i in unigram_count:
```

```
    u_count += unigram_count[i][0]
for i in unigram_count:
    unigram_count[i][1] = unigram_count[i][0] / u_count

print("Unigrams with [count, probability]:\n")
unigram_count
```

2) **Finding Bigrams :**
   a) **I use** `make_bigram(sentence)` **to make the bigram of all the vocabulary:**
      i) **Take dictionary** `bigram_count = {}`
      ii) **Apply and filling elements of dictionary :**

```
unigram_count = {}
bigram_count={}
for i in range(line):
    sentence  = dt.loc[i][0]
    sentence = "$ "+sentence+" #"
    # sentence  = dt[i]
    sentence = lower(sentence)
    bigra = make_bigram(sentence)
    # print(bigra)
    unigra = make_unigram(sentence)
    # print(unigra)

    for j in unigra:
        if j in unigram_count:
            unigram_count[j]=[unigram_count[j][0]+1,0]
        else:
            unigram_count[j]=[1,0]

    for j in bigra:
        if j in bigram_count:
            bigram_count[j]=[bigram_count[j][0]+1,0]
        else:
            bigram_count[j]=[1,0]
```

      iii) **I calculate all bigrams with the help of a dictionary and also stores the pair count and probability as the value:**

```
# finding bigram probabilities
# Without smoothing
for i in bigram_count:
```

```
  bigram_count[i][1] = bigram_count[i][0] /
unigram_count[i[0]][0]    # computing probabilities

print("Co-occurence matrix ")
print("Bigrams with [count, probability]:\n")
bigram_count
```

iv) **This is our Co-occurrence matrix in the form of a dictionary:**

(1) Assumption: I assume that dictionary is our Co-occurrence matrix, the bigrams with probability zero can't appear in this.

v) **We get this dictionary with bigram- probabilities as well count:**

(1) As you can see e.g.:

```
(':', 'she'): [762, 0.011226353939536802],
 ('she', 'said'): [20527, 0.051443923441858365],
 ('said', 'that'): [6549, 0.025479020366876107],
 ('that', 'they'): [17516, 0.032579481771225674],
 ('they', 'had'): [21893, 0.08825722912694157],
 ('had', 'never'): [8246, 0.02251652012451532],
 ('never', 'existed'): [22, 0.00027915947619531014],
 ('existed', ';'): [15, 0.06521739130434782],
 (';', 'and'): [76378, 0.283985870979736],
 ('and', 'that'): [20209, 0.011796081709273194],
```

vi) **This is without smoothing case.**

3) **Json file : Test on without smoothing:**

a) **I load the validation set :**

```
path = "/content/drive/MyDrive/nlp_data_ass_2/validation (2).jsonl"
df = pd.read_json(path, lines = True)
df
```

b) **Now I go to each sentence and split it into tokens. And find the mask word "xxxxx" and store it's index in a variable y.**

```
l = "$ "+sent.lower()+" #"
  l = l.split()
  y=l.index('xxxxx')
```

c) **So that I can get all the option tuples. And then find the probabilities of all the tuples. Select the word with maximum probability.**

d) **I also put a counter that counts the predicted word with the answer.**

```
if Keymax[1] == ans:
    print("TRUE")
    counter += 1
  print()
```

e) **Finally finding the accuracy of the model:**
Accuracy = counter / len(data) x 100

```
acc = counter/len(df) * 100
print("Accuracy (%):", acc)
```

f) **Calculating Efficiency - Bonus- Without Smoothing (BONUS PART)**

i) **Suppose we have to find mask word `'would', 'xxxxx', 'you'`**
ii) **We have to compute the probability of `'would', 'xxxxx'` and `'xxxxx', 'you'` we can simply search it our bigram dictionary as I have already stored the probabilities of all bigrams.**
iii) **P = p_previous_tuple * p_next_tuple**
iv) **Then we can easily select the word with max P.**
v) **It increases our accuracy.**

4) **Add-1 smoothing:**
a) **We have to repeat the same steps that we have repeated in 2) and 3) but we should the formula :**

$$P_{\text{Laplace}}(w_i) = \frac{c_i + 1}{N + V}$$

i)

```
# Add 1 search
def add1_prob(a,b):
  for i in bigram_count:
    k = (a,b)
    # print(k)
    if k in bigram_count:
      return (bigram_count[k][1])
      break
    else:
      # print("case-2")
      den = unigram_count[search_bigram[0]][0] + V
      num = 0 + 1
      return (num/den)
      break

search_bigram = input().split()
add1_prob(search_bigram[0],search_bigram[1])
```

5) **Add-k smoothing:**
   a) **We have to repeat the same steps that we have repeated in 2) and 3) but we should the formula :**

$$P_{LAP,k}(x) = \frac{c(x) + k}{N + k|X|}$$

   i)

```
# Add K search
def addk_prob(a,b):
  for i in bigram_count:
    q = (a,b)
    # print(k)
    if q in bigram_count:
      return (bigram_count[q][1])
      break
    else:
      # print("case-2")
      den = unigram_count[search_bigram[0]][0] + (k*V)
      num = 0 + k
      return (num/den)
      break

search_bigram = input().split()
# print(search_bigram)
addk_prob(search_bigram[0],search_bigram[1])
```

6) **Outputs:**

a) Finding Unigrams

```
V = 53170
Unigrams with [count, probability]:

{'$': [2206532, 0.04073321079026581],
 'however': [11140, 0.0002056475810020254],
 ',': [3213950, 0.05933043473621719],
 'she': [399017, 0.0073659677584098],
 'too': [44055, 0.0008132678798064837],
 'in': [551857, 0.010187437801529151],
 'spite': [6190, 0.00011426916754062273],
 'of': [824578, 0.015221945336399291],
 'all': [231121, 0.004266559656083403],
……...
```

b) Finding Bigrams

```
Co-occurence matrix
Bigrams with [count, probability]:

{('$', 'however'): [3355, 0.0015204855402051727],
 ('however', ','): [9571, 0.8591561938958707],
 (',', 'she'): [32483, 0.010106877829462188],
 ('she', ','): [4891, 0.012257623108789851],
 (',', 'too'): [8087, 0.0025162183605843278],
 ('too', 'in'): [9, 0.00020429009193054137],
 ('in', 'spite'): [5785, 0.010482788113587396],
 ('spite', 'of'): [5878, 0.9495961227786753],
 ('of', 'all'): [11843, 0.01436249815056914],
 ('all', 'the'): [48303, 0.20899442283479217],
 ('the', 'books'): [699, 0.0003061606620779067],
 ('books', 'she'): [46, 0.009575353871773521],
...
```

c) Function :

```
# Without smoothing search
def ws_prob(a,b):
  for i in bigram_count:
    k = (a,b)
    # print(k)
    if k in bigram_count:
      # print("freq :",bigram_count[k][0])
      return (bigram_count[k][1])
```

```
            break
        else:
            # print("case-2")
            return 0
            break

search_bigram = input().split()
ws_prob(search_bigram[0],search_bigram[1])

thing .
0.08879799445837182
```

d) Json file - test on Without smoothing

```
['$', '``', 'oh', ',', 'mr.', 'dale', ',', 'can', 'you', 'xxxxx',
'us', 'anything', 'of', 'paddy', '?', "''", '#']
['find', 'happen', 'infected', 'lost', 'repeated', 'repeating', 'started',
'tell', 'turning', 'wish']
{('you', 'find'): 0.0016332506083473835, ('you', 'happen'):
0.00042205130121493624, ('you', 'infected'): 0, ('you', 'lost'):
0.00040666401419147503, ('you', 'repeated'): 0, ('you', 'repeating'): 0,
('you', 'started'): 0.00012749466390867865, ('you', 'tell'):
0.0037918671593529427, ('you', 'turning'): 0, ('you', 'wish'):
0.0030708628531107597}
tell
Answer : tell
TRUE

['$', 'she', 'let', 'him', 'lead', 'her', 'on', ',', 'and', 'presently',
'reached', 'a', 'little', 'hill', ',', 'from', 'which', 'she', 'xxxxx',
'a', 'valley', 'full', 'of', 'lovely', 'fruit', 'trees', ',', 'bearing',
'flowers', 'and', 'fruit', 'together', '.', '#']
['consented', 'entrusted', 'excepting', 'pretended', 'put', 'retired',
'saw', 'sent', 'swallowed', 'undeceived']
{('she', 'consented'): 0.0005588734314578075, ('she', 'entrusted'):
3.508622439645429e-05, ('she', 'excepting'): 0, ('she', 'pretended'):
0.00048619482377943796, ('she', 'put'): 0.004488530563860687, ('she',
'retired'): 0.00010024635541844081, ('she', 'saw'): 0.015365260126761517,
('she', 'sent'): 0.0015036953312766123, ('she', 'swallowed'):
6.516013102198653e-05, ('she', 'undeceived'): 0}
saw
Answer : saw
TRUE
```

```
['$', 'through', 'the', 'dim', 'rosiness', 'of', 'the', 'cheeks', ',',
'i', 'could', 'xxxxx', 'the', 'brown', 'leaves', ',', 'the', 'slimy',
'twigs', ',', 'the', 'acorns', ',', 'and', 'the', 'sparkling', 'sand',
'.', '#']
['assume', 'clear', 'filled', 'find', 'found', 'gush', 'rambled', 'see',
'set', 'sparkled']
{('could', 'assume'): 2.395190457561217e-05, ('could', 'clear'): 0,
('could', 'filled'): 0, ('could', 'find'): 0.01687012478942284, ('could',
'found'): 0, ('could', 'gush'): 0, ('could', 'rambled'): 0, ('could',
'see'): 0.05141675515564746, ('could', 'set'): 0.0005429098370472092,
('could', 'sparkled'): 0}
see
Answer : see
TRUE

['$', 'they', 'xxxxx', 'mrs.', 'griggs', 'must', 'be', 'drawing',
'considerably', 'upon', 'her', 'imagination', ';', 'there', 'were', 'not',
'lacking', 'those', 'who', 'declared', 'that', 'she', 'had', 'invented',
'the', 'whole', 'account', ',', 'since', 'her', 'reputation', 'for',
'strict', 'veracity', 'was', 'not', 'wholly', 'unquestioned', '.', '#']
['concerning', 'getting', 'questioned', 'read', 'reserved', 'said',
'thought', 'tried', 'went', 'wondered']
{('they', 'concerning'): 0, ('they', 'getting'): 4.031299005478535e-05,
('they', 'questioned'): 8.06259801095707e-05, ('they', 'read'):
0.0005361627677286452, ('they', 'reserved'): 0, ('they', 'said'):
0.01015484219480043, ('they', 'thought'): 0.00622029436545338, ('they',
'tried'): 0.0018100532534598623, ('they', 'went'): 0.026292132113731007,
('they', 'wondered'): 0.000511974973695774}
went
Answer : thought

Total : 1055
Accuracy (%): 52.75
```

e) Calculating Efficiency - Bonus- Without Smoothing

```
['$', 'she', 'let', 'him', 'lead', 'her', 'on', ',', 'and',
'presently', 'reached', 'a', 'little', 'hill', ',', 'from',
'which', 'she', 'xxxxx', 'a', 'valley', 'full', 'of',
'lovely', 'fruit', 'trees', ',', 'bearing', 'flowers', 'and',
'fruit', 'together', '.', '#']
['consented', 'entrusted', 'excepting', 'pretended', 'put',
'retired', 'saw', 'sent', 'swallowed', 'undeceived']
{('she', 'consented'): 0.0005588734314578075, ('she',
'entrusted'): 3.508622439645429e-05, ('she', 'excepting'): 0,
```

```
('she', 'pretended'): 0.00048619482377943796, ('she', 'put'):
0.004488530563860687, ('she', 'retired'):
0.00010024635541844081, ('she', 'saw'): 0.015365260126761517,
('she', 'sent'): 0.0015036953312766123, ('she', 'swallowed'):
6.516013102198653e-05, ('she', 'undeceived'): 0}
{('consented', 'a'): 0, ('entrusted', 'a'): 0, ('excepting',
'a'): 0.07692307692307693, ('pretended', 'a'): 0, ('put',
'a'): 0.042978890782745506, ('retired', 'a'):
0.0065116279069767444, ('saw', 'a'): 0.10687915333497415,
('sent', 'a'): 0.06321131959397108, ('swallowed', 'a'):
0.11225444340505145, ('undeceived', 'a'): 0}
{'consented': 0.0, 'entrusted': 0.0, 'excepting': 0.0,
'pretended': 0.0, 'put': 0.0001929120648791836, 'retired':
6.527669655154286e-07, 'saw': 0.0016422259931199086, 'sent':
9.505056615728816e-05, 'swallowed': 7.314514240073324e-06,
'undeceived': 0}
saw
Answer : saw
TRUE

['$', 'through', 'the', 'dim', 'rosiness', 'of', 'the',
'cheeks', ',', 'i', 'could', 'xxxxx', 'the', 'brown',
'leaves', ',', 'the', 'slimy', 'twigs', ',', 'the', 'acorns',
',', 'and', 'the', 'sparkling', 'sand', '.', '#']
['assume', 'clear', 'filled', 'find', 'found', 'gush',
'rambled', 'see', 'set', 'sparkled']
{('could', 'assume'): 2.395190457561217e-05, ('could',
'clear'): 0, ('could', 'filled'): 0, ('could', 'find'):
0.01687012478942284, ('could', 'found'): 0, ('could', 'gush'):
0, ('could', 'rambled'): 0, ('could', 'see'):
0.05141675515564746, ('could', 'set'): 0.0005429098370472092,
('could', 'sparkled'): 0}
{('assume', 'the'): 0.20535714285714285, ('clear', 'the'):
0.024174724908302768, ('filled', 'the'): 0.10973402033143016,
('find', 'the'): 0.09590876413306319, ('found', 'the'):
0.1001332671352792, ('gush', 'the'): 0, ('rambled', 'the'): 0,
('see', 'the'): 0.08939125134690945, ('set', 'the'):
0.05212666291924302, ('sparkled', 'the'):
0.009575923392612859}
{'assume': 4.918694689634642e-06, 'clear': 0.0, 'filled': 0.0,
'find': 0.0016179928193240975, 'found': 0.0, 'gush': 0,
'rambled': 0, 'see': 0.004596208083560985, 'set':
2.8300078071301028e-05, 'sparkled': 0.0}
see
Answer : see
TRUE
```

```
['$', 'they', 'xxxxx', 'mrs.', 'griggs', 'must', 'be',
'drawing', 'considerably', 'upon', 'her', 'imagination', ';',
'there', 'were', 'not', 'lacking', 'those', 'who', 'declared',
'that', 'she', 'had', 'invented', 'the', 'whole', 'account',
',', 'since', 'her', 'reputation', 'for', 'strict',
'veracity', 'was', 'not', 'wholly', 'unquestioned', '.', '#']
['concerning', 'getting', 'questioned', 'read', 'reserved',
'said', 'thought', 'tried', 'went', 'wondered']
{('they', 'concerning'): 0, ('they', 'getting'):
4.031299005478535e-05, ('they', 'questioned'):
8.06259801095707e-05, ('they', 'read'): 0.0005361627677286452,
('they', 'reserved'): 0, ('they', 'said'):
0.01015484219480043, ('they', 'thought'): 0.00622029436545338,
('they', 'tried'): 0.0018100532534598623, ('they', 'went'):
0.026292132113731007, ('they', 'wondered'):
0.000511974973695774}
{('concerning', 'mrs.'): 0.011210762331838564, ('getting',
'mrs.'): 0, ('questioned', 'mrs.'): 0, ('read', 'mrs.'): 0,
('reserved', 'mrs.'): 0, ('said', 'mrs.'):
0.011772715778006886, ('thought', 'mrs.'):
0.0010545556805399326, ('tried', 'mrs.'): 0, ('went', 'mrs.'):
7.361601884570082e-05, ('wondered', 'mrs.'): 0}
{'concerning': 0.0, 'getting': 0.0, 'questioned': 0.0, 'read':
0.0, 'reserved': 0, 'said': 0.0001195500709298971, 'thought':
6.559646757719397e-06, 'tried': 0.0, 'went':
1.9355220931780776e-06, 'wondered': 0.0}
said
Answer : thought

Total : 1431
Accuracy (%): 71.55
```

f) Add-1 Laplace smoothing

[38]

1s

```
# With Add-1 Laplace smoothing

# Co-occurence matrix

# p = 0

for i in bigram_count:

  bigram_count[i][1] = (bigram_count[i][0] + 1) / (unigram_count[i[0]][0]
+ V)    # computing probabilities Add-1

  # if i[0] == 'w':
```

```
  #    p += bigram_count[i][1]

print("Bigram probability with L-smoothing function [count,
probability]:\n")

bigram_count

# p
```

Bigram probability with L-smoothing function [count, probability]:

{('$', 'however'): [3355, 0.001485151581934255],
('however', ','): [9571, 0.14884154874825067],
(',', 'she'): [32483, 0.009942701829133916],
('she', ','): [4891, 0.010818533040534116],
(',', 'too'): [8087, 0.0024755748181884965],
('too', 'in'): [9, 0.00010285420416559527],

g) Json file - test on With add-1

```
['$', 'through', 'the', 'dim', 'rosiness', 'of', 'the',
'cheeks', ',', 'i', 'could', 'xxxxx', 'the', 'brown', 'leaves', ',',
'the', 'slimy', 'twigs', ',', 'the', 'acorns', ',', 'and', 'the',
'sparkling', 'sand', '.', '#']
['assume', 'clear', 'filled', 'find', 'found', 'gush', 'rambled',
'see', 'set', 'sparkled']
{('could', 'assume'): 2.241888566928781e-05, ('could', 'clear'):
5.66134085461337e-07, ('could', 'filled'): 5.66134085461337e-07,
('could', 'find'): 0.011848381076218606, ('could', 'found'):
5.66134085461337e-07, ('could', 'gush'): 5.66134085461337e-07,
('could', 'rambled'): 5.66134085461337e-07, ('could', 'see'):
0.03610001064897069, ('could', 'set'): 0.00038672577779521466,
('could', 'sparkled'): 5.66134085461337e-07}
see
Answer : see
TRUE

['$', 'they', 'xxxxx', 'mrs.', 'griggs', 'must', 'be', 'drawing',
'considerably', 'upon', 'her', 'imagination', ';', 'there', 'were',
'not', 'lacking', 'those', 'who', 'declared', 'that', 'she', 'had',
'invented', 'the', 'whole', 'account', ',', 'since', 'her',
'reputation', 'for', 'strict', 'veracity', 'was', 'not', 'wholly',
'unquestioned', '.', '#']
['concerning', 'getting', 'questioned', 'read', 'reserved', 'said',
'thought', 'tried', 'went', 'wondered']
{('they', 'concerning'): 5.66134085461337e-07, ('they', 'getting'):
3.6517068409748064e-05, ('they', 'questioned'):
```

```
6.971440332770086e-05, ('they', 'read'): 0.00044484428790056733,
('they', 'reserved'): 5.66134085461337e-07, ('they', 'said'):
0.008365728399324103, ('they', 'thought'): 0.00512566851133191,
('they', 'tried'): 0.0014938800713078754, ('they', 'went'):
0.021654621566980604, ('they', 'wondered'): 0.00042492588694979566}
went
Answer : thought

Total : 1055
Accuracy (%): 52.75
```

h) Calculating Effieciency - Bonus- With Add-1

```
['$', 'through', 'the', 'dim', 'rosiness', 'of', 'the',
'cheeks', ',', 'i', 'could', 'xxxxx', 'the', 'brown',
'leaves', ',', 'the', 'slimy', 'twigs', ',', 'the', 'acorns',
',', 'and', 'the', 'sparkling', 'sand', '.', '#']
['assume', 'clear', 'filled', 'find', 'found', 'gush',
'rambled', 'see', 'set', 'sparkled']
{('could', 'assume'): 2.241888566928781e-05, ('could',
'clear'): 5.66134085461337e-07, ('could', 'filled'):
5.66134085461337e-07, ('could', 'find'): 0.011848381076218606,
('could', 'found'): 5.66134085461337e-07, ('could', 'gush'):
5.66134085461337e-07, ('could', 'rambled'):
5.66134085461337e-07, ('could', 'see'): 0.03610001064897069,
('could', 'set'): 0.00038672577779521466, ('could',
'sparkled'): 5.66134085461337e-07}
{('assume', 'the'): 0.00045043354228444877, ('clear', 'the'):
0.0024675500270416443, ('filled', 'the'): 0.01307351990853507,
('find', 'the'): 0.0350475099078451, ('found', 'the'):
0.04191865615328252, ('gush', 'the'): 5.66134085461337e-07,
('rambled', 'the'): 5.66134085461337e-07, ('see', 'the'):
0.05662991995697946, ('set', 'the'): 0.017431594765512492,
('sparkled', 'the'): 0.00014842025194337766}
{'assume': 1.0098218086087372e-08, 'clear':
1.3969641778893187e-09, 'filled': 7.401365237179084e-09,
'find': 0.000415256253160696, 'found': 2.3731580065106842e-08,
'gush': 3.2050780272114437e-13, 'rambled':
3.2050780272114437e-13, 'see': 0.0020443407134973163, 'set':
6.741247043903811e-06, 'sparkled': 8.402576359790534e-11}
see
Answer : see
TRUE

['$', 'they', 'xxxxx', 'mrs.', 'griggs', 'must', 'be',
'drawing', 'considerably', 'upon', 'her', 'imagination', ';',
'there', 'were', 'not', 'lacking', 'those', 'who', 'declared',
```

```
'that', 'she', 'had', 'invented', 'the', 'whole', 'account',
',', 'since', 'her', 'reputation', 'for', 'strict',
'veracity', 'was', 'not', 'wholly', 'unquestioned', '.', '#']
['concerning', 'getting', 'questioned', 'read', 'reserved',
'said', 'thought', 'tried', 'went', 'wondered']
{('they', 'concerning'): 5.66134085461337e-07, ('they',
'getting'): 3.6517068409748064e-05, ('they', 'questioned'):
6.971440332770086e-05, ('they', 'read'):
0.00044484428790056733, ('they', 'reserved'):
5.66134085461337e-07, ('they', 'said'): 0.008365728399324103,
('they', 'thought'): 0.00512566851133191, ('they', 'tried'):
0.0014938800713078754, ('they', 'went'): 0.021654621566980604,
('they', 'wondered'): 0.00042492588694979566}
{('concerning', 'mrs.'): 0.00011190689346463742, ('getting',
'mrs.'): 5.66134085461337e-07, ('questioned', 'mrs.'):
5.66134085461337e-07, ('read', 'mrs.'): 5.66134085461337e-07,
('reserved', 'mrs.'): 5.66134085461337e-07, ('said', 'mrs.'):
0.009758063216260215, ('thought', 'mrs.'):
0.0005542129267893809, ('tried', 'mrs.'):
5.66134085461337e-07, ('went', 'mrs.'): 5.197736756909277e-05,
('wondered', 'mrs.'): 5.66134085461337e-07}
{'concerning': 6.335430678842177e-11, 'getting':
2.06735571278818e-11, 'questioned': 3.946769997141071e-11,
'read': 2.5184151410328737e-10, 'reserved':
3.2050780272114437e-13, 'said': 8.163330657066798e-05,
'thought': 2.840711747417427e-06, 'tried':
8.457364279588009e-10, 'went': 1.1255502247565545e-06,
'wondered': 2.4056502839717e-10}
said
Answer : thought

Total : 1352
Accuracy (%): 67.60000000000001
```

i) Add-k smoothing

```
Bigram probability with L-smoothing function [count,
probability]:

{('$', 'however'): [3355, 0.0002768576876878875],
 ('however', ','): [9571, 0.0009178836539491261],
 (',', 'she'): [32483, 0.0023601327272267736],
 ('she', ','): [4891, 0.00046143316918663315],
 (',', 'too'): [8087, 0.0005984279261551349],
 ('too', 'in'): [9, 1.95728529212483e-05],
 ('in', 'spite'): [5785, 0.0005350506447561416],
```

j) Json file - test on With add-K

```
['$', 'she', 'let', 'him', 'lead', 'her', 'on', ',', 'and',
'presently', 'reached', 'a', 'little', 'hill', ',', 'from', 'which',
'she', 'xxxxx', 'a', 'valley', 'full', 'of', 'lovely', 'fruit',
'trees', ',', 'bearing', 'flowers', 'and', 'fruit', 'together', '.',
'#']

['consented', 'entrusted', 'excepting', 'pretended', 'put',
'retired', 'saw', 'sent', 'swallowed', 'undeceived']

{('she', 'consented'): 3.8339467799242946e-05, ('she', 'entrusted'):
1.939632649890778e-05, ('she', 'excepting'): 1.8127407942904467e-05,
('she', 'pretended'): 3.5710993647521795e-05, ('she', 'put'):
0.00018045834607161395, ('she', 'retired'): 2.1752889531485357e-05,
('she', 'saw'): 0.0005738230984326408, ('she', 'sent'):
7.250963177161787e-05, ('she', 'swallowed'): 2.0483970975482045e-05,
('she', 'undeceived'): 1.8127407942904467e-05}

saw

Answer : saw

TRUE


['$', 'through', 'the', 'dim', 'rosiness', 'of', 'the', 'cheeks',
',', 'i', 'could', 'xxxxx', 'the', 'brown', 'leaves', ',', 'the',
'slimy', 'twigs', ',', 'the', 'acorns', ',', 'and', 'the',
'sparkling', 'sand', '.', '#']

['assume', 'clear', 'filled', 'find', 'found', 'gush', 'rambled',
'see', 'set', 'sparkled']

{('could', 'assume'): 1.8867484362991438e-05, ('could', 'clear'):
1.8127407942904467e-05, ('could', 'filled'): 1.8127407942904467e-05,
('could', 'find'): 0.00021497778981083347, ('could', 'found'):
1.8127407942904467e-05, ('could', 'gush'): 1.8127407942904467e-05,
('could', 'rambled'): 1.8127407942904467e-05, ('could', 'see'):
0.0006171433308879958, ('could', 'set'): 2.4908797090057664e-05,
('could', 'sparkled'): 1.8127407942904467e-05}

see

Answer : see

TRUE
```

['$', 'they', 'xxxxx', 'mrs.', 'griggs', 'must', 'be', 'drawing', 'considerably', 'upon', 'her', 'imagination', ';', 'there', 'were', 'not', 'lacking', 'those', 'who', 'declared', 'that', 'she', 'had', 'invented', 'the', 'whole', 'account', ',', 'since', 'her', 'reputation', 'for', 'strict', 'veracity', 'was', 'not', 'wholly', 'unquestioned', '.', '#']

['concerning', 'getting', 'questioned', 'read', 'reserved', 'said', 'thought', 'tried', 'went', 'wondered']

{('they', 'concerning'): 1.8127407942904467e-05, ('they', 'getting'): 1.9297818547023133e-05, ('they', 'questioned'): 2.0216762287357566e-05, ('they', 'read'): 3.0600826553136683e-05, ('they', 'reserved'): 1.8127407942904467e-05, ('they', 'said'): 0.00024986080299693286, ('they', 'thought'): 0.000160171893940292, ('they', 'tried'): 5.9639448747704824e-05, ('they', 'went'): 0.0006177139822528071, ('they', 'wondered'): 3.004946030893602e-05}

went

Answer : thought

Total : 1055

Accuracy (%): 52.75

k) Calculating Effieciency - Bonus - With Add-k

['$', 'through', 'the', 'dim', 'rosiness', 'of', 'the', 'cheeks', ',', 'i', 'could', 'xxxxx', 'the', 'brown', 'leaves', ',', 'the', 'slimy', 'twigs', ',', 'the', 'acorns', ',', 'and', 'the', 'sparkling', 'sand', '.', '#']

['assume', 'clear', 'filled', 'find', 'found', 'gush', 'rambled', 'see', 'set', 'sparkled']

{('could', 'assume'): 1.8867484362991438e-05, ('could', 'clear'): 1.6198009653365834e-05, ('could', 'filled'): 1.6198009653365834e-05, ('could', 'find'): 0.00021497778981083347, ('could', 'found'): 1.6198009653365834e-05, ('could', 'gush'): 1.6198009653365834e-05, ('could', 'rambled'): 1.6198009653365834e-05, ('could', 'see'): 0.0006171433308879958, ('could', 'set'): 2.4908797090057664e-05, ('could', 'sparkled'): 1.6198009653365834e-05}

{('assume', 'the'): 2.0970251206682796e-05, ('clear', 'the'): 3.242481812496581e-05, ('filled', 'the'): 9.284683720726112e-05, ('find', 'the'): 0.00029396315024226877, ('found', 'the'): 0.0003778015715308525, ('gush', 'the'): 1.6198009653365834e-05, ('rambled', 'the'): 1.6198009653365834e-05, ('see', 'the'): 0.0007843647656970148, ('set',

'the'): 0.00014923994096064183, ('sparkled', 'the'): 1.9464526183125834e-05}

{'assume': 3.9565588673009e-10, 'clear': 5.252175169968277e-10, 'filled': 1.5039339653677016e-09, 'find': 6.319554832491291e-08, 'found': 6.119633502713532e-09, 'gush': 2.623755167305328e-10, 'rambled': 2.623755167305328e-10, 'see': 4.840654841334381e-07, 'set': 3.7173874071208126e-09, 'sparkled': 3.1528658301246427e-10}

see

Answer : see

TRUE

['$', 'they', 'xxxxx', 'mrs.', 'griggs', 'must', 'be', 'drawing', 'considerably', 'upon', 'her', 'imagination', ';', 'there', 'were', 'not', 'lacking', 'those', 'who', 'declared', 'that', 'she', 'had', 'invented', 'the', 'whole', 'account', ',', 'since', 'her', 'reputation', 'for', 'strict', 'veracity', 'was', 'not', 'wholly', 'unquestioned', '.', '#']

['concerning', 'getting', 'questioned', 'read', 'reserved', 'said', 'thought', 'tried', 'went', 'wondered']

{('they', 'concerning'): 1.6198009653365834e-05, ('they', 'getting'): 1.9297818547023133e-05, ('they', 'questioned'): 2.0216762287357566e-05, ('they', 'read'): 3.0600826553136683e-05, ('they', 'reserved'): 1.6198009653365834e-05, ('they', 'said'): 0.00024986080299693286, ('they', 'thought'): 0.000160171893940292, ('they', 'tried'): 5.9639448747704824e-05, ('they', 'went'): 0.0006177139822528071, ('they', 'wondered'): 3.004946030893602e-05}

{('concerning', 'mrs.'): 1.927697973171334e-05, ('getting', 'mrs.'): 1.6198009653365834e-05, ('questioned', 'mrs.'): 1.6198009653365834e-05, ('read', 'mrs.'): 1.6198009653365834e-05, ('reserved', 'mrs.'): 1.6198009653365834e-05, ('said', 'mrs.'): 0.0002962069261553195, ('thought', 'mrs.'): 2.4319757670451568e-05, ('tried', 'mrs.'): 1.6198009653365834e-05, ('went', 'mrs.'): 1.922448071504616e-05, ('wondered', 'mrs.'): 1.6198009653365834e-05}

{'concerning': 3.122487037820302e-10, 'getting': 3.1258625111358295e-10, 'questioned': 3.274713106904202e-10, 'read': 4.956724839086815e-10, 'reserved': 2.623755167305328e-10, 'said': 7.401050042242133e-08, 'thought': 3.895341646245172e-09, 'tried': 9.660403665367396e-10, 'went': 1.1875230539233456e-08, 'wondered': 4.867414481625791e-10}

said

Answer : thought

Total : 1170

Accuracy (%): 58.5