Lab1. Understanding Large Text Files

EXERCISE-1

In [1]:
```
import nltk
nltk.download('wordnet')
nltk.download('punkt')
text = "This is Andrew's text, isn't it?"
```

```
[nltk_data] Downloading package wordnet to
[nltk_data]     C:\Users\1mscdsa18\AppData\Roaming\nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to
[nltk_data]     C:\Users\1mscdsa18\AppData\Roaming\nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

1. How many tokens are there if you use WhitespaceTokenizer?. Print tokens.

In [2]:
```
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tokens = tokenizer.tokenize(text)
print(len(tokens))
print(tokens)
```

```
6
['This', 'is', "Andrew's", 'text,', "isn't", 'it?']
```

2. How many tokens are there if you use TreebankWordTokenizer?. Print tokens.

In [3]:
```python
tokenizer = nltk.tokenize.TreebankWordTokenizer()
tokens = tokenizer.tokenize(text)
print(tokens)
```

```
['This', 'is', 'Andrew', "'s", 'text', ',', 'is', "n't", 'it', '?']
```

3. How many tokens there are if you use WordPunctTokenizer?. Print tokens.

In [4]:
```python
tokenizer = nltk.tokenize.WordPunctTokenizer()
tokens = tokenizer.tokenize(text)
print(tokens)
```

```
['This', 'is', 'Andrew', "'", 's', 'text', ',', 'isn', "'", 't', 'it', '?']
```

EXERCISE-2

1. Open the file: O. Henry's The Gift of the Magi (gift-of-magi.txt).

In [5]:
```python
f=open("The Gift of the Magi.txt","r",encoding='utf-8')
contents = f.read()
print(contents)
```

```
Gift of the Magi by O’Henry


One dollar and eighty-seven cents. That was all. And sixty cents of it was in pennies. Pennies saved one and two at
a time by bulldozing the grocer and the vegetable man and the butcher until one's cheeks burned with the silent impu
tation of parsimony that such close dealing implied. Three times Della counted it. One dollar and eighty- seven cent
s. And the next day would be Christmas.


There was clearly nothing to do but flop down on the shabby little couch and howl. So Della did it. Which instigates
the moral reflection that life is made up of sobs, sniffles, and smiles, with sniffles predominating.


While the mistress of the home is gradually subsiding from the first stage to the second, take a look at the home. A
furnished flat at $8 per week. It did not exactly beggar description, but it certainly had that word on the lookout
for the mendicancy squad.


In the vestibule below was a letter-box into which no letter would go, and an electric button from which no mortal f
```

2. Write a Python script to print out the following:

1. How many word tokens there are

In [6]:
```python
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tokens = tokenizer.tokenize(contents)
print(len(tokens))
```

```
2063
```

2. How many word types there are, (word types are a unique set of words)

In [7]:
```python
from nltk import *
data=FreqDist(tokens)
data
```

Out[7]:
```
FreqDist({'"Cut': 1,
          '"Dell,"': 1,
          '"Dillingham"': 1,
          '"Don\'t': 2,
          '"Give': 1,
          '"I': 1,
          '"If': 1,
          '"Isn\'t': 1,
          '"It\'s': 1,
          '"Jim"': 1,
          '"Jim,': 1,
          '"Mne.': 1,
          '"Mr.': 1,
          '"My': 1,
          '"Oh,': 1,
          '"Please': 1,
          '"Sofronie."': 1,
          '"Take': 1,
          '"Twenty': 1,
```

3. Top 20 most frequent words and their counts

```
In [8]: data.most_common(20 )

Out[8]: [('the', 106),
         ('and', 74),
         ('a', 63),
         ('of', 47),
         ('to', 41),
         ('was', 26),
         ('she', 25),
         ('in', 24),
         ('her', 23),
         ('had', 21),
         ('that', 20),
         ('at', 19),
         ('with', 19),
         ('for', 19),
         ('it', 18),
         ('his', 17),
         ('on', 16),
         ('I', 14),
         ('Jim', 13),
         ('were', 11)]
```

4. Words that are at least 10 characters long and their counts

```
In [9]: test=[w for w in tokens if len(w)>10]
        print(test)
        freq=FreqDist(test)
        freq
```

```
['eighty-seven', 'predominating.', 'description,', 'appertaining', '"Dillingham"', 'contracting', 'calculated.', 'sterl
ing--something', 'longitudinal', 'brilliantly,', 'possessions', "grandfather's.", '"Sofronie."', 'proclaiming', 'meretr
icious', 'ornamentation--as', 'description', 'intoxication', 'close-lying', 'wonderfully', 'critically.', 'twenty-two--
and', 'disapproval,', "Christmas!'", 'laboriously,', 'inconsequential', 'difference?', 'mathematician', 'illuminated',
'necessitating', 'possession.', 'men--wonderfully', 'duplication.']
```

```
Out[9]: FreqDist({'"Dillingham"': 1,
                  '"Sofronie."': 1,
                  "Christmas!'": 1,
                  'appertaining': 1,
                  'brilliantly,': 1,
                  'calculated.': 1,
                  'close-lying': 1,
                  'contracting': 1,
                  'critically.': 1,
                  'description': 1,
                  'description,': 1,
                  'difference?': 1,
                  'disapproval,': 1,
                  'duplication.': 1,
                  'eighty-seven': 1,
                  "grandfather's.": 1,
                  'illuminated': 1,
                  'inconsequential': 1,
                  'intoxication': 1,
                  'laboriously,': 1,
                  'longitudinal': 1,
                  'mathematician': 1,
                  'men--wonderfully': 1,
                  'meretricious': 1,
                  'necessitating': 1,
                  'ornamentation--as': 1,
                  'possession.': 1,
                  'possessions': 1,
                  'predominating.': 1,
                  'proclaiming': 1,
                  'sterling--something': 1,
                  'twenty-two--and': 1,
                  'wonderfully': 1})
```

EXERCISE -3: List Comprehension

STEP-1

Download the document Austen's Emma (“austen-emma.txt"). Read it in and apply the usual text processing steps, building three objects: etoks (a list of word tokens, all in lowercase), etypes (an alphabetically sorted word type list), and efreq (word frequency distribution).

In [10]:

```
with open("austen-emma.txt") as f:
    text=f.read()
text
```

Out[10]:

```
'[Emma by Jane Austen 1816]\n\nVOLUME I\n\nCHAPTER I\n\n\nEmma Woodhouse, handsome, clever, and rich, with a comfort
able home\nand happy disposition, seemed to unite some of the best blessings\nof existence; and had lived nearly twe
nty-one years in the world\nwith very little to distress or vex her.\n\nShe was the youngest of the two daughters of
a most affectionate,\nindulgent father; and had, in consequence of her sister\'s marriage,\nbeen mistress of his hou
se from a very early period.  Her mother\nhad died too long ago for her to have more than an indistinct\nremembrance
of her caresses; and her place had been supplied\nby an excellent woman as governess, who had fallen little short\no
f a mother in affection.\n\nSixteen years had Miss Taylor been in Mr. Woodhouse\'s family,\nless as a governess than
a friend, very fond of both daughters,\nbut particularly of Emma.  Between _them_ it was more the intimacy\nof siste
rs.  Even before Miss Taylor had ceased to hold the nominal\noffice of governess, the mildness of her temper had har
dly allowed\nher to impose any restraint; and the shadow of authority being\nnow long passed away, they had been liv
ing together as friend and\nfriend very mutually attached, and Emma doing just what she liked;\nhighly esteeming Mis
s Taylor\'s judgment, but directed chiefly by\nher own.\n\nThe real evils, indeed, of Emma\'s situation were the pow
er of having\nrather too much her own way, and a disposition to think a little\ntoo well of herself; these were the
disadvantages which threatened\nalloy to her many enjoyments.  The danger, however, was at present\nso unperceived,
that they did not by any means rank as misfortunes\nwith her.\n\nSorrow came--a gentle sorrow--but not at all in the
shape of any\ndisagreeable consciousness.--Miss Taylor married.  It was Miss\nTaylor\'s loss which first brought gri
ef.  It was on the wedding-day\nof this beloved friend that Emma first sat in mournful thought\nof any continuance.
The wedding over, and the bride-people gone,\nher father and herself were left to dine together, with no prospect\no
f a third to cheer a long evening.  Her father composed himself\nto sleep after dinner, as usual, and she had then o
```

In [11]:
```python
fname = "./austen-emma.txt"
f = open(fname, 'r')
txt = f.read()
f.close()
wlist = txt.split()
wlist[:50]
```

Out[11]:
```
['[Emma',
 'by',
 'Jane',
 'Austen',
 '1816]',
 'VOLUME',
 'I',
 'CHAPTER',
 'I',
 'Emma',
 'Woodhouse,',
 'handsome,',
 'clever,',
 'and',
 'rich,',
 'with',
 'a',
 'comfortable',
 'home',
 'and',
 'happy',
 'disposition,',
 'seemed',
 'to',
 'unite',
 'some',
 'of',
 'the',
 'best',
 'blessings',
 'of',
 'existence;',
 'and',
 'had',
 'lived',
 'nearly',
 'twenty-one',
 'years',
 'in',
 'the',
 'world',
 'with',
 'very',
 'little',
 'to',
 'distress',
 'or',
 'vex',
 'her.',
 'She']
```

In [12]:
```
fname = "./austen-emma.txt"
f = open(fname, 'r')
etxt = f.read()
f.close()
etxt[-200:]
```

Out[12]:
```
'e deficiencies, the wishes,\nthe hopes, the confidence, the predictions of the small band\nof true friends who witness
ed the ceremony, were fully answered\nin the perfect happiness of the union.\n\n\nFINIS\n'
```

```
In [13]: etoks = nltk.word_tokenize(etxt.lower())
         etoks[-20:]
```

```
Out[13]: ['of',
          'true',
          'friends',
          'who',
          'witnessed',
          'the',
          'ceremony',
          ',',
          'were',
          'fully',
          'answered',
          'in',
          'the',
          'perfect',
          'happiness',
          'of',
          'the',
          'union',
          '.',
          'finis']
```

```
In [14]: len(etoks)
```

```
Out[14]: 191669
```

```
In [15]: etypes = sorted(set(etoks))
         etypes[-10:]

Out[15]: ['younger',
          'youngest',
          'your',
          'yours',
          'yourself',
          'yourself.',
          'youth',
          'youthful',
          'zeal',
          'zigzags']
```

```
In [16]: len(etypes)

Out[16]: 8000
```

```
In [17]: efreq = nltk.FreqDist(etoks)
         efreq['beautiful']

Out[17]: 24
```

STEP 2: list-comprehend Emma

Now, explore the three objects wlist, efreq, and etypes to answer the following questions. Do NOT use the for loop! Every solution must involve use of LIST COMPREHENSION.

Question 1: Words with prefix and suffix

What are the words that start with 'un' and end in 'able'?

```
In [18]: [word for word in etoks if word.startswith("un") & word.endswith("able")]

Out[18]: ['unexceptionable',
          'unsuitable',
          'unreasonable',
          'unreasonable',
          'uncomfortable',
          'unfavourable',
          'unexceptionable',
          'unexceptionable',
          'uncomfortable',
          'unpersuadable',
          'unavoidable',
          'unreasonable',
          'uncomfortable',
          'unsuitable',
          'unmanageable',
          'unexceptionable',
          'unreasonable',
          'unobjectionable',
          'unpersuadable',
          'unsuitable',
          'unreasonable',
          'uncomfortable',
          'unexceptionable',
          'unpardonable',
          'unmanageable',
          'unanswerable',
          'unfavourable',
          'unpersuadable',
          'unaccountable',
          'undesirable',
          'unable',
          'unable',
          'unpardonable',
          'unexceptionable',
          'unreasonable',
          'unreasonable',
          'uncomfortable',
          'unreasonable',
          'unpardonable',
          'unaccountable',
          'unexceptionable',
          'unreasonable',
          'unaccountable']
```

Question 2: Length

How many Emma word types are 15 characters or longer? Exclude hyphenated words.

```
In [19]: tokenizer = nltk.tokenize.WordPunctTokenizer()
         tok = tokenizer.tokenize(etxt)
```

```
In [20]: [word for word in tok if len(word)>15]
```

```
Out[20]: ['companionableness',
          'misunderstanding',
          'incomprehensible',
          'undistinguishing',
          'unceremoniousness',
          'Disingenuousness',
          'disagreeableness',
          'misunderstandings',
          'misunderstandings',
          'misunderstandings',
          'misunderstandings',
          'disinterestedness',
          'unseasonableness']
```

Question 3: Average word length

What's the average length of all Emma word types?

```
In [21]: Avg=sum(len(word) for word in tok)/ len(tok)
         print(Avg)
```

```
3.755268231589122
```

Question 4: Word frequency

How many Emma word types have a frequency count of 200 or more? How many word types appear only once?

In [33]:
```
from nltk import *
fdi = FreqDist(tok)
fdi
```

Out[33]:
```
FreqDist({'[Emma': 1,
          'by': 543,
          'Jane': 199,
          'Austen': 1,
          '1816]': 1,
          'VOLUME': 3,
          'I': 2602,
          'CHAPTER': 55,
          'Emma': 481,
          'Woodhouse,': 110,
          'handsome,': 5,
          'clever,': 7,
          'and': 4305,
          'rich,': 3,
          'with': 1135,
          'a': 2958,
          'comfortable': 27,
          'home': 65,
          'happy': 91,
          'disposition,': 5
```

In [23]:
```
for i,j in fdi.items():
    if j > 200:
        print(i,j)
```

```
Emma 865
by 558
Jane 301
I 3178
Woodhouse 313
, 11454
and 4672
with 1187
a 3004
to 5183
some 248
of 4279
the 4844
; 2199
had 1606
- 574
one 413
in 2118
very 1151
```

Question 5: Emma words not in wlist

Of the Emma word types, how many of them are not found in our list of ENABLE English words, i.e., wlist?

In [34]:
```python
for i,j in fdi.items():
    if j == 1:
        print(i,j)
```

```
[Emma 1
Austen 1
1816] 1
twenty-one 1
vex 1
indulgent 1
indistinct 1
caresses; 1
Between 1
sisters. 1
nominal 1
mildness 1
impose 1
restraint; 1
shadow 1
liked; 1
esteeming 1
disadvantages 1
enjoyments. 1
```

STEP 3: bigrams in Emma

Let's now try out bigrams. Build two objects: e2grams (a list of word bigrams; make sure to cast it as a list) and e2gramfd (a frequency distribution of bigrams) as shown below, and then answer the following questions.

In [24]:
```python
e2grams = list(nltk.bigrams(etoks))
```

In [25]:
```python
 e2gramfd = nltk.FreqDist(e2grams)
```

Question 6: Bigrams

What are the last 10 bigrams?

```
In [35]: last_ten = FreqDist(dict(e2gramfd.most_common()[-10:]))
         last_ten
```

```
Out[35]: FreqDist({('answered', 'in'): 1,
                   ('ceremony,', 'were'): 1,
                   ('fully', 'answered'): 1,
                   ('perfect', 'happiness'): 1,
                   ('the', 'ceremony,'): 1,
                   ('the', 'perfect'): 1,
                   ('the', 'union.'): 1,
                   ('union.', 'FINIS'): 1,
                   ('were', 'fully'): 1,
                   ('witnessed', 'the'): 1})
```

Question 7: Bigram top frequency

What are the top 20 most frequent bigrams?

In [27]:
```python
tokenizer = nltk.tokenize.WhitespaceTokenizer()
tok = tokenizer.tokenize(etxt)
e2grams = list(nltk.bigrams(tok))
e2gramfd = nltk.FreqDist(e2grams)
e2gramfd.most_common(20)
```

Out[27]:
```
[(('to', 'be'), 562),
 (('of', 'the'), 556),
 (('in', 'the'), 431),
 (('I', 'am'), 302),
 (('had', 'been'), 299),
 (('could', 'not'), 270),
 (('it', 'was'), 253),
 (('she', 'had'), 242),
 (('to', 'the'), 236),
 (('have', 'been'), 233),
 (('of', 'her'), 230),
 (('I', 'have'), 214),
 (('and', 'the'), 208),
 (('would', 'be'), 208),
 (('she', 'was'), 206),
 (('do', 'not'), 196),
 (('of', 'his'), 182),
 (('that', 'she'), 178),
 (('to', 'have'), 176),
 (('such', 'a'), 176)]
```

Question 8: Bigram frequency count

How many times does the bigram 'so happy' appear?

In [28]:
```python
for i , j in e2gramfd.items():
    if i == ('so', 'happy'):
        print(i,j)
```

```
('so', 'happy') 3
```

Question 9: Word following 'so'

What are the words that follow 'so'? What are their frequency counts? (For loop will be easier; see if you can utilize list comprehension for this.)

```
In [29]: import re
         from collections import Counter
```

In [30]:
```
words=re.findall(r'so+ \w+',open('austen-emma.txt').read())
ab=Counter(zip(words))
print(ab)
```

```
Counter({('so much',): 95, ('so very',): 76, ('so well',): 30, ('so many',): 27, ('so long',): 27, ('so little',): 20,
('so far',): 17, ('so I',): 14, ('so kind',): 13, ('so good',): 12, ('so often',): 10, ('so soon',): 9, ('so great',):
8, ('so to',): 7, ('so fond',): 7, ('so she',): 7, ('so it',): 6, ('so anxious',): 6, ('so as',): 6, ('so you',): 6,
('so truly',): 6, ('so completely',): 5, ('so obliging',): 5, ('so extremely',): 5, ('so entirely',): 4, ('so happy',):
4, ('so interesting',): 4, ('so fast',): 4, ('so near',): 4, ('so pleased',): 4, ('so few',): 4, ('so that',): 4, ('so
strong',): 4, ('so liberal',): 4, ('so miserable',): 4, ('so happily',): 3, ('so proper',): 3, ('so pleasantly',): 3,
('so superior',): 3, ('so warmly',): 3, ('so bad',): 3, ('so odd',): 3, ('so ill',): 3, ('so delighted',): 3, ('so part
icularly',): 3, ('so easily',): 3, ('so on',): 3, ('so attentive',): 3, ('so fortunate',): 3, ('so glad',): 3, ('so sho
cked',): 3, ('so at',): 3, ('so obliged',): 2, ('so perfectly',): 2, ('so dear',): 2, ('so busy',): 2, ('so did',): 2,
('so forth',): 2, ('so totally',): 2, ('so remarkably',): 2, ('so plainly',): 2, ('so charming',): 2, ('so surprize
d',): 2, ('so early',): 2, ('so too',): 2, ('so easy',): 2, ('so decidedly',): 2, ('so absolutely',): 2, ('so particula
r',): 2, ('so deceived',): 2, ('so palpably',): 2, ('so clever',): 2, ('so short',): 2, ('so cold',): 2, ('so high',):
2, ('so happened',): 2, ('so full',): 2, ('so thoroughly',): 2, ('so equal',): 2, ('so off',): 2, ('so naturally',): 2,
('so afraid',): 2, ('so deep',): 2, ('so kindly',): 2, ('so pale',): 2, ('so noble',): 2, ('so lovely',): 2, ('so ma
d',): 2, ('so nearly',): 2, ('so sorry',): 2, ('so cheerful',): 2, ('so unfeeling',): 2, ('so ready',): 2, ('so unperce
ived',): 1, ('so mild',): 1, ('so constantly',): 1, ('so comfortably',): 1, ('so avowed',): 1, ('so deservedly',): 1,
('so convenient',): 1, ('so just',): 1, ('so apparent',): 1, ('so sorrowful',): 1, ('so spent',): 1, ('so artlessly',):
1, ('so plain',): 1, ('so firmly',): 1, ('so genteel',): 1, ('so _then_',): 1, ('so brilliant',): 1, ('so seldom',): 1,
('so nervous',): 1, ('so indeed',): 1, ('so pack',): 1, ('so doubtful',): 1, ('so with',): 1, ('so contemptible',): 1,
('so slightingly',): 1, ('so by',): 1, ('so loudly',): 1, ('so materially',): 1, ('so hard',): 1, ('so delightful',):
1, ('so pointed',): 1, ('so equalled',): 1, ('so evidently',): 1, ('so immediately',): 1, ('so sought',): 1, ('so excel
lent',): 1, ('so prettily',): 1, ('so extreme',): 1, ('so wonder',): 1, ('so always',): 1, ('so silly',): 1, ('so satis
fied',): 1, ('so smiling',): 1, ('so prosing',): 1, ('so undistinguishing',): 1, ('so apt',): 1, ('so dreadful',): 1,
('so respected',): 1, ('so tenderly',): 1, ('so grieved',): 1, ('so shocking',): 1, ('so conceited',): 1, ('so befor
e',): 1, ('so prevalent',): 1, ('so heavy',): 1, ('so swiftly',): 1, ('so spoken',): 1, ('so or',): 1, ('so overcharge
d',): 1, ('so pleasant',): 1, ('so fenced',): 1, ('so hospitable',): 1, ('so interested',): 1, ('so sanguine',): 1, ('s
o sure',): 1, ('so careless',): 1, ('so rapidly',): 1, ('so frequent',): 1, ('so sensible',): 1, ('so misled',): 1, ('s
o blind',): 1, ('so complaisant',): 1, ('so misinterpreted',): 1, ('so active',): 1, ('so pointedly',): 1, ('so strikin
g',): 1, ('so sudden',): 1, ('so industriously',): 1, ('so partial',): 1, ('so natural',): 1, ('so inevitable',): 1,
('so lately',): 1, ('so beautifully',): 1, ('so distinct',): 1, ('so considerate',): 1, ('so light',): 1, ('so intimat
e',): 1, ('so magnified',): 1, ('so cautious',): 1, ('so confined',): 1, ('so wish',): 1, ('so he',): 1, ('so gloriou
s',): 1, ('so quick',): 1, ('so sweetly',): 1, ('so inseparably',): 1, ('so deserving',): 1, ('so disappointed',): 1,
('so ended',): 1, ('so sluggish',): 1, ('so amiable',): 1, ('so quiet',): 1, ('so idolized',): 1, ('so cried',): 1, ('s
o acceptable',): 1, ('so properly',): 1, ('so reasonable',): 1, ('so delightfully',): 1, ('so rich',): 1, ('so warm',):
1, ('so large',): 1, ('so handsomely',): 1, ('so abundant',): 1, ('so outree',): 1, ('so thoughtful',): 1, ('so mus
t',): 1, ('so effectually',): 1, ('so beautiful',): 1, ('so Patty',): 1, ('so honoured',): 1, ('so close',): 1, ('so im
prudent',): 1, ('so limited',): 1, ('so from',): 1, ('so amusing',): 1, ('so indifferent',): 1, ('so indignant',): 1,
('so said',): 1, ('so right',): 1, ('so wretched',): 1, ('so now',): 1, ('so occupied',): 1, ('so unhappy',): 1, ('so h
ighly',): 1, ('so generally',): 1, ('so exactly',): 1, ('so double',): 1, ('so secluded',): 1, ('so regular',): 1, ('so
determined',): 1, ('so motherly',): 1, ('so the',): 1, ('so glibly',): 1, ('so calculated',): 1, ('so thrown',): 1, ('s
o exclusively',): 1, ('so disgustingly',): 1, ('so needlessly',): 1, ('so does',): 1, ('so resolutely',): 1, ('so woul
d',): 1, ('so infinitely',): 1, ('so fluently',): 1, ('so they',): 1, ('so impatient',): 1, ('so briskly',): 1, ('so vi
gorously',): 1, ('so young',): 1, ('so hardened',): 1, ('so gratified',): 1, ('so received',): 1, ('so then',): 1, ('so
and',): 1, ('so gratefully',): 1, ('so found',): 1, ('so placed',): 1, ('so lain',): 1, ('so his',): 1, ('so arrange
d',): 1, ('so moving',): 1, ('so walking',): 1, ('so when',): 1, ('so favourable',): 1, ('so late',): 1, ('so silen
t',): 1, ('so dull',): 1, ('so irksome',): 1, ('so agitated',): 1, ('so brutal',): 1, ('so cruel',): 1, ('so depresse
d',): 1, ('so no',): 1, ('so justly',): 1, ('so astonished',): 1, ('so will',): 1, ('so simple',): 1, ('so dignifie
d',): 1, ('so suddenly',): 1, ('so a',): 1, ('so herself',): 1, ('so peremptorily',): 1, ('so uneasy',): 1, ('so wonder
ful',): 1, ('so _very_',): 1, ('so expressly',): 1, ('so angry',): 1, ('so anxiously',): 1, ('so strange',): 1, ('so st
outly',): 1, ('so mistake',): 1, ('so mistaken',): 1, ('so dreadfully',): 1, ('so voluntarily',): 1, ('so satisfactor
y',): 1, ('so disinterested',): 1, ('so foolishly',): 1, ('so ingeniously',): 1, ('so entreated',): 1, ('so like',): 1,
('so cordially',): 1, ('so essential',): 1, ('so designedly',): 1, ('so hasty',): 1, ('so richly',): 1, ('so gratefu
l',): 1, ('so tenaciously',): 1, ('so feeling',): 1, ('so engaging',): 1, ('so engaged',): 1, ('so hot',): 1, ('so usef
ul',): 1, ('so attached',): 1, ('so peculiarly',): 1, ('so singularly',): 1, ('so taken',): 1, ('so recently',): 1, ('s
o fresh',): 1, ('so hateful',): 1, ('so heartily',): 1, ('so steady',): 1, ('so complete',): 1, ('so in',): 1, ('so suf
fered',): 1})
```

Question 10: Trigrams

What are the last 10 trigrams? (You can use nltk.util.ngrams() method)

In [36]:
```
last_ten = FreqDist(dict(e3gramfd.most_common()[-10:]))
last_ten
```

Out[36]:
```
FreqDist({('answered', 'in', 'the'): 1,
          ('ceremony,', 'were', 'fully'): 1,
          ('fully', 'answered', 'in'): 1,
          ('in', 'the', 'perfect'): 1,
          ('of', 'the', 'union.'): 1,
          ('perfect', 'happiness', 'of'): 1,
          ('the', 'ceremony,', 'were'): 1,
          ('the', 'perfect', 'happiness'): 1,
          ('the', 'union.', 'FINIS'): 1,
          ('were', 'fully', 'answered'): 1})
```

Question 11: Trigram top frequency

What are the top 10 most frequent trigrams?

```
In [31]: e3grams = list(nltk.trigrams(tok))
         e3gramfd = nltk.FreqDist(e3grams)
         e3gramfd.most_common(10)
```

```
Out[31]: [(('I', 'do', 'not'), 94),
          (('I', 'am', 'sure'), 75),
          (('would', 'have', 'been'), 55),
          (('a', 'great', 'deal'), 55),
          (('she', 'could', 'not'), 49),
          (('could', 'not', 'be'), 45),
          (('she', 'had', 'been'), 44),
          (('it', 'would', 'be'), 43),
          (('do', 'not', 'know'), 43),
          (('Mr.', 'and', 'Mrs.'), 37)]
```

Question 12: Trigram frequency count

How many times does the trigram 'so happy to' appear?

```
In [32]: for i , j in e3gramfd.items():
             if i == ('so', 'happy','to'):
                 print(i,j)
```