Lab 02 NLTK: Language Models

Task 1: Bigrams & Trigrams

In [3]:
```python
import nltk
from nltk.book import *
```
```
*** Introductory Examples for the NLTK Book ***
Loading text1, ..., text9 and sent1, ..., sent9
Type the name of the text or sentence to view it.
Type: 'texts()' or 'sents()' to list the materials.
text1: Moby Dick by Herman Melville 1851
text2: Sense and Sensibility by Jane Austen 1811
text3: The Book of Genesis
text4: Inaugural Address Corpus
text5: Chat Corpus
text6: Monty Python and the Holy Grail
text7: Wall Street Journal
text8: Personals Corpus
text9: The Man Who Was Thursday by G . K . Chesterton 1908
```

In [4]:
```python
words = sorted(set(text1))[280:]
```

In [5]:
```python
print(len(words))
```
```
19037
```

In [6]:
```python
longwords = [w for w in words if len(w) > 16]
```

In [7]:
```python
longwords
```
Out[7]:
```
['cannibalistically',
 'characteristically',
 'circumnavigations',
 'comprehensiveness',
 'indispensableness',
 'preternaturalness',
 'subterraneousness',
 'superstitiousness',
 'uncomfortableness',
 'uncompromisedness',
 'uninterpenetratingly']
```

In [8]:
```python
fdist1 = FreqDist(text1)
```

In [9]:
```python
high_freq = [w for w in words if fdist1[w] > 500]
```

In [10]:
```python
print(high_freq)
```
```
['Ahab', 'But', 'I', 'The', 'a', 'all', 'an', 'and', 'are', 'as', 'at', 'be', 'but', 'by', 'for', 'from', 'had', 'have', 'he', 'him', 'his', 'in', 'into', 'is', 'it', 'like', 'man', 'me', 'more', 'my', 'not', 'now', 'of', 'on',
'one', 'or', 'out', 's', 'ship', 'so', 'some', 'that', 'the', 'their', 'then', 'there', 'they', 'this', 'to', 'up', 'upon', 'was', 'were', 'whale', 'when', 'which', 'with', 'you']
```

In [12]:
```python
import latexify
```

In [13]:
```python
import math

@latexify.function
def IDF(corpus_list, token):
    document_frequency = sum(1 for corpus in corpus_list if token in corpus)

    # Add 1 to both numerator and denominator for smoothing
    inverse_document_frequency = math.log((len(corpus_list) + 1) / (document_frequency + 1), 10)
    return inverse_document_frequency
```

In [14]:
```python
IDF
```
Out[14]:

$$\mathrm{document\_frequency} = \sum_{(\mathrm{corpus} \in \mathrm{corpus\_list}) \land (\mathrm{token} \in \mathrm{corpus})} (1)$$
$$\mathrm{inverse\_document\_frequency} = \log\left(\frac{\mathrm{len}(\mathrm{corpus\_list}) + 1}{\mathrm{document\_frequency} + 1}, 10\right)$$
$$\mathrm{IDF}(\mathrm{corpus\_list}, \mathrm{token}) = \mathrm{inverse\_document\_frequency}$$

In [15]:
```python
high_freq_idf = [w for w in words if IDF(text1, w) > 1]
```

In [16]:
```python
len(high_freq_idf)
```
Out[16]:
```
19027
```

In [17]:
```python
print(high_freq_idf[:10])
```
```
['ABOUT', 'ACCOUNT', 'ADDITIONAL', 'ADVANCING', 'ADVENTURES', 'AFFGHANISTAN', 'AFRICA', 'AFTER', 'AGAINST', 'AHAB']
```

In [18]:
```python
eign_words = [w for w in words if w.endswith('eign') ]
```

In [19]:
```python
eign_words
```
Out[19]:
```
['Sovereign', 'foreign', 'reign', 'sovereign']
```

In [20]:
```python
for w in words:
    if w.endswith('eign'):
        print(w)
```
```
Sovereign
foreign
reign
sovereign
```

In [21]:
```python
len(list(bigrams(text1)))
```
Out[21]:
```
260818
```

In [22]:
```python
text1.collocation_list()
```
Out[22]:
```
[('Sperm', 'Whale'),
 ('Moby', 'Dick'),
 ('White', 'Whale'),
 ('old', 'man'),
 ('Captain', 'Ahab'),
 ('sperm', 'whale'),
 ('Right', 'Whale'),
 ('Captain', 'Peleg'),
 ('New', 'Bedford'),
 ('Cape', 'Horn'),
 ('cried', 'Ahab'),
 ('years', 'ago'),
 ('lower', 'jaw'),
 ('never', 'mind'),
 ('Father', 'Mapple'),
 ('cried', 'Stubb'),
 ('chief', 'mate'),
 ('white', 'whale'),
 ('ivory', 'leg'),
 ('one', 'hand')]
```

In [23]:
```python
from nltk.util import ngrams

list(ngrams(text1, 3))[:10]
```
Out[23]:
```
[('[', 'Moby', 'Dick'),
 ('Moby', 'Dick', 'by'),
 ('Dick', 'by', 'Herman'),
 ('by', 'Herman', 'Melville'),
 ('Herman', 'Melville', '1851'),
 ('Melville', '1851', ']'),
 ('1851', ']', 'ETYMOLOGY'),
 (']', 'ETYMOLOGY', '.'),
 ('ETYMOLOGY', '.', '('),
 ('.', '(', 'Supplied')]
```

In [24]:
```python
from nltk.collocations import TrigramCollocationFinder, TrigramAssocMeasures

TrigramCollocationFinder.from_words(text1).nbest(TrigramAssocMeasures().pmi, 10)
```
Out[24]:
```
[('AFTER', 'EXCHANGING', 'HAILS'),
 ('Anacharsis', 'Clootz', 'deputation'),
 ('CAULKING', 'ITS', 'SEAMS'),
 ('ELIZABETH', 'OAKES', 'SMITH'),
 ('Et', 'tu', 'Brute'),
 ('Ex', 'officio', 'professors'),
 ('Fogo', 'Von', 'Slack'),
 ('Ganders', 'formally', 'indite'),
 ('Kentucky', 'Mammoth', 'Cave'),
 ('LANTERNS', 'BUSILY', 'FILING')]
```

10 frequently occuring Bigrams

In [25]:
```python
from nltk import bigrams, FreqDist

def top_frequent_bigrams(text, n = 10):
    bigrams_list = list(bigrams(text))
    bigram_freq = FreqDist(bigrams_list)

    return bigram_freq.most_common(n)

text1_bigrams = top_frequent_bigrams(text1)
text2_bigrams = top_frequent_bigrams(text2)
text3_bigrams = top_frequent_bigrams(text3)

print("Top 10 Frequently Occurring Bigrams in Text1:")
print(text1_bigrams)
print("\nTop 10 Frequently Occurring Bigrams in Text2:")
print(text2_bigrams)
print("\nTop 10 Frequently Occurring Bigrams in Text3:")
print(text3_bigrams)
```
```
Top 10 Frequently Occurring Bigrams in Text1:
[((',', 'and'), 2607), (('of', 'the'), 1847), (("'", 's'), 1737), (('in', 'the'), 1120), ((',', 'the'), 908), ((';', 'and'), 853), (('to', 'the'), 712), (('.', 'But'), 596), ((',', 'that'), 584), (('.', '"'), 557)]

Top 10 Frequently Occurring Bigrams in Text2:
[((',', 'and'), 1598), (("'", 's'), 700), ((';', 'and'), 605), (('Mrs', '.'), 529), (('of', 'the'), 430), (('to', 'be'), 428), (('."', '"'), 428), ((',', '"'), 392), (('.', '"'), 369), (('in', 'the'), 348)]

Top 10 Frequently Occurring Bigrams in Text3:
[((',', 'and'), 1491), (('.', 'And'), 1038), (('of', 'the'), 372), (('in', 'the'), 287), ((';', 'and'), 262), (('said', ','), 259), (("'", 's'), 255), (('And', 'he'), 192), (('And', 'the'), 185), (('said', 'unto'), 178)]
```

5 frequently occuring Trigrams

In [26]:
```python
from nltk import trigrams, FreqDist
from nltk.book import text1, text2, text3

def top_frequent_trigrams(text, n=5):
    trigrams_list = list(trigrams(text))
    trigram_freq = FreqDist(trigrams_list)
    return trigram_freq.most_common(n)

text1_trigrams = top_frequent_trigrams(text1, n=5)
text2_trigrams = top_frequent_trigrams(text2, n=5)
text3_trigrams = top_frequent_trigrams(text3, n=5)

print("\nTop 5 Frequently Occurring Trigrams in Text1:")
print(text1_trigrams)
print("\nTop 5 Frequently Occurring Trigrams in Text2:")
print(text2_trigrams)
print("\nTop 5 Frequently Occurring Trigrams in Text3:")
print(text3_trigrams)
```
```
Top 5 Frequently Occurring Trigrams in Text1:
[((',', 'and', 'the'), 187), (('don', "'", 't'), 103), (('of', 'the', 'whale'), 101), ((',', 'in', 'the'), 93), ((',', 'then', ','), 87)]

Top 5 Frequently Occurring Trigrams in Text2:
[(('Mrs', '.', 'Jennings'), 230), (('Mrs', '.', 'Dashwood'), 121), ((',', 'however', ','), 88), ((',', 'and', 'the'), 87), (('.', 'Mrs', '.'), 80)]

Top 5 Frequently Occurring Trigrams in Text3:
[(('.', 'And', 'he'), 162), (('.', 'And', 'the'), 158), (('the', 'land', 'of'), 101), (('he', 'said', ','), 86), (('And', 'he', 'said'), 84)]
```

Number of Words with Length > 16:

In [27]:
```python
def count_words_length_greater_than(text, length=16):
    long_words = [word for word in text if len(word) > length]
    return len(long_words)

text1_long_words_count = count_words_length_greater_than(text1)
text2_long_words_count = count_words_length_greater_than(text2)
text3_long_words_count = count_words_length_greater_than(text3)

print("\nNumber of Words with Length > 16 in Text1:", text1_long_words_count)
print("Number of Words with Length > 16 in Text2:", text2_long_words_count)
print("Number of Words with Length > 16 in Text3:", text3_long_words_count)
```
```
Number of Words with Length > 16 in Text1: 14
Number of Words with Length > 16 in Text2: 3
Number of Words with Length > 16 in Text3: 0
```

Number of words with frequency > 500

In [28]:
```python
def count_words_frequency_greater_than(text, frequency=500):
    word_freq = FreqDist(text)
    high_frequency_words = [word for word, freq in word_freq.items() if freq > frequency]
    return len(high_frequency_words)

text1_high_freq_words_count = count_words_frequency_greater_than(text1)
text2_high_freq_words_count = count_words_frequency_greater_than(text2)
text3_high_freq_words_count = count_words_frequency_greater_than(text3)

print("\nNumber of Words with Frequency > 500 in Text1:", text1_high_freq_words_count)
print("Number of Words with Frequency > 500 in Text2:", text2_high_freq_words_count)
print("Number of Words with Frequency > 500 in Text3:", text3_high_freq_words_count)
```
```
Number of Words with Frequency > 500 in Text1: 67
Number of Words with Frequency > 500 in Text2: 45
Number of Words with Frequency > 500 in Text3: 13
```

Number of ending in “ed”

In [29]:
```python
def count_words_ending_in_ed(text):
    ed_words = [word for word in text if word.endswith("ed")]
    return len(ed_words)

text1_ed_words_count = count_words_ending_in_ed(text1)
text2_ed_words_count = count_words_ending_in_ed(text2)
text3_ed_words_count = count_words_ending_in_ed(text3)

print("\nNumber of Words Ending in 'ed' in Text1:", text1_ed_words_count)
print("Number of Words Ending in 'ed' in Text2:", text2_ed_words_count)
print("Number of Words Ending in 'ed' in Text3:", text3_ed_words_count)
```
```
Number of Words Ending in 'ed' in Text1: 8491
Number of Words Ending in 'ed' in Text2: 4866
Number of Words Ending in 'ed' in Text3: 1238
```

Task 2: Accessing Corpora

In [30]:
```python
nltk.corpus.gutenberg.fileids()
```
Out[30]:
```
['austen-emma.txt',
 'austen-persuasion.txt',
 'austen-sense.txt',
 'bible-kjv.txt',
 'blake-poems.txt',
 'bryant-stories.txt',
 'burgess-busterbrown.txt',
 'carroll-alice.txt',
 'chesterton-ball.txt',
 'chesterton-brown.txt',
 'chesterton-thursday.txt',
 'edgeworth-parents.txt',
 'melville-moby_dick.txt',
 'milton-paradise.txt',
 'shakespeare-caesar.txt',
 'shakespeare-hamlet.txt',
 'shakespeare-macbeth.txt',
 'whitman-leaves.txt']
```

In [31]:
```python
gutenberg_sc = nltk.corpus.gutenberg.words('shakespeare-caesar.txt')
```

In [32]:
```python
len(gutenberg_sc)
```
Out[32]:
```
25833
```

In [33]:
```python
from nltk.corpus import brown
brown.words()
```
Out[33]:
```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

In [34]:
```python
type(brown)
```
Out[34]:
```
nltk.corpus.reader.tagged.CategorizedTaggedCorpusReader
```

In [35]:
```python
import nltk

# Download the necessary data
nltk.download('abc')

# Load the file from the NLTK corpus
psh_txt = nltk.corpus.abc.raw('science.txt')
```
```
[nltk_data] Downloading package abc to /home/jadi/nltk_data...
[nltk_data]   Package abc is already up-to-date!
```

In [36]:
```python
from nltk.util import ngrams
words = nltk.word_tokenize(psh_txt)
psh_bigrams = list(ngrams(words, 2))
psh_trigrams = list(ngrams(words, 3))
```

In [39]:
```python
import nltk
from nltk.corpus import brown
from nltk import ngrams

# Download the Brown corpus if not already downloaded
nltk.download('brown')

# Get the wordlist from the Brown corpus and create bigrams
wordlist = brown.words()
bigramlist = list(ngrams(wordlist, 2))

# Print the first 10 bigrams as an example
print(bigramlist[:10])
```
```
[nltk_data] Downloading package brown to /home/jadi/nltk_data...
[nltk_data]   Package brown is already up-to-date!
[('The', 'Fulton'), ('Fulton', 'County'), ('County', 'Grand'), ('Grand', 'Jury'), ('Jury', 'said'), ('said', 'Friday'), ('Friday', 'an'), ('an', 'investigation'), ('investigation', 'of'), ('of', "Atlanta's")]
```

Task 3: Generating Random Text with Bigrams

In [40]:
```python
cfd = nltk.ConditionalFreqDist(bigramlist)
```

In [41]:
```python
cfd
```
Out[41]:
```
<ConditionalFreqDist with 56057 conditions>
```

In [42]:
```python
def generate_model(cfdist, word, num):
    for i in range(num):
        print(word, end=' ')
        word = cfdist[word].max()
```

In [43]:
```python
generate_model(cfd, 'Today', 10)
```
```
Today , and the same time , and the same
```

In [44]:
```python
generate_model(cfd, 'Today', 30)
```
```
Today , and the same time , and the same time , and the same time , and the same time , and the same time , and the same
```

Task 30 word generated sentence

In [56]:
```python
len(cfd)
```
Out[56]:
```
56057
```

In [57]:
```python
print('30 word generated sentence')
```
```
30 word generated sentence
```

In [58]:
```python
generate_model(cfd, 'Today', 30)
```
```
Today , and the same time , and the same time , and the same time , and the same time , and the same time , and the same
```

In [ ]:

In [ ]: