Task 1: Bigrams & Trigrams

In [57]:
```
import nltk # Importing NLTK
```

In [ ]:
```
```

In [95]:
```
from nltk.book import *
from nltk.util import ngrams
from nltk.collocations import *
from nltk.collocations import TrigramCollocationFinder, TrigramAssocMeasures
```

In [ ]:
```
```

In [96]:
```
list1 = [text1, text2, text3]
for idx,i in enumerate(list1, start = 1):
    print(f"Start for Text {idx}-> \n", i)
    words = sorted(set(i))[280:]
    long_words = [w for w in words if len(w) > 16]
    print("\nPrint Long Words : \n",long_words)
    freqdist = FreqDist(text1)
    high_freq = [w for w in words if freqdist[w] > 500]
    print("\nPrint High Frequency Words : \n",high_freq)
    words = sorted(set(i))
    ed_words = [w for w in words if w.endswith('ed')]
    print("\nLength of Words End with ed \n",len(ed_words))
    print(ed_words)
    bigram = list(bigrams(i))
    bigram_10 = list(bigrams(i))[:10]
    print("\nPrint 10 Bigrams : \n ", bigram_10)
    trigram = list(ngrams(i, 3))
    trigram_5 = TrigramCollocationFinder.from_words(i).nbest(TrigramAssocMeasures().pmi, 5)
    print("\nPrint 5 Trigrams :\n ", trigram_5)
    print("================================ \n")
```

```
oduced', 'profaned', 'professed', 'projected', 'prolonged', 'promised', 'pronged', 'pronounced', 'prope
lled', 'prophesied', 'proportioned', 'proposed', 'propped', 'prosecuted', 'protected', 'protested', 'pr
otracted', 'protruded', 'proved', 'provided', 'provoked', 'pryed', 'published', 'puffed', 'pulled', 'pu
nctured', 'purchased', 'purposed', 'pursed', 'pursued', 'pushed', 'puzzled', 'quadruped', 'quaffed', 'q
uailed', 'quaked', 'qualified', 'quarried', 'questioned', 'quilted', 'quitted', 'quivered', 'quoted',
'raced', 'rafted', 'ragged', 'rainbowed', 'raised', 'raked', 'rallied', 'rambled', 'rammed', 'ranged',
'raved', 'ravished', 'razeed', 'reached', 'reaped', 'reappeared', 'rebelled', 'recalled', 'received',
'rechristened', 'rechurned', 'reckoned', 'recognised', 'recommended', 'reconciled', 'recorded', 'recoun
ted', 'recovered', 'recrossed', 'recurred', 'red', 'reddened', 'redeemed', 'redoubled', 'redoubted', 'r
educed', 'reefed', 'reeled', 'reeved', 'referred', 'reflected', 'refrained', 'refused', 'regained', 're
garded', 'rehearsed', 'reigned', 'reined', 'reinforced', 'reiterated', 'rejoined', 'related', 'relaxe
d', 'relented', 'relied', 'relieved', 'remained', 'remarked', 'remembered', 'reminded', 'remonstrated',
'removed', 'rendered', 'renewed', 'renounced', 'renowned', 'rented', 'repaired', 'repeated', 'repelle
d', 'replaced', 'replenished', 'replied', 'reported', 'represented', 'repressed', 'reputed', 'require
d', 'rescued', 'resembled', 'reserved', 'resided', 'resigned', 'resisted', 'resolved', 'resounded', 're
spected', 'responded', 'rested', 'restored', 'restrained', 'restricted', 'resumed', 'retained', 'retard
ed', 'retired', 'retraced', 'retreated', 'returned', 'revealed', 'revelled', 'revered', 'reverenced',
'reversed', 'reviewed', 'revived', 'revivified', 'revolved', 'ribbed', 'ribboned', 'rifled', 'rigged',
'righted', 'rimmed', 'ringed', 'ripped', 'risked', 'riveted', 'roared', 'roasted', 'robbed', 'robed',
```

In [97]:
```
text1.collocations()
```

```
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 [98]:
```
text3.collocations()
```

```
said unto; pray thee; thou shalt; thou hast; thy seed; years old;
spake unto; thou art; LORD God; every living; God hath; begat sons;
seven years; shalt thou; little ones; living creature; creeping thing;
savoury meat; thirty years; every beast
```

In [99]:
```
text2.collocations()
```

```
Colonel Brandon; Sir John; Lady Middleton; Miss Dashwood; every thing;
thousand pounds; dare say; Miss Steeles; said Elinor; Miss Steele;
every body; John Dashwood; great deal; Harley Street; Berkeley Street;
Miss Dashwoods; young man; Combe Magna; every day; next morning
```

Task 2: Accessing Corpora

In [100]:
```
nltk.corpus.gutenberg.fileids()
```

Out[100]:
```
['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 [101]:
```
gutenberg_sc = nltk.corpus.gutenberg.words('shakespeare-caesar.txt')
```

In [102]:
```
gutenberg_sc
```

Out[102]:
```
['[', 'The', 'Tragedie', 'of', 'Julius', 'Caesar', ...]
```

In [103]:
```
from nltk.corpus import brown
brown.words()
```

Out[103]:
```
['The', 'Fulton', 'County', 'Grand', 'Jury', 'said', ...]
```

In [104]:
```
daw_raw = nltk.data.load('dawood.txt', format ='raw')
daw_txt = nltk.data.load('dawood.txt', format ='text')
```

In [105]:
```
from nltk.util import ngrams
words = nltk.word_tokenize(daw_txt)
daw_bigrams = list(ngrams(words, 2))
daw_trigrams = list(ngrams(words, 3))
```

In [106]:
```
daw_bigrams
```

Out[106]:
```
[('Natural', 'Language'),
 ('Language', 'Processing'),
 ('Processing', '('),
 ('(', 'NLP'),
 ('NLP', ')'),
 (')', 'represents'),
 ('represents', 'a'),
 ('a', 'transformative'),
 ('transformative', 'field'),
 ('field', 'within'),
 ('within', 'the'),
 ('the', 'domain'),
 ('domain', 'of'),
 ('of', 'artificial'),
 ('artificial', 'intelligence'),
 ('intelligence', ','),
 (',', 'dedicated'),
 ('dedicated', 'to'),
 ('to', 'unraveling'),
```

In [107]:
```
daw_trigrams
```

Out[107]:
```
[('Natural', 'Language', 'Processing'),
 ('Language', 'Processing', '('),
 ('Processing', '(', 'NLP'),
 ('(', 'NLP', ')'),
 ('NLP', ')', 'represents'),
 (')', 'represents', 'a'),
 ('represents', 'a', 'transformative'),
 ('a', 'transformative', 'field'),
 ('transformative', 'field', 'within'),
 ('field', 'within', 'the'),
 ('within', 'the', 'domain'),
 ('the', 'domain', 'of'),
 ('domain', 'of', 'artificial'),
 ('of', 'artificial', 'intelligence'),
 ('artificial', 'intelligence', ','),
 ('intelligence', ',', 'dedicated'),
 (',', 'dedicated', 'to'),
 ('dedicated', 'to', 'unraveling'),
 ('to', 'unraveling', 'the'),
```

In [108]:
```
from nltk.util import ngrams
```

In [126]:
```
import os
from nltk.corpus import PlaintextCorpusReader
corpus_root = os.path.expanduser('/home/chattha/nltk_data/corpora/dawood/')
daw_corpus = PlaintextCorpusReader(corpus_root, '.*', encoding = 'utf-8')
wordlist = daw_corpus.words()
bigramlist = list(ngrams(wordlist,2))
# Get the raw text from the corpus
daw_corpus_raw = daw_corpus.raw()
```

In [127]:
```
daw_corpus_raw
```

Out[127]:
```
'Natural Language Processing (NLP) represents a transformative field within the domain of artificial in
telligence, dedicated to unraveling the complexities of human language and enabling machines to compreh
end, interpret, and generate human-like text. At its core, NLP seeks to bridge the communication gap be
tween humans and computers, opening up avenues for seamless interaction and understanding. This interdi
sciplinary field draws upon linguistics, computer science, and machine learning, fusing these domains t
o create algorithms and models that can navigate the nuances of natural language.One of the foundationa
l challenges that NLP addresses is the inherent ambiguity and variability present in human language. Un
like structured data, natural language is rich, context-dependent, and laden with intricacies such as i
dioms, metaphors, and cultural nuances. NLP algorithms grapple with these challenges, aiming to impart
machines with the capability to comprehend the subtleties of language, discern sentiment, and extract m
eaningful information from vast corpora of text.The evolution of NLP can be traced back to the mid-20th
century when researchers began exploring the possibility of teaching computers to understand and genera
te human language. Early endeavors were marked by rule-based systems, where linguistic rules were manua
lly crafted to parse and analyze text. However, the inherent limitations of rule-based approaches becam
e evident as the scale and complexity of linguistic patterns expanded.The advent of machine learning an
d the availability of large-scale linguistic datasets revolutionized the NLP landscape. Statistical mod
els, particularly those based on probabilistic approaches, gained prominence. These models, often utili
zing techniques like Hidden Markov Models and n-gram analysis, demonstrated improved language understan
ding capabilities by learning patterns and probabilities from data. Yet, they struggled with the contex
```

In [128]:
```
from nltk.util import ngrams
words = nltk.word_tokenize(daw_corpus_raw)
daw_bigrams = list(ngrams(words, 2))
daw_trigrams = list(ngrams(words, 3))
```

In [129]:
```
daw_bigrams
```

Out[129]:
```
[('Natural', 'Language'),
 ('Language', 'Processing'),
 ('Processing', '('),
 ('(', 'NLP'),
 ('NLP', ')'),
 (')', 'represents'),
 ('represents', 'a'),
 ('a', 'transformative'),
 ('transformative', 'field'),
 ('field', 'within'),
 ('within', 'the'),
 ('the', 'domain'),
 ('domain', 'of'),
 ('of', 'artificial'),
 ('artificial', 'intelligence'),
 ('intelligence', ','),
 (',', 'dedicated'),
 ('dedicated', 'to'),
 ('to', 'unraveling'),
```

In [130]:
```
daw_trigrams
```

Out[130]:
```
[('Natural', 'Language', 'Processing'),
 ('Language', 'Processing', '('),
 ('Processing', '(', 'NLP'),
 ('(', 'NLP', ')'),
 ('NLP', ')', 'represents'),
 (')', 'represents', 'a'),
 ('represents', 'a', 'transformative'),
 ('a', 'transformative', 'field'),
 ('transformative', 'field', 'within'),
 ('field', 'within', 'the'),
 ('within', 'the', 'domain'),
 ('the', 'domain', 'of'),
 ('domain', 'of', 'artificial'),
 ('of', 'artificial', 'intelligence'),
 ('artificial', 'intelligence', ','),
 ('intelligence', ',', 'dedicated'),
 (',', 'dedicated', 'to'),
 ('dedicated', 'to', 'unraveling'),
 ('to', 'unraveling', 'the'),
```

In [131]:
```
wordlist = daw_corpus.words()
bigramlist = list(ngrams(wordlist,2))
```

In [132]:
```
bigramlist
```

```
 ('for', 'seamless'),
 ('seamless', 'interaction'),
 ('interaction', 'and'),
 ('and', 'understanding'),
 ('understanding', '.'),
 ('.', 'This'),
 ('This', 'interdisciplinary'),
 ('interdisciplinary', 'field'),
 ('field', 'draws'),
 ('draws', 'upon'),
 ('upon', 'linguistics'),
 ('linguistics', ','),
 (',', 'computer'),
 ('computer', 'science'),
 ('science', ','),
 (',', 'and'),
 ('and', 'machine'),
 ('machine', 'learning'),
 ('learning', ','),
 (',', 'fusing'),
```

Task 3: Generating Random Text with Bigrams

In [116]:
```
cfd = nltk.ConditionalFreqDist(bigramlist)
```

In [117]:
```
def generate_model(cfdist, word, num):
    for i in range(num):
        print(word, end=' ')
        # Check if the word is in the set of available samples
        if word in cfdist:
            word = cfdist[word].max()
        else:
            break  # Break the loop if there are no more samples
```

In [118]:
```
generate_model(cfd, 'computer', 30)
```

```
computer vision , and generate human language . The future trajectory of NLP . The future trajectory of N
LP . The future trajectory of NLP . The future trajectory of
```

In [119]:
```
len(wordlist)
```

Out[119]:
```
1628
```

In [ ]:
```

```