Lab Assignment - 1

Reg ex

In [1]:
```
a=5
```

In [13]:
```
import re
```

In [14]:
```
allstr= "sat,hat,mat,pat"
```

In [22]:
```
re.findall(r'[a-m]',allstr)
```

Out[22]:
```
['a', 'h', 'a', 'm', 'a', 'a']
```

In [23]:
```
a = "vai abc bdh irs ola ubol vid asc"
```

In [37]:
```
re.findall(r'\w\w',a)
```

Out[37]:
```
['va', 'ab', 'bd', 'ir', 'ol', 'ub', 'ol', 'vi', 'as']
```

In [50]:
```
re.findall(r'\b\w\w',a)
```

Out[50]:
```
['va', 'ab', 'bd', 'ir', 'ol', 'ub', 'vi', 'as']
```

In [38]:
```
re.findall(r'\b(\w{1,2})\w*\b',a)
```

Out[38]:
```
['va', 'ab', 'bd', 'ir', 'ol', 'ub', 'vi', 'as']
```

In [51]:
```
s = '''abc@gmail.com , pqr@gmail.com , step@gmail.in , vaibhav@email.edu'''
```

In [54]:
```
re.findall(r'^[a-zA-Z0-9]+@[a-zA-Z0-9]+\.[a-zA-Z]{2,}$',s)
```

Out[54]:
```
[]
```

In [55]:
```
re.findall(r'^\w+@+\w+\.+[a-zA-Z]{2,}$',"abc@gmail.com")
```

Out[55]:
```
['abc@gmail.com']
```

In [59]:
```
re.findall(r'^\w+@+\w+\.+[a-zA-Z]{2,}$')
```

Out[59]:
```
[]
```

In [67]:
```
re.findall(r'\@\w+.(\w+)',s)
```

Out[67]:
```
['com', 'com', 'in', 'edu']
```

In [69]:
```
b= 'virat kohli is scoring centuries like flowing water in river'
```

In [70]:
```
b.split(" ")
```

Out[70]:
```
['virat',
 'kohli',
 'is',
 'scoring',
 'centuries',
 'like',
 'flowing',
 'water',
 'in',
 'river']
```

In [74]:
```
p= "This is a sample paragraph. It contains multiple sentences. Each sentence ends with a p
```

In [75]:
```
re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s', p)
```

Out[75]:
```
['This is a sample paragraph.',
 'It contains multiple sentences.',
 'Each sentence ends with a period.',
 'mr.x is very happy!']
```

In [76]:
```
re.split(r'(?<=[.!?])\s+', p)
```

Out[76]:
```
['This is a sample paragraph.',
 'It contains multiple sentences.',
 'Each sentence ends with a period.',
 'mr.x is very happy!']
```

In [80]:
```
re.split(r'(?<=[.!?])', p)
```

Out[80]:
```
['This is a sample paragraph.',
 ' It contains multiple sentences.',
 ' Each sentence ends with a period.',
 ' mr.',
 'x is very happy!',
 '']
```

In [81]:
```
re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)', p)
```

Out[81]:
```
['This is a sample paragraph.',
 ' It contains multiple sentences.',
 ' Each sentence ends with a period.',
 ' mr.',
 'x is very happy!']
```

In [82]:
```
re.split(r'([.!?])', p)
```

Out[82]:
```
['This is a sample paragraph',
 '.',
 ' It contains multiple sentences',
 '.',
 ' Each sentence ends with a period',
 '.',
 ' mr',
 '.',
 'x is very happy',
 '!',
 '']
```

In [83]:
```
re.split(r'[.!?]', p)
```

Out[83]:
```
['This is a sample paragraph',
 ' It contains multiple sentences',
 ' Each sentence ends with a period',
 ' mr',
 'x is very happy',
 '']
```

In [84]:
```
re.split(r'[.!?]\s', p)
```

Out[84]:
```
['This is a sample paragraph',
 'It contains multiple sentences',
 'Each sentence ends with a period',
 'mr.x is very happy!']
```

In [105]:
```
d = "This is a sample paragraph. It contains multiple sentences. Each sentence ends with a
```

In [106]:
```
re.split(r'[.!?]\s+', d)
```

Out[106]:
```
['This is a sample paragraph',
 'It contains multiple sentences',
 'Each sentence ends with a period',
 'Mr',
 'Doctor is very happy!']
```

In [107]:
```
re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s', d)
```

Out[107]:
```
['This is a sample paragraph.',
 'It contains multiple sentences.',
 'Each sentence ends with a period.',
 'Mr. Doctor is very happy!']
```

In [108]:
```
d.split(r'(?<=[.!?])\s+|\s+(?=[A-Z](?<![a-z]))')
```

Out[108]:
```
['This is a sample paragraph. It contains multiple sentences. Each sentence ends with a pe
riod. Mr. Doctor is very happy!']
```

In [109]:
```
re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?|\!)\s',d)
```

Out[109]:
```
['This is a sample paragraph.',
 'It contains multiple sentences.',
 'Each sentence ends with a period.',
 'Mr. Doctor is very happy!']
```

In [110]:
```
re.split(r'(?<=[.!?])\s+',d)
```

Out[110]:
```
['This is a sample paragraph.',
 'It contains multiple sentences.',
 'Each sentence ends with a period.',
 'Mr.',
 'Doctor is very happy!']
```

In [111]:
```
import os
```

In [ ]:
```

```

In [115]:
```
with open(r"C:\Users\raval\Downloads\stopwords-en.txt", 'r',encoding='utf-8') as file:
        st = file.read()
        print(st)
```

```
'll
'tis
'twas
've
10
39
a
a's
able
ableabout
about
above
abroad
abst
accordance
according
accordingly
across
act
```

In [118]:
```
# Sample paragraph
paragraph = """
Natural language processing (NLP) is a subfield of artificial intelligence content improve
"""

stopwords_list = st.split('\n')

stopwords_list = [stopword.strip() for stopword in stopwords_list if stopword.strip()]

words_in_paragraph = paragraph.split()

non_stopwords_list = [word for word in words_in_paragraph if word.lower() not in stopwords_

print("\nNon-Stopwords:", non_stopwords_list)
```

```
Non-Stopwords: ['Natural', 'language', 'processing', '(NLP)', 'subfield', 'artificial', 'i
ntelligence', 'content', 'improve', 'efficiency', 'algorithms', 'highlight', 'informatio
n.', 'create', 'list', 'stopwords', 'filter', 'sample', 'paragraph.']
```

In [117]:

```
# Sample paragraph
paragraph = """
Natural language processing (NLP) is a subfield of artificial intelligence content improve
"""

# Split the content into a list of stopwords
stopwords_list = st.split('\n')

# Remove any empty strings or whitespaces from the list
stopwords_list = [stopword.strip() for stopword in stopwords_list if stopword.strip()]

# Split the paragraph into words
words_in_paragraph = paragraph.split()

# Create a list of non-stopwords
non_stopwords_list = [word for word in words_in_paragraph if word.lower() not in stopwords_

# Print the list of stopwords and non-stopwords
print("Stopwords:", stopwords_list)
print("\nNon-Stopwords:", non_stopwords_list)
```

Stopwords: ["'ll", "'tis", "'twas", "'ve", '10', '39', 'a', "a's", 'able', 'ableabout', 'about', 'above', 'abroad', 'abst', 'accordance', 'according', 'accordingly', 'across', 'act', 'actually', 'ad', 'added', 'adj', 'adopted', 'ae', 'af', 'affected', 'affecting', 'affects', 'after', 'afterwards', 'ag', 'again', 'against', 'ago', 'ah', 'ahead', 'ai', "ain't", 'aint', 'al', 'all', 'allow', 'allows', 'almost', 'alone', 'along', 'alongside', 'already', 'also', 'although', 'always', 'am', 'amid', 'amidst', 'among', 'amongst', 'amoungst', 'amount', 'an', 'and', 'announce', 'another', 'any', 'anybody', 'anyhow', 'anymore', 'anyone', 'anything', 'anyway', 'anyways', 'anywhere', 'ao', 'apart', 'apparently', 'appear', 'appreciate', 'appropriate', 'approximately', 'aq', 'ar', 'are', 'area', 'areas', 'aren', "aren't", 'arent', 'arise', 'around', 'arpa', 'as', 'aside', 'ask', 'asked', 'asking', 'asks', 'associated', 'at', 'au', 'auth', 'available', 'aw', 'away', 'awfully', 'az', 'b', 'ba', 'back', 'backed', 'backing', 'backs', 'backward', 'backwards', 'bb', 'bd', 'be', 'became', 'because', 'become', 'becomes', 'becoming', 'been', 'before', 'beforehand', 'began', 'begin', 'beginning', 'beginnings', 'begins', 'behind', 'being', 'beings', 'believe', 'below', 'beside', 'besides', 'best', 'better', 'between', 'beyond', 'bf', 'bg', 'bh', 'bi', 'big', 'bill', 'billion', 'biol', 'bj', 'bm', 'bn', 'bo', 'both', 'bottom', 'br', 'brief', 'briefly', 'bs', 'bt', 'but', 'buy', 'bv', 'bw', 'by', 'bz', 'c', "c'mon", "c's", 'ca', 'call', 'came', 'can', "can't", 'cannot', 'cant', 'caption', 'case', 'cases', 'cause', 'causes', 'cc', 'cd', 'certain', 'certainly', 'cf', 'cg', 'ch', 'changes', 'ci', 'ck', 'cl', 'clear', 'clearly', 'click', 'cm', 'cmon', 'cn', 'co', 'co.', 'com', 'come', 'comes', 'computer', 'con', 'concerning', 'consequently', 'consider', 'considering', 'contain', 'containing', 'contains', 'copy', 'corresponding', 'could', "could've", 'couldn', "couldn't", 'couldnt', 'course', 'cr', 'cry', 'cs', 'cu', 'currently', 'cv', 'cx', 'cy', 'cz', 'd', 'dare', "daren't", 'darent', 'date', 'de', 'dear', 'definitely', 'describe', 'described', 'despite', 'detail', 'did', 'didn', "didn't", 'didnt', 'differ', 'different', 'differently', 'directly', 'dj', 'dk', 'dm', 'do', 'does', 'doesn', "doesn't", 'doesnt', 'doing', 'don', "don't", 'done', 'dont', 'doubtful', 'down', 'downed', 'downing', 'downs', 'downwards', 'due', 'during', 'dz', 'e', 'each', 'early', 'ec', 'ed', 'edu', 'ee', 'effect', 'eg', 'eh', 'eight', 'eighty', 'either', 'eleven', 'else', 'elsewhere', 'empty', 'end', 'ended', 'ending', 'ends', 'enough', 'entirely', 'er', 'es', 'especially', 'et', 'et-al', 'etc', 'even', 'evenly', 'ever', 'evermore', 'every', 'everybody', 'everyone', 'everything', 'everywhere', 'ex', 'exactly', 'example', 'except', 'f', 'face', 'faces', 'fact', 'facts', 'fairly', 'far', 'farther', 'felt', 'few', 'fewer', 'ff', 'fi', 'fifteen', 'fifth', 'fifty', 'fify', 'fill', 'find', 'finds', 'fire', 'first', 'five', 'fix', 'fj', 'fk', 'fm', 'fo', 'followed', 'following', 'follows', 'for', 'forever', 'former', 'formerly', 'forth', 'forty', 'forward', 'found', 'four', 'fr', 'free', 'from', 'front', 'full', 'fully', 'further', 'furthered', 'furthering', 'furthermore', 'furthers', 'fx', 'g', 'ga', 'gave', 'gb', 'gd', 'ge', 'general', 'generally', 'get', 'gets', 'getting', 'gf', 'gg', 'gh', 'gi', 'give', 'given', 'gives', 'giving', 'gl', 'gm', 'gmt', 'gn', 'go', 'goes', 'going', 'gone', 'good', 'goods', 'got', 'gotten', 'gov', 'gp', 'gq', 'gr', 'great', 'greater', 'greatest', 'greetings', 'group', 'grouped', 'grouping', 'groups', 'gs', 'gt', 'gu', 'gw', 'gy', 'h', 'had', "hadn't", 'hadnt', 'half', 'happens', 'hardly', 'has', 'hasn', "hasn't", 'hasnt', 'have', 'haven', "haven't", 'havent', 'having', 'he', "he'd", "he'll", "he's", 'hed', 'hell', 'hello', 'help', 'hence', 'her', 'here', "here's", 'hereafter', 'hereby', 'herein', 'heres', 'hereupon', 'hers', 'herself', 'herse”', 'hes', 'hi', 'hid', 'high', 'higher', 'highest', 'him', 'himself', 'himse”', 'his', 'hither', 'hk', 'hm', 'hn', 'home', 'homepage', 'hopefully', 'how', "how'd", "how'll", "how's", 'howbeit', 'however', 'hr', 'ht', 'htm', 'html', 'http', 'hu', 'hundred', 'i', "i'd", "i'll", "i'm", "i've", 'i.e.', 'id', 'ie', 'if', 'ignored', 'ii', 'il', 'ill', 'im', 'immediate', 'immediately', 'importance', 'important', 'in', 'inasmuch', 'inc', 'inc.', 'indeed', 'index', 'indicate', 'indicated', 'indicates', 'information', 'inner', 'inside', 'insofar', 'instead', 'int', 'interest', 'interested', 'interesting', 'interests', 'into', 'invention', 'inward', 'io', 'iq', 'ir', 'is', 'isn', "isn't", 'isnt', 'it', "it'd", "it'll", "it's", 'itd', 'itll', 'its', 'itself', 'itse”', 'ive', 'j', 'je', 'jm', 'jo', 'join', 'jp', 'just', 'k', 'ke', 'keep', 'keeps', 'kept', 'keys', 'kg', 'kh', 'ki', 'kind', 'km', 'kn', 'knew', 'know', 'known', 'knows', 'kp', 'kr', 'kw', 'ky', 'kz', 'l', 'la', 'large', 'largely', 'last', 'lately', 'later', 'latest', 'latter', 'latterly', 'lb', 'lc', 'least', 'length', 'less', 'lest', 'let', "let's", 'lets', 'li', 'like', 'liked', 'likely', 'likewise', 'line', 'little', 'lk', 'll', 'long', 'longer', 'longest', 'look', 'looking', 'looks', 'low', 'lower', 'lr', 'ls', 'lt', 'ltd', 'lu', 'lv', 'ly', 'm', 'ma', 'made', 'mainly', 'make', 'makes', 'making', 'man', 'many', 'may', 'maybe', "mayn't", 'maynt', 'mc', 'md', 'me', 'mean', 'means', 'meantime', 'meanwhile', 'member', 'members', 'men', 'merely', 'mg', 'mh', 'microsoft', 'might', "might've", "mightn't", 'mightnt', 'mil', 'mill', 'million', 'mine', 'minus', 'miss', 'mk', 'ml', 'mm', 'mn', 'mo', 'more', 'moreover', 'most', 'mostly', 'move', 'mp', 'mq', 'mr', 'mrs', 'ms', 'msie', 'mt', 'mu', 'much', 'mug', 'must', "must've", "mustn't", 'mustnt', 'mv', 'mw', 'mx', 'my', 'myself', 'myse”', 'mz', 'n', 'na', 'name', 'namely', 'nay', 'nc', 'nd', 'ne', 'near', 'nearly', 'necessarily', 'necessary', 'need', 'needed', 'needing', "needn't", 'neednt', 'needs', 'neither', 'net', 'netscape', 'never', 'neverf', 'neverless', 'nevertheless', 'new', 'newer', 'newest', 'next', 'nf', 'ng', 'ni', 'nine', 'ninety', 'nl', 'no', 'no-one', 'nobody', 'non', 'none', 'nonetheless', 'noone', 'nor', 'normally', 'nos', 'not', 'noted', 'nothing', 'notwithstanding', 'novel', 'now', 'nowhere', 'np', 'nr', 'nu', 'null', 'number', 'numbers', 'nz', 'o', 'obtain', 'obtained', 'obviously', 'of', 'off', 'often', 'o

```
h', 'ok', 'okay', 'old', 'older', 'oldest', 'om', 'omitted', 'on', 'once', 'one', "one's",
'ones', 'only', 'onto', 'open', 'opened', 'opening', 'opens', 'opposite', 'or', 'ord', 'or
der', 'ordered', 'ordering', 'orders', 'org', 'other', 'others', 'otherwise', 'ought', "ou
ghtn't", 'oughtnt', 'our', 'ours', 'ourselves', 'out', 'outside', 'over', 'overall', 'owin
g', 'own', 'p', 'pa', 'page', 'pages', 'part', 'parted', 'particular', 'particularly', 'pa
rting', 'parts', 'past', 'pe', 'per', 'perhaps', 'pf', 'pg', 'ph', 'pk', 'pl', 'place', 'p
laced', 'places', 'please', 'plus', 'pm', 'pmid', 'pn', 'point', 'pointed', 'pointing', 'p
oints', 'poorly', 'possible', 'possibly', 'potentially', 'pp', 'pr', 'predominantly', 'pre
sent', 'presented', 'presenting', 'presents', 'presumably', 'previously', 'primarily', 'pr
obably', 'problem', 'problems', 'promptly', 'proud', 'provided', 'provides', 'pt', 'put',
'puts', 'pw', 'py', 'q', 'qa', 'que', 'quickly', 'quite', 'qv', 'r', 'ran', 'rather', 'r
d', 're', 'readily', 'really', 'reasonably', 'recent', 'recently', 'ref', 'refs', 'regardi
ng', 'regardless', 'regards', 'related', 'relatively', 'research', 'reserved', 'respective
ly', 'resulted', 'resulting', 'results', 'right', 'ring', 'ro', 'room', 'rooms', 'round',
'ru', 'run', 'rw', 's', 'sa', 'said', 'same', 'saw', 'say', 'saying', 'says', 'sb', 'sc',
'sd', 'se', 'sec', 'second', 'secondly', 'seconds', 'section', 'see', 'seeing', 'seem', 's
eemed', 'seeming', 'seems', 'seen', 'sees', 'self', 'selves', 'sensible', 'sent', 'seriou
s', 'seriously', 'seven', 'seventy', 'several', 'sg', 'sh', 'shall', "shan't", 'shant', 's
he', "she'd", "she'll", "she's", 'shed', 'shell', 'shes', 'should', "should've", 'should
n', "shouldn't", 'shouldnt', 'show', 'showed', 'showing', 'shown', 'showns', 'shows', 's
i', 'side', 'sides', 'significant', 'significantly', 'similar', 'similarly', 'since', 'sin
cere', 'site', 'six', 'sixty', 'sj', 'sk', 'sl', 'slightly', 'sm', 'small', 'smaller', 'sm
allest', 'sn', 'so', 'some', 'somebody', 'someday', 'somehow', 'someone', 'somethan', 'som
ething', 'sometime', 'sometimes', 'somewhat', 'somewhere', 'soon', 'sorry', 'specificall
y', 'specified', 'specify', 'specifying', 'sr', 'st', 'state', 'states', 'still', 'stop',
'strongly', 'su', 'sub', 'substantially', 'successfully', 'such', 'sufficiently', 'sugges
t', 'sup', 'sure', 'sv', 'sy', 'system', 'sz', 't', "t's", 'take', 'taken', 'taking', 't
c', 'td', 'tell', 'ten', 'tends', 'test', 'text', 'tf', 'tg', 'th', 'than', 'thank', 'than
ks', 'thanx', 'that', "that'll", "that's", "that've", 'thatll', 'thats', 'thatve', 'the',
'their', 'theirs', 'them', 'themselves', 'then', 'thence', 'there', "there'd", "there'll",
"there're", "there's", "there've", 'thereafter', 'thereby', 'thered', 'therefore', 'therei
n', 'therell', 'thereof', 'therere', 'theres', 'thereto', 'thereupon', 'thereve', 'these',
'they', "they'd", "they'll", "they're", "they've", 'theyd', 'theyll', 'theyre', 'theyve',
'thick', 'thin', 'thing', 'things', 'think', 'thinks', 'third', 'thirty', 'this', 'thoroug
h', 'thoroughly', 'those', 'thou', 'though', 'thoughh', 'thought', 'thoughts', 'thousand',
'three', 'throug', 'through', 'throughout', 'thru', 'thus', 'til', 'till', 'tip', 'tis',
'tj', 'tk', 'tm', 'tn', 'to', 'today', 'together', 'too', 'took', 'top', 'toward', 'toward
s', 'tp', 'tr', 'tried', 'tries', 'trillion', 'truly', 'try', 'trying', 'ts', 'tt', 'tur
n', 'turned', 'turning', 'turns', 'tv', 'tw', 'twas', 'twelve', 'twenty', 'twice', 'two',
'tz', 'u', 'ua', 'ug', 'uk', 'um', 'un', 'under', 'underneath', 'undoing', 'unfortunatel
y', 'unless', 'unlike', 'unlikely', 'until', 'unto', 'up', 'upon', 'ups', 'upwards', 'us',
'use', 'used', 'useful', 'usefully', 'usefulness', 'uses', 'using', 'usually', 'uucp', 'u
y', 'uz', 'v', 'va', 'value', 'various', 'vc', 've', 'versus', 'very', 'vg', 'vi', 'via',
'viz', 'vn', 'vol', 'vols', 'vs', 'vu', 'w', 'want', 'wanted', 'wanting', 'wants', 'was',
'wasn', "wasn't", 'wasnt', 'way', 'ways', 'we', "we'd", "we'll", "we're", "we've", 'web',
'webpage', 'website', 'wed', 'welcome', 'well', 'wells', 'went', 'were', 'weren', "were
n't", 'werent', 'weve', 'wf', 'what', "what'd", "what'll", "what's", "what've", 'whateve
r', 'whatll', 'whats', 'whatve', 'when', "when'd", "when'll", "when's", 'whence', 'wheneve
r', 'where', "where'd", "where'll", "where's", 'whereafter', 'whereas', 'whereby', 'wherei
n', 'wheres', 'whereupon', 'wherever', 'whether', 'which', 'whichever', 'while', 'whilst',
'whim', 'whither', 'who', "who'd", "who'll", "who's", 'whod', 'whoever', 'whole', 'wholl',
'whom', 'whomever', 'whos', 'whose', 'why', "why'd", "why'll", "why's", 'widely', 'width',
'will', 'willing', 'wish', 'with', 'within', 'without', 'won', "won't", 'wonder', 'wont',
'words', 'work', 'worked', 'working', 'works', 'world', 'would', "would've", 'wouldn', "wo
uldn't", 'wouldnt', 'ws', 'www', 'x', 'y', 'ye', 'year', 'years', 'yes', 'yet', 'you', "yo
u'd", "you'll", "you're", "you've", 'youd', 'youll', 'young', 'younger', 'youngest', 'you
r', 'youre', 'yours', 'yourself', 'yourselves', 'youve', 'yt', 'yu', 'z', 'za', 'zero', 'z
m', 'zr']

Non-Stopwords: ['Natural', 'language', 'processing', '(NLP)', 'subfield', 'artificial', 'i
ntelligence', 'content', 'improve', 'efficiency', 'algorithms', 'highlight', 'informatio
n.', 'create', 'list', 'stopwords', 'filter', 'sample', 'paragraph.']
```

In [ ]:

Lab Assignment - 2

Textblob

In [1]:
```
import nltk
# nltk.download('punkt')
```

In [2]:
```
nltk.download('averaged_perceptron_tagger')
```

```
[nltk_data] Downloading package averaged_perceptron_tagger to
[nltk_data]     C:\Users\raval\AppData\Roaming\nltk_data...
[nltk_data]   Package averaged_perceptron_tagger is already up-to-
[nltk_data]       date!
```

Out[2]:
```
True
```

In [3]:
```
from textblob import TextBlob
```

In [4]:
```
text = '''
The titular threat of The Blob has always struck me as the ultimate movie
monster: an insatiably hungry, amoeba-like mass able to penetrate
virtually any safeguard, capable of--as a doomed doctor chillingly
describes it--"assimilating flesh on contact.
Snide comparisons to gelatin be damned, it's a concept with the most
devastating of potential consequences, not unlike the grey goo scenario
proposed by technological theorists fearful of
artificial intelligence run rampant.
'''
```

- pos tagging

In [5]:
```
blob = TextBlob(text)
blob.tags
```

Out[5]:
```
[('The', 'DT'),
 ('titular', 'JJ'),
 ('threat', 'NN'),
 ('of', 'IN'),
 ('The', 'DT'),
 ('Blob', 'NNP'),
 ('has', 'VBZ'),
 ('always', 'RB'),
 ('struck', 'VBN'),
 ('me', 'PRP'),
 ('as', 'IN'),
 ('the', 'DT'),
 ('ultimate', 'JJ'),
 ('movie', 'NN'),
 ('monster', 'NN'),
 ('an', 'DT'),
 ('insatiably', 'RB'),
 ('hungry', 'JJ'),
 ('amoeba-like', 'JJ'),
 ('mass', 'NN'),
 ('able', 'JJ'),
 ('to', 'TO'),
 ('penetrate', 'VB'),
 ('virtually', 'RB'),
 ('any', 'DT'),
 ('safeguard', 'NN'),
 ('capable', 'JJ'),
 ('of', 'IN'),
 ('as', 'IN'),
 ('a', 'DT'),
 ('doomed', 'JJ'),
 ('doctor', 'NN'),
 ('chillingly', 'RB'),
 ('describes', 'VBZ'),
 ('it', 'PRP'),
 ('assimilating', 'VBG'),
 ('flesh', 'NN'),
 ('on', 'IN'),
 ('contact', 'NN'),
 ('Snide', 'JJ'),
 ('comparisons', 'NNS'),
 ('to', 'TO'),
 ('gelatin', 'VB'),
 ('be', 'VB'),
 ('damned', 'VBN'),
 ('it', 'PRP'),
 ("'s", 'VBZ'),
 ('a', 'DT'),
 ('concept', 'NN'),
 ('with', 'IN'),
 ('the', 'DT'),
 ('most', 'RBS'),
 ('devastating', 'JJ'),
 ('of', 'IN'),
 ('potential', 'JJ'),
 ('consequences', 'NNS'),
 ('not', 'RB'),
 ('unlike', 'IN'),
 ('the', 'DT'),
 ('grey', 'NN'),
 ('goo', 'NN'),
 ('scenario', 'NN'),
 ('proposed', 'VBN'),
 ('by', 'IN'),
 ('technological', 'JJ'),
 ('theorists', 'NNS'),
 ('fearful', 'NN'),
 ('of', 'IN'),
 ('artificial', 'JJ'),
 ('intelligence', 'NN'),
```

```
 ('run', 'NN'),
 ('rampant', 'NN')]
```

In [6]:
```
nltk.download('brown')
```

```
[nltk_data] Downloading package brown to
[nltk_data]     C:\Users\raval\AppData\Roaming\nltk_data...
[nltk_data]   Package brown is already up-to-date!
```

Out[6]:
```
True
```

- Noun Phrase Extraction

In [7]:
```
blob.noun_phrases
```

Out[7]:
```
WordList(['titular threat', 'blob', 'ultimate movie monster', 'amoeba-like mass', 'snide',
'potential consequences', 'grey goo scenario', 'technological theorists fearful', 'artific
ial intelligence run rampant'])
```

- sentiment

In [8]:
```
for sentence in blob.sentences:
    print(sentence.sentiment.polarity)
```

```
0.06000000000000001
-0.34166666666666673
```

In [9]:
```
testimonial = TextBlob("Textblob is amazingly simple to use. What great fun!")
testimonial.sentiment
```

Out[9]:
```
Sentiment(polarity=0.39166666666666666, subjectivity=0.4357142857142857)
```

In [10]:
```
testimonial = TextBlob("This computer is good , although it is very costly !")
testimonial.sentiment
```

Out[10]:
```
Sentiment(polarity=0.475, subjectivity=0.45000000000000007)
```

In [11]:
```
testimonial = TextBlob("This computer is very bad , although it is very costly !")
testimonial.sentiment
```

Out[11]:
```
Sentiment(polarity=-0.3299999999999999, subjectivity=0.5833333333333334)
```

- tokanization

In [12]:
```
testimonial.words
```

Out[12]:
```
WordList(['This', 'computer', 'is', 'very', 'bad', 'although', 'it', 'is', 'very', 'costl
y'])
```

- Get Word and Noun Phrase Frequencies

In [13]:
```
testimonial.word_counts['computer']
```

Out[13]:
```
1
```

In [14]:
```
testimonial.words.count('computer', case_sensitive=True)
```

Out[14]:
```
1
```

- n grams

In [15]:
```
testimonial.ngrams(n=3)
```

Out[15]:
```
[WordList(['This', 'computer', 'is']),
 WordList(['computer', 'is', 'very']),
 WordList(['is', 'very', 'bad']),
 WordList(['very', 'bad', 'although']),
 WordList(['bad', 'although', 'it']),
 WordList(['although', 'it', 'is']),
 WordList(['it', 'is', 'very']),
 WordList(['is', 'very', 'costly'])]
```

NLTK

In [16]:
```
# import nltk
```

In [17]:
```
sentence = """At eight o'clock on Thursday morning Arthur didn't feel very good."""
```

- tokenization

In [18]:
```
tokens = nltk.word_tokenize(sentence)
tokens
```

Out[18]:
```
['At',
 'eight',
 "o'clock",
 'on',
 'Thursday',
 'morning',
 'Arthur',
 'did',
 "n't",
 'feel',
 'very',
 'good',
 '.']
```

- pos tagging

In [19]:
```
tagged = nltk.pos_tag(tokens)
tagged
```

Out[19]:
```
[('At', 'IN'),
 ('eight', 'CD'),
 ("o'clock", 'NN'),
 ('on', 'IN'),
 ('Thursday', 'NNP'),
 ('morning', 'NN'),
 ('Arthur', 'NNP'),
 ('did', 'VBD'),
 ("n't", 'RB'),
 ('feel', 'VB'),
 ('very', 'RB'),
 ('good', 'JJ'),
 ('.', '.')]
```

In [20]:
```
# nltk.download('maxent_ne_chunker')
# nltk.download('words')
!pip install svgling
```

```
Requirement already satisfied: svgling in c:\python311\lib\site-packages (0.4.0)
Requirement already satisfied: svgwrite in c:\python311\lib\site-packages (from svgling)
(1.4.3)


[notice] A new release of pip is available: 23.0.1 -> 23.3.2
[notice] To update, run: python.exe -m pip install --upgrade pip
```

In [21]:
```
entities = nltk.chunk.ne_chunk(tagged)
entities
```

Out[21]:
```
Tree('S', [('At', 'IN'), ('eight', 'CD'), ("o'clock", 'NN'), ('on', 'IN'), ('Thursday', 'N
NP'), ('morning', 'NN'), Tree('PERSON', [('Arthur', 'NNP')]), ('did', 'VBD'), ("n't", 'R
B'), ('feel', 'VB'), ('very', 'RB'), ('good', 'JJ'), ('.', '.')])
```

In [22]:
```
from nltk.corpus import treebank
```

In [23]:
```
nltk.download('treebank')
```

```
[nltk_data] Downloading package treebank to
[nltk_data]     C:\Users\raval\AppData\Roaming\nltk_data...
[nltk_data]   Package treebank is already up-to-date!
```

Out[23]:
```
True
```

In [24]:
```
# import nltk
# from nltk.corpus import treebank

# # Load the Penn Treebank data
# nltk.download('treebank')

# # Get a parsed sentence from the treebank
# t = treebank.parsed_sents('wsj_0001.mrg')[0]

# # Draw the parse tree
# t.draw()
98+31
```

Out[24]:
```
129
```

In [28]:
```
# t = treebank.parsed_sents('wsj_0001.mrg')[0]
# t.draw(max_depth=2)  # Adjust the depth as needed
```

In [26]:
```
t = treebank.parsed_sents('wsj_0001.mrg')[0]
print(t)
```

```
(S
  (NP-SBJ
    (NP (NNP Pierre) (NNP Vinken))
    (, ,)
    (ADJP (NP (CD 61) (NNS years)) (JJ old))
    (, ,))
  (VP
    (MD will)
    (VP
      (VB join)
      (NP (DT the) (NN board))
      (PP-CLR (IN as) (NP (DT a) (JJ nonexecutive) (NN director)))
      (NP-TMP (NNP Nov.) (CD 29))))
  (. .))
```

In [29]:
```
!pip install spacy
```

```
Collecting spacy
  Downloading spacy-3.7.2-cp311-cp311-win_amd64.whl (12.1 MB)
     ---------------------------------------- 0.0/12.1 MB ? eta -:--:--
     ---------------------------------------- 0.0/12.1 MB ? eta -:--:--
     ---------------------------------------- 0.0/12.1 MB 660.6 kB/s eta 0:00:19
     ---------------------------------------- 0.1/12.1 MB 375.8 kB/s eta 0:00:32
     ---------------------------------------- 0.1/12.1 MB 469.7 kB/s eta 0:00:26
     ---------------------------------------- 0.1/12.1 MB 403.5 kB/s eta 0:00:30
      --------------------------------------- 0.2/12.1 MB 787.7 kB/s eta 0:00:16
      --------------------------------------- 0.3/12.1 MB 827.5 kB/s eta 0:00:15
     - -------------------------------------- 0.3/12.1 MB 1.0 MB/s eta 0:00:12
     - -------------------------------------- 0.4/12.1 MB 998.3 kB/s eta 0:00:12
     - -------------------------------------- 0.4/12.1 MB 998.3 kB/s eta 0:00:12
     - -------------------------------------- 0.4/12.1 MB 998.3 kB/s eta 0:00:12
     - -------------------------------------- 0.4/12.1 MB 998.3 kB/s eta 0:00:12
     - -------------------------------------- 0.4/12.1 MB 726.4 kB/s eta 0:00:17
     - -------------------------------------- 0.5/12.1 MB 819.2 kB/s eta 0:00:15
     -- ------------------------------------- 0.6/12.1 MB 922.8 kB/s eta 0:00:13
     -- ------------------------------------- 0.7/12.1 MB 982.7 kB/s eta 0:00:12
```

In [32]:
```
!python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.7.1
  Downloading https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-any.whl (https://github.com/explosion/spacy-models/releases/download/en_core_web_sm-3.7.1/en_core_web_sm-3.7.1-py3-none-any.whl) (12.8 MB)
     ---------------------------------------- 0.0/12.8 MB ? eta -:--:--
     ---------------------------------------- 0.0/12.8 MB 682.7 kB/s eta 0:00:19
     ---------------------------------------- 0.1/12.8 MB 525.1 kB/s eta 0:00:25
     ---------------------------------------- 0.1/12.8 MB 657.6 kB/s eta 0:00:20
     ---------------------------------------- 0.1/12.8 MB 535.8 kB/s eta 0:00:24
     ---------------------------------------- 0.1/12.8 MB 602.4 kB/s eta 0:00:22
     ---------------------------------------- 0.2/12.8 MB 579.6 kB/s eta 0:00:22
     ---------------------------------------- 0.2/12.8 MB 579.6 kB/s eta 0:00:22
      --------------------------------------- 0.2/12.8 MB 628.1 kB/s eta 0:00:20
      --------------------------------------- 0.3/12.8 MB 682.7 kB/s eta 0:00:19
     - -------------------------------------- 0.3/12.8 MB 720.5 kB/s eta 0:00:18
     - -------------------------------------- 0.4/12.8 MB 839.7 kB/s eta 0:00:15
     - -------------------------------------- 0.5/12.8 MB 829.2 kB/s eta 0:00:15
     - -------------------------------------- 0.5/12.8 MB 814.9 kB/s eta 0:00:16
```

In [33]:
```
import spacy

# Load the English language model
nlp = spacy.load('en_core_web_sm')

# Process a text
doc = nlp("spaCy is a powerful NLP library.")

# Access various annotations
for token in doc:
    print(token.text, token.pos_, token.dep_)
```

```
spaCy INTJ nsubj
is AUX ROOT
a DET det
powerful ADJ amod
NLP PROPN compound
library NOUN attr
. PUNCT punct
```

In [ ]:
```
```

Lab Assignment - 3

In [ ]:

```
from __future__ import division #To avoid integer division
from operator import itemgetter
###Training Phase###

with open("wsj_training.txt", "r") as myfile:
    tr_str = myfile.read()
tr_li = tr_str.split()
num_words_train = len(tr_li)


train_li_words = ['']
train_li_words*= num_words_train

train_li_tags = ['']
train_li_tags*= num_words_train

noun_reduced_list = ['NN','NNS','NNP','NNPS']
verb_reduced_list = ['VB','VBD','VBG','VBN','VBP','VBZ']
adjec_reduced_list = ['JJ', 'JJR', 'JJS']
adv_reduced_list = ['RB', 'RBR', 'RBS']
pronoun_reduced_list = ['PRP', 'PRP$', 'RP']

for i in range(num_words_train):
    temp_li = tr_li[i].split("/")
    train_li_words[i] = temp_li[0]
    if temp_li[1] in noun_reduced_list:
        train_li_tags[i] = 'N'
    elif temp_li[1] in verb_reduced_list:
        train_li_tags[i] = 'V'
    elif temp_li[1] in adjec_reduced_list:
        train_li_tags[i] = 'ADJ'
    elif temp_li[1] in adv_reduced_list:
        train_li_tags[i] = 'ADV'
    elif temp_li[1] in pronoun_reduced_list:
        train_li_tags[i] = 'PRO'
    else:
        train_li_tags[i] = temp_li[1]

k = sorted(list(set(train_li_tags)))
print (k)
dict2_tag_follow_tag_ = {}
"""Nested dictionary to store the transition probabilities
each tag A is a key of the outer dictionary
the inner dictionary is the corresponding value
The inner dictionary's key is the tag B following A
and the corresponding value is the number of times B follows A
"""

dict2_word_tag = {}
"""Nested dictionary to store the emission probabilities.
Each word W is a key of the outer dictionary
The inner dictionary is the corresponding value
The inner dictionary's key is the tag A of the word W
and the corresponding value is the number of times A is a tag of W
"""
```

In [ ]:
```
dict_word_tag_baseline = {}
#Dictionary with word as key and its most frequent tag as value

for i in range(num_words_train-1):
    outer_key = train_li_tags[i]
    inner_key = train_li_tags[i+1]
    dict2_tag_follow_tag_[outer_key]=dict2_tag_follow_tag_.get(outer_key,{})
    dict2_tag_follow_tag_[outer_key][inner_key] = dict2_tag_follow_tag_[outer_key].get(inne
    dict2_tag_follow_tag_[outer_key][inner_key]+=1

    outer_key = train_li_words[i]
    inner_key = train_li_tags[i]
    dict2_word_tag[outer_key]=dict2_word_tag.get(outer_key,{})
    dict2_word_tag[outer_key][inner_key] = dict2_word_tag[outer_key].get(inner_key,0)
    dict2_word_tag[outer_key][inner_key]+=1


"""The 1st token is indicated by being the 1st word of a senetence, that is the word after
Adjusting for the fact that the first word of the document is not accounted for that way
"""

dict2_tag_follow_tag_['.'] = dict2_tag_follow_tag_.get('.',{})
dict2_tag_follow_tag_['.'][train_li_tags[0]] = dict2_tag_follow_tag_['.'].get(train_li_tags
dict2_tag_follow_tag_['.'][train_li_tags[0]]+=1


print (dict2_tag_follow_tag_['IN'])
print (dict2_word_tag['made'])

last_index = num_words_train-1
```

In [ ]:
```
#Accounting for the last word-tag pair
outer_key = train_li_words[last_index]
inner_key = train_li_tags[last_index]
dict2_word_tag[outer_key]=dict2_word_tag.get(outer_key,{})
dict2_word_tag[outer_key][inner_key] = dict2_word_tag[outer_key].get(inner_key,0)
dict2_word_tag[outer_key][inner_key]+=1


"""Converting counts to probabilities in the two nested dictionaries
& also converting the nested dictionaries to outer dictionary with inner sorted lists
"""
for key in dict2_tag_follow_tag_:
    di = dict2_tag_follow_tag_[key]
    s = sum(di.values())
    for innkey in di:
        di[innkey] /= s
    di = di.items()
    di = sorted(di,key=lambda x: x[0])
    dict2_tag_follow_tag_[key] = di

for key in dict2_word_tag:
    di = dict2_word_tag[key]
    dict_word_tag_baseline[key] = max(di, key=di.get)
    s = sum(di.values())
    for innkey in di:
        di[innkey] /= s
    di = di.items()
    di = sorted(di,key=lambda x: x[0])
    dict2_word_tag[key] = di
```

In [ ]:

```
with open("test.txt", "r") as myfile:
    te_str = myfile.read()

te_li = te_str.split()
num_words_test = len(te_li)

test_li_words = ['']
test_li_words*= num_words_test

test_li_tags = ['']
test_li_tags*= num_words_test

output_li = ['']
output_li*= num_words_test

output_li_baseline = ['']
output_li_baseline*= num_words_test

num_errors = 0
num_errors_baseline = 0
```

In [ ]:

```
for i in range(num_words_test):
    temp_li = te_li[i].split("/")
    test_li_words[i] = temp_li[0]
    if temp_li[1] in noun_reduced_list:
        test_li_tags[i] = 'N'
    elif temp_li[1] in verb_reduced_list:
        test_li_tags[i] = 'V'
    elif temp_li[1] in adjec_reduced_list:
        test_li_tags[i] = 'ADJ'
    elif temp_li[1] in adv_reduced_list:
        test_li_tags[i] = 'ADV'
    elif temp_li[1] in pronoun_reduced_list:
        test_li_tags[i] = 'PRO'
    else:
        test_li_tags[i] = temp_li[1]


    output_li_baseline[i] = dict_word_tag_baseline.get(temp_li[0],'')
    #If unknown word - tag = 'N'
    if output_li_baseline[i]=='':
        output_li_baseline[i]='N'


    if output_li_baseline[i]!=test_li_tags[i]:
        num_errors_baseline+=1


    if i==0:    #Accounting for the 1st word in the test document for the Viterbi
        di_transition_probs = dict2_tag_follow_tag_['.']
    else:
        di_transition_probs = dict2_tag_follow_tag_[output_li[i-1]]

    di_emission_probs = dict2_word_tag.get(test_li_words[i],'')

    #If unknown word  - tag = 'N'
    if di_emission_probs=='':
        output_li[i]='N'

    else:
        max_prod_prob = 0
        counter_trans = 0
        counter_emis =0
        prod_prob = 0
        while counter_trans < len(di_transition_probs) and counter_emis < len(di_emission_p
            tag_tr = di_transition_probs[counter_trans][0]
            tag_em = di_emission_probs[counter_emis][0]
            if tag_tr < tag_em:
                counter_trans+=1
            elif tag_tr > tag_em:
                counter_emis+=1
            else:
                prod_prob = di_transition_probs[counter_trans][1] * di_emission_probs[count
                if prod_prob > max_prod_prob:
                    max_prod_prob = prod_prob
                    output_li[i] = tag_tr
                    print ("i=",i," and output=",output_li[i])
                counter_trans+=1
                counter_emis+=1

    if output_li[i]=='': #In case there are no matching entries between the transition tags
        output_li[i] = max(di_emission_probs,key=itemgetter(1))[0]

    if output_li[i]!=test_li_tags[i]:
        num_errors+=1
```

In [1]:
```
print ("Fraction of errors (Baseline) :",(num_errors_baseline/num_words_test))
print ("Fraction of errors (Viterbi):",(num_errors/num_words_test))

print ("Tags suggested by Baseline Algorithm:", output_li_baseline)

print ("Tags suggested by Viterbi Algorithm:", output_li)

print ("Correct tags:",test_li_tags)
```

```
[')', ',', '.', ':', 'ADJ', 'ADV', 'CC', 'CD', 'DT', 'FW', 'IN', 'MD', 'N', 'POS', 'PRO',
'TO', 'V', 'WDT', 'WP', '``']
{'DT': 19, 'N': 17, 'PRO': 3, 'CD': 7, ',': 1, 'ADV': 1, 'ADJ': 3}
{'V': 2}
i= 1  and output= V
i= 2  and output= ADJ
Fraction of errors (Baseline) : 0.0
Fraction of errors (Viterbi): 0.0
Tags suggested by Baseline Algorithm: ['N', 'V', 'ADJ', 'N']
Tags suggested by Viterbi Algorithm: ['N', 'V', 'ADJ', 'N']
Correct tags: ['N', 'V', 'ADJ', 'N']
```

In [ ]:
```
```

In [3]:
```
import pandas as pd
df =  pd.read_csv(r"C:\Users\raval\jupyter_notebook\NLP\words_pos.csv")
df.head()
```

Out[3]:

	Unnamed: 0	word	pos_tag
0	0	aa	NN
1	1	aaa	NN
2	2	aah	NN
3	3	aahed	VBN
4	4	aahing	VBG

In [4]:
```
df = df.drop(["Unnamed: 0"],axis=1)
```

In [13]:
```
tuple(df["word"]),tuple(df["pos_tag"])
```

```
  'aaronic',
  'aaronical',
  'aaronite',
  'aaronitic',
  'aarrgh',
  'aarrghh',
  'aaru',
  'aas',
  'aasvogel',
  'aasvogels',
  'ab',
  'aba',
  'ababdeh',
  'ababua',
  'abac',
  'abaca',
  'abacay',
  'abacas',
  'abacate',
```

In [16]:
```
# Initialize lists as empty lists
train_li_words = []
train_li_tags = []

# Define reduced tag lists
noun_reduced_list = ['NN', 'NNS', 'NNP', 'NNPS']
verb_reduced_list = ['VB', 'VBD', 'VBG', 'VBN', 'VBP', 'VBZ']
adjec_reduced_list = ['JJ', 'JJR', 'JJS']
adv_reduced_list = ['RB', 'RBR', 'RBS']
pronoun_reduced_list = ['PRP', 'PRP$', 'RP']

# Assuming df is a DataFrame containing word and pos_tag columns
for index, row in df.iterrows():
    train_li_words.append(row["word"])
    if row["pos_tag"] in noun_reduced_list:
        train_li_tags.append('N')
    elif row["pos_tag"] in verb_reduced_list:
        train_li_tags.append('V')
    elif row["pos_tag"] in adjec_reduced_list:
        train_li_tags.append('ADJ')
    elif row["pos_tag"] in adv_reduced_list:
        train_li_tags.append('ADV')
    elif row["pos_tag"] in pronoun_reduced_list:
        train_li_tags.append('PRO')
    else:
        train_li_tags.append(row["pos_tag"])

# Training phase
dict2_tag_follow_tag_ = {}
dict2_word_tag = {}
dict_word_tag_baseline = {}

for i in range(len(train_li_words) - 1):
    outer_key = train_li_tags[i]
    inner_key = train_li_tags[i + 1]
    dict2_tag_follow_tag_.setdefault(outer_key, {}).setdefault(inner_key, 0)
    dict2_tag_follow_tag_[outer_key][inner_key] += 1

    outer_key = train_li_words[i]
    inner_key = train_li_tags[i]
    dict2_word_tag.setdefault(outer_key, {}).setdefault(inner_key, 0)
    dict2_word_tag[outer_key][inner_key] += 1

    dict_word_tag_baseline.setdefault(outer_key, inner_key)

dict2_tag_follow_tag_['.'] = {train_li_tags[0]: 1}  # Accounting for the first word

# Converting counts to probabilities
for key in dict2_tag_follow_tag_:
    di = dict2_tag_follow_tag_[key]
    s = sum(di.values())
    for innkey in di:
        di[innkey] /= s
    di = di.items()
    di = sorted(di, key=lambda x: x[0])
    dict2_tag_follow_tag_[key] = di

for key in dict2_word_tag:
    di = dict2_word_tag[key]
    s = sum(di.values())
    for innkey in di:
        di[innkey] /= s
    di = di.items()
    di = sorted(di, key=lambda x: x[0])
    dict2_word_tag[key] = di

# Testing phase
with open("test.txt", "r") as myfile:
    te_str = myfile.read()

te_li = te_str.split()
num_words_test = len(te_li)
```

```
test_li_words = []
test_li_tags = []
output_li = []
output_li_baseline = []
num_errors = 0
num_errors_baseline = 0

for i in range(num_words_test):
    temp_li = te_li[i].split("/")
    test_li_words.append(temp_li[0])

    if temp_li[1] in noun_reduced_list:
        test_li_tags.append('N')
    elif temp_li[1] in verb_reduced_list:
        test_li_tags.append('V')
    elif temp_li[1] in adjec_reduced_list:
        test_li_tags.append('ADJ')
    elif temp_li[1] in adv_reduced_list:
        test_li_tags.append('ADV')
    elif temp_li[1] in pronoun_reduced_list:
        test_li_tags.append('PRO')
    else:
        test_li_tags.append(temp_li[1])

    output_li_baseline.append(dict_word_tag_baseline.get(temp_li[0], 'N'))

    if output_li_baseline[i] != test_li_tags[i]:
        num_errors_baseline += 1

    if i == 0:
        di_transition_probs = dict2_tag_follow_tag_['.']
    else:
        di_transition_probs = dict2_tag_follow_tag_[output_li[i - 1]]

    di_emission_probs = dict2_word_tag.get(test_li_words[i], '')

    if di_emission_probs == '':
        output_li.append('N')
    else:
        max_prod_prob = 0
        best_tag = 'N'
        for tag_tr, prob_tr in di_transition_probs:
            if tag_tr in di_emission_probs:
                prob_em = di_emission_probs[tag_tr]
                prod_prob = prob_tr * prob_em
                if prod_prob > max_prod_prob:
                    max_prod_prob = prod_prob
                    best_tag = tag_tr
        output_li.append(best_tag)

    if output_li[i] != test_li_tags[i]:
        num_errors += 1

print("Fraction of errors (Baseline):", (num_errors_baseline / num_words_test))
print("Fraction of errors (Viterbi):", (num_errors / num_words_test))

print("Tags suggested by Baseline Algorithm:", output_li_baseline)
print("Tags suggested by Viterbi Algorithm:", output_li)
print("Correct tags:", test_li_tags)
```

```
Fraction of errors (Baseline): 0.0
Fraction of errors (Viterbi): 0.5
Tags suggested by Baseline Algorithm: ['N', 'V', 'ADJ', 'N']
Tags suggested by Viterbi Algorithm: ['N', 'N', 'N', 'N']
Correct tags: ['N', 'V', 'ADJ', 'N']
```

trying to improve viterbi

In [20]:
```
# Testing phase
with open("test.txt", "r") as myfile:
    te_str = myfile.read()

te_li = te_str.split()
num_words_test = len(te_li)

test_li_words = []
test_li_tags = []
output_li = []
output_li_baseline = []
num_errors = 0
num_errors_baseline = 0

# Smoothing parameter for Laplace smoothing
alpha = 0.01

for i in range(num_words_test):
    temp_li = te_li[i].split("/")
    test_word = temp_li[0]

    test_tag = None
    if temp_li[1] in noun_reduced_list:
        test_tag = 'N'
    elif temp_li[1] in verb_reduced_list:
        test_tag = 'V'
    elif temp_li[1] in adjec_reduced_list:
        test_tag = 'ADJ'
    elif temp_li[1] in adv_reduced_list:
        test_tag = 'ADV'
    elif temp_li[1] in pronoun_reduced_list:
        test_tag = 'PRO'
    else:
        test_tag = temp_li[1]

    test_li_words.append(test_word)
    test_li_tags.append(test_tag)

    output_li_baseline.append(dict_word_tag_baseline.get(test_word, 'N'))

    if output_li_baseline[i] != test_tag:
        num_errors_baseline += 1

    if i == 0:
        di_transition_probs = dict2_tag_follow_tag_['.']
    else:
        # Expand context window to consider multiple previous tags
        prev_tags = output_li[max(0, i - 3):i]
        tag_set = set(tag for sublist in [dict2_tag_follow_tag_.get(prev_tag, []) for prev_
        di_transition_probs = {tag: alpha for tag in tag_set}
        for prev_tag in prev_tags:
            if prev_tag in dict2_tag_follow_tag_:
                for next_tag, prob in dict2_tag_follow_tag_[prev_tag]:
                    di_transition_probs[next_tag] += prob

    di_emission_probs = dict2_word_tag.get(test_word, {})

    if not di_emission_probs:  # Unknown word
        output_li.append('N')
        if 'N' != test_tag:  # Increment error count if 'N' is not the correct tag
            num_errors += 1
    else:
        max_prod_prob = 0
        best_tag = 'N'
        for tag_tr, prob_tr in di_transition_probs.items():
            if tag_tr in di_emission_probs:
                prob_em = di_emission_probs[tag_tr]
                prod_prob = prob_tr * prob_em
                if prod_prob > max_prod_prob:
                    max_prod_prob = prod_prob
                    best_tag = tag_tr
        output_li.append(best_tag)
```

```
    if output_li[i] != test_tag:
        num_errors += 1

print("Fraction of errors (Baseline):", (num_errors_baseline / num_words_test))
print("Fraction of errors (Viterbi):", (num_errors / num_words_test))

print("Tags suggested by Baseline Algorithm:", output_li_baseline)
print("Tags suggested by Viterbi Algorithm:", output_li)
print("Correct tags:", test_li_tags)
```

```
Fraction of errors (Baseline): 0.0
Fraction of errors (Viterbi): 0.5
Tags suggested by Baseline Algorithm: ['N', 'V', 'ADJ', 'N']
Tags suggested by Viterbi Algorithm: ['N', 'N', 'N', 'N']
Correct tags: ['N', 'V', 'ADJ', 'N']
```

In [ ]:

In [20]:
```
from __future__ import division #To avoid integer division
from operator import itemgetter
```

In [21]:
```
with open("words_pos.csv", "r") as myfile:
    tr_str = myfile.read()
tr_li = tr_str.split()
num_words_train = len(tr_li)
```

In [22]:
```
train_li_words = ['']
train_li_words*= num_words_train

train_li_tags = ['']
train_li_tags*= num_words_train
```

In [23]:
```
noun_reduced_list = ['NN','NNS','NNP','NNPS']
verb_reduced_list = ['VB','VBD','VBG','VBN','VBP','VBZ']
adjec_reduced_list = ['JJ', 'JJR', 'JJS']
adv_reduced_list = ['RB', 'RBR', 'RBS']
pronoun_reduced_list = ['PRP', 'PRP$', 'RP']
```

In [24]:
```
for i in range(num_words_train):
    temp_li = tr_li[i]
    train_li_words[i] = temp_li[0]
    if temp_li[1] in noun_reduced_list:
        train_li_tags[i] = 'N'
    elif temp_li[1] in verb_reduced_list:
        train_li_tags[i] = 'V'
    elif temp_li[1] in adjec_reduced_list:
        train_li_tags[i] = 'ADJ'
    elif temp_li[1] in adv_reduced_list:
        train_li_tags[i] = 'ADV'
    elif temp_li[1] in pronoun_reduced_list:
        train_li_tags[i] = 'PRO'
    else:
        train_li_tags[i] = temp_li[1]
```

In [7]:
```
k = sorted(list(set(train_li_tags)))
print(k)
dict2_tag_follow_tag_ = {}
"""Nested dictionary to store the transition probabilities
each tag A is a key of the outer dictionary
the inner dictionary is the corresponding value
The inner dictionary's key is the tag B following A
and the corresponding value is the number of times B follows A
"""

dict2_word_tag = {}
"""Nested dictionary to store the emission probabilities.
Each word W is a key of the outer dictionary
The inner dictionary is the corresponding value
The inner dictionary's key is the tag A of the word W
and the corresponding value is the number of times A is a tag of W
"""

dict_word_tag_baseline = {}
#Dictionary with word as key and its most frequent tag as value
```

```
[',', '0', '1', '2', '3', '4', '5', '6', '7', '8', '9', 'w']
```

In [8]:
```
for i in range(num_words_train-1):
    outer_key = train_li_tags[i]
    inner_key = train_li_tags[i+1]
    dict2_tag_follow_tag_[outer_key]=dict2_tag_follow_tag_.get(outer_key,{})
    dict2_tag_follow_tag_[outer_key][inner_key] = dict2_tag_follow_tag_[outer_key].get(inne
    dict2_tag_follow_tag_[outer_key][inner_key]+=1

    outer_key = train_li_words[i]
    inner_key = train_li_tags[i]
    dict2_word_tag[outer_key]=dict2_word_tag.get(outer_key,{})
    dict2_word_tag[outer_key][inner_key] = dict2_word_tag[outer_key].get(inner_key,0)
    dict2_word_tag[outer_key][inner_key]+=1
```

In [9]:
```
# Check if 'NN' key exists in dict2_tag_follow_tag_
if 'NN' in dict2_tag_follow_tag_:
    print(dict2_tag_follow_tag_['NN'])
else:
    print("Key 'NN' not found in dict2_tag_follow_tag_")

# Check if 'Rudolph' key exists in dict2_word_tag
if 'Rudolph' in dict2_word_tag:
    print(dict2_word_tag['Rudolph'])
else:
    print("Key 'Rudolph' not found in dict2_word_tag")
```

```
Key 'NN' not found in dict2_tag_follow_tag_
Key 'Rudolph' not found in dict2_word_tag
```

In [10]:
```
"""Converting counts to probabilities in the two nested dictionaries
& also converting the nested dictionaries to outer dictionary with inner sorted lists
"""
for key in dict2_tag_follow_tag_:
    di = dict2_tag_follow_tag_[key]
    s = sum(di.values())
    for innkey in di:
        di[innkey] /= s
    di = di.items()
    di = sorted(di,key=lambda x: x[0])
    dict2_tag_follow_tag_[key] = di

for key in dict2_word_tag:
    di = dict2_word_tag[key]
    dict_word_tag_baseline[key] = max(di, key=di.get)
    s = sum(di.values())
    for innkey in di:
        di[innkey] /= s
    di = di.items()
    di = sorted(di,key=lambda x: x[0])
    dict2_word_tag[key] = di
```

In [11]:

```
###Testing Phase###
with open("output.txt", "r") as myfile:
    te_str = myfile.read()

te_li = te_str.split()
num_words_test = len(te_li)

test_li_words = ['']
test_li_words*= num_words_test

test_li_tags = ['']
test_li_tags*= num_words_test

output_li = ['']
output_li*= num_words_test

output_li_baseline = ['']
output_li_baseline*= num_words_test

num_errors = 0
num_errors_baseline = 0
```

In [12]:

```
for i in range(num_words_test):
    temp_li = te_li[i].split("/")
    test_li_words[i] = temp_li[0]
    if temp_li[1] in noun_reduced_list:
        test_li_tags[i] = 'N'
    elif temp_li[1] in verb_reduced_list:
        test_li_tags[i] = 'V'
    elif temp_li[1] in adjec_reduced_list:
        test_li_tags[i] = 'ADJ'
    elif temp_li[1] in adv_reduced_list:
        test_li_tags[i] = 'ADV'
    elif temp_li[1] in pronoun_reduced_list:
        test_li_tags[i] = 'PRO'
    else:
        test_li_tags[i] = temp_li[1]

    output_li_baseline[i] = dict_word_tag_baseline.get(temp_li[0], '')
    # If unknown word - tag = 'N'
    if output_li_baseline[i] == '':
        output_li_baseline[i] = 'N'

    if output_li_baseline[i] != test_li_tags[i]:
        num_errors_baseline += 1

    if i == 0:  # Accounting for the 1st word in the test document for the Viterbi
        di_transition_probs = dict2_tag_follow_tag_.get('.', [])  # Get value for key '.' o
    else:
        di_transition_probs = dict2_tag_follow_tag_.get(output_li[i - 1], [])

    di_emission_probs = dict2_word_tag.get(test_li_words[i], '')

    # If unknown word - tag = 'N'
    if di_emission_probs == '':
        output_li[i] = 'N'
    else:
        max_prod_prob = 0
        counter_trans = 0
        counter_emis = 0
        prod_prob = 0
        while counter_trans < len(di_transition_probs) and counter_emis < len(di_emission_p
            tag_tr = di_transition_probs[counter_trans][0]
            tag_em = di_emission_probs[counter_emis][0]
            if tag_tr < tag_em:
                counter_trans += 1
            elif tag_tr > tag_em:
                counter_emis += 1
            else:
                prod_prob = di_transition_probs[counter_trans][1] * di_emission_probs[count
                if prod_prob > max_prod_prob:
                    max_prod_prob = prod_prob
                    output_li[i] = tag_tr
                    print("i=", i, " and output=", output_li[i])
                counter_trans += 1
                counter_emis += 1

        if output_li[i] == '':  # In case there are no matching entries between the transit
            output_li[i] = max(di_emission_probs, key=itemgetter(1))[0]

    if output_li[i] != test_li_tags[i]:
        num_errors += 1
```

In [13]:
```
print("Fraction of errors (Baseline) :",(num_errors_baseline/num_words_test))
print("Fraction of errors (Viterbi):",(num_errors/num_words_test))

print("Tags suggested by Baseline Algorithm:", output_li_baseline)

print("Tags suggested by Viterbi Algorithm:", output_li)

print("Correct tags:",test_li_tags)
```

```
Fraction of errors (Baseline) : 0.7303252885624344
Fraction of errors (Viterbi): 0.7303252885624344
Tags suggested by Baseline Algorithm: ['N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N',
'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'w', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N', 'N',
```

In [ ]:

In [ ]:

In [ ]:

Lab Assignment - 4

In [ ]:
```
import nltk
from nltk.corpus import stopwords
stopwords=set(stopwords.words('english'))

pos_tweets=[('It is not impossible', 'positive'),
                    ('You are my lovely friend', 'Positive'),
                    ('She is beautiful girl', 'Positive'),
                    ('He is looking handsome', 'Positive'),
                    ('Exercise is good for health', 'Positive'),
                    ('Today\'s weather is fantastic', 'Positive'),
                    ('I love Mango', 'Positive')]

neg_tweets=[('You are my enemy friend', 'Negative'),
                    ('She is looking ugly ', 'Negative'),
                    ('He is looking horrible', 'Negative'),
                    ('Sleeping more makes you lazy', 'Negative'),
                    ('Today\'s weather is very bad', 'Negative'),
                    ('I hate Banana', 'Negative')]
#print(pos_tweets)
#print(neg_tweets)
```

In [ ]:
```
Senti_tweets=[]
for (words, sentiment) in pos_tweets + neg_tweets:
    words_filtered=[e.lower() for e in words.split() if len(e)>=3]
    Senti_tweets.append((words_filtered, sentiment))
print(Senti_tweets)

def get_words_in_tweets(tweets):
    all_words=[]
    for (words, sentiment) in Senti_tweets:
        all_words.extend(words)
    return (all_words)


def get_word_features(wordlist):
    wordlist=nltk.FreqDist(wordlist)
    word_features=wordlist.keys()
    return word_features
```

In [ ]:
```
word_features=get_word_features(get_words_in_tweets(Senti_tweets))
print(word_features)

word_features_filtered=[]
for w in word_features:
    if w not in stopwords:
        word_features_filtered.append(w)

print(word_features_filtered)
```

In [4]:
```
def extract_features(document):
    document_words=set(document)
    features={}
    for word in word_features_filtered:
        features['contains(%s)' %word] = (word in document_words)
    return features

training_set = nltk.classify.apply_features(extract_features, Senti_tweets)
classifier = nltk.NaiveBayesClassifier.train(training_set)

test_tweet='This is a horrible book'
print("{}: Sentiment={}".format(test_tweet, classifier.classify(extract_features(test_tweet
```

```
[(['not', 'impossible'], 'positive'), (['you', 'are', 'lovely', 'friend'], 'Positive'),
(['she', 'beautiful', 'girl'], 'Positive'), (['looking', 'handsome'], 'Positive'), (['exer
cise', 'good', 'for', 'health'], 'Positive'), (["today's", 'weather', 'fantastic'], 'Posit
ive'), (['love', 'mango'], 'Positive'), (['you', 'are', 'enemy', 'friend'], 'Negative'),
(['she', 'looking', 'ugly'], 'Negative'), (['looking', 'horrible'], 'Negative'), (['sleepi
ng', 'more', 'makes', 'you', 'lazy'], 'Negative'), (["today's", 'weather', 'very', 'bad'],
'Negative'), (['hate', 'banana'], 'Negative')]
dict_keys(['not', 'impossible', 'you', 'are', 'lovely', 'friend', 'she', 'beautiful', 'gir
l', 'looking', 'handsome', 'exercise', 'good', 'for', 'health', "today's", 'weather', 'fan
tastic', 'love', 'mango', 'enemy', 'ugly', 'horrible', 'sleeping', 'more', 'makes', 'laz
y', 'very', 'bad', 'hate', 'banana'])
['impossible', 'lovely', 'friend', 'beautiful', 'girl', 'looking', 'handsome', 'exercise',
'good', 'health', "today's", 'weather', 'fantastic', 'love', 'mango', 'enemy', 'ugly', 'ho
rrible', 'sleeping', 'makes', 'lazy', 'bad', 'hate', 'banana']
This is a horrible book: Sentiment=Negative
```

In [2]:
```
```

```
'nltk.download' is not recognized as an internal or external command,
operable program or batch file.
```

In [3]:
```
>>> import nltk
>>> nltk.download('stopwords')
```

```
[nltk_data] Downloading package stopwords to
[nltk_data]     C:\Users\raval\AppData\Roaming\nltk_data...
[nltk_data]   Unzipping corpora\stopwords.zip.
```

Out[3]:
```
True
```

In [13]:
```python
import csv

# Read the CSV file
csv_file_path = r"C:\Users\raval\Downloads\full_training_dataset.csv"

# Assuming your CSV file has columns named 'Sentence' and 'Label'
sentences = []
labels = []

with open(csv_file_path, 'r',encoding='latin-1') as csv_file:
    csv_reader = csv.reader(csv_file)
    next(csv_reader)  # Skip the header row if it exists
    for row in csv_reader:
        sentence = row[0]
        label = row[1]
        sentences.append(sentence)
        labels.append(label)

# Tokenize and filter words
tokenized_sentences = [([e.lower() for e in sentence.split() if len(e) >= 3], label) for se

# Apply the trained classifier
results = []

for sentence, label in tokenized_sentences:
    test_features = extract_features(sentence)
    sentiment = classifier.classify(test_features)
    results.append((sentence, sentiment))

# Print the results
for sentence, sentiment in results:
    print("{}: Sentiment={}".format(sentence, sentiment))
```

```
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
['positive']: Sentiment=Positive
```

In [18]:

```
import nltk
from nltk.corpus import stopwords
import csv

stopwords = set(stopwords.words('english'))

# Your positive and negative tweets
pos_tweets = [('It is not impossible', 'positive'),
              ('You are my lovely friend', 'positive'),
              ('She is a beautiful girl', 'positive'),
              ('He is looking handsome', 'positive'),
              ('Exercise is good for health', 'positive'),
              ('Today\'s weather is fantastic', 'positive'),
              ('I love Mango', 'positive')]

neg_tweets = [('You are my enemy friend', 'negative'),
              ('She is looking ugly', 'negative'),
              ('He is looking horrible', 'negative'),
              ('Sleeping more makes you lazy', 'negative'),
              ('Today\'s weather is very bad', 'negative'),
              ('I hate Banana', 'negative')]

# Combine positive and negative tweets
Senti_tweets = []
for (words, sentiment) in pos_tweets + neg_tweets:
    words_filtered = [e.lower() for e in words.split() if len(e) >= 3]
    Senti_tweets.append((words_filtered, sentiment))

# Define functions
def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
        all_words.extend(words)
    return all_words

def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features

def extract_features(document, word_features_filtered):
    document_words = set(document)
    features = {}
    for word in word_features_filtered:
        features['contains(%s)' % word] = (word in document_words)
    return features

# Get word features
word_features = get_word_features(get_words_in_tweets(Senti_tweets))
word_features_filtered = [w for w in word_features if w not in stopwords]

# Train the classifier
training_set = nltk.classify.apply_features(lambda doc: extract_features(doc, word_features
classifier = nltk.NaiveBayesClassifier.train(training_set)

# Apply the classifier to new data from the CSV file
csv_file_path = r"C:\Users\raval\Downloads\full_training_dataset.csv"

with open(csv_file_path, 'r', encoding='latin-1') as csv_file:
    csv_reader = csv.reader(csv_file)
    next(csv_reader)  # Skip the header row if it exists
    for row in csv_reader:
        test_tweet = row[0]
        features = extract_features(test_tweet.split(), word_features_filtered)
        sentiment = classifier.classify(features)
        print(f"{test_tweet}: Sentiment={sentiment.capitalize()}")
```

```
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
positive: Sentiment=Positive
```

In [25]:
```
import nltk
from nltk.corpus import stopwords
import csv
import numpy as np

stopwords = set(stopwords.words('english'))
```

In [23]:
```
import pandas as pd

df = pd.read_csv(r"C:\Users\raval\Downloads\full_training_dataset.csv",names=["sentiment","
```

In [24]:
```
df
```

Out[24]:

	sentiment	sentence
0	positive	the rock is destined to be the 21st century's ...
1	positive	the gorgeously elaborate continuation of " the...
2	positive	effective but too-tepid biopic
3	positive	if you sometimes like to go to the movies to h...
4	positive	emerges as something rare , an issue movie tha...
...	...	...
21599	neutral	@madtruckman 'Modern Day Autograph", I like th...
21600	neutral	62 Ways to Use #Twitter for Business: http://t...
21601	neutral	Log off #Facebook On #Twitter , But I Think i'...
21602	neutral	"#twitter's dumb, I don't like it." Hush up, ...
21603	neutral	It's almost 4:20. Where is your bong? Is it pa...

21604 rows × 2 columns

In [26]:
```
df[df.sentiment=="positive"]
```

Out[26]:

	sentiment	sentence
0	positive	the rock is destined to be the 21st century's ...
1	positive	the gorgeously elaborate continuation of " the...
2	positive	effective but too-tepid biopic
3	positive	if you sometimes like to go to the movies to h...
4	positive	emerges as something rare , an issue movie tha...
...	...	...
19328	positive	goodmorning, preparing for conference call wit...
19329	positive	she makes every everything bad in my life seem...
19330	positive	"We'll be a beets cover band". I wou...
19331	positive	Yep, I'm receiving DMs, so at least some of yo...
19332	positive	Hi Rhian this is my first post on twitter so h...

9667 rows × 2 columns

In [27]:
```
positive_sentences = df[df['sentiment'] == 'positive']['sentence'].tolist()

# Create a list of tuples
positive_tuples = [(sentiment, sentence) for sentiment, sentence in zip(['positive'] * len(

print(positive_tuples)
```

```
[('positive', 'the rock is destined to be the 21st century\'s new " conan " and that h
e\'s going to make a splash even greater than arnold schwarzenegger , jean-claud van d
amme or steven segal .'), ('positive', 'the gorgeously elaborate continuation of " the
lord of the rings " trilogy is so huge that a column of words cannot adequately descri
be co-writer/director peter jackson\'s expanded vision of j . r . r . tolkien\'s middl
e-earth .'), ('positive', 'effective but too-tepid biopic'), ('positive', 'if you some
times like to go to the movies to have fun , wasabi is a good place to start .'), ('po
sitive', "emerges as something rare , an issue movie that's so honest and keenly obser
ved that it doesn't feel like one ."), ('positive', 'the film provides some great insi
ght into the neurotic mindset of all comics -- even those who have reached the absolut
e top of the game .'), ('positive', 'offers that rare combination of entertainment and
education .'), ('positive', 'perhaps no picture ever made has more literally showed th
at the road to hell is paved with good intentions .'), ('positive', "steers turns in a
snappy screenplay that curls at the edges ; it's so clever you want to hate it . but h
e somehow pulls it off ."), ('positive', 'take care of my cat offers a refreshingly di
fferent slice of asian cinema .'), ('positive', 'this is a film well worth seeing , ta
lking and singing heads and all .'), ('positive', 'what really surprises about wisegir
ls is its low-key quality and genuine tenderness .'), ('positive', '( wendigo is ) why
we go to the cinema : to be fed through the eye , the heart , the mind .'), ('positiv
```

In [29]:
```
# positive = df[df['sentiment'] == 'positive']['sentence'].tolist()
negative_sentences = df[df['sentiment'] == 'negative']['sentence'].tolist()

# Create a list of tuples
negative_tuples = [(sentiment, sentence) for sentiment, sentence in zip(['negative'] * len(

print(negative_tuples)
```

```
[('negative', 'simplistic , silly and tedious .'), ('negative', "it's so laddish and j
uvenile , only teenage boys could possibly find it funny ."), ('negative', 'exploitati
ve and largely devoid of the depth or sophistication that would make watching such a g
raphic treatment of the crimes bearable .'), ('negative', '[garbus] discards the poten
tial for pathological study , exhuming instead , the skewed melodrama of the circumsta
ntial situation .'), ('negative', 'a visually flashy but narratively opaque and emotio
nally vapid exercise in style and mystification .'), ('negative', "the story is also a
s unoriginal as they come , already having been recycled more times than i'd care to c
ount ."), ('negative', "about the only thing to give the movie points for is bravado -
- to take an entirely stale concept and push it through the audience's meat grinder on
e more time ."), ('negative', 'not so much farcical as sour .'), ('negative', 'unfortu
nately the story and the actors are served with a hack script .'), ('negative', 'all t
he more disquieting for its relatively gore-free allusions to the serial murders , but
it falls down in its attempts to humanize its subject .'), ('negative', 'a sentimental
mess that never rings true .'), ('negative', 'while the performances are often engagin
g , this loose collection of largely improvised numbers would probably have worked bet
ter as a one-hour tv documentary .'), ('negative', 'interesting , but not compelling
.'), ('negative', 'on a cutting room floor somewhere lies . . . footage that might hav
e made no such thing a trenchant , ironic cultural satire instead of a frustrating mis
```

In [32]:
```
# positive = df[df['sentiment'] == 'positive']['sentence'].tolist()
neutral_sentences = df[df['sentiment'] == 'neutral']['sentence'].tolist()

# Create a list of tuples
neutral_tuples = [(sentiment, sentence) for sentiment, sentence in zip(['neutral'] * len(ne

print(neutral_tuples)
```

```
[('neutral', '@Late_Show I would have watched but the folks at @apple have a jihad aga
inst adobe flash. Plse consider a YouTube link in future on UR site'), ('neutral', 'RT
@rdingwell: .@Apple has a record quarter and because a bunch of professional guessers
(aka analysts) wanted more, its a disappointmen ...'), ('neutral', "Hey @apple, androi
ds releasing brand new state of the art phones, whens your new phone come out? What's
that? (cont) http://t.co/2sko9l3d"), (http://t.co/2sko9l3d"),) ('neutral', '.@Apple ha
s a record quarter and because a bunch of professional guessers (aka analysts) wanted
more, its a disappointment #wtf'), ('neutral', "@Apple how fun wouldn't it be if it wa
s possible to integrate  ( soon to be named ) with notifications?"), ('neutral', 'Inte
resting read on war b/w @Apple & @Samsung- http://t.co/Vt9d24Yi (http://t.co/Vt9d24Yi)
-using latter, agree lack of innovation, but better specs at same price!'), ('neutra
l', 'RT @adamnash: The takeaway from the @Apple earnings call?  Even Apple needs a new
iPhone release every 12 months to stay competitive. cc ...'), ('neutral', 'The takeawa
y from the @Apple earnings call?  Even Apple needs a new iPhone release every 12 month
s to stay competitive. cc: @hblodget'), ('neutral', "Today's headline: @apple reports
lower 4Q earnings. Headline in 3 months: @Apple reports record Q1 earnings."), ('neutr
al', 'Win an @Apple iPod Touch from @Mommy_gaga, get the @Pampers Hello World Baby Mem
ories App! http://t.co/XVcch6Os (http://t.co/XVcch6Os) #PampersHelloApps"'), ('neutra
l', '@apple expanded the app store to more than 20 new countries in the december quart
```

In [37]:
```
# Combine positive and negative tweets
Senti_tweets = []
for (sentiment, sentence) in positive_tuples + negative_tuples:
    words_filtered = [e.lower() for e in sentence.split() if len(e) >= 3]
    Senti_tweets.append((words_filtered, sentiment))
Senti_tweets
```

Out[37]:
```
[(['the',
   'rock',
   'destined',
   'the',
   '21st',
   "century's",
   'new',
   'conan',
   'and',
   'that',
   "he's",
   'going',
   'make',
   'splash',
   'even',
   'greater',
   'than',
   'arnold',
   'schwarzenegger',
```

In [38]:
```
def get_words_in_tweets(tweets):
    all_words=[]
    for (sentiment, sentence) in Senti_tweets:
        all_words.extend(words)
    return (all_words)

def get_word_features(wordlist):
    wordlist=nltk.FreqDist(wordlist)
    word_features=wordlist.keys()
    return word_features

word_features=get_word_features(get_words_in_tweets(Senti_tweets))
print(word_features)

word_features_filtered=[]
for w in word_features:
    if w not in stopwords:
        word_features_filtered.append(w)

print(word_features_filtered)
```

```
dict_keys(['I', ' ', 'h', 'a', 't', 'e', 'B', 'n'])
['I', ' ', 'h', 'e', 'B', 'n']
```

In [ ]:

In [ ]:

In [39]:
```
def extract_features(document):
    document_words=set(document)
    features={}
    for word in word_features_filtered:
        features['contains(%s)' %word] = (word in document_words)
    return features

training_set = nltk.classify.apply_features(extract_features, Senti_tweets)
classifier = nltk.NaiveBayesClassifier.train(training_set)

test_tweet='This is a horrible book'
print("{}: Sentiment={}".format(test_tweet, classifier.classify(extract_features(test_tweet
```

```
This is a horrible book: Sentiment=positive
```

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [ ]:

In [1]:
```
import nltk
from nltk.corpus import stopwords
import csv
import numpy as np

stopwords = set(stopwords.words('english'))
```

In [2]:
```
import pandas as pd
df = pd.read_csv(r"C:\Users\raval\Downloads\full_training_dataset.csv",names=["sentiment","
```

In [3]:
```
positive_sentences = df[df['sentiment'] == 'positive']['sentence'].tolist()

# Create a list of tuples
positive_tuples = [(sentiment, sentence) for sentiment, sentence in zip(['positive'] * len(

print(positive_tuples)


# positive = df[df['sentiment'] == 'positive']['sentence'].tolist()
negative_sentences = df[df['sentiment'] == 'negative']['sentence'].tolist()

# Create a list of tuples
negative_tuples = [(sentiment, sentence) for sentiment, sentence in zip(['negative'] * len(

print(negative_tuples)
```

```
[('positive', 'the rock is destined to be the 21st century\'s new " conan " and that h
e\'s going to make a splash even greater than arnold schwarzenegger , jean-claud van d
amme or steven segal .'), ('positive', 'the gorgeously elaborate continuation of " the
lord of the rings " trilogy is so huge that a column of words cannot adequately descri
be co-writer/director peter jackson\'s expanded vision of j . r . r . tolkien\'s middl
e-earth .'), ('positive', 'effective but too-tepid biopic'), ('positive', 'if you some
times like to go to the movies to have fun , wasabi is a good place to start .'), ('po
sitive', "emerges as something rare , an issue movie that's so honest and keenly obser
ved that it doesn't feel like one ."), ('positive', 'the film provides some great insi
ght into the neurotic mindset of all comics -- even those who have reached the absolut
e top of the game .'), ('positive', 'offers that rare combination of entertainment and
education .'), ('positive', 'perhaps no picture ever made has more literally showed th
at the road to hell is paved with good intentions .'), ('positive', "steers turns in a
snappy screenplay that curls at the edges ; it's so clever you want to hate it . but h
e somehow pulls it off ."), ('positive', 'take care of my cat offers a refreshingly di
fferent slice of asian cinema .'), ('positive', 'this is a film well worth seeing , ta
lking and singing heads and all .'), ('positive', 'what really surprises about wisegir
ls is its low-key quality and genuine tenderness .'), ('positive', '( wendigo is ) why
we go to the cinema : to be fed through the eye , the heart , the mind .'), ('positiv
```

In [4]:
```
# Combine positive and negative tuples
Senti_tweets = []
for (sentiment, sentence) in positive_tuples + negative_tuples:
    words_filtered = [e.lower() for e in sentence.split() if len(e) >= 3]
    Senti_tweets.append((words_filtered, sentiment))

print(Senti_tweets)
```

```
[(['the', 'rock', 'destined', 'the', '21st', "century's", 'new', 'conan', 'and', 'tha
t', "he's", 'going', 'make', 'splash', 'even', 'greater', 'than', 'arnold', 'schwarzen
egger', 'jean-claud', 'van', 'damme', 'steven', 'segal'], 'positive'), (['the', 'gorge
ously', 'elaborate', 'continuation', 'the', 'lord', 'the', 'rings', 'trilogy', 'huge',
'that', 'column', 'words', 'cannot', 'adequately', 'describe', 'co-writer/director',
'peter', "jackson's", 'expanded', 'vision', "tolkien's", 'middle-earth'], 'positive'),
(['effective', 'but', 'too-tepid', 'biopic'], 'positive'), (['you', 'sometimes', 'lik
e', 'the', 'movies', 'have', 'fun', 'wasabi', 'good', 'place', 'start'], 'positive'),
(['emerges', 'something', 'rare', 'issue', 'movie', "that's", 'honest', 'and', 'keenl
y', 'observed', 'that', "doesn't", 'feel', 'like', 'one'], 'positive'), (['the', 'fil
m', 'provides', 'some', 'great', 'insight', 'into', 'the', 'neurotic', 'mindset', 'al
l', 'comics', 'even', 'those', 'who', 'have', 'reached', 'the', 'absolute', 'top', 'th
e', 'game'], 'positive'), (['offers', 'that', 'rare', 'combination', 'entertainment',
'and', 'education'], 'positive'), (['perhaps', 'picture', 'ever', 'made', 'has', 'mor
e', 'literally', 'showed', 'that', 'the', 'road', 'hell', 'paved', 'with', 'good', 'in
tentions'], 'positive'), (['steers', 'turns', 'snappy', 'screenplay', 'that', 'curls',
'the', 'edges', "it's", 'clever', 'you', 'want', 'hate', 'but', 'somehow', 'pulls', 'o
ff'], 'positive'), (['take', 'care', 'cat', 'offers', 'refreshingly', 'different', 'sl
ice', 'asian', 'cinema'], 'positive'), (['this', 'film', 'well', 'worth', 'seeing', 't
```

In [5]:
```
def get_words_in_tweets(tweets):
    all_words = []
    for (words, sentiment) in tweets:
        all_words.extend(words)
    return all_words
```

In [6]:
```
def get_word_features(wordlist):
    wordlist = nltk.FreqDist(wordlist)
    word_features = wordlist.keys()
    return word_features
```

In [7]:
```
# Assuming Senti_tweets is already defined
words_in_tweets = get_words_in_tweets(Senti_tweets)
word_features = get_word_features(words_in_tweets)
print(word_features)

word_features_filtered = [w for w in word_features if w not in stopwords]
print(word_features_filtered)
```

```
dict_keys(['the', 'rock', 'destined', '21st', "century's", 'new', 'conan', 'and', 'tha
t', "he's", 'going', 'make', 'splash', 'even', 'greater', 'than', 'arnold', 'schwarzen
egger', 'jean-claud', 'van', 'damme', 'steven', 'segal', 'gorgeously', 'elaborate', 'c
ontinuation', 'lord', 'rings', 'trilogy', 'huge', 'column', 'words', 'cannot', 'adequa
tely', 'describe', 'co-writer/director', 'peter', "jackson's", 'expanded', 'vision',
"tolkien's", 'middle-earth', 'effective', 'but', 'too-tepid', 'biopic', 'you', 'someti
mes', 'like', 'movies', 'have', 'fun', 'wasabi', 'good', 'place', 'start', 'emerges',
'something', 'rare', 'issue', 'movie', "that's", 'honest', 'keenly', 'observed', "does
n't", 'feel', 'one', 'film', 'provides', 'some', 'great', 'insight', 'into', 'neuroti
c', 'mindset', 'all', 'comics', 'those', 'who', 'reached', 'absolute', 'top', 'game',
'offers', 'combination', 'entertainment', 'education', 'perhaps', 'picture', 'ever',
'made', 'has', 'more', 'literally', 'showed', 'road', 'hell', 'paved', 'with', 'intent
ions', 'steers', 'turns', 'snappy', 'screenplay', 'curls', 'edges', "it's", 'clever',
'want', 'hate', 'somehow', 'pulls', 'off', 'take', 'care', 'cat', 'refreshingly', 'dif
ferent', 'slice', 'asian', 'cinema', 'this', 'well', 'worth', 'seeing', 'talking', 'si
nging', 'heads', 'what', 'really', 'surprises', 'about', 'wisegirls', 'its', 'low-ke
y', 'quality', 'genuine', 'tenderness', 'wendigo', 'why', 'fed', 'through', 'eye', 'he
art', 'mind', 'greatest', 'family-oriented', 'fantasy-adventure', 'ultimately', 'ponde
rs', 'reasons', 'need', 'stories', 'much', 'utterly', 'compelling', "'who", 'wrote',
```

In [8]:
```
def extract_features(document, word_features_filtered):
    document_words = set(document)
    features = {}
    for word in word_features_filtered:
        features['contains(%s)' % word] = (word in document_words)
    return features
```

In [9]:
```
# Assuming Senti_tweets and word_features_filtered are already defined
training_set = nltk.classify.apply_features(lambda doc: extract_features(doc, word_features
classifier = nltk.NaiveBayesClassifier.train(training_set)

test_tweet = 'This is a horrible book'
features = extract_features(test_tweet.split(), word_features_filtered)
sentiment = classifier.classify(features)
print("{}: Sentiment={}".format(test_tweet, sentiment))
```

```
This is a horrible book: Sentiment=negative
```

In [21]:
```
test_tweet = 'explaination is very good'
features = extract_features(test_tweet.split(), word_features_filtered)
sentiment = classifier.classify(features)
print("{}: Sentiment={}".format(test_tweet, sentiment))
```

```
explaination is very good: Sentiment=positive
```

In [ ]:

Lab Assignment - 5

In [1]:
```
import numpy as np
import pandas as pd
import matplotlib.pyplot as plt
import seaborn as sns
import regex
import nltk
from nltk.corpus import stopwords
```

In [15]:
```
text= '''2001–02: Mike Denness incident, Kolkata Test, and breaking Bradman's record
During India's 2001 tour of South Africa, in the second Test match, referee Mike Denness fined four Indian players for excessive appealing, and fined

On the final day of the Kolkata Test against Australia in 2001, Tendulkar took three wickets, including the key wickets of Matthew Hayden and Adam Gi

In the 2002 series in the West Indies, Tendulkar started well, scoring 79 in the first Test. In the second Test at Port of Spain, Sachin Tendulkar sc

Then, in an unprecedented sequence, he scored just 0, 0, 8, and 0 in the next four innings.[130] He returned to form in the last Test scoring 41 and
```

In [16]:
```
words = nltk.word_tokenize(text)
tags = nltk.pos_tag(words)
print(tags)
```

```
[('2001–02', 'CD'), (':', ':'), ('Mike', 'NNP'), ('Denness', 'NNP'), ('incident', 'NN'), (',', ','), ('Kolkata', 'NNP'), ('Test', 'NNP'), (',',
','), ('and', 'CC'), ('breaking', 'VBG'), ('Bradman', 'NNP'), ("'s", 'POS'), ('record', 'NN'), ('During', 'IN'), ('India', 'NNP'), ("'s", 'POS'),
('2001', 'CD'), ('tour', 'NN'), ('of', 'IN'), ('South', 'NNP'), ('Africa', 'NNP'), (',', ','), ('in', 'IN'), ('the', 'DT'), ('second', 'JJ'), ('Tes
t', 'NNP'), ('match', 'NN'), (',', ','), ('referee', 'VBP'), ('Mike', 'NNP'), ('Denness', 'NNP'), ('fined', 'VBD'), ('four', 'CD'), ('Indian', 'J
J'), ('players', 'NNS'), ('for', 'IN'), ('excessive', 'JJ'), ('appealing', 'NN'), (',', ','), ('and', 'CC'), ('fined', 'VBD'), ('the', 'DT'), ('Indi
an', 'JJ'), ('captain', 'NN'), ('Sourav', 'NNP'), ('Ganguly', 'NNP'), ('for', 'IN'), ('not', 'RB'), ('controlling', 'VBG'), ('his', 'PRP$'), ('tea
m', 'NN'), ('.', '.'), ('[', 'CC'), ('119', 'CD'), (']', 'JJ'), ('Denness', 'NNP'), ('suspended', 'VBD'), ('Tendulkar', 'NNP'), ('from', 'IN'), ('on
e', 'CD'), ('match', 'NN'), ('for', 'IN'), ('alleged', 'JJ'), ('ball', 'NN'), ('tampering', 'NN'), ('.', '.'), ('Television', 'NN'), ('cameras', 'NN
S'), ('captured', 'VBD'), ('images', 'NNS'), ('that', 'IN'), ('suggested', 'VBD'), ('Tendulkar', 'NNP'), ('may', 'MD'), ('have', 'VB'), ('been', 'VB
N'), ('involved', 'VBN'), ('in', 'IN'), ('cleaning', 'VBG'), ('the', 'DT'), ('seam', 'NN'), ('of', 'IN'), ('the', 'DT'), ('cricket', 'NN'), ('ball',
'NN'), ('.', '.'), ('[', 'CC'), ('120', 'CD'), (']', 'JJ'), ('[', '$'), ('121', 'CD'), (']', 'IN'), ('The', 'DT'), ('incident', 'NN'), ('escalated',
'VBD'), ('to', 'TO'), ('include', 'VB'), ('sports', 'NNS'), ('journalists', 'NNS'), ('accusing', 'VBG'), ('Denness', 'NNP'), ('of', 'IN'), ('racis
m', 'NN'), (',', ','), ('[', 'VBZ'), ('122', 'CD'), (']', 'NN'), ('and', 'CC'), ('led', 'VBD'), ('to', 'TO'), ('Denness', 'NNP'), ('being', 'VBG'),
('barred', 'VBN'), ('from', 'IN'), ('entering', 'VBG'), ('the', 'DT'), ('venue', 'NN'), ('of', 'IN'), ('the', 'DT'), ('third', 'JJ'), ('Test', 'NN
P'), ('match', 'NN'), ('.', '.'), ('The', 'DT'), ('ICC', 'NNP'), ('revoked', 'VBD'), ('the', 'DT'), ('status', 'NN'), ('of', 'IN'), ('the', 'DT'),
('match', 'NN'), ('as', 'IN'), ('a', 'DT'), ('Test', 'NN'), ('as', 'IN'), ('the', 'DT'), ('teams', 'NNS'), ('rejected', 'VBD'), ('the', 'DT'), ('app
ointed', 'VBN'), ('referee', 'NN'), ('.', '.'), ('[', 'CC'), ('123', 'CD'), (']', 'VBZ'), ('The', 'DT'), ('charges', 'NNS'), ('against', 'IN'), ('Te
ndulkar', 'NNP'), ('triggered', 'VBD'), ('a', 'DT'), ('massive', 'JJ'), ('backlash', 'NN'), ('from', 'IN'), ('the', 'DT'), ('Indian', 'JJ'), ('publi
c', 'NN'), ('.', '.'), ('[', 'CC'), ('124', 'CD'), (']', 'NN'), ('On', 'IN'), ('the', 'DT'), ('final', 'JJ'), ('day', 'NN'), ('of', 'IN'), ('the',
'DT'), ('Kolkata', 'NNP'), ('Test', 'NNP'), ('against', 'IN'), ('Australia', 'NNP'), ('in', 'IN'), ('2001', 'CD'), (',', ','), ('Tendulkar', 'NNP'),
('took', 'VBD'), ('three', 'CD'), ('wickets', 'NNS'), (',', ','), ('including', 'VBG'), ('the', 'DT'), ('key', 'JJ'), ('wickets', 'NNS'), ('of', 'I
N'), ('Matthew', 'NNP'), ('Hayden', 'NNP'), ('and', 'CC'), ('Adam', 'NNP'), ('Gilchrist', 'NNP'), (',', ','), ('who', 'WP'), ('were', 'VBD'), ('cent
urions', 'NNS'), ('in', 'IN'), ('the', 'DT'), ('previous', 'JJ'), ('Test', 'NNP'), ('.', '.'), ('His', 'PRP$'), ('three', 'CD'), ('wickets', 'NNS'),
('haul', 'VBP'), ('helped', 'VBD'), ('India', 'NNP'), ('win', 'VB'), ('the', 'DT'), ('match', 'NN'), ('.', '.'), ('[', 'CC'), ('125', 'CD'), (']',
'NN'), ('In', 'IN'), ('the', 'DT'), ('five-match', 'JJ'), ('ODI', 'NNP'), ('series', 'NN'), ('that', 'WDT'), ('followed', 'VBD'), (',', ','), ('he',
'PRP'), ('took', 'VBD'), ('his', 'PRP$'), ('100th', 'JJ'), ('wicket', 'NN'), ('in', 'IN'), ('ODIs', 'NNP'), (',', ','), ('claiming', 'VBG'), ('the',
'DT'), ('wicket', 'NN'), ('of', 'IN'), ('then', 'RB'), ('Australian', 'NNP'), ('captain', 'NN'), ('Steve', 'NNP'), ('Waugh', 'NNP'), ('in', 'IN'),
('the', 'DT'), ('final', 'JJ'), ('match', 'NN'), ('at', 'IN'), ('the', 'DT'), ('Fatorda', 'NNP'), ('Stadium', 'NNP'), ('in', 'IN'), ('Goa', 'NNP'),
('.', '.'), ('[', 'CC'), ('126', 'CD'), (']', 'NN'), ('In', 'IN'), ('the', 'DT'), ('2002', 'CD'), ('series', 'NN'), ('in', 'IN'), ('the', 'DT'), ('W
est', 'NNP'), ('Indies', 'NNPS'), (',', ','), ('Tendulkar', 'NNP'), ('started', 'VBD'), ('well', 'RB'), (',', ','), ('scoring', 'VBG'), ('79', 'C
D'), ('in', 'IN'), ('the', 'DT'), ('first', 'JJ'), ('Test', 'NNP'), ('.', '.'), ('In', 'IN'), ('the', 'DT'), ('second', 'JJ'), ('Test', 'NNP'), ('a
t', 'IN'), ('Port', 'NNP'), ('of', 'IN'), ('Spain', 'NNP'), (',', ','), ('Sachin', 'NNP'), ('Tendulkar', 'NNP'), ('scored', 'VBD'), ('117', 'CD'),
('in', 'IN'), ('the', 'DT'), ('first', 'JJ'), ('innings', 'NNS'), (',', ','), ('his', 'PRP$'), ('29th', 'CD'), ('Test', 'NNP'), ('century', 'NN'),
('in', 'IN'), ('his', 'PRP$'), ('93rd', 'CD'), ('Test', 'NNP'), ('match', 'NN'), (',', ','), ('to', 'TO'), ('equal', 'VB'), ('Donald', 'NNP'), ('Bra
dman', 'NNP'), ("'s", 'POS'), ('record', 'NN'), ('of', 'IN'), ('29', 'CD'), ('Test', 'NNP'), ('hundreds', 'NNS'), ('.', '.'), ('He', 'PRP'), ('was',
'VBD'), ('gifted', 'VBN'), ('a', 'DT'), ('Ferrari', 'NNP'), ('360', 'CD'), ('Modena', 'NNP'), ('by', 'IN'), ('Fiat', 'NNP'), ('through', 'IN'), ('Mi
chael', 'NNP'), ('Schumacher', 'NNP'), ('for', 'IN'), ('achieving', 'VBG'), ('this', 'DT'), ('feat', 'NN'), ('.', '.'), ('[', 'CC'), ('127', 'CD'),
(']', 'JJ'), ('[', '$'), ('128', 'CD'), (']', 'NNP'), ('[', 'VBD'), ('129', 'CD'), (']', 'NN'), ('Then', 'RB'), (',', ','), ('in', 'IN'), ('an', 'D
T'), ('unprecedented', 'JJ'), ('sequence', 'NN'), (',', ','), ('he', 'PRP'), ('scored', 'VBD'), ('just', 'RB'), ('0', 'CD'), (',', ','), ('0', 'C
D'), (',', ','), ('8', 'CD'), (',', ','), ('and', 'CC'), ('0', 'CD'), ('in', 'IN'), ('the', 'DT'), ('next', 'JJ'), ('four', 'CD'), ('innings', 'NN
S'), ('.', '.'), ('[', '$'), ('130', 'CD'), (']', 'NN'), ('He', 'PRP'), ('returned', 'VBD'), ('to', 'TO'), ('form', 'VB'), ('in', 'IN'), ('the', 'D
T'), ('last', 'JJ'), ('Test', 'NNP'), ('scoring', 'VBG'), ('41', 'CD'), ('and', 'CC'), ('86', 'CD'), (',', ','), ('one', 'CD'), ('half', 'JJ'), ('ce
ntury', 'NN'), ('.', '.'), ('However', 'RB'), (',', ','), ('India', 'NNP'), ('lost', 'VBD'), ('the', 'DT'), ('series', 'NN'), ('.', '.'), ('[', 'C
C'), ('131', 'CD'), (']', 'NN'), ('In', 'IN'), ('this', 'DT'), ('period', 'NN'), (',', ','), ('in', 'IN'), ('the', 'DT'), ('third', 'JJ'), ('Test',
'NNP'), ('match', 'NN'), ('against', 'IN'), ('England', 'NNP'), ('in', 'IN'), ('August', 'NNP'), ('2002', 'CD'), (',', ','), ('Tendulkar', 'NNP'),
('scored', 'VBD'), ('his', 'PRP$'), ('30th', 'JJ'), ('Test', 'NNP'), ('century', 'NN'), ('to', 'TO'), ('surpass', 'VB'), ('Bradman', 'NNP'), ("'s",
'POS'), ('haul', 'NN'), (',', ','), ('in', 'IN'), ('his', 'PRP$'), ('99th', 'CD'), ('Test', 'NNP'), ('match', 'NN'), ('.', '.'), ('[', 'CC'), ('13
2', 'CD'), (']', 'JJ'), ('[', '$'), ('133', 'CD'), (']', 'NN')]
```

In [17]:
```
from nltk import bigrams
bigrams = list(bigrams(words))
print(bigrams)
```

```
[('2001-02', ':'), (':', 'Mike'), ('Mike', 'Denness'), ('Denness', 'incident'), ('incident', ','), (',', 'Kolkata'), ('Kolkata', 'Test'), ('Test',
','), (',', 'and'), ('and', 'breaking'), ('breaking', 'Bradman'), ('Bradman', "'s"), ("'s", 'record'), ('record', 'During'), ('During', 'India'),
('India', "'s"), ("'s", '2001'), ('2001', 'tour'), ('tour', 'of'), ('of', 'South'), ('South', 'Africa'), ('Africa', ','), (',', 'in'), ('in', 'th
e'), ('the', 'second'), ('second', 'Test'), ('Test', 'match'), ('match', ','), (',', 'referee'), ('referee', 'Mike'), ('Mike', 'Denness'), ('Dennes
s', 'fined'), ('fined', 'four'), ('four', 'Indian'), ('Indian', 'players'), ('players', 'for'), ('for', 'excessive'), ('excessive', 'appealing'),
('appealing', ','), (',', 'and'), ('and', 'fined'), ('fined', 'the'), ('the', 'Indian'), ('Indian', 'captain'), ('captain', 'Sourav'), ('Sourav', 'G
anguly'), ('Ganguly', 'for'), ('for', 'not'), ('not', 'controlling'), ('controlling', 'his'), ('his', 'team'), ('team', '.'), ('.', '['), ('[', '11
9'), ('119', ']'), (']', 'Denness'), ('Denness', 'suspended'), ('suspended', 'Tendulkar'), ('Tendulkar', 'from'), ('from', 'one'), ('one', 'match'),
('match', 'for'), ('for', 'alleged'), ('alleged', 'ball'), ('ball', 'tampering'), ('tampering', '.'), ('.', 'Television'), ('Television', 'camera
s'), ('cameras', 'captured'), ('captured', 'images'), ('images', 'that'), ('that', 'suggested'), ('suggested', 'Tendulkar'), ('Tendulkar', 'may'),
('may', 'have'), ('have', 'been'), ('been', 'involved'), ('involved', 'in'), ('in', 'cleaning'), ('cleaning', 'the'), ('the', 'seam'), ('seam', 'o
f'), ('of', 'the'), ('the', 'cricket'), ('cricket', 'ball'), ('ball', '.'), ('.', '['), ('[', '120'), ('120', ']'), (']', '['), ('[', '121'), ('12
1', ']'), (']', 'The'), ('The', 'incident'), ('incident', 'escalated'), ('escalated', 'to'), ('to', 'include'), ('include', 'sports'), ('sports', 'j
ournalists'), ('journalists', 'accusing'), ('accusing', 'Denness'), ('Denness', 'of'), ('of', 'racism'), ('racism', ','), (',', '['), ('[', '122'),
('122', ']'), (']', 'and'), ('and', 'led'), ('led', 'to'), ('to', 'Denness'), ('Denness', 'being'), ('being', 'barred'), ('barred', 'from'), ('fro
m', 'entering'), ('entering', 'the'), ('the', 'venue'), ('venue', 'of'), ('of', 'the'), ('the', 'third'), ('third', 'Test'), ('Test', 'match'), ('ma
tch', '.'), ('.', 'The'), ('The', 'ICC'), ('ICC', 'revoked'), ('revoked', 'the'), ('the', 'status'), ('status', 'of'), ('of', 'the'), ('the', 'matc
h'), ('match', 'as'), ('as', 'a'), ('a', 'Test'), ('Test', 'as'), ('as', 'the'), ('the', 'teams'), ('teams', 'rejected'), ('rejected', 'the'), ('th
e', 'appointed'), ('appointed', 'referee'), ('referee', '.'), ('.', '['), ('[', '123'), ('123', ']'), (']', 'The'), ('The', 'charges'), ('charges',
'against'), ('against', 'Tendulkar'), ('Tendulkar', 'triggered'), ('triggered', 'a'), ('a', 'massive'), ('massive', 'backlash'), ('backlash', 'fro
m'), ('from', 'the'), ('the', 'Indian'), ('Indian', 'public'), ('public', '.'), ('.', '['), ('[', '124'), ('124', ']'), (']', 'On'), ('On', 'the'),
('the', 'final'), ('final', 'day'), ('day', 'of'), ('of', 'the'), ('the', 'Kolkata'), ('Kolkata', 'Test'), ('Test', 'against'), ('against', 'Austral
ia'), ('Australia', 'in'), ('in', '2001'), ('2001', ','), (',', 'Tendulkar'), ('Tendulkar', 'took'), ('took', 'three'), ('three', 'wickets'), ('wick
ets', ','), (',', 'including'), ('including', 'the'), ('the', 'key'), ('key', 'wickets'), ('wickets', 'of'), ('of', 'Matthew'), ('Matthew', 'Hayde
n'), ('Hayden', 'and'), ('and', 'Adam'), ('Adam', 'Gilchrist'), ('Gilchrist', ','), (',', 'who'), ('who', 'were'), ('were', 'centurions'), ('centuri
ons', 'in'), ('in', 'the'), ('the', 'previous'), ('previous', 'Test'), ('Test', '.'), ('.', 'His'), ('His', 'three'), ('three', 'wickets'), ('wicket
s', 'haul'), ('haul', 'helped'), ('helped', 'India'), ('India', 'win'), ('win', 'the'), ('the', 'match'), ('match', '.'), ('.', '['), ('[', '125'),
('125', ']'), (']', 'In'), ('In', 'the'), ('the', 'five-match'), ('five-match', 'ODI'), ('ODI', 'series'), ('series', 'that'), ('that', 'followed'),
('followed', ','), (',', 'he'), ('he', 'took'), ('took', 'his'), ('his', '100th'), ('100th', 'wicket'), ('wicket', 'in'), ('in', 'ODIs'), ('ODIs',
','), (',', 'claiming'), ('claiming', 'the'), ('the', 'wicket'), ('wicket', 'of'), ('of', 'then'), ('then', 'Australian'), ('Australian', 'captai
n'), ('captain', 'Steve'), ('Steve', 'Waugh'), ('Waugh', 'in'), ('in', 'the'), ('the', 'final'), ('final', 'match'), ('match', 'at'), ('at', 'the'),
('the', 'Fatorda'), ('Fatorda', 'Stadium'), ('Stadium', 'in'), ('in', 'Goa'), ('Goa', '.'), ('.', '['), ('[', '126'), ('126', ']'), (']', 'In'), ('I
n', 'the'), ('the', '2002'), ('2002', 'series'), ('series', 'in'), ('in', 'the'), ('the', 'West'), ('West', 'Indies'), ('Indies', ','), (',', 'Tendu
lkar'), ('Tendulkar', 'started'), ('started', 'well'), ('well', ','), (',', 'scoring'), ('scoring', '79'), ('79', 'in'), ('in', 'the'), ('the', 'fir
st'), ('first', 'Test'), ('Test', '.'), ('.', 'In'), ('In', 'the'), ('the', 'second'), ('second', 'Test'), ('Test', 'at'), ('at', 'Port'), ('Port',
'of'), ('of', 'Spain'), ('Spain', ','), (',', 'Sachin'), ('Sachin', 'Tendulkar'), ('Tendulkar', 'scored'), ('scored', '117'), ('117', 'in'), ('in',
'the'), ('the', 'first'), ('first', 'innings'), ('innings', ','), (',', 'his'), ('his', '29th'), ('29th', 'Test'), ('Test', 'century'), ('century',
'in'), ('in', 'his'), ('his', '93rd'), ('93rd', 'Test'), ('Test', 'match'), ('match', ','), (',', 'to'), ('to', 'equal'), ('equal', 'Donald'), ('Don
ald', 'Bradman'), ('Bradman', "'s"), ("'s", 'record'), ('record', 'of'), ('of', '29'), ('29', 'Test'), ('Test', 'hundreds'), ('hundreds', '.'),
('.', 'He'), ('He', 'was'), ('was', 'gifted'), ('gifted', 'a'), ('a', 'Ferrari'), ('Ferrari', '360'), ('360', 'Modena'), ('Modena', 'by'), ('by', 'F
iat'), ('Fiat', 'through'), ('through', 'Michael'), ('Michael', 'Schumacher'), ('Schumacher', 'for'), ('for', 'achieving'), ('achieving', 'this'),
('this', 'feat'), ('feat', '.'), ('.', '['), ('[', '127'), ('127', ']'), (']', '['), ('[', '128'), ('128', ']'), (']', '['), ('[', '129'), ('129',
']'), (']', 'Then'), ('Then', ','), (',', 'in'), ('in', 'an'), ('an', 'unprecedented'), ('unprecedented', 'sequence'), ('sequence', ','), (',', 'h
e'), ('he', 'scored'), ('scored', 'just'), ('just', '0'), ('0', ','), (',', '0'), ('0', ','), (',', '8'), ('8', ','), (',', 'and'), ('and', '0'),
('0', 'in'), ('in', 'the'), ('the', 'next'), ('next', 'four'), ('four', 'innings'), ('innings', '.'), ('.', '['), ('[', '130'), ('130', ']'), (']',
'He'), ('He', 'returned'), ('returned', 'to'), ('to', 'form'), ('form', 'in'), ('in', 'the'), ('the', 'last'), ('last', 'Test'), ('Test', 'scorin
g'), ('scoring', '41'), ('41', 'and'), ('and', '86'), ('86', ','), (',', 'one'), ('one', 'half'), ('half', 'century'), ('century', '.'), ('.', 'Howe
ver'), ('However', ','), (',', 'India'), ('India', 'lost'), ('lost', 'the'), ('the', 'series'), ('series', '.'), ('.', '['), ('[', '131'), ('131',
']'), (']', 'In'), ('In', 'this'), ('this', 'period'), ('period', ','), (',', 'in'), ('in', 'the'), ('the', 'third'), ('third', 'Test'), ('Test', 'm
atch'), ('match', 'against'), ('against', 'England'), ('England', 'in'), ('in', 'August'), ('August', '2002'), ('2002', ','), (',', 'Tendulkar'),
('Tendulkar', 'scored'), ('scored', 'his'), ('his', '30th'), ('30th', 'Test'), ('Test', 'century'), ('century', 'to'), ('to', 'surpass'), ('surpas
s', 'Bradman'), ('Bradman', "'s"), ("'s", 'haul'), ('haul', ','), (',', 'in'), ('in', 'his'), ('his', '99th'), ('99th', 'Test'), ('Test', 'match'),
('match', '.'), ('.', '['), ('[', '132'), ('132', ']'), (']', '['), ('[', '133'), ('133', ']')]
```

In [18]:
```
def filter_bigrams(tags):
    filtered_bigrams = []
    for i in range(len(tags) - 1):
        first_word, first_tag = tags[i]
        second_word, second_tag = tags[i + 1]

        if (first_tag.startswith('JJ') and second_tag in ['NN', 'NNS']) \
            or (first_tag in ['RB', 'RBR', 'RBS'] and second_tag.startswith('JJ') and not (second_tag == 'NN' or second_tag == 'NNS')) \
            or (first_tag.startswith('JJ') and second_tag.startswith('JJ') and not (second_tag == 'NN' or second_tag == 'NNS')) \
            or ((first_tag == 'NN' or first_tag == 'NNS') and second_tag.startswith('JJ') and not (second_tag == 'NN' or second_tag == 'NNS')) \
            or (first_tag in ['RB', 'RBR', 'RBS'] and second_tag.startswith('VB') and second_tag != 'NNS'):
            filtered_bigrams.append((first_word, second_word))
    return filtered_bigrams

filtered_bigrams = filter_bigrams(tags)

for bigram in filtered_bigrams:
  print(bigram)
```

```
('Indian', 'players')
('excessive', 'appealing')
('Indian', 'captain')
('not', 'controlling')
('alleged', 'ball')
('massive', 'backlash')
('Indian', 'public')
('final', 'day')
('key', 'wickets')
('100th', 'wicket')
('final', 'match')
('first', 'innings')
('unprecedented', 'sequence')
('half', 'century')
```

In [19]:
```
from nltk.probability import FreqDist
from math import log

def compute_pmi(filtered_bigrams, words):
  pmi_dict = {}
  word_freq = FreqDist(words)
  bigram_freq = FreqDist(filtered_bigrams)
  for bigram in filtered_bigrams:
    word1, word2 = bigram
    pmi = log((bigram_freq[bigram] * len(words)) / (word_freq[word1] * word_freq[word2]))
    pmi_dict[bigram] = pmi
  return pmi_dict

pmi_dict = compute_pmi(filtered_bigrams, words)

for bigram, pmi in pmi_dict.items():
  print(f"{bigram}: {pmi}")
```

```
('Indian', 'players'): 4.962844630259907
('excessive', 'appealing'): 6.061456918928017
('Indian', 'captain'): 4.269697449699962
('not', 'controlling'): 6.061456918928017
('alleged', 'ball'): 5.368309738368072
('massive', 'backlash'): 6.061456918928017
('Indian', 'public'): 4.962844630259907
('final', 'day'): 5.368309738368072
('key', 'wickets'): 4.962844630259907
('100th', 'wicket'): 5.368309738368072
('final', 'match'): 3.1710851610318525
('first', 'innings'): 4.675162557808126
('unprecedented', 'sequence'): 6.061456918928017
('half', 'century'): 4.962844630259907
```

In [20]:
```
positive_words = ["good", "great", "excellent", "amazing", "wonderful"]
negative_words = ["not", "stopped", "difficult", "horrible", "Problem"]
positive_bigrams = [bigram for bigram in filtered_bigrams if bigram[0] in positive_words or bigram[1] in positive_words]
negative_bigrams = [bigram for bigram in filtered_bigrams if bigram[0] in negative_words or bigram[1] in negative_words]
positive_pmi_dict = compute_pmi(positive_bigrams, words)
negative_pmi_dict = compute_pmi(negative_bigrams, words)
print("Positive Bigrams:")
for bigram, pmi in positive_pmi_dict.items():
  print(f"{bigram}: {pmi}")

print("\nNegative Bigrams:")
for bigram, pmi in negative_pmi_dict.items():
  print(f"{bigram}: {pmi}")
```

```
Positive Bigrams:

Negative Bigrams:
('not', 'controlling'): 6.061456918928017
```

In [21]:
```
positive_avg_pmi = sum(pmi for pmi in positive_pmi_dict.values()) / 1
negative_avg_pmi = sum(pmi for pmi in negative_pmi_dict.values()) / len(negative_pmi_dict)

print("Average PMI for positive bigrams:", positive_avg_pmi)
print("Average PMI for negative bigrams:", negative_avg_pmi)
if positive_avg_pmi > negative_avg_pmi:
  print("The overall sentiment of the text is positive.")
else:
  print("The overall sentiment of the text is negative.")
```

```
Average PMI for positive bigrams: 0.0
Average PMI for negative bigrams: 6.061456918928017
The overall sentiment of the text is negative.
```

In [ ]:

In [ ]:

In [ ]:

Lab Assignment - 6

In [3]:
```
d1= '''being an at from that are a they be or such for does same were she you the us am bee
d2= '''does with had was she or and has are from an could same be is they must you when the
d3= '''any where he who it so would do there they how it she could with a an have we more b
d4= '''he they more has how for may by same what I here from were a on there at we where do
d5= '''had many could will some would and I they the must been that such he who she then wh
d6= '''might be such he for he none how did I you we because should is will she at where it
```

In [5]:
```
# set(d1)
```

In [ ]:
```
# List of documents
documents = [d1, d2, d3, d4, d5, d6]

# Function to create the table
def create_table(documents):
    # Initialize an empty set for unique words
    unique_words = set()

    # Add unique words from each document to the set
    for doc in documents:
        unique_words.update(doc.split())

    # Creating the table
    table = {}
    for word in unique_words:
        table[word] = [1 if word in doc.split() else 0 for doc in documents]
    return table
```

In [7]:

```python
# Create the table
table = create_table(documents)

# Print the table
print("Unique Words\t", end="")
for i in range(1, 7):
    print(f"doc{i}\t", end="")
print()

for word, occurrences in table.items():
    print(f"{word}\t", end="")
    for occurrence in occurrences:
        print(f"{occurrence}\t", end="")
    print()
```

Unique Words	doc1	doc2	doc3	doc4	doc5	doc6
should 0	0	0	0	0	1	
an 1	1	1	0	0	0	
how 1	0	1	1	0	1	
you 1	1	1	0	0	1	
here 0	0	0	1	0	0	
because 0	0	0	0	0	1	
be 1	1	0	0	0	1	
same 1	1	0	1	0	0	
she 1	1	1	0	1	1	
been 1	0	0	0	1	1	
to 1	0	0	1	0	0	
had 0	1	1	0	1	0	
has 0	1	0	1	0	0	
are 1	1	0	0	1	0	
other 1	0	0	0	0	0	
at 1	0	0	1	0	1	
do 0	1	1	1	1	0	
on 0	0	0	1	0	1	
a 1	0	1	1	0	1	
us 1	0	0	0	0	1	
was 0	1	0	0	1	0	
none 1	0	0	1	0	1	
of 0	1	0	0	1	0	
with 0	1	1	1	0	0	
it 0	0	1	1	1	1	
would 0	0	1	0	1	0	
must 0	1	0	0	1	0	
he 0	0	1	1	1	1	
could 0	1	1	1	1	0	
then 1	0	0	0	1	0	
there 0	0	1	1	0	1	
some 0	1	0	1	1	0	
all 1	0	0	0	0	0	
they 1	1	1	1	1	0	
many 0	0	0	0	1	0	
may 1	0	0	1	0	0	
is 0	1	0	0	0	1	
the 1	1	0	0	1	1	
this 0	0	1	0	0	0	
by 0	0	1	1	0	0	
so 0	0	1	0	1	0	
will 0	0	0	0	1	1	
or 1	1	1	0	0	0	
were 1	1	0	1	0	0	
when 0	1	1	0	1	0	
more 0	0	1	1	0	0	
did 0	0	0	0	0	1	
in 0	0	1	0	1	1	
that 1	0	0	0	1	0	
am 1	1	0	0	0	0	
I 0	1	0	1	1	1	
we 1	0	1	1	0	1	
might 0	0	1	0	1	1	
and 0	1	0	0	1	0	
from 1	1	1	1	0	0	
being 1	1	0	0	1	0	
any 0	0	1	1	0	1	
have 0	0	1	0	0	1	
where 0	0	1	1	1	1	
does 1	1	1	1	0	1	
such 1	1	0	1	1	1	
what 0	1	0	1	0	0	
why 0	0	0	0	1	0	
who 0	0	1	0	1	0	
for 1	0	0	1	0	1	

In [37]:
```
import pandas as pd

# Document strings
d1 = '''being an at from that are a they be or such for does same were she you the us am be
d2 = '''does with had was she or and has are from an could same be is they must you when th
d3 = '''any where he who it so would do there they how it she could with a an have we more
d4 = '''he they more has how for may by same what I here from were a on there at we where d
d5 = '''had many could will some would and I they the must been that such he who she then w
d6 = '''might be such he for he none how did I you we because should is will she at where i

# List of document strings
d = [d1, d2, d3, d4, d5, d6]

# Initialize an empty set for unique words
unique_words = set()

# Add unique words from each document to the set
for i in d:
    unique_words.update(i.split())

# Create an empty DataFrame
df = pd.DataFrame(columns=['Unique Words'] + [f"d{i}" for i in range(1, 7)])

# Iterate over unique words to populate the DataFrame
for word in unique_words:
    occurrences = [1 if word in doc.split() else 0 for doc in documents]
    df.loc[len(df)] = [word] + occurrences

# Display the DataFrame
print(df)
```

```
   Unique Words  d1  d2  d3  d4  d5  d6
0        should   0   0   0   0   0   1
1            an   1   1   1   0   0   0
2           how   1   0   1   1   0   1
3           you   1   1   1   0   0   1
4          here   0   0   0   1   0   0
..          ...  ..  ..  ..  ..  ..  ..
60         such   1   1   0   1   1   1
61         what   0   1   0   1   0   0
62          why   0   0   0   0   1   0
63          who   0   0   1   0   1   0
64          for   1   0   0   1   0   1

[65 rows x 7 columns]
```

In [41]:
```
# df[(df['Unique Words'].str.contains('should')) & (df['Unique Words'].str.contains('for'))

df[(df['Unique Words'].str.contains('should')) & (df['Unique Words'].str.contains('you'))]
```

Out[41]:

Unique Words	d1	d2	d3	d4	d5	d6

In [44]:
```
df[(df['Unique Words'].str.contains('should'))],df[(df['Unique Words'].str.contains('you'))
```

Out[44]:
```
(  Unique Words  d1  d2  d3  d4  d5  d6
 0        should   0   0   0   0   0   1,
   Unique Words  d1  d2  d3  d4  d5  d6
 3           you   1   1   1   0   0   1)
```

In [45]:
```
# DataFrames for comparison
df_should = {'d1': 0, 'd2': 0, 'd3': 0, 'd4': 0, 'd5': 0, 'd6': 1}
df_you = {'d1': 1, 'd2': 1, 'd3': 1, 'd4': 0, 'd5': 0, 'd6': 1}

# Documents where 'should' and 'you' are present
should_docs = [doc for doc, presence in df_should.items() if presence == 1]
you_docs = [doc for doc, presence in df_you.items() if presence == 1]

# Columns where both 'should' and 'you' are present
common_columns = [doc for doc in should_docs if doc in you_docs]

# Display the common columns
print("Columns where both 'should' and 'you' are present:")
print(common_columns)
```

```
Columns where both 'should' and 'you' are present:
['d6']
```

In [49]:
```
query
```

Out[49]:
```
['should', 'you']
```

In [ ]:

In [ ]:

In [ ]:

In [24]:
```
def check_word_presence():
    word = input("Enter the word to check: ")
    presence_in_docs = []
    for i in range(1, 7):
        presence = "present" if word in df['Unique Words'].values and df[f"doc{i}"][df[df['
        presence_in_docs.append((i, presence))
    for doc_number, presence in presence_in_docs:
        print(f"The word '{word}' is {presence} in document {doc_number}.")

# Example usage
check_word_presence()
```

```
Enter the word to check: should
The word 'should' is not present in document 1.
The word 'should' is not present in document 2.
The word 'should' is not present in document 3.
The word 'should' is not present in document 4.
The word 'should' is not present in document 5.
The word 'should' is present in document 6.
```

In [25]:
```
def check_word_presence():
    word = input("Enter the word to check: ")
    for i in range(1, 7):
        if word in df['Unique Words'].values and df[f"doc{i}"][df[df['Unique Words'] == wor
            print(f"The word '{word}' is present in document {i}.")

# Example usage
check_word_presence()
```

```
Enter the word to check: do
The word 'do' is present in document 2.
The word 'do' is present in document 3.
The word 'do' is present in document 4.
The word 'do' is present in document 5.
```

In [ ]:
```
word = input("Enter the word to check: ")
for i in range(1, 7):
    if word in df['Unique Words'].values and df[f"doc{i}"][df[df['Unique Words'] == word].i
        print(f"The word '{word}' is present in document {i}.")
```

In [26]:
```
a = input()
for i in range(1, 7):
    if word in df['Unique Words'].values :
        print("yes")
```

```
Enter the word to check: do
yes
yes
yes
yes
yes
yes
```

In [32]:
```
word = input("Enter the word to check: ")
for i in range(1, 7):
    index = df[df['Unique Words'] == word].index
    if word in df['Unique Words'].values and df.iloc[index[0], i] == 1:
        print(i)
```

```
Enter the word to check: do
2
3
4
5
```

In [35]:
```
import pandas as pd

# Document strings
d1 = '''being an at from that are a they be or such for does same were she you the us am be
d2 = '''does with had was she or and has are from an could same be is they must you when th
d3 = '''any where he who it so would do there they how it she could with a an have we more
d4 = '''he they more has how for may by same what I here from were a on there at we where d
d5 = '''had many could will some would and I they the must been that such he who she then w
d6 = '''might be such he for he none how did I you we because should is will she at where i

# List of document strings
documents = [d1, d2, d3, d4, d5, d6]

# Create a dictionary to store the document names where each word is present
word_presence = {}

# Iterate over each document
for doc_num, doc_content in enumerate(documents, start=1):
    # Split the document content into words
    words = doc_content.split()
    # Iterate over each word in the document
    for word in words:
        # Check if the word is already present in the dictionary
        if word in word_presence:
            # If present, append the current document name to the list
            word_presence[word].append(f"d{doc_num}")
        else:
            # If not present, create a new list with the current document name
            word_presence[word] = [f"d{doc_num}"]

# Create a pandas DataFrame from the word_presence dictionary
df = pd.DataFrame(word_presence.items(), columns=['Word', 'Documents'])

# Display the DataFrame
print(df)
```

```
       Word         Documents
0     being      [d1, d2, d5]
1        an      [d1, d2, d3]
2        at      [d1, d4, d6]
3      from  [d1, d2, d3, d4]
4      that          [d1, d5]
..      ...               ...
60     will          [d5, d6]
61      why              [d5]
62      did              [d6]
63  because              [d6]
64   should              [d6]

[65 rows x 2 columns]
```

In [ ]:

Lab Assignment - 7

In [19]:
```
import re

def tokenize(text):
    # Split the text into words using regular expressions
    words = re.findall(r'\b\w+\b', text.lower())
    return words

d = '''Virender Sehwag (pronunciationⓘ, born 20 October 1978) is a former Indian cricketer

# Tokenize the document
tokens = tokenize(d)
print(tokens)
```

```
['virender', 'sehwag', 'pronunciation', 'born', '20', 'october', '1978', 'is', 'a', 'forme
r', 'indian', 'cricketer', 'who', 'represented', 'india', 'from', '1999', 'to', '2013', 'w
idely', 'regarded', 'as', 'one', 'of', 'the', 'most', 'destructive', 'openers', '1', 'an
d', 'one', 'of', 'the', 'greatest', 'batsman', 'of', 'his', 'era', 'he', 'played', 'for',
'delhi', 'capitals', 'in', 'ipl', 'and', 'delhi', 'and', 'haryana', 'in', 'indian', 'domes
tic', 'cricket', 'he', 'played', 'his', 'first', 'one', 'day', 'international', 'in', '199
9', 'and', 'joined', 'the', 'indian', 'test', 'side', 'in', '2001', '2', 'in', 'april', '2
009', 'sehwag', 'became', 'the', 'first', 'indian', 'to', 'be', 'honoured', 'as', 'the',
'wisden', 'leading', 'cricketer', 'in', 'the', 'world', 'for', 'his', 'performance', 'in',
'2008', '3', 'subsequently', 'becoming', 'the', 'first', 'player', 'of', 'any', 'nationali
ty', 'to', 'retain', 'the', 'award', 'for', '2009', '4', 'he', 'worked', 'as', 'stand', 'i
n', 'captain', 'occasionally', 'during', 'absence', 'of', 'main', 'captain', 'of', 'indi
a', 'also', 'worked', 'as', 'vice', 'captain', 'for', 'indian', 'squad', 'he', 'is', 'form
er', 'captain', 'of', 'delhi', 'daredevils', 'and', 'delhi', 'ranji', 'team', 'during', 'h
is', 'time', 'with', 'india', 'sehwag', 'was', 'a', 'member', 'of', 'the', 'team', 'that',
'was', 'one', 'of', 'the', 'joint', 'winners', 'of', 'the', '2002', 'icc', 'champions', 't
rophy', 'the', 'winners', 'of', 'the', '2007', 't20', 'world', 'cup', 'and', 'the', 'winne
rs', 'of', 'the', '2011', 'cricket', 'world', 'cup', 'during', 'the', '2002', 'icc', 'cham
pions', 'trophy', 'sehwag', 'was', 'the', 'highest', 'run', 'scorer', 'with', '271', 'run
s', 'in', '2023', 'he', 'was', 'inducted', 'into', 'icc', 'cricket', 'hall', 'of', 'fame',
'5']
```

In [20]:
```
from collections import Counter

def calculate_tf(tokens):
    # Count the frequency of each word in the document
    tf = Counter(tokens)
    return tf

# Calculate term frequency for the tokens
term_frequency = calculate_tf(tokens)
print(term_frequency)
```

```
Counter({'the': 17, 'of': 13, 'in': 9, 'and': 6, 'indian': 5, 'he': 5, 'sehwag': 4, 'as':
4, 'one': 4, 'his': 4, 'for': 4, 'delhi': 4, 'captain': 4, 'was': 4, 'india': 3, 'to': 3,
'cricket': 3, 'first': 3, 'world': 3, 'during': 3, 'winners': 3, 'icc': 3, 'is': 2, 'a':
2, 'former': 2, 'cricketer': 2, '1999': 2, 'played': 2, '2009': 2, 'worked': 2, 'team': 2,
'with': 2, '2002': 2, 'champions': 2, 'trophy': 2, 'cup': 2, 'virender': 1, 'pronunciatio
n': 1, 'born': 1, '20': 1, 'october': 1, '1978': 1, 'who': 1, 'represented': 1, 'from': 1,
'2013': 1, 'widely': 1, 'regarded': 1, 'most': 1, 'destructive': 1, 'openers': 1, '1': 1,
'greatest': 1, 'batsman': 1, 'era': 1, 'capitals': 1, 'ipl': 1, 'haryana': 1, 'domestic':
1, 'day': 1, 'international': 1, 'joined': 1, 'test': 1, 'side': 1, '2001': 1, '2': 1, 'ap
ril': 1, 'became': 1, 'be': 1, 'honoured': 1, 'wisden': 1, 'leading': 1, 'performance': 1,
'2008': 1, '3': 1, 'subsequently': 1, 'becoming': 1, 'player': 1, 'any': 1, 'nationality':
1, 'retain': 1, 'award': 1, '4': 1, 'stand': 1, 'occasionally': 1, 'absence': 1, 'main':
1, 'also': 1, 'vice': 1, 'squad': 1, 'daredevils': 1, 'ranji': 1, 'time': 1, 'member': 1,
'that': 1, 'joint': 1, '2007': 1, 't20': 1, '2011': 1, 'highest': 1, 'run': 1, 'scorer':
1, '271': 1, 'runs': 1, '2023': 1, 'inducted': 1, 'into': 1, 'hall': 1, 'fame': 1, '5':
1})
```

In [21]:
```
def find_rarest_word(term_frequency):
    # Find the word with the lowest frequency
    rarest_word = min(term_frequency, key=term_frequency.get)
    return rarest_word

# Find the rarest word
rarest_word = find_rarest_word(term_frequency)
print("Rarest word:", rarest_word)
```

```
Rarest word: virender
```

In [ ]:

In [ ]:

In [25]:
```
# Split the document into sentences
sentences = re.split(r'(?<!\w\.\w.)(?<![A-Z][a-z]\.)(?<=\.|\?)\s', d)

# Tokenize the sentences and find the rarest word
min_word_count = float('inf')
rarest_word = None

for sentence in sentences:
    words = re.findall(r'\b\w+\b', sentence.lower())
    for word in words:
        if words.count(word) < min_word_count:
            min_word_count = words.count(word)
            rarest_word = word

# Print the sentences containing the rarest word
for sentence in sentences:
    if rarest_word in sentence.lower():
        print("rarest word: ",rarest_word)
        print("Sentence containing the rarest word:", sentence.strip())
```

```
rarest word:  virender
Sentence containing the rarest word: Virender Sehwag (pronunciationⓘ, born 20 October 197
8) is a former Indian cricketer who represented India from 1999 to 2013.
```

In [ ]:

In [1]:
```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize, sent_tokenize
from docx import Document
```

In [2]:

```
def read_word_files(file_paths):
    all_content = []
    for file_path in file_paths:
        try:
            doc = Document(file_path)
            text = []
            for paragraph in doc.paragraphs:
                text.append(paragraph.text)
            all_content.append('\n'.join(text))
        except Exception as e:
            print(f"Error reading the Word file '{file_path}': {e}")
    return all_content

word_file_paths = [r"D:\code\NLP\doc\Doc1.docx", r"D:\code\NLP\doc\Doc 2.docx", r"D:\code\N


contents = read_word_files(word_file_paths)

def tokenize_and_remove_stopwords(contents):
    stop_words = set(stopwords.words('english'))
    tokenized_docs = []
    for doc_content in contents:
        words = word_tokenize(doc_content)
        words = [word for word in words if word.lower() not in stop_words]
        sentences = sent_tokenize(doc_content)
        tokenized_docs.append((words, sentences))
    return tokenized_docs

tokenized_docs_no_stopwords = tokenize_and_remove_stopwords(contents)

for i, (words, sentences) in enumerate(tokenized_docs_no_stopwords, start=1):
    print(f"\nTokens for Document {i} after removing stopwords:")
    print("Words:", words)
    print("Sentences:", sentences)
```

Tokens for Document 1 after removing stopwords:
Words: ['Formula', 'One', ',', 'commonly', 'known', 'Formula', '1', 'F1', ',', 'highest',
'class', 'international', 'racing', 'open-wheel', 'single-seater', 'formula', 'racing', 'c
ars', 'sanctioned', 'Fédération', 'Internationale', 'de', "l'Automobile", '(', 'FIA', ')',
'.', 'FIA', 'Formula', 'One', 'World', 'Championship', 'one', 'premier', 'forms', 'racin
g', 'around', 'world', 'since', 'inaugural', 'running', '1950', '.', 'word', 'formula', 'n
ame', 'refers', 'set', 'rules', 'participants', "'", 'cars', 'must', 'conform', '.', 'Form
ula', 'One', 'season', 'consists', 'series', 'races', ',', 'known', 'Grands', 'Prix', '.',
'Grands', 'Prix', 'take', 'place', 'multiple', 'countries', 'continents', 'around', 'worl
d', 'either', 'purpose-built', 'circuits', 'closed', 'public', 'roads', '.', 'point-syste
m', 'used', 'Grands', 'Prix', 'determine', 'two', 'annual', 'World', 'Championships', ':',
'one', 'drivers', ',', 'one', 'constructors', '(', 'teams', ')', '.', 'driver', 'must', 'h
old', 'valid', 'Super', 'Licence', ',', 'highest', 'class', 'racing', 'licence', 'issued',
'FIA', ',', 'races', 'must', 'held', 'grade', 'one', 'tracks', ',', 'highest', 'grade-rati
ng', 'issued', 'FIA', 'tracks', '.']
Sentences: ["Formula One, commonly known as Formula 1 or F1, is the highest class of inter
national racing for open-wheel single-seater formula racing cars sanctioned by the Fédérat
ion Internationale de l'Automobile (FIA).", 'The FIA Formula One World Championship has be
en one of the premier forms of racing around the world since its inaugural running in 195
0.', "The word formula in the name refers to the set of rules to which all participants' c
ars must conform.", 'A Formula One season consists of a series of races, known as Grands P
rix.', 'Grands Prix take place in multiple countries and continents around the world on ei
ther purpose-built circuits or closed public roads.', 'A point-system is used at Grands Pr
ix to determine two annual World Championships: one for the drivers, and one for the const
ructors (the teams).', 'Each driver must hold a valid Super Licence, the highest class of
racing licence issued by the FIA, and the races must be held on grade one tracks, the high
est grade-rating issued by the FIA for tracks.']

Tokens for Document 2 after removing stopwords:
Words: ['track', ',', 'McLaren', 'Williams', 'teams', 'dominated', '1980s', '1990s', '.',
'Brabham', 'also', 'competitive', 'early', 'part', '1980s', ',', 'winning', 'two', 'Driver
s', "'", 'Championships', 'Nelson', 'Piquet', '.', 'Powered', 'Porsche', ',', 'Honda',
',', 'Mercedes-Benz', ',', 'McLaren', 'sixteen', 'championships', '(', 'seven', 'construct
ors', "'", 'nine', 'drivers', "'", ')', 'period', ',', 'Williams', 'used', 'engines', 'For
d', ',', 'Honda', ',', 'Renault', 'also', 'win', 'sixteen', 'titles', '(', 'nine', 'constr
uctors', "'", 'seven', 'drivers', "'", ')', '.', 'rivalry', 'racers', 'Ayrton', 'Senna',
'Alain', 'Prost', 'became', 'F1', "'s", 'central', 'focus', '1988', 'continued', 'Prost',
'retired', 'end', '1993', '.', 'Senna', 'died', '1994', 'San', 'Marino', 'Grand', 'Prix',
'crashing', 'wall', 'exit', 'notorious', 'curve', 'Tamburello', '.', 'FIA', 'worked', 'imp
rove', 'sport', "'s", 'safety', 'standards', 'since', 'weekend', ',', 'Roland', 'Ratzenber
ger', 'also', 'died', 'accident', 'Saturday', 'qualifying', '.', 'driver', 'died', 'injuri
es', 'sustained', 'track', 'wheel', 'Formula', 'One', 'car', '20', 'years', '2014', 'Japan
ese', 'Grand', 'Prix', ',', 'Jules', 'Bianchi', 'collided', 'recovery', 'vehicle', 'aquapl
aning', 'circuit', ',', 'dying', 'nine', 'months', 'later', 'injuries', '.', 'Since', '199
4', ',', 'three', 'track', 'marshals', 'died', ',', 'one', '2000', 'Italian', 'Grand', 'Pr
ix', ',', '[', '31', ']', 'second', '2001', 'Australian', 'Grand', 'Prix', '[', '31', ']',
'third', '2013', 'Canadian', 'Grand', 'Prix', '.']
Sentences: ['On the track, the McLaren and Williams teams dominated the 1980s and 1990s.',
"Brabham were also being competitive during the early part of the 1980s, winning two Drive
rs' Championships with Nelson Piquet.", "Powered by Porsche, Honda, and Mercedes-Benz, McL
aren won sixteen championships (seven constructors' and nine drivers') in that period, whi
le Williams used engines from Ford, Honda, and Renault to also win sixteen titles (nine co
nstructors' and seven drivers').", "The rivalry between racers Ayrton Senna and Alain Pros
t became F1's central focus during 1988 and continued until Prost retired at the end of 19
93.", 'Senna died at the 1994 San Marino Grand Prix after crashing into a wall on the exit
of the notorious curve Tamburello.', "The FIA worked to improve the sport's safety standar
ds since that weekend, during which Roland Ratzenberger also died in an accident during Sa
turday qualifying.", 'No driver died of injuries sustained on the track at the wheel of a
Formula One car for 20 years until the 2014 Japanese Grand Prix, where Jules Bianchi colli
ded with a recovery vehicle after aquaplaning off the circuit, dying nine months later fro
m his injuries.', 'Since 1994, three track marshals have died, one at the 2000 Italian Gra
nd Prix,[31] the second at the 2001 Australian Grand Prix[31] and the third at the 2013 Ca
nadian Grand Prix.']

Tokens for Document 3 after removing stopwords:
Words: ['major', 'rule', 'shake-up', '2014', 'saw', '2.4-litre', 'naturally', 'aspirated',
'V8', 'engines', 'replaced', '1.6-litre', 'turbocharged', 'hybrid', 'power', 'units', '.',
'prompted', 'Honda', 'return', 'sport', '2015', 'championship', "'s", 'fourth', 'power',
'unit', 'manufacturer', '.', 'Mercedes', 'emerged', 'dominant', 'force', 'rule', 'shake-u
p', ',', 'Lewis', 'Hamilton', 'winning', 'championship', 'closely', 'followed', 'main', 'r
ival', 'teammate', ',', 'Nico', 'Rosberg', ',', 'team', 'winning', '16', '19', 'races', 's

```
eason', '.', 'team', 'continued', 'form', 'following', 'two', 'seasons', ',', 'winning',
'16', 'races', '2015', 'taking', 'record', '19', 'wins', '2016', ',', 'Hamilton', 'claimin
g', 'title', 'former', 'year', 'Rosberg', 'winning', 'latter', 'five', 'points', '.', '201
6', 'season', 'also', 'saw', 'new', 'team', ',', 'Haas', ',', 'join', 'grid', ',', 'Max',
'Verstappen', 'became', 'youngest-ever', 'race', 'winner', 'age', '18', 'Spain', '.']
Sentences: ['A major rule shake-up in 2014 saw the 2.4-litre naturally aspirated V8 engine
s replaced by 1.6-litre turbocharged hybrid power units.', "This prompted Honda to return
to the sport in 2015 as the championship's fourth power unit manufacturer.", 'Mercedes eme
rged as the dominant force after the rule shake-up, with Lewis Hamilton winning the champi
onship closely followed by his main rival and teammate, Nico Rosberg, with the team winnin
g 16 out of the 19 races that season.', 'The team continued this form in the following two
seasons, again winning 16 races in 2015 before taking a record 19 wins in 2016, with Hamil
ton claiming the title in the former year and Rosberg winning it in the latter by five poi
nts.', 'The 2016 season also saw a new team, Haas, join the grid, while Max Verstappen bec
ame the youngest-ever race winner at the age of 18 in Spain.']

Tokens for Document 4 after removing stopwords:
Words: ['race', 'begins', 'warm-up', 'lap', ',', 'cars', 'assemble', 'starting', 'grid',
'order', 'qualified', '.', 'lap', 'often', 'referred', 'formation', 'lap', ',', 'cars', 'l
ap', 'formation', 'overtaking', '(', 'although', 'driver', 'makes', 'mistake', 'may', 'reg
ain', 'lost', 'ground', ')', '.', 'warm-up', 'lap', 'allows', 'drivers', 'check', 'conditi
on', 'track', 'car', ',', 'gives', 'tyres', 'chance', 'warm', 'increase', 'traction', 'gri
p', ',', 'also', 'gives', 'pit', 'crews', 'time', 'clear', 'equipment', 'grid', 'race', 's
tart', '.']
Sentences: ['The race begins with a warm-up lap, after which the cars assemble on the star
ting grid in the order they qualified.', 'This lap is often referred to as the formation l
ap, as the cars lap in formation with no overtaking (although a driver who makes a mistake
may regain lost ground).', 'The warm-up lap allows drivers to check the condition of the t
rack and their car, gives the tyres a chance to warm up to increase traction and grip, and
also gives the pit crews time to clear themselves and their equipment from the grid for th
e race start.']

Tokens for Document 5 after removing stopwords:
Words: ['Formula', 'One', 'constructor', 'entity', 'credited', 'designing', 'chassis', 'en
gine', '.', '[', '97', ']', 'designed', 'company', ',', 'company', 'receives', 'sole', 'cr
edit', 'constructor', '(', 'e.g.', ',', 'Ferrari', ')', '.', 'designed', 'different', 'com
panies', ',', 'credited', ',', 'name', 'chassis', 'designer', 'placed', 'engine', 'designe
r', '(', 'e.g.', ',', 'McLaren-Mercedes', ')', '.', 'constructors', 'scored', 'individuall
y', ',', 'even', 'share', 'either', 'chassis', 'engine', 'another', 'constructor', '(',
'e.g.', ',', 'Williams-Ford', ',', 'Williams-Honda', '1983', ')', '.']
Sentences: ['A Formula One constructor is the entity credited for designing the chassis an
d the engine.', '[97] If both are designed by the same company, that company receives sole
credit as the constructor (e.g., Ferrari).', 'If they are designed by different companies,
both are credited, and the name of the chassis designer is placed before that of the engin
e designer (e.g., McLaren-Mercedes).', 'All constructors are scored individually, even if
they share either chassis or engine with another constructor (e.g., Williams-Ford, William
s-Honda in 1983).']

Tokens for Document 6 after removing stopwords:
Words: ['use', 'volunteers', 'integral', 'making', 'maintaining', 'Wikipedia', '.', 'Howev
er', ',', 'even', 'without', 'internet', ',', 'huge', 'complex', 'projects', 'similar', 'n
ature', 'made', 'use', 'volunteers', '.', 'Specifically', ',', 'creation', 'Oxford', 'Engl
ish', 'Dictionary', 'conceived', 'speech', 'London', 'Library', ',', 'Guy', 'Fawkes', 'Da
y', ',', '5', 'November', '1857', ',', 'Richard', 'Chenevix', 'Trench', '.', 'took', '70',
'years', 'complete', '.', 'Dr.', 'Trench', 'envisioned', 'grand', 'new', 'dictionary', 'ev
ery', 'word', 'English', 'language', ',', 'used', 'democratically', 'freely', '.', 'Accord
ing', 'author', 'Simon', 'Winchester', ',', '``', 'undertaking', 'scheme', ',', 'said',
',', 'beyond', 'ability', 'one', 'man', '.', 'peruse', 'English', 'literature', '–', 'com
b', 'London', 'New', 'York', 'newspapers', 'literate', 'magazines', 'journals', '–', 'mus
t', 'instead', "'the", 'combined', 'action', 'many', '.', "'", 'would', 'necessary', 'recr
uit', 'team', '–', 'moreover', ',', 'huge', 'one', '–', 'probably', 'comprising', 'hundred
s', 'hundreds', 'unpaid', 'amateurs', ',', 'working', 'volunteers', '.']
Sentences: ['The use of volunteers was integral in making and maintaining Wikipedia.', 'Ho
wever, even without the internet, huge complex projects of similar nature had made use of
volunteers.', 'Specifically, the creation of the Oxford English Dictionary was conceived w
ith the speech at the London Library, on Guy Fawkes Day, 5 November 1857, by Richard Chene
vix Trench.', 'It took about 70 years to complete.', 'Dr. Trench envisioned a grand new di
ctionary of every word in the English language, and to be used democratically and freel
y.', 'According to author Simon Winchester, "The undertaking of the scheme, he said, was b
eyond the ability of any one man.', "To peruse all of English literature – and to comb the
London and New York newspapers and the most literate of the magazines and journals – must
be instead 'the combined action of many.'", 'It would be necessary to recruit a team – mor
```

```
eover, a huge one – probably comprising hundreds and hundreds of unpaid amateurs, all of t
hem working as volunteers.']
```

In [9]:

```
import numpy as np
from sklearn.feature_extraction.text import TfidfVectorizer
from nltk.tokenize import sent_tokenize, word_tokenize
```

In [10]:
```
def select_content_tfidf(tokenized_docs, num_sentences):
    documents = [' '.join(words) for words, _ in tokenized_docs]

    vectorizer = TfidfVectorizer()
    tfidf_matrix = vectorizer.fit_transform(documents)
    feature_names = vectorizer.get_feature_names_out()
    word_tfidf_means = np.mean(tfidf_matrix, axis=0)
    word_tfidf_means = np.array(word_tfidf_means).reshape(-1)
    top_word_indices = np.argsort(word_tfidf_means)[::-1][:5]

    print("Top 5 words based on TF-IDF scores:")
    for idx in top_word_indices[:5]:
        word = feature_names[idx]
        tfidf_score = word_tfidf_means[idx]
        print(f"{word}: {tfidf_score}")

    selected_sentences = []
    for doc_idx, (words, _) in enumerate(tokenized_docs):
        doc_content = ' '.join(words)
        sentences = sent_tokenize(doc_content)
        sentence_tfidf_scores = []
        for sentence in sentences:
            sentence_words = word_tokenize(sentence.lower())
            sentence_tfidf = 0
            for word_idx in top_word_indices:
                word = feature_names[word_idx]
                if word in sentence_words:
                    word_tfidf = word_tfidf_means[word_idx]
                    sentence_tfidf += word_tfidf
            sentence_tfidf_scores.append(sentence_tfidf)
        top_sentence_indices = np.argsort(sentence_tfidf_scores)[::-1][:num_sentences]
        selected_sentences.extend([sentences[idx] for idx in top_sentence_indices])
    return selected_sentences

selected_sentences_tfidf = select_content_tfidf(tokenized_docs_no_stopwords, num_sentences=
print("\nSelected Sentences based on TF-IDF:")
for i, sentence in enumerate(selected_sentences_tfidf, start=1):
    print(f"{i}. {sentence}")
```

```
Top 5 words based on TF-IDF scores:
one: 0.09446063755636129
lap: 0.09063245606237069
prix: 0.07810972780528723
formula: 0.07234073035134353
chassis: 0.061396642684729705

Selected Sentences based on TF-IDF:
1. Formula One season consists series races , known Grands Prix .
2. point-system used Grands Prix determine two annual World Championships : one drivers ,
one constructors ( teams ) .
3. driver died injuries sustained track wheel Formula One car 20 years 2014 Japanese Grand
Prix , Jules Bianchi collided recovery vehicle aquaplaning circuit , dying nine months lat
er injuries .
4. Since 1994 , three track marshals died , one 2000 Italian Grand Prix , [ 31 ] second 20
01 Australian Grand Prix [ 31 ] third 2013 Canadian Grand Prix .
5. 2016 season also saw new team , Haas , join grid , Max Verstappen became youngest-ever
race winner age 18 Spain .
6. team continued form following two seasons , winning 16 races 2015 taking record 19 wins
2016 , Hamilton claiming title former year Rosberg winning latter five points .
7. warm-up lap allows drivers check condition track car , gives tyres chance warm increase
traction grip , also gives pit crews time clear equipment grid race start .
8. lap often referred formation lap , cars lap formation overtaking ( although driver make
s mistake may regain lost ground ) .
9. Formula One constructor entity credited designing chassis engine .
10. constructors scored individually , even share either chassis engine another constructo
r ( e.g.
11. would necessary recruit team – moreover , huge one – probably comprising hundreds hund
reds unpaid amateurs , working volunteers .
12. According author Simon Winchester , `` undertaking scheme , said , beyond ability one
man .
```

In [ ]:

In [1]:
```
d1 = '''Formula One, commonly known as Formula 1 or F1, is the highest class of internation

A point-system is used at Grands Prix to determine two annual World Championships: one for
```

In [2]:
```
d2 = '''On the track, the McLaren and Williams teams dominated the 1980s and 1990s. Brabham
```

In [3]:
```
d3 = '''A major rule shake-up in 2014 saw the 2.4-litre naturally aspirated V8 engines repl
```

In [4]:
```
d4 = '''The race begins with a warm-up lap, after which the cars assemble on the starting g
```

In [5]:
```
d5 = '''A Formula One constructor is the entity credited for designing the chassis and the
```

In [6]:
```
d6 = '''The use of volunteers was integral in making and maintaining Wikipedia. However, ev
```

In [18]:
```
import nltk
from nltk.corpus import stopwords
from collections import Counter
from nltk.tokenize import word_tokenize, sent_tokenize
```

In [9]:
```
documents = [d1, d2, d3, d4, d5, d6]
```

In [26]:
```
# Function to calculate TF scores for words in a paragraph
def calculate_tf(paragraph):
    words = word_tokenize(paragraph.lower())
    word_counts = Counter(words)
    total_words = len(words)
    return {word: count / total_words for word, count in word_counts.items()}
```

In [27]:
```
# Function to find sentences containing the top word with the highest TF score
def find_sentences_with_top_word(paragraph, top_word):
    sentences_containing_top_word = []
    for sentence in sent_tokenize(paragraph):
        if top_word in word_tokenize(sentence.lower()):
            sentences_containing_top_word.append(sentence)
    return sentences_containing_top_word
```

In [28]:
```
# Iterate through each paragraph
for idx, paragraph in enumerate(documents):
    # Calculate TF scores for words in the paragraph
    tf_scores = calculate_tf(paragraph)
    # Find the word with the highest TF score
    top_word = max(tf_scores, key=tf_scores.get)
    # Find sentences containing the top word
    sentences_with_top_word = find_sentences_with_top_word(paragraph, top_word)
    # Print summary and sentences containing the top word
    print(f"Paragraph {idx + 1}:")
    print("The word with the highest TF score is:", top_word)
    print("Sentences containing the top word:")
    for sentence in sentences_with_top_word:
        print("-", sentence)
    print()
```

Paragraph 1:
The word with the highest TF score is: the
Sentences containing the top word:
- Formula One, commonly known as Formula 1 or F1, is the highest class of international ra
cing for open-wheel single-seater formula racing cars sanctioned by the Fédération Interna
tionale de l'Automobile (FIA).
- The FIA Formula One World Championship has been one of the premier forms of racing aroun
d the world since its inaugural running in 1950.
- The word formula in the name refers to the set of rules to which all participants' cars
must conform.
- Grands Prix take place in multiple countries and continents around the world on either p
urpose-built circuits or closed public roads.
- A point-system is used at Grands Prix to determine two annual World Championships: one f
or the drivers, and one for the constructors (the teams).
- Each driver must hold a valid Super Licence, the highest class of racing licence issued
by the FIA, and the races must be held on grade one tracks, the highest grade-rating issue
d by the FIA for tracks.

Paragraph 2:
The word with the highest TF score is: the
Sentences containing the top word:
- On the track, the McLaren and Williams teams dominated the 1980s and 1990s.
- Brabham were also being competitive during the early part of the 1980s, winning two Driv
ers' Championships with Nelson Piquet.
- The rivalry between racers Ayrton Senna and Alain Prost became F1's central focus during
1988 and continued until Prost retired at the end of 1993.
- Senna died at the 1994 San Marino Grand Prix after crashing into a wall on the exit of t
he notorious curve Tamburello.
- The FIA worked to improve the sport's safety standards since that weekend, during which
Roland Ratzenberger also died in an accident during Saturday qualifying.
- No driver died of injuries sustained on the track at the wheel of a Formula One car for
20 years until the 2014 Japanese Grand Prix, where Jules Bianchi collided with a recovery
vehicle after aquaplaning off the circuit, dying nine months later from his injuries.
- Since 1994, three track marshals have died, one at the 2000 Italian Grand Prix,[31] the
second at the 2001 Australian Grand Prix[31] and the third at the 2013 Canadian Grand Pri
x.

Paragraph 3:
The word with the highest TF score is: the
Sentences containing the top word:
- A major rule shake-up in 2014 saw the 2.4-litre naturally aspirated V8 engines replaced
by 1.6-litre turbocharged hybrid power units.
- This prompted Honda to return to the sport in 2015 as the championship's fourth power un
it manufacturer.
- Mercedes emerged as the dominant force after the rule shake-up, with Lewis Hamilton winn
ing the championship closely followed by his main rival and teammate, Nico Rosberg, with t
he team winning 16 out of the 19 races that season.
- The team continued this form in the following two seasons, again winning 16 races in 201
5 before taking a record 19 wins in 2016, with Hamilton claiming the title in the former y
ear and Rosberg winning it in the latter by five points.
- The 2016 season also saw a new team, Haas, join the grid, while Max Verstappen became th
e youngest-ever race winner at the age of 18 in Spain.

Paragraph 4:
The word with the highest TF score is: the
Sentences containing the top word:
- The race begins with a warm-up lap, after which the cars assemble on the starting grid i
n the order they qualified.
- This lap is often referred to as the formation lap, as the cars lap in formation with no
overtaking (although a driver who makes a mistake may regain lost ground).
- The warm-up lap allows drivers to check the condition of the track and their car, gives
the tyres a chance to warm up to increase traction and grip, and also gives the pit crews
time to clear themselves and their equipment from the grid for the race start.

Paragraph 5:
The word with the highest TF score is: the
Sentences containing the top word:
- A Formula One constructor is the entity credited for designing the chassis and the engin
e.
- [97] If both are designed by the same company, that company receives sole credit as the
constructor (e.g., Ferrari).
- If they are designed by different companies, both are credited, and the name of the chas

sis designer is placed before that of the engine designer (e.g., McLaren-Mercedes).

Paragraph 6:
The word with the highest TF score is: the
Sentences containing the top word:
- The use of volunteers was integral in making and maintaining Wikipedia.
- However, even without the internet, huge complex projects of similar nature had made use of volunteers.
- Specifically, the creation of the Oxford English Dictionary was conceived with the speech at the London Library, on Guy Fawkes Day, 5 November 1857, by Richard Chenevix Trench.
- Dr. Trench envisioned a grand new dictionary of every word in the English language, and to be used democratically and freely.
- According to author Simon Winchester, "The undertaking of the scheme, he said, was beyond the ability of any one man.
- To peruse all of English literature – and to comb the London and New York newspapers and the most literate of the magazines and journals – must be instead 'the combined action of many.'

In [30]:

```python
# Concatenate all paragraphs into a single string
document = ' '.join(documents)

# Function to calculate TF scores for words in the document
def calculate_tf(document):
    words = word_tokenize(document.lower())
    word_counts = Counter(words)
    total_words = len(words)
    return {word: count / total_words for word, count in word_counts.items()}

# Function to find sentences containing the top word with the highest TF score
def find_sentences_with_top_word(document, top_word):
    sentences_containing_top_word = []
    for sentence in sent_tokenize(document):
        if top_word in word_tokenize(sentence.lower()):
            sentences_containing_top_word.append(sentence)
    return sentences_containing_top_word

# Calculate TF scores for words in the document
tf_scores = calculate_tf(document)

# Find the word with the highest TF score
top_word = max(tf_scores, key=tf_scores.get)

# Find sentences containing the top word
sentences_with_top_word = find_sentences_with_top_word(document, top_word)

# Print summary and sentences containing the top word
print("Summary:")
print("The word with the highest TF score is:", top_word)
print("Sentences containing the top word:")
for sentence in sentences_with_top_word:
    print("-", sentence)
```

Summary:
The word with the highest TF score is: the
Sentences containing the top word:
- Formula One, commonly known as Formula 1 or F1, is the highest class of international ra cing for open-wheel single-seater formula racing cars sanctioned by the Fédération Interna tionale de l'Automobile (FIA).
- The FIA Formula One World Championship has been one of the premier forms of racing aroun d the world since its inaugural running in 1950.
- The word formula in the name refers to the set of rules to which all participants' cars must conform.
- Grands Prix take place in multiple countries and continents around the world on either p urpose-built circuits or closed public roads.
- A point-system is used at Grands Prix to determine two annual World Championships: one f or the drivers, and one for the constructors (the teams).
- Each driver must hold a valid Super Licence, the highest class of racing licence issued by the FIA, and the races must be held on grade one tracks, the highest grade-rating issue d by the FIA for tracks.
- On the track, the McLaren and Williams teams dominated the 1980s and 1990s.
- Brabham were also being competitive during the early part of the 1980s, winning two Driv ers' Championships with Nelson Piquet.
- The rivalry between racers Ayrton Senna and Alain Prost became F1's central focus during 1988 and continued until Prost retired at the end of 1993.
- Senna died at the 1994 San Marino Grand Prix after crashing into a wall on the exit of t he notorious curve Tamburello.
- The FIA worked to improve the sport's safety standards since that weekend, during which Roland Ratzenberger also died in an accident during Saturday qualifying.
- No driver died of injuries sustained on the track at the wheel of a Formula One car for 20 years until the 2014 Japanese Grand Prix, where Jules Bianchi collided with a recovery vehicle after aquaplaning off the circuit, dying nine months later from his injuries.
- Since 1994, three track marshals have died, one at the 2000 Italian Grand Prix,[31] the second at the 2001 Australian Grand Prix[31] and the third at the 2013 Canadian Grand Pri x.
- A major rule shake-up in 2014 saw the 2.4-litre naturally aspirated V8 engines replaced by 1.6-litre turbocharged hybrid power units.
- This prompted Honda to return to the sport in 2015 as the championship's fourth power un it manufacturer.
- Mercedes emerged as the dominant force after the rule shake-up, with Lewis Hamilton winn ing the championship closely followed by his main rival and teammate, Nico Rosberg, with t he team winning 16 out of the 19 races that season.
- The team continued this form in the following two seasons, again winning 16 races in 201 5 before taking a record 19 wins in 2016, with Hamilton claiming the title in the former y ear and Rosberg winning it in the latter by five points.
- The 2016 season also saw a new team, Haas, join the grid, while Max Verstappen became th e youngest-ever race winner at the age of 18 in Spain.
- The race begins with a warm-up lap, after which the cars assemble on the starting grid i n the order they qualified.
- This lap is often referred to as the formation lap, as the cars lap in formation with no overtaking (although a driver who makes a mistake may regain lost ground).
- The warm-up lap allows drivers to check the condition of the track and their car, gives the tyres a chance to warm up to increase traction and grip, and also gives the pit crews time to clear themselves and their equipment from the grid for the race start.
- A Formula One constructor is the entity credited for designing the chassis and the engin e.
- [97] If both are designed by the same company, that company receives sole credit as the constructor (e.g., Ferrari).
- If they are designed by different companies, both are credited, and the name of the chas sis designer is placed before that of the engine designer (e.g., McLaren-Mercedes).
- The use of volunteers was integral in making and maintaining Wikipedia.
- However, even without the internet, huge complex projects of similar nature had made use of volunteers.
- Specifically, the creation of the Oxford English Dictionary was conceived with the speec h at the London Library, on Guy Fawkes Day, 5 November 1857, by Richard Chenevix Trench.
- Dr. Trench envisioned a grand new dictionary of every word in the English language, and to be used democratically and freely.
- According to author Simon Winchester, "The undertaking of the scheme, he said, was beyon d the ability of any one man.
- To peruse all of English literature – and to comb the London and New York newspapers and the most literate of the magazines and journals – must be instead 'the combined action of many.'

In [1]:

```
import nltk
from nltk.corpus import stopwords
from nltk.tokenize import word_tokenize

def remove_stopwords(text):
    stop_words = set(stopwords.words('english'))
    word_tokens = word_tokenize(text)
    filtered_text = [word for word in word_tokens if word.lower() not in stop_words]
    return filtered_text

document = "On the track, the McLaren and Williams teams dominated the 1980s and 1990s. Bra
tokenized_document = remove_stopwords(document)
print(tokenized_document)
```

```
['track', ',', 'McLaren', 'Williams', 'teams', 'dominated', '1980s', '1990s', '.', 'Brabha
m', 'also', 'competitive', 'early', 'part', '1980s', ',', 'winning', 'two', 'Drivers',
"'", 'Championships', 'Nelson', 'Piquet', '.', 'Powered', 'Porsche', ',', 'Honda', ',', 'M
ercedes-Benz', ',', 'McLaren', 'sixteen', 'championships', '(', 'seven', 'constructors',
"'", 'nine', 'drivers', "'", ')', 'period', ',', 'Williams', 'used', 'engines', 'Ford',
',', 'Honda', ',', 'Renault', 'also', 'win', 'sixteen', 'titles', '(', 'nine', 'constructo
rs', "'", 'seven', 'drivers', "'", ')', '.', 'rivalry', 'racers', 'Ayrton', 'Senna', 'Alai
n', 'Prost', 'became', 'F1', "'s", 'central', 'focus', '1988', 'continued', 'Prost', 'reti
red', 'end', '1993', '.', 'Senna', 'died', '1994', 'San', 'Marino', 'Grand', 'Prix', 'cras
hing', 'wall', 'exit', 'notorious', 'curve', 'Tamburello', '.', 'FIA', 'worked', 'improv
e', 'sport', "'s", 'safety', 'standards', 'since', 'weekend', ',', 'Roland', 'Ratzenberge
r', 'also', 'died', 'accident', 'Saturday', 'qualifying', '.', 'driver', 'died', 'injurie
s', 'sustained', 'track', 'wheel', 'Formula', 'One', 'car', '20', 'years', '2014', 'Japane
se', 'Grand', 'Prix', ',', 'Jules', 'Bianchi', 'collided', 'recovery', 'vehicle', 'aquapla
ning', 'circuit', ',', 'dying', 'nine', 'months', 'later', 'injuries', '.', 'Since', '199
4', ',', 'three', 'track', 'marshals', 'died', ',', 'one', '2000', 'Italian', 'Grand', 'Pr
ix', ',', '[', '31', ']', 'second', '2001', 'Australian', 'Grand', 'Prix', '[', '31', ']',
'third', '2013', 'Canadian', 'Grand', 'Prix', '.']
```

In [2]:
```
from collections import Counter

def calculate_tf(tokens):
    cleaned_tokens = [''.join(char for char in token if char.isalnum()) for token in tokens
    cleaned_tokens = [token for token in cleaned_tokens if token]  # Remove empty tokens
    tf_scores = Counter(cleaned_tokens)
    total_tokens = len(cleaned_tokens)
    for token in tf_scores:
        tf_scores[token] /= total_tokens
    return tf_scores

tf_scores = calculate_tf(tokenized_document)
sorted_tf_scores = sorted(tf_scores.items(), key=lambda x: x[1], reverse=True)

def print_tf_scores(sorted_tf_scores):
    print("Term              TF Score")
    print("----------------------------------------")
    for term, tf_score in sorted_tf_scores:
        print(f"{term.ljust(20)}{tf_score:.6f}")

print_tf_scores(sorted_tf_scores)
```

```
Term              TF Score
----------------------------------------
Grand               0.035461
Prix                0.035461
died                0.028369
track               0.021277
also                0.021277
nine                0.021277
McLaren             0.014184
Williams            0.014184
1980s               0.014184
Honda               0.014184
sixteen             0.014184
seven               0.014184
constructors        0.014184
drivers             0.014184
Senna               0.014184
Prost               0.014184
s                   0.014184
```

In [6]:
```
def get_sentence_tf(sentence, tf_scores):
    tokens = nltk.word_tokenize(sentence.lower())
    total_words = len(tokens)
    tf_score = sum(tf_scores[word] for word in tokens) / total_words
    return tf_score
```

In [7]:
```
def get_top_sentences(document, tf_scores, num_sentences=3):
    sentences = nltk.sent_tokenize(document)
    sentence_tfs = [(sentence, get_sentence_tf(sentence, tf_scores)) for sentence in senten
    sorted_sentence_tfs = sorted(sentence_tfs, key=lambda x: x[1], reverse=True)[:num_sente
    return [sentence for sentence, _ in sorted_sentence_tfs]

top_sentences = get_top_sentences(document, tf_scores)
def print_top_sentences(top_sentences):
    print("Top Sentences:")
    print("---------------------------------------")
    for i, sentence in enumerate(top_sentences, 1):
        print(f"Sentence {i}: {sentence}")
        print("---------------------------------------")

print_top_sentences(top_sentences)
```

```
Top Sentences:
---------------------------------------
Sentence 1: No driver died of injuries sustained on the track at the wheel of a Formula On
e car for 20 years until the 2014 Japanese Grand Prix, where Jules Bianchi collided with a
recovery vehicle after aquaplaning off the circuit, dying nine months later from his injur
ies.
---------------------------------------
Sentence 2: Powered by Porsche, Honda, and Mercedes-Benz, McLaren won sixteen championship
s (seven constructors' and nine drivers') in that period, while Williams used engines from
Ford, Honda, and Renault to also win sixteen titles (nine constructors' and seven driver
s').
---------------------------------------
Sentence 3: The FIA worked to improve the sport's safety standards since that weekend, dur
ing which Roland Ratzenberger also died in an accident during Saturday qualifying.
---------------------------------------
```

In [8]:
```
top_sentences = get_top_sentences(document, tf_scores)
def print_top_sentences(top_sentences):
    print("Top Sentences:")
    print("---------------------------------------")
    for i, sentence in enumerate(top_sentences, 1):
        print(f"Sentence {i}: {sentence}")
        print("---------------------------------------")

print_top_sentences(top_sentences)
```

```
Top Sentences:
---------------------------------------
Sentence 1: No driver died of injuries sustained on the track at the wheel of a Formula On
e car for 20 years until the 2014 Japanese Grand Prix, where Jules Bianchi collided with a
recovery vehicle after aquaplaning off the circuit, dying nine months later from his injur
ies.
---------------------------------------
Sentence 2: Powered by Porsche, Honda, and Mercedes-Benz, McLaren won sixteen championship
s (seven constructors' and nine drivers') in that period, while Williams used engines from
Ford, Honda, and Renault to also win sixteen titles (nine constructors' and seven driver
s').
---------------------------------------
Sentence 3: The FIA worked to improve the sport's safety standards since that weekend, dur
ing which Roland Ratzenberger also died in an accident during Saturday qualifying.
---------------------------------------
```

In [ ]:

In [4]:
```
from docx import Document
```

In [6]:
```
def read_word_files(file_paths):
    all_content = []
    for file_path in file_paths:
        try:
            doc = Document(file_path)
            text = []
            for paragraph in doc.paragraphs:
                text.append(paragraph.text)
            all_content.append('\n'.join(text))
        except Exception as e:
            print(f"Error reading the Word file '{file_path}': {e}")
    return all_content

word_file_paths = [r"D:\code\NLP\doc\Doc1.docx", r"D:\code\NLP\doc\Doc 2.docx", r"D:\code\N
contents = read_word_files(word_file_paths)
```

In [8]:
```
if contents:
    for i, content in enumerate(contents, start=1):
        print(f"Content of file {i}:")
        print(content)
```

Content of file 1:
Formula One, commonly known as Formula 1 or F1, is the highest class of international racing for open-wheel single-seater formula racing cars sanctioned by the Fédération Internationale de l'Automobile (FIA). The FIA Formula One World Championship has been one of the premier forms of racing around the world since its inaugural running in 1950. The word formula in the name refers to the set of rules to which all participants' cars must conform. A Formula One season consists of a series of races, known as Grands Prix. Grands Prix take place in multiple countries and continents around the world on either purpose-built circuits or closed public roads.

A point-system is used at Grands Prix to determine two annual World Championships: one for the drivers, and one for the constructors (the teams). Each driver must hold a valid Super Licence, the highest class of racing licence issued by the FIA, and the races must be held on grade one tracks, the highest grade-rating issued by the FIA for tracks.
Content of file 2:
On the track, the McLaren and Williams teams dominated the 1980s and 1990s. Brabham were also being competitive during the early part of the 1980s, winning two Drivers' Championships with Nelson Piquet. Powered by Porsche, Honda, and Mercedes-Benz, McLaren won sixteen championships (seven constructors' and nine drivers') in that period, while Williams used engines from Ford, Honda, and Renault to also win sixteen titles (nine constructors' and seven drivers'). The rivalry between racers Ayrton Senna and Alain Prost became F1's central focus during 1988 and continued until Prost retired at the end of 1993. Senna died at the 1994 San Marino Grand Prix after crashing into a wall on the exit of the notorious curve Tamburello. The FIA worked to improve the sport's safety standards since that weekend, during which Roland Ratzenberger also died in an accident during Saturday qualifying. No driver died of injuries sustained on the track at the wheel of a Formula One car for 20 years until the 2014 Japanese Grand Prix, where Jules Bianchi collided with a recovery vehicle after aquaplaning off the circuit, dying nine months later from his injuries. Since 1994, three track marshals have died, one at the 2000 Italian Grand Prix,[31] the second at the 2001 Australian Grand Prix[31] and the third at the 2013 Canadian Grand Prix.
Content of file 3:
A major rule shake-up in 2014 saw the 2.4-litre naturally aspirated V8 engines replaced by 1.6-litre turbocharged hybrid power units. This prompted Honda to return to the sport in 2015 as the championship's fourth power unit manufacturer. Mercedes emerged as the dominant force after the rule shake-up, with Lewis Hamilton winning the championship closely followed by his main rival and teammate, Nico Rosberg, with the team winning 16 out of the 19 races that season. The team continued this form in the following two seasons, again winning 16 races in 2015 before taking a record 19 wins in 2016, with Hamilton claiming the title in the former year and Rosberg winning it in the latter by five points. The 2016 season also saw a new team, Haas, join the grid, while Max Verstappen became the youngest-ever race winner at the age of 18 in Spain.
Content of file 4:
The race begins with a warm-up lap, after which the cars assemble on the starting grid in the order they qualified. This lap is often referred to as the formation lap, as the cars lap in formation with no overtaking (although a driver who makes a mistake may regain lost ground). The warm-up lap allows drivers to check the condition of the track and their car, gives the tyres a chance to warm up to increase traction and grip, and also gives the pit crews time to clear themselves and their equipment from the grid for the race start.
Content of file 5:
A Formula One constructor is the entity credited for designing the chassis and the engine.[97] If both are designed by the same company, that company receives sole credit as the constructor (e.g., Ferrari). If they are designed by different companies, both are credited, and the name of the chassis designer is placed before that of the engine designer (e.g., McLaren-Mercedes). All constructors are scored individually, even if they share either chassis or engine with another constructor (e.g., Williams-Ford, Williams-Honda in 1983).
Content of file 6:
The use of volunteers was integral in making and maintaining Wikipedia. However, even without the internet, huge complex projects of similar nature had made use of volunteers. Specifically, the creation of the Oxford English Dictionary was conceived with the speech at the London Library, on Guy Fawkes Day, 5 November 1857, by Richard Chenevix Trench. It took about 70 years to complete. Dr. Trench envisioned a grand new dictionary of every word in the English language, and to be used democratically and freely. According to author Simon Winchester, "The undertaking of the scheme, he said, was beyond the ability of any one man. To peruse all of English literature – and to comb the London and New York newspapers and the most literate of the magazines and journals – must be instead 'the combined action of many.' It would be necessary to recruit a team – moreover, a huge one – probably comprising hundreds and hundreds of unpaid amateurs, all of them working as volunteers.

In [9]:
```
import nltk
from nltk.tokenize import word_tokenize, sent_tokenize
```

In [13]:
```
def tokenize_documents(contents):
    tokenized_docs = []
    for doc_content in contents:
        words = word_tokenize(doc_content)
        sentences = sent_tokenize(doc_content)
        tokenized_docs.append((words, sentences))
    return tokenized_docs

tokenized_docs = tokenize_documents(contents)
```

In [14]:
```
for i, (words, sentences) in enumerate(tokenized_docs, start=1):
    print(f"\nTokens for Document {i}:")
    print("Words:", words)
    print("Sentences:", sentences)
```

Tokens for Document 1:
Words: ['Formula', 'One', ',', 'commonly', 'known', 'as', 'Formula', '1', 'or', 'F1', ',', 'is', 'the', 'highest', 'class', 'of', 'international', 'racing', 'for', 'open-wheel', 'single-seater', 'formula', 'racing', 'cars', 'sanctioned', 'by', 'the', 'Fédération', 'Internationale', 'de', "l'Automobile", '(', 'FIA', ')', '.', 'The', 'FIA', 'Formula', 'One', 'World', 'Championship', 'has', 'been', 'one', 'of', 'the', 'premier', 'forms', 'of', 'racing', 'around', 'the', 'world', 'since', 'its', 'inaugural', 'running', 'in', '1950', '.', 'The', 'word', 'formula', 'in', 'the', 'name', 'refers', 'to', 'the', 'set', 'of', 'rules', 'to', 'which', 'all', 'participants', "'", 'cars', 'must', 'conform', '.', 'A', 'Formula', 'One', 'season', 'consists', 'of', 'a', 'series', 'of', 'races', ',', 'known', 'as', 'Grands', 'Prix', '.', 'Grands', 'Prix', 'take', 'place', 'in', 'multiple', 'countries', 'and', 'continents', 'around', 'the', 'world', 'on', 'either', 'purpose-built', 'circuits', 'or', 'closed', 'public', 'roads', '.', 'A', 'point-system', 'is', 'used', 'at', 'Grands', 'Prix', 'to', 'determine', 'two', 'annual', 'World', 'Championships', ':', 'one', 'for', 'the', 'drivers', ',', 'and', 'one', 'for', 'the', 'constructors', '(', 'the', 'teams', ')', '.', 'Each', 'driver', 'must', 'hold', 'a', 'valid', 'Super', 'Licence', ',', 'the', 'highest', 'class', 'of', 'racing', 'licence', 'issued', 'by', 'the', 'FIA', ',', 'and', 'the', 'races', 'must', 'be', 'held', 'on', 'grade', 'one', 'tracks', ',', 'the', 'highest', 'grade-rating', 'issued', 'by', 'the', 'FIA', 'for', 'tracks', '.']
Sentences: ["Formula One, commonly known as Formula 1 or F1, is the highest class of international racing for open-wheel single-seater formula racing cars sanctioned by the Fédération Internationale de l'Automobile (FIA).", 'The FIA Formula One World Championship has been one of the premier forms of racing around the world since its inaugural running in 1950.', "The word formula in the name refers to the set of rules to which all participants' cars must conform.", 'A Formula One season consists of a series of races, known as Grands Prix.', 'Grands Prix take place in multiple countries and continents around the world on either purpose-built circuits or closed public roads.', 'A point-system is used at Grands Prix to determine two annual World Championships: one for the drivers, and one for the constructors (the teams).', 'Each driver must hold a valid Super Licence, the highest class of racing licence issued by the FIA, and the races must be held on grade one tracks, the highest grade-rating issued by the FIA for tracks.']

Tokens for Document 2:
Words: ['On', 'the', 'track', ',', 'the', 'McLaren', 'and', 'Williams', 'teams', 'dominated', 'the', '1980s', 'and', '1990s', '.', 'Brabham', 'were', 'also', 'being', 'competitive', 'during', 'the', 'early', 'part', 'of', 'the', '1980s', ',', 'winning', 'two', 'Drivers', "'", 'Championships', 'with', 'Nelson', 'Piquet', '.', 'Powered', 'by', 'Porsche', ',', 'Honda', ',', 'and', 'Mercedes-Benz', ',', 'McLaren', 'won', 'sixteen', 'championships', '(', 'seven', 'constructors', "'", 'and', 'nine', 'drivers', "'", ')', 'in', 'that', 'period', ',', 'while', 'Williams', 'used', 'engines', 'from', 'Ford', ',', 'Honda', ',', 'and', 'Renault', 'to', 'also', 'win', 'sixteen', 'titles', '(', 'nine', 'constructors', "'", 'and', 'seven', 'drivers', "'", ')', '.', 'The', 'rivalry', 'between', 'racers', 'Ayrton', 'Senna', 'and', 'Alain', 'Prost', 'became', 'F1', "'s", 'central', 'focus', 'during', '1988', 'and', 'continued', 'until', 'Prost', 'retired', 'at', 'the', 'end', 'of', '1993', '.', 'Senna', 'died', 'at', 'the', '1994', 'San', 'Marino', 'Grand', 'Prix', 'after', 'crashing', 'into', 'a', 'wall', 'on', 'the', 'exit', 'of', 'the', 'notorious', 'curve', 'Tamburello', '.', 'The', 'FIA', 'worked', 'to', 'improve', 'the', 'sport', "'s", 'safety', 'standards', 'since', 'that', 'weekend', ',', 'during', 'which', 'Roland', 'Ratzenberger', 'also', 'died', 'in', 'an', 'accident', 'during', 'Saturday', 'qualifying', '.', 'No', 'driver', 'died', 'of', 'injuries', 'sustained', 'on', 'the', 'track', 'at', 'the', 'wheel', 'of', 'a', 'Formula', 'One', 'car', 'for', '20', 'years', 'until', 'the', '2014', 'Japanese', 'Grand', 'Prix', ',', 'where', 'Jules', 'Bianchi', 'collided', 'with', 'a', 'recovery', 'vehicle', 'after', 'aquaplaning', 'off', 'the', 'circuit', ',', 'dying', 'nine', 'months', 'later', 'from', 'his', 'injuries', '.', 'Since', '1994', ',', 'three', 'track', 'marshals', 'have', 'died', ',', 'one', 'at', 'the', '2000', 'Italian', 'Grand', 'Prix', ',', '[', '31', ']', 'the', 'second', 'at', 'the', '2001', 'Australian', 'Grand', 'Prix', '[', '31', ']', 'and', 'the', 'third', 'at', 'the', '2013', 'Canadian', 'Grand', 'Prix', '.']
Sentences: ['On the track, the McLaren and Williams teams dominated the 1980s and 1990s.', "Brabham were also being competitive during the early part of the 1980s, winning two Drivers' Championships with Nelson Piquet.", "Powered by Porsche, Honda, and Mercedes-Benz, McLaren won sixteen championships (seven constructors' and nine drivers') in that period, while Williams used engines from Ford, Honda, and Renault to also win sixteen titles (nine constructors' and seven drivers').", "The rivalry between racers Ayrton Senna and Alain Prost became F1's central focus during 1988 and continued until Prost retired at the end of 1993.", 'Senna died at the 1994 San Marino Grand Prix after crashing into a wall on the exit of the notorious curve Tamburello.', "The FIA worked to improve the sport's safety standards since that weekend, during which Roland Ratzenberger also died in an accident during Saturday qualifying.", 'No driver died of injuries sustained on the track at the wheel of a Formula One car for 20 years until the 2014 Japanese Grand Prix, where Jules Bianchi collided with a recovery vehicle after aquaplaning off the circuit, dying nine months later fro

m his injuries.', 'Since 1994, three track marshals have died, one at the 2000 Italian Gra
nd Prix,[31] the second at the 2001 Australian Grand Prix[31] and the third at the 2013 Ca
nadian Grand Prix.']

Tokens for Document 3:
Words: ['A', 'major', 'rule', 'shake-up', 'in', '2014', 'saw', 'the', '2.4-litre', 'natura
lly', 'aspirated', 'V8', 'engines', 'replaced', 'by', '1.6-litre', 'turbocharged', 'hybri
d', 'power', 'units', '.', 'This', 'prompted', 'Honda', 'to', 'return', 'to', 'the', 'spor
t', 'in', '2015', 'as', 'the', 'championship', "'s", 'fourth', 'power', 'unit', 'manufactu
rer', '.', 'Mercedes', 'emerged', 'as', 'the', 'dominant', 'force', 'after', 'the', 'rul
e', 'shake-up', ',', 'with', 'Lewis', 'Hamilton', 'winning', 'the', 'championship', 'close
ly', 'followed', 'by', 'his', 'main', 'rival', 'and', 'teammate', ',', 'Nico', 'Rosberg',
',', 'with', 'the', 'team', 'winning', '16', 'out', 'of', 'the', '19', 'races', 'that', 's
eason', '.', 'The', 'team', 'continued', 'this', 'form', 'in', 'the', 'following', 'two',
'seasons', ',', 'again', 'winning', '16', 'races', 'in', '2015', 'before', 'taking', 'a',
'record', '19', 'wins', 'in', '2016', ',', 'with', 'Hamilton', 'claiming', 'the', 'title',
'in', 'the', 'former', 'year', 'and', 'Rosberg', 'winning', 'it', 'in', 'the', 'latter',
'by', 'five', 'points', '.', 'The', '2016', 'season', 'also', 'saw', 'a', 'new', 'team',
',', 'Haas', ',', 'join', 'the', 'grid', ',', 'while', 'Max', 'Verstappen', 'became', 'th
e', 'youngest-ever', 'race', 'winner', 'at', 'the', 'age', 'of', '18', 'in', 'Spain', '.']
Sentences: ['A major rule shake-up in 2014 saw the 2.4-litre naturally aspirated V8 engine
s replaced by 1.6-litre turbocharged hybrid power units.', "This prompted Honda to return
to the sport in 2015 as the championship's fourth power unit manufacturer.", 'Mercedes eme
rged as the dominant force after the rule shake-up, with Lewis Hamilton winning the champi
onship closely followed by his main rival and teammate, Nico Rosberg, with the team winnin
g 16 out of the 19 races that season.', 'The team continued this form in the following two
seasons, again winning 16 races in 2015 before taking a record 19 wins in 2016, with Hamil
ton claiming the title in the former year and Rosberg winning it in the latter by five poi
nts.', 'The 2016 season also saw a new team, Haas, join the grid, while Max Verstappen bec
ame the youngest-ever race winner at the age of 18 in Spain.']

Tokens for Document 4:
Words: ['The', 'race', 'begins', 'with', 'a', 'warm-up', 'lap', ',', 'after', 'which', 'th
e', 'cars', 'assemble', 'on', 'the', 'starting', 'grid', 'in', 'the', 'order', 'they', 'qu
alified', '.', 'This', 'lap', 'is', 'often', 'referred', 'to', 'as', 'the', 'formation',
'lap', ',', 'as', 'the', 'cars', 'lap', 'in', 'formation', 'with', 'no', 'overtaking',
'(', 'although', 'a', 'driver', 'who', 'makes', 'a', 'mistake', 'may', 'regain', 'lost',
'ground', ')', '.', 'The', 'warm-up', 'lap', 'allows', 'drivers', 'to', 'check', 'the', 'c
ondition', 'of', 'the', 'track', 'and', 'their', 'car', ',', 'gives', 'the', 'tyres', 'a',
'chance', 'to', 'warm', 'up', 'to', 'increase', 'traction', 'and', 'grip', ',', 'and', 'al
so', 'gives', 'the', 'pit', 'crews', 'time', 'to', 'clear', 'themselves', 'and', 'their',
'equipment', 'from', 'the', 'grid', 'for', 'the', 'race', 'start', '.']
Sentences: ['The race begins with a warm-up lap, after which the cars assemble on the star
ting grid in the order they qualified.', 'This lap is often referred to as the formation l
ap, as the cars lap in formation with no overtaking (although a driver who makes a mistake
may regain lost ground).', 'The warm-up lap allows drivers to check the condition of the t
rack and their car, gives the tyres a chance to warm up to increase traction and grip, and
also gives the pit crews time to clear themselves and their equipment from the grid for th
e race start.']

Tokens for Document 5:
Words: ['A', 'Formula', 'One', 'constructor', 'is', 'the', 'entity', 'credited', 'for', 'd
esigning', 'the', 'chassis', 'and', 'the', 'engine', '.', '[', '97', ']', 'If', 'both', 'a
re', 'designed', 'by', 'the', 'same', 'company', ',', 'that', 'company', 'receives', 'sol
e', 'credit', 'as', 'the', 'constructor', '(', 'e.g.', ',', 'Ferrari', ')', '.', 'If', 'th
ey', 'are', 'designed', 'by', 'different', 'companies', ',', 'both', 'are', 'credited',
',', 'and', 'the', 'name', 'of', 'the', 'chassis', 'designer', 'is', 'placed', 'before',
'that', 'of', 'the', 'engine', 'designer', '(', 'e.g.', ',', 'McLaren-Mercedes', ')', '.',
'All', 'constructors', 'are', 'scored', 'individually', ',', 'even', 'if', 'they', 'shar
e', 'either', 'chassis', 'or', 'engine', 'with', 'another', 'constructor', '(', 'e.g.',
',', 'Williams-Ford', ',', 'Williams-Honda', 'in', '1983', ')', '.']
Sentences: ['A Formula One constructor is the entity credited for designing the chassis an
d the engine.', '[97] If both are designed by the same company, that company receives sole
credit as the constructor (e.g., Ferrari).', 'If they are designed by different companies,
both are credited, and the name of the chassis designer is placed before that of the engin
e designer (e.g., McLaren-Mercedes).', 'All constructors are scored individually, even if
they share either chassis or engine with another constructor (e.g., Williams-Ford, William
s-Honda in 1983).']

Tokens for Document 6:
Words: ['The', 'use', 'of', 'volunteers', 'was', 'integral', 'in', 'making', 'and', 'maint
aining', 'Wikipedia', '.', 'However', ',', 'even', 'without', 'the', 'internet', ',', 'hug

```
e', 'complex', 'projects', 'of', 'similar', 'nature', 'had', 'made', 'use', 'of', 'volunte
ers', '.', 'Specifically', ',', 'the', 'creation', 'of', 'the', 'Oxford', 'English', 'Dict
ionary', 'was', 'conceived', 'with', 'the', 'speech', 'at', 'the', 'London', 'Library',
',', 'on', 'Guy', 'Fawkes', 'Day', ',', '5', 'November', '1857', ',', 'by', 'Richard', 'Ch
enevix', 'Trench', '.', 'It', 'took', 'about', '70', 'years', 'to', 'complete', '.', 'D
r.', 'Trench', 'envisioned', 'a', 'grand', 'new', 'dictionary', 'of', 'every', 'word', 'i
n', 'the', 'English', 'language', ',', 'and', 'to', 'be', 'used', 'democratically', 'and',
'freely', '.', 'According', 'to', 'author', 'Simon', 'Winchester', ',', '``', 'The', 'unde
rtaking', 'of', 'the', 'scheme', ',', 'he', 'said', ',', 'was', 'beyond', 'the', 'abilit
y', 'of', 'any', 'one', 'man', '.', 'To', 'peruse', 'all', 'of', 'English', 'literature',
'–', 'and', 'to', 'comb', 'the', 'London', 'and', 'New', 'York', 'newspapers', 'and', 'th
e', 'most', 'literate', 'of', 'the', 'magazines', 'and', 'journals', '–', 'must', 'be', 'i
nstead', "'the", 'combined', 'action', 'of', 'many', '.', "'", 'It', 'would', 'be', 'neces
sary', 'to', 'recruit', 'a', 'team', '–', 'moreover', ',', 'a', 'huge', 'one', '–', 'proba
bly', 'comprising', 'hundreds', 'and', 'hundreds', 'of', 'unpaid', 'amateurs', ',', 'all',
'of', 'them', 'working', 'as', 'volunteers', '.']
Sentences: ['The use of volunteers was integral in making and maintaining Wikipedia.', 'Ho
wever, even without the internet, huge complex projects of similar nature had made use of
volunteers.', 'Specifically, the creation of the Oxford English Dictionary was conceived w
ith the speech at the London Library, on Guy Fawkes Day, 5 November 1857, by Richard Chene
vix Trench.', 'It took about 70 years to complete.', 'Dr. Trench envisioned a grand new di
ctionary of every word in the English language, and to be used democratically and freel
y.', 'According to author Simon Winchester, "The undertaking of the scheme, he said, was b
eyond the ability of any one man.', "To peruse all of English literature – and to comb the
London and New York newspapers and the most literate of the magazines and journals – must
be instead 'the combined action of many.'", 'It would be necessary to recruit a team – mor
eover, a huge one – probably comprising hundreds and hundreds of unpaid amateurs, all of t
hem working as volunteers.']
```

In [11]:
```python
def unique_words_in_documents(tokenized_docs):
    unique_words_per_doc = []
    for words, _ in tokenized_docs:
        unique_words = set(words)
        unique_words_per_doc.append(unique_words)
    return unique_words_per_doc

unique_words_per_doc = unique_words_in_documents(tokenized_docs)
```

In [12]:
```
for i, unique_words in enumerate(unique_words_per_doc, start=1):
    print(f"\nUnique words in Document {i}:")
    print(unique_words)
```

Unique words in Document 1:
{'conform', 'formula', 'racing', 'F1', 'roads', 'teams', 'the', 'valid', 'running', 'consi
sts', 'annual', 'forms', 'sanctioned', 'Internationale', 'premier', 'held', 'been', 'wor
d', 'One', 'name', 'since', 'Each', 'its', 'constructors', "l'Automobile", '1950', 'plac
e', 'continents', 'rules', 'purpose-built', 'is', 'or', 'in', 'multiple', 'open-wheel', 'm
ust', 'used', 'and', 'known', 'on', 'highest', 'World', 'for', 'refers', 'be', 'all', 'Sup
er', 'series', 'de', 'determine', 'driver', 'Championship', 'either', 'grade', 'internatio
nal', 'closed', "'", '1', 'licence', 'by', ':', 'take', 'Grands', 'a', 'which', 'issued',
'public', 'two', 'has', 'single-seater', 'A', 'Prix', 'one', 'hold', 'tracks', 'circuits',
'at', 'The', 'FIA', 'around', 'class', 'races', 'commonly', 'Formula', 'grade-rating',
')', 'inaugural', 'cars', ',', 'drivers', 'set', 'point-system', 'participants', 'Champion
ships', '.', '(', 'as', 'season', 'Fédération', 'to', 'countries', 'of', 'world', 'Licenc
e'}

Unique words in Document 2:
{'1990s', 'Mercedes-Benz', 'engines', 'being', '31', 'sustained', 'Honda', 'teams', 'F1',
'2000', 'years', 'the', 'Bianchi', 'part', 'his', 'have', 'during', 'until', 'nine', 'ca
r', 'wall', 'track', '1993', 'One', 'while', 'circuit', 'crashing', 'Ratzenberger', 'quali
fying', 'since', 'constructors', 'safety', 'improve', 'Powered', 'later', 'worked', 'champ
ionships', 'Tamburello', 'in', 'marshals', 'Alain', 'between', 'sixteen', 'that', 'used',
'and', 'after', 'winning', 'Piquet', "'s", 'on', 'collided', 'Roland', 'for', 'recovery',
'2001', 'third', 'Japanese', 'from', 'months', 'also', 'curve', 'Ford', 'period', 'wheel',
'Jules', 'Grand', 'competitive', 'continued', 'into', 'driver', 'Ayrton', 'early', 'an',
'accident', '2013', "'", 'Renault', 'by', 'notorious', 'with', 'vehicle', 'Brabham', 'a',
'which', 'two', 'Since', 'seven', '1994', 'titles', 'dominated', '2014', 'won', 'William
s', 'Senna', 'Prix', 'one', '1988', ']', 'at', 'three', 'were', 'The', 'On', 'standards',
'FIA', 'injuries', '[', 'Italian', 'died', 'Marino', '1980s', 'Formula', 'Prost', 'Canadia
n', ')', 'Nelson', 'weekend', 'rivalry', 'aquaplaning', 'off', 'Porsche', ',', 'exit', 'dr
ivers', 'racers', 'sport', 'where', 'No', 'retired', 'Championships', '.', '(', 'McLaren',
'win', 'Saturday', 'end', 'Australian', 'became', 'to', 'dying', 'Drivers', 'San', 'centra
l', 'focus', '20', 'second', 'of'}

Unique words in Document 3:
{'Max', 'engines', 'force', 'before', 'Honda', 'the', '18', 'it', 'his', 'latter', 'Lewi
s', 'wins', 'emerged', 'power', 'while', 'followed', 'former', 'teammate', 'unit', 'Nico',
'again', 'age', 'points', 'rule', 'hybrid', 'shake-up', 'in', 'return', 'Verstappen', '2.4
-litre', '16', 'that', 'and', '2015', 'after', 'winning', "'s", 'out', '1.6-litre', 'turbo
charged', 'following', 'claiming', 'record', 'winner', 'also', 'seasons', 'units', 'new',
'closely', 'continued', 'year', 'grid', 'youngest-ever', 'team', 'aspirated', 'by', 'wit
h', 'naturally', 'title', 'a', 'fourth', 'two', 'join', '2014', '19', 'A', 'form', 'race',
'Mercedes', 'at', 'replaced', 'The', 'dominant', 'championship', 'taking', 'races', 'majo
r', 'manufacturer', 'prompted', ',', 'sport', 'saw', 'main', '.', 'Haas', 'Hamilton', 'a
s', 'season', 'rival', 'This', 'became', 'to', 'Spain', 'this', 'V8', 'five', '2016', 'o
f', 'Rosberg'}

Unique words in Document 4:
{'referred', 'who', 'often', 'begins', 'makes', 'time', 'allows', 'start', 'pit', 'groun
d', 'is', 'chance', 'in', 'up', 'the', ')', 'with', 'regain', 'condition', 'their', 'start
ing', 'grip', 'and', 'cars', 'a', ',', 'after', 'order', 'drivers', 'which', 'car', 'on',
'traction', 'although', 'may', 'formation', 'track', 'gives', 'increase', 'mistake', '.',
'(', 'themselves', 'for', 'as', 'qualified', 'clear', 'equipment', 'from', 'assemble', 'lo
st', 'they', 'also', 'tyres', 'This', 'race', 'to', 'driver', 'warm', 'warm-up', 'lap', 'n
o', 'overtaking', 'check', 'crews', 'The', 'grid', 'of'}

Unique words in Document 5:
{'[', 'before', 'are', 'e.g.', 'entity', 'different', 'scored', 'is', 'or', 'if', 'eithe
r', 'Formula', 'in', 'the', 'credit', 'by', 'If', 'receives', 'of', 'even', ')', 'both',
'with', 'credited', 'chassis', 'that', 'designing', 'Ferrari', 'Williams-Ford', 'and',
',', 'All', '.', 'One', '(', 'for', 'companies', 'A', 'engine', 'as', 'sole', '1983', 'nam
e', 'Williams-Honda', 'company', 'they', ']', 'same', 'constructor', 'individually', 'anot
her', 'share', 'constructors', 'McLaren-Mercedes', 'designer', 'designed', '97', 'placed'}

Unique words in Document 6:
{'magazines', 'Oxford', 'Trench', 'them', 'years', 'the', 'action', 'ability', 'making',
'envisioned', 'To', 'word', 'beyond', 'many', 'amateurs', 'grand', 'Fawkes', 'Day', 'ever
y', 'moreover', '5', 'without', 'comprising', 'he', 'Library', 'complete', 'in', 'unpaid',
'maintaining', 'even', 'similar', 'Dictionary', 'speech', 'literature', 'York', 'used', 'n
ature', 'must', 'and', 'According', 'freely', 'on', '```', 'scheme', 'creation', 'any', 'b
e', 'November', 'combined', '70', 'Specifically', 'all', 'would', 'volunteers', 'newspaper
s', 'projects', 'new', 'took', 'Wikipedia', 'working', 'Simon', 'team', 'about', 'democrat
ically', 'internet', "'", 'by', 'had', 'with', 'a', 'London', 'man', 'complex', 'was', 'ne

```
cessary', 'peruse', 'author', '1857', 'one', "'the", 'huge', 'at', 'Winchester', 'dictiona
ry', 'The', 'It', 'undertaking', 'New', 'Chenevix', 'Dr.', 'Guy', 'journals', 'However',
'probably', 'English', ',', 'language', 'most', 'hundreds', 'Richard', '–', 'instead',
'.', 'literate', 'made', 'as', 'integral', 'conceived', 'to', 'comb', 'use', 'recruit', 'o
f', 'said'}
```

In [15]:
```python
def combine_unique_words(unique_words_per_doc):
    combined_unique_words = set()
    for unique_words in unique_words_per_doc:
        combined_unique_words.update(unique_words)
    return combined_unique_words

combined_unique_words = combine_unique_words(unique_words_per_doc)
```

In [16]:
```python
print("\nCombined unique words from all documents:")
print(combined_unique_words)
```

```
Combined unique words from all documents:
{'1990s', 'conform', 'Mercedes-Benz', 'formula', 'Max', 'force', 'before', 'often', 'magaz
ines', 'Oxford', 'start', '31', 'pit', 'them', 'racing', 'teams', 'valid', '2000', 'year
s', 'if', 'receives', 'action', 'it', 'part', 'latter', 'ability', 'both', 'Williams-For
d', 'until', 'forms', 'nine', 'order', 'sanctioned', 'traction', 'Lewis', 'car', 'wall',
'envisioned', 'premier', 'track', 'word', 'power', 'One', 'many', 'circuit', 'followed',
'former', 'sole', 'teammate', 'name', 'crashing', 'amateurs', 'grand', 'Fawkes', 'tyres',
'Day', 'again', 'qualifying', 'since', 'Each', 'its', "l'Automobile", '1950', 'designer',
'97', 'place', 'he', 'shake-up', 'later', 'are', 'allows', 'complete', 'or', 'in', 'Alai
n', 'return', 'similar', 'Dictionary', '2.4-litre', 'between', 'multiple', 'open-wheel',
'that', 'York', 'must', 'used', 'winning', 'freely', 'out', '1.6-litre', 'claiming', 'ma
y', 'collided', 'any', 'for', 'record', 'engine', '2001', 'combined', 'Specifically', 'fro
m', 'all', 'months', 'lost', 'also', 'series', 'seasons', 'volunteers', 'Ford', 'no', 'per
iod', 'wheel', 'de', 'Grand', 'closely', 'competitive', 'year', 'driver', 'youngest-ever',
'Championship', 'either', 'Ayrton', 'team', 'closed', 'an', 'accident', 'about', '2013',
'democratically', 'internet', 'Renault', 'aspirated', 'grip', 'starting', 'Grands', 'a',
'which', 'two', 'man', 'seven', '1994', 'titles', 'dominated', 'mistake', 'necessary', 'pe
ruse', 'Senna', '1857', 'one', 'hold', '1988', 'they', ']', 'race', 'huge', 'lap', 'indivi
dually', 'share', 'were', 'dictionary', 'standards', 'FIA', 'It', 'championship', 'aroun
d', 'taking', 'New', 'Italian', 'Marino', 'Dr.', 'commonly', 'Formula', 'Prost', 'If', 'Ca
nadian', ')', 'However', 'weekend', 'rivalry', 'aquaplaning', 'their', 'probably', 'off',
'prompted', 'exit', ',', 'English', 'most', 'hundreds', 'set', 'sport', 'Richard', 'saw',
'point-system', 'participants', 'No', 'retired', 'main', '(', 'win', 'Haas', 'Saturday',
'end', 'as', 'qualified', 'season', 'Fédération', 'Williams-Honda', 'assemble', 'became',
'comb', 'same', 'dying', 'San', 'V8', 'central', 'five', 'focus', 'recruit', '20', '2016',
'injuries', 'of', 'world', 'Licence', 'Rosberg', 'instead', 'engines', 'being', 'Trench',
'ground', 'sustained', 'Honda', 'chance', 'roads', 'F1', 'up', 'the', 'credit', 'Bianchi',
'18', 'running', 'consists', 'annual', 'his', 'have', 'Ferrari', 'during', 'making', 'To',
'Internationale', 'held', 'although', 'wins', '1993', 'emerged', 'been', 'beyond', 'whil
e', '1983', 'unit', 'Nico', 'Ratzenberger', 'every', 'warm-up', 'overtaking', 'another',
'age', 'constructors', 'points', 'moreover', '5', 'rule', 'without', 'safety', 'improve',
'comprising', 'continents', 'hybrid', 'Powered', 'Library', 'rules', 'worked', 'championsh
ips', 'entity', 'purpose-built', 'is', 'Tamburello', 'marshals', 'unpaid', 'maintaining',
'Verstappen', 'even', '16', 'speech', 'sixteen', 'regain', 'credited', 'chassis', 'literat
ure', 'nature', 'and', '2015', 'after', 'According', 'Piquet', 'known', "'s", 'on', 'highe
st', 'turbocharged', 'following', 'formation', 'World', '``', 'Roland', 'scheme', 'creatio
n', 'refers', 'be', 'recovery', 'November', 'third', '70', 'Japanese', 'winner', 'would',
'Super', 'warm', 'units', 'curve', 'newspapers', 'new', 'projects', 'Jules', 'continued',
'into', 'McLaren-Mercedes', 'determine', 'grid', 'took', 'Wikipedia', 'grade', 'working',
'international', 'Simon', 'who', 'time', 'early', 'makes', 'e.g.', "'", '1', 'different',
'licence', 'by', 'notorious', ':', 'with', 'had', 'vehicle', 'take', 'Brabham', 'naturall
y', 'title', 'designing', 'London', 'issued', 'fourth', 'public', 'Since', 'join', 'comple
x', 'All', 'has', '2014', 'single-seater', 'gives', 'themselves', 'won', '19', 'was', 'com
panies', 'A', 'Williams', 'equipment', 'form', 'author', 'Prix', 'tracks', "'the", 'circui
ts', 'Mercedes', 'constructor', 'check', 'at', 'three', 'replaced', 'Winchester', 'The',
'On', 'crews', 'designed', 'dominant', 'class', '[', 'referred', 'undertaking', 'begins',
'Chenevix', 'died', 'scored', 'races', '1980s', 'major', 'Guy', 'grade-rating', 'manufactu
rer', 'Nelson', 'journals', 'condition', 'inaugural', 'cars', 'Porsche', 'language', 'driv
ers', 'racers', 'where', '–', 'Championships', 'increase', '.', 'McLaren', 'literate', 'ma
de', 'Hamilton', 'clear', 'rival', 'company', 'Australian', 'integral', 'This', 'to', 'con
ceived', 'use', 'Drivers', 'Spain', 'this', 'countries', 'second', 'said', 'placed'}
```

In [22]:

```
import string

def preprocess_text(text):
    text = text.translate(str.maketrans('', '', string.punctuation))
    text = text.replace('[', '').replace(']', '').replace('-', '')
    return text

preprocessed_contents = [preprocess_text(content) for content in contents]
preprocessed_tokenized_docs = [word_tokenize(content) for content in preprocessed_contents]

def get_unique_words(tokenized_docs):
    unique_words = set()
    for words in tokenized_docs:
        unique_words.update(words)
    return unique_words

unique_words_after_preprocessing = get_unique_words(preprocessed_tokenized_docs)

print("Unique words after preprocessing:")
print(unique_words_after_preprocessing)
```

```
Unique words after preprocessing:
{'1990s', 'conform', 'before', 'Max', 'often', 'Oxford', 'teams', '2000', 'if', 'it', 'lat
ter', 'both', 'traction', 'envisioned', 'car', 'wall', 'premier', 'track', 'youngestever',
'power', 'many', 'circuit', 'former', 'name', 'grand', 'Fawkes', 'tyres', 'qualifying', 's
ince', '1950', 'designer', 'later', 'are', 'complete', 'in', 'Alain', 'return', 'between',
'multiple', 'that', 'York', 'must', 'used', 'winning', 'freely', 'out', 'may', 'collided',
'WilliamsHonda', 'any', 'openwheel', 'record', 'engine', '2001', 'from', 'months', 'lost',
'also', 'seasons', 'Ford', 'no', 'de', 'Grand', 'closely', 'driver', 'Championship', 'tea
m', 'an', 'about', '2013', 'democratically', 'internet', 'aspirated', 'shakeup', 'lAutomob
ile', 'grip', 'starting', 'two', 'mistake', 'graderating', 'necessary', 'peruse', '1857',
'they', 'lap', 'share', 'purposebuilt', 'It', 'championship', 'taking', '24litre', 'Formul
a', 'weekend', 'rivalry', 'aquaplaning', 'their', 'probably', 'sport', 'saw', 'win', 'Haa
s', 'Saturday', 'Fédération', 'assemble', 'comb', 'same', 'dying', 'focus', '2016', 'injur
ies', 'world', 'Rosberg', 'Licence', 'instead', 'engines', 'ground', 'sustained', 'Honda',
'chance', 'roads', 'F1', 'the', 'credit', 'Bianchi', '18', 'annual', 'his', 'Ferrari', 'du
ring', 'held', 'wins', '1993', 'emerged', 'been', 'while', '1983', 'Ratzenberger', 'ever
y', 'overtaking', 'age', 'moreover', '5', 'rule', 'without', 'improve', 'Powered', 'Librar
y', 'championships', 'WilliamsFord', 'maintaining', 'Verstappen', 'Dr', 'sixteen', 'regai
n', 'credited', 'and', '2015', 'after', 'Piquet', 'known', 'highest', 'turbocharged', 'fol
lowing', 'World', 'scheme', 'creation', 'refers', 'recovery', 'November', 'third', '70',
'winner', 'would', 'warm', 'units', 'curve', 'new', 'projects', 'Jules', 'continued', 'int
o', 'determine', 'grid', 'Wikipedia', 'working', 'Simon', 'time', 'early', '1', 'licence',
'notorious', 'with', 'vehicle', 'take', 'Brabham', 'naturally', 'title', 'designing', 'iss
ued', 'fourth', 'Since', 'complex', 'All', '19', 'was', 'A', 'Williams', 'form', 'author',
'tracks', 'at', 'three', 'replaced', 'Winchester', 'The', 'On', 'crews', 'engine97', 'desi
gned', 'dominant', 'class', 'referred', 'undertaking', 'begins', 'died', 'races', 'major',
'manufacturer', 'journals', 'cars', 'language', 'where', 'McLaren', 'Hamilton', 'clear',
'rival', 'Australian', 'integral', 'use', 'Spain', 'countries', 'second', 'said', 'formul
a', 'force', 'magazines', 'start', 'pit', 'them', 'racing', 'valid', 'years', 'receives',
'action', 'part', 'ability', 'until', 'forms', 'nine', 'order', 'sanctioned', 'Lewis', 'wo
rd', 'One', 'followed', 'sole', 'teammate', 'amateurs', 'crashing', 'Day', 'again', 'Eac
h', 'its', 'place', 'he', 'allows', 'or', 'similar', 'Dictionary', 'claiming', 'for', 'com
bined', 'sports', 'Specifically', 'all', 'series', 'volunteers', 'period', 'wheel', 'compe
titive', 'year', 'either', 'Ayrton', 'closed', 'accident', 'Renault', 'Grands', 'a', 'whic
h', 'man', 'McLarenMercedes', 'seven', '1994', 'titles', 'dominated', 'warmup', 'singlesea
ter', 'Senna', 'one', 'hold', '1988', 'race', 'huge', 'individually', 'were', 'dictionar
y', 'standards', 'FIA', 'around', 'New', 'Italian', 'Marino', 'commonly', 'Prost', 'If',
'Canadian', 'However', 'off', 'prompted', 'exit', 'English', 'most', 'hundreds', 'set', 'R
ichard', 'No', 'participants', 'retired', 'main', 'end', 'as', 'qualified', 'season', 'e
g', 'became', 'San', 'V8', 'central', 'five', '16litre', 'recruit', '20', 'of', 'being',
'Trench', 'up', 'F1s', 'running', 'consists', 'have', 'making', 'To', 'Internationale', 'a
lthough', 'beyond', 'unit', 'Nico', 'another', 'constructors', 'points', 'safety', 'compri
sing', 'continents', 'hybrid', 'rules', 'worked', 'entity', 'is', 'Tamburello', 'marshal
s', 'unpaid', 'Prix31', 'even', '16', 'speech', 'literature', 'chassis', 'nature', 'Accord
ing', 'on', 'MercedesBenz', 'formation', 'Roland', 'be', 'Japanese', 'Super', 'newspaper
s', 'took', 'grade', 'international', 'who', 'makes', 'different', 'by', 'had', 'pointsyst
em', 'London', 'public', 'join', 'has', '2014', 'gives', 'themselves', 'won', 'companies',
'equipment', 'Prix', 'circuits', 'Mercedes', 'constructor', 'check', 'Chenevix', 'scored',
'1980s', 'Guy', 'Nelson', 'condition', 'inaugural', 'Porsche', 'drivers', 'racers', 'Champ
ionships', 'increase', 'literate', 'made', 'company', 'This', 'to', 'conceived', 'Driver
s', 'this', 'placed'}
```

In [31]:
```
unique_words_list = sorted(list(unique_words_after_preprocessing))
```

In [43]:
```
import pandas as pd
```

In [56]:
```
df = pd.DataFrame(unique_words_list, columns=["Word"])

for doc in range(1, 7):
    df[f"Doc{doc}"] = df['Word'].apply(lambda word: 1 if word in preprocessed_tokenized_doc
```

In [57]:
```
print(unique_words_list)
```

```
['1', '16', '16litre', '18', '1857', '19', '1950', '1980s', '1983', '1988', '1990s', '199
3', '1994', '20', '2000', '2001', '2013', '2014', '2015', '2016', '24litre', '5', '70',
'A', 'According', 'Alain', 'All', 'Australian', 'Ayrton', 'Bianchi', 'Brabham', 'Canadia
n', 'Championship', 'Championships', 'Chenevix', 'Day', 'Dictionary', 'Dr', 'Drivers', 'Ea
ch', 'English', 'F1', 'F1s', 'FIA', 'Fawkes', 'Ferrari', 'Ford', 'Formula', 'Fédération',
'Grand', 'Grands', 'Guy', 'Haas', 'Hamilton', 'Honda', 'However', 'If', 'Internationale',
'It', 'Italian', 'Japanese', 'Jules', 'Lewis', 'Library', 'Licence', 'London', 'Marino',
'Max', 'McLaren', 'McLarenMercedes', 'Mercedes', 'MercedesBenz', 'Nelson', 'New', 'Nico',
'No', 'November', 'On', 'One', 'Oxford', 'Piquet', 'Porsche', 'Powered', 'Prix', 'Prix31',
'Prost', 'Ratzenberger', 'Renault', 'Richard', 'Roland', 'Rosberg', 'San', 'Saturday', 'Se
nna', 'Simon', 'Since', 'Spain', 'Specifically', 'Super', 'Tamburello', 'The', 'This', 'T
o', 'Trench', 'V8', 'Verstappen', 'Wikipedia', 'Williams', 'WilliamsFord', 'WilliamsHond
a', 'Winchester', 'World', 'York', 'a', 'ability', 'about', 'accident', 'action', 'after',
'again', 'age', 'all', 'allows', 'also', 'although', 'amateurs', 'an', 'and', 'annual', 'a
nother', 'any', 'aquaplaning', 'are', 'around', 'as', 'aspirated', 'assemble', 'at', 'auth
or', 'be', 'became', 'been', 'before', 'begins', 'being', 'between', 'beyond', 'both', 'b
y', 'car', 'cars', 'central', 'championship', 'championships', 'chance', 'chassis', 'chec
k', 'circuit', 'circuits', 'claiming', 'class', 'clear', 'closed', 'closely', 'collided',
'comb', 'combined', 'commonly', 'companies', 'company', 'competitive', 'complete', 'comple
x', 'comprising', 'conceived', 'condition', 'conform', 'consists', 'constructor', 'constru
ctors', 'continents', 'continued', 'countries', 'crashing', 'creation', 'credit', 'credite
d', 'crews', 'curve', 'de', 'democratically', 'designed', 'designer', 'designing', 'determ
ine', 'dictionary', 'died', 'different', 'dominant', 'dominated', 'driver', 'drivers', 'du
ring', 'dying', 'early', 'eg', 'either', 'emerged', 'end', 'engine', 'engine97', 'engine
s', 'entity', 'envisioned', 'equipment', 'even', 'every', 'exit', 'five', 'focus', 'follow
ed', 'following', 'for', 'force', 'form', 'formation', 'former', 'forms', 'formula', 'four
th', 'freely', 'from', 'gives', 'grade', 'graderating', 'grand', 'grid', 'grip', 'ground',
'had', 'has', 'have', 'he', 'held', 'highest', 'his', 'hold', 'huge', 'hundreds', 'hybri
d', 'if', 'improve', 'in', 'inaugural', 'increase', 'individually', 'injuries', 'instead',
'integral', 'international', 'internet', 'into', 'is', 'issued', 'it', 'its', 'join', 'jou
rnals', 'known', 'lAutomobile', 'language', 'lap', 'later', 'latter', 'licence', 'literat
e', 'literature', 'lost', 'made', 'magazines', 'main', 'maintaining', 'major', 'makes', 'm
aking', 'man', 'manufacturer', 'many', 'marshals', 'may', 'mistake', 'months', 'moreover',
'most', 'multiple', 'must', 'name', 'naturally', 'nature', 'necessary', 'new', 'newspaper
s', 'nine', 'no', 'notorious', 'of', 'off', 'often', 'on', 'one', 'openwheel', 'or', 'orde
r', 'out', 'overtaking', 'part', 'participants', 'period', 'peruse', 'pit', 'place', 'plac
ed', 'points', 'pointsystem', 'power', 'premier', 'probably', 'projects', 'prompted', 'pub
lic', 'purposebuilt', 'qualified', 'qualifying', 'race', 'racers', 'races', 'racing', 'rec
eives', 'record', 'recovery', 'recruit', 'referred', 'refers', 'regain', 'replaced', 'reti
red', 'return', 'rival', 'rivalry', 'roads', 'rule', 'rules', 'running', 'safety', 'said',
'same', 'sanctioned', 'saw', 'scheme', 'scored', 'season', 'seasons', 'second', 'series',
'set', 'seven', 'shakeup', 'share', 'similar', 'since', 'singleseater', 'sixteen', 'sole',
'speech', 'sport', 'sports', 'standards', 'start', 'starting', 'sustained', 'take', 'takin
g', 'team', 'teammate', 'teams', 'that', 'the', 'their', 'them', 'themselves', 'they', 'th
ird', 'this', 'three', 'time', 'title', 'titles', 'to', 'took', 'track', 'tracks', 'tracti
on', 'turbocharged', 'two', 'tyres', 'undertaking', 'unit', 'units', 'unpaid', 'until', 'u
p', 'use', 'used', 'valid', 'vehicle', 'volunteers', 'wall', 'warm', 'warmup', 'was', 'wee
kend', 'were', 'wheel', 'where', 'which', 'while', 'who', 'win', 'winner', 'winning', 'win
s', 'with', 'without', 'won', 'word', 'worked', 'working', 'world', 'would', 'year', 'year
s', 'youngestever']
```

In [58]:
```
df
```

Out[58]:

	Word	Doc1	Doc2	Doc3	Doc4	Doc5	Doc6
0	1	1	0	0	0	0	0
1	16	0	0	1	0	0	0
2	16litre	0	0	1	0	0	0
3	18	0	0	1	0	0	0
4	1857	0	0	0	0	0	1
...	...	...	...	...	...	...	...
437	world	1	0	0	0	0	0
438	would	0	0	0	0	0	1
439	year	0	0	1	0	0	0
440	years	0	1	0	0	0	1
441	youngestever	0	0	1	0	0	0

442 rows × 7 columns

In [64]:
```
def binary_search_query(tokenized_docs, query_word):
    matching_docs = []
    for i, (words, _) in enumerate(tokenized_docs, start=1):
        if query_word in words:
            matching_docs.append(i)
    return matching_docs

query_word = "race"
result = binary_search_query(tokenized_docs, query_word)
print(f"Documents containing the word '{query_word}': {result}")
```

```
Documents containing the word 'race': [3, 4]
```

In [67]:
```
def binary_search_query(tokenized_docs, query_words):
    matching_docs = []
    for i, (words, _) in enumerate(tokenized_docs, start=1):
        if all(word in words for word in query_words):
            matching_docs.append(i)
    return matching_docs

query_words = ["forms", "formula"]
result = binary_search_query(tokenized_docs, query_words)
print(f"Documents containing all the words '{', '.join(query_words)}': {result}")
```

```
Documents containing all the words 'forms, formula': [1]
```

In [68]:
```
def binary_search_query(tokenized_docs, query_words, not_word):
    matching_docs = []
    for i, (words, _) in enumerate(tokenized_docs, start=1):
        if all(word in words for word in query_words) and not_word not in words:
            matching_docs.append(i)
    return matching_docs

query_words = ["forms", "formula"]
not_word = "grand"
result = binary_search_query(tokenized_docs, query_words, not_word)
print(f"Documents containing all the words '{', '.join(query_words)}' but not '{not_word}':
```

```
Documents containing all the words 'forms, formula' but not 'grand': [1]
```

In [70]:
```
def binary_search_query(tokenized_docs, query_words, not_words):
    matching_docs = []
    for i, (words, _) in enumerate(tokenized_docs, start=1):
        if all(word in words for word in query_words) and not any(not_word in words for not
            matching_docs.append(i)
    return matching_docs

query_words = ["forms", "formula"]
not_words = ["track", "hundreds"]
result = binary_search_query(tokenized_docs, query_words, not_words)
print(f"Documents containing all the words '{', '.join(query_words)}' but not any of '{', '
```

```
Documents containing all the words 'forms, formula' but not any of 'track, hundreds': [1]
```

In [ ]: