In [1]:
```
import nltk
```

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

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

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

In [3]:
```
#Creating frequency distribution of words using nltk
from nltk.tokenize import sent_tokenize
from nltk.tokenize import word_tokenize
from sklearn.feature_extraction.text import CountVectorizer
text = """Achievers are not afraid of Challenges, rather they relish them,
        thrive in them, use them. Challenges makes is stronger.
        Challenges makes us uncomfortable. If you get comfortable with uncomfort th
        Challenge the challenge. """
#Tokenize the sentences from the text corpus
tokenized_text=sent_tokenize(text)#using CountVectorizer and removing stopwords in
cvl= CountVectorizer(lowercase=True,stop_words='english')#fitting the tonized sene
text_counts=cvl.fit_transform(tokenized_text)# printing the vocabulary and the fre
print(cvl.vocabulary_)
print(text_counts.toarray())
```

```
{'achievers': 0, 'afraid': 1, 'challenges': 3, 'relish': 7, 'thrive': 9, 'use': 1
2, 'makes': 6, 'stronger': 8, 'uncomfortable': 11, 'comfortable': 4, 'uncomfort':
10, 'grow': 5, 'challenge': 2}
[[1 1 0 1 0 0 0 1 0 1 0 0 1]
 [0 0 0 1 0 0 1 0 1 0 0 0 0]
 [0 0 0 1 0 0 1 0 0 0 0 1 0]
 [0 0 0 0 1 1 0 0 0 0 1 0 0]
 [0 0 2 0 0 0 0 0 0 0 0 0 0]]
```

In [4]:
```
import collections
import pandas as pd
import numpy as np

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
```

In [5]:
```
doc = "India is my country. India is very beautiful country."
count_vec = CountVectorizer()
count_occurs = count_vec.fit_transform([doc])
count_occur_df = pd.DataFrame((count, word) for word, count in zip(count_occurs.to
                    count_vec.get_feature_names_out()))
count_occur_df.columns = ['Word', 'Count']
count_occur_df.sort_values('Count', ascending=False)
count_occur_df.head()
```

Out[5]:

	Word	Count
0	beautiful	1
1	country	2
2	india	2
3	is	2
4	my	1

In [6]:
```
doc = "India is my country. India is very beautiful country."
norm_count_vec = TfidfVectorizer(use_idf=False, norm='l2')
norm_count_occurs = norm_count_vec. fit_transform( [doc] )
norm_count_occur_df = pd.DataFrame((count, word) for word, count in zip(
    norm_count_occurs.toarray().tolist()[0], norm_count_vec.get_feature_names_out(
norm_count_occur_df.columns = ['Word', 'Count']
norm_count_occur_df.sort_values('Count', ascending=False, inplace=True)
norm_count_occur_df.head()
```

Out[6]:

	Word	Count
1	country	0.516398
2	india	0.516398
3	is	0.516398
0	beautiful	0.258199
4	my	0.258199

In [7]:
```
doc = "India is my country. India is very beautiful country."
tfidf_vec = TfidfVectorizer()
tfidf_count_occurs = tfidf_vec.fit_transform([doc])
tfidf_count_occur_df = pd.DataFrame((count, word) for word, count in zip(
    tfidf_count_occurs.toarray().tolist()[0], tfidf_vec.get_feature_names_out()))
tfidf_count_occur_df.columns = ['Word', 'Count']
tfidf_count_occur_df.sort_values('Count', ascending=True, inplace=True)
tfidf_count_occur_df.head()
```

Out[7]:

	Word	Count
0	beautiful	0.258199
4	my	0.258199
5	very	0.258199
1	country	0.516398
2	india	0.516398

In [8]:
```
import pandas as pd
```

In [9]:
```
!wget https://raw.githubusercontent.com/PICT-NLP/BE-NLP-Elective/main/2-Embeddings/
```

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

In [10]:
```
df = pd.read_csv('data.csv')
df.head()
```

Out[10]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Ma
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Tun
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxu
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxu
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	

In [11]:
```
df['Maker Model']= df['Make']+ " " + df['Model']
```

In [12]:
```
df1 = df[['Engine Fuel Type','Transmission Type','Driven_Wheels', 'Market Category
df2 = df1.apply(lambda x: ','.join(x.astype(str)), axis=1)
df_clean = pd.DataFrame({'clean': df2})
sent = [row.split(',') for row in df_clean['clean']]
```

In [13]:
```
df_clean
```

Out[13]:

	clean
0	premium unleaded (required),MANUAL,rear wheel ...
1	premium unleaded (required),MANUAL,rear wheel ...
2	premium unleaded (required),MANUAL,rear wheel ...
3	premium unleaded (required),MANUAL,rear wheel ...
4	premium unleaded (required),MANUAL,rear wheel ...
...	...
11909	premium unleaded (required),AUTOMATIC,all whee...
11910	premium unleaded (required),AUTOMATIC,all whee...
11911	premium unleaded (required),AUTOMATIC,all whee...
11912	premium unleaded (recommended),AUTOMATIC,all w...
11913	regular unleaded,AUTOMATIC,front wheel drive,L...

11914 rows × 1 columns

In [14]:
```
from gensim.models.word2vec import Word2Vec
```

In [15]:
```
model = Word2Vec(sent, min_count=1,vector_size= 50,workers=3, window =3, sg= 1)
```

In [16]:
```
model.save("word2vec.model")
```

In [17]:
```
model = Word2Vec.load("word2vec.model")
```

In [18]:
```
model.wv['Toyota Camry']
```

Out[18]:
```
array([ 0.0255812 ,  0.13900828,  0.03715063, -0.10744484, -0.08373349,
       -0.17542394, -0.01428106,  0.23986633, -0.12642862, -0.02938929,
        0.00610763,  0.00171292,  0.10102752, -0.04844876, -0.05860265,
        0.15094078,  0.11549113,  0.24968036, -0.10165373, -0.27455732,
       -0.07546954, -0.00415546,  0.2097979 ,  0.06391793,  0.18079066,
        0.01044053, -0.04835675,  0.3443316 , -0.03039352,  0.00251802,
       -0.04621544,  0.02360438,  0.06433985, -0.0023059 ,  0.0633148 ,
       -0.0951896 ,  0.15698262, -0.03316173,  0.01716618,  0.00770351,
        0.04372074, -0.02992754, -0.20013988,  0.10589921,  0.3074159 ,
       -0.02980877, -0.01941405, -0.11001477, -0.015744  ,  0.03643061],
      dtype=float32)
```

In [19]:
```
sims = model.wv.most_similar('Toyota Camry', topn=10)
sims
```

Out[19]:
```
[('Nissan Altima', 0.9789924621582031),
 ('Suzuki Aerio', 0.97333824634552),
 ('Nissan Sentra', 0.9725769758224487),
 ('Chevrolet Cruze', 0.9711400866508484),
 ('Toyota Avalon', 0.9710570573806763),
 ('Mazda 6', 0.9701946377754211),
 ('Oldsmobile Alero', 0.9683876633644104),
 ('Oldsmobile Eighty-Eight Royale', 0.9677125811576843),
 ('Hyundai Azera', 0.9655715227127075),
 ('Nissan Cube', 0.9648975133895874)]
```

In [20]:
```
model.wv.similarity('Toyota Camry', 'Mazda 6')
```

Out[20]:
```
0.9701946
```

In [21]:
```
model.wv.similarity('Dodge Dart', 'Mazda 6')
```

Out[21]:
```
0.9827326
```