In [1]:
```
pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (1.4.4)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: python-dateutil>=2.8.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2020.1 in /usr/local/lib/python3.9/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.8.1->pandas) (1.15.0)
```

In [2]:
```
pip install sklearn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: sklearn in /usr/local/lib/python3.9/dist-packages (0.0.post1)
```

In [ ]:
```
pip install scikit-learn
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: scikit-learn in /usr/local/lib/python3.9/dist-packages (1.2.2)
Requirement already satisfied: threadpoolctl>=2.0.0 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: joblib>=1.1.1 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.1.1)
Requirement already satisfied: scipy>=1.3.2 in /usr/local/lib/python3.9/dist-packages (from scikit-learn) (1.10.1)
```

In [3]:
```
import nltk
```

In [4]:
```
nltk.download('punkt')
```

```
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]   Package punkt is already up-to-date!
```

Out[4]:
```
True
```

In [5]:
```
#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 then
        Challenge the challenge. """
#Tokenize the sentences from the text corpus
tokenized_text=sent_tokenize(text)#using CountVectorizer and removing stopwords in eng
cvl= CountVectorizer(lowercase=True,stop_words='english')#fitting the tonized senetnec
text_counts=cvl.fit_transform(tokenized_text)# printing the vocabulary and the frequer
```

```
print(cvl.vocabulary_)
print(text_counts.toarray())
```

```
{'achievers': 0, 'afraid': 1, 'challenges': 3, 'relish': 7, 'thrive': 9, 'use': 12,
'makes': 6, 'stronger': 8, 'uncomfortable': 11, 'comfortable': 4, 'uncomfort': 10, 'g
row': 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 [6]:
```
import collections
import pandas as pd
import numpy as np

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

In [7]:
```
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.toarra
                    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[7]:

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

In [8]:
```
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[8]:

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

In [9]:
```
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[9]:

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

In [10]:
```
!pip install --upgrade gensim
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: gensim in /usr/local/lib/python3.9/dist-packages (4.3.1)
Requirement already satisfied: smart-open>=1.8.1 in /usr/local/lib/python3.9/dist-packages (from gensim) (6.3.0)
Requirement already satisfied: numpy>=1.18.5 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.22.4)
Requirement already satisfied: scipy>=1.7.0 in /usr/local/lib/python3.9/dist-packages (from gensim) (1.10.1)
```

In [11]:
```
import pandas as pd
```

In [12]:
```
!wget https://raw.githubusercontent.com/PICT-NLP/BE-NLP-Elective/main/2-Embeddings/dat
```

```
--2023-03-16 05:53:45--  https://raw.githubusercontent.com/PICT-NLP/BE-NLP-Elective/main/2-Embeddings/data.csv
Resolving raw.githubusercontent.com (raw.githubusercontent.com)... 185.199.108.133, 185.199.109.133, 185.199.110.133, ...
Connecting to raw.githubusercontent.com (raw.githubusercontent.com)|185.199.108.133|:443... connected.
HTTP request sent, awaiting response... 200 OK
Length: 1475504 (1.4M) [text/plain]
Saving to: ‘data.csv.2’

data.csv.2          100%[===================>]   1.41M  --.-KB/s    in 0.05s

2023-03-16 05:53:46 (25.6 MB/s) - ‘data.csv.2’ saved [1475504/1475504]
```

In [13]:
```
df = pd.read_csv('data.csv')
df.head()
```

Out[13]:

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

In [14]:
```
df['Maker Model']= df['Make']+ " " + df['Model']
```

In [15]:
```
df1 = df[['Engine Fuel Type','Transmission Type','Driven_Wheels', 'Market Category','V
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 [16]:
```
df_clean
```

Out[16]:

	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 [17]:
```
from gensim.models.word2vec import Word2Vec
```

In [18]:
```
model = Word2Vec(sent, min_count=1,vector_size= 50,workers=3, window =3, sg= 1)
```

In [19]:
```
model.save("word2vec.model")
```

In [20]:
```
model = Word2Vec.load("word2vec.model")
```

In [21]:
```
model.wv['Toyota Camry']
```

Out[21]:
```
array([ 8.9228517e-05,  7.3498771e-02, -2.3057928e-02, -4.7667548e-02,
       -4.0306769e-02, -1.9859192e-01,  3.8488999e-02,  2.4063455e-01,
       -1.5224655e-01, -1.0690487e-01, -2.7456619e-02, -1.1472496e-02,
        7.2121747e-02, -1.2351338e-02, -4.4147927e-02,  1.7149018e-01,
        1.3655110e-01,  2.6667419e-01, -1.4331250e-01, -3.6577389e-01,
        5.7174638e-02,  2.7550142e-02,  2.7431577e-01,  1.6144255e-01,
        1.1508633e-01, -1.0168314e-02, -1.6284121e-02,  3.1293055e-01,
       -6.7235008e-02,  2.0680863e-02,  8.1909493e-02,  7.4196860e-02,
        5.0731484e-02, -3.6126714e-02,  5.6436330e-02, -1.1329097e-01,
        2.3158364e-01,  2.7895425e-02,  7.5678818e-02,  1.0964664e-01,
        1.0817333e-01, -7.7248402e-02, -1.5139697e-01,  3.4829028e-02,
        3.2538468e-01,  1.2758501e-01,  1.0050910e-02, -1.5367827e-01,
        8.1078731e-02, -8.0028936e-02], dtype=float32)
```

In [22]:
```
sims = model.wv.most_similar('Toyota Camry', topn=10)
sims
```

Out[22]:
```
[('Ford Fusion', 0.9835054874420166),
 ('Nissan Altima', 0.9826465249061584),
 ('Volvo S80', 0.9805996417999268),
 ('Volvo S70', 0.9792745113372803),
 ('Pontiac G6', 0.9759504199028015),
 ('Chevrolet Malibu', 0.9737828373908997),
 ('Oldsmobile Eighty-Eight Royale', 0.9735535979270935),
 ('Oldsmobile Alero', 0.9732433557510376),
 ('Hyundai Sonata', 0.9720818996429443),
 ('Kia Optima Hybrid', 0.9717768430709839)]
```

In [23]:
```
model.wv.similarity('Toyota Camry', 'Mazda 6')
```

Out[23]:
```
0.95703727
```

In [24]:
```
model.wv.similarity('Dodge Dart', 'Mazda 6')
```

Out[24]:
```
0.98113453
```