In [1]:
```
# Akshata NLP_02
```

In [2]:
```
# Perform bag-of-words approach (count occurrence, normalized count occurrence), TF-IDF on
# data.Create embeddings using Word2Vec.

# Word Embedding
```

In [3]:
```
pip install pandas
```

```
Requirement already satisfied: pandas in c:\users\tech bazaar\anaconda3\lib\site-packages (1.3.4)
Requirement already satisfied: python-dateutil>=2.7.3 in c:\users\tech bazaar\anaconda3\lib\site-packages (from pandas) (2.8.2)
Requirement already satisfied: pytz>=2017.3 in c:\users\tech bazaar\anaconda3\lib\site-packages (from pandas) (2020.1)
Requirement already satisfied: numpy>=1.17.3 in c:\users\tech bazaar\anaconda3\lib\site-packages (from pandas) (1.22.4)
Requirement already satisfied: six>=1.5 in c:\users\tech bazaar\anaconda3\lib\site-packages (from python-dateutil>=2.7.3->pandas) (1.16.0)
Note: you may need to restart the kernel to use updated packages.
```

In [4]:
```
pip install sklearn
```

```
Requirement already satisfied: sklearn in c:\users\tech bazaar\anaconda3\lib\site-packages (0.0)
Requirement already satisfied: scikit-learn in c:\users\tech bazaar\anaconda3\lib\site-packages (from sklearn) (0.24.2)
Requirement already satisfied: numpy>=1.13.3 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.22.4)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn->sklearn) (3.1.0)
Requirement already satisfied: scipy>=0.19.1 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.7.1)
Requirement already satisfied: joblib>=0.11 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn->sklearn) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [5]:
```
pip install scikit-learn
```

```
Requirement already satisfied: scikit-learn in c:\users\tech bazaar\anaconda3\lib\site-packages (0.24.2)
Requirement already satisfied: numpy>=1.13.3 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn) (1.22.4)
Requirement already satisfied: scipy>=0.19.1 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn) (1.7.1)
Requirement already satisfied: threadpoolctl>=2.0.0 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn) (3.1.0)
Requirement already satisfied: joblib>=0.11 in c:\users\tech bazaar\anaconda3\lib\site-packages (from scikit-learn) (1.1.0)
Note: you may need to restart the kernel to use updated packages.
```

In [6]:
```
# Bag of Words (BOW)
```

In [7]:
```
from nltk.tokenize import word_tokenize
from nltk.tokenize import sent_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 un
tokenized_text = sent_tokenize(text)
cvl = CountVectorizer(lowercase=True,stop_words='english')
text_counts = cvl.fit_transform(tokenized_text)
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, '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 [8]:
```
# Count Occurrence
```

In [9]:
```
import collections
import pandas as pd
import numpy as np

from sklearn.feature_extraction.text import CountVectorizer, TfidfVectorizer
doc = "India is my country. India is a 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.toarray().tolist()[0],count_vec.get_feature_names()))
count_occur_df.columns = ['Word','Count']
count_occur_df.sort_values('Count',ascending = False)
count_occur_df.head()
```

Out[9]:

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

In [10]:
```
# TF-IDF
```

In [11]:
```
doc = "India is my country. India is a 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()))
tfidf_count_occur_df.columns = ['Word','Count']
tfidf_count_occur_df.sort_values('Count',ascending = False , inplace=True)
tfidf_count_occur_df.head()
```

Out[11]:

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

In [12]:
```
# Word2Vec
```

In [13]:
```
pip install gensim
```

```
Requirement already satisfied: gensim in c:\users\tech bazaar\anaconda3\lib\site-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\tech bazaar\anaconda3\lib\site-packages (from gensim) (1.22.4)
Requirement already satisfied: scipy>=1.7.0 in c:\users\tech bazaar\anaconda3\lib\site-packages (from gensim) (1.7.1)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\tech bazaar\anaconda3\lib\site-packages (from gensim) (6.4.0)
Note: you may need to restart the kernel to use updated packages.
```

In [14]:
```
!pip install --upgrade gensim
```

```
Requirement already satisfied: gensim in c:\users\tech bazaar\anaconda3\lib\site-packages (4.3.2)
Requirement already satisfied: numpy>=1.18.5 in c:\users\tech bazaar\anaconda3\lib\site-packages (from gensim) (1.22.4)
Requirement already satisfied: smart-open>=1.8.1 in c:\users\tech bazaar\anaconda3\lib\site-packages (from gensim) (6.4.0)
Requirement already satisfied: scipy>=1.7.0 in c:\users\tech bazaar\anaconda3\lib\site-packages (from gensim) (1.7.1)
```

In [15]:
```
import pandas as pd
df = pd.read_csv('data.csv')
df.head()
```

Out[15]:

	Make	Model	Year	Engine Fuel Type	Engine HP	Engine Cylinders	Transmission Type	Driven_Wheels	Number of Doors	Market Category	Vehicle Size	Vehicle Style	highway MPG	city mpg	Popularity	MSRP
0	BMW	1 Series M	2011	premium unleaded (required)	335.0	6.0	MANUAL	rear wheel drive	2.0	Factory Tuner,Luxury,High-Performance	Compact	Coupe	26	19	3916	46135
1	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Convertible	28	19	3916	40650
2	BMW	1 Series	2011	premium unleaded (required)	300.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,High-Performance	Compact	Coupe	28	20	3916	36350
3	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury,Performance	Compact	Coupe	28	18	3916	29450
4	BMW	1 Series	2011	premium unleaded (required)	230.0	6.0	MANUAL	rear wheel drive	2.0	Luxury	Compact	Convertible	28	18	3916	34500

In [16]:
```
df['Maker_Model']= df['Make']+ " " + df['Model']
```

In [17]:
```
df1 = df[['Engine Fuel Type','Transmission Type','Driven_Wheels','Market Category','Vehicle Size', 'Vehicle Style', 'Maker_Model']]

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 [18]:
```
from gensim.models.word2vec import Word2Vec
```

In [19]:
```
model = Word2Vec(sent, min_count=1,vector_size= 50,workers=3, window =3, sg = 1)
```

In [20]:
```
model.save("word2vec.model")
```

In [21]:
```
model = Word2Vec.load("word2vec.model")
```

In [22]:
```
model.wv['Toyota Camry']
```

Out[22]:
```
array([-0.02961558,  0.10300414,  0.00875859, -0.09137409, -0.07950881,
       -0.2255967 , -0.0246627 ,  0.28155595, -0.10072266, -0.08841565,
        0.08007437,  0.03715292,  0.10704777, -0.02026604, -0.02575339,
        0.16276057,  0.13623777,  0.2638806 , -0.11732008, -0.2860225 ,
       -0.04365759, -0.07523099,  0.23747516,  0.05622908,  0.17819187,
       -0.01858381, -0.04224301,  0.36844185, -0.04709727, -0.03083538,
        0.06603304,  0.04594645,  0.02543521,  0.02866244,  0.08821486,
       -0.10684751,  0.1623007 , -0.03510394,  0.03319214,  0.08149256,
        0.06184779, -0.03666685, -0.23906158,  0.09023587,  0.30693412,
        0.02985471, -0.03609032, -0.1394142 ,  0.0067737 ,  0.03754365],
      dtype=float32)
```

In [24]:
```
sims = model.wv.most_similar('Toyota Camry', topn=10)
sims
```

Out[24]:
```
[('Suzuki Kizashi', 0.9879509210586548),
 ('Ford Fusion', 0.9874153137207031),
 ('Hyundai Sonata', 0.9850292801856995),
 ('Nissan Altima', 0.9833672046661377),
 ('Toyota Avalon', 0.9821192622184753),
 ('Kia Optima', 0.981264054775238),
 ('Dodge Dart', 0.9808627367019653),
 ('Suzuki Verona', 0.9799851775169373),
 ('Subaru Legacy', 0.9782137870788574),
 ('Pontiac G6', 0.9781390428543091)]
```

In [25]:
```
model.wv.similarity('Toyota Camry','Mazda 6')
```

Out[25]:
```
0.9733996
```

In [26]:
```
model.wv.similarity('Dodge Dart','Mazda 6')
```

Out[26]:
```
0.9632021
```