SENTIMENTAL ANALYSIS :

PAGE 71

In [ ]:

In [ ]:

In [ ]:

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

```
C:\Users\my pc\anaconda3\lib\site-packages\numpy\_distributor_init.py:30: UserWarning: loaded more than 1 DLL from .
libs:
C:\Users\my pc\anaconda3\lib\site-packages\numpy\.libs\libopenblas.FB5AE2TYXYH2IJRDKGDGQ3XBKLKTF43H.gfortran-win_amd
64.dll
C:\Users\my pc\anaconda3\lib\site-packages\numpy\.libs\libopenblas64__v0.3.23-gcc_10_3_0.dll
  warnings.warn("loaded more than 1 DLL from .libs:"
C:\Users\my pc\anaconda3\lib\site-packages\scipy\__init__.py:155: UserWarning: A NumPy version >=1.18.5 and <1.25.0
is required for this version of SciPy (detected version 1.25.0
  warnings.warn(f"A NumPy version >={np_minversion} and <{np_maxversion}"
```

In the 'nltk' download 'vader_lexicon' :

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

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

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

In [3]:
```
from nltk.sentiment.vader import SentimentIntensityAnalyzer
```

In [4]:
```
sid = SentimentIntensityAnalyzer()
```

Sid polarity :

page 72

In [5]:
```
a = "This is a Good Movie"
```

In [6]:
```
# Here Negative, positive, compound :
# We are going to get it here the scores of particular Reviews on particular Movie or e.t.c
```

In [7]:
```
# This Particular Working Environment is Considered as 'VECTOR',
# Each and Every Word Considered as 'VECTOR'. This Particular VECTOR We need to utilize :
```

In [8]:
```
sid.polarity_scores(a)
```

Out[8]:
```
{'neg': 0.0, 'neu': 0.508, 'pos': 0.492, 'compound': 0.4404}
```

In [9]:
```
b = "This Was the best, most awesome movie Ever Made"
```

In [10]:
```
sid.polarity_scores(b)
```

Out[10]:
```
{'neg': 0.0, 'neu': 0.449, 'pos': 0.551, 'compound': 0.8622}
```

In [11]:
```
c = "This is a Worst Movie, that has ever disgraced the Screen"
```

In [12]:
```
sid.polarity_scores(c)
```

Out[12]:
```
{'neg': 0.47, 'neu': 0.53, 'pos': 0.0, 'compound': -0.7964}
```

In [13]:
```
d = "Too Worst Movie"
```

In [14]:
```
sid.polarity_scores(d)
```

Out[14]:
```
{'neg': 0.672, 'neu': 0.328, 'pos': 0.0, 'compound': -0.6249}
```

In [15]:
```
# Here, Where eadh and every word we considered as an a 'VECTOR',
# And we are going to identify 'Near by Words' for that particular 'VECTOR' by 'importing Spacy library'
```

In [16]:
```
import spacy
```

In [17]:
```
# In Spacy - Each and Everything We are Calling as an a 'VECTOR',
# So, Now we are Calling particular 'English Library' :

# The Size of an a particular Dictionary (or) This ENGLISH LIBRARY has 300*678000 e.t.c
```

In [18]:
```
nlp = spacy.load('en_core_web_lg')
```

```
C:\Users\my pc\anaconda3\lib\site-packages\spacy\util.py:887: UserWarning: [W095] Model 'en_core_web_lg' (3.3.0) was
trained with spaCy v3.3 and may not be 100% compatible with the current version (3.5.4). If you see errors or degrad
ed performance, download a newer compatible model or retrain your custom model with the current spaCy version. For m
ore details and available updates, run: python -m spacy validate
  warnings.warn(warn_msg)
```

In [19]:
```
# Means, Now we are 'Comparing' with an a 'Nearest words'
```

In [20]:
```
nlp(u'lion').vector
```

Out[20]:
```
array([ 1.8963e-01, -4.0309e-01,  3.5350e-01, -4.7907e-01, -4.3311e-01,
        2.3857e-01,  2.6962e-01,  6.4332e-02,  3.0767e-01,  1.3712e+00,
       -3.7582e-01, -2.2713e-01, -3.5657e-01, -2.5355e-01,  1.7543e-02,
        3.3962e-01,  7.4723e-02,  5.1226e-01, -3.9759e-01,  5.1333e-03,
       -3.0929e-01,  4.8911e-02, -1.8610e-01, -4.1702e-01, -8.1639e-01,
       -1.6908e-01, -2.6246e-01, -1.5983e-02,  1.2479e-01, -3.7276e-02,
       -5.7125e-01, -1.6296e-01,  1.2376e-01, -5.5464e-02,  1.3244e-01,
        2.7519e-02,  1.2592e-01, -3.2722e-01, -4.9165e-01, -3.5559e-01,
       -3.0630e-01,  6.1185e-02, -1.6932e-01, -6.2405e-02,  6.5763e-01,
       -2.7925e-01, -3.0450e-03, -2.2400e-02, -2.8015e-01, -2.1975e-01,
       -4.3188e-01,  3.9864e-02, -2.2102e-01, -4.2693e-02,  5.2748e-02,
        2.8726e-01,  1.2315e-01, -2.8662e-02,  7.8294e-02,  4.6754e-01,
       -2.4589e-01, -1.1064e-01,  7.2250e-02, -9.4980e-02, -2.7548e-01,
       -5.4097e-01,  1.2823e-01, -8.2408e-02,  3.1035e-01, -6.3394e-02,
       -7.3755e-01, -5.4992e-01,  9.9999e-02, -2.0758e-01, -3.9674e-02,
        2.0664e-01, -9.7557e-02, -3.7092e-01,  2.7901e-01, -6.2218e-01,
       -1.0280e-01,  2.3271e-01,  4.3838e-01,  3.2445e-02, -2.9866e-01,
       -7.3611e-02,  7.1594e-01,  1.4241e-01,  2.7770e-01, -3.9892e-01,
        3.6656e-02,  1.5759e-01,  8.2014e-02, -5.7343e-01,  3.5457e-01,
        2.2491e-01, -6.2699e-01, -8.8106e-02,  2.4361e-01,  3.8533e-01,
       -1.4083e-01,  1.7691e-01,  7.0897e-02,  1.7951e-01, -4.5907e-01,
       -8.2120e-01, -2.6631e-02,  6.2549e-02,  4.2415e-01, -8.9630e-02,
       -2.4654e-01,  1.4156e-01,  4.0187e-01, -4.1232e-01,  8.4516e-02,
       -1.0626e-01,  7.3145e-01,  1.9217e-01,  1.4240e-01,  2.8511e-01,
       -2.9454e-01, -2.1948e-01,  9.0460e-01, -1.9098e-01, -1.0340e+00,
       -1.5754e-01, -1.1964e-01,  4.9888e-01, -1.0624e+00, -3.2820e-01,
       -1.1232e-02, -7.9482e-01,  3.7275e-01, -6.8710e-03, -2.5772e-01,
       -4.7005e-01, -4.1387e-01, -6.4089e-02, -2.8033e-01, -4.0778e-02,
       -2.4866e+00,  6.2494e-03, -1.0210e-02,  1.2752e-01,  3.4965e-01,
       -1.2571e-01,  3.1570e-01,  4.1926e-01,  2.0056e-01, -5.5984e-01,
       -2.2801e-01,  1.2012e-01, -2.0518e-03, -8.9764e-02, -8.0373e-02,
        1.1969e-02, -2.6978e-01,  3.4829e-01,  7.3664e-03, -1.1137e-01,
        6.3410e-01,  3.8449e-01, -6.2248e-01,  4.1145e-02,  2.5922e-01,
        6.5811e-01, -4.9548e-01, -1.3030e-01, -3.8279e-01,  1.1156e-01,
       -4.3085e-01,  3.4473e-01,  2.7109e-02, -2.5108e-01, -2.8011e-01,
        2.1662e-01,  3.2660e-01,  5.5895e-02,  7.6077e-02, -5.2480e-02,
        4.5928e-02, -2.5266e-01,  5.2845e-01, -1.3145e-01, -1.2453e-01,
        4.0556e-01,  3.1877e-01,  2.4415e-02, -2.2620e-01, -6.1960e-01,
       -4.0886e-01, -3.5534e-02, -5.5123e-03,  2.3438e-01,  8.7854e-01,
       -2.5161e-01,  4.0600e-01, -4.4284e-01,  3.4934e-01, -5.6429e-01,
       -2.3676e-01,  6.2199e-01, -2.8175e-01,  4.2024e-01,  1.0043e-01,
```

```
 -1.4720e-01,  4.9593e-01, -3.5850e-01, -1.3998e-01, -2.7494e-01,
  2.3827e-01,  5.7268e-01,  7.9025e-02,  1.7872e-02, -2.1829e-01,
  5.5050e-02, -5.4200e-01,  1.6788e-01,  3.9065e-01,  3.0209e-01,
  2.3040e-01, -3.9351e-02, -2.1078e-01, -2.7224e-01,  1.6907e-01,
  5.4819e-01,  9.4888e-02,  7.9798e-01, -6.6158e-02,  1.9844e-01,
  2.0307e-01,  4.4808e-02, -1.0240e-01, -6.9909e-02, -3.6756e-02,
  9.5159e-02, -2.7830e-01, -1.0597e-01, -1.6276e-01, -1.8211e-01,
 -3.1897e-01, -2.1633e-01,  1.4994e-01, -7.2057e-02,  2.2264e-01,
 -4.5551e-01,  3.0341e-01,  1.8431e-01,  2.1681e-01, -3.1940e-01,
  2.6426e-01,  5.8106e-01,  5.4635e-02,  6.3238e-01,  4.3169e-01,
  9.0343e-02,  1.9494e-01,  3.5483e-01, -2.0706e-02, -7.3117e-01,
  1.2941e-01,  1.7418e-01, -1.5065e-01,  5.3355e-02,  4.4794e-02,
 -1.6600e-01,  2.2007e-01, -5.3970e-01, -2.4968e-01, -2.6464e-01,
 -5.5515e-01,  5.8242e-01,  2.2295e-01,  2.4433e-01,  4.5275e-01,
  3.4693e-01,  1.2255e-01, -3.9059e-02, -3.2749e-01, -2.7891e-01,
  1.3766e-01,  3.8392e-01,  1.0543e-03, -1.0242e-02,  4.9205e-01,
 -1.7922e-01,  4.1215e-02,  1.3547e-01, -2.0598e-01, -2.3194e-01,
 -7.7701e-01, -3.8237e-01, -7.6383e-01,  1.9418e-01, -1.5441e-01,
  8.9740e-01,  3.0626e-01,  4.0376e-01,  2.1738e-01, -3.8050e-01],
dtype=float32)
```

In [21]:
```
nlp(u'The Quick Brown fox Jumped').vector
```

Out[21]:
```
array([-2.09217995e-01, -2.78227981e-02, -3.57064009e-02,  1.55218393e-01,
       -1.28050027e-02,  1.31627038e-01, -1.99465990e-01,  4.75811996e-02,
        1.26798794e-01,  1.64792800e+00, -3.57592016e-01, -1.39875397e-01,
       -1.26122087e-02, -2.02728346e-01, -2.25237608e-01,  2.15431936e-02,
        7.78958052e-02,  9.29676056e-01, -2.75549982e-02, -3.71005982e-01,
       -1.42800003e-01, -3.66641544e-02, -1.07376035e-02, -1.84352830e-01,
        2.29006782e-02, -5.17717972e-02, -2.78652012e-01, -1.19738199e-01,
        5.10960072e-03, -2.85990000e-01, -1.58261746e-01,  2.96241999e-01,
        1.09597601e-01, -4.18331996e-02,  1.87256075e-02, -1.03439607e-01,
       -5.10879979e-02, -3.51091917e-03, -6.81461841e-02, -2.05657601e-01,
        1.66347414e-01, -9.31599736e-03, -4.61134054e-02, -1.05457589e-01,
        2.31313989e-01,  1.80005193e-01, -2.06444815e-01, -1.37050152e-02,
        1.70106202e-01, -2.19812002e-02, -2.14003205e-01,  1.07415602e-01,
       -2.80592032e-02, -5.23634031e-02, -4.86331955e-02, -1.04047179e-01,
        1.27018047e-02,  2.02107817e-01, -1.18217587e-01, -1.51981995e-01,
        5.12168184e-02,  5.64177930e-02,  5.44355996e-02,  5.15560023e-02,
        5.14240041e-02, -1.37740612e-01, -8.45800620e-03, -1.13289997e-01,
        1.34828404e-01, -2.25100014e-02,  1.19754001e-01,  5.12372032e-02,
        5.40098026e-02,  7.36430064e-02, -1.59269981e-02, -2.37861007e-01,
       -7.90134817e-02, -1.30640596e-01,  4.45272028e-02, -4.98013981e-02,
       -7.30358064e-02,  1.80923387e-01,  4.17133979e-02,  4.45965007e-02,
        1.77781999e-01,  1.66720040e-02,  6.30389988e-01,  4.30720389e-01,
        2.10017413e-01,  1.68576598e-01,  7.29599595e-03,  1.58338398e-01,
        3.79000008e-02, -3.46915185e-01,  1.33294016e-01, -9.09054056e-02,
       -6.90027997e-02, -5.71460137e-03,  6.47759885e-02, -2.06994608e-01,
        1.64740205e-01, -2.13680025e-02, -1.30543396e-01, -3.02743949e-02,
       -1.03020016e-02, -6.46848023e-01,  1.80351987e-01, -9.54739973e-02,
       -1.40800001e-02, -3.42049934e-02, -4.12111953e-02, -1.11505605e-01,
        1.27132609e-01, -1.88302785e-01,  1.13260604e-01,  1.80767015e-01,
        6.53811991e-02, -1.58561952e-02,  9.68780071e-02,  7.01044053e-02,
        4.83391993e-02, -1.63396388e-01, -4.51015905e-02, -9.61597934e-02,
       -3.00761998e-01,  1.63111001e-01,  4.52036038e-02, -7.97460005e-02,
       -1.00108802e-01, -5.79720028e-02, -1.80559454e-03, -8.24856013e-02,
       -1.32837012e-01,  7.58986026e-02,  1.25125393e-01, -1.17681786e-01,
       -2.02512026e-01, -5.02933972e-02, -3.09894979e-02, -1.96231812e-01,
       -2.05938005e+00, -1.78128034e-02,  1.31141797e-01,  5.38920052e-02,
        2.48921394e-01, -8.28451961e-02,  2.29120012e-02,  4.11577933e-02,
        6.28991947e-02,  1.26971409e-01,  3.54279988e-02, -1.65512592e-01,
        1.54437393e-01, -8.91011953e-02, -2.38956213e-01,  3.74409966e-02,
       -1.47978395e-01,  1.23184405e-01,  7.47255981e-02, -8.21532011e-02,
       -3.02814040e-02, -1.99475795e-01, -2.98164397e-01, -5.18049970e-02,
```

```
 -5.98729961e-02, -1.95260614e-01, -1.14224993e-01, -7.07461983e-02,
 -1.35402009e-01, -1.59228414e-01,  1.33558795e-01,  1.44477993e-01,
 -2.50075996e-01, -2.07789987e-01, -3.69598001e-01, -1.38345197e-01,
  2.83456177e-01, -4.93402767e-04, -2.13945992e-02,  1.83038004e-02,
 -3.45086008e-02, -1.53184995e-01, -2.21591994e-01, -1.14851996e-01,
  9.88069996e-02,  1.29306391e-01, -5.40879965e-02, -4.65750024e-02,
 -4.59119631e-03,  1.25648379e-02,  2.73273796e-01,  3.86317968e-02,
  6.49093613e-02,  3.70982066e-02,  1.26923593e-02,  1.85716420e-01,
  1.69222593e-01,  1.06119812e-01,  1.43199565e-03, -7.52920061e-02,
 -1.51146010e-01,  4.95879985e-02, -2.87192404e-01, -2.74599995e-02,
  2.10097998e-01,  1.37594968e-01, -1.26467999e-02, -1.47451401e-01,
 -6.98073283e-02,  1.16517963e-02,  7.65941963e-02,  8.24979991e-02,
 -3.28646004e-02,  2.22982407e-01,  8.53662044e-02,  1.13203190e-01,
 -3.06712002e-01,  6.65521994e-02, -2.42485963e-02,  1.85453802e-01,
  6.33899868e-03,  2.36580381e-03, -6.98355958e-02,  9.46911126e-02,
 -2.32161760e-01, -1.91601396e-01,  2.09780186e-01,  1.91669196e-01,
  2.42458023e-02,  4.20044035e-01, -1.50683997e-02,  4.16760286e-03,
 -7.33660609e-02,  5.19229956e-02, -1.21735949e-02, -8.69527981e-02,
 -1.26489595e-01, -6.72094822e-02,  1.63832814e-01,  2.75269568e-01,
 -1.64249986e-01, -1.52959794e-01, -5.64128049e-02, -9.03348029e-02,
  3.59305963e-02, -1.62403792e-01, -9.81536135e-02,  3.03588063e-02,
 -2.10355401e-01,  5.90659976e-02, -3.81862000e-02, -1.31029800e-01,
 -1.98952004e-01,  1.02112450e-01,  1.69432998e-01,  4.52830084e-02,
  2.06268996e-01,  1.30002588e-01,  3.64079997e-02,  1.67501979e-02,
  1.52415991e-01,  8.69902000e-02,  1.43315807e-01, -1.03890002e-01,
  2.32943982e-01, -2.00447604e-01,  8.02273974e-02, -4.50324044e-02,
 -2.33428001e-01,  9.64580029e-02, -8.72388482e-04,  2.65751779e-01,
 -4.15487401e-02, -1.38139397e-01, -3.27680036e-02, -4.31547984e-02,
  3.97300012e-02,  1.05935797e-01,  9.52792179e-04,  3.66709009e-02,
  1.43723994e-01,  9.49178040e-02, -1.12830400e-01,  1.76364794e-01,
 -2.80858018e-02, -3.65898088e-02,  7.58539960e-02,  7.08954111e-02,
 -7.30042011e-02,  1.27999904e-02, -2.66412795e-01,  1.81497976e-01,
 -2.59018000e-02, -1.16812006e-01, -7.90375918e-02,  2.10512038e-02,
  2.83426046e-02,  4.26702015e-02, -1.21463798e-01, -1.45020094e-02],
dtype=float32)
```

The Shape of an a 43rd sum : (size)

page 74

In [22]:
```
nlp(u'The Quick Brown fox Jumped').vector.shape
```

Out[22]:
```
(300,)
```

SIMILARITY : VADER SENTIMENT ANALYSIS IN 'NLTK' :

PAGE 74

In [23]:
```
tokens = nlp(u'lion cat pet')
```

In [24]:
```
# Here, lion lion 1.0(means, 100% matching)
# lion cat 0.52(means, 52% matching)

for token1 in tokens:
    for token2 in tokens:
        print(token1.text, token2.text, token1.similarity(token2))
```

```
lion lion 1.0
lion cat 0.5265437960624695
lion pet 0.39923766255378723
cat lion 0.5265437960624695
cat cat 1.0
cat pet 0.7505456805229187
pet lion 0.39923766255378723
pet cat 0.7505456805229187
pet pet 1.0
```

In [ ]:
```

```

In [25]:
```
# Till now the regular NLP, the Sentimental Analysis we are working with,
# In that polarity,sentiment we have gone with - 'Polarity','Sid Polarity'
# 'Sid' - Means, 'Comparing the Words'
# Means, Each and Every Word im 'NLP' - Going to be Called as 'VECTOR'.
```

In [26]:
```
# Now let's call - 'Spacy Library' once again and We Start Working on that Particular 'ENGLISH LIBRARY'
```

In [27]:
```
import spacy
```

In [28]:
```
# Each and Every word in this particular English Library.
# It is going to be 'Cross Checked' with an a '300*almost 680000 Changed Words'
# Means, We are going to be Declared is Considered as ana a 'VECTOR'.

nlp = spacy.load('en_core_web_lg')
```

In [29]:
```
nlp.vocab.vectors.shape
```

Out[29]:
```
(342918, 300)
```

In [ ]:
```

```

In [30]:
```
# 'Spatial' - Means, 'Co-sine Mathematical Expression' from Co-sine Value.
# Each and Every Word we are describing some particular value like, 69%, 89%,...
# Eg: 'lion' - We are comparing with all words.
# lion lion 1.0 (Means, Comparing 100%) and rest of the Words are 67%,89%,90%... Mapping with 'lion''
# Means, that particular differentiation, we are going to work with an a 'Co-sine Value'.
```

In [31]:
```
from scipy import spatial
```

In [32]:
```
# Here, Co-sine Similarity(left hand variable), by using an a 'Lambda Expression'.
# Vector1 of an a Vector2 Where i'm taking "1- Spatial Distance"(Sometimes it won't work and Sometimes it will work)
# dot.cosine of vec1 of an a vector2
```

In [33]:
```
cosine_similarity = lambda vec1,vec2 : 1 - spatial.distance.cosine(vec1,vec2)
```

In [34]:
```
# Now i'm keeping an a '3' particular words in the Vocabulary(nlp.vocab). These Words name is 'king' i'm keeping here
```

In [35]:
```
king = nlp.vocab['king'].vector
man = nlp.vocab['man'].vector
women = nlp.vocab['women'].vector
```

In [36]:
```
# Now
```

In [37]:
```
new_vector = king - man + women
```

In [38]:
```
#
```

In [39]:
```
computed_similarities = []  # Empty list[]
for word in nlp.vocab:
    if word.has_vector:
        if word.is_lower:
            if word.is_alpha:
                similarity = cosine_similarity(new_vector,word.vector)
                computed_similarities.append((word,similarity))
```

In [40]:
```
# Now
```

In [41]:
```
computed_similarities = sorted(computed_similarities, key = lambda item : -item[1])
```

In [42]:
```
# Now we take an a 'Print'
```

In [43]:
```
# part of part data([]) we need to declare :
# These are the particular - 'The Compared Words' we are getting here :


print([t[0].text for t in computed_similarities[:10]])
```

```
['king', 'women', 'these', 'those', 'are', 'all', 'and', 'were', 'they', 'who']
```

In [44]:
```
# Now let's Change to 15 words [:15]
```

In [45]:
```
print([t[0].text for t in computed_similarities[:15]])
```

```
['king', 'women', 'these', 'those', 'are', 'all', 'and', 'were', 'they', 'who', 'dare', 'have', 'or', 'not', 'shoul
d']
```

TOPIC MODELLING :

PAGE 79

In [46]:
```
# The Exact Meaning of Topic Modelling Means,
#
# LDA & NMF METHOD :
```

In [47]:
```
import pandas as pd
```

In [48]:
```
npr = pd.read_csv('DataS/articles1.csv')
```

In [49]:
```
npr.head()
```

Out[49]:

	Unnamed: 0	id	title	publication	author	date	year	month	url	content
0	0	17283	House Republicans Fret About Winning Their Hea...	New York Times	Carl Hulse	2016-12-31	2016.0	12.0	NaN	WASHINGTON — Congressional Republicans have...
1	1	17284	Rift Between Officers and Residents as Killing...	New York Times	Benjamin Mueller and Al Baker	2017-06-19	2017.0	6.0	NaN	After the bullet shells get counted, the blood...
2	2	17285	Tyrus Wong, ‘Bambi’ Artist Thwarted by Racial ...	New York Times	Margalit Fox	2017-01-06	2017.0	1.0	NaN	When Walt Disney’s “Bambi” opened in 1942, cri...
3	3	17286	Among Deaths in 2016, a Heavy Toll in Pop Musi...	New York Times	William McDonald	2017-04-10	2017.0	4.0	NaN	Death may be the great equalizer, but it isn’t...
4	4	17287	Kim Jong-un Says North Korea Is Preparing to T...	New York Times	Choe Sang-Hun	2017-01-02	2017.0	1.0	NaN	SEOUL, South Korea — North Korea’s leader, ...

Now i want to get one particular column : Content Column :

page 80

In [50]:
```
npr['content']
```

Out[50]:
```
0        WASHINGTON  —   Congressional Republicans have...
1        After the bullet shells get counted, the blood...
2        When Walt Disney’s “Bambi” opened in 1942, cri...
3        Death may be the great equalizer, but it isn’t...
4        SEOUL, South Korea  —   North Korea’s leader, ...
                              ...
49995    As chairman and CEO of ExxonMobil, Rex Tillers...
49996    I’ve spent nearly 20 years looking at intellig...
49997      Donald Trump will not be taking necessary st...
49998    Dozens of   colleges could be forced to close ...
49999    The force of gravity can be described using a ...
Name: content, Length: 50000, dtype: object
```

Length of a DataSet :

page 81

In [51]:
```
# Now, What is the length of a DataSet :
```

In [52]:
```
len(npr)
```

Out[52]:
```
50000
```

Reading Particular Amount of Data :

In [53]:
```
# This is an a Particular Record, Particular Amount of Data : We Can Read it here :


npr['content'][4000]
```

Out[53]:
```
'This week, we meet for breakfast to talk through our conflicting feelings about the new film “When the Bough Break
s,” the No. 2 movie in America. Jenna loved it Wesley not so much. We also decode the inherent racism of the sharing
economy. Airbnb recently issued a report about how they plan to fight discrimination on the site. One of the propose
d solutions is to make the avatar photos of   smaller, making it harder to discriminate. Um, what? “I’ve got to be h
onest with you,” Wesley tells Jenna on the show. “If AirbnB wants to make the avatars smaller, make mine twice as bi
g! If you don’t want me staying in your house, don’t front like you’re happy to have me. ” “This moment of reckoning
is bigger than has ever happened before with the whole tech wave,” Jenna says. ”The fact that we have these founders
that want to be woke,” Jenna says, is a big step in the right direction. But even with that awareness, she says, the
fact remains: “You can’t code yourself out of racism. ” We also bring in the dance writer Shanti Crawford to review
the moves we watched at the United States Open. From a desktop or laptop, you can listen by pressing play on the but
ton above. Or if you’re on a mobile device, the instructions below will help you find and subscribe to the series. O
n your iPhone or iPad: 1. Open your podcast app. It’s a   app called “Podcasts” with a purple icon. (This link may h
elp.) 2. Search for the series. Tap on the “search” magnifying glass icon at the bottom of the screen, type in “Stil
l Processing” and select it from the list of results. 3. Subscribe. Once on the series page, tap on the “subscribe”
button to have new episodes sent to your phone free. You may want to adjust your notifications to be alerted when a
new episode arrives. 4. Or just sample. If you would rather listen to an episode or two before deciding to subscrib
e, tap on the episode title from the list on the series page. If you have an internet connection, you’ll be able to
stream the episode. On your Android phone or tablet: 1. Open your podcast app. It’s a   app called “Play Music” with
an    icon. (This link may help.) 2. Search for the series. Click on the magnifying glass icon at the top of the scr
een, search for “Still Processing” and select it from the list of results. You may have to scroll down to find the
“Podcasts” search results. 3. Subscribe. Once on the series page, click on the word “subscribe” to have new episodes
sent to your phone free. 4. Or just sample. If you would rather listen to an episode or two before deciding to subsc
ribe, click on the episode title from the list on the series page. If you have an internet connection, you’ll be abl
e to stream the episode.'
```

In [ ]:
```

```

In [54]:
```
from sklearn.feature_extraction.text import CountVectorizer
```

In [55]:
```
cv = CountVectorizer(max_df = 0.90, min_df = 2, stop_words = 'english')
```

In [56]:
```
# Now, i'm taking left hand variable 'dtm'
```

In [57]:
```
# from here onwards, it will take lot of time, Because 'The Data is Very Huge',
# That's why it will take Lot of Time in Working :

dtm = cv.fit_transform(npr['content'])
```

In [58]:
```
dtm
```

Out[58]:
```
<50000x91380 sparse matrix of type '<class 'numpy.int64'>'
        with 10807637 stored elements in Compressed Sparse Row format>
```

LDA : LatentDirichletAllocation :

PAGE 82

In [59]:
```
# Here, from sklearn of decomposition, i'm getting LatentDirichletAllocation : 'LDA'

from sklearn.decomposition import LatentDirichletAllocation
```

In [60]:
```
# Here LDA = How many Topics represent here - Means, How many Records;
# components = only '7', This One Only take an a Long time. and
# random_state = '101


LDA = LatentDirichletAllocation(n_components = 7, random_state = 101)
```

In [61]:
```
# Now it will take Long time, we need to wait for it, Nothing we can do it here, Because it need to process :

LDA.fit(dtm)
```

Out[61]:
```
LatentDirichletAllocation(n_components=7, random_state=101)
```

In [62]:
```
# Now i want to identify, What is the 'Length' of an a particular 'cv' :
# Means, '91380' are getting 'Matched' :


len(cv.get_feature_names())
```

```
C:\Users\my pc\anaconda3\lib\site-packages\sklearn\utils\deprecation.py:87: FutureWarning: Function get_feature_name
s is deprecated; get_feature_names is deprecated in 1.0 and will be removed in 1.2. Please use get_feature_names_out
instead.
  warnings.warn(msg, category=FutureWarning)
```

Out[62]: 91380

What is an a Data Type :

page 83

In [63]:
```
type(cv.get_feature_names())
```

Out[63]: list

In [66]:
```
# Now, Here, We are Searching with 'Random Numbers' - like (0,5477)
# cv.get_feature_names([random_word_id]) - i'm taking to get any Random Number.
```

In [78]:
```
# Here, if we 'Re-Run' - We will get an a 'Alternative Names'

import random
random_word_id = random.randint(0,5477)
cv.get_feature_names()[random_word_id]
```

Out[78]: 'advisories'

In [73]:
```
# Length of LDA COMPONENTS :
# Already, we have taken '7' LDA Components, Now we will 'Cross Check' and Verify Once again :
```

In [75]:
```
len(LDA.components_)
```

Out[75]:
```
7
```

SINGLE COMPONENT WORKING ENVIRONMENT :

PAGE 84

In [80]:
```
# or else, let me take an a directly apply - single_topic.argsort(). let's take an a Single Topic :
```

In [86]:
```
single_topic = LDA.components_[0]
single_topic.argsort()
```

Out[86]:
```
array([90580, 90156, 59593, ..., 55955, 83979, 60740], dtype=int64)
```

In [87]:
```
# Now, single_topic.argsort() of '1st 10 Topics' :
```

In [88]:
```
single_topic.argsort()[-10:]
```

Out[88]:
```
array([81400, 24631, 12069, 43699, 47634, 70549, 83421, 55955, 83979,
       60740], dtype=int64)
```

In [90]:
```
top_ten_words = single_topic.argsort()[-10:]
```

In [91]:
```
for index in top_ten_words:
    print(cv.get_feature_names()[index])
```

```
think
don
breitbart
just
like
said
trump
news
twitter
people
```

In [102]:
```
# Now i want to get Top 15 Words, Only from the 'LDA' Components :
# Here, 'i' means, it will Present the 'Index' :
# feature_names of index for index in topic.argsort()method of [-15:]

# output : 15 words of 6 topics :
# Here, # of (i) position means, it will present the 'Index' :
```

In [101]:
```
for i,topic in enumerate(LDA.components_):
    print(f"THE TOP 15 WORDS #{i}")
    print([cv.get_feature_names()[index] for index in topic.argsort()[-15:]])
    print('\n')
    print('\n')
```

```
THE TOP 15 WORDS #0
['black', 'going', 'women', 'know', 'media', 'think', 'don', 'breitbart', 'just', 'like', 'said', 'trump', 'news',
'twitter', 'people']
```

```
THE TOP 15 WORDS #1
['reported', 'man', 'officials', 'isis', 'cnn', 'killed', 'officers', 'city', 'attack', 'according', 'state', 'tol
d', 'people', 'police', 'said']
```

```
THE TOP 15 WORDS #2
['season', 'million', 'games', 'apple', 'years', 'world', 'time', 'just', 'year', 'new', 'team', 'game', 'like', 'co
mpany', 'said']
```

```
THE TOP 15 WORDS #3
['american', 'state', 'trump', 'people', 'world', 'china', 'north', 'government', 'country', 'court', 'states', 'uni
ted', 'president', 'said', 'mr']
```

```
THE TOP 15 WORDS #4
['new', 'state', 'presidential', 'election', 'hillary', 'house', 'obama', 'donald', 'mr', 'republican', 'campaign',
'president', 'said', 'clinton', 'trump']
```

```
THE TOP 15 WORDS #5
['work', 'city', 'day', 'family', 'year', 'life', 'just', 'time', 'years', 'ms', 'people', 'like', 'new', 'mr', 'sai
d']
```

```
THE TOP 15 WORDS #6
['department', '000', 'million', 'people', 'president', 'health', 'state', 'states', 'federal', 'percent', 'mr', 'go
vernment', 'new', 'trump', 'said']
```

In [ ]: