```
from google.colab import drive
drive.mount('/content/drive')
```

```
    Mounted at /content/drive
```

```
!pip install sentence-transformers
```

```
    Collecting sentence-transformers
      Downloading https://files.pythonhosted.org/packages/f5/5a/6e41e8383913dd2ba923cdcd
         |████████████████████████████████| 71kB 6.9MB/s
    Collecting transformers<3.6.0,>=3.1.0
      Downloading https://files.pythonhosted.org/packages/3a/83/e74092e7f24a08d751aa59b3
         |████████████████████████████████| 1.3MB 29.2MB/s
    Requirement already satisfied: tqdm in /usr/local/lib/python3.6/dist-packages (from
    Requirement already satisfied: torch>=1.6.0 in /usr/local/lib/python3.6/dist-package
    Requirement already satisfied: numpy in /usr/local/lib/python3.6/dist-packages (from
    Requirement already satisfied: scikit-learn in /usr/local/lib/python3.6/dist-package
    Requirement already satisfied: scipy in /usr/local/lib/python3.6/dist-packages (from
    Requirement already satisfied: nltk in /usr/local/lib/python3.6/dist-packages (from
    Collecting tokenizers==0.9.3
      Downloading https://files.pythonhosted.org/packages/4c/34/b39eb9994bc3c999270b69c9
         |████████████████████████████████| 2.9MB 53.8MB/s
    Requirement already satisfied: dataclasses; python_version < "3.7" in /usr/local/lib
    Requirement already satisfied: requests in /usr/local/lib/python3.6/dist-packages (f
    Requirement already satisfied: protobuf in /usr/local/lib/python3.6/dist-packages (f
    Requirement already satisfied: packaging in /usr/local/lib/python3.6/dist-packages (
    Requirement already satisfied: filelock in /usr/local/lib/python3.6/dist-packages (f
    Collecting sentencepiece==0.1.91
      Downloading https://files.pythonhosted.org/packages/d4/a4/d0a884c4300004a78cca907a
         |████████████████████████████████| 1.1MB 34.3MB/s
    Requirement already satisfied: regex!=2019.12.17 in /usr/local/lib/python3.6/dist-pa
    Collecting sacremoses
      Downloading https://files.pythonhosted.org/packages/7d/34/09d19aff26edcc8eb2a01bed
         |████████████████████████████████| 890kB 49.6MB/s
    Requirement already satisfied: future in /usr/local/lib/python3.6/dist-packages (fro
    Requirement already satisfied: typing-extensions in /usr/local/lib/python3.6/dist-pa
    Requirement already satisfied: joblib>=0.11 in /usr/local/lib/python3.6/dist-package
    Requirement already satisfied: six in /usr/local/lib/python3.6/dist-packages (from n
    Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.6/dist-pa
    Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.6/dist-package
    Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /usr/local
    Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.6/dist-p
    Requirement already satisfied: setuptools in /usr/local/lib/python3.6/dist-packages
    Requirement already satisfied: pyparsing>=2.0.2 in /usr/local/lib/python3.6/dist-pac
    Requirement already satisfied: click in /usr/local/lib/python3.6/dist-packages (from
    Building wheels for collected packages: sentence-transformers, sacremoses
      Building wheel for sentence-transformers (setup.py) ... done
      Created wheel for sentence-transformers: filename=sentence_transformers-0.3.9-cp36
      Stored in directory: /root/.cache/pip/wheels/fc/89/43/f2f5bc00b03ef9724b0f6254a97e
      Building wheel for sacremoses (setup.py) ... done
      Created wheel for sacremoses: filename=sacremoses-0.0.43-cp36-none-any.whl size=89
      Stored in directory: /root/.cache/pip/wheels/29/3c/fd/7ce5c3f0666dab31a50123635e6f
    Successfully built sentence-transformers sacremoses
    Installing collected packages: tokenizers, sentencepiece, sacremoses, transformers,
    Successfully installed sacremoses-0.0.43 sentence-transformers-0.3.9 sentencepiece-0
```

```
# import all the necessary libraries
# import all the necessary libraries
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import re
from sklearn.cluster import DBSCAN
import string
import unicodedata
# from sklearn.feature_extraction.text import TfidfVectorizer
from gensim.models.doc2vec import Doc2Vec, TaggedDocument
import spacy
from sentence_transformers import SentenceTransformer
from sklearn.manifold import TSNE
```

```
# loading the dataset
train=pd.read_csv("/content/drive/My Drive/3rd Sem/Code v0.2/excel_data/summarydata-bert.c
```

```
train.head()
```

	News_ID	Newspaper3k	BERT
0	1	Abu Dhabi [UAE], October 7 (ANI): USA pacer Al...	Abu Dhabi [UAE], October 7 (ANI): USA pacer Al...
1	2	Abu Dhabi [UAE], October 6 (ANI): England and ...	Abu Dhabi [UAE], October 6 (ANI): England and ...
2	3	Sydney [Australia], October 7 (ANI): Arjun Nai...	Sydney [Australia], October 7 (ANI): Arjun Nai...
		Sydney [Australia], October 7 (ANI):	Sydney [Australia], October 7 (ANI):

```
train.dropna(inplace=True)
```

```
train.isnull().sum()
```

```
    News_ID        0
    Newspaper3k    0
    BERT           0
    dtype: int64
```

```
#convert each question to a list of string
data = pd.Series(train["BERT"].tolist()).astype(str)
```

```
data.head()
```

```
    0    Abu Dhabi [UAE], October 7 (ANI): USA pacer Al...
    1    Abu Dhabi [UAE], October 6 (ANI): England and ...
    2    Sydney [Australia], October 7 (ANI): Arjun Nai...
    3    Sydney [Australia], October 7 (ANI): Sydney Th...
    4    Abu Dhabi [UAE], October 6 (ANI): Mumbai India...
    dtype: object
```

```
data1 = data[:100]
```

```
sentences_list = data
```

Text Preprocessing

```
nlp = spacy.load('en_core_web_sm')
# stop_list = ['best','different',"won\'t", "couldn\'t", "mustn\'t", "didn\'t", "dtype obj
# for word in stop_list:
#     spacy.lang.en.stop_words.STOP_WORDS.add(word)
#     nlp.vocab[word].is_stop = True


def normalize(data):
    """Run all the functions for preprocessing in a pipeline"""
    clean_data = re.sub(re.compile('<.*?>'), '', data)
    cleaned_list = [ unicodedata.normalize('NFKD', word.text).encode('ascii', 'ignore').de
    cleaned_list = " ".join(cleaned_list)
    cleaned_list = [word.text.rstrip('0123456789').lower() for word in nlp(cleaned_list) i
    return cleaned_list


# Preprocess the text data
normalized_data = []
for i, batch in data.groupby(np.arange(len(data)) // 10):
    for batch_data in batch:
        normalized_data.append(normalize(batch_data))

    print(i)
```

```
0
1
2
3
4
5
6
7
8
9
10
11
12
13
14
15
16
17
18
19
20
21
22
23
24
25
```

```
26
27
28
29
30
31
32
33
34
35
36
37
38
39
40
41
42
43
44
45
46
47
48
49
50
51
52
53
54
55
56
57
58
```

```
# Data after prerocessing
print(normalized_data[0])
len(normalized_data)
```

```
['abu', 'dhabi', 'uae', 'october', 'ani', 'usa', 'pacer', 'ali', 'khan', 'ruled', 'i
9032
```

```
# function to form sentences from token
sentence = []
sentences = []


def token_2_sentence(normalized_data):
    """Join the tokens in each list with space to form a sentence"""
    for i in normalized_data:
      sentence = " ".join(i)
      sentences.append(sentence)
      sentence = []
    return sentences

sentences_list = token_2_sentence(normalized_data)


sentences_list[:10]
```

```
sentences_list[:10]
```

```
['abu dhabi uae october ani usa pacer ali khan ruled indian premier league ipl injur
 'abu dhabi uae october ani england rajasthan royals rounder ben stokes reckons kart
 'sydney australia october ani arjun nair signed big bash league bbl season sydney t
 'sydney australia october ani sydney thunder completed squad women big bash league
 'abu dhabi uae october ani mumbai indians brigade continued impress edition indian
 'abu dhabi uae october ani reminiscing catch dismiss rajasthan royals mahipal lomro
 'adelaide australia october ani west indies captain stafanie taylor rejoin adelaide
 'abu dhabi uae october ani rajasthan royals skipper steve smith fined maintaining r
 'abu dhabi uae october ani registering win rajasthan royals mumbai indians bowling
 'new delhi india october ani india head coach ravi shastri rounder yuvraj singh pra
```

```
import csv
with open('./normalized.csv', 'w', newline='') as file:
    writer = csv.writer(file)
    writer.writerow("Normalized")
```

```
for item in sentences_list:
  with open('./normalized.csv', 'a', newline='') as file:
      writer = csv.writer(file)
      writer.writerow([item])
```

```
# sentences_list = pd.read_csv("./normalized.csv")
# sentences_list = sentences_list.values.tolist()
# sentences_list[0]
```

```
# sentences_list[0][0]
```

Clustering with Embedding

```
model = SentenceTransformer('distilbert-base-nli-mean-tokens')
```

```
100%|██████████| 245M/245M [00:15<00:00, 15.8MB/s]
```

```
def data_gen(data):
    for sen in data:
      yield sen
```

```
a = data_gen(sentences_list)
```

```
encoding_arr = list()
current = 1
for item in a:
  embeddings = model.encode(item)
  encoding_arr.append(embeddings)
  print("Current:",current)
  current += 1
```

```
Streaming output truncated to the last 5000 lines.
Current: 4033
Current: 4034
Current: 4035
Current: 4036
Current: 4037
Current: 4038
Current: 4039
Current: 4040
Current: 4041
Current: 4042
Current: 4043
Current: 4044
Current: 4045
Current: 4046
Current: 4047
Current: 4048
Current: 4049
Current: 4050
Current: 4051
Current: 4052
Current: 4053
Current: 4054
Current: 4055
Current: 4056
Current: 4057
Current: 4058
Current: 4059
Current: 4060
Current: 4061
Current: 4062
Current: 4063
Current: 4064
Current: 4065
Current: 4066
Current: 4067
Current: 4068
Current: 4069
Current: 4070
Current: 4071
Current: 4072
Current: 4073
Current: 4074
Current: 4075
Current: 4076
Current: 4077
Current: 4078
Current: 4079
Current: 4080
Current: 4081
Current: 4082
Current: 4083
Current: 4084
Current: 4085
Current: 4086
Current: 4087
Current: 4088
Current: 4089
Current: 4090
```

```
encoded_arr = np.array(encoding_arr)
encoded_arr_bert = encoded_arr
encoded_arr_bert.shape
```

```
(9032, 768)
```

```
# from gensim.models.doc2vec import Doc2Vec, TaggedDocument
```

```
def tagged_document(normalized_data):
    tagged_corpus = []
    tagged_corpus = [TaggedDocument(words = d, tags=[str(i)]) for i,d in enumerate(normali
    return tagged_corpus
```

```
tagged_corpus =  tagged_document(normalized_data)
```

```
tagged_corpus
```

```
TaggedDocument(words=['mumbai', 'maharashtra', 'india', 'october', 'ani', 'mahar
TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'janata', 'da
TaggedDocument(words=['buxar', 'bihar', 'india', 'october', 'ani', 'bjp', 'presi
TaggedDocument(words=['wayanad', 'kerala', 'india', 'october', 'ani', 'congress'
TaggedDocument(words=['amit', 'kumarnew', 'delhi', 'india', 'october', 'ani', 'b
TaggedDocument(words=['gaya', 'bihar', 'india', 'october', 'ani', 'lok', 'jansha
TaggedDocument(words=['bettiah', 'bihar', 'india', 'october', 'ani', 'bjp', 'pre
TaggedDocument(words=['jalgaon', 'maharashtra', 'india', 'october', 'ani', 'ekna
TaggedDocument(words=['chapra', 'bihar', 'india', 'october', 'ani', 'aishwarya',
TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'union', 'min
TaggedDocument(words=['mukesh', 'singh', 'sahil', 'pandeypatna', 'bihar', 'india
TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'congress', '
TaggedDocument(words=['srinagar', 'jammu', 'kashmir', 'india', 'october', 'ani',
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'congress', 'le
TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'mahagathband
TaggedDocument(words=['anuppur', 'madhya', 'pradesh', 'india', 'october', 'ani',
TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'prime', 'min
TaggedDocument(words=['bhatinda', 'punjab', 'india', 'october', 'ani', 'man', 'd
TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'bjp', 'leade
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'bjp', 'chief',

TaggedDocument(words=['patna', 'bihar', 'india', 'october', 'ani', 'promise', 'v
TaggedDocument(words=['washington', 'october', 'ani', 'netflix', 'anticipated',
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'occasion', 'bi
TaggedDocument(words=['washington', 'october', 'ani', 'book', 'lumberjanes', 'tu
TaggedDocument(words=['mumbai', 'maharashtra', 'india', 'october', 'ani', 'vacat
TaggedDocument(words=['ashoke', 'rajnew', 'delhi', 'india', 'october', 'ani', 'b
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'actor', 'alia'
TaggedDocument(words=['washington', 'october', 'ani', 'video', 'streaming', 'pla
TaggedDocument(words=['washington', 'october', 'ani', 'band', 'ac', 'dc', 'givin
TaggedDocument(words=['washington', 'october', 'ani', 'look', 'trailer', 'news',
TaggedDocument(words=['mumbai', 'maharashtra', 'india', 'october', 'ani', 'month
TaggedDocument(words=['california', 'october', 'ani', 'megastar', 'priyanka', 'c
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'actor', 'karti
TaggedDocument(words=['washington', 'october', 'ani', 'king', 'monsters', 'godzi
TaggedDocument(words=['washington', 'october', 'ani', 'rock', 'roll', 'guitarist
TaggedDocument(words=['washington', 'october', 'ani', 'singer', 'johnny', 'nash'
TaggedDocument(words=['mumbai', 'maharashtra', 'india', 'october', 'ani', 'talki
TaggedDocument(words=['washington', 'october', 'ani', 'week', 'saturday', 'night
TaggedDocument(words=['washington', 'october', 'ani', 'coronavirus', 'pandemic',
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'commencement',
TaggedDocument(words=['mumbai', 'maharashtra', 'india', 'october', 'ani', 'strea
TaggedDocument(words=['british', 'columbia', 'canada', 'october', 'ani', 'loomin
TaggedDocument(words=['washington', 'october', 'ani', 'marvel', 'studios', 'sony
TaggedDocument(words=['washington', 'october', 'ani', 'shooting', 'television',
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'giving', 'glim
TaggedDocument(words=['washington', 'october', 'ani', 'pixar', 'soul', 'skipping
TaggedDocument(words=['washington', 'october', 'ani', 'country', 'music', 'singe
TaggedDocument(words=['washington', 'october', 'ani', 'karlovy', 'vary', 'intern
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'kangana', 'ran
TaggedDocument(words=['washington', 'october', 'ani', 'month', 'welcoming', 'bab
TaggedDocument(words=['washington', 'october', 'ani', 'talking', 'comeback', 'co
TaggedDocument(words=['washington', 'october', 'ani', 'update', 'eligibility', '
TaggedDocument(words=['washington', 'october', 'ani', 'people', 'tested', 'sets'
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'bollywood', 'c
TaggedDocument(words=['washington', 'october', 'ani', 'emerging', 'actor', 'sara
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'dropping', 'te
TaggedDocument(words=['washington', 'october', 'ani', 'rapper', 'tory', 'lanez',
TaggedDocument(words=['new', 'delhi', 'india', 'october', 'ani', 'megastar', 'sa
```

```
def build_model(tagged_corpus,max_epochs,vec_size, alpha):
    model = Doc2Vec(size=vec_size, alpha=alpha,min_alpha=0.001, min_count=1,dm =1)
    model.build_vocab(tagged_corpus)

    for epoch in range(max_epochs):
        model.train(tagged_corpus,total_examples=model.corpus_count, epochs=model.iter)
        # decrease the learning rate
        model.alpha -= 0.002
        # fix the learning rate, no decay
        model.min_alpha = model.alpha


    model.save("d2v.model")
    print("Model Saved")
    model_name = "d2v.model"
    return model_name



# from gensim.models.doc2vec import Doc2Vec

def load_model(model_name, data):
    corpus_vector = []
    model= Doc2Vec.load(model_name)
    for doc in data:
        corpus_vector.append(model.infer_vector(doc.split()))
    return corpus_vector


max_epochs = 100
vec_size = 100
alpha = 0.001
model_name = build_model(tagged_corpus,max_epochs,vec_size, alpha)
```

```
/usr/local/lib/python3.6/dist-packages/gensim/models/doc2vec.py:570: UserWarning: Th
  warnings.warn("The parameter `size` is deprecated, will be removed in 4.0.0, use `
/usr/local/lib/python3.6/dist-packages/ipykernel_launcher.py:7: DeprecationWarning:
  import sys
Model Saved
```

```
corpus_vector = load_model("d2v.model",data)
```

```
corpus_vector = np.array(corpus_vector)
```

```
corpus_vector.shape
```

```
(9032, 100)
```

```
#PCA
from sklearn.decomposition import PCA
pca = PCA(n_components=2)
corpus_vector_pca = pca.fit_transform(corpus_vector)
print(corpus_vector_pca.shape)

pca = PCA(n_components=2)
encoded_arr_bert_pca = pca.fit_transform(encoded_arr_bert)
print(encoded_arr_bert_pca.shape)
```

```
(9032, 2)
(9032, 2)
```

```
#t-SNE
from sklearn.manifold import TSNE
tsne = TSNE(n_components = 2, init = 'random', random_state = 10, perplexity = 100)
# Use only 400 rows to shorten processing time
corpus_vector_tsne = tsne.fit_transform(corpus_vector)
print(corpus_vector_tsne.shape)

tsne = TSNE(n_components = 2, init = 'random', random_state = 10, perplexity = 100)
# Use only 400 rows to shorten processing time
encoded_arr_bert_tsne = tsne.fit_transform(encoded_arr_bert)
print(encoded_arr_bert_tsne.shape)
```

```
(9032, 2)
(9032, 2)
```

```
#KMeans (WITHOUT Dimensionality Reduction)

from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

def kmeans(corpus_vector):
    """Function to form dbscan clusters and display them"""
#     eps = 0.005# how close points should be to each other to be considered a part of a c
#     min_samples = 3# the minimum number of points to form a dense region
#     dbscan = DBSCAN( eps=eps, min_samples=min_samples,metric = "cosine" )
#     dbscan_model = dbscan.fit(corpus_vector)

    # pca = PCA(n_components=2)
    # result = pca.fit_transform(corpus_vector)
    # print(result.shape)

    Sum_of_squared_distances = []
    K = range(10,30)
    for k in K:
      km = KMeans(n_clusters=k, max_iter=200, n_init=10)
      km = km.fit(corpus_vector)
      Sum_of_squared_distances.append(km.inertia_)
      print(k,":",Sum_of_squared_distances[-1])
    plt.plot(K, Sum_of_squared_distances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Sum_of_squared_distances')
    plt.title('Elbow Method For Optimal k')
    plt.show()




#K-Means on BERT Embedding


kmeans(encoded_arr_bert)
```

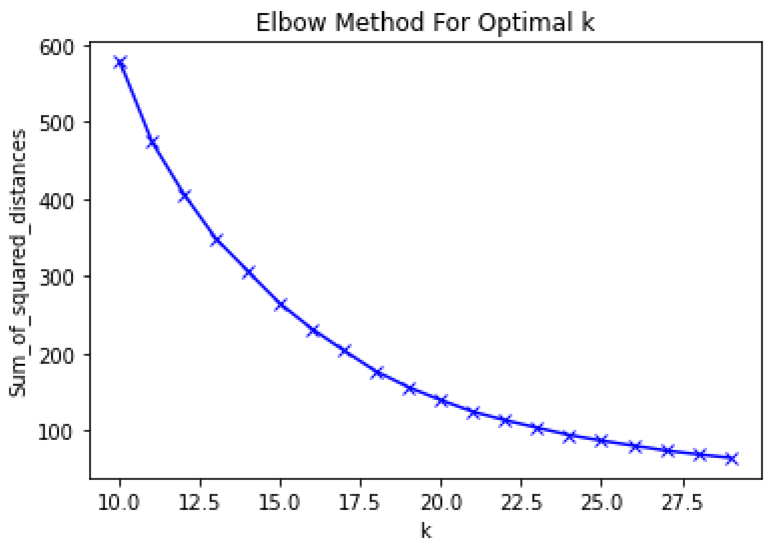
```
    10 : 678934.357737165
    11 : 671269.917885134
    12 : 664449.4732211847
    13 : 658180.0726003905
    14 : 652480.3868368929
    15 : 647256.4101899003
```

```
#K-Means on Doc2Vec Embedding
```

```
    18 : 635518.9174276853
```

```
kmeans(corpus_vector)
```

```
    10 : 578.1801830175549
    11 : 473.7791707309858
    12 : 405.74697022867764
    13 : 347.9309583199347
    14 : 305.8596349695069
    15 : 264.34772804087794
    16 : 231.50305087306094
    17 : 203.38615047842308
    18 : 176.27131669143435
    19 : 155.80004330279664
    20 : 139.21593019426928
    21 : 124.19267834226291
    22 : 113.19523780072845
    23 : 103.5165667200043
    24 : 93.99763896526612
    25 : 86.68742704532912
    26 : 80.27234145251865
    27 : 73.78372332740909
    28 : 68.90875076390834
    29 : 64.46465027380036
```

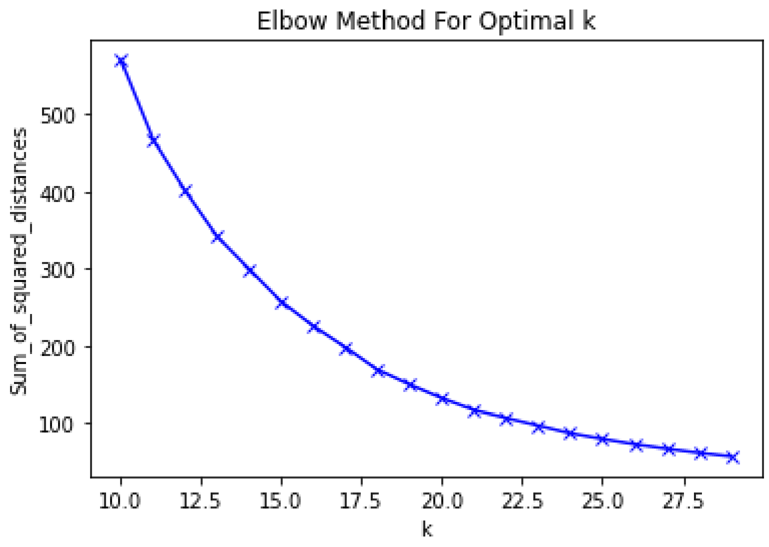


```
#KMeans (WITH Dimensionality Reduction PCA)

from sklearn.decomposition import PCA
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

def kmeans_pca(result):
    """Function to form dbscan clusters and display them"""
#     eps = 0.005# how close points should be to each other to be considered a part of a c
#     min_samples = 3# the minimum number of points to form a dense region
#     dbscan = DBSCAN( eps=eps, min_samples=min_samples,metric = "cosine" )
#     dbscan_model = dbscan.fit(corpus_vector)



    Sum_of_squared_distances = []
    K = range(10,30)
    for k in K:
      km = KMeans(n_clusters=k, max_iter=200, n_init=10)
      km = km.fit(result)
      Sum_of_squared_distances.append(km.inertia_)
      print(k,":",Sum_of_squared_distances[-1])
    plt.plot(K, Sum_of_squared_distances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Sum_of_squared_distances')
    plt.title('Elbow Method For Optimal k')
    plt.show()




##K-Means on BERT Embedding + PCA
kmeans_pca(encoded_arr_bert_pca)
```

```
     10 : 18663.131569219302
     11 : 16927.739983946252
     12 : 15602.142129586944
     13 : 14478.357857129326
```

```
##K-Means on Doc2Vec Embedding + PCA
kmeans_pca(corpus_vector_pca)
```

```
     10 : 569.8090701311665
     11 : 467.4161627071941
     12 : 400.9215719910248
     13 : 341.5167067297723
     14 : 299.20278725232333
     15 : 257.13844081147477
     16 : 225.46092576452568
     17 : 197.9101003153454
     18 : 168.70950768063526
     19 : 149.42286723988192
     20 : 131.88078796434326
     21 : 116.69723145854144
     22 : 106.5051598335738
     23 : 96.83675401240006
     24 : 86.8586170706276
     25 : 79.53534612236572
     26 : 72.76859250065583
     27 : 66.8428391817057
     28 : 61.57486278307697
     29 : 56.976686917769136
```

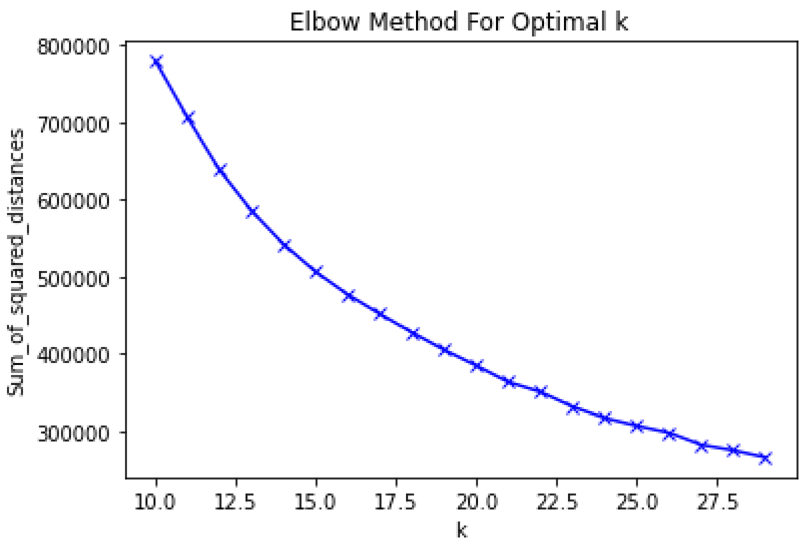


```
#KMeans (WITH Dimensionality Reduction T-SNE)

from sklearn.manifold import TSNE
import matplotlib.pyplot as plt
from sklearn.cluster import KMeans

def kmeans_tsne(result):
    """Function to form dbscan clusters and display them"""
#     eps = 0.005# how close points should be to each other to be considered a part of a c
#     min_samples = 3# the minimum number of points to form a dense region
#     dbscan = DBSCAN( eps=eps, min_samples=min_samples,metric = "cosine" )
#     dbscan_model = dbscan.fit(corpus_vector)
```

```
    Sum_of_squared_distances = []
    K = range(10,30)
    for k in K:
      km = KMeans(n_clusters=k, max_iter=200, n_init=10)
      km = km.fit(result)
      Sum_of_squared_distances.append(km.inertia_)
      print(k,":",Sum_of_squared_distances[-1])
    plt.plot(K, Sum_of_squared_distances, 'bx-')
    plt.xlabel('k')
    plt.ylabel('Sum_of_squared_distances')
    plt.title('Elbow Method For Optimal k')
    plt.show()
```

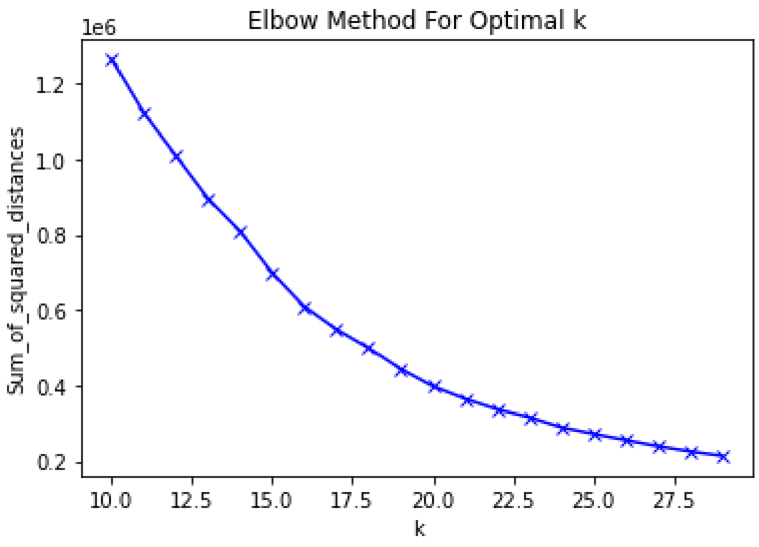
```
##K-Means on BERT Embedding + t-SNE
kmeans_tsne(encoded_arr_bert_tsne)
```

```
     10 : 779583.3333103703
     11 : 706285.015202999
     12 : 638611.8071002079
     13 : 585632.4333104667
     14 : 542154.1778960562
     15 : 506330.44656415435
     16 : 476825.6325201758
     17 : 451676.93344793934
     18 : 427334.46493293735
     19 : 404758.1823942027
     20 : 384374.2826819775
     21 : 363105.69380347856
     22 : 350395.4257919667
     23 : 331397.04073191935
     24 : 316626.9587301924
     25 : 306537.50362020975
     26 : 297687.9655245012
     27 : 282051.3344248115
     28 : 274851.42856734316
     29 : 265712.317972205
```



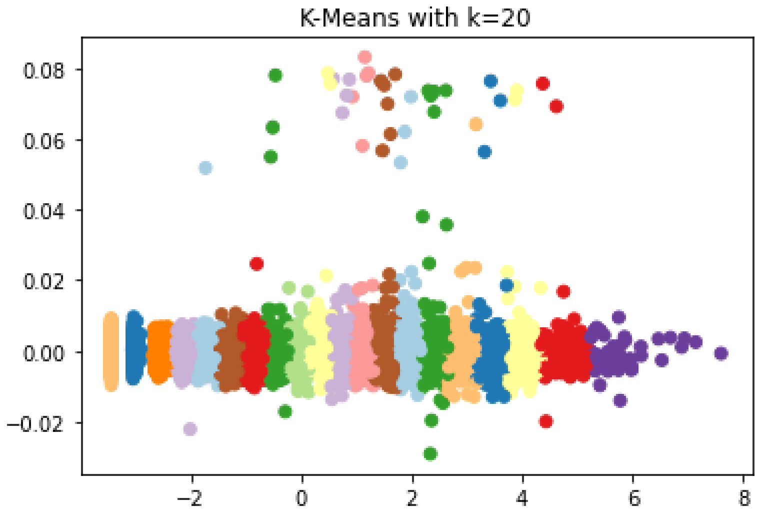
```
##K-Means on Doc2Vec Embedding + t-SNE
```

```
kmeans_tsne(corpus_vector_tsne)

     10 : 1263058.656888814
     11 : 1120722.071145338
     12 : 1007804.7785556671
     13 : 894660.4313426887
     14 : 808396.1118480843
     15 : 698047.2302083657
     16 : 608793.6901322852
     17 : 548380.4938254217
     18 : 498885.3549466238
     19 : 443493.17044337635
     20 : 397763.2520348772
     21 : 365009.9640806653
     22 : 337280.50620825414
     23 : 316093.7917830592
     24 : 289521.8773465089
     25 : 272431.67277330445
     26 : 256032.12558750235
     27 : 239827.14258700315
     28 : 226246.8955806042
     29 : 214393.43441520914
```



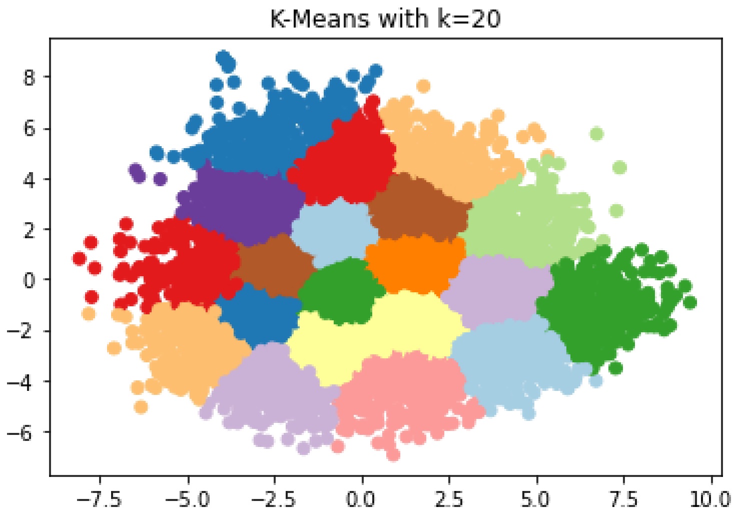
```
def plot_kmeans_pca(true_k, result_pca):
  # pca = PCA(n_components=2)
  # result_pca = pca.fit_transform(corpus_vector)
  # print(result_pca.shape)

  model = KMeans(n_clusters=true_k, init='k-means++', max_iter=200, n_init=10)
  model.fit(result_pca)
  print("SSD:",model.inertia_)
  labels=model.labels_
  print("Labels:",labels)
  y_pred = model.fit_predict(result_pca)
  plt.scatter(result_pca[:,0], result_pca[:,1],c=y_pred, cmap='Paired')
  plt.title("K-Means with k="+str(true_k))
  return labels


 labels_kmeans_pca = plot_kmeans_pca(20, corpus_vector_pca)
```

```
    SSD: 132.46613718925713
    Labels: [15 19  3 ...  6 12  4]
```



```
plot_kmeans_pca(20, encoded_arr_bert_pca)
```

```
    SSD: 9708.433495675923
    Labels: [ 6 10  1 ...  9  9 15]
    array([ 6, 10,  1, ...,  9,  9, 15], dtype=int32)
```

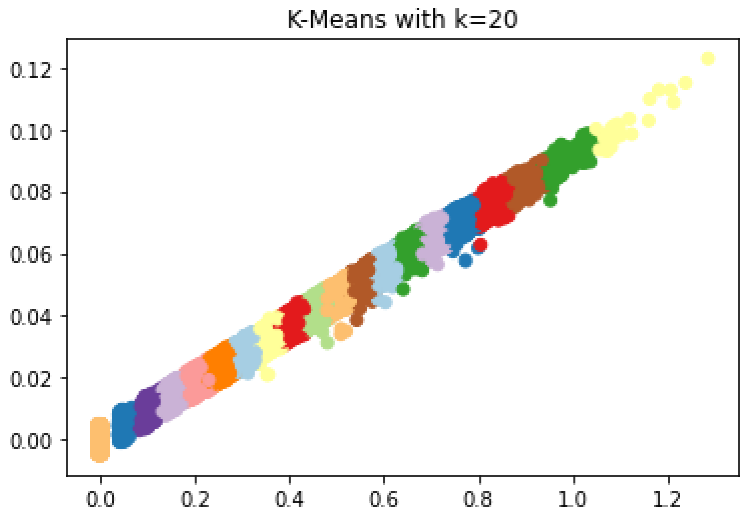


```
plot_kmeans_pca(20, encoded_arr_bert)
```

```
    SSD: 627917.7559826617
    Labels: [ 3 11 11 ...  0 13  6]
```

```
plot_kmeans_pca(20, corpus_vector)
```

```
    SSD: 138.8381352914954
    Labels: [13 17  3 ...  8 10 10]
    array([13, 17,  3, ...,  8, 10, 10], dtype=int32)
```

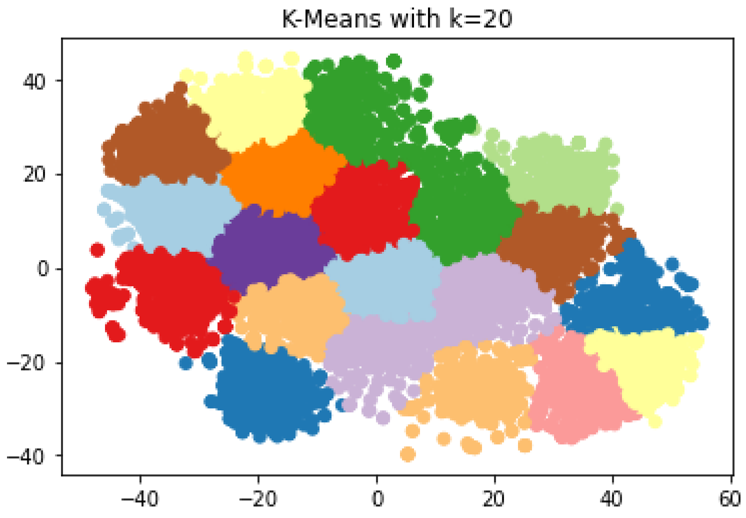


```
def plot_kmeans_tsne(true_k, result_tsne):
  # tsne = TSNE(n_components = 2, init = 'random', random_state = 10, perplexity = 100)
  # # Use only 400 rows to shorten processing time
  # result_tsne = tsne.fit_transform(corpus_vector)
  # print(result_tsne.shape)

  model = KMeans(n_clusters=true_k, init='k-means++', max_iter=200, n_init=10)
  model.fit(result_tsne)
  print("SSD:",model.inertia_)
  labels=model.labels_
  print(labels)
  y_pred = model.fit_predict(result_tsne)
  plt.scatter(result_tsne[:,0], result_tsne[:,1],c=y_pred, cmap='Paired')
  plt.title("K-Means with k="+str(true_k))


plot_kmeans_tsne(20, corpus_vector_tsne)
```

```
    SSD: 397759.4628152194
    [ 2  5 10 ... 13 12 12]
```

```
plot_kmeans_tsne(20, encoded_arr_bert_tsne)
```

```
    SSD: 384877.6035738
    [2 2 2 ... 0 0 7]
```



```
labels1 = labels_kmeans_pca.tolist()


total = 0
for each in range(-1, 21):
  print(each, ":", labels1.count(each))
  total += labels1.count(each)

print(total)
```

```
    -1 : 0
    0 : 472
    1 : 475
    2 : 436
    3 : 1191
    4 : 533
    5 : 269
    6 : 516
    7 : 494
    8 : 55
    9 : 534
    10 : 323
    11 : 532
    12 : 495
    13 : 286
    14 : 334
    15 : 388
    16 : 566
    17 : 158
    18 : 459
    19 : 516
    20 : 0
    9032
```

```
list3 = [i for i, e in enumerate(labels1) if e == 2]
```

```
combined_sent = ""
for each in list3:
  print(sentences_list[each])
  print()
  combined_sent += sentences_list[each]
```

```
sydney australia october ani arjun nair signed big bash league bbl season sydney

abu dhabi uae october ani registering win rajasthan royals mumbai indians bowling

london uk october ani expressing sadness gunnersaurus years midfielder mesut ozil

abu dhabi uae october ani playing match winning knock runs rajasthan royals mumba

abu dhabi uae october ani facing defeat indian premier league ipl rajasthan royal

abu dhabi uae october ani mumbai indians performance trounce rajasthan royals run

uk october ani celtic football club wednesday said odsonne edouard tested coronav

new delhi india october ani skipper virat kohli childhood coach rajkumar sharma s

uk october ani stuart armstrong tested coronavirus scottish football association

london uk october ani arsenal kieran tierney expressed disappointment frustration

abu dhabi uae october ani suffering defeat mumbai indians indian premier league i

abu dhabi uae october ani chennai super kings csk failed chase target kolkata kni

abu dhabi uae october ani kolkata knight riders kkr skipper dinesh karthik praise

brisbane australia october ani women team wednesday equalled world record odi vic

dubai uae october ani order provide fans scenes updates multimedia messaging app

uk october ani arsenal confirmed club closed hale end academy staff member tested

new delhi india october ani completion ahf education workshops hockey india coach

bern switzerland october ani switzerland xherdan shaqiri set fly spain testing co

liverpool uk october ani liverpool thursday announced signing goalkeeper marcelo

meerut uttar pradesh india october ani boxer sunil chauhan thursday thanked union

new delhi india october ani defender sandesh jhingan feels sporting action resume

london uk october ani edouard mendy set miss senegal match morocco injury returne

london uk october ani ollie pope replaced wicket keeper batsman jonny bairstow re

dubai uae october ani kings xi punjab kxip cricketer nicholas pooran said team ch

sharjah uae october ani spinner shane warne hailed rajasthan royals bowling perfo

canterbury uk october ani rounder calum haggett left club following conclusion se
```

```
birmingham uk october ani aston villa women team member tested coronavirus club a

brussels belgium october ani substituted ivory coast match belgium manchester uni

dubai uae october ani match chennai super kings csk royal challengers bangalore r
```

```
wordlist = combined_sent.split()
wordfreq = {}
for w in wordlist:
  if w not in wordfreq:
    wordfreq[w] = 0
  wordfreq[w] += 1
```

```
sorted_words = dict(sorted(wordfreq.items(), key=lambda item: item[1],reverse=True))
print(sorted_words)
```

```
{'said': 452, 'october': 367, 'ani': 283, 'india': 280, 'oct': 177, 'delhi': 168, 'm
```

```
# kmeans_pca(encoded_arr)
```

```
# kmeans(encoded_arr)
```

```
from sklearn.decomposition import PCA

def dbscan(corpus_vector, eps= 0.005, min_samples = 3):
    """Function to form dbscan clusters and display them"""
#     eps = 0.005# how close points should be to each other to be considered a part of a c
#     min_samples = 3# the minimum number of points to form a dense region
#     dbscan = DBSCAN( eps=eps, min_samples=min_samples,metric = "cosine" )
#     dbscan_model = dbscan.fit(corpus_vector)

    pca = PCA(n_components=2)
    result = pca.fit_transform(corpus_vector)
    print(result.shape)
    db = DBSCAN(eps=eps, min_samples=min_samples)
    dbscan_model = db.fit(result)
    #Forming the clusters

    core_samples_mask = np.zeros_like(dbscan_model.labels_, dtype=bool)
    core_samples_mask[dbscan_model.core_sample_indices_] = True
    labels1 = dbscan_model.labels_
    n_clusters_ = len(set(labels1)) - (1 if -1 in labels1 else 0) # Number of clusters in
    print(labels1)
    print(len(labels1))
    print(n_clusters_) # number of clusters

    clusters1 = {} # a dictionary for different cluster
    for c, i in enumerate(labels1):
        if i == -1:
            continue
        elif i in clusters1:
            clusters1[i].append( data[c] )
        else:
            clusters1[i] = [data[c]]

    for c in clusters1: # print the different clusters
        # print("Cluster No."+" "+str(c)+" "+str(clusters1[c]))
        # print()
        pass

    return labels1, clusters1
```

```
labels1, clusters1 = dbscan(corpus_vector,0.005,3)
```

```
     (9032, 2)
     [ 0  1  2 ...  5  4 33]
     9032
     197
```

```
labels1, clusters1 = dbscan(corpus_vector,0.01,3)
```

```
     (9032, 2)
     [0 1 2 ... 1 1 1]
     9032
     32
```

```
labels1, clusters1 = dbscan(encoded_arr_bert,0.005,3)
```

```
     (9032, 2)
     [-1 -1 -1 ... -1 -1 -1]
     9032
     15
```

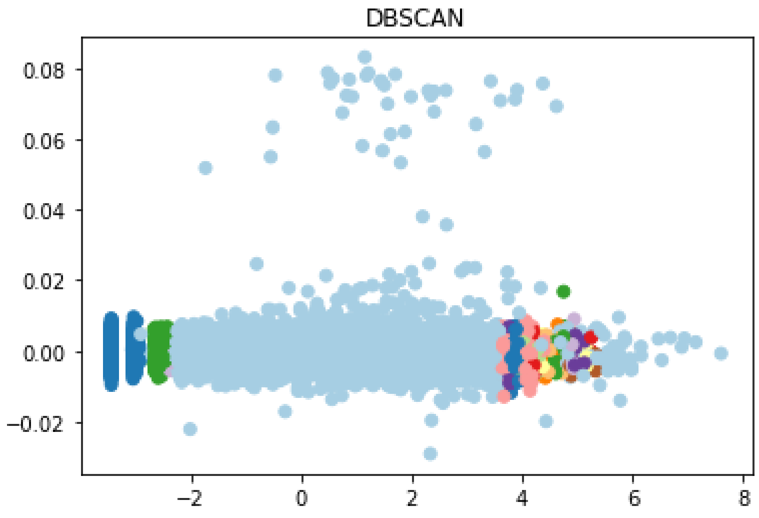
```
labels1, clusters1 = dbscan(encoded_arr_bert,0.02,3)
```

```
     (9032, 2)
     [-1 -1 -1 ... -1 -1 -1]
     9032
     49
```

```
from sklearn.decomposition import PCA

def plot_dbscan(X , eps, min_samples):
    """Function to plot clusters"""
    pca = PCA(n_components=2)
    result = pca.fit_transform(X)
    print(result.shape)
    db = DBSCAN(eps=eps, min_samples=min_samples)
    db.fit(result)
    y_pred = db.fit_predict(result)
    plt.scatter(result[:,0], result[:,1],c=y_pred, cmap='Paired')
    plt.title("DBSCAN")
```

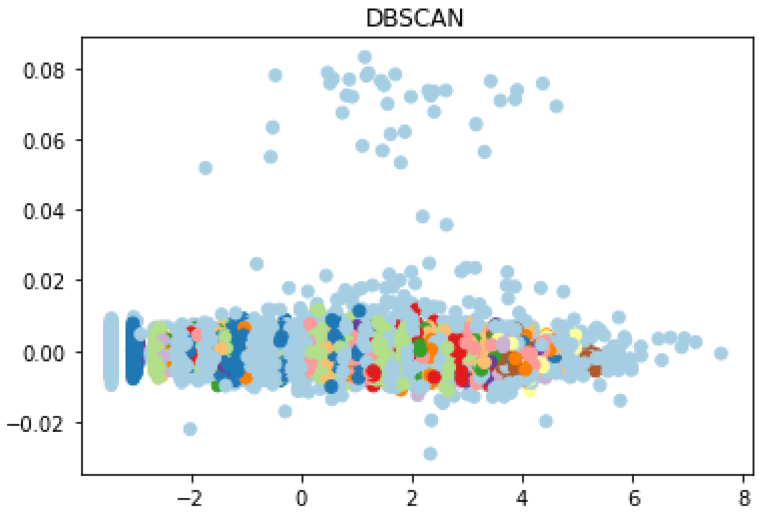
```
plot_dbscan(corpus_vector,0.01,3)
```

```
(9032, 2)
```



```
plot_dbscan(corpus_vector,0.005,3)
```

```
(9032, 2)
```

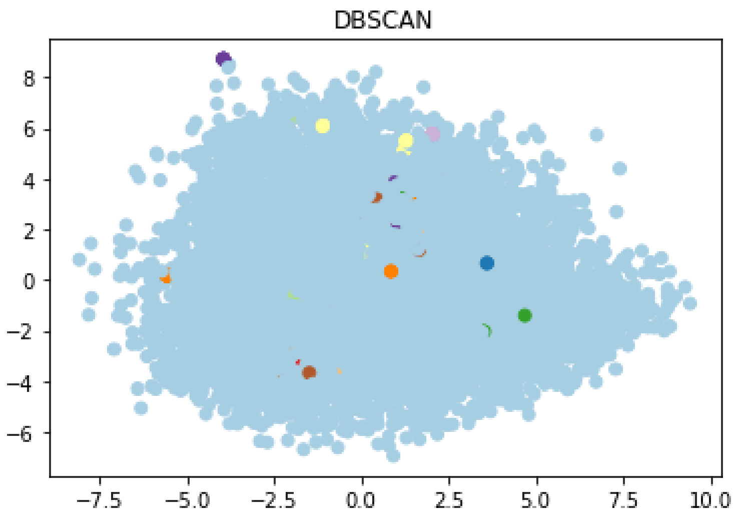


```
plot_dbscan(encoded_arr_bert,0.01,3)
```

```
(9032, 2)
```

```
plot_dbscan(encoded_arr_bert,0.02,3)
```

```
(9032, 2)
```



```
labels1 = labels1.tolist()


total = 0
for each in range(-1, 150):
  print(each, ":", labels1.count(each))
  total += labels1.count(each)

print(total)
```

```
-1 : 177
0 : 301
1 : 33
2 : 1188
3 : 5252
4 : 420
5 : 472
6 : 48
7 : 371
8 : 281
9 : 5
10 : 73
11 : 44
12 : 96
13 : 3
14 : 6
15 : 36
16 : 9
17 : 10
18 : 14
19 : 5
20 : 12
21 : 3
22 : 4
23 : 15
24 : 5
```

```
25 : 8
26 : 23
27 : 5
28 : 4
29 : 11
30 : 18
31 : 7
32 : 7
33 : 7
34 : 7
35 : 5
36 : 12
37 : 3
38 : 4
39 : 4
40 : 8
41 : 4
42 : 3
43 : 3
44 : 3
45 : 3
46 : 0
47 : 0
48 : 0
49 : 0
50 : 0
51 : 0
52 : 0
53 : 0
54 : 0
55 : 0
56 : 0
57 : 0
```

```
[i for i, e in enumerate(labels1) if e == 3]
```

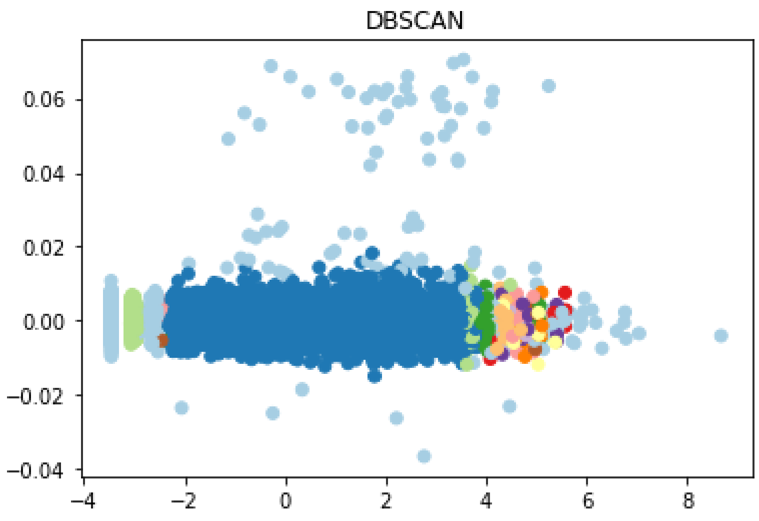
```
[3,
 5,
 7,
 10,
 11,
 13,
 14,
 15,
 23,
 24,
 26,
 27,
 29,
 30,
 31,
 32,
 34,
 36,
 37,
 39,
 40,
 41,
 42,
 45,
 48,
      50,
      52,
      54,
      61,
      63,
      67,
      72,
      73,
      74,
      75,
      76,
      78,
      80,
      84,
      85,
      87,
      88,
      92,
      93,
      97,
      99,
      102,
      103,
      104,
      111,
      112,
      114,
      116,
      117,
      119,
      120,
      127,
      128,
      129,
```

```
print(sentences_list[20])
print()
print(sentences_list[1468])
print()
print(sentences_list[1523])
```

```
abu dhabi uae october ani stumbling defeat hands mumbai indians rajasthan royals wic

new york usa october ani newsvoir support expansion testing contact tracing india ro

new delhi india october ani newsvoir arjun anand author art photographer launched bo
```

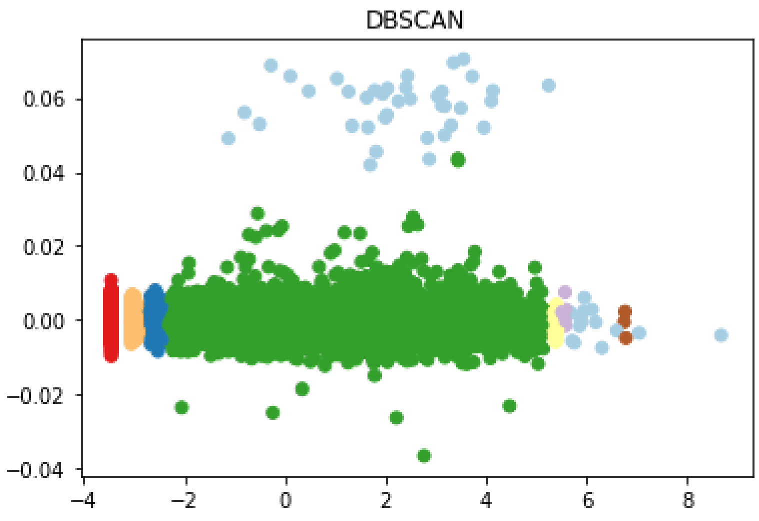
```
from sklearn.decomposition import PCA

def plot_dbscan(X , eps, min_samples):
    """Function to plot clusters"""
    pca = PCA(n_components=2)
    result = pca.fit_transform(X)
    print(result.shape)
    db = DBSCAN(eps=eps, min_samples=min_samples)
    db.fit(result)
    y_pred = db.fit_predict(result)
    plt.scatter(result[:,0], result[:,1],c=y_pred, cmap='Paired')
    plt.title("DBSCAN")

plot_dbscan(corpus_vector,0.01,3)
```

```
(9032, 2)
```



```
plot_dbscan(corpus_vector,0.03, 3)
```

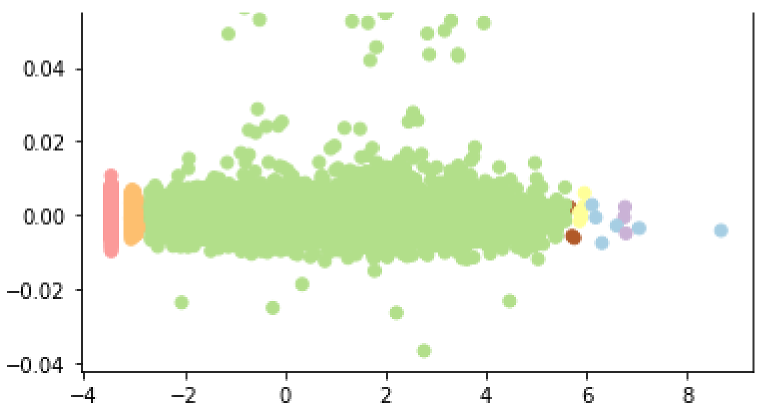
```
(9032, 2)
```



```
plot_dbscan(corpus_vector,0.05, 3)
```

```
(9032, 2)
```

DBSCAN



0.04
0.02
0.00
−0.02
−0.04
−4
−2
0
2
4
6
8