```
import numpy as np
import pandas as pd
import nltk
import re
import string
import scipy.sparse as sp
import matplotlib.pyplot as plt
from sklearn.preprocessing import normalize


from google.colab import drive

drive.mount('/content/gdrive')
```

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive
```

```
raw_articles_data = pd.read_excel('/content/gdrive/MyDrive/data/news_dataset.xlsx')


raw_articles_data
```

		Verge'}		India	voice DMs o...
1	1	{'id': 'engadget', 'name': 'Engadget'}	Daniel Cooper	Amazon follows Netflix with mobile-only video ...	Amazon Prime Video and Bharti Airtel, India's ...
2	2	{'id': 'techcrunch', 'name': 'TechCrunch'}	Manish Singh	India bans PUBG and over 100 additional Chines...	India has banned more than 100 additional Chin...
3	3	{'id': 'engadget', 'name': 'Engadget'}	Steve Dent	Samsung begins offering support requests via W...	With the COVID-19 crisis continuing unabated i...
4	4	{'id': 'engadget', 'name': 'Engadget'}	Mariella Moon	Sony is launching the PS5 in India on February...	PlayStation gamers in India will finally have ...
...	...	...	...	...	...
9895	9895	{'id': None, 'name': 'New	Allyson Waller	A Cat Is Said to Be Joining the	The last cat to live in the White House

```
titles=[]
dates=[]
descriptions=[]
contents=[]
for index,item in raw_articles_data.iterrows():
  titles.append(item['title'])
  dates.append(item['publishedAt'])
  descriptions.append(item['description'])
  contents.append(item['content'])
```

India

```
dataset=pd.DataFrame({'title': titles, 'date': dates, 'desc': descriptions, 'content': con
dataset=dataset.drop_duplicates(subset='title').reset_index(drop=True)
dataset=dataset.dropna()
```

```
dataset.head()
```

	title	date	desc	content
0	Twitter’s voice DMs arrive in India	2021-02-17T13:18:32Z	Twitter has rolled out support for voice DMs	For when theres just way too much to type\r\n\...

```
dataset.shape
```

```
(100, 4)
```

...

```
# Create function to process and tokenize raw texts
def preprocess(text, stopwords={}, lemmatizer=nltk.stem.wordnet.WordNetLemmatizer()):
    # Lower case
    text = text.lower()
    # Handle URL
    text = re.sub(r"https?://t.co/\w{10}"," ", text)
    # Deal with "'s"
    text = re.sub(r"'s", "", text)
    # Deal with "'"
    translator2 = str.maketrans({key: None for key in string.punctuation[6]})
    text = text.translate(translator2)
    # Deal with the rest of punctuations
    translator3 = str.maketrans(string.punctuation, ' '*len(string.punctuation))
    text = text.translate(translator3)
    # Handle unicode
    text = re.sub(r'[^\x00-\x7F]+',' ', text)
    # Split the text
    r1 = nltk.word_tokenize(text)
    # Lemmatize the text
    r2 = [lemmatizer.lemmatize(word) for word in r1]
    # Remove the stopwords
    r3 = [word for word in r2 if not word in stopwords]
    # Remove digits
    r4 = [word for word in r3 if word.isalpha()]
    return r4


# Import NLTK stopwords
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
extra_stopwords = set()
stopwords = set(nltk.corpus.stopwords.words('english')) | extra_stopwords
```

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

```
# Put the preprocessed texts into a list
articles = []
from collections import defaultdict
import math
```

```
DF = defaultdict(int)
for i in range(0,dataset.shape[0]):
    tokenized_text = preprocess(dataset['content'][i], stopwords)
    words = tokenized_text
    for word in set(words):
        if len(word) >= 3 and word.isalpha():
            DF[word] += 1
    articles.append(' '.join(tokenized_text))


def cluster_centroids(DF, gt=0.1, to=100):
  centroids=[];
  for x, y in DF.items():
    z=y/to
    if z>gt:
      centroids.append(x)
  return centroids



centroids=cluster_centroids(DF)


centroids
```

```
['india',
 'char',
 'service',
 'largest',
 'second',
 'announced',
 'new',
 'market',
 'world',
 'facebook',
 'said',
 'country',
 'year',
 'million',
 'indian']
```

```
len(centroids)
```

```
15
```

```
#Cluster initialization
def cluster_in_table(centroids,article_check):
  clusters=set()
  words=article_check;
  for word in words:
   for i in range(len(centroids)):
    #  print(i)
     if centroids[i]==word:
       clusters.add(i)

  if(len(clusters)==0):
    clusters.add(0)
```

```
      clusters.add(0)
    final_cluster=[]
    for i in clusters:
      final_cluster.append(i)

    return final_cluster
```

```
cluster_table=[]
for i in range(0,dataset.shape[0]):
  tokenized_text = preprocess(dataset['content'][i], stopwords)
  clusters =cluster_in_table(centroids,tokenized_text)
  cluster_table.append(clusters)
```

```
cluster_table
```

```
      [1, 10, 13, 6],
      [0, 1, 5],
      [1, 11, 14],
      [1, 10, 13],
      [0, 1, 5, 7],
      [0, 8, 1],
      [0, 1, 5],
      [0, 1],
      [9, 1],
      [0, 9, 10, 1],
      [0, 1, 2, 6],
      [0, 1, 3, 4, 7, 8, 12],
      [1, 13, 14],
      [0, 1, 10, 12, 13],
      [0, 1, 6],
      [0, 1],
      [0, 1, 6],
      [1],
      [1],
      [0, 1, 11, 14],
      [0, 1, 11],
      [0, 1, 4, 7, 8, 10, 14],
      [0, 9, 11, 1],
      [0, 1, 2, 3, 4, 7, 8, 12],
      [0, 1, 2, 3, 4, 8],
      [0, 1, 3, 5, 10],
      [0, 1, 10],
      [0, 1, 5, 6],
      [0, 9, 11, 1],
      [0, 1, 3, 4, 7, 8],
      [0, 1, 6],
      [1, 11, 6],
      [0, 1],
      [9, 12, 14, 1],
      [1],
      [9, 1],
      [1, 2],
      [1, 6],
      [0, 1, 14],
      [0, 1, 2, 6],
      [0, 1],
```

```
        [0, 1, 11, 6],
        [0, 1],
        [0, 1],
        [8, 9],
        [0, 1],
        [0, 1, 13],
        [1, 14],
        [1],
        [0, 1],
        [0, 1],
        [0, 1, 11, 14],
        [1, 14],
        [0, 1, 5, 6, 11],
        [0, 1],
        [1, 10, 12],
        [8, 1, 3],
        [0, 1],
        [1],
        [0, 9, 5, 1]]
```

```
len(cluster_table)
```

```
    100
```

```
def counter_and_articles(table):
  cluster_articles=[]
  for i in range(len(centroids)):
    temp=[]
    cluster_articles.append(temp)

  for i in range(0,dataset.shape[0]):
    for j in range(len(table[i])):
      cluster_articles[table[i][j]].append(i)

  cluster_counter=[]
  for i in range(len(centroids)):
    cluster_counter.append(len(cluster_articles[i]))

  return (cluster_articles,cluster_counter)
```

```
articles_in_cluster,counter=counter_and_articles(cluster_table)
```

```
len(articles_in_cluster)
```

```
    15
```

```
articles_in_cluster[0]
```

```
     20,
     21,
     22,
     23,
     24,
     25,
```

```
        26,
        27,
        28,
        29,
        30,
        31,
        32,
        33,
        35,
        36,
        37,
        38,
        39,
        41,
        44,
        45,
        46,
        47,
        49,
        50,
        51,
        53,
        54,
        55,
        56,
        59,
        60,
        61,
        62,
        63,
        64,
        65,
        66,
        67,
        68,
        69,
        70,
        72,
        78,
        79,
        80,
        81,
        82,
        83,
        85,
        86,
        89,
        90,
        91,
        93,
        94,
        97,
        99]
```

```
counter
```

```
[77, 99, 13, 18, 14, 11, 19, 16, 17, 12, 13, 16, 13, 12, 11]
```

```
def TP_function(cluster_set) :
```

```
    import math
    # lambda = (|cx| - 1) * w^2 where w = 2

    lamb = (len(cluster_set) - 1) * 4

    #theta = summation of |di - di+1| ^ 2

    theta = 0
    c_list = list(cluster_set)

    for i in range(len(c_list) - 1):
        theta = theta + (c_list[i] - c_list[i+1]) * (c_list[i] - c_list[i+1])

    # TP = e^(lambda - theta) / (1 + e^(lambda - theta))
    # print(lamb-theta)
    expo = math.exp(lamb - theta)
    tp = expo / (1 + expo)
    return tp


def cs(articles_of_cluster_i,cluster_table,cluster_index):
  counter_1=0
  for i in range(len(articles_of_cluster_i)):
    for j in range(len(cluster_table[i])):
      if cluster_index==cluster_table[i][j]:
        counter_1=counter_1+1

  return counter_1/(len(articles_of_cluster_i))




def tfidf(article_index,cluster_index):
  t_f=0
  words=articles[article_index].split()
  # print(centroids[cluster_index])
  for word in words:
    # print(word)
    if word==centroids[cluster_index]:
      t_f=t_f+1
  # print(t_f)
  # print('\n')
  # print('\n')
  return t_f*(math.log(100/DF[centroids[cluster_index]],2))




def fitness(articles_of_cluster_i,cluster_table,cluster_index,article_index):
  ans=0
  for final_index in cluster_table[article_index]:
    # tp_val=TP_function(articles_of_cluster_i)
    tp_val=1
    # print(tp_value)
```

```
      cs_val=cs(articles_of_cluster_i,cluster_table,final_index)
      # print(cs_val)
      tfidf_val=tfidf(article_index,final_index)
      # print(tfidf_value)
      ans+=(tp_val*cs_val*tfidf_val)

    return ans



#cluster finalization
cluster_final_table=[]
for i in range(0,15):
  temp=[]
  cluster_final_table.append(temp)

for i in range(0,dataset.shape[0]):
  v=-1e100
  ind=-1
  for j in range(len(cluster_table[i])):
    if(v<fitness(articles_in_cluster[cluster_table[i][j]],cluster_table,cluster_table[i][j
      ind=j
      v=fitness(articles_in_cluster[cluster_table[i][j]],cluster_table,cluster_table[i][j]
  cluster_final_table[ind].append(i)



cluster_final_table

       32,
       33,
       34,
       35,
       36,
       37,
       38,
       39,

       41,
       43,
       44,
       45,
       46,
       47,
       50,
       51,
       52,
       53,
       54,
       55,
       56,
       57,
       58,
       63,
       67,
       69,
       70,
       72,
       74,
```

```
        77,
        79,
        80,
        82,
        83,
        85,
        86,
        87,
        88,
        89,
        90,
        92,
        94,
        97,
        98,
        99],
       [42, 48, 49, 59, 71, 75, 76, 78, 84, 95, 96],
       [1, 6, 8, 9, 10, 12, 14, 20, 60, 62, 64, 66, 68, 81, 91],
       [16, 22, 40, 65, 73],
       [93],
       [61],
       [],
       [29],
       [],
       [],
       [],
       [],
       [],
       [],
       []]

# nc2 function
def nc2(cluster_tables):
  event_list = []
  for i in range(0, 15):
    for j in range(len(cluster_tables[i])):
      for k in range(j+1, len(cluster_tables[i])):
        temp=[]
        temp.append(cluster_tables[i][j])
        temp.append(cluster_tables[i][k])
        event_list.append(temp)
  return event_list


def calc(ga_event_lists,ta_event_lists):
  comm=0
  for i in range(len(ga_event_lists)):
    for j in range(len(ta_event_lists)):
      if ga_event_lists[i] == ta_event_lists[j]:
        comm = comm + 1
  ca=comm
  ga=len(ga_event_lists)/15
  ta=len(ta_event_lists)/1.5
  return ca,ga,ta


# nc2 GA and TA

#GA list
```

```
ga_event_list = nc2(cluster_final_table)

#TA list
ta_event_list = nc2(cluster_table)


# CA = common tuples from GA and TA
ca,ga,ta =calc(ga_event_list,ta_event_list)


#recall and precision
rec=ca/ta
pre=ca/ga
print(rec)
print(pre)
```

```
0.6055045871559632
0.285097192224622
```

```
f1=2*pre*rec/(pre+rec)
print(f1)
```

```
0.3876651982378854
```

✓ 0s completed at 21:16