Machine Learning Theory (CS351) Report

Discovering event episodes from sequences of online news articles: A time adjoining frequent itemset-based clustering method

Team Members:

1. Mayur Bhat(181CO132)
2. Sukruth N Bhat(181CO154)

1. INTRODUCTION

Environmental Surveillance is conducted by an organization in the business environment to identify important events concerning customers, competitors, industry, technology, broad economic conditions, and government policies and regulations. Information gained from this kind of surveillance helps in strategy formulations, decision making, and business actions which is crucial to their performance and competitiveness. In this paper focus is online news articles. Online news articles have become an integral part of environmental surveillance because of unprecedented growth of the internet. Companies use Event evolution patterns (EEPs) in order to perform environmental surveillance. EEPs actually show the evolution of a particular event type over time which helps them to get prepared for similar events in the near future. EEPs do this by discovering from sequences of news articles (or documents) a common evolution pattern for distinct events of the same type. Focus is given for temporal relationships as well. There are several different event structures or taxonomies discussed in this paper.

Some of them are :

1. Story→Event→Topic
2. Story→Simple Event→Complex Event
3. Story→Component Event→Event

4. Story→Episode→Event.

In this project, Story→Episode→Event structure has been followed.

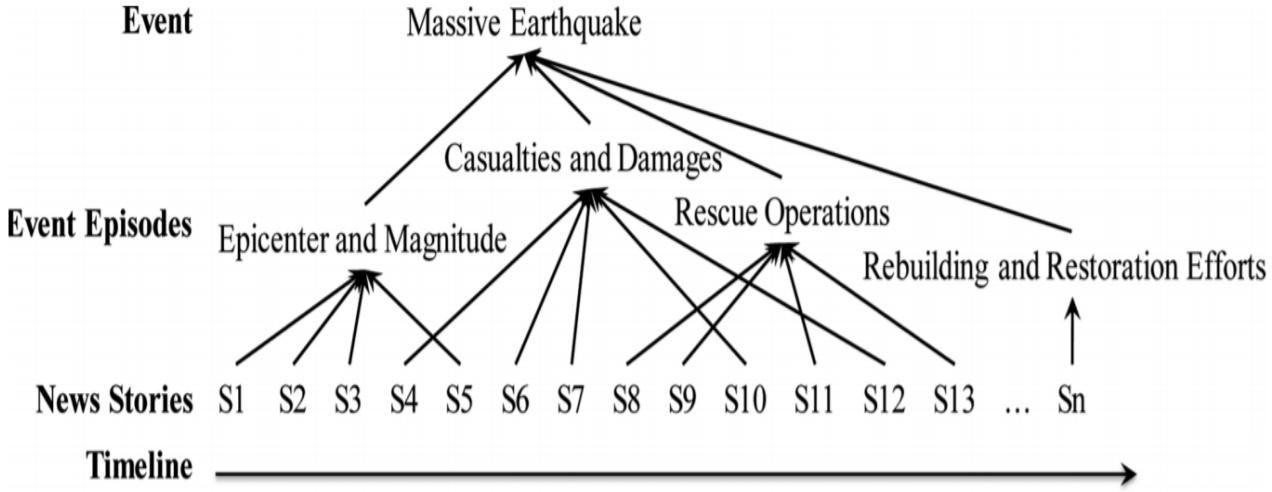


Fig1:- Example for the Story→Episode→Event structure

This image explains the Earthquake event. The initial episode may focus on the date and time, location, and magnitude of the quake; subsequent episodes could relate to casualties and damages, rescue operations, logistic support and survivor placement, and rebuilding and restoration efforts. These news articles sometimes might have similar content but their themes will differ noticeably thereby they represent different episodes of the same event. Before performing EEP,identifying and grouping articles which represent distinct episodes of an event from a sequence of news articles (documents) that pertain to that event is needed. The traditional, manual approach to event episode discovery is ineffective and infeasible. There are two approaches mentioned in this paper for doing the same. They are retrospective event detection and event episode discovery.

2. Retrospective Event Detection and Event Episode Discovery.

Retrospective event detection techniques generally discover events from a stream of news articles. Basically it identifies events from a stream of articles by segmenting the different events described by these articles. Event episode discovery focuses on how an event evolves through different development stages and identifying news articles that pertain to each stage. Let’s consider a collection of articles about pandemics in the

21st century. If Retrospective event detection is used it may reveal distinct events i.e. Covid 19, Ebola outbreak in Africa and such other events and find news articles associated with each event. But with event episode discovery we will be able to identify the different development stages of these events over time i.e. if different stages of development of covid 19 is considered as the event then it will be identified by episodes such as initial breakout of Covid 19 in china, development of vaccines for covid 19 will be the episodes of covid19 and find news articles pertinent to each stage. Overall, event episode discovery identifies distinct episodes of an event from a sequence of news articles pertinent to that event, whereas retrospective event detection identifies events from a stream of articles by segmenting the different events described by these articles. As a result, event episode discovery tends to perform analyses at a finer-grained level than retrospective event detection. The above reasons make event episode discovery more suitable for discovering event episodes from sequences of online news articles.

3. Frequent itemset-based hierarchical clustering

In FIHC, the documents are called transactions and the features of a document are called items. Items with a document frequency greater than the prespecified minimum (gf) threshold are identified as frequent features (items) which become the cluster centroids. These items are also called cluster labels. A document dj based of its own frequent items is initially assigned a set of candidate clusters. The most appropriate cluster for the document are determined according to the goodness of fit which is measured by a variable score.The score for a doc dj in the cluster cx is calculated by using the formula shown below

$$Score(c_x \longleftarrow d_j) = [\sum_i (n(t_i) \times Cluster\_Support(t_i))] - [\sum_i (n(t_i') \times Global\_Support(t_i'))]$$

Here ti represents a global frequent item in document dj and is also a frequent item in cluster cx, ti' denotes a global frequent item in dj but not a frequent item in cx, n(ti) indicates the weight of ti in dj, n(ti') is the weight of ti' in dj, Cluster_Support(ti) reveals the percentage of the documents in cx that contain ti, and Global_Support(ti') depicts the percentage of the entire documents that contain ti'.

4. Time-Adjoining Frequent Itemset-based Event-Episode Discovery (TAFIED) method

TAFIED groups news articles by using both frequent items as cluster centroids as well as the temporal adjacency of documents in a cluster to ensure goodness of fit between a cluster and a document. In addition to steps mentioned in FHIC the adjacency of the respective time stamps of different news articles that belong to the same cluster (event episode) is property weighted. Thus, TAFIED can create clusters in which documents are temporally adjacent and share features that frequently appear in a stream of news articles. The overall processing of this method consists of document preprocessing, cluster initialization, cluster distinction, and cluster adjustment.

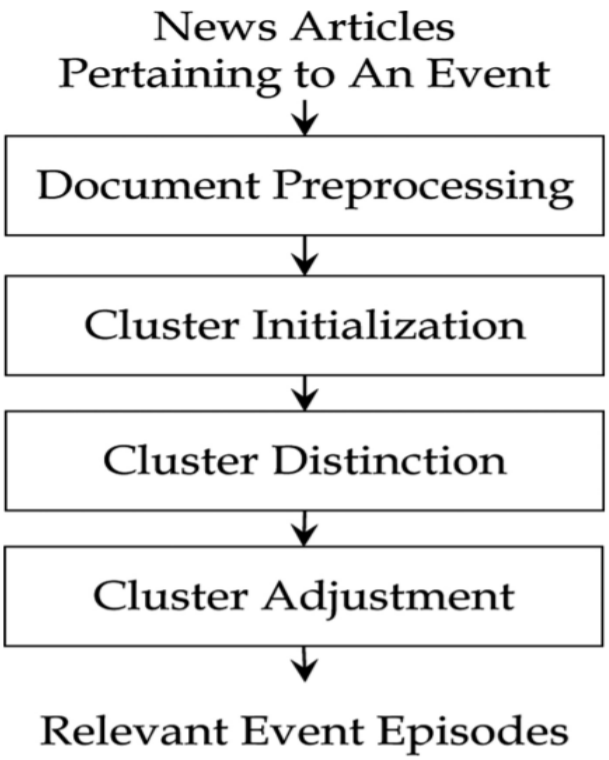


Fig2:- Steps in the TAFIED method

4.1. Document Preprocessing

In **document preprocessing,** meaningful terms like nouns, noun phrases, and verbs from each news article are extracted and a parser to select nouns, noun phrases, and verbs from the article is also used. Stop words such as non semantic-bearing words are also removed, and the remaining words are stemmed into their respective original forms. Basically the document is made ready for future steps.

4.2. Cluster Initialization

In **cluster initialization**, TAFIED constructs a set of initial clusters and assigns each news article to candidate clusters according to its own frequent items. Frequent items (terms) are first identified from the entire corpus of news articles under analysis. The process is similar to FIHC Term ti is a frequent item if the ratio between its document frequency (number of news articles with term ti) and the total number of news articles exceeds the prespecified minimum global support gt. By viewing each frequent item as a class label, our method can create a set of initial clusters. News articles then get assigned to candidate clusters on the basis of its own frequent items (class labels). We may have as many clusters as the number of frequent items identified, and a news article can be labeled as a member of multiple clusters.

4.3. Cluster Distinction

After cluster initialization, each news document will have at least one candidate cluster. Each news article should belong to one and only one event episode so **cluster distinction** has to be done. During this process, TAFIED assesses the fit between a document and each candidate cluster, selects the most appropriate cluster, and generates a final set of clusters. A fitness function is used for this purpose

$$Fitness(c_x \longleftarrow d_j) = \sum_{i=1}^{|T|} (\alpha \times CS(t_i, c_x) \times TFIDF(t_i, d_j) \times TP(c_x))$$

Likelihood that a document dj belongs to a cluster cx a fitness function has been defined by this function. The main idea behind this function was that a news article describing a specific event episode should share features of high appearance frequency and temporal adjacency with articles about that same episode, compared with articles pertaining to another episode.

T is a set of frequent items, ti denotes a frequent item, CS(ti, cx) is the cluster support calculated as the percentage of documents in cx that contain ti, α is a parameter to control the impact direction of ti, TFIDF(ti, dj) represents the within-document term frequency × inverted document frequency for ti appearing in dj, and TP(cx) is a temporal proximity (TP) function for measuring the temporal adjacency of documents when dj is assigned to cx.

Table 1
Example of Term Frequency of Frequent Items in Each Document.

	t_1	t_2	t_3	t_4	t_5
d_1	3	-	2	-	-
d_2	5	-	2	2	-
d_3	7	-	-	3	4
d_4	1	-	1	4	-
d_5	2	-	-	5	-
d_6	-	2	-	-	2
d_7	-	3	-	-	-
d_8	-	1	16	-	2
d_9	-	1	-	-	1
d_{10}	2	-	12	-	-

Fig 3 : Table showing term frequency of frequent items in each document.

Table 2
Example Initial Clusters and Their Respective Member Documents.

Initial Set of Clusters	Member Documents
c_{t1}	$d_1, d_2, d_3, d_4, d_5, d_{10}$
c_{t2}	d_6, d_7, d_8, d_9
c_{t3}	$d_1, d_2, d_4, d_8, d_{10}$
c_{t4}	d_2, d_3, d_4, d_5
c_{t5}	d_3, d_6, d_8, d_9

Fig 4 : Table showing the documents in each cluster.

$$Fitness(c_{t1} \longleftarrow d_3) = ((1 \times \tfrac{6}{6} \times 7 \times \log_2 \tfrac{10}{6}) + (1 \times \tfrac{4}{6} \times 3 \times \log_2 \tfrac{10}{4}) + (-1 \times \tfrac{1}{6} \times 4 \times \log_2 \tfrac{10}{4})) \times \frac{e^{20-29}}{1+e^{20-29}} = 0.001,$$

where $\lambda_{t1} = |6-1| \times 2^2 = 20$ and $\theta_{t1} = |2-1|^2 + |3-2|^2 + |4-3|^2 + |5-4|^2 + |10-5|^2 = 29.$

$$Fitness(c_{t4} \longleftarrow d_3) = ((1 \times \frac{4}{4} \times 7 \times \log_2 \frac{10}{6}) + (1 \times \frac{4}{4} \times 3 \times \log_2 \frac{10}{4}) + (-1 \times \frac{1}{4} \times 4 \times \log_2 \frac{10}{4})) \times \frac{e^{12-3}}{1+e^{12-3}} = 7.802$$

where $\lambda_{t4} = |4-1| \times 2^2 = 12$ and $\theta_{t4} = |3-2|^2 + |4-3|^2 + |5-4|^2 = 3.$

$$Fitness(c_{t5} \longleftarrow d_3) = ((-1 \times \frac{1}{4} \times 7 \times \log_2 \frac{10}{6}) + (-1 \times \frac{1}{4} \times 3 \times \log_2 \frac{10}{4}) + (1 \times \frac{4}{4} \times 4 \times \log_2 \frac{10}{4})) \times \frac{e^{12-14}}{1+e^{12-14}} = 0.358,$$

where $\lambda_{t5} = |4-1| \times 2^2 = 12$ and $\theta_{t5} = |6-3|^2 + |8-6|^2 + |9-8|^2 = 14.$

Fig 5 : Example calculation of fitness of document 3 in cluster 1, 4 and 5 respectively.

The fitness scores of d3 with respect to candidate clusters 1, 4 and 5 thus are 0.001, 7.802, and 0.358, respectively. Thus, document d3 is assigned to cluster 4.

4.4. Cluster Adjustment

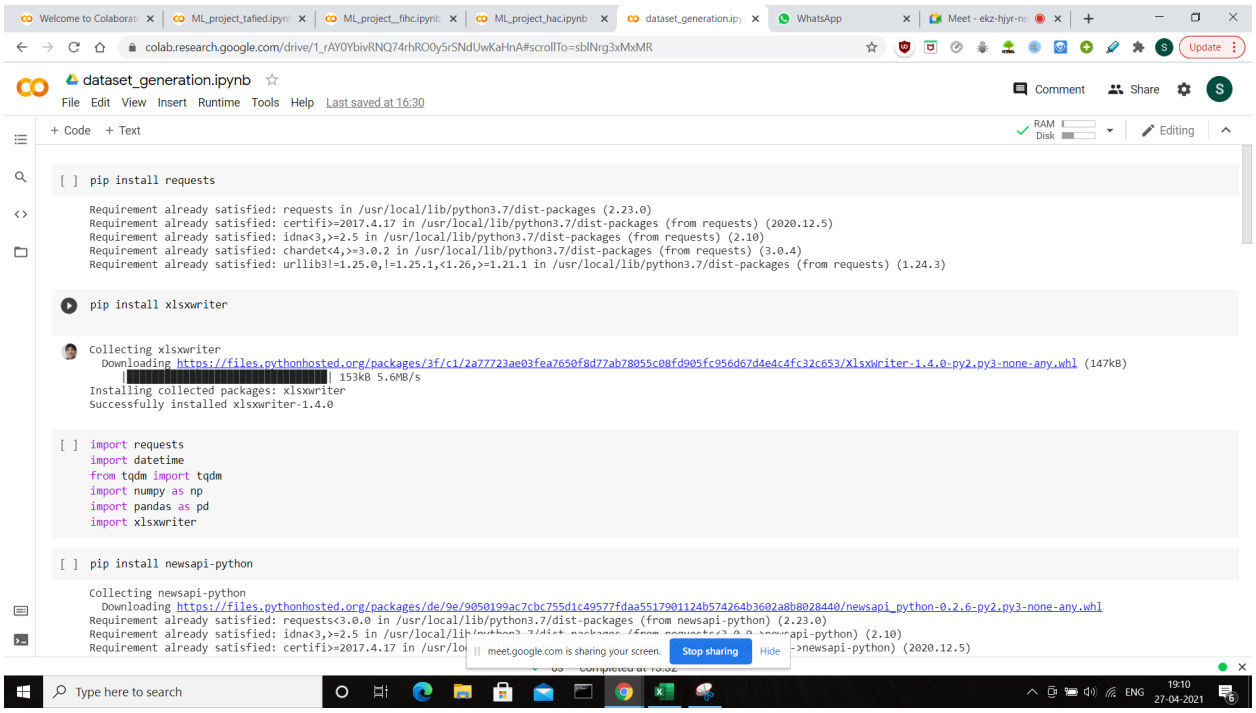
Sometimes the use of frequent items as the base to cluster articles could lead to news articles that describe the same event episode scattered across multiple clusters after cluster distinction. As a remedy, we assess the need to merge two clusters in the **cluster adjustment** step. To perform cluster adjustment, TAFIED merges the clusters that contain highly similar or relevant documents. A combined cohesion measure evaluates the appropriateness of merging two clusters.

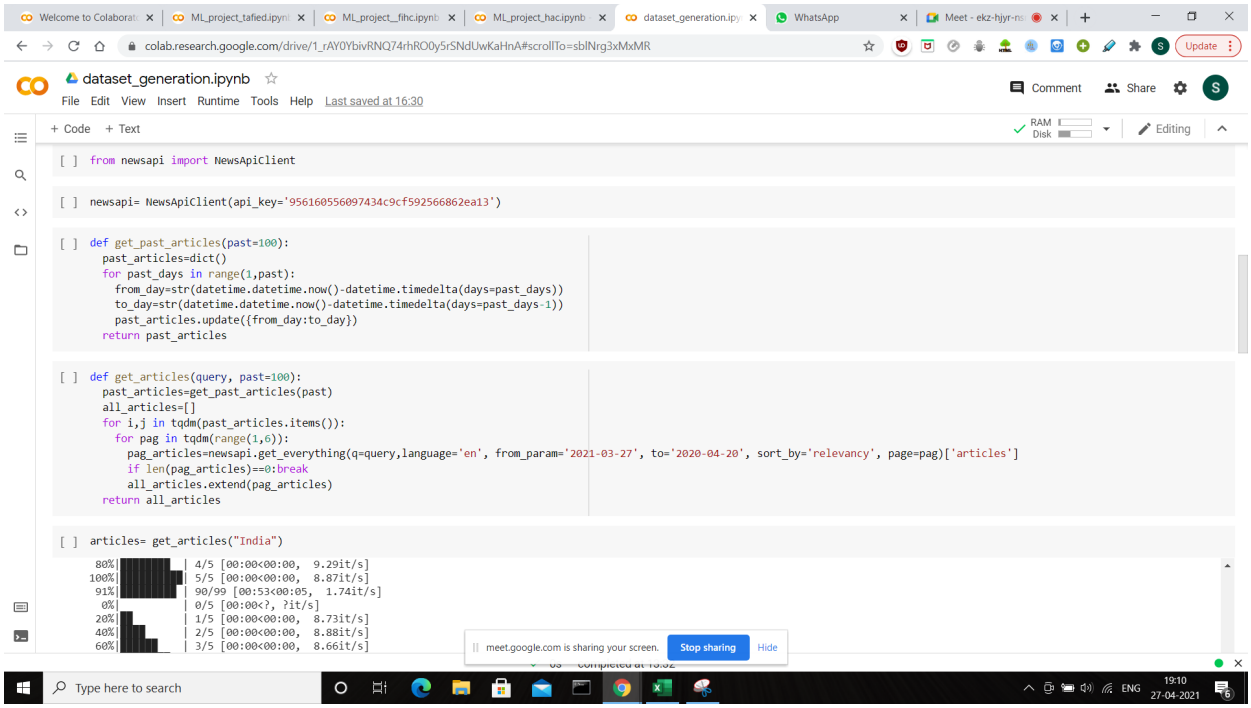
Event episode discovery should properly consider two issues: news articles describing different episodes of a particular event have similar content, and different episodes could emerge concurrently within a time window. TAFIED satisfies both the above requirements as it basically extends FHIC by incorporating temporal locality, according to the fit between a cluster and a document.

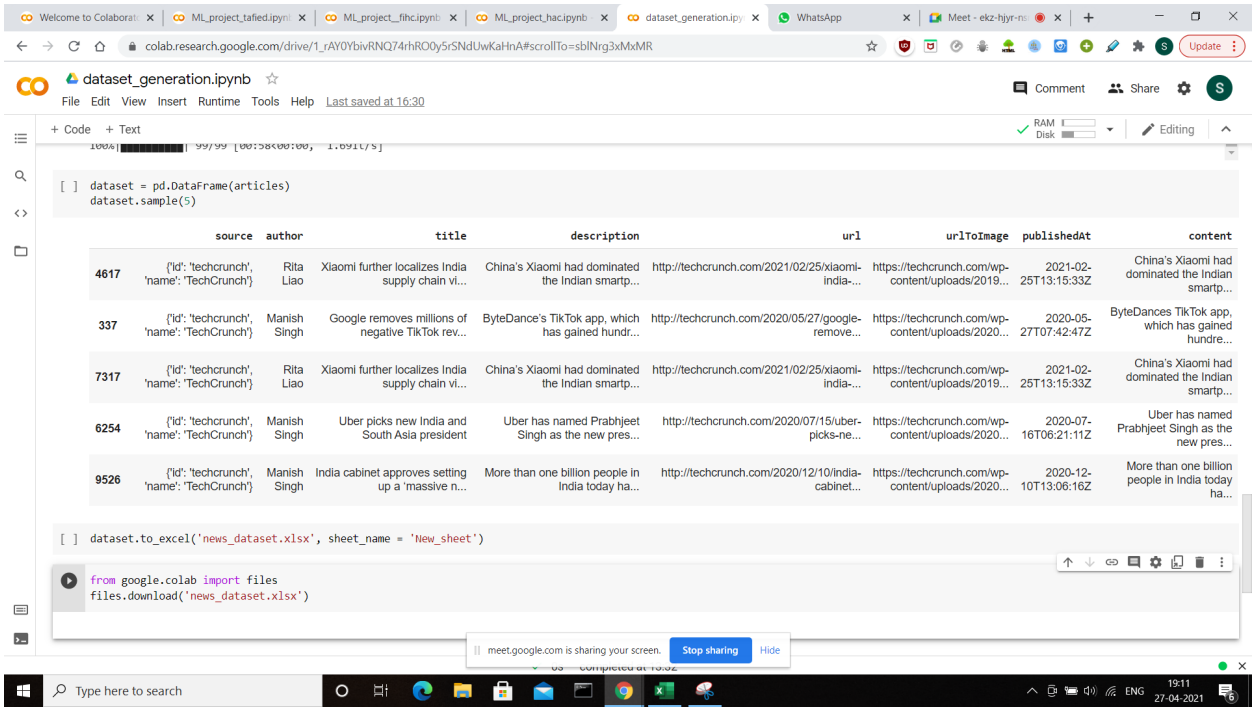
5. IMPLEMENTATION

5.1. Dataset Generation

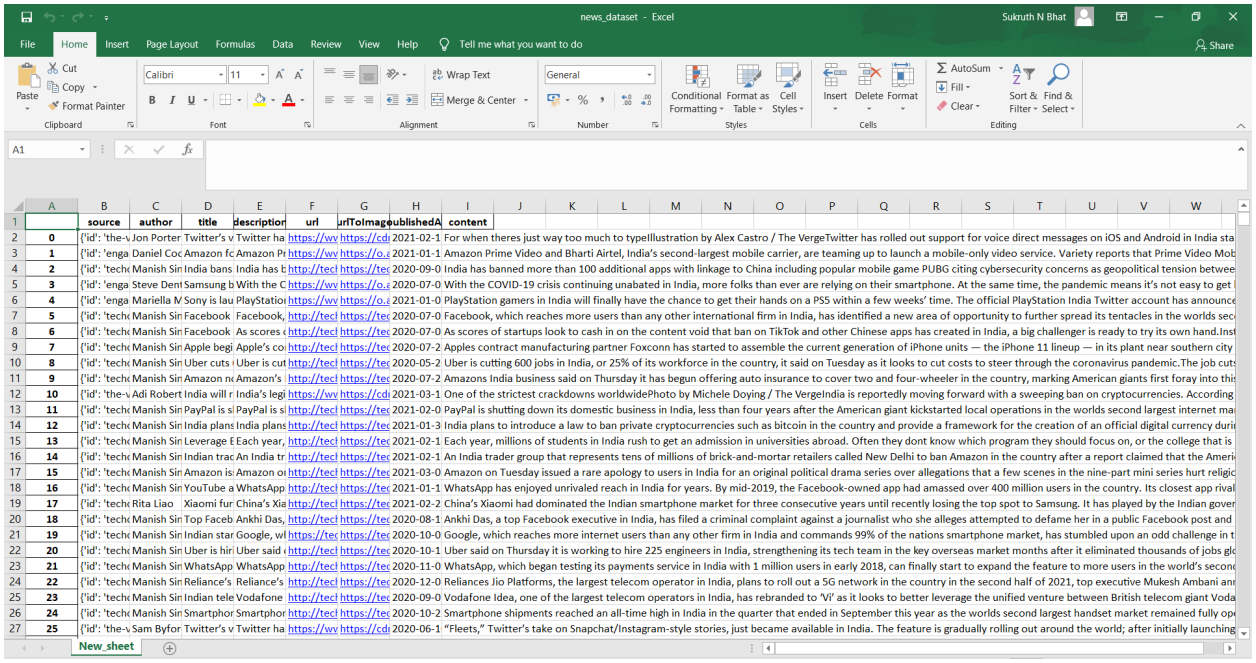
Python code



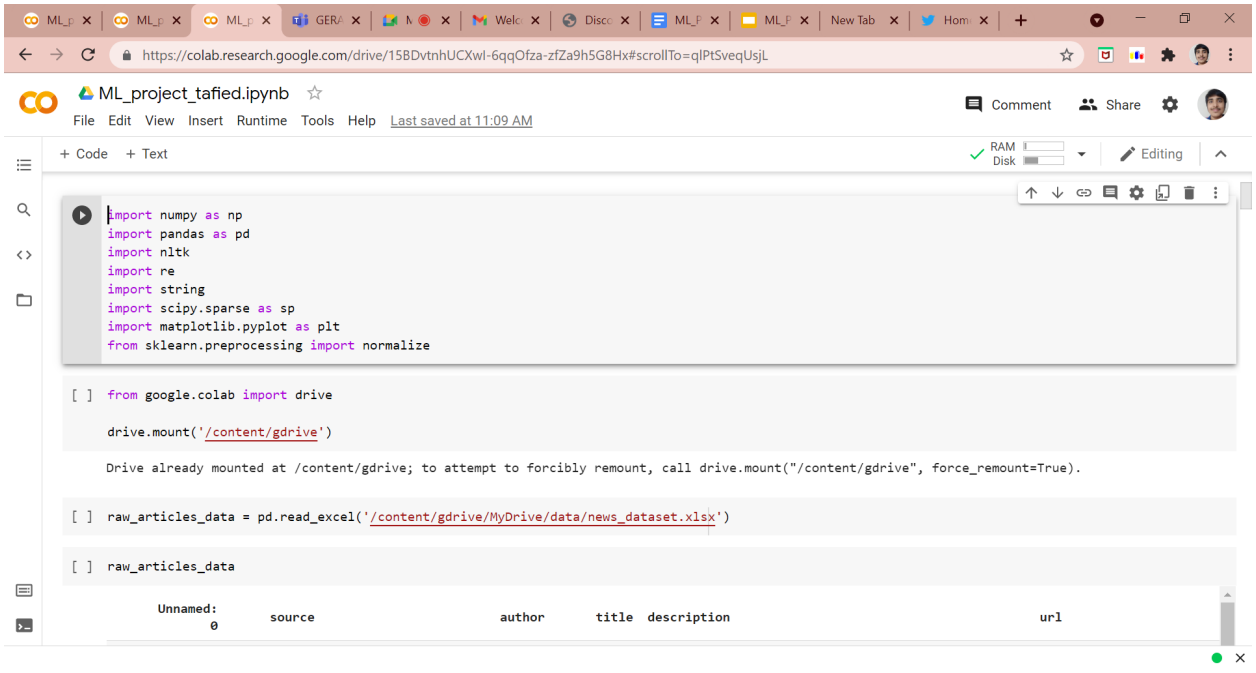

dataset_generation.ipynb
File Edit View Insert Runtime Tools Help Last saved at 16:30
Comment Share
+ Code + Text
RAM Disk
Editing
[ ] from newsapi import NewsApiClient
[ ] newsapi= NewsApiClient(api_key='956160556097434c9cf592566862ea13')
[ ] def get_past_articles(past=100):
past_articles=dict()
for past_days in range(1,past):
from_day=str(datetime.datetime.now()-datetime.timedelta(days=past_days))
to_day=str(datetime.datetime.now()-datetime.timedelta(days=past_days-1))
past_articles.update({from_day:to_day})
return past_articles
[ ] def get_articles(query, past=100):
past_articles=get_past_articles(past)
all_articles=[]
for i,j in tqdm(past_articles.items()):
for pag in tqdm(range(1,6)):
pag_articles=newsapi.get_everything(q=query,language='en', from_param='2021-03-27', to='2020-04-20', sort_by='relevancy', page=pag)['articles']
if len(pag_articles)==0:break
all_articles.extend(pag_articles)
return all_articles
[ ] articles= get_articles("India")
80%| | 4/5 [00:00<00:00, 9.29it/s]
100%| | 5/5 [00:00<00:00, 8.87it/s]
91%| | 90/99 [00:53<00:05, 1.74it/s]
0%| | 0/5 [00:00<?, ?it/s]
20%| | 1/5 [00:00<00:00, 8.73it/s]
40%| | 2/5 [00:00<00:00, 8.88it/s]
60%| | 3/5 [00:00<00:00, 8.66it/s]
meet.google.com is sharing your screen. Stop sharing Hide
Type here to search
ENG 19:10 27-04-2021

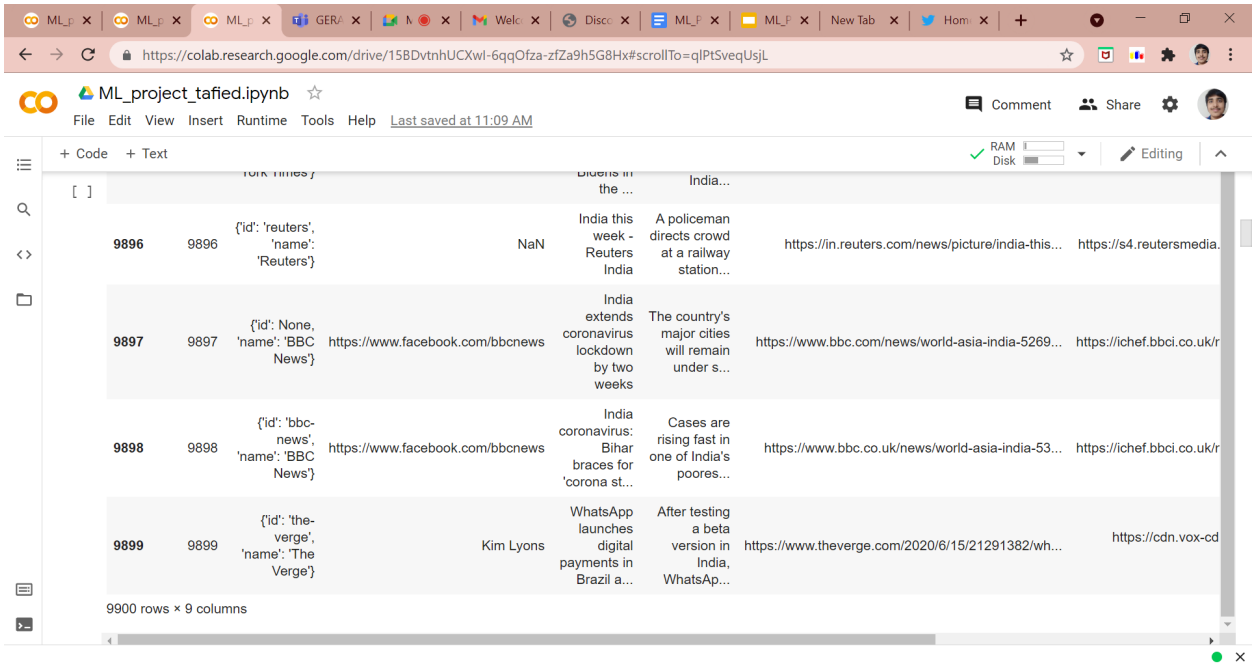

dataset_generation.ipynb
File Edit View Insert Runtime Tools Help Last saved at 16:30
Comment Share
+ Code + Text
RAM Disk
Editing
[ ] dataset = pd.DataFrame(articles)
dataset.sample(5)
source author title description url urlToImage publishedAt content
4617 {'id': 'techcrunch', 'name': 'TechCrunch'} Rita Liao Xiaomi further localizes India supply chain vi... China's Xiaomi had dominated the Indian smartp... http://techcrunch.com/2021/02/25/xiaomi-india-... https://techcrunch.com/wp-content/uploads/2019... 2021-02-25T13:15:33Z China's Xiaomi had dominated the Indian smartp...
337 {'id': 'techcrunch', 'name': 'TechCrunch'} Manish Singh Google removes millions of negative TikTok rev... ByteDance's TikTok app, which has gained hundr... http://techcrunch.com/2020/05/27/google-remove... https://techcrunch.com/wp-content/uploads/2020... 2020-05-27T07:42:47Z ByteDances TikTok app, which has gained hundre...
7317 {'id': 'techcrunch', 'name': 'TechCrunch'} Rita Liao Xiaomi further localizes India supply chain vi... China's Xiaomi had dominated the Indian smartp... http://techcrunch.com/2021/02/25/xiaomi-india-... https://techcrunch.com/wp-content/uploads/2019... 2021-02-25T13:15:33Z China's Xiaomi had dominated the Indian smartp...
6254 {'id': 'techcrunch', 'name': 'TechCrunch'} Manish Singh Uber picks new India and South Asia president Uber has named Prabhjeet Singh as the new pres... http://techcrunch.com/2020/07/15/uber-picks-ne... https://techcrunch.com/wp-content/uploads/2020... 2020-07-16T06:21:11Z Uber has named Prabhjeet Singh as the new pres...
9526 {'id': 'techcrunch', 'name': 'TechCrunch'} Manish Singh India cabinet approves setting up a 'massive n... More than one billion people in India today ha... http://techcrunch.com/2020/12/10/india-cabinet... https://techcrunch.com/wp-content/uploads/2020... 2020-12-10T13:06:16Z More than one billion people in India today ha...
[ ] dataset.to_excel('news_dataset.xlsx', sheet_name = 'New_sheet')
from google.colab import files
files.download('news_dataset.xlsx')
meet.google.com is sharing your screen. Stop sharing Hide
Type here to search
ENG 19:11 27-04-2021

Output Dataset



5.2. Python TAFIED implementation





ML_project_tafied.ipynb

File Edit View Insert Runtime Tools Help Last saved at 11:09 AM

Comment Share

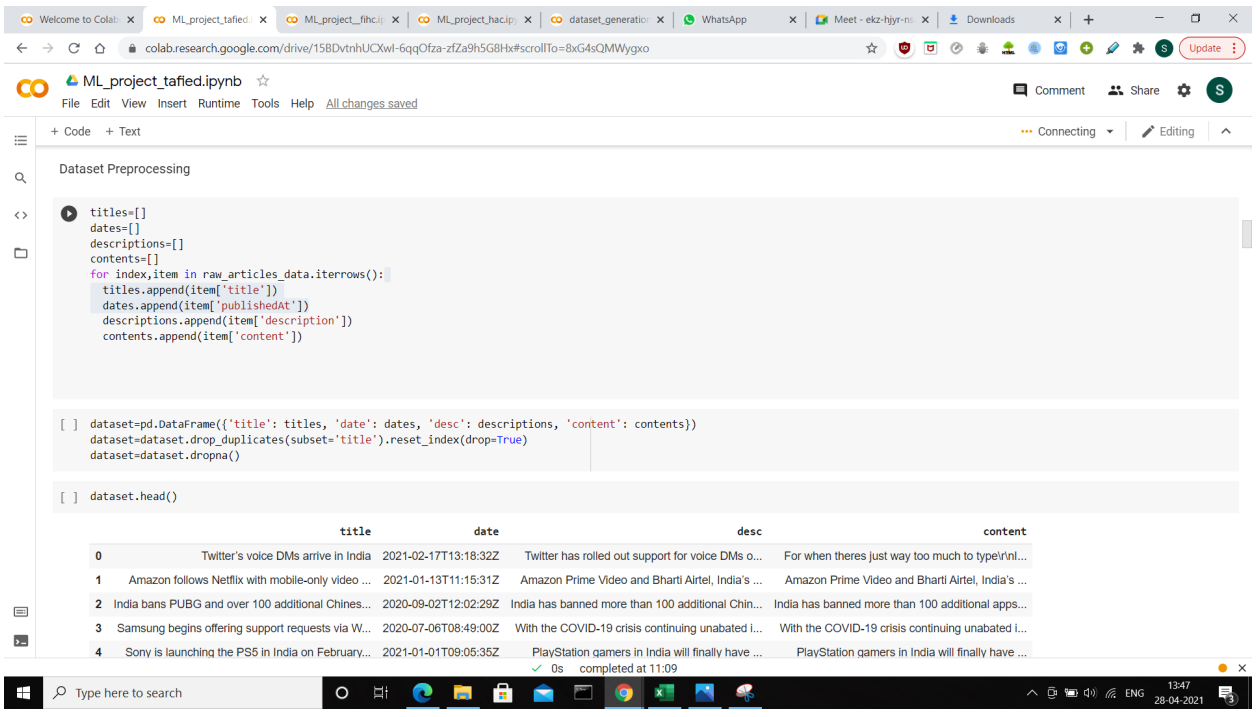
+ Code + Text

RAM Disk Editing

[ ]

				India...			
9896	9896	{'id': 'reuters', 'name': 'Reuters'}	NaN	India this week - Reuters India	A policeman directs crowd at a railway station...	https://in.reuters.com/news/picture/india-this...	https://s4.reutersmedia.
9897	9897	{'id': None, 'name': 'BBC News'}	https://www.facebook.com/bbcnews	India extends coronavirus lockdown by two weeks	The country's major cities will remain under s...	https://www.bbc.com/news/world-asia-india-5269...	https://ichef.bbci.co.uk/r
9898	9898	{'id': 'bbc-news', 'name': 'BBC News'}	https://www.facebook.com/bbcnews	India coronavirus: Bihar braces for 'corona st...	Cases are rising fast in one of India's poores...	https://www.bbc.co.uk/news/world-asia-india-53...	https://ichef.bbci.co.uk/r
9899	9899	{'id': 'the-verge', 'name': 'The Verge'}	Kim Lyons	WhatsApp launches digital payments in Brazil a...	After testing a beta version in India, WhatsAp...	https://www.theverge.com/2020/6/15/21291382/wh...	https://cdn.vox-cd

9900 rows × 9 columns



ML_project_tafied.ipynb

File Edit View Insert Runtime Tools Help All changes saved

Comment Share

+ Code + Text

Connecting Editing

Dataset Preprocessing

```
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'])
```

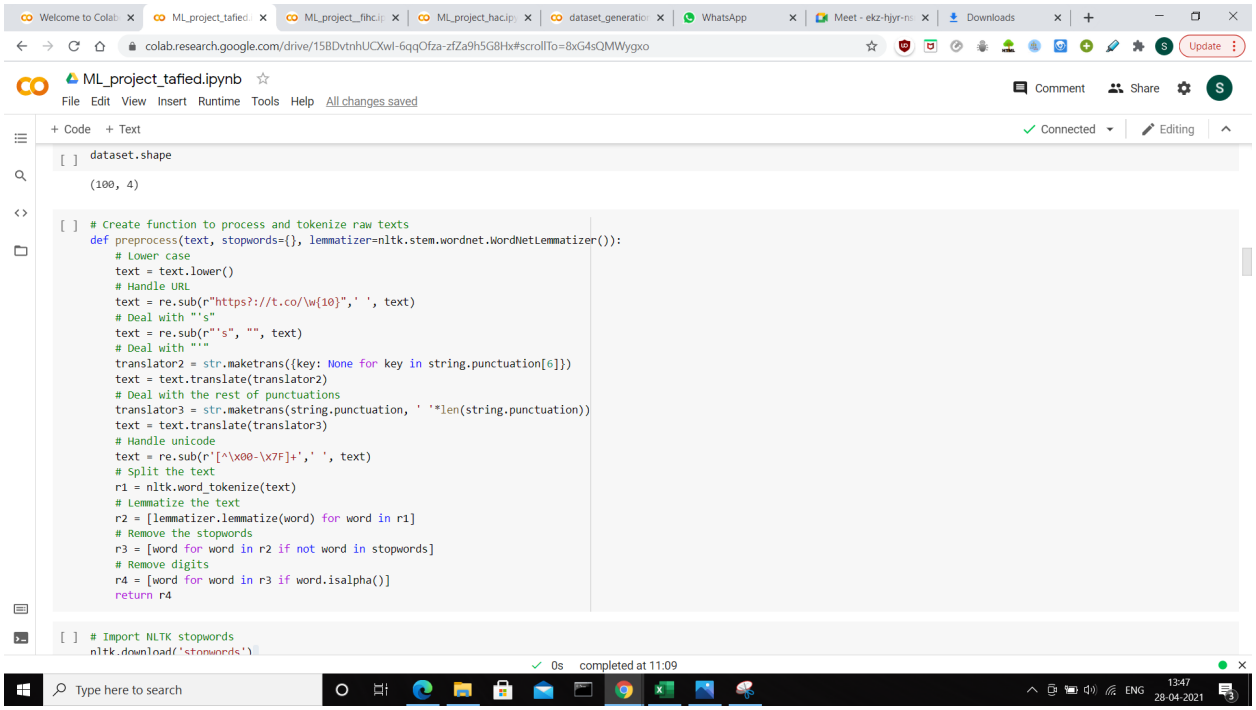
[ ]
```
dataset=pd.DataFrame({'title': titles, 'date': dates, 'desc': descriptions, 'content': contents})
dataset=dataset.drop_duplicates(subset='title').reset_index(drop=True)
dataset=dataset.dropna()
```

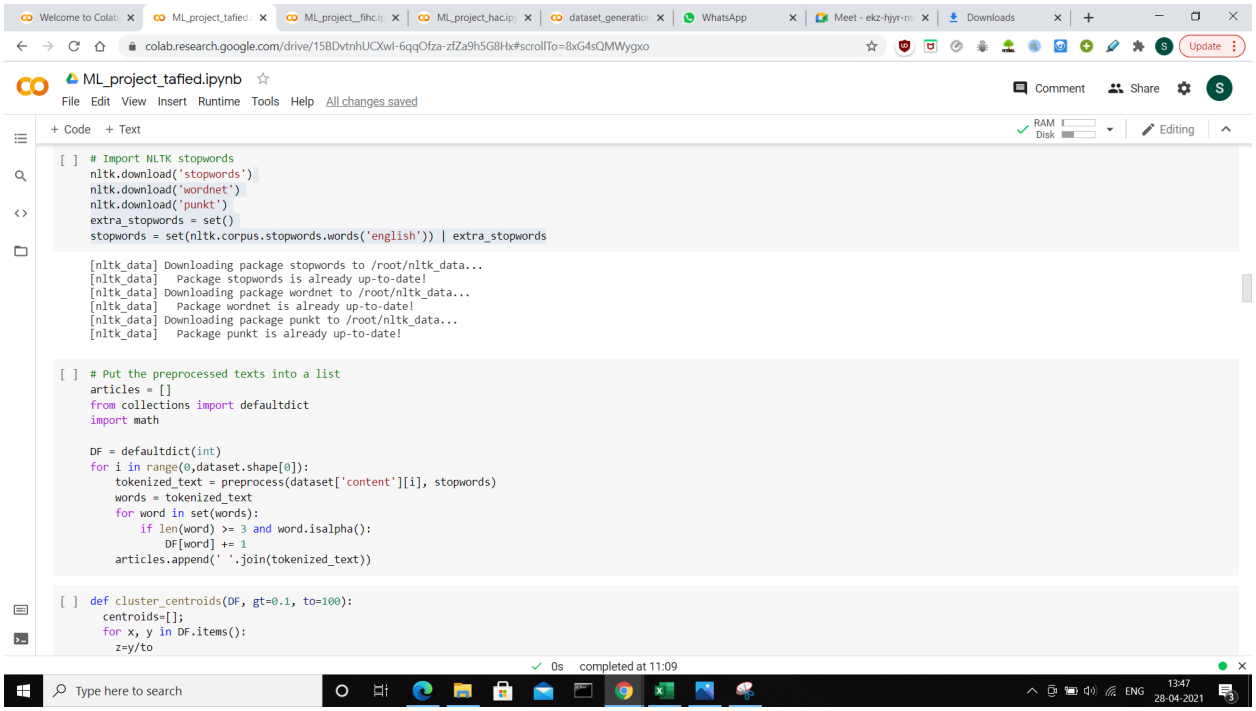
[ ]
```
dataset.head()
```

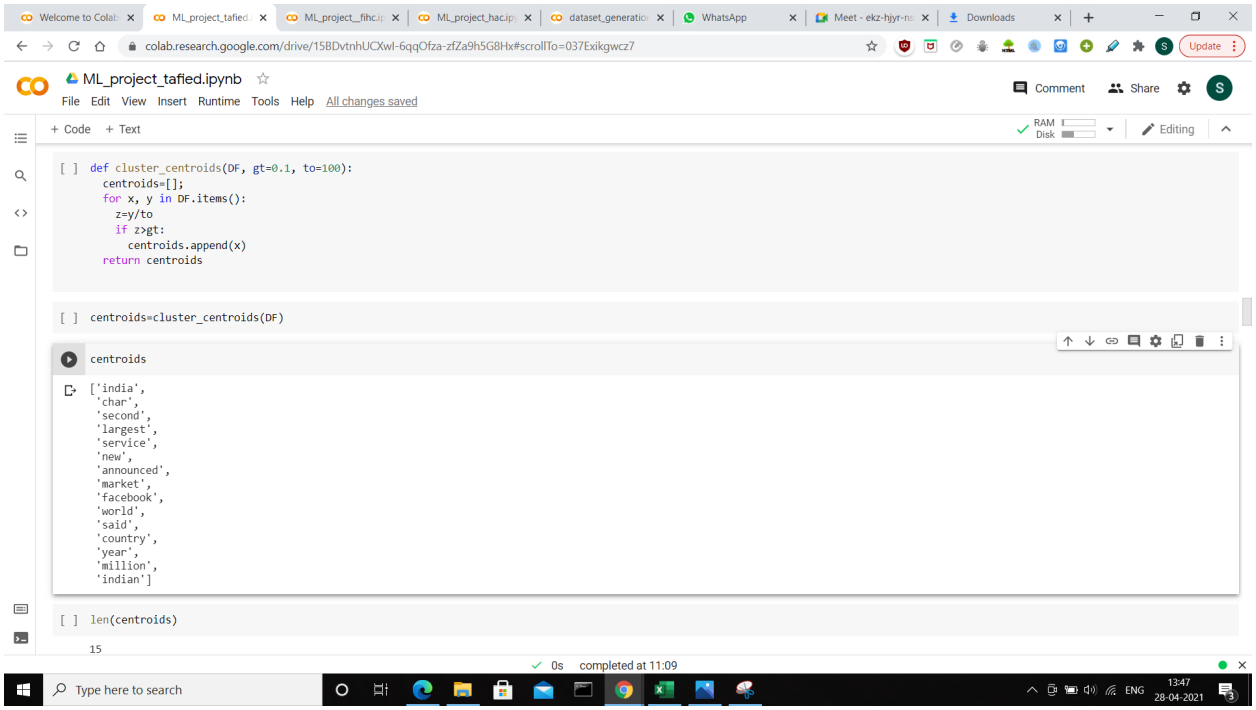
	title	date	desc	content
0	Twitter's voice DMs arrive in India	2021-02-17T13:18:32Z	Twitter has rolled out support for voice DMs o...	For when theres just way too much to type\r\n\...
1	Amazon follows Netflix with mobile-only video ...	2021-01-13T11:15:31Z	Amazon Prime Video and Bharti Airtel, India's ...	Amazon Prime Video and Bharti Airtel, India's ...
2	India bans PUBG and over 100 additional Chines...	2020-09-02T12:02:29Z	India has banned more than 100 additional Chin...	India has banned more than 100 additional apps...
3	Samsung begins offering support requests via W...	2020-07-06T08:49:00Z	With the COVID-19 crisis continuing unabated i...	With the COVID-19 crisis continuing unabated i...
4	Sony is launching the PS5 in India on February...	2021-01-01T09:05:35Z	PlayStation gamers in India will finally have ...	PlayStation gamers in India will finally have ...

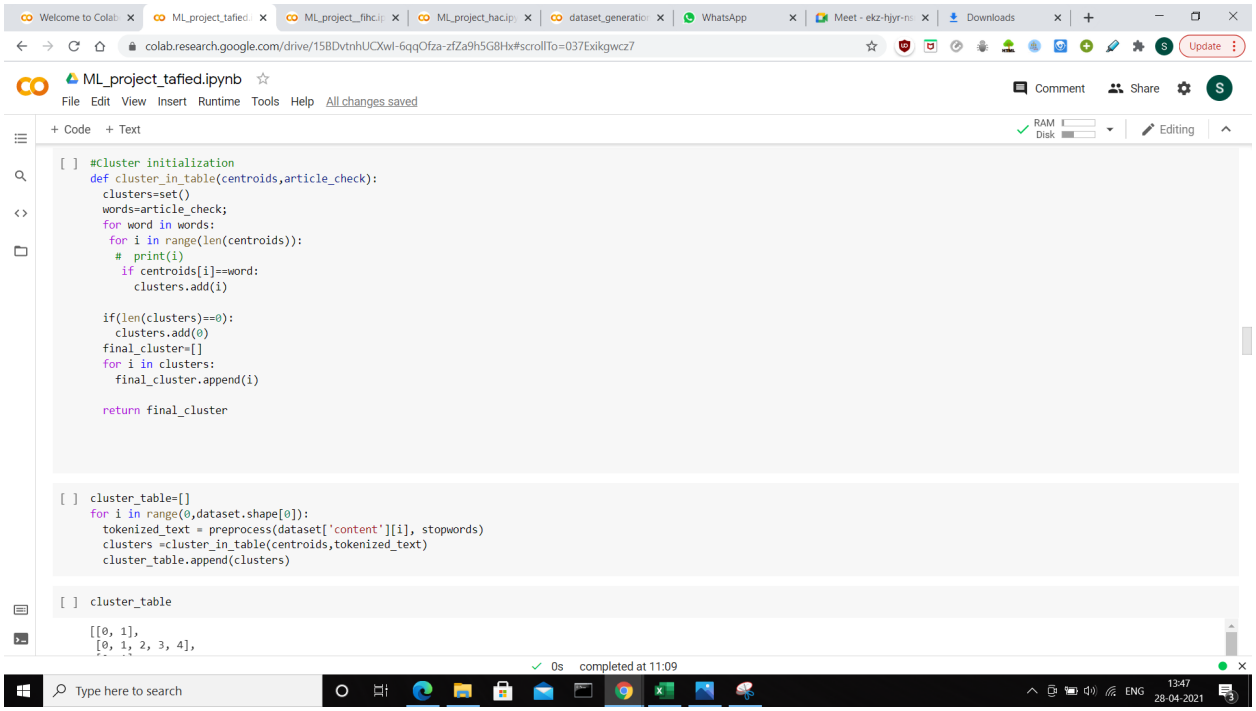
0s completed at 11:09

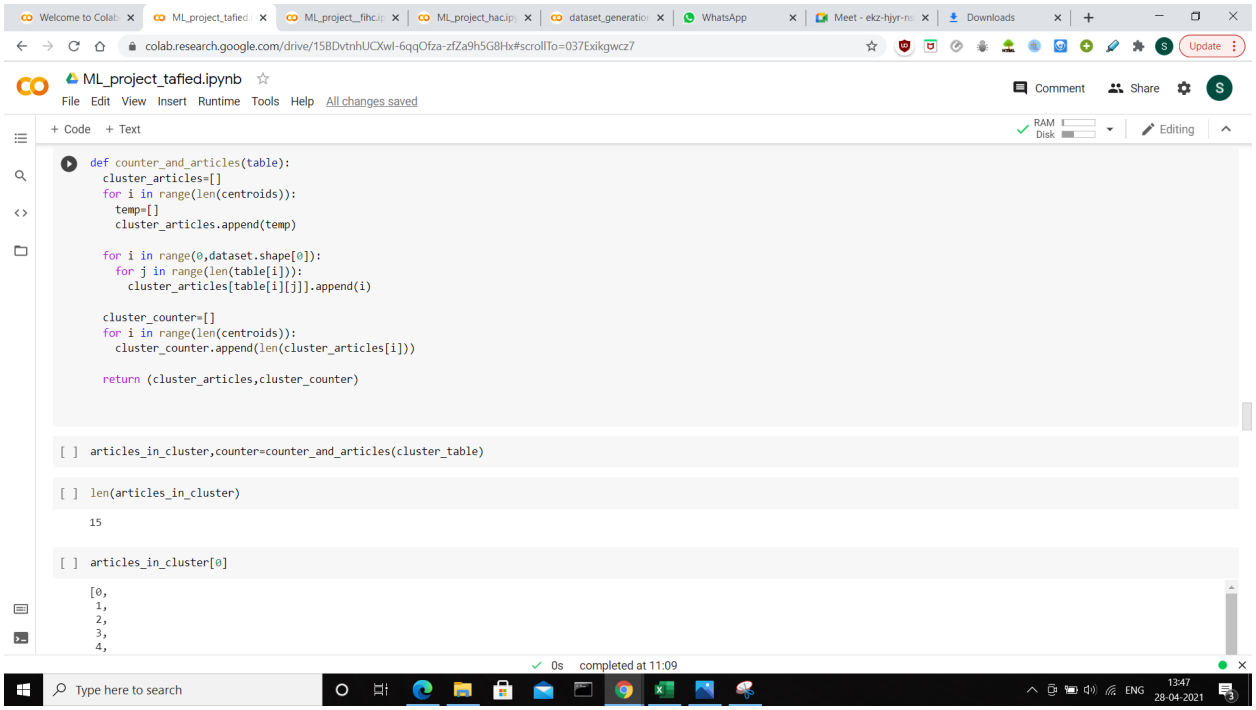
Type here to search ENG 13:47 28-04-2021


Welcome to Colab
ML_project_tafied.ipynb
ML_project_fihc.ipynb
ML_project_hac.ipynb
dataset_generation
WhatsApp
Meet - ekz-hjyr-ns
Downloads
colab.research.google.com/drive/15BDvtnhUCXwI-6qqOfza-zfZa9h5G8Hx#scrollTo=8xG4sQMWygxo
Update
ML_project_tafied.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
Connected
Editing
[ ] 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')
0s completed at 11:09
Type here to search
ENG
13:47
28-04-2021


Welcome to Colab
ML_project_tafied.ipynb
ML_project_fihc.ipynb
ML_project_hac.ipynb
dataset_generation
WhatsApp
Meet - ekz-hjyr-ns
Downloads
colab.research.google.com/drive/15BDvtnhUCXwI-6qqOfza-zfZa9h5G8Hx#scrollTo=8xG4sQMWygxo
Update
ML_project_tafied.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment
Share
+ Code + Text
RAM
Disk
Editing
[ ] # 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
0s completed at 11:09
Type here to search
ENG
13:47
28-04-2021


ML_project_tafied.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
[ ] 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',
'second',
'largest',
'service',
'new',
'announced',
'market',
'facebook',
'world',
'said',
'country',
'year',
'million',
'indian']
[ ] len(centroids)
15
0s completed at 11:09
Type here to search
ENG
13:47
28-04-2021


ML_project_tafied.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
[ ] #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)
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
[[0, 1],
[0, 1, 2, 3, 4],
0s completed at 11:09
Type here to search
ENG
13:47
28-04-2021



```
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)
```

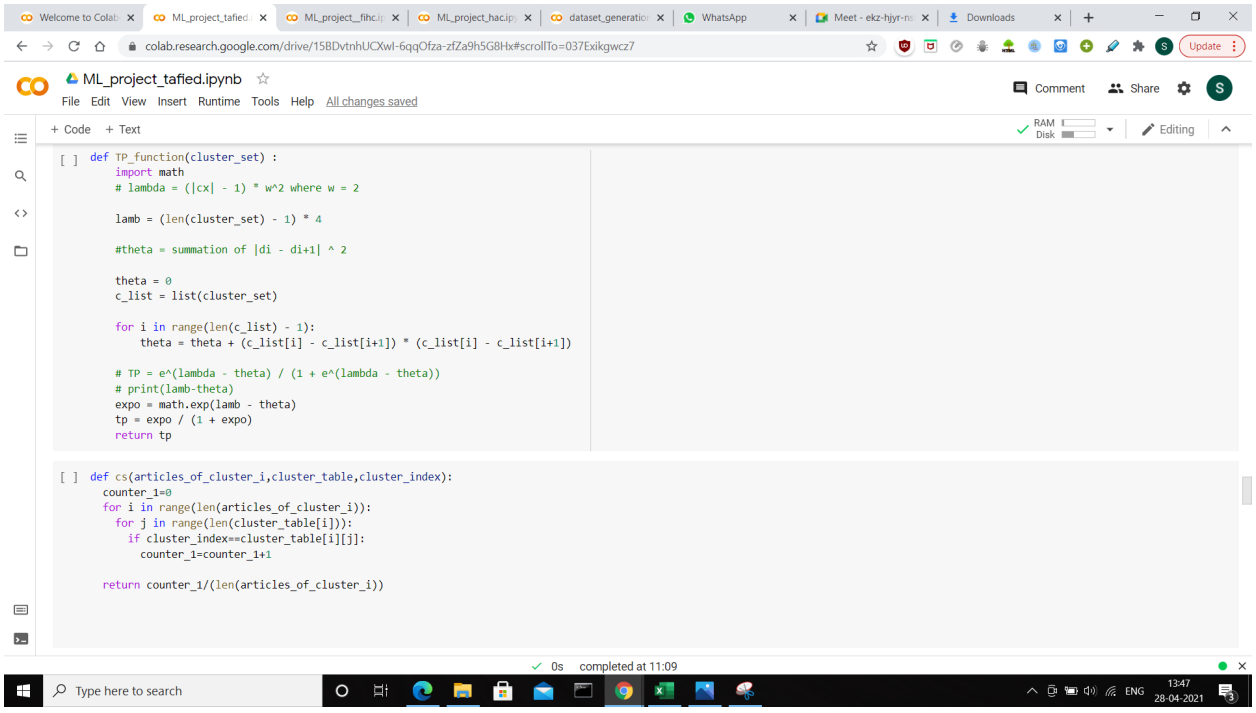
```
articles_in_cluster,counter=counter_and_articles(cluster_table)
```

```
len(articles_in_cluster)
```

```
15
```

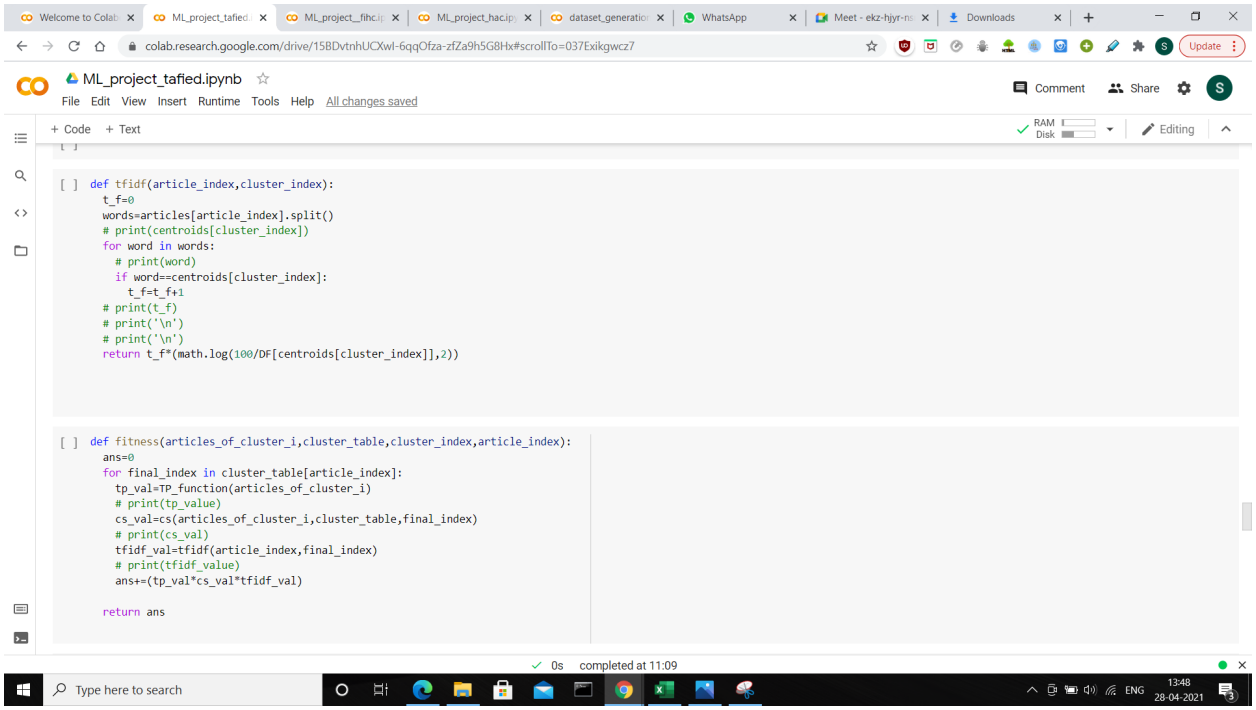
```
articles_in_cluster[0]
```

```
[0,
 1,
 2,
 3,
 4,
```



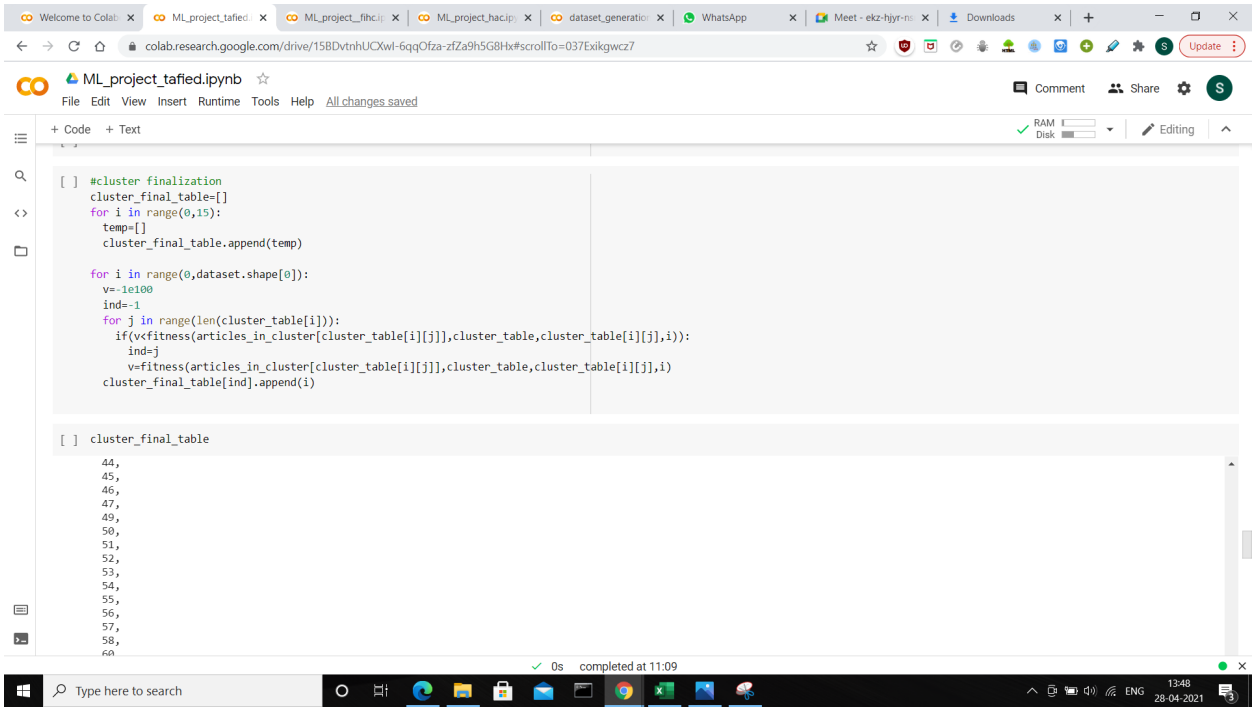
```
def TP_function(cluster_set) :
    import math
    # lambda = (|cx| - 1) * w^2 where w = 2

    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))
```

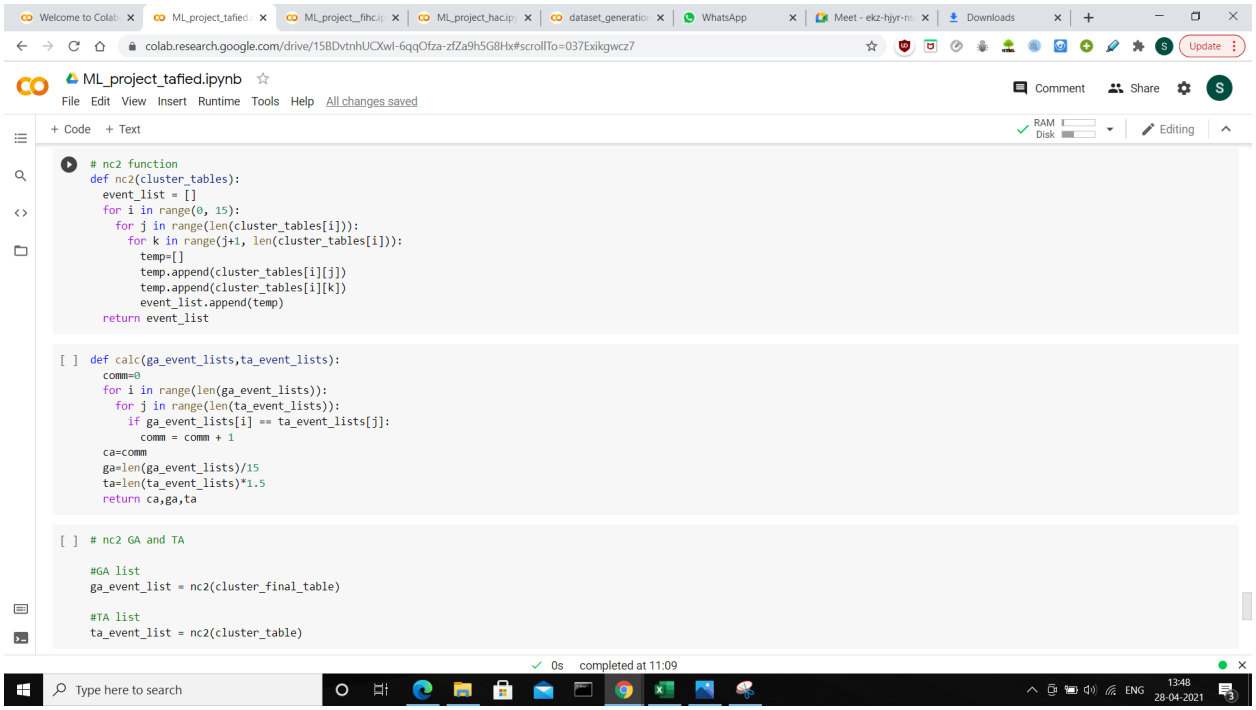
```
[ ] def fitness(articles_of_cluster_i,cluster_table,cluster_index,article_index):
      ans=0
      for final_index in cluster_table[article_index]:
        tp_val=TP_function(articles_of_cluster_i)
        # print(tp_value)
        cs_val=cs(articles_of_cluster_i,cluster_table,final_index)
        # print(cs_val)
        tfidf_val=tfidf(article_index,final_index)
        # print(tfidf_value)
        ans+=(tp_val*cs_val*tfidf_val)

      return ans
```

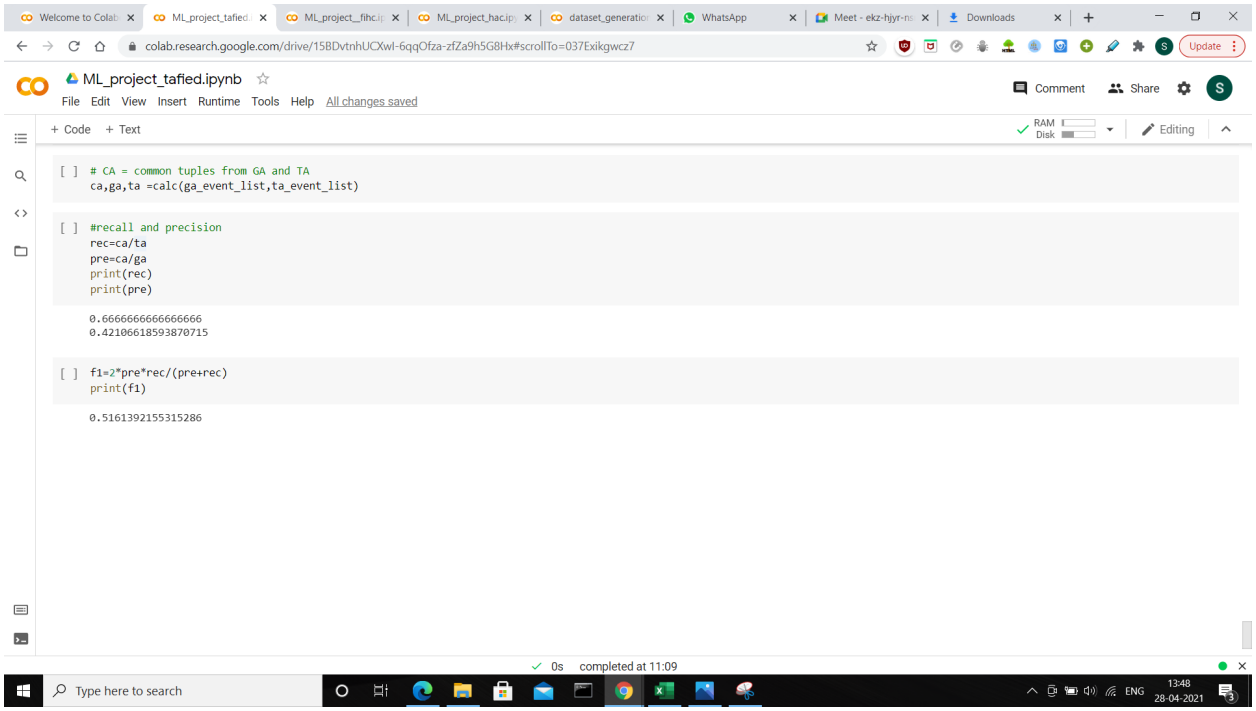


```
[ ] #cluster finalization
    cluster_final_table=[]
    for i in range(0,15):
      temp=[]
      cluster_final_table.append(temp)

    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],i)):
          ind=j
          v=fitness(articles_in_cluster[cluster_table[i][j]],cluster_table,cluster_table[i][j],i)
      cluster_final_table[ind].append(i)
```

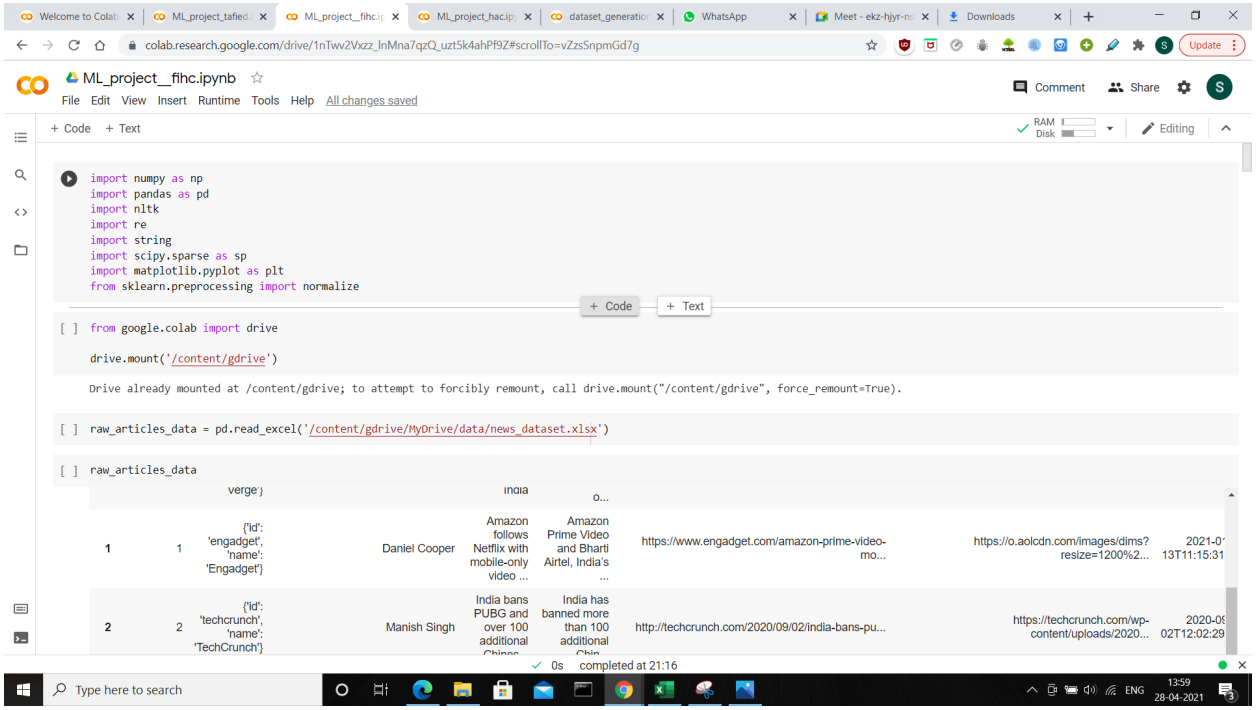
```
[ ] cluster_final_table
```

```
    44,
    45,
    46,
    47,
    49,
    50,
    51,
    52,
    53,
    54,
    55,
    56,
    57,
    58,
```


ML_project_tafied.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
# 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)
0s completed at 11:09
Type here to search
ENG 13:48 28-04-2021


ML_project_tafied.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
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)
0.6666666666666666
0.42106618593870715
f1=2*pre*rec/(pre+rec)
print(f1)
0.5161392155315286
0s completed at 11:09
Type here to search
ENG 13:48 28-04-2021

5.3. Python FIHC implementation



```
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
```

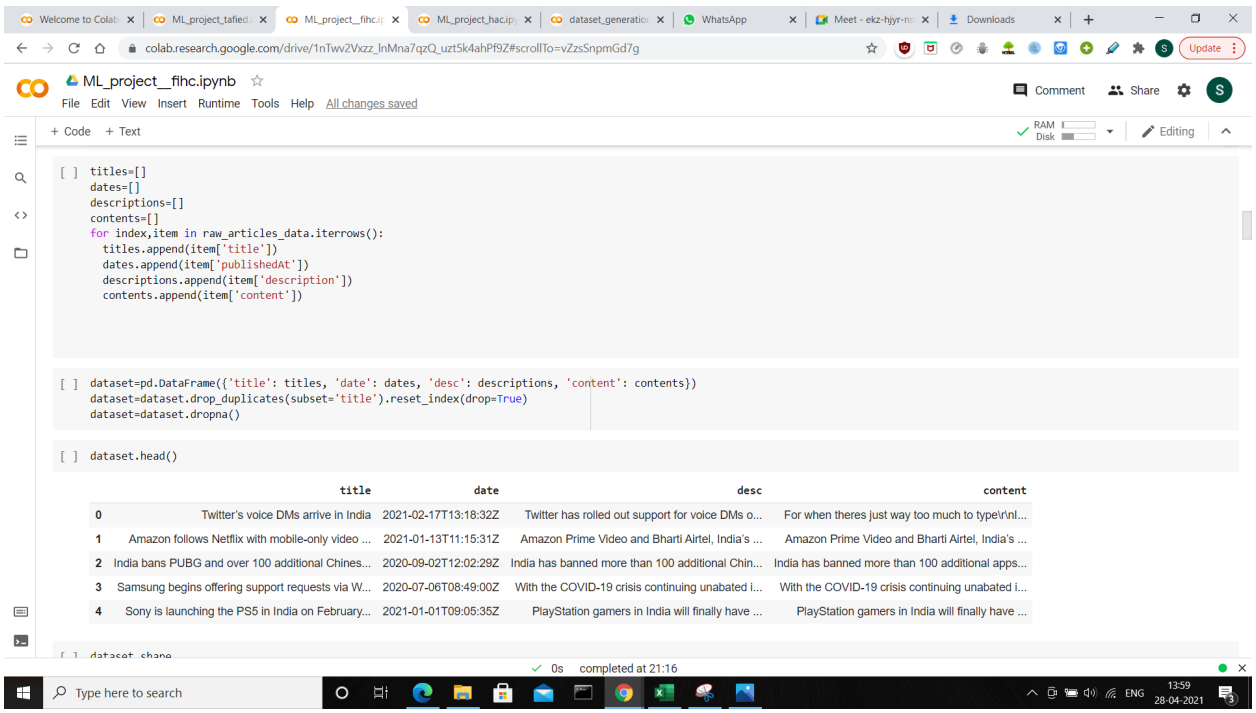
```
from google.colab import drive

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

```
Drive already mounted at /content/gdrive; to attempt to forcibly remount, call drive.mount("/content/gdrive", force_remount=True).
```

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

```
raw_articles_data
```

		verge}		India	o...		
1	1	{'id': 'engadget', 'name': 'Engadget'}	Daniel Cooper	Amazon follows Netflix with mobile-only video ...	Amazon Prime Video and Bharti Airtel, India's ...	https://www.engadget.com/amazon-prime-video-mo...	https://o.aolcdn.com/images/dims?resize=1200%2...	2021-01-13T11:15:31
2	2	{'id': 'techcrunch', 'name': 'TechCrunch'}	Manish Singh	India bans PUBG and over 100 additional Chines...	India has banned more than 100 additional Chin...	http://techcrunch.com/2020/09/02/india-bans-pu...	https://techcrunch.com/wp-content/uploads/2020...	2020-09-02T12:02:29



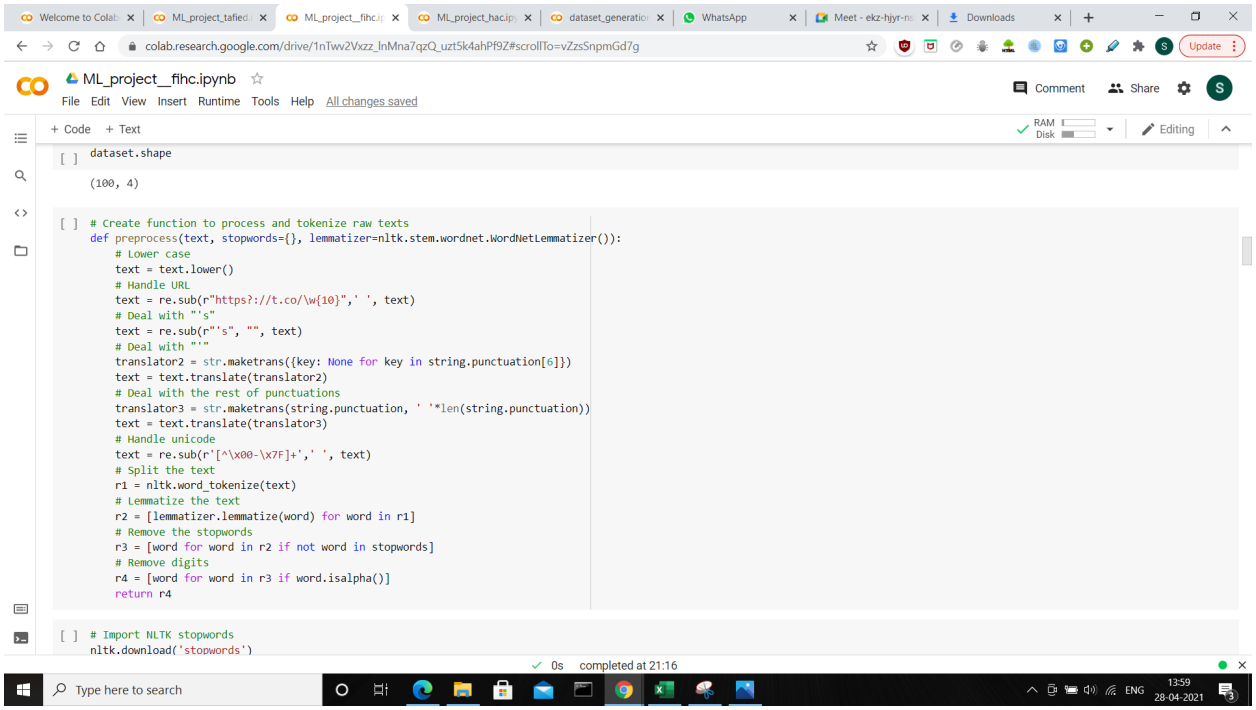
```
titles=[]
dates=[]
descriptions=[]
contents=[]
for index,item in raw_articles_data.iterrows():
  titles.append(item['title'])
  dates.append(item['publishedAt'])
  descriptions.append(item['description'])
  contents.append(item['content'])
```

```
dataset=pd.DataFrame({'title': titles, 'date': dates, 'desc': descriptions, 'content': contents})
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 o...	For when theres just way too much to type\r\nI...
1	Amazon follows Netflix with mobile-only video ...	2021-01-13T11:15:31Z	Amazon Prime Video and Bharti Airtel, India's ...	Amazon Prime Video and Bharti Airtel, India's ...
2	India bans PUBG and over 100 additional Chines...	2020-09-02T12:02:29Z	India has banned more than 100 additional Chin...	India has banned more than 100 additional apps...
3	Samsung begins offering support requests via W...	2020-07-06T08:49:00Z	With the COVID-19 crisis continuing unabated i...	With the COVID-19 crisis continuing unabated i...
4	Sony is launching the PS5 in India on February...	2021-01-01T09:05:35Z	PlayStation gamers in India will finally have ...	PlayStation gamers in India will finally have ...

```
dataset.shape
```

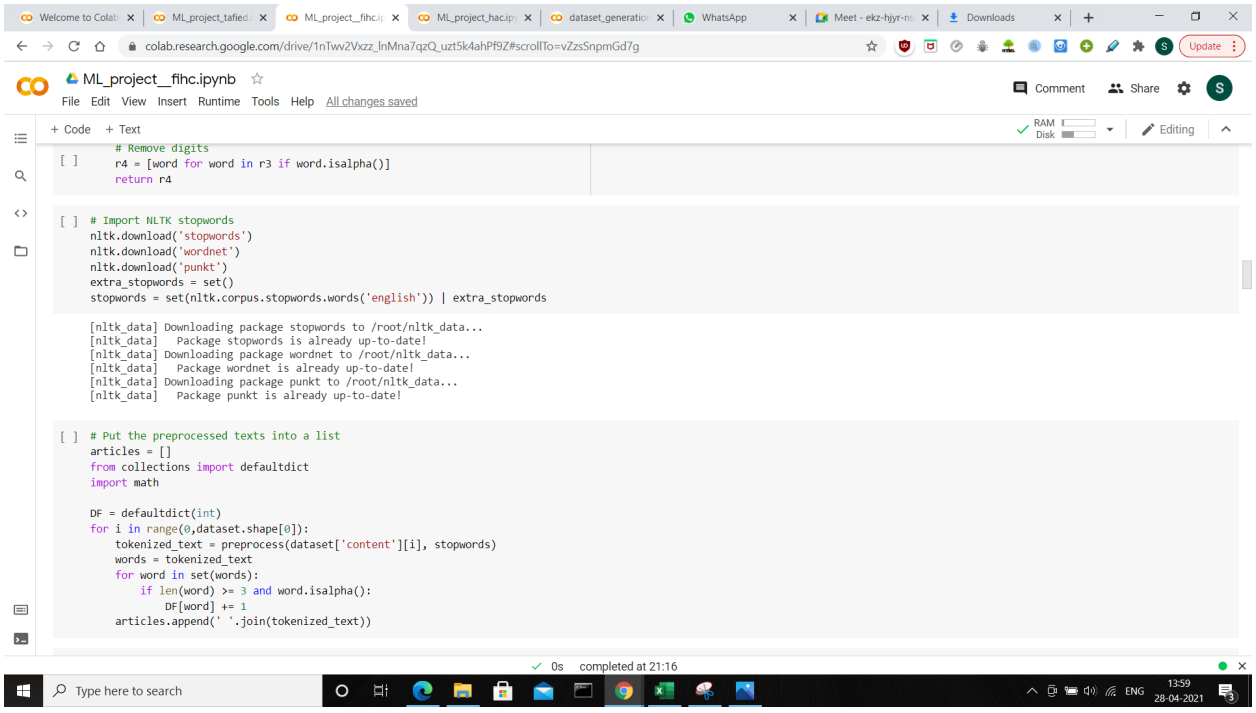


```
dataset.shape
```

```
(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')
```

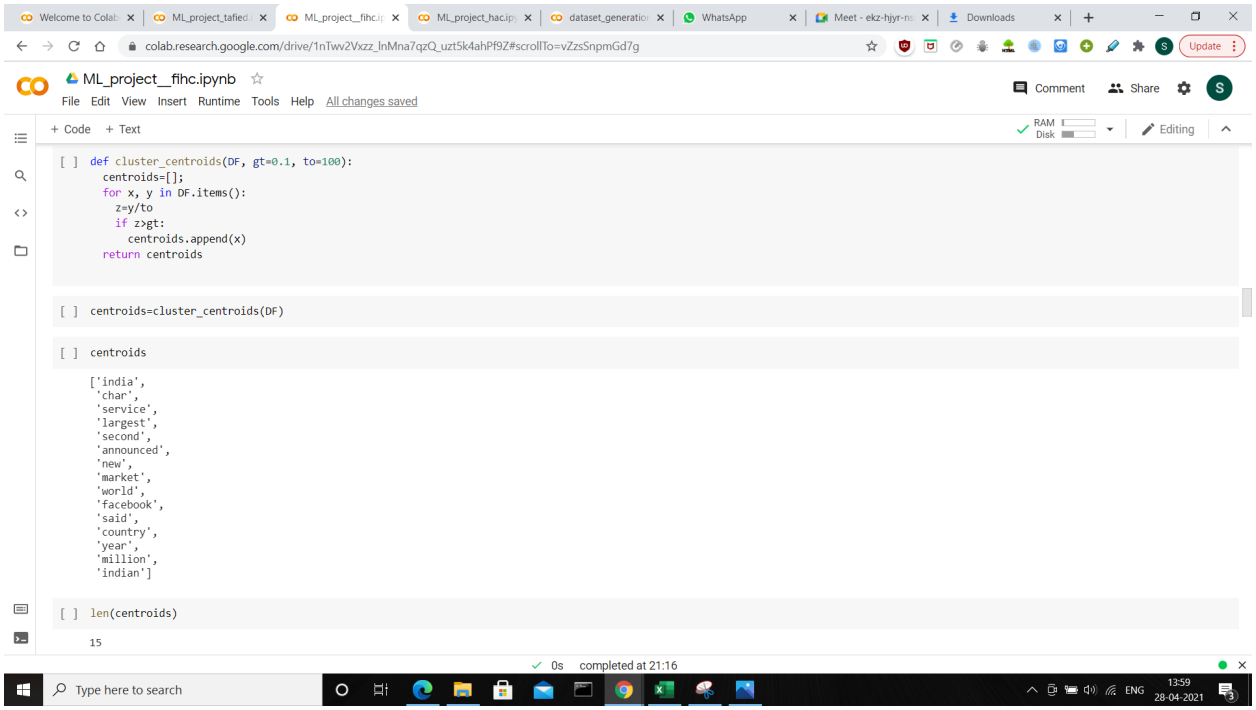


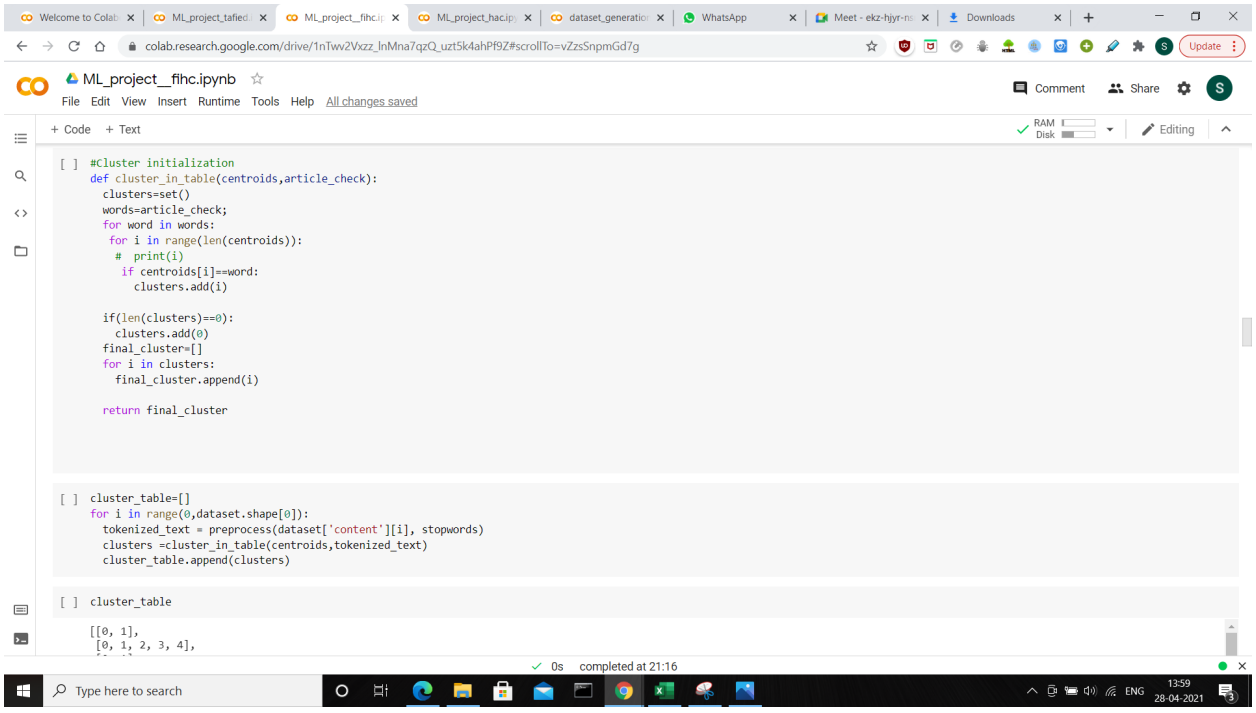
```
    # Remove digits
    r4 = [word for word in r3 if word.isalpha()]
    return r4
```

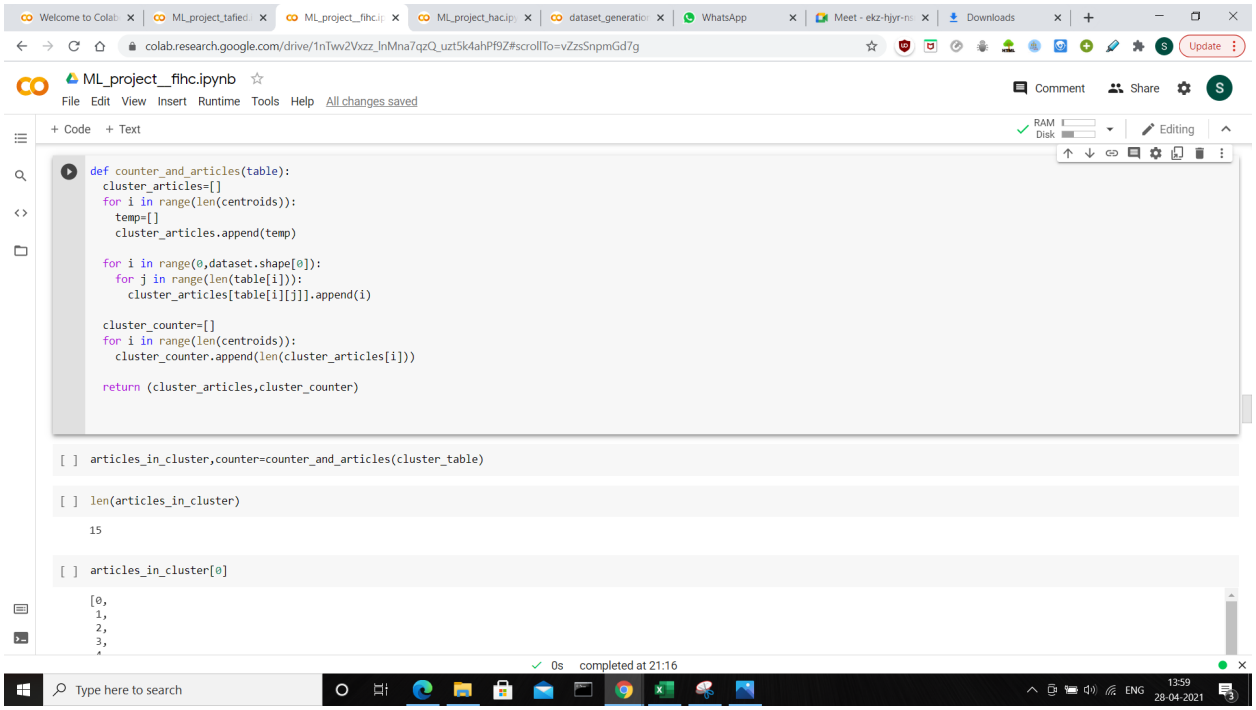
```
# Import NLTK stopwords
nltk.download('stopwords')
nltk.download('wordnet')
nltk.download('punkt')
extra_stopwords = set()
stopwords = set(nltk.corpus.stopwords.words('english')) | extra_stopwords
```

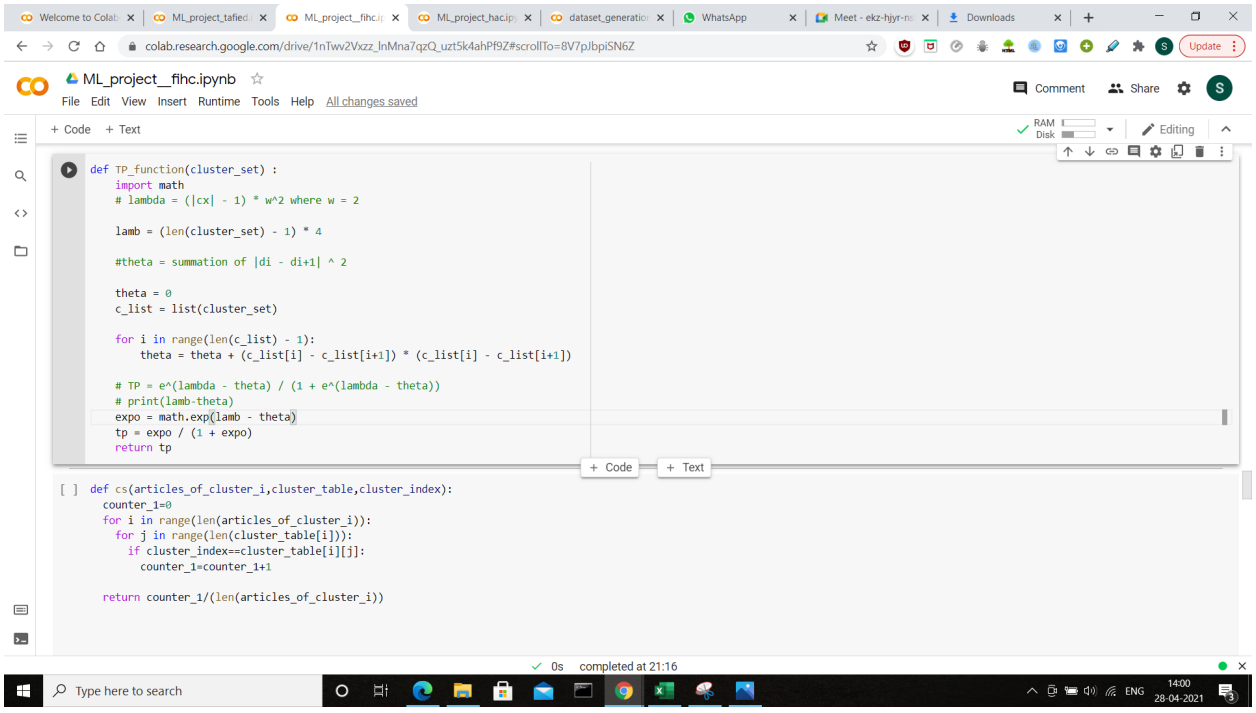
```
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Package stopwords is already up-to-date!
[nltk_data] Downloading package wordnet to /root/nltk_data...
[nltk_data]   Package wordnet is already up-to-date!
[nltk_data] Downloading package punkt to /root/nltk_data...
[nltk_data]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

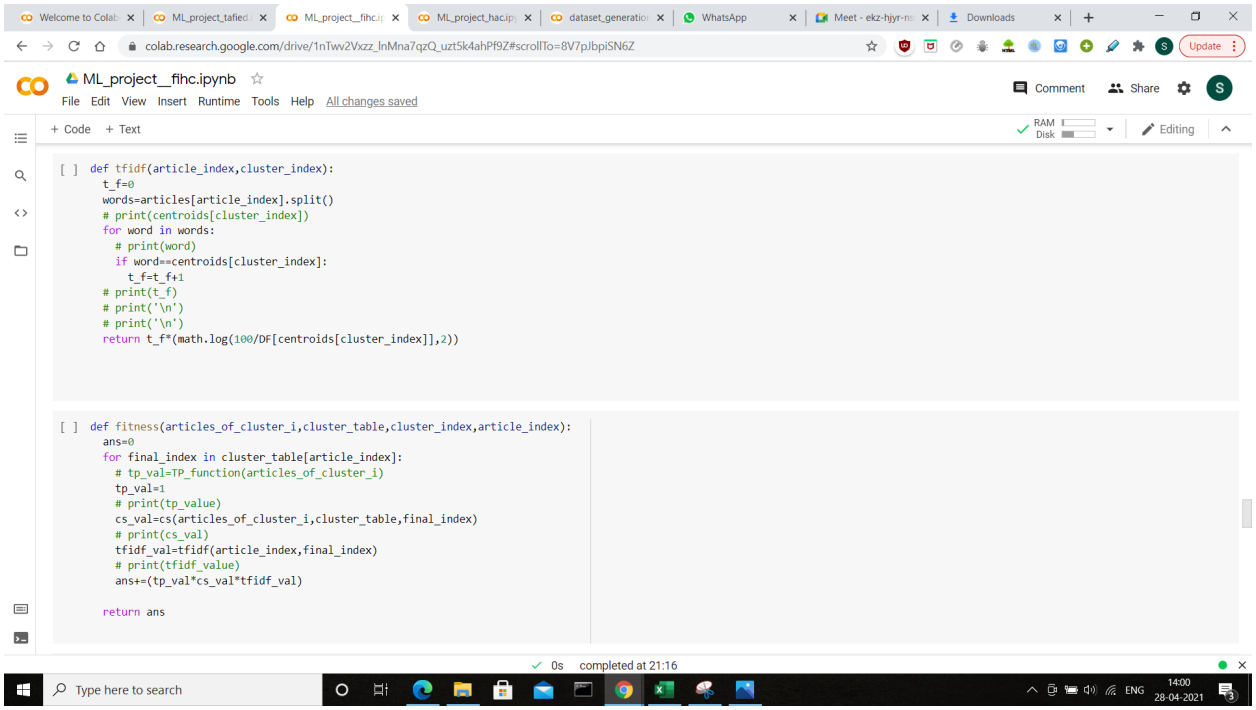
```
# Put the preprocessed texts into a list
articles = []
from collections import defaultdict
import math

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))
```


ML_project__fihc.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
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
0s completed at 21:16
Type here to search
ENG
13:59
28-04-2021


ML_project__fihc.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
#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)
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
[[0, 1],
[0, 1, 2, 3, 4],
0s completed at 21:16
Type here to search
ENG
13:59
28-04-2021

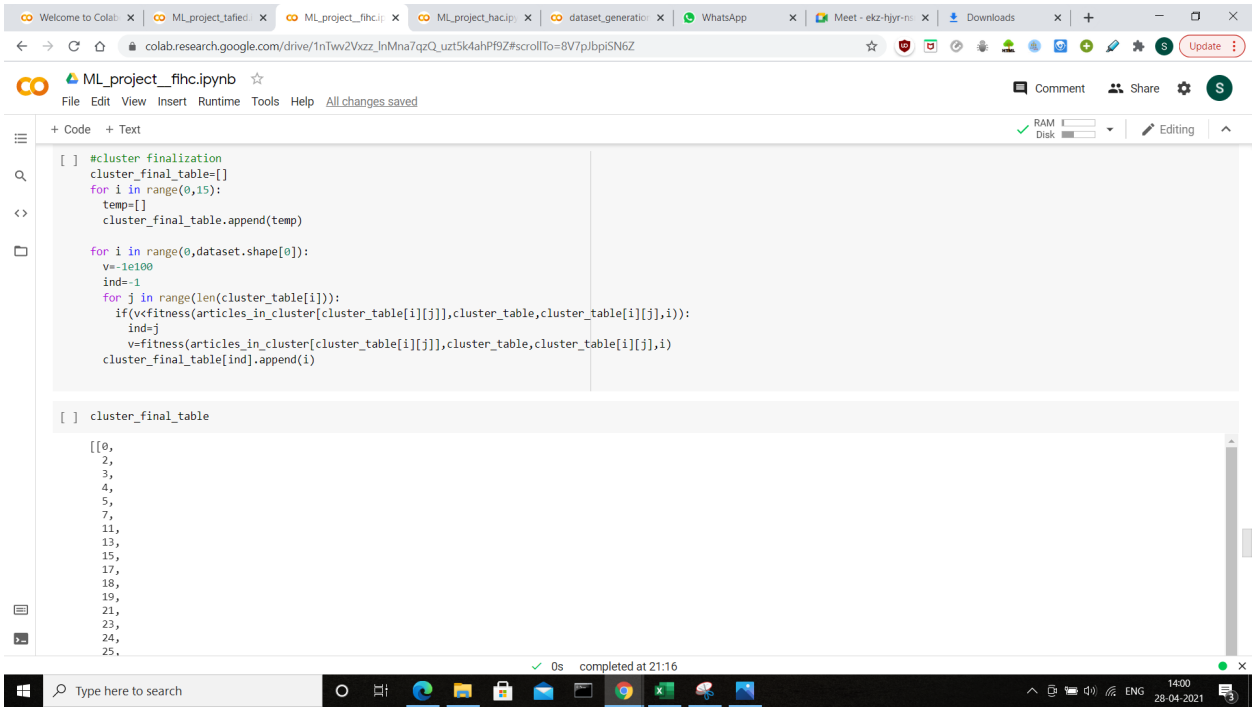

ML_project__fihc.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Editing
def counter_and_articles(table):
cluster_articles=[]
for i in range(len(centroids)):
temp=[]
cluster_articles.append(temp)
for i in range(0,dataset.shape[0]):
for j in range(len(table[i])):
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]
[0,
1,
2,
3,
0s completed at 21:16
Type here to search
ENG
13:59
28-04-2021


ML_project__fihc.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
Editing
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
+ Code + Text
[ ] 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))
0s completed at 21:16
Type here to search
ENG
14:00
28-04-2021



```
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))
```

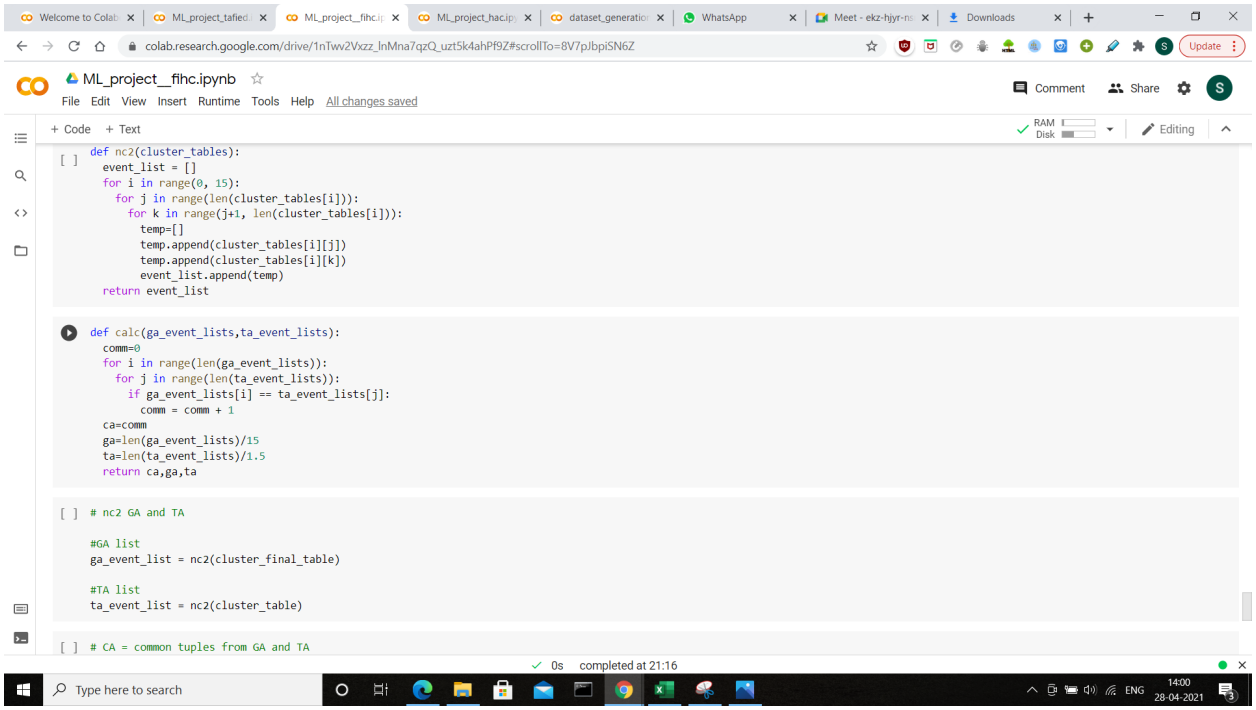
```
def fitness(articles_of_cluster_i,cluster_table,cluster_index,article_index):
    ans=0
    for final_index in cluster_table[article_index]:
      # tp_val=TP_function(articles_of_cluster_i)
      tp_val=1
      # print(tp_value)
      cs_val=cs(articles_of_cluster_i,cluster_table,final_index)
      # print(cs_val)
      tfidf_val=tfidf(article_index,final_index)
      # print(tfidf_value)
      ans+=(tp_val*cs_val*tfidf_val)

    return ans
```

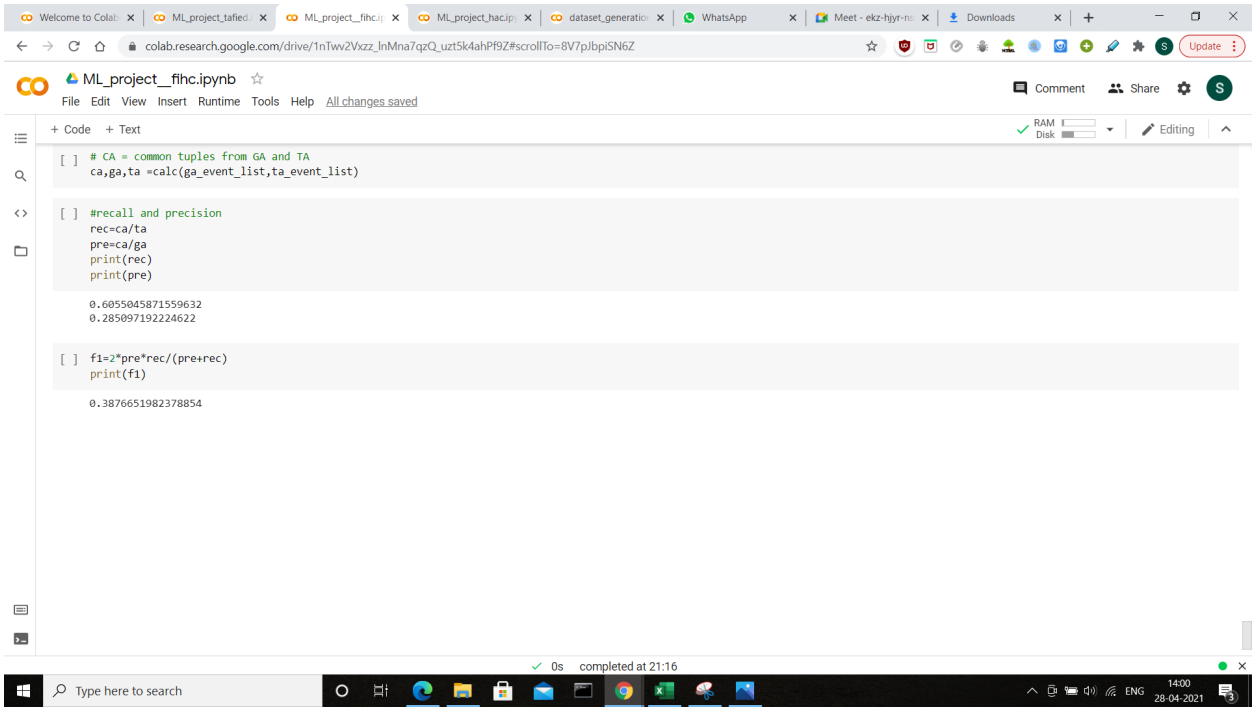



```
#cluster finalization
cluster_final_table=[]
for i in range(0,15):
  temp=[]
  cluster_final_table.append(temp)

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],i)):
      ind=j
      v=fitness(articles_in_cluster[cluster_table[i][j]],cluster_table,cluster_table[i][j],i)
  cluster_final_table[ind].append(i)
```

```
cluster_final_table
```

```
[[0,
  2,
  3,
  4,
  5,
  7,
  11,
  13,
  15,
  17,
  18,
  19,
  21,
  23,
  24,
  25,
```


ML_project__fihc.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
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
0s completed at 21:16
Type here to search
ENG 14:00 28-04-2021


ML_project__fihc.ipynb
File Edit View Insert Runtime Tools Help All changes saved
Comment Share
+ Code + Text
RAM Disk
Editing
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
Type here to search
ENG 14:00 28-04-2021

6. Result Comparison

	Cluster Recall	Cluster Precision	F-measure
TAFIED	0.667	0.421	0.516
FIHC	0.606	0.285	0.388

7. Future Scope

Using API's, web scraping the TAFIED algorithm can be run in real time making the process of maintaining the episodes of an event less cumbersome. The fitness function used in cluster distinction step has 4 more sub functions which are mathematically complex and take a lot of computational time. Thus this method may fail if the dataset is too large. A common efficient implementation of the algorithm which can overcome its current drawbacks.

8. Improvements Done

Due to the unavailability of the dataset given in the paper we created our own dataset using an Api. This enabled the dataset to become much more tailor made for our requirement as we could create it based on the events we specified. All the functions used in the TAFIED method were implemented in python without using libraries. HAC implementation for the same dataset has also been done.

9. References

1. **Yen-Hsien Leea, Paul Jen-Hwa Hub , Hongquan Zhuc , Hsin-Wei Chend ,"Discovering event episodes from sequences of online news articles: A time adjoining frequent itemset-based clustering method ", 3 May 2019**
2. **Ramesh Nallapati, Ao Feng, Fuchun Peng, James Allan, "Event Threading within News Topics", 8 November 2004.**

3. Scikit Documentation
4. NewsApi documentation