Updated Problem Formulation

During a calamity, Twitter is flooded with millions of tweets, many of which may pertain to the same event. To determine the precise location and timing of the incident, we intend to utilize the tweets and perform statistical analysis on the tweets. We can gather crucial information on earthquakes by continuously monitoring Twitter in real-time which in turn will facilitate in assisting individuals trapped in a natural calamity.

The gaps identified in the existing system

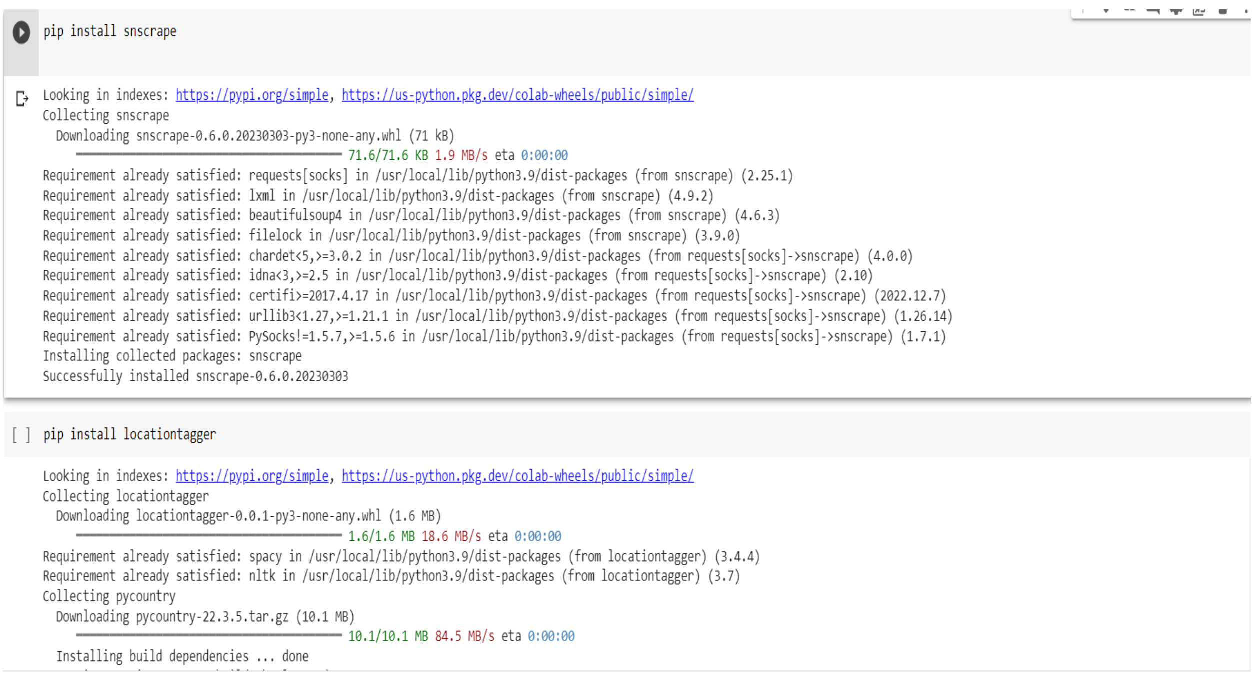
- We have observed that the existing systems mostly utilize convolutional neural networks to extract the location of the affected area from tweets. We are using a different approach where we will extract the location of the affected area by considering the geo-location of the user who recently tweeted, using Twitter API. We will also employ the BERT algorithm to extract the location tweet itself even when the user's geo-location is unavailable and also consider the context of the tweet. This approach is expected to enhance the model's accuracy, resulting in a more precise prediction for the affected area.
- The delay in reporting natural calamities on news channels can hinder the process of providing prompt assistance to those in need. To address this issue, we will add a feature to our system to notify NGOs about such incidents in real-time. This system will enable them to swiftly organize and deliver aid to the affected regions.

Baseline results (system/prototype)

First, we prepared a dataset to perform data modeling.
Using snscrape scraping tool(python package)we scraped tweets using a pre-defined keyword from Twitter.
Why snscrape?
Other options are available to do this job like GetOldTweets3(GOT), TWINT, Octoparse, etc.
Problems with other tools and packages.
1. GOT is no longer useful as Twitter has removed the endpoint the GOT uses.
2. Twint is an advanced tool written in python but Twitter has a more strict device +IP-ban.
3. Octoparse has problems like time consumption and a tough learning curve.

Code Snippets and Explanation

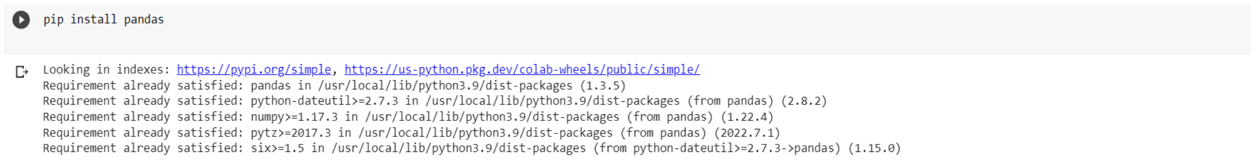
Installing all the packages required
pip install snscrape- package for social network scraping



```
pip install snscrape

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting snscrape
  Downloading snscrape-0.6.0.20230303-py3-none-any.whl (71 kB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 71.6/71.6 KB 1.9 MB/s eta 0:00:00
Requirement already satisfied: requests[socks] in /usr/local/lib/python3.9/dist-packages (from snscrape) (2.25.1)
Requirement already satisfied: lxml in /usr/local/lib/python3.9/dist-packages (from snscrape) (4.9.2)
Requirement already satisfied: beautifulsoup4 in /usr/local/lib/python3.9/dist-packages (from snscrape) (4.6.3)
Requirement already satisfied: filelock in /usr/local/lib/python3.9/dist-packages (from snscrape) (3.9.0)
Requirement already satisfied: chardet<5,>=3.0.2 in /usr/local/lib/python3.9/dist-packages (from requests[socks]->snscrape) (4.0.0)
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.9/dist-packages (from requests[socks]->snscrape) (2.10)
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3.9/dist-packages (from requests[socks]->snscrape) (2022.12.7)
Requirement already satisfied: urllib3<1.27,>=1.21.1 in /usr/local/lib/python3.9/dist-packages (from requests[socks]->snscrape) (1.26.14)
Requirement already satisfied: PySocks!=1.5.7,>=1.5.6 in /usr/local/lib/python3.9/dist-packages (from requests[socks]->snscrape) (1.7.1)
Installing collected packages: snscrape
Successfully installed snscrape-0.6.0.20230303
```

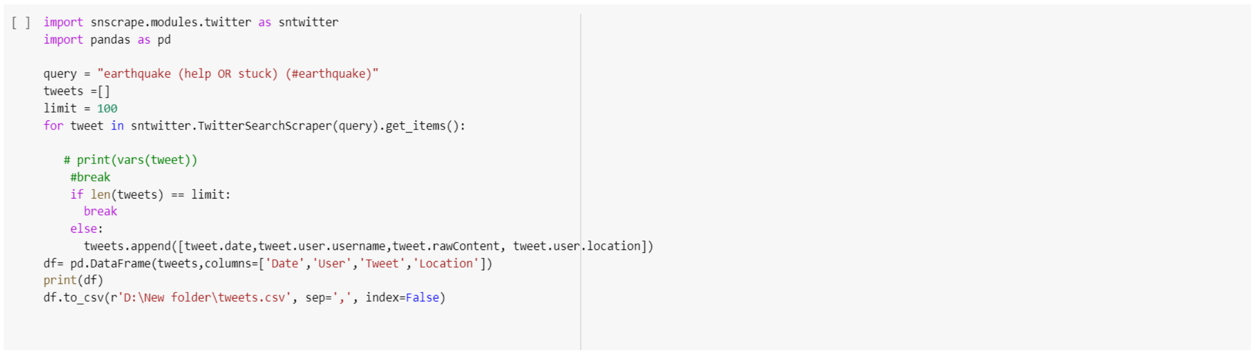
```
[ ] pip install locationtagger

Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Collecting locationtagger
  Downloading locationtagger-0.0.1-py3-none-any.whl (1.6 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 1.6/1.6 MB 18.6 MB/s eta 0:00:00
Requirement already satisfied: spacy in /usr/local/lib/python3.9/dist-packages (from locationtagger) (3.4.4)
Requirement already satisfied: nltk in /usr/local/lib/python3.9/dist-packages (from locationtagger) (3.7)
Collecting pycountry
  Downloading pycountry-22.3.5.tar.gz (10.1 MB)
     ━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━━ 10.1/10.1 MB 84.5 MB/s eta 0:00:00
  Installing build dependencies ... done
```

pip install pandas- using pandas library to show the tweets data that we are getting
import all the modules
1. We are defining a query to identify the hashtags or if the user used keywords like help, stuck, or earthquake
2. Creating panadas data frame to represent the fetched tweets in a particular format of columns('Date', 'User', 'Tweet')
3. Setting a predefined limit for the fetched tweets to avoid runtime error. Here we have set up a limit of 100 tweets to be displayed.
4. Writing the data frames to CSV file.

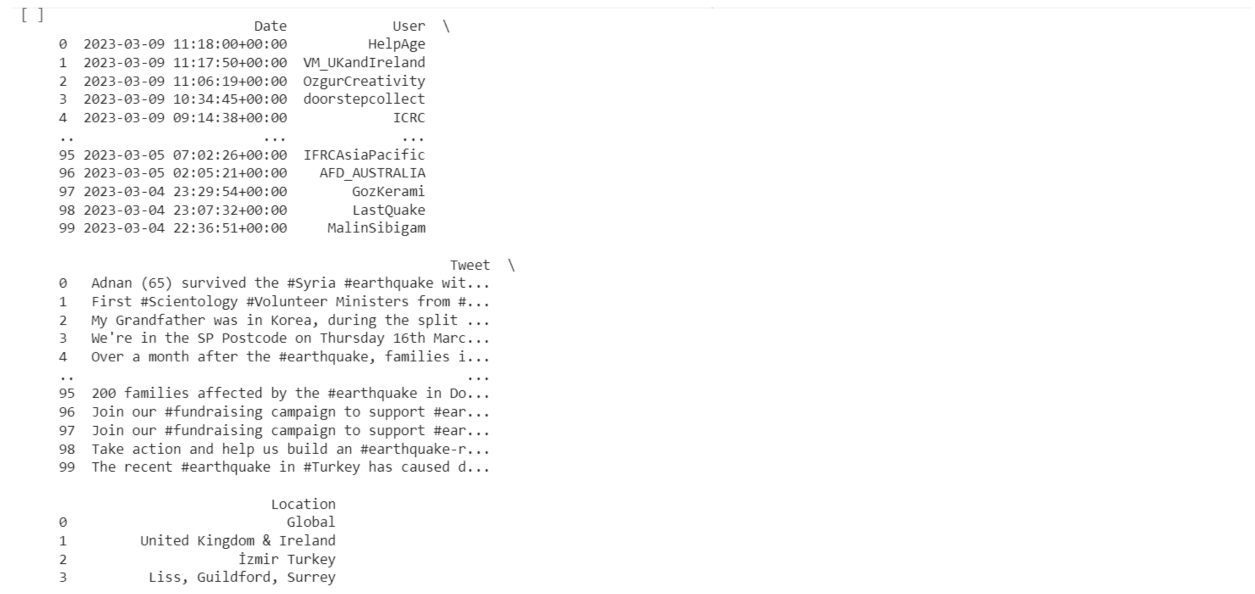


```
pip install pandas
```

```
Looking in indexes: https://pypi.org/simple, https://us-python.pkg.dev/colab-wheels/public/simple/
Requirement already satisfied: pandas in /usr/local/lib/python3.9/dist-packages (1.3.5)
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (2.8.2)
Requirement already satisfied: numpy>=1.17.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (1.22.4)
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.9/dist-packages (from pandas) (2022.7.1)
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.9/dist-packages (from python-dateutil>=2.7.3->pandas) (1.15.0)
```

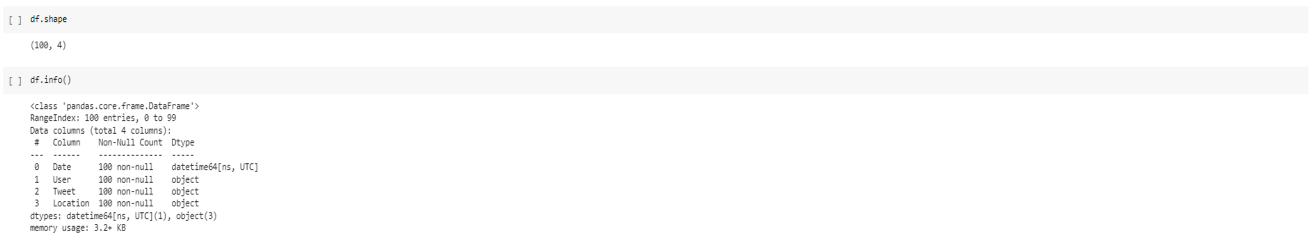


```
import snscrape.modules.twitter as sntwitter
import pandas as pd

query = "earthquake (help OR stuck) (#earthquake)"
tweets =[]
limit = 100
for tweet in sntwitter.TwitterSearchScraper(query).get_items():

   # print(vars(tweet))
    #break
    if len(tweets) == limit:
      break
    else:
      tweets.append([tweet.date,tweet.user.username,tweet.rawContent, tweet.user.location])
df= pd.DataFrame(tweets,columns=['Date','User','Tweet','Location'])
print(df)
df.to_csv(r'D:\New folder\tweets.csv', sep=',', index=False)
```



```
                         Date             User  \
0  2023-03-09 11:18:00+00:00          HelpAge
1  2023-03-09 11:17:50+00:00  VM_UKandIreland
2  2023-03-09 11:06:19+00:00  OzgurCreativity
3  2023-03-09 10:34:45+00:00  doorstepcollect
4  2023-03-09 09:14:38+00:00             ICRC
..                        ...              ...
95 2023-03-05 07:02:26+00:00  IFRCAsiaPacific
96 2023-03-05 02:05:21+00:00    AFD_AUSTRALIA
97 2023-03-04 23:29:54+00:00        GozKerami
98 2023-03-04 23:07:32+00:00        LastQuake
99 2023-03-04 22:36:51+00:00     MalinSibigam

                                                Tweet  \
0   Adnan (65) survived the #Syria #earthquake wit...
1   First #Scientology #Volunteer Ministers from #...
2   My Grandfather was in Korea, during the split ...
3   We're in the SP Postcode on Thursday 16th Marc...
4   Over a month after the #earthquake, families i...
..                                                ...
95  200 families affected by the #earthquake in Do...
96  Join our #fundraising campaign to support #ear...
97  Join our #fundraising campaign to support #ear...
98  Take action and help us build an #earthquake-r...
99  The recent #earthquake in #Turkey has caused d...

                        Location
0                         Global
1        United Kingdom & Ireland
2                    İzmir Turkey
3          Liss, Guildford, Surrey
```

Data Preprocessing, Analysis and Visualization



```
df.shape
```

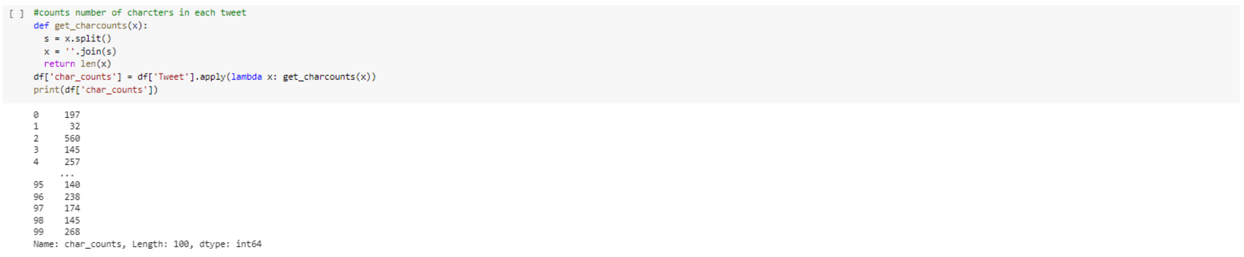
```
(100, 4)
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 100 entries, 0 to 99
Data columns (total 4 columns):
 #   Column    Non-Null Count  Dtype
---  ------    --------------  -----
 0   Date      100 non-null    datetime64[ns, UTC]
 1   User      100 non-null    object
 2   Tweet     100 non-null    object
 3   Location  100 non-null    object
dtypes: datetime64[ns, UTC](1), object(3)
memory usage: 3.2+ KB
```

No Null values in any of the column so going for Visualization

Count number of characters in each tweet



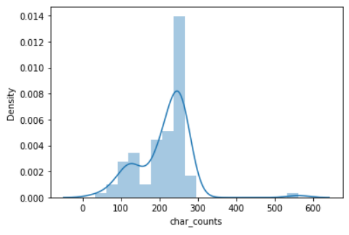
```
[ ] #counts number of charcters in each tweet
    def get_charcounts(x):
      s = x.split()
      x = ''.join(s)
      return len(x)
    df['char_counts'] = df['Tweet'].apply(lambda x: get_charcounts(x))
    print(df['char_counts'])
```

```
0     197
1      32
2     560
3     145
4     257
     ...
95    140
96    238
97    174
98    145
99    268
Name: char_counts, Length: 100, dtype: int64
```

Graph showing the distribution for the number of characters for tweets

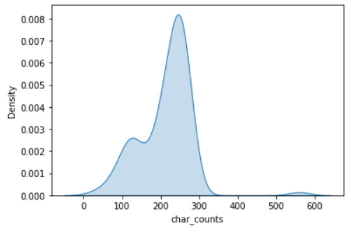
```
#plot a graph showing the distribution for the number of characters for tweets
import seaborn as sns
sns.distplot(df['char_counts'])
```

```
/usr/local/lib/python3.9/dist-packages/seaborn/distributions.py:2619: FutureWarning: `distplot` is a deprecated function and will be removed in a future version. Please adapt you
  warnings.warn(msg, FutureWarning)
<AxesSubplot:xlabel='char_counts', ylabel='Density'>
```

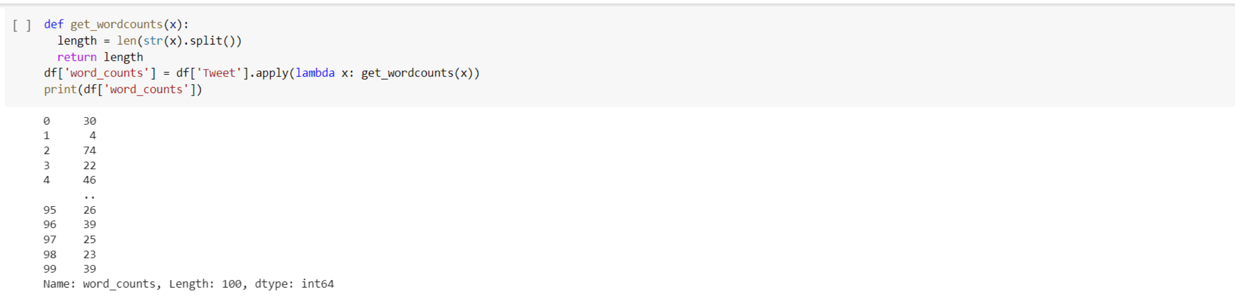


```
sns.kdeplot(df['char_counts'], shade= True)
```

```
<AxesSubplot:xlabel='char_counts', ylabel='Density'>
```



Count number of characters in each tweet



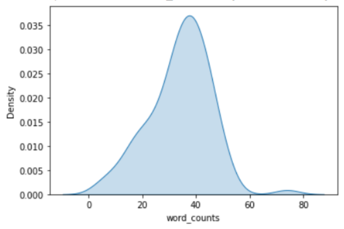
```
[ ] def get_wordcounts(x):
      length = len(str(x).split())
      return length
    df['word_counts'] = df['Tweet'].apply(lambda x: get_wordcounts(x))
    print(df['word_counts'])
```

```
0     30
1      4
2     74
3     22
4     46
      ..
95    26
96    39
97    25
98    23
99    39
Name: word_counts, Length: 100, dtype: int64
```

Graph showing highest number of words in a tweet are around 40 to 45

```
[ ] sns.kdeplot(df['word_counts'],shade=True)
    #showing highest number of words in a tweet are around 40 to 45
```

```
<AxesSubplot:xlabel='word_counts', ylabel='Density'>
```



Count number of stopwords in each tweet

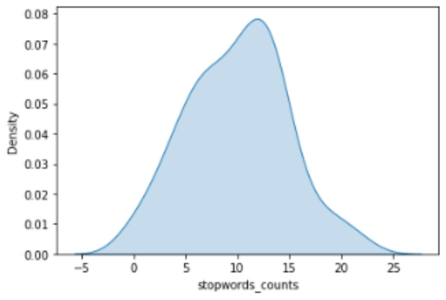
```
[ ] import nltk
    nltk.download('stopwords')
    from nltk.corpus import stopwords
    stopwords=stopwords.words('english')
    def _get_stopwords_counts(x):
      l = len([t for t in x.split() if t in stopwords])
      return l
    df['stopwords_counts'] = df['Tweet'].apply(lambda x: _get_stopwords_counts(x))
    print(df['stopwords_counts'])
```

```
0     11
1      0
2     11
3      3
4     16
      ..
95    10
96    14
97     4
98     6
99    11
Name: stopwords_counts, Length: 100, dtype: int64
[nltk_data] Downloading package stopwords to /root/nltk_data...
[nltk_data]   Unzipping corpora/stopwords.zip.
```

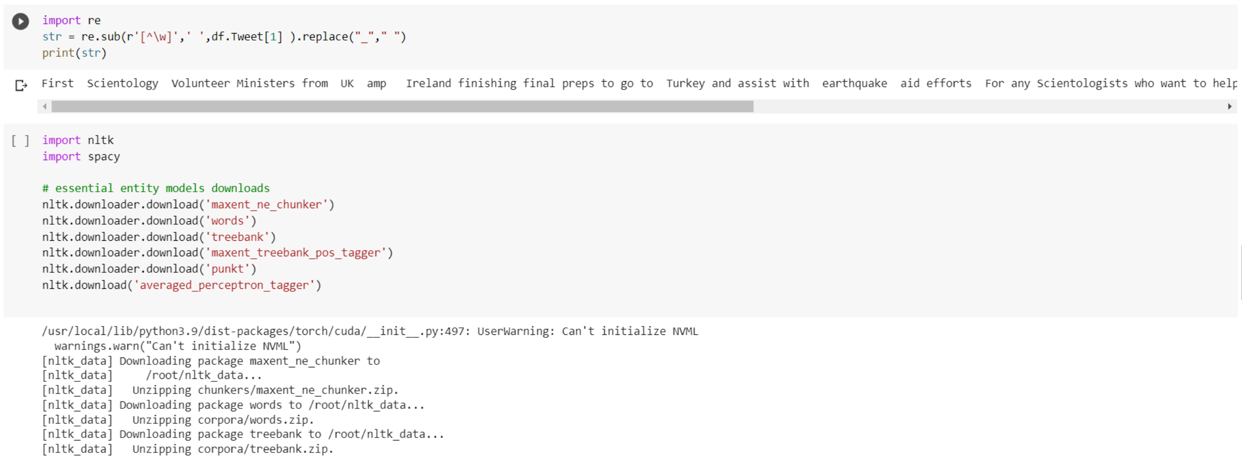
Graph showing number of stopwords in each tweet

```
[ ] sns.kdeplot(df['stopwords_counts'],shade=True)
```

```
<AxesSubplot:xlabel='stopwords_counts', ylabel='Density'>
```



Importing and using re.sub() function to replace underscores with white spaces to handle the cases like '#nepal_Earthquake'.



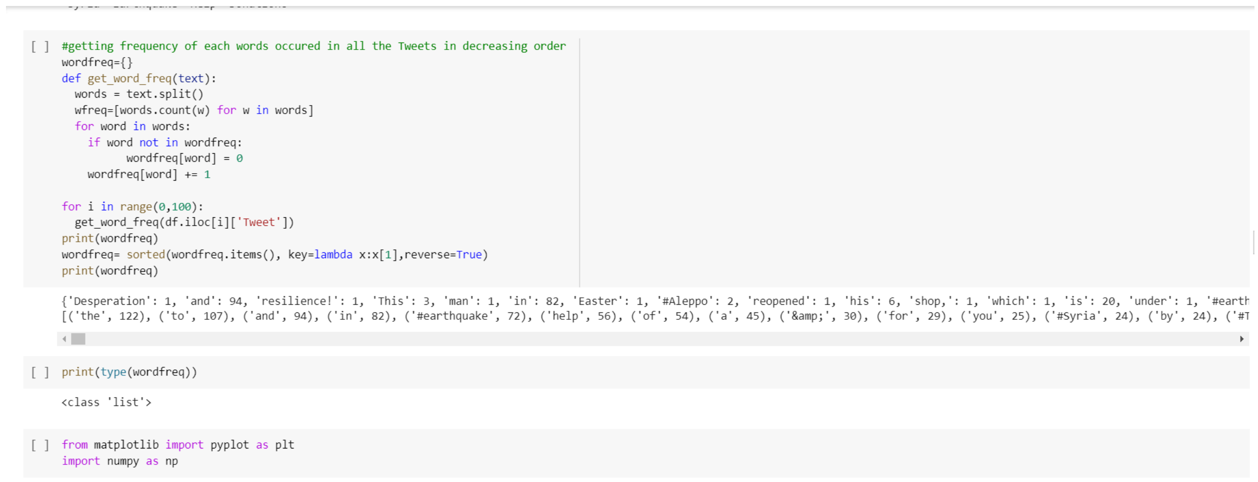
```
import re
str = re.sub(r'[^\w]',' ',df.Tweet[1] ).replace("_"," ")
print(str)
```

```
First  Scientology  Volunteer Ministers from  UK  amp   Ireland finishing final preps to go to  Turkey and assist with  earthquake  aid efforts  For any Scientologists who want to help
```

```
[ ] import nltk
    import spacy

    # essential entity models downloads
    nltk.downloader.download('maxent_ne_chunker')
    nltk.downloader.download('words')
    nltk.downloader.download('treebank')
    nltk.downloader.download('maxent_treebank_pos_tagger')
    nltk.downloader.download('punkt')
    nltk.download('averaged_perceptron_tagger')
```

```
/usr/local/lib/python3.9/dist-packages/torch/cuda/__init__.py:497: UserWarning: Can't initialize NVML
  warnings.warn("Can't initialize NVML")
[nltk_data] Downloading package maxent_ne_chunker to
[nltk_data]     /root/nltk_data...
[nltk_data]   Unzipping chunkers/maxent_ne_chunker.zip.
[nltk_data] Downloading package words to /root/nltk_data...
[nltk_data]   Unzipping corpora/words.zip.
[nltk_data] Downloading package treebank to /root/nltk_data...
[nltk_data]   Unzipping corpora/treebank.zip.
```

Finding the frequency of each word occurred in all the tweets in decreasing order



```
[ ] #getting frequency of each words occured in all the Tweets in decreasing order
    wordfreq={}
    def get_word_freq(text):
      words = text.split()
      wfreq=[words.count(w) for w in words]
      for word in words:
        if word not in wordfreq:
              wordfreq[word] = 0
        wordfreq[word] += 1

    for i in range(0,100):
      get_word_freq(df.iloc[i]['Tweet'])
    print(wordfreq)
    wordfreq= sorted(wordfreq.items(), key=lambda x:x[1],reverse=True)
    print(wordfreq)
```

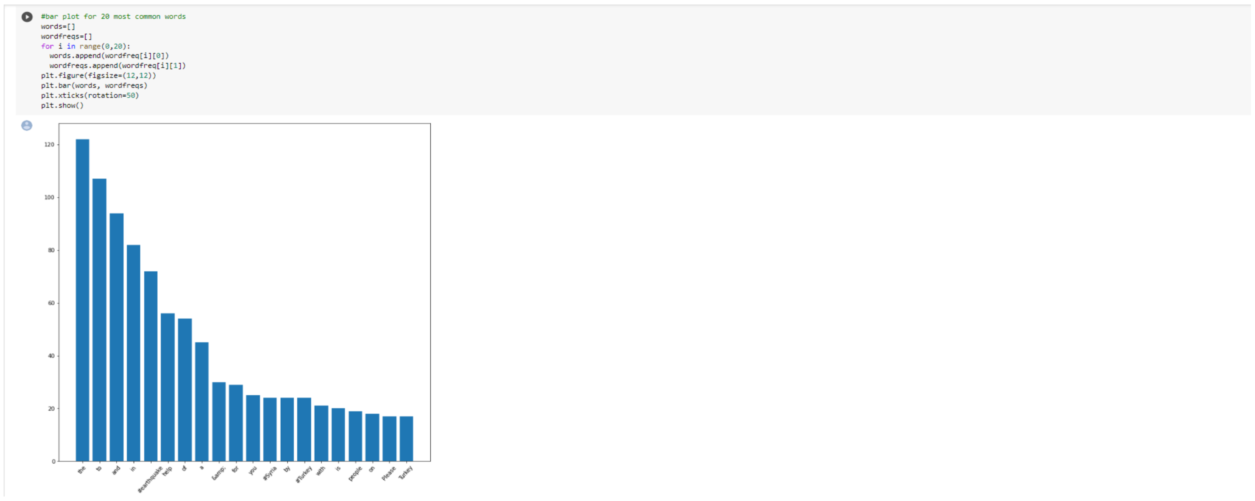
```
{'Desperation': 1, 'and': 94, 'resilience!': 1, 'This': 3, 'man': 1, 'in': 82, 'Easter': 1, '#Aleppo': 2, 'reopened': 1, 'his': 6, 'shop,': 1, 'which': 1, 'is': 20, 'under': 1, '#earth
[('the', 122), ('to', 107), ('and', 94), ('in', 82), ('#earthquake', 72), ('help', 56), ('of', 54), ('a', 45), ('&amp;', 30), ('for', 29), ('you', 25), ('#Syria', 24), ('by', 24), ('#T
```

```
[ ] print(type(wordfreq))
```

```
<class 'list'>
```

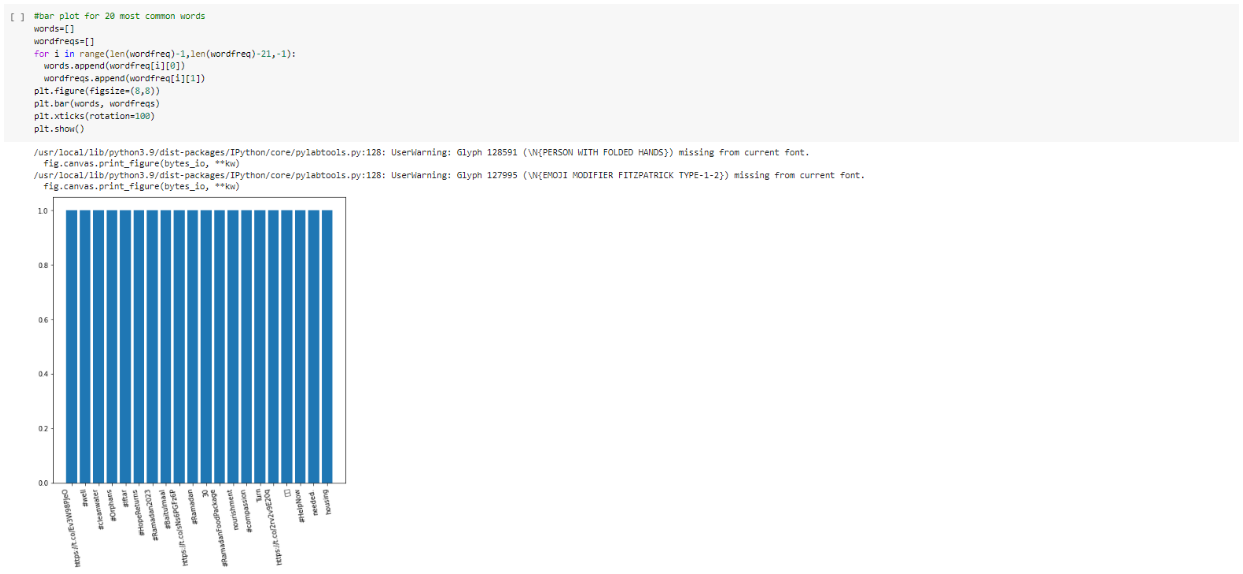
```
[ ] from matplotlib import pyplot as plt
    import numpy as np
```

Bar Plot showing the frequency of occurrence of 20 most common words



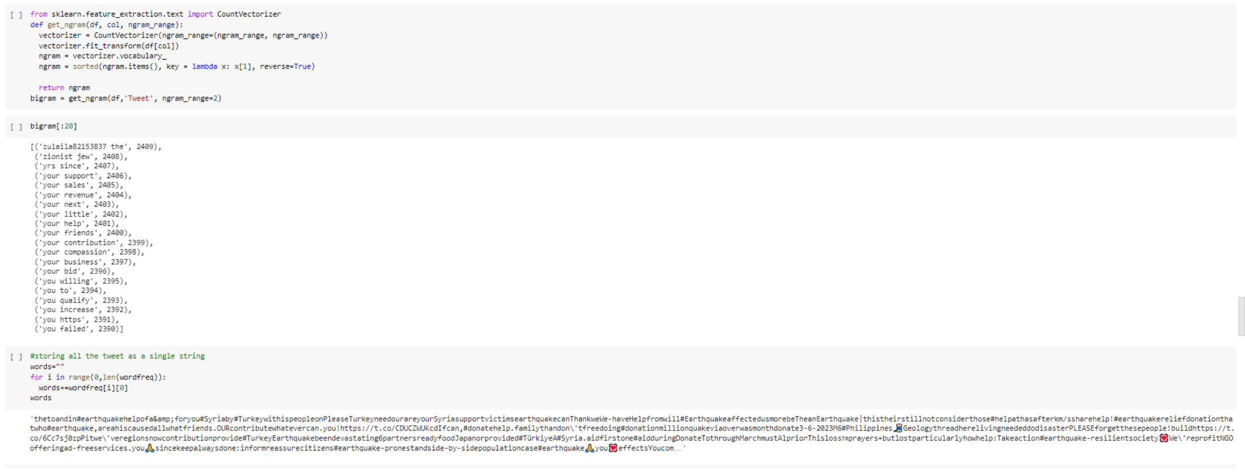
```
#bar plot for 20 most common words
words=[]
wordfreqs=[]
for i in range(0,20):
  words.append(wordfreq[i][0])
  wordfreqs.append(wordfreq[i][1])
plt.figure(figsize=(12,12))
plt.bar(words, wordfreqs)
plt.xticks(rotation=50)
plt.show()
```

Bar Plot for 20 most frequent words



```
[ ] #bar plot for 20 most common words
    words=[]
    wordfreqs=[]
    for i in range(len(wordfreq)-1,len(wordfreq)-21,-1):
      words.append(wordfreq[i][0])
      wordfreqs.append(wordfreq[i][1])
    plt.figure(figsize=(8,8))
    plt.bar(words, wordfreqs)
    plt.xticks(rotation=100)
    plt.show()

    /usr/local/lib/python3.9/dist-packages/IPython/core/pylabtools.py:128: UserWarning: Glyph 128591 (\N{PERSON WITH FOLDED HANDS}) missing from current font.
      fig.canvas.print_figure(bytes_io, **kw)
    /usr/local/lib/python3.9/dist-packages/IPython/core/pylabtools.py:128: UserWarning: Glyph 127995 (\N{EMOJI MODIFIER FITZPATRICK TYPE-1-2}) missing from current font.
      fig.canvas.print_figure(bytes_io, **kw)
```

Finding the List of biwords and there frequencies
Storing all the tweets as a single string



```
[ ] from sklearn.feature_extraction.text import CountVectorizer
    def get_ngram(df, col, ngram_range):
      vectorizer = CountVectorizer(ngram_range=(ngram_range, ngram_range))
      vectorizer.fit_transform(df[col])
      ngram = vectorizer.vocabulary_
      ngram = sorted(ngram.items(), key = lambda x: x[1], reverse=True)

      return ngram
    bigram = get_ngram(df,'Tweet', ngram_range=2)
```

```
[ ] bigram[:20]
```

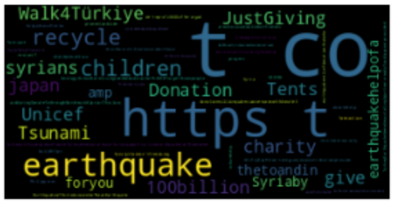
```
[('zulaila82153837 the', 2409),
 ('zionist jew', 2408),
 ('yrs since', 2407),
 ('your support', 2406),
 ('your sales', 2405),
 ('your revenue', 2404),
 ('your next', 2403),
 ('your little', 2402),
 ('your help', 2401),
 ('your friends', 2400),
 ('your contribution', 2399),
 ('your compassion', 2398),
 ('your business', 2397),
 ('your bid', 2396),
 ('you willing', 2395),
 ('you to', 2394),
 ('you qualify', 2393),
 ('you increase', 2392),
 ('you https', 2391),
 ('you failed', 2390)]
```

```
[ ] #storing all the tweet as a single string
    words=""
    for i in range(0,len(wordfreq)):
      words+=wordfreq[i][0]
    words
```

```
'thetoandin#earthquakehelpofa&amp;foryou#Syriaby#TurkeywithispeopleonPleaseTurkeyneedourareyourSyriasupportvictimsearthquakecanThankweWe-haveHelpfromwill#EarthquakeaffectedusmorebeTheanEarthquake|thistheirstillnotconsiderthose#helpathasafterkm/ssharehelp!#earthquakereliefdonationtha
twho#earthquake,areahiscausedallwhatfriends.OURcontributewhatevercan.you!https://t.co/CDUCZWUKcdIfcan,#donatehelp.familythandon\'tfreedoing#donationmillionquakeviaoverwasmonthdonate3-6-2023M6#Philippines🧵Geologythreadherelivingneededdodisaster PLEASEforgetthesepeople!buildhttps://t.
co/6Cc7sj0zpPitwe\'veregionsnowcontributionprovide#TurkeyEarthquakebeendevastating6partnersreadyfoodJapanorprovided#TürkiyeA#Syria.aidfirstone#aidduringDonateTothroughMarchmustAlpriorThislossTRprayers+butlostparticularlyhowhelp:Takeaction#earthquake-resilientsociety💌We\'reprofitNGO
offeringad-freeservices.you🙏sincekeepalwaysdone:informreassurecitizens#earthquake-pronestandside-by-sidepopulationcase#earthquake🙏you💌effectsYoucom...'
```

Printing wordcloud for the Tweets

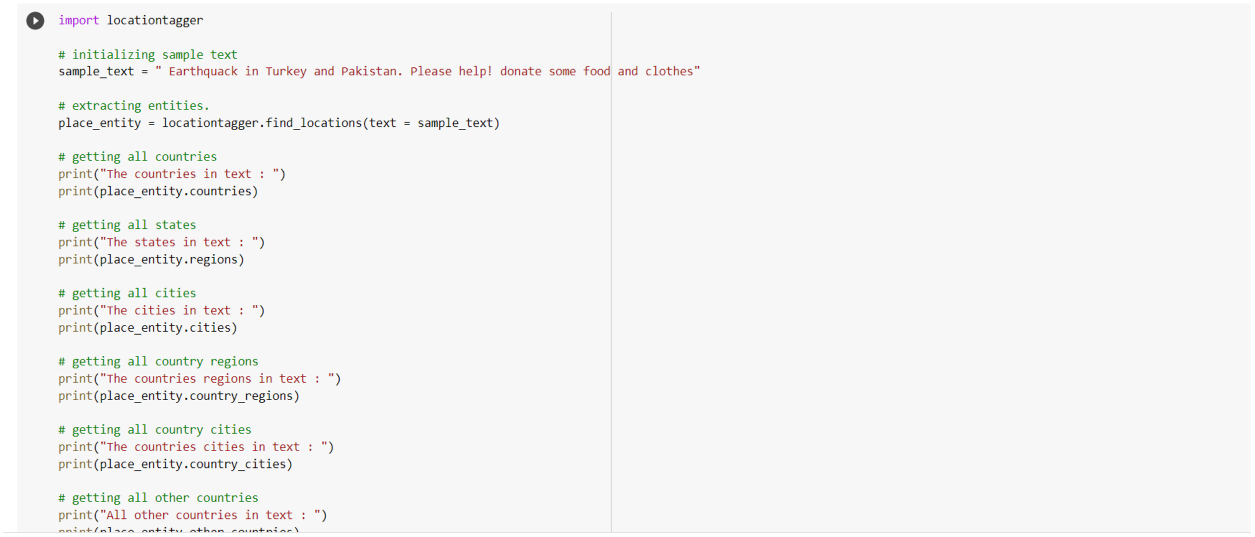
```
[ ] #printing wordcloud for the tweets
    from wordcloud import WordCloud
    word_cloud = WordCloud(max_font_size=80).generate(words)
    plt.imshow(word_cloud)
    plt.axis('off')
    plt.show()
```



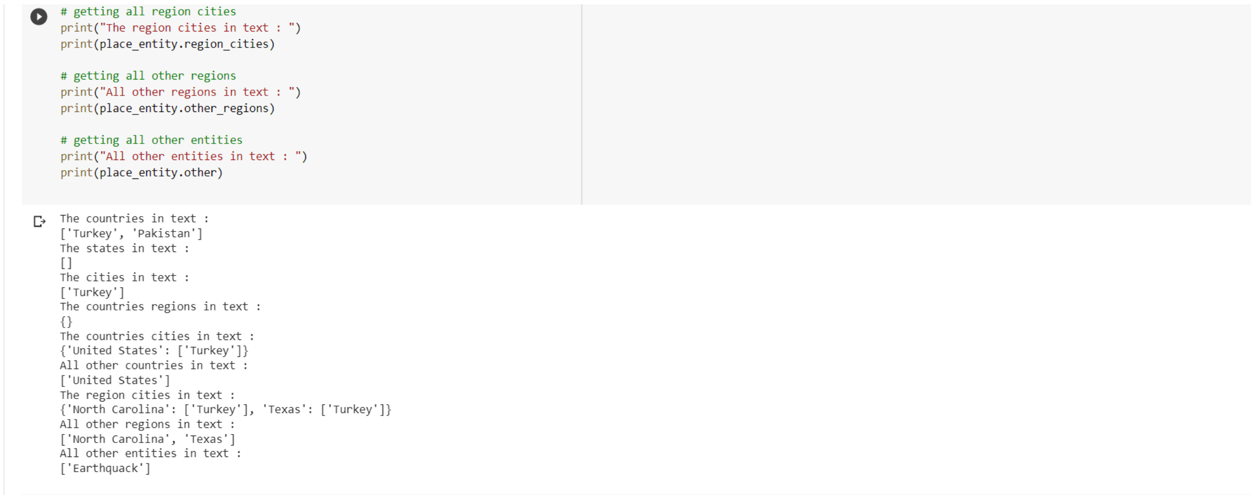
Trying out different algorithms to fetch out the entities like countries, cities, and region from the database collected to find out specifically where the earthquake occurred.

Here we are trying out the python package location tagger which is used to filter out place names(locations) amongst all the entities found with NER(Named Entity Recognition).

Installing package using pip
Install all the useful libraries(nltk, spacy etc)
Using the find_locations() function to return all the entities with location information in the given text.
printing the list of countries, cities, and regions from the entities obtained.

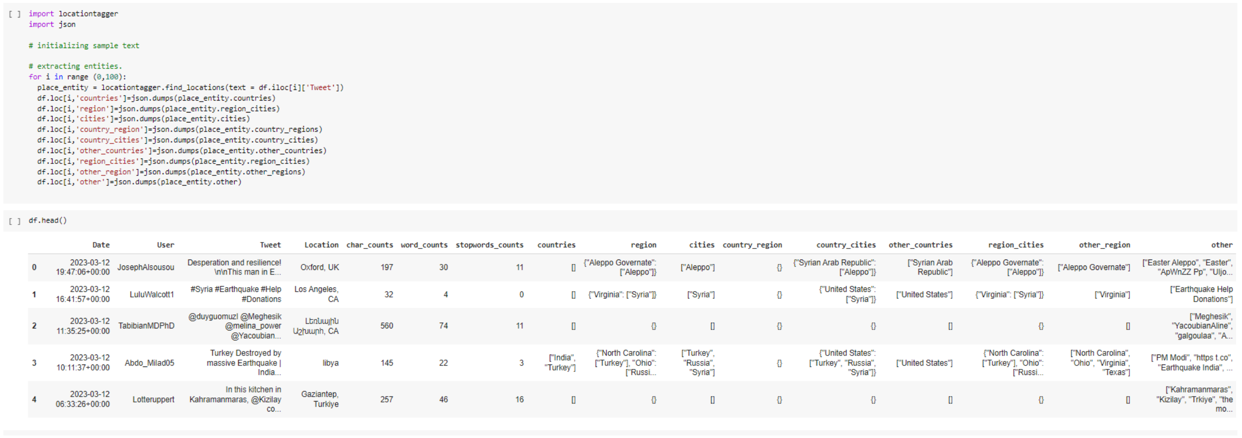


```python
import locationtagger

# initializing sample text
sample_text = " Earthquack in Turkey and Pakistan. Please help! donate some food and clothes"

# extracting entities.
place_entity = locationtagger.find_locations(text = sample_text)

# getting all countries
print("The countries in text : ")
print(place_entity.countries)

# getting all states
print("The states in text : ")
print(place_entity.regions)

# getting all cities
print("The cities in text : ")
print(place_entity.cities)

# getting all country regions
print("The countries regions in text : ")
print(place_entity.country_regions)

# getting all country cities
print("The countries cities in text : ")
print(place_entity.country_cities)

# getting all other countries
print("All other countries in text : ")
print(place_entity.other_countries)
```



```python
# getting all region cities
print("The region cities in text : ")
print(place_entity.region_cities)

# getting all other regions
print("All other regions in text : ")
print(place_entity.other_regions)

# getting all other entities
print("All other entities in text : ")
print(place_entity.other)
```

```
The countries in text :
['Turkey', 'Pakistan']
The states in text :
[]
The cities in text :
['Turkey']
The countries regions in text :
{}
The countries cities in text :
{'United States': ['Turkey']}
All other countries in text :
['United States']
The region cities in text :
{'North Carolina': ['Turkey'], 'Texas': ['Turkey']}
All other regions in text :
['North Carolina', 'Texas']
All other entities in text :
['Earthquack']
```



```python
[ ] import locationtagger
    import json

    # initializing sample text

    # extracting entities.
    for i in range (0,100):
      place_entity = locationtagger.find_locations(text = df.iloc[i]['Tweet'])
      df.loc[i,'countries']=json.dumps(place_entity.countries)
      df.loc[i,'region']=json.dumps(place_entity.region_cities)
      df.loc[i,'cities']=json.dumps(place_entity.cities)
      df.loc[i,'country_region']=json.dumps(place_entity.country_regions)
      df.loc[i,'country_cities']=json.dumps(place_entity.country_cities)
      df.loc[i,'other_countries']=json.dumps(place_entity.other_countries)
      df.loc[i,'region_cities']=json.dumps(place_entity.region_cities)
      df.loc[i,'other_region']=json.dumps(place_entity.other_regions)
      df.loc[i,'other']=json.dumps(place_entity.other)
```

```python
[ ] df.head()
```

	Date	User	Tweet	Location	char_counts	word_counts	stopwords_counts	countries	region	cities	country_region	country_cities	other_countries	region_cities	other_region	other
0	2023-03-12 19:47:06+00:00	JosephAlsousou	Desperation and resilience! \n\nThis man in E...	Oxford, UK	197	30	11	[]	{"Aleppo Governate": ["Aleppo"]}	["Aleppo"]	{}	{"Syrian Arab Republic": ["Aleppo"]}	["Syrian Arab Republic"]	{"Aleppo Governate": ["Aleppo"]}	["Aleppo Governate"]	["Easter Aleppo", "Easter", "ApWnZZ Pp", "Uljo...
1	2023-03-12 16:41:57+00:00	LuluWalcott1	#Syria #Earthquake #Help #Donations	Los Angeles, CA	32	4	0	[]	{"Virginia": ["Syria"]}	["Syria"]	{}	{"United States": ["Syria"]}	["United States"]	{"Virginia": ["Syria"]}	["Virginia"]	["Earthquake Help Donations"]
2	2023-03-12 11:35:25+00:00	TabibianMDPhD	@duyguomuzl @Meghesik @melina_power @Yacoubian...	Լեռնային Աշխարհ, CA	560	74	11	[]	{}	[]	{}	{}	[]	{}	[]	["Meghesik", "YacoubianAline", "galgoulaa", "A...
3	2023-03-12 10:11:37+00:00	Abdo_Milad05	Turkey Destroyed by massive Earthquake | India...	libya	145	22	3	["India", "Turkey"]	{"North Carolina": ["Turkey"], "Ohio": ["Russi...	["Turkey", "Russia", "Syria"]	{}	{"United States": ["Turkey", "Russia", "Syria"]}	["United States"]	{"North Carolina": ["Turkey"], "Ohio": ["Russi...	["North Carolina", "Ohio", "Virginia", "Texas"]	["PM Modi", "https t.co", "Earthquake India", ...
4	2023-03-12 06:33:26+00:00	Lotteruppert	In this kitchen in Kahramanmaras, @Kizilay co...	Gaziantep, Turkiye	257	46	16	[]	{}	[]	{}	{}	[]	{}	[]	["Kahramanmaras", "Kizilay", "Trkiye", "the mo...

[ ]

```python
import nltk
named_entities = []
nes = nltk.ne_chunk(nltk.pos_tag(nltk.word_tokenize(sample_text)))
for ne in nes:
    if type(ne) is nltk.tree.Tree:
        if (ne.label() == 'GPE' or ne.label() == 'PERSON' or ne.label() == 'ORGANIZATION'):
            l = []
            for i in ne.leaves():
                l.append(i[0])
            s = u' '.join(l)
            if not (s in named_entities):
                named_entities.append(s)

print(named_entities)
```

```
['PO', 'LeeOnTheSolent', 'Fratton', 'Donate', 'REUSE', 'RECYCLE', 'SECONDLIFE']
```