IM-UH 1511 **Introduction to Digital Humanities**

HOMEWORK 5a

Twitter Search Mining

25 points totally

In [1]:
```
import time
start_time = time.perf_counter()
import pandas as pd
import json
from datetime import datetime, timedelta
import twitter ## pip install --user python-twitter
import os
import twitter
from bs4 import BeautifulSoup
from urllib.parse import quote, unquote
import requests

import warnings
warnings.filterwarnings("ignore", category=RuntimeWarning)
warnings.filterwarnings("ignore", category=UserWarning)
warnings.simplefilter('ignore')
```

In [2]:
```
def tweet_cleaner(status):
    payload = {}
    payload['screen_name'] = status['user']['screen_name']
    payload['created'] = pd.to_datetime(status['created_at'])
    payload['retweets'] = status['retweet_count']
    payload['favorites'] = status['favorite_count']
    payload['id'] = status['id']
    payload['reply_screen_name'] = status['in_reply_to_screen_name']
    payload['reply_id'] = status['in_reply_to_status_id']
    payload['source'] = BeautifulSoup(status['source']).text
    payload['lang'] = status['lang']

    if status['place']:
        payload['place'] = status['place']['country']
    else:
        payload['place'] = None

    if len(status['entities']['user_mentions']) > 0:
        payload['user_mentions'] = '; '.join([m['screen_name'] for m in sta
    else:
        payload['user_mentions'] = None

    if len(status['entities']['hashtags']) > 0:
        payload['hashtags'] = '; '.join([h['text'] for h in status['entitie
    else:
        payload['hashtags'] = None

    # If an account retweets another account, we should store that informat
    if 'retweeted_status' in status:
        rt_status = status['retweeted_status']
        if 'extended_tweet' in rt_status:
            payload['text'] = rt_status['extended_tweet']['full_text']
            if len(rt_status['extended_tweet']['entities']['hashtags']) > 0
                payload['hashtags'] = '; '.join([h['text'] for h in rt_stat
            else:
                payload['hashtags'] = None
        else:
            try:
                payload['text'] = rt_status['text']
            except:
                payload['text'] = rt_status['full_text']
            if len(rt_status['entities']['hashtags']) > 0:
                payload['hashtags'] = '; '.join([h['text'] for h in rt_stat
            else:
                payload['hashtags'] = None
        payload['is_retweet'] = True
        payload['retweeted_screen_name'] = rt_status['user']['screen_name']
        payload['retweeted_created'] = rt_status['created_at']
        payload['retweeted_source'] = BeautifulSoup(rt_status['source'], "l

    else:
        if status['truncated']:
            payload['text'] = status['extended_tweet']['full_text']
        else:
            try:
                payload['text'] = status['text']
```

```
            except:
                payload['text'] = status['full_text']
        payload['is_retweet'] = False
        payload['retweeted_screen_name'] = False
        payload['retweeted_created'] = False
        payload['retweeted_source'] = False

    return payload
```

In [3]:
```
# # My key information file is a text file ('twitter_keys.json') of the fol

# {
#     "consumer_key":"API key",
#  "consumer_secret":"API secret key",
#  "access_token_key":"Access token",
#  "access_token_secret":"Access token secret"
# }
```

In [4]:
```
# Load my key information from disk

with open('twitter_keys.json','r') as f:
    twitter_keys = json.load(f)

# Authenticate with the Twitter API using the twitter_keys dictionary
# The "tweet_mode='extended' allows us to see the full 280 characters in tw

api = twitter.Api(**twitter_keys,tweet_mode='extended')
```

In [5]:
```
# # Alternatively, you can just enter your keys directly into the Api funct

# api = twitter.Api(consumer_key = 'API key',
#                   consumer_secret = 'API secret key',
#                   access_token_key = 'Access token',
#                   access_token_secret = 'Access token secret',
#                   tweet_mode='extended')
```

Search API

Twitter's search API (https://developer.twitter.com/en/docs/tweets/search/api-reference/get-search-tweets) provides an endpoint to search for tweets matching a query for terms, accounts, hashtags, language, locations, and date ranges. This API endpoint has a rate limit of 180 requests per 15-minute window with 100 statuses per request: or 18,000 statuses per window or 72,000 statuses per hour.

You can explore some of the search functionality through Twitter's advanced search interface (https://twitter.com/search-advanced). Note that the standard search API (https://developer.twitter.com/en/docs/tweets/search/overview/standard) only provides a limited access to sample of tweets in the past 7 days, you'll need to pay more to access historical APIs (https://developer.twitter.com/en/docs/tutorials/choosing-historical-api.html).

In [6]:
```
search_term = "Louvre"
no_tweets = 2000 # This number should be less than 18000! [read above]

search_tweets = []

while True:
    # Get the first set of tweets
    if len(search_tweets) == 0:
        query = api.GetSearch(term=search_term,count=100,result_type='recen
        search_tweets += query['statuses']
    # Keep getting tweets
    else:
        # Find the last tweet to use as a max_id
        max_id = search_tweets[-1]['id']
        # Get the next set of tweets
        query = api.GetSearch(term=search_term,count=100,return_json=True,m
        # Add them to the list of tweets
        search_tweets += query['statuses']
    # When to stop?
    if len(search_tweets) > no_tweets:
        break

print("There are {0:,} tweets in the collection.".format(len(search_tweets)
```

```
There are 2,100 tweets in the collection.
```

In [7]:
```
search_statuses_flat = []
search_errors = []

for i,status in enumerate(search_tweets):
    try:
        payload = tweet_cleaner(status)
        search_statuses_flat.append(payload)
    except:
        search_errors.append(str(i))

if len(search_errors) == 0:
    print("There were no errors!")
else:
    print("There were errors at the following indices:", '; '.join(search_e

search_df = pd.DataFrame(search_statuses_flat)
try:
    search_df['created'] = pd.to_datetime(search_df['created'])
#     search_df['created'] = search_df['created'].dt.tz_convert(None)
except KeyError as ex:
    print(ex)
search_df['created'] = search_df['created'].dt.tz_convert(None)
search_df.tail()
```

```
There were no errors!
```

Out[7]:

	screen_name	created	retweets	favorites	id	reply_screen_name	
2095	BanuOzdemirChp	2020-03-17 19:58:19	3	21	1240004558425862144	None	
2096	bar_evrim	2020-03-17 19:58:14	0	1	1240004535529193472	BiraGobegi01	1.24
2097	Disfosgen	2020-03-17 19:58:02	85	0	1240004484945924096	None	
2098	zzabaa1	2020-03-17 19:57:54	44	0	1240004452788195330	None	
2099	DraggyslZ	2020-03-17 19:57:30	797	0	1240004353211215872	None	

In [8]:
```
# fdf=search_df[search_df['text'].astype(str).str.contains(search_term)] #"
# print(len(fdf))
```

In [10]:
```
stc=search_term.replace(" ","")
plname=stc+'_df.pic'
search_df.to_pickle(plname)
```

In [ ]:
```
print("Run in %.2f seconds (%.2f minutes)" %(time.perf_counter() - start_ti
```

In [ ]:
```

```