DATA COLLECTION

Twitter API
Python
Tweepy
PRE-PROCESSING

Parsing
So, what data columns does the Streaming API provide?

```
list(df)  # view available columns
```

```
'user.notifications',
'user.profile_background_color',
'user.profile_background_image_url',
'user.profile_background_image_url_https',
'user.profile_background_tile',
'user.profile_banner_url',
'user.profile_image_url',
'user.profile_image_url_https',
'user.profile_link_color',
'user.profile_sidebar_border_color',
'user.profile_sidebar_fill_color',
'user.profile_text_color',
'user.profile_use_background_image',
'user.protected',
'user.screen_name',
'user.statuses_count',
'user.time_zone',
'user.url',
'user.utc_offset',
'user.verified'
```
PRE-PROCESSING

Parsing
Cleaning/filtering
Let's keep a set of *sociologically interesting* columns.

```python
# get the desired columns by name
cols = list([
    'created_at',  # time of tweet
    'user.name',   # about the author
    'user.screen_name',
    'user.description',
    'user.location',
    'user.followers_count',
    'user.friends_count',
    'in_reply_to_screen_name',  # addressee
    'entities.user_mentions.screen_name',
    'entities.hashtags.text',  # tweet contents
    'entities.media.type',
    'entities.media.url',
    'text',
    'favorite_count',  # tweet metadata
    #['reply_count'],
    #['retweet_count'],
    #['quote_count'],
    ['lang']
])

tweets_df = df[cols]
tweets_df.columns = [ 'time', 'user', 'user_sn', 'desc', 'loc', 'followers', 'friends',
    'to_user_sn', 'mentioning_sns', 'hashtags',
    'media_type', 'media_url', 'text', 'faves', 'lang']

tweets_df = tweets_df.replace({'\n': ' '}, regex=True)  # remove linebreaks in the dataframe
tweets_df = tweets_df.replace({'\t': ' '}, regex=True)  # remove tabs in the dataframe
tweets_df = tweets_df.replace({'\r': ' '}, regex=True)  # remove carriage return in the dataframe
```
Let's keep a set of **sociologically interesting** columns.

```r
# get the desired columns by name
cols = list([ ['created_at'] + # time of tweet
              ['user.name'] + # about the author
              ['user.screen_name'] +
              ['user.description'] +
              ['user.location'] +
              ['user.followers_count'] +
              ['user.friends_count'] +
              ['in_reply_to_screen_name'] +  # addressee
              ['entities.user_mentions.screen_name'] +
              ['entities.hashtags.text'] + # tweet contents
              ['entities.media.type'] +
              ['entities.media.url'] +
              ['text'] +
              ['favorite_count'] +
              ['#reply_count'] +
              ['#retweet_count'] +
              ['#quote_count'] +
              ['lang'])
```
# Create weighted graph from M

```python
G = nx.Graph()
for u, v, data in M.edges(data=True):
    w = data['weight'] if 'weight' in data else 1.0
    if G.has_edge(u, v):
        G[u][v]['weight'] += w
    else:
        G.add_edge(u, v, weight=w)
```

We now have a graph object:

```python
print(nx.info(G))
```

Name:
Type: Graph
Number of nodes: 99262
Number of edges: 149684
Average degree: 3.0159
# How many subgraphs does G consist of?
```
nx.number_connected_components(G)
```
2165

# See sizes of the top 10 connected components
```
for g in sorted(nx.connected_component_subgraphs(G), key=len, reverse=True)[:10]:
    print(len(g.nodes()))
```
```
94097
39
27
19
19
17
17
14
13
11
```
Let's keep a set of sociologically interesting columns.

```r
# get the desired columns by name
cols = list(["created_at"]+    # time of tweet
             ["user.name"]+    # about the author
             ["user.screen_name"]+
             ["user.description"]+
             ["user.location"]+
             ["user.followers_count"]+
             ["user.friends_count"]+
             ["in_reply_to_screen_name"]+  # addressee
             ["entities.user_mentions.screen_name"]+
             ["entities.hashtags.text"]+  # tweet contents
             ["entities.media.type"]+
             ["entities.media.url"]+
             ["text"]+                    # tweet metadata
             ["favorite_count"]+
             ["reply_count"]+
             ["retweet_count"]+
             ["quote_count"]+
             ["lang"])
```
RT @CatSmithMP: Look out for this ad van in Fleetwood, Knott End and Lancaster today! #VoteLabour https://t.co/1zegSYZ9x8

RT @davidschneider: A personal appeal to the young. From a man who's very much on their wavelength. #GeneralElection17 https://t.co/1nP5R3J...

RT @JeremyCorbyn4PM: The money shot. #TimeForCorbyn #VoteLabour https://t.co/BcgikdBy5

RT @rosa_francesca: remember for tomorrow #GeneralElection17 https://t.co/9UDU5HSdOp

I mean I do care. If you have a conscious and like having healthcare available. Vote #labour don't be an idiot.

The night before an election do May and Corbyn hear the music that plays the night before the boardroom on the apprentice? #GE2017

RT @PierceRooney: BBC didn’t want you to see this #Corbyn video. Wonder why #GE17 https://t.co/BPGpG8BKyI

@TheSun Did my last bit of campaigning in York Central this evening with @EdYoung4York and @ConservativesYO. Best of luck! #GE2017

RT @alexnunns: Extraordinary scenes in Islington where #Corbyn has arrived. People chanting "He's coming home, Corbyn's coming hom...

RT @lively63: If you're not voting tomorrow, please keep it to yourself cos actively not caring about your future is embarrassing #Genera...

RT @Dorsetghost: Vote #Labour tomorrow for a fairer society, No more Austerity, If you're young don't forget to vote https://t.co/5tN921F0V7

RT @Harryslistand: I can't begin to express the pride I have in #JeremyCorbyn &amp; the whole Labour election team for the mountains they...

Great article about voting choice and whether you are actually making one or being coerced into one of 2 only... https://t.co/Yg6LkwvmIZ

RT @JamillaTweets: this better not be how we get woken up on Friday morning so go out and vote!!! #GE2017 https://t.co/h33TPqGvDSD

Right-wing parties often make electoral gains after violent attacks, but the #GE2017 isn't as clear cut... https://t.co/1Qqq9eFDL

@ShaziaAhwan See https://t.co/DdKCO1nHXI for impartial #GeneralElection17 media data analysis. Full report @ https://t.co/rccZWGd4bg RT!

RT @4Salbannach: #VoteSNP #GE17 NHS chiefs told to secretly draw up more 'unthinkable cuts' - including A&amp;E https://t.co/5NaUC1Hf9a

RT @PeoplesMomentum: Powerful Stuff. #VoteLabour #GE2017 https://t.co/6KHRfj6Wup

Vote LibDem RT @AlanmakExposed: #Havant #Tory expenses scandal, worth a read. https://t.co/DrkEKEaiNh

i now# im annoying u all but it is important that everyone who sees this to vote corbyn.tomorrow lets change this country #GeneralElection17
ThingLinker

Python script by Simon Lindgren // @simonlindgren // simonlindgren.com.

Actor-Network theory (ANT) is about how human and non-human actants are connected in relational systems. It sees entities (humans, texts, machines, activitiees, ideas) as linked to each other in heterogeneous networks. Actors appear in any shape or material. The important thing is not if they have human agency, but whether they have the capacity to cause difference in the course of action of other entitites or not.

One way of analysing processes like these is to look at mechanisms through which an actor is connected to to other actors, and how those other actors in turn are linked to each other. Such analyses can be starting point of making closer assessments of things such as obligatory passage points, interressement, enrolment, and mobilisation). In short, networks are continually made and re-made, by actors who draw links and associations.

In [1]: # Required Python libraries
import glob, re
import pandas as pd
import spacy
nlp = spacy.load("en") # Set up spaCy with the the default model for English
In [6]:

dataset = []
data = open('tweets.txt', 'r').readlines()
for l in data:
    dataset.append(l)

We then clean the data from things that we do not want. The code below removes
urls, any non-alphanumeric characters, double spaces, double line-breaks, and
empty lines. Note however, that from the perspective of ANT, things such as urls or
emojis can definitely be interesting as actors, so the code below must be
customised for the research task at hand.

In [7]:
clean = []
for line in dataset:
    line = re.sub('^[^0-9a-zA-Z]+',' ', line)
    line = re.sub(' ',' ', line)
    line = re.sub(r'\n\n','\n', line)
    if not len(line.strip()) == 0:
        clean.append(line)

dataset = clean

In [8]:

    # Inspect the
dataset

Out[8]: ['RT CatSmithMP Look out for this ad van in Fleetwood Knott End and Lancaster today VoteLabour ','
'RT david schizophrenia A personal appeal to the young From a man who is very much on their wavelength GeneralElection17 ','
'RT JeremyCorbyn4PM The money shot TimeForCorbyn VoteLabour ']
Extracting Things

As our next step, we use the `spacy` Python library to extract what language technologists call 'Named Entities'. These entities – ‘Things’ – are considered here as potential actors, in the sense of ANT. With `spacy`, we will get the following tags:

- PERSON People, including fictional.
- NORP Nationalities or religious or political groups.
- FACILITY Buildings, airports, highways, bridges, etc.
- ORG Companies, agencies, institutions, etc.
- GPE Countries, cities, states.
- LOC Non-GPE locations, mountain ranges, bodies of water.
- PRODUCT Objects, vehicles, foods, etc. (Not services.)
- EVENT Named hurricanes, battles, wars, sports events, etc.
- WORK_OF_ART Titles of books, songs, etc.
- LANGUAGE Any named language.

It also extracts the following values:

- DATE Absolute or relative dates or periods.
- TIME Times smaller than a day.
- PERCENT Percentage, including "%".
- MONEY Monetary values, including unit.
- QUANTITY Measurements, as of weight or distance.
- ORDINAL "first", "second", etc.
- CARDINAL Numerals that do not fall under another type.
DATE: tomorrow
PERSON: Llanelli
CARDINAL: 92
CARDINAL: 92
ORG: NHS
ORG: VoteLabour
DATE: tomorrow
ORG: NHS
NORP: Indian
ORG: Lyca
CARDINAL: 26
ORG: HMR
DATE: the 50 days
PERSON: Theresa May
ORG: Labour
DATE: tomorrow