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

HOMEWORK 5b ¶

Twitter Networks

25 points totally

In [1]:
```
import twitter, random, operator, os, math, re, string, copy, itertools, pi
from collections import Counter, OrderedDict
import operator
from wordcloud import WordCloud
import pygraphviz
from networkx.drawing.nx_agraph import graphviz_layout

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

In [2]:
```
search_term = "coronavirus"
stc = search_term.replace(" ","")
plname = stc+"_df.pic"
search_df = pd.read_pickle(plname)
search_df = search_df.sort_values(by='created')
search_df.head()
```

Out[2]:

	screen_name	created	retweets	favorites	id	reply_screen_name	ı
999	OratorBlog	2020-02-24 16:17:42	57	0	1231976504923344897	None	
998	rickterp752	2020-02-24 16:17:42	0	0	1231976505925869568	SharylAttkisson	1.2319
997	GamboneBoeing	2020-02-24 16:17:42	2	0	1231976506269880323	None	
996	WonterghemVan	2020-02-24 16:17:43	40	0	1231976509499465728	None	
994	KaraMar111	2020-02-24 16:17:44	83	0	1231976515362934789	None	

In [3]:

```python
hashtags_list=[]
mentions_list=[]
for i in range(len(search_df)):
    h=search_df.iloc[i]['hashtags']
    if type(h)==str:
        h=h.replace("'","").split("; ")
        h=["#"+ht for ht in h]
    else:
        h=[]
    hashtags_list.append(h)
    me=search_df.iloc[i]['user_mentions']
    if type(me)==str:
        me=me.replace("'","").split("; ")
        me=["@"+men for men in me]
    else:
        me=[]
    mentions_list.append(me)
search_df['hashtags_list']=hashtags_list
search_df['mentions_list']=mentions_list
search_df['sender']=["@"+s for s in search_df["screen_name"].tolist()]
evd=search_df['created']
evd=pd.to_datetime(evd)
search_df['date']=evd
df=search_df[['date','sender','hashtags_list','mentions_list','lang','place
mind=df.date.min().strftime("%d-%m-%Y %H:%M:%S")
maxd=df.date.max().strftime("%d-%m-%Y %H:%M:%S")
print("The", search_term, "dataframe contains", len(df), "tweets", "from",
df.head(50)
```

```
The coronavirus dataframe contains 1000 tweets from 24-02-2020 16:17:42 t
o 24-02-2020 16:45:55
```

Out[3]:

	date	sender	hashtags_list	mentions_list	lang	place
999	2020-02-24 16:17:42	@OratorBlog	[]	[@j_ankrom]	en	None
998	2020-02-24 16:17:42	@rickterp752	[#AOC]	[@SharylAttkisson]	en	None
997	2020-02-24 16:17:42	@GamboneBoeing	[]	[@angela214, @AngelGotti5, @AOC]	en	None
996	2020-02-24 16:17:43	@WonterghemVan	[]	[@Deplorable_Man]	en	None
994	2020-02-24 16:17:44	@KaraMar111	[#Socialism101, #Socialism, #SocialismKills]	[@Jillibean557]	en	None
995	2020-02-24 16:17:44	@jsheas_smith	[]	[@MeghanMcCain]	en	None
993	2020-02-24 16:17:48	@glangendorf01	[#FightFor15, #LivingWageNow]	[@KimforSC]	en	None

	date	sender	hashtags_list	mentions_list	lang	place
992	2020-02-24 16:17:50	@super_mario04	[]	[@Clues, @tedcruz, @AOC]	en	None
990	2020-02-24 16:17:52	@vicky_vglend	[]	[@sugarrae, @RealCandaceO]	en	None
991	2020-02-24 16:17:52	@Jax6655	[]	[@MattMurph24]	en	None
988	2020-02-24 16:17:53	@WWG1WGA1962	[]	[@AOC]	en	None
989	2020-02-24 16:17:53	@dakane51	[]	[@JeffNelson966, @AOC]	en	None
987	2020-02-24 16:17:55	@crpgmcnamara	[]	[@AOC]	en	None
986	2020-02-24 16:17:57	@tlmichiels	[]	[]	en	None
985	2020-02-24 16:17:58	@PurpleEggsNHam	[]	[@CK33011698, @EllePole22, @free2expressvus, @...	en	None
984	2020-02-24 16:18:03	@citizenchnnl	[]	[@_waleedshahid, @AOC]	en	None
983	2020-02-24 16:18:03	@Tatyana_Hill	[]	[@jbplic]	en	None
982	2020-02-24 16:18:07	@sailinggirl73	[]	[@CortesSteve]	en	None
981	2020-02-24 16:18:09	@Z71199869	[]	[@diamondsoul317, @AOC]	en	None
980	2020-02-24 16:18:10	@RevTimCallow	[]	[@ThomasDierson, @RyWig, @EspritMouvant]	en	None
979	2020-02-24 16:18:11	@MeBeBlacksheep	[]	[]	en	None
978	2020-02-24 16:18:13	@fpedro1988	[]	[@AlytaDeLeon]	en	None
977	2020-02-24 16:18:15	@Pirula1315	[]	[@WayneDupreeShow]	en	None
976	2020-02-24 16:18:18	@DodgUSA24	[]	[@julie_kelly2]	en	None

	date	sender	hashtags_list	mentions_list	lang	place
975	2020-02-24 16:18:19	@deplorable_chet	[]	[@DrShayPhD, @cavaliereinesi1, @maineiacgirl71...	en	None
974	2020-02-24 16:18:21	@Lucy59jarvis	[]	[@PatVPeters, @nypost]	en	None
973	2020-02-24 16:18:23	@chiefragingbull	[]	[@BIZPACReview]	en	None
972	2020-02-24 16:18:24	@johnvitolopez2	[]	[@AOC]	en	None
971	2020-02-24 16:18:26	@SueRichter_Mann	[]	[]	en	None
970	2020-02-24 16:18:27	@marlowe_edward	[]	[]	en	None
969	2020-02-24 16:18:30	@hiro2pro	[]	[@Yangels6, @AGsurfer6, @cricketsateve, @Charl...	en	None
968	2020-02-24 16:18:30	@Rip_Narfer	[]	[]	en	None
967	2020-02-24 16:18:32	@SueRichter_Mann	[#MAGA]	[@aji_ley, @RaychelTania, @AOC]	en	None
966	2020-02-24 16:18:35	@Jewonemein	[]	[@justicedems, @julito77, @BernieSanders, @Mar...	en	None
965	2020-02-24 16:18:36	@ilana_esther_	[]	[@abesilbe, @jalsayyed]	en	None
964	2020-02-24 16:18:37	@KingBroly	[]	[@julie_kelly2]	en	None
963	2020-02-24 16:18:39	@jennife49899002	[]	[@deplorable_chet, @DrShayPhD, @cavaliereinesi...	en	None
962	2020-02-24 16:18:42	@dresswhisperer	[]	[@HoneBrandon, @citizengatsby, @lacadri34]	en	None
961	2020-02-24 16:18:42	@drpot89	[]	[@diegoro34033614, @SalsanRio, @DearAuntCrabby...	en	None
960	2020-02-24 16:18:44	@danolson1962	[#AOC, #RashidaTlaib, #IlhanOmar]	[@RealJamesWoods]	en	None
959	2020-02-24 16:18:45	@NikhilP72108560	[]	[@SeanMcElwee, @AOC, @DataProgress]	en	None

	date	sender	hashtags_list	mentions_list	lang	place
958	2020-02-24 16:18:45	@wonderchosen121	[]	[]	en	None
957	2020-02-24 16:18:45	@DireMakerBand	[#MAHA, #Bernie2020, #Sema2020]	[@Mokum_Misfit, @AllCps, @LadyReverbs, @lmorih...	en	None
956	2020-02-24 16:18:46	@blueskydriving	[]	[@CabbagetownMatt, @steph12581, @AOC]	en	None
955	2020-02-24 16:18:51	@Ms1Scs	[#AOC, #AOCStillAMoron, #OccasionalCortex]	[]	en	None
954	2020-02-24 16:18:51	@ckaprolet	[]	[@LouisAMarks56, @NY1, @AOC, @AOC]	en	None
953	2020-02-24 16:18:54	@Z71199869	[]	[@Arjiunastream, @AOC]	en	None
952	2020-02-24 16:18:59	@skytopranch	[]	[@julie_kelly2]	en	None
951	2020-02-24 16:18:59	@SueRichter_Mann	[]	[@Chris_1791, @AOC]	en	None
950	2020-02-24 16:18:59	@jackarm65081193	[]	[@AOC]	en	None

1. Counting Tweets, Tweeple, Hashtags, Mentions, Languages and Places

In [4]:

```
def flis(list):
    return [i for sl in list for i in sl]
```

In [5]:

```
print(len(df), "tweets")
senders=df["sender"].tolist()
usenders=set(senders)
print(len(usenders), "unique senders-tweeple")
hashtags=[df["hashtags_list"].tolist()[i] for i in range(len(df))]
hashtags=flis(hashtags)
uhashtags=set(hashtags)
print (len(uhashtags), "unique hashtags in tweets")
mentions=[df["mentions_list"].tolist()[i] for i in range(len(df))]
mentions=flis(mentions)
umentions=set(mentions)
print(len(umentions), "unique mentioned-tweeple in tweets")
languages=[df["lang"].tolist()[i] for i in range(len(df))]
ulanguages=set(languages)
print(len(ulanguages), "unique languages in tweets")
places=[df["place"].tolist()[i] for i in range(len(df))]
places=[p for p in places if type(p)==str]
uplaces=set(places)
print(len(uplaces), "unique places in tweets")
```

```
1000 tweets
847 unique senders-tweeple
200 unique hashtags in tweets
913 unique mentioned-tweeple in tweets
11 unique languages in tweets
1 unique places in tweets
```

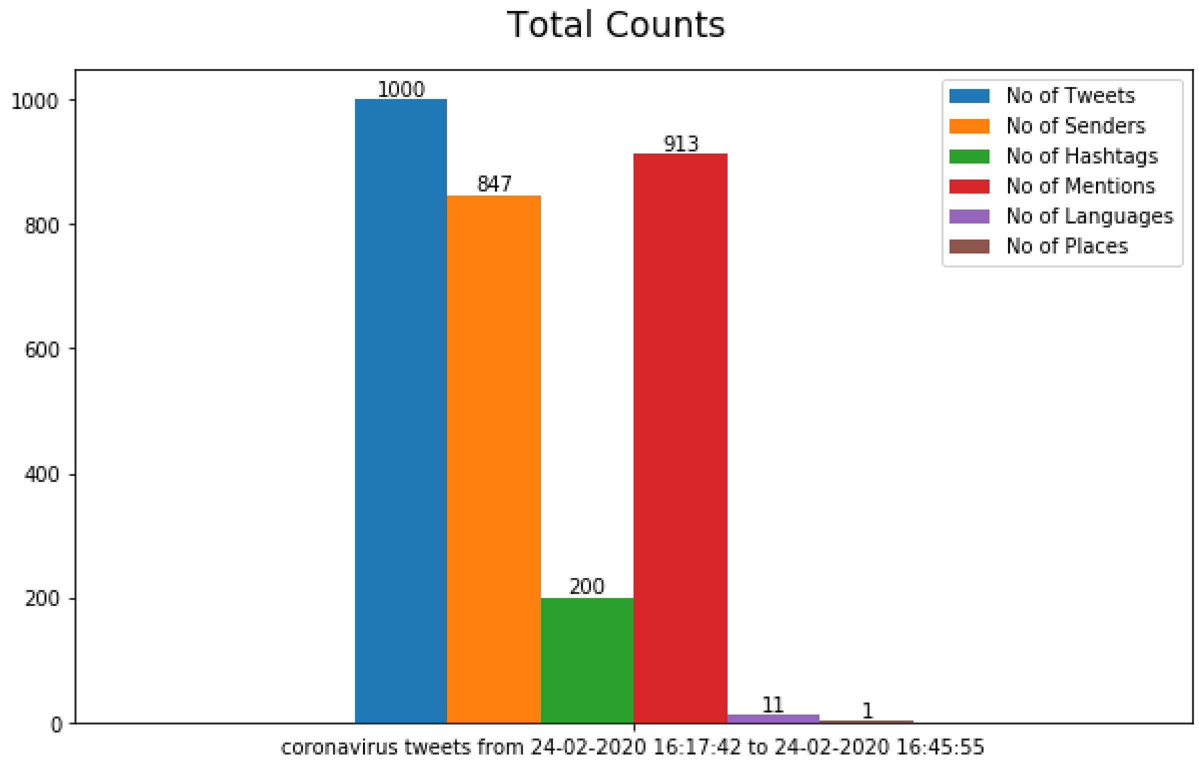
In [6]:

```
dd={}
dd["all_tweets"]=[len(df),len(usenders),len(uhashtags),len(umentions),len(u
ddf = pd.DataFrame.from_dict(dd, orient='index').reset_index()
ddf.rename(columns={'index': 'all_tweets', 0: 'tweets',1:"senders",2:"hasht
ddf
```

Out[6]:

	all_tweets	tweets	senders	hashtags	mentions	4	languages
0	all_tweets	1000	847	200	913	11	1

```
In [7]: ax=ddf.plot.bar(figsize=(10,6),rot=0);
        ax.legend(["No of Tweets", "No of Senders","No of Hashtags","No of Mentions
        labels=[search_term+" tweets from "+mind+" to "+maxd]
        ax.set_xticklabels(labels, rotation=0);
        for p in ax.patches:
            ax.annotate("%i" % p.get_height(), (p.get_x() + p.get_width() / 2., p.g
        plt.suptitle('Total Counts', x=0.5, y=0.95, ha='center', fontsize='xx-large
```



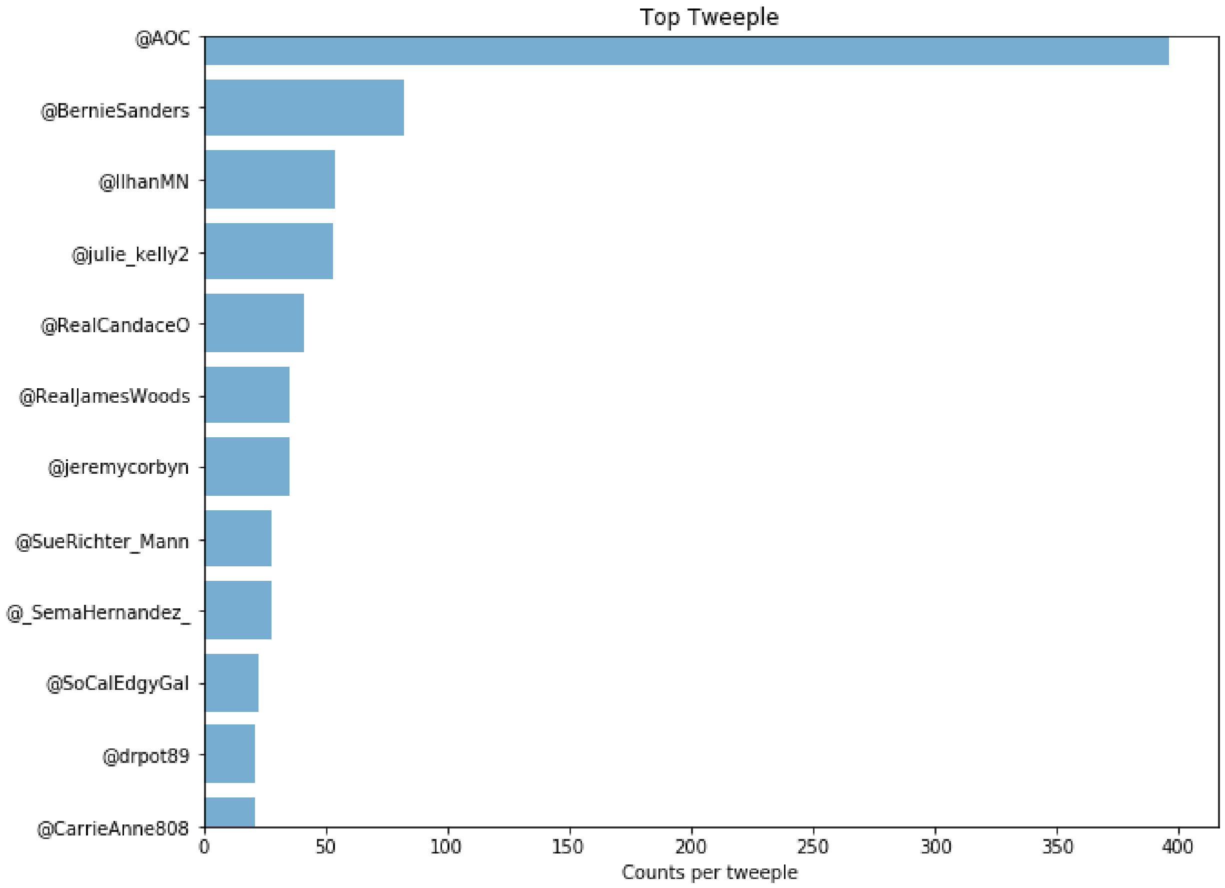
In [8]:

```
tweeple=senders+mentions
for s in senders:
    tweeple.append(s)
x=Counter(tweeple)
x=x.most_common()
print(len(x),"unique tweeple")
x
```

```
1663 unique tweeple
```

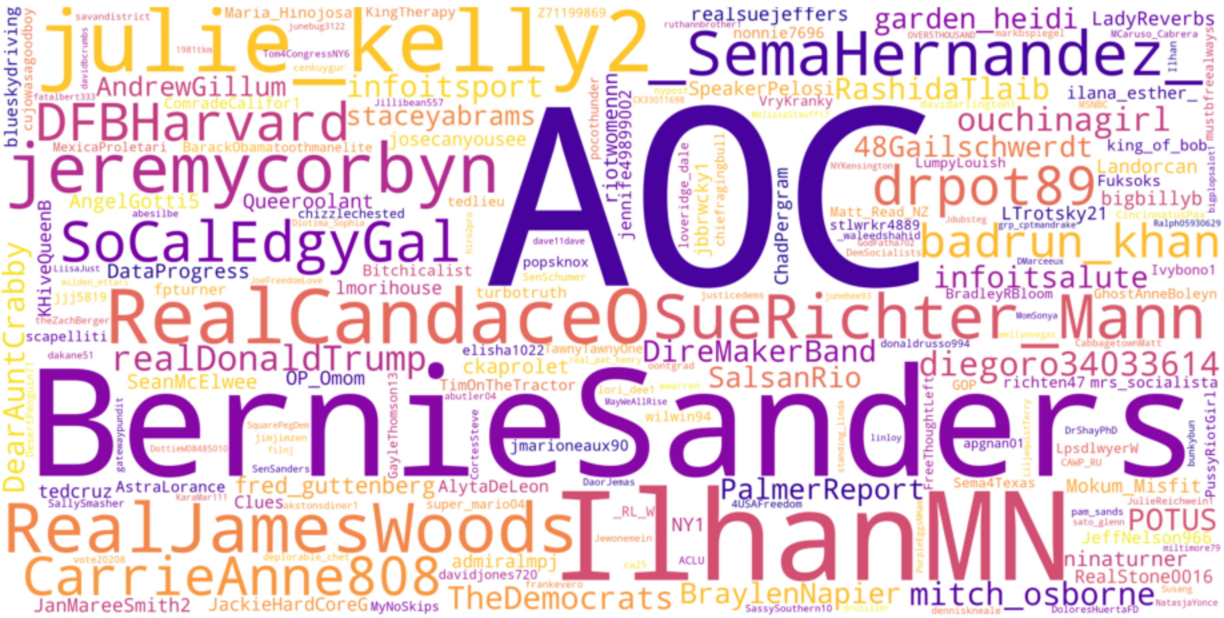
In [9]:

```
keys = [i for (i,j) in x if j>20]
y_pos = np.arange(len(keys))
performance = [j for (i,j) in x if j>20]
plt.figure(figsize=(10,8))
ax = plt.axes()
plt.barh(y_pos, performance, align='center', alpha=0.6)
ax.invert_yaxis()
plt.yticks(y_pos, keys)
plt.xlabel('Counts per tweeple')
plt.title('Top Tweeple')
plt.show()
```



In [10]:
```
t=[]
for (i,j) in x:
    for k in range(j):
#        print(i.replace(" ","_").replace("-","_"))
        t.append(i.replace(" ","_").replace("-","_"))
ttd=' '.join(t)
wordcloud = WordCloud(collocations=False,background_color="white",colormap=
fig = plt.figure(figsize=(13,13))
default_colors = wordcloud.to_array()
plt.imshow(default_colors, interpolation="bilinear")
plt.axis("off")
ss="Tweeple in the '%s' tweets" %search_term
plt.suptitle(ss,fontsize=25)
plt.tight_layout(rect=[0, 0, 1, 1.4])
plt.show()
```

Tweeple in the 'coronavirus' tweets

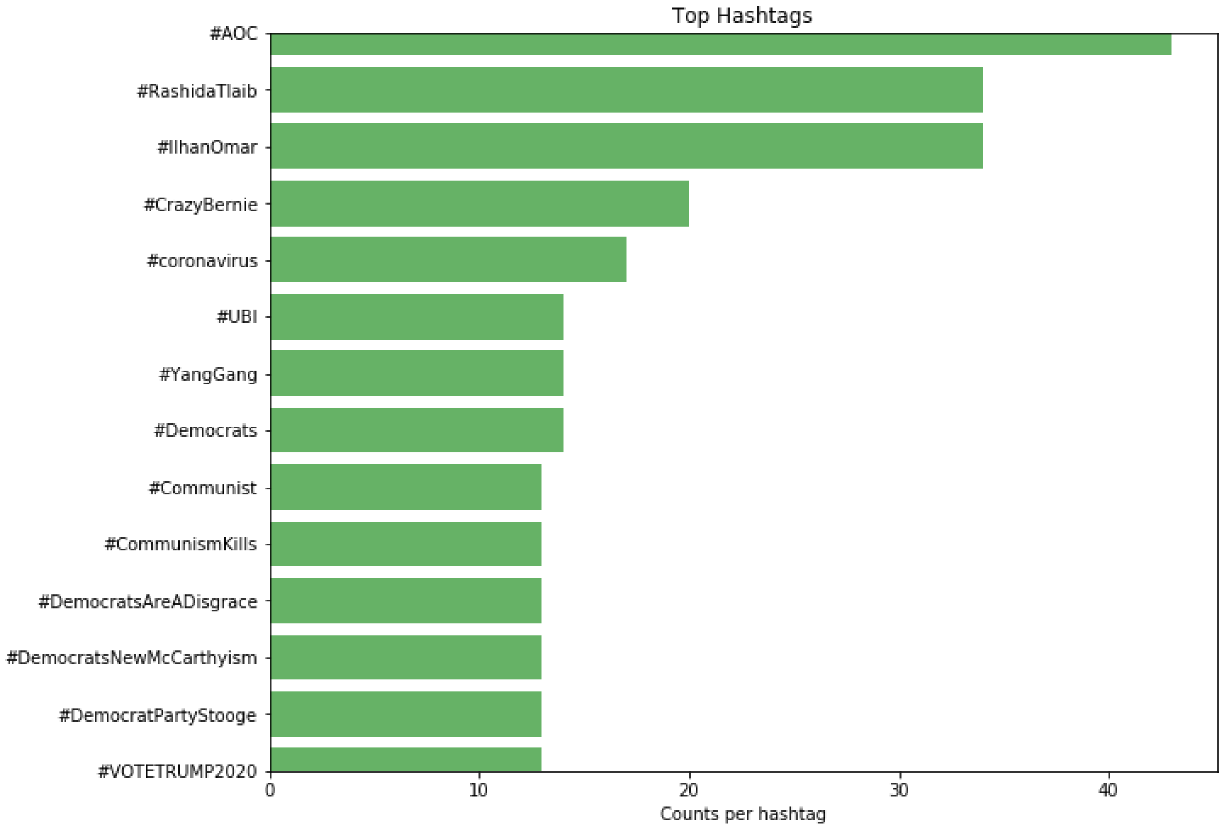


In [11]:

```
y=Counter(hashtags)
y=y.most_common()
print(len(y),"unique hashtags")
y
```

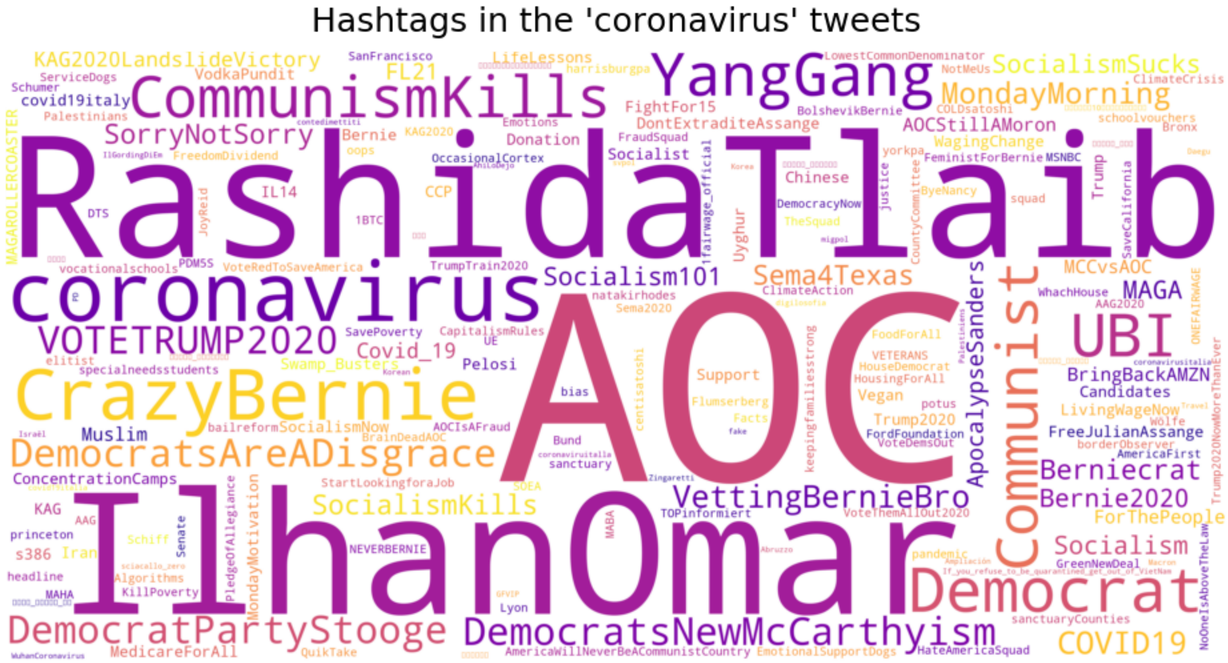
```
200 unique hashtags
```

In [12]:
```
keys = [i for (i,j) in y if j>10]
y_pos = np.arange(len(keys))
performance = [j for (i,j) in y if j>10]
plt.figure(figsize=(10,8))
ax = plt.axes()
plt.barh(y_pos, performance, align='center',color='green', alpha=0.6)
ax.invert_yaxis()
plt.yticks(y_pos, keys)
plt.xlabel('Counts per hashtag')
plt.title('Top Hashtags')
plt.show()
```



In [13]:

```
t=[]
for (i,j) in y:
    for k in range(j):
#         print(i.replace(" ","_").replace("-","_"))
        t.append(i.replace(" ","_").replace("-","_"))
ttd=' '.join(t)
wordcloud = WordCloud(collocations=False,background_color="white",colormap=
fig = plt.figure(figsize=(13,13))
default_colors = wordcloud.to_array()
plt.imshow(default_colors, interpolation="bilinear")
plt.axis("off")
ss="Hashtags in the '%s' tweets" %search_term
plt.suptitle(ss,fontsize=25)
plt.tight_layout(rect=[0, 0, 1, 1.4])
plt.show()
```



In [14]:

```
z=Counter(languages)
z=z.most_common()
print(len(z),"unique languages")
z
```

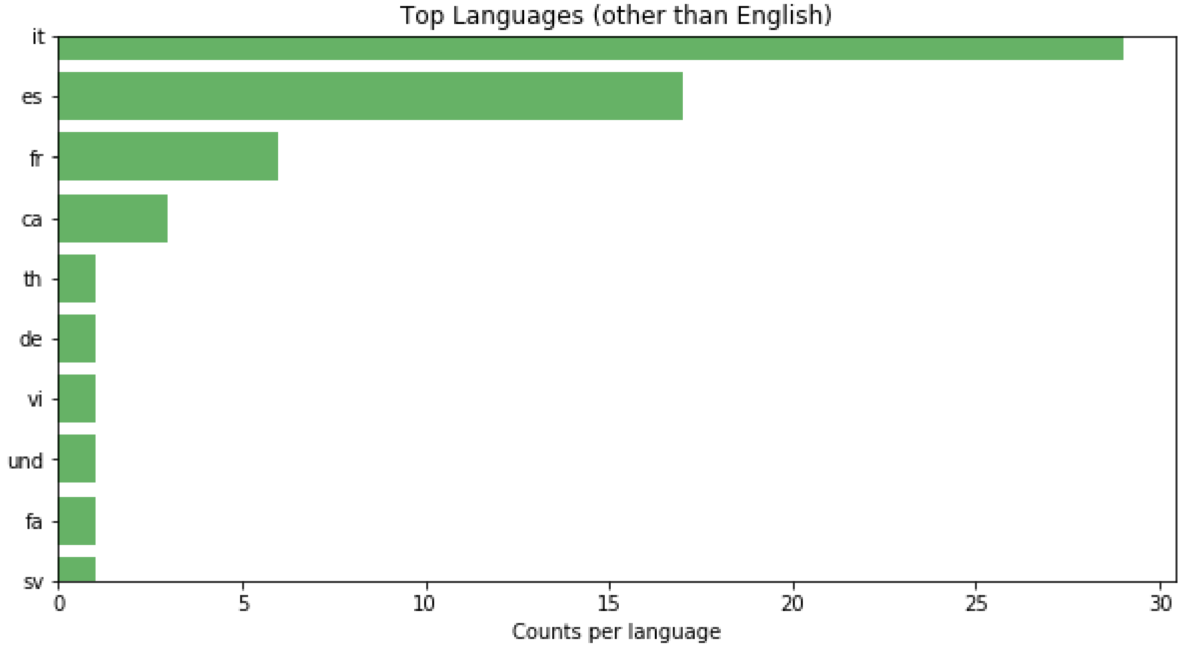
```
11 unique languages
```

Out[14]:

```
[('en', 939),
 ('it', 29),
 ('es', 17),
 ('fr', 6),
 ('ca', 3),
 ('th', 1),
 ('de', 1),
 ('vi', 1),
 ('und', 1),
 ('fa', 1),
 ('sv', 1)]
```

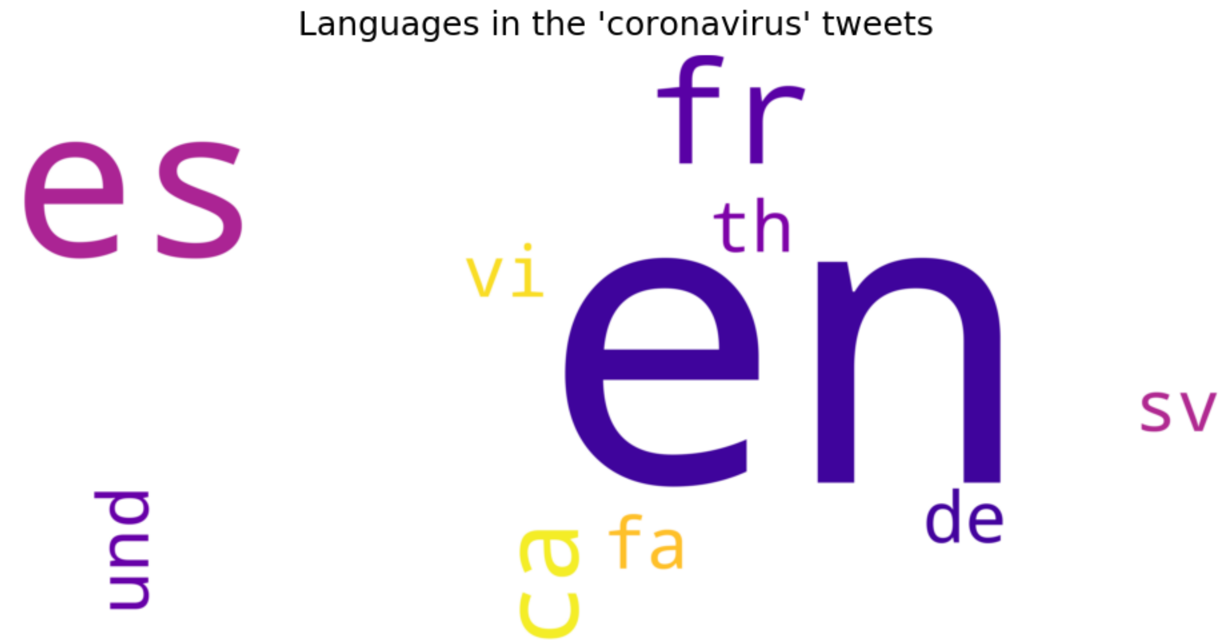
In [15]:

```
keys = [i for (i,j) in z if i!="en"]
y_pos = np.arange(len(keys))
performance = [j for (i,j) in z if i!="en"]
plt.figure(figsize=(10,5))
ax = plt.axes()
plt.barh(y_pos, performance, align='center',color='green', alpha=0.6)
ax.invert_yaxis()
plt.yticks(y_pos, keys)
plt.xlabel('Counts per language')
plt.title('Top Languages (other than English)')
plt.show()
```



In [16]:

```
t=[]
for (i,j) in z:
    for k in range(j):
#         print(i.replace(" ","_").replace("-","_"))
        t.append(i.replace(" ","_").replace("-","_"))
ttd=' '.join(t)
wordcloud = WordCloud(collocations=False,background_color="white",colormap=
fig = plt.figure(figsize=(13,13))
default_colors = wordcloud.to_array()
plt.imshow(default_colors, interpolation="bilinear")
plt.axis("off")
ss="Languages in the '%s' tweets" %search_term
plt.suptitle(ss,fontsize=25)
plt.tight_layout(rect=[0, 0, 1, 1.4])
plt.show()
```



In [17]:

```
u=Counter(places)
u=u.most_common()
print(len(u),"unique places")
u
```

```
1 unique places
```

Out[17]:

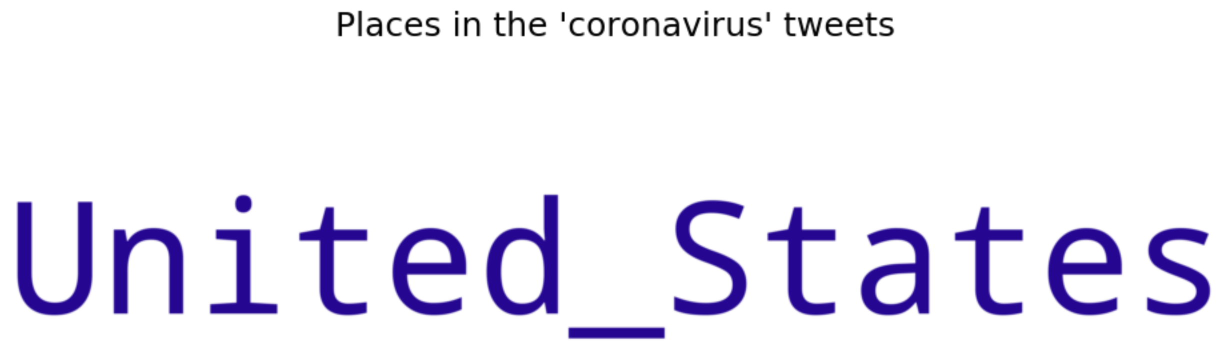
```
[('United States', 7)]
```

In [18]:

```
# keys = [i for (i,j) in u if i!="None"]
# y_pos = np.arange(len(keys))
# performance = [j for (i,j) in u if i!="None"]
# plt.figure(figsize=(10,8))
# ax = plt.axes()
# plt.barh(y_pos, performance, align='center',color='green', alpha=0.6)
# ax.invert_yaxis()
# plt.yticks(y_pos, keys)
# plt.xlabel('Counts per place')
# plt.title('Top Places (other than None)')
# plt.show()
```

In [19]:

```
t=[]
for (i,j) in u:
    for k in range(j):
        if type(i)==str:
#             print(i.replace(" ","_").replace("-","_"))
            t.append(i.replace(" ","_").replace("-","_"))
ttd=' '.join(t)
wordcloud = WordCloud(collocations=False,background_color="white",colormap=
fig = plt.figure(figsize=(13,13))
default_colors = wordcloud.to_array()
plt.imshow(default_colors, interpolation="bilinear")
plt.axis("off")
ss="Places in the '%s' tweets" %search_term
plt.suptitle(ss,fontsize=25)
plt.tight_layout(rect=[0, 0, 1, 1.4])
plt.show()
```



3. Graph of Co-Occurring Hashtags

In [20]:
```
heds=[]
for i in range(len(df)):
    iterable=df.iloc[i]['hashtags_list']
    if type(iterable)!=float:
        if len(iterable)>1:
            for j in itertools.combinations(iterable, 2):
                heds.append((j[0],j[1],df.iloc[i]['date']))
print("Number of hashtag-co-occurrences:")
print("%i multiple (%i unique)" %(len(heds),len(set(heds))))
```

```
Number of hashtag-co-occurrences:
819 multiple (814 unique)
```

In [21]:
```
G=nx.MultiGraph()
for k,v in dict(Counter(heds)).items():
    G.add_edge(k[0],k[1],date=k[2])

weight={(x,y):v for (x, y), v in Counter(G.edges()).items()}
w_edges=[(x,y,z) for (x,y),z in weight.items()]
Gw = nx.Graph()
Gw.add_weighted_edges_from(w_edges)

print("The graph of co-occurrent hashtags of the '%s' tweets is a weighted
if nx.is_connected(Gw)==True:
    print ("This graph is a connected graph")
else:
    print ("This graph is a disconnected graph and it has",nx.number_connec
    giant = max(nx.connected_component_subgraphs(Gw), key=len)
    Gwlcc=Gw.subgraph(giant)
    print ("The largest connected component of this graph has %i nodes and
```

```
The graph of co-occurrent hashtags of the 'coronavirus' tweets is a weigh
ted graph and it has 170 nodes and 384 edges

This graph is a disconnected graph and it has 31 connected components
The largest connected component of this graph has 46 nodes and 139 edges
```

In [22]:

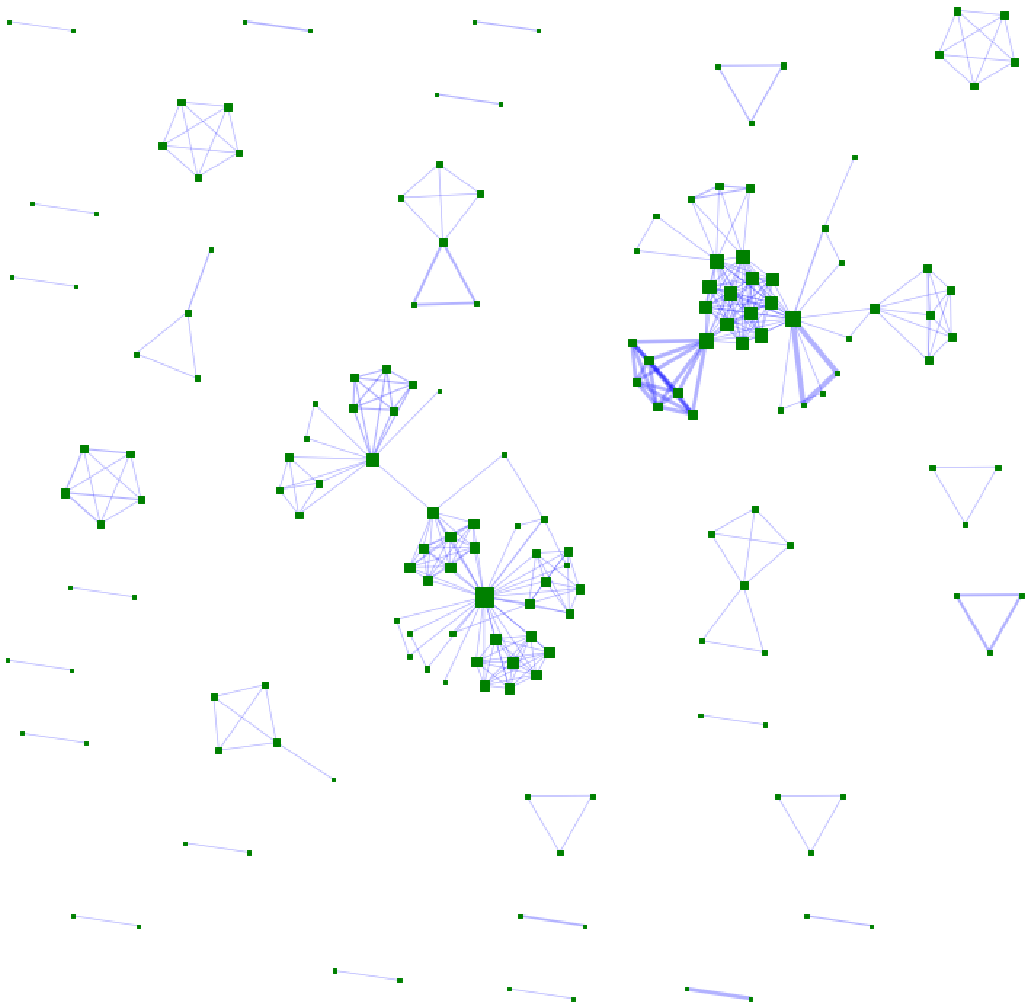
```
edge_width=[Gw[u][v]['weight'] for u,v in Gw.edges()]
edge_width=[math.log(1+w) for w in edge_width]
# cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in Gw.edges(data=True) ]
# edge_color=weight_list
# vmin = min(edge_color)
# vmax = max(edge_color)
# width_list=[2*math.log(2+w) for w in weight_list]
width_list=[1.5*math.log(abs(min(weight_list))+2+w) for w in weight_list] #
nsi=[5*Gw.degree(n) for n in Gw.nodes()]

figsize=(15,15)

pos=graphviz_layout(Gw)

plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(Gw, pos, node_color='g',node_shape="s",node_
nx.draw_networkx_edges(Gw, pos, edge_color='b',width=edge_width,alpha=0.3)
nol={}
for n in Gw.nodes():
    nol[n]=""
nx.draw_networkx_labels(Gw, pos,labels=nol)
plt.axis('off');
yoffset = {}
y_off = -5 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
# nx.draw_networkx_labels(Gw, yoffset,font_size=12);
# sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=
# sm.set_array([])
# cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
# cbar.set_label('Average sentiment of sentences')
sst="The graph of co-occurrent hashtags of the '%s' tweets" %search_term
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```

The graph of co-occurrent hashtags of the 'coronavirus' tweets



In [23]:

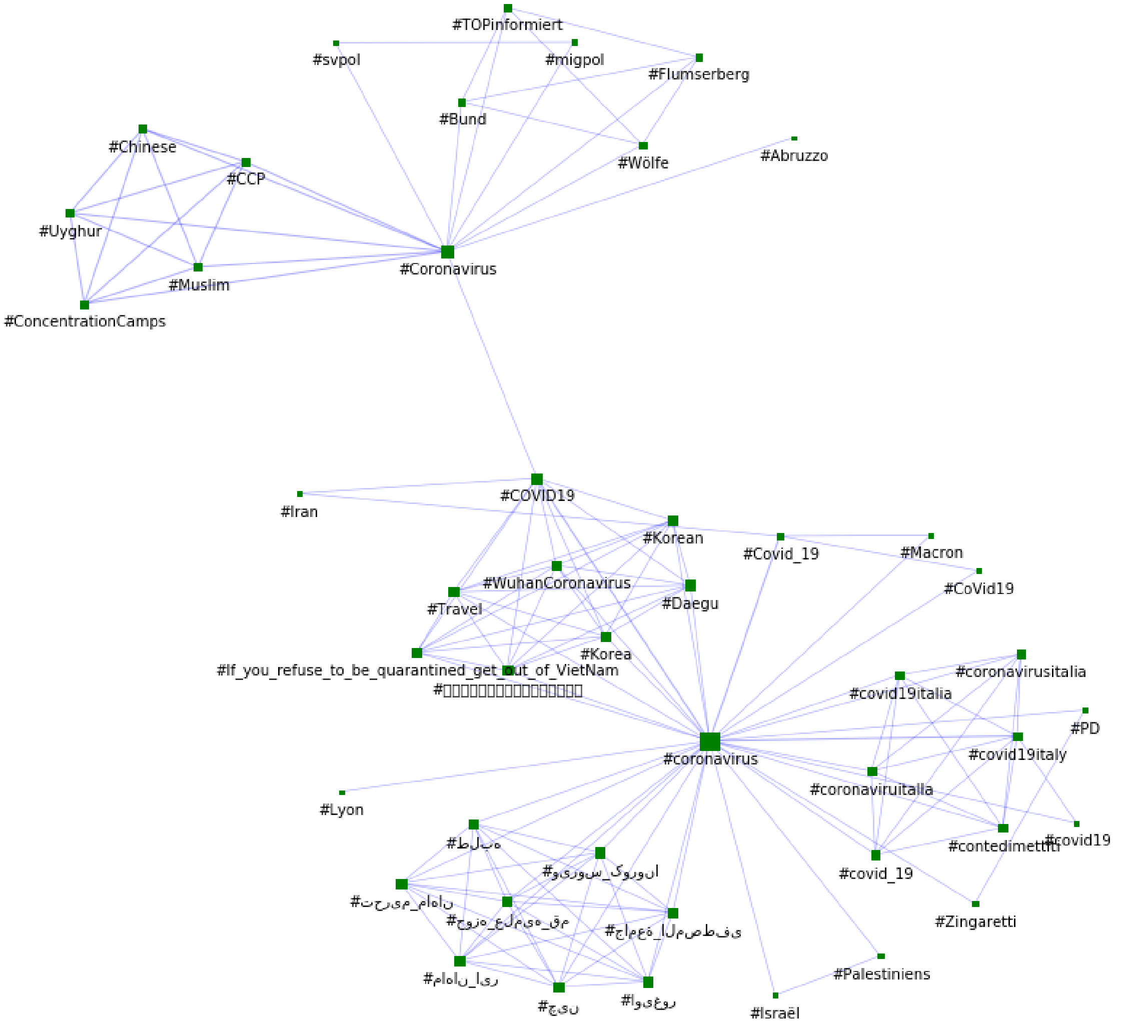
```
edge_width=[Gwlcc[u][v]['weight'] for u,v in Gwlcc.edges()]
edge_width=[math.log(1+w) for w in edge_width]
# cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in Gwlcc.edges(data=True) ]
# edge_color=weight_list
# vmin = min(edge_color)
# vmax = max(edge_color)
# width_list=[2*math.log(2+w) for w in weight_list]
width_list=[1.5*math.log(abs(min(weight_list))+2+w) for w in weight_list] #
nsi=[5*Gwlcc.degree(n) for n in Gwlcc.nodes()]

figsize=(15,15)

pos=graphviz_layout(Gwlcc)

plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(Gwlcc, pos, node_color='g',node_shape="s",no
nx.draw_networkx_edges(Gwlcc, pos, edge_color='b',width=edge_width,alpha=0.
plt.axis('off');
yoffset = {}
y_off = -7 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
nx.draw_networkx_labels(Gwlcc, yoffset,font_size=10);
# sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=
# sm.set_array([])
# cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
# cbar.set_label('Average sentiment of sentences')
sst="The largest connected component of the \n graph of co-occurrent hashta
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```

The largest connected component of the
graph of co-occurrent hashtags of the 'coronavirus' tweets



In [24]:

```

# plt.figure(figsize=figsize);
# nodes = nx.draw_networkx_nodes(Gwlcc, pos, node_color='g',node_shape="s",
# nx.draw_networkx_edges(Gwlcc, pos, edge_color='b',width=edge_width,alpha=
# plt.axis('off');
# yoffset = {}
# y_off = -7 # offset on the y axis
# for k, v in pos.items():
#     yoffset[k] = (v[0], v[1]+y_off)
# nx.draw_networkx_labels(Gwlcc, yoffset,font_size=10);
# # sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vma
# # sm.set_array([])
# # cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.0
# # cbar.set_label('Average sentiment of sentences')
# sst="The largest connected component of the \n graph of co-occurrent hash
# plt.title(sst,fontsize=15);
# plt.margins(x=0.1, y=0.1)
```

4. Graph of Mention-ing/-ed Tweeple

```
In [25]: meds=[]
         for i in range(len(df)):
             iterable=df.iloc[i]['mentions_list']
             if type(iterable)!=float:
                 for k in iterable:
                     meds.append((df.iloc[i]['sender'],k,df.iloc[i]['date']))
         print("Number of mentions among tweeple:")
         print("%i multiple (%i unique)" %(len(meds),len(set(meds))))
```

```
Number of mentions among tweeple:
2722 multiple (2707 unique)
```

```
In [26]: mG=nx.MultiDiGraph()
         for k,v in dict(Counter(meds)).items():
             mG.add_edge(k[0],k[1],date=k[2])

         weight={(x,y):v for (x, y), v in Counter(mG.edges()).items()}
         w_edges=[(x,y,z) for (x,y),z in weight.items()]
         mGw = nx.DiGraph()
         mGw.add_weighted_edges_from(w_edges)

         print("The graph of mention-ing/-ed tweeple of the '%s' tweets is a weighte
         if nx.is_weakly_connected(mGw)==True:
             print ("This graph is a weakly connected graph")
         else:
             print ("This graph is a weakly disconnected graph and it has",nx.number
             giant = max(nx.weakly_connected_component_subgraphs(mGw), key=len)
             mGwlcc=mGw.subgraph(giant)
             print ("The largest weakly connected component of this graph has %i nod
```

```
The graph of mention-ing/-ed tweeple of the 'coronavirus' tweets is a wei
ghted digraph and it has 1609 nodes and 2183 edges

This graph is a weakly disconnected graph and it has 180 weakly connected
components
The largest weakly connected component of this graph has 1061 nodes and 1
766 edges
```

In [27]:

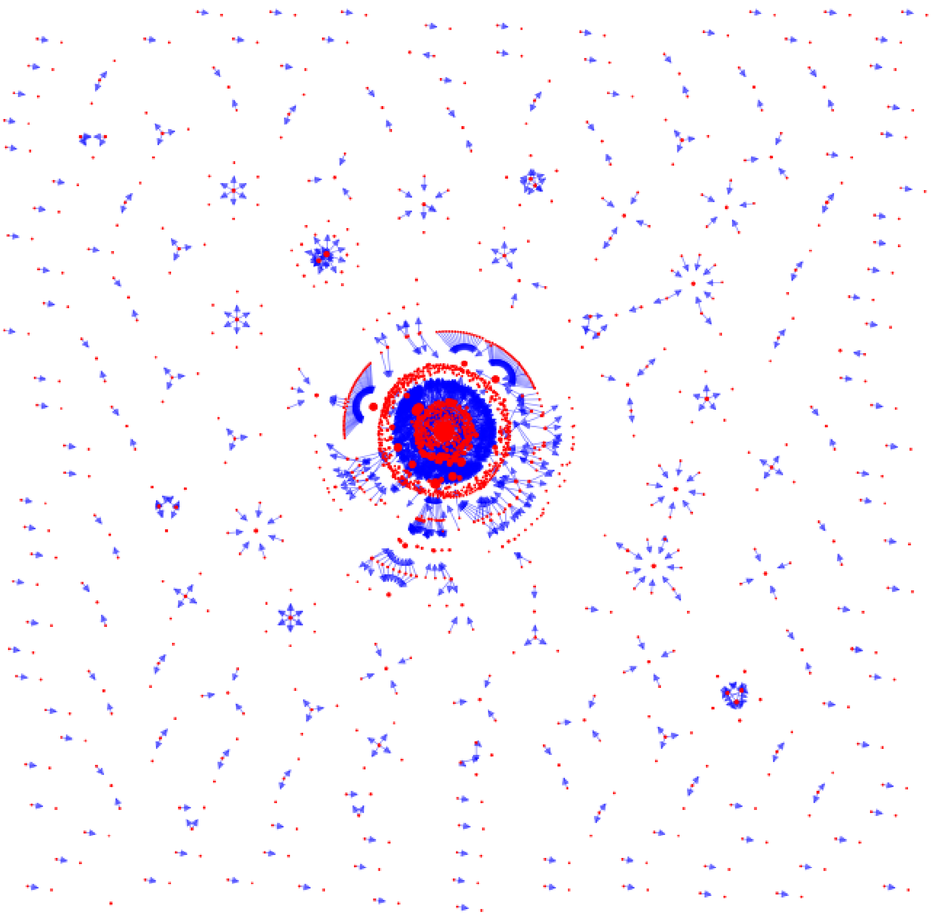
```
edge_width=[mGw[u][v]['weight'] for u,v in mGw.edges()]
edge_width=[math.log(1+w) for w in edge_width]
# cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in mGw.edges(data=True) ]
# edge_color=weight_list
# vmin = min(edge_color)
# vmax = max(edge_color)
# width_list=[2*math.log(2+w) for w in weight_list]
width_list=[1.5*math.log(abs(min(weight_list))+2+w) for w in weight_list] #
nsi=[0.6*mGw.degree(n) for n in mGw.nodes()]

figsize=(15,15)

pos=graphviz_layout(mGw)

plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(mGw, pos, node_color='r',node_size=nsi)
nx.draw_networkx_edges(mGw, pos, edge_color='b',width=edge_width,alpha=0.5)
nol={}
for n in mGw.nodes():
    nol[n]=""
nx.draw_networkx_labels(mGw, pos,labels=nol)
plt.axis('off');
yoffset = {}
y_off = -5 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
# nx.draw_networkx_labels(Gw, yoffset,font_size=12);
# sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=
# sm.set_array([])
# cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
# cbar.set_label('Average sentiment of sentences')
sst="The graph of mention-ing/-ed tweeple of the '%s' tweets" %search_term
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```

The graph of mention-ing/-ed tweeple of the 'coronavirus' tweets



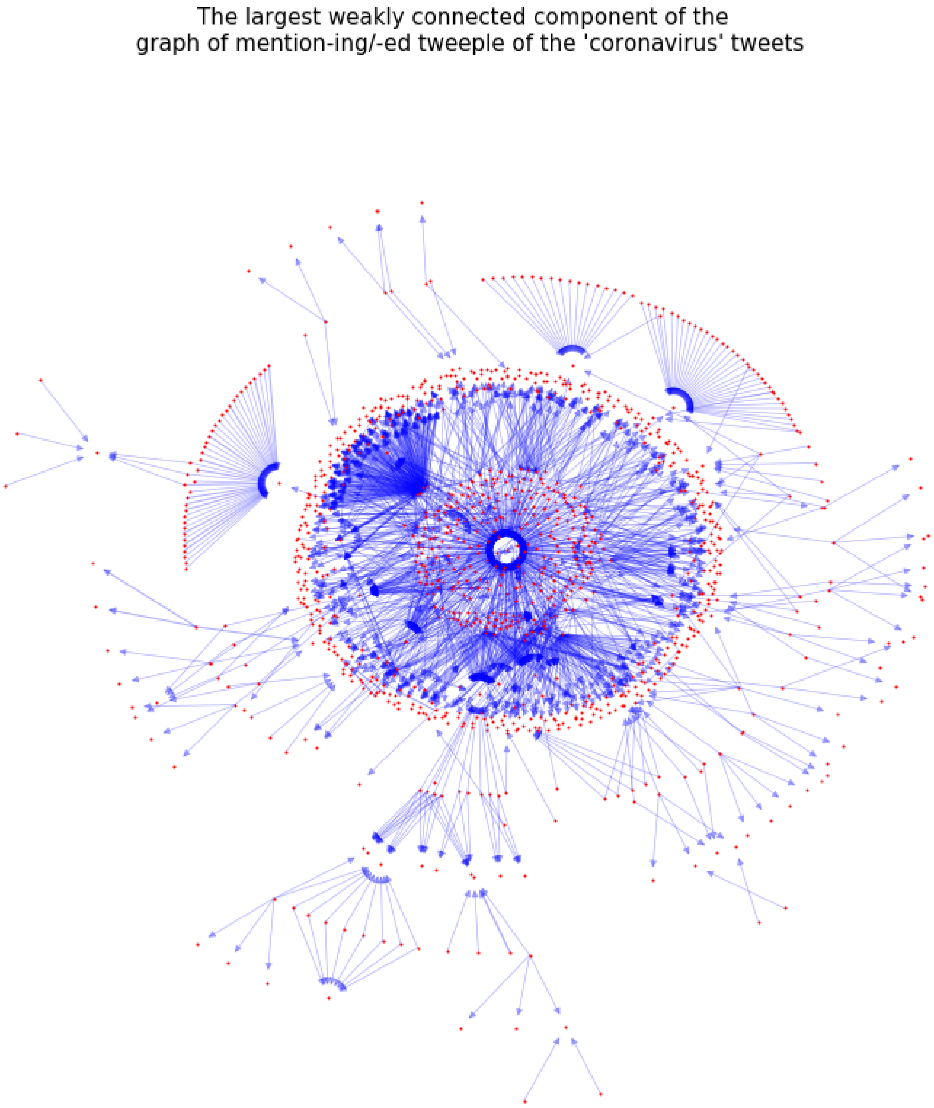
In [28]:

```
edge_width=[mGwlcc[u][v]['weight'] for u,v in mGwlcc.edges()]
edge_width=[math.log(1+w) for w in edge_width]
# cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in mGwlcc.edges(data=True) ]
# edge_color=weight_list
# vmin = min(edge_color)
# vmax = max(edge_color)
width_list=[0.5*math.log(1+w) for w in weight_list]
# width_list=[0.5*math.log(abs(min(weight_list))+w) for w in weight_list] #
nsi=[mGwlcc.degree(n) for n in mGwlcc.nodes()]

figsize=(15,15)

pos=graphviz_layout(mGwlcc)

plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(mGwlcc, pos, node_color='r',node_size=nsi)
nx.draw_networkx_edges(mGwlcc, pos, edge_color='b',width=edge_width,alpha=0
nol={}
for n in mGwlcc.nodes():
    nol[n]=""
nx.draw_networkx_labels(mGwlcc, pos,labels=nol)
plt.axis('off');
yoffset = {}
y_off = -7 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
# nx.draw_networkx_labels(mGwlcc, yoffset,font_size=10);
# sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=
# sm.set_array([])
# cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
# cbar.set_label('Average sentiment of sentences')
sst="The largest weakly connected component of the \n graph of mention-ing/
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```


The largest weakly connected component of the
graph of mention-ing/-ed tweeple of the 'coronavirus' tweets

A Random Sample of the Graph of Mention-ing/-ed Tweeple

In [29]:

```
sample_size=100
sample_nodes=random.sample(mGwlcc.nodes(),sample_size)
RG=mGwlcc.subgraph(sample_nodes)

weight={(x,y):v for (x, y), v in Counter(RG.edges()).items()}
w_edges=[(x,y,z) for (x,y),z in weight.items()]
RGw = nx.DiGraph()
RGw.add_edges_from(w_edges)

print("The random sample subgraph of mention-ing/-ed tweeple of the '%s' tw
if nx.is_weakly_connected(RGw)==True:
    print ("This graph is a weakly connected graph")
else:
    print ("This graph is a weakly disconnected graph and it has",nx.number
    giant = max(nx.weakly_connected_component_subgraphs(RGw), key=len)
    RGwlcc=RGw.subgraph(giant)
    print ("The largest weakly connected component of this graph has %i nod
```

```
The random sample subgraph of mention-ing/-ed tweeple of the 'coronaviru
s' tweets is a weighted digraph and it has 27 nodes and 19 edges

This graph is a weakly disconnected graph and it has 9 weakly connected c
omponents
The largest weakly connected component of this graph has 7 nodes and 6 ed
ges
```

In [30]:

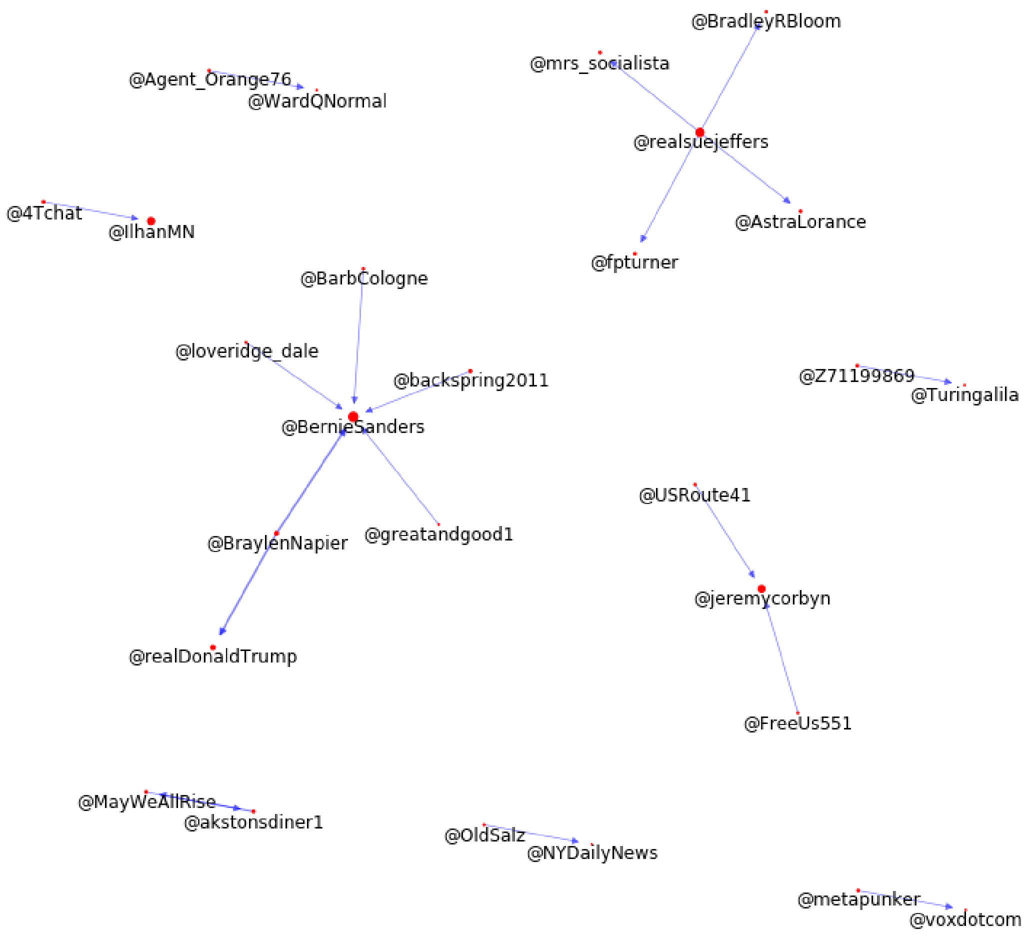
```
edge_width=[RGw[u][v]['weight'] for u,v in RGw.edges()]
edge_width=[math.log(1+w) for w in edge_width]
# cmap=plt.cm.cool
weight_list = [ e[2]['weight'] for e in RGw.edges(data=True) ]
# edge_color=weight_list
# vmin = min(edge_color)
# vmax = max(edge_color)
# width_list=[2*math.log(2+w) for w in weight_list]
width_list=[1.5*math.log(abs(min(weight_list))+2+w) for w in weight_list] #
nsi=[0.6*mGw.degree(n) for n in RGw.nodes()]

figsize=(15,15)

pos=graphviz_layout(RGw)

plt.figure(figsize=figsize);
nodes = nx.draw_networkx_nodes(RGw, pos, node_color='r',node_size=nsi)
nx.draw_networkx_edges(RGw, pos, edge_color='b',width=edge_width,alpha=0.5)
# nol={}
# for n in RGw.nodes():
#     nol[n]=""
# nx.draw_networkx_labels(RGw, pos,labels=nol)
plt.axis('off');
yoffset = {}
y_off = -5 # offset on the y axis
for k, v in pos.items():
    yoffset[k] = (v[0], v[1]+y_off)
nx.draw_networkx_labels(RGw, yoffset,font_size=12);
# sm = plt.cm.ScalarMappable(cmap=cmap, norm=plt.Normalize(vmin=vmin, vmax=
# sm.set_array([])
# cbar = plt.colorbar(sm, orientation='horizontal', shrink=0.7, pad = 0.02)
# cbar.set_label('Average sentiment of sentences')
sst="The random sample subgraph of mention-ing/-ed tweeple of the '%s' twee
plt.title(sst,fontsize=15);
plt.margins(x=0.1, y=0.1)
```

The random sample subgraph of mention-ing/-ed tweeple of the 'coronavirus' tweets


@BradleyRBloom
@mrs_socialista
@Agent_Orange76
@WardQNormal
@realsuejeffers
@4Tchat
@IlhanMN
@AstraLorance
@fpturner
@BarbCologne
@loveridge_dale
@backspring2011
@Z71199869
@Turingalila
@BernieSanders
@USRoute41
@BraylenNapier
@greatandgood1
@jeremycorbyn
@realDonaldTrump
@FreeUs551
@MayWeAllRise
@akstonsdiner1
@OldSalz
@NYDailyNews
@metapunker
@voxdotcom