bot-tweet-entities

December 5, 2016

1 Bot Entities

(at long last)

Import our libraries needed for the data handling.

```
In [1]: import pandas as pd
        import numpy as np
        import json
        import glob
```

```
In [2]: #Set PANDAS to show all columns in DataFrame
        pd.set_option('display.max_columns', None)
```

Libraries for stupid text encoding

```
In [3]: from urllib2 import quote
        # Unicode strings
        from __future__ import unicode_literals
```

Import libraries needed for visualization.

```
In [4]: import matplotlib
        import matplotlib.pyplot as plt
        # Within notebook viewing
        %matplotlib inline

        print (matplotlib.__version__)
```

```
1.5.3
```

```
In [5]: # Import for axes, color, etc
        from pylab import *
```

1.0.1 Directories

```
In [6]: testDir = '../../data/external/trump-bots/'
        botDir = '../../data/external/botresults/'
        outDir = '../../data/processed/bot-tweets/'
```

Read in the data files by combining the extracted files.

```
In [7]: # Crudely combine
        process = []
        for f in glob.glob((botDir + "*.txt")):
            with open(f, "rb") as infile:
                for line in infile:
                    process.append(json.loads(line))
        raw = pd.DataFrame.from_records(process)

        # save memory
        del process

        print ("(# tweets, # columns): {}".format(raw.shape))
```

```
(# tweets, # columns): (77722, 33)
```

```
In [8]: # Helper functions
        def countEntities(df, col):
            '''
            Function to aggregate entities from tweets.
            Returns a dataframe with value counts
            '''
            # Hold the entities
            ent_list = []

            # assign the appropriate key to get info
            cols = {
                'hashtags': 'text',
                'urls': 'expanded_url',
                'user_mentions': 'screen_name'
            }

            def iterEnt(l):
                '''
                Function to keep your sanity. Checks if the entry is null,
                adds values to our list if it isn't.
                '''
                if (l is not None):
                    for ent in l:
                        try:
                            val = cols[col]
```

```
                        if ent[val] is not None:
                            escaped = ent[val].lower().encode('utf-8')
                            ent_list.append(escaped)
                    except:
                        print (ent)

        df.apply(lambda x: iterEnt(x[col]), axis=1)
        # Create a dataframe, and then aggregate
        counts = pd.DataFrame(pd.Series(ent_list).value_counts())
        counts.reset_index(drop=False, inplace=True)

        # Give it a column
        counts.columns = [col, 'frequency']
        return counts

    def countHashtags(tweet):
        try:
            if tweet['hashtags'] is not None:
                return len(tweet['hashtags'])
        except:
            return 0
```

Extract only the entities. This contains a tweet's hashtags (#'s), *URLs*, and user mentions (@'s).

```
In [9]: # content from tweets
        entities = pd.DataFrame.from_records(raw['entities'])
        # Calculate the number of hashtags in a tweet
        entities['hashtag_count'] = entities.apply(lambda x: countHashtags(x), axis

        print (entities['hashtags'].iloc[3])
        entities.head()
```

```
[{u'indices': [0, 13], u'text': u'ModiMinistry'}]
```

Out[9]:

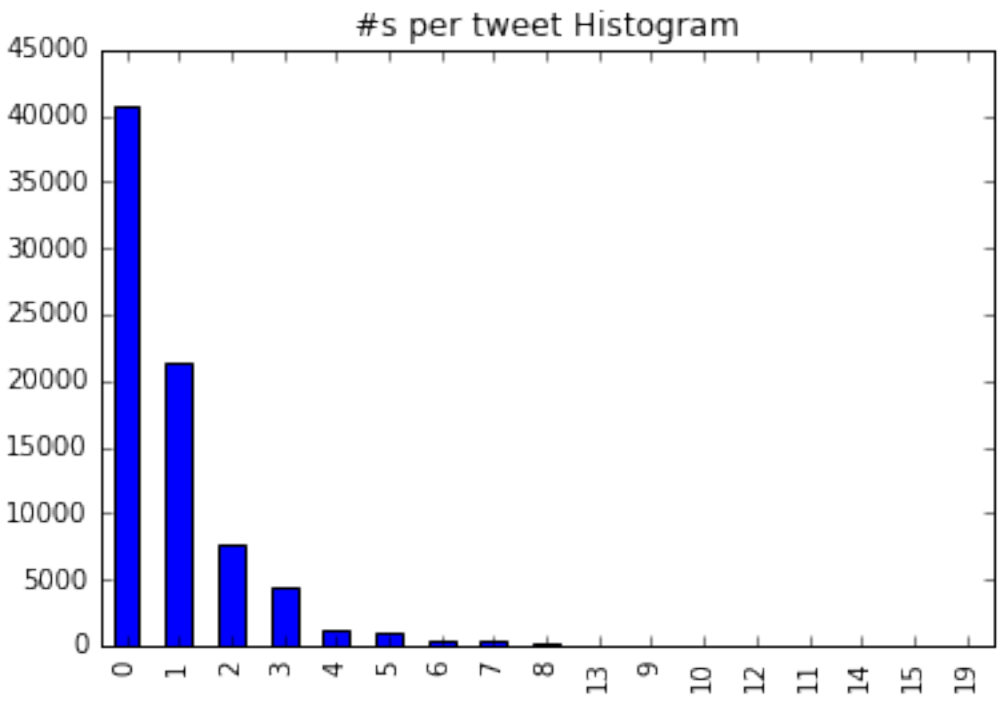
	hashtags	media	symbols	trends	urls	user_mentions
0	[]	NaN	[]	[]	[{u'url': u'http://t.co/uvF94Se7N1', u'indices...	[]
1	[]	NaN	[]	[]	[{u'url': u'http://t.co/BXCXee3Ra1', u'indices...	[]
2	[]	NaN	[]	[]	[{u'url': u'http://t.co/wfUA1uJWOh', u'indices...	[]
3	[{u'indices': [0, 13], u'text': u'ModiMinistry'}]	NaN	[]	[]	[{u'url': u'http://t.co/MyrrJb1Ex8', u'indices...	[]
4	[]	NaN	[]	[]	[{u'url': u'http://t.co/XF3TWcmQEl', u'indices...	[]

```
   hashtag_count
0              0
1              0
2              0
3              1
4              0
```

1.0.2 Hashtags

What are the most common hashtags? Summary statistics below.

```
In [10]: # Overall bot tweets
         print ('Descriptive stats for hashtag counts')
         print (entities['hashtag_count'].describe())
         print ('\n---\n...for tweets with at least 1 hashtag')
         print (entities[entities['hashtag_count'] > 0].describe())

Descriptive stats for hashtag counts
count    77722.000000
mean         0.899050
std          1.430082
min          0.000000
25%          0.000000
50%          0.000000
75%          1.000000
max         19.000000
Name: hashtag_count, dtype: float64

---
...for tweets with at least 1 hashtag
       hashtag_count
count   36897.000000
mean        1.893813
std         1.556955
min         1.000000
25%         1.000000
50%         1.000000
75%         2.000000
max        19.000000


In [11]: # #'s per tweet distribution
         entities['hashtag_count'].value_counts().plot(kind='bar', title='#s per tw

Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01e0041790>
```



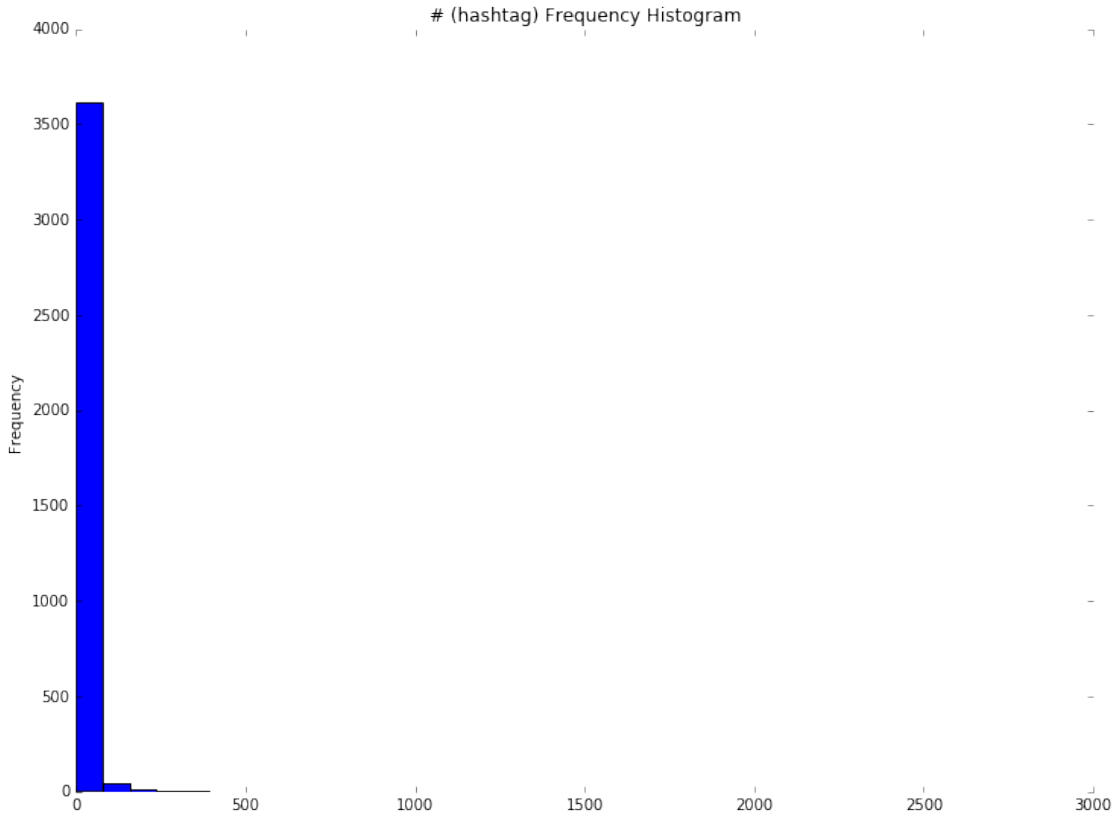

```
In [12]: hashtags = countEntities(entities, 'hashtags')
         print (hashtags.head(50))
```

	hashtags	frequency
0	trump2016	7867
1	trump	5280
2	clinton	5019
3	thenewsclub	4883
4	donaldtrump	3451
5	redstate	2609
6	boycottstarbucks	2495
7	trumptrain	2194
8	hillarytapes	1839
9	news	1400
10	microaggression	1362
11	ca	1108
12	realdonaldtrump	847
13	makeamericagreatagain	815
14	gop	733
15	pa	695
16	ny	693
17	modiministry	628
18	in	617
19	teamtrump	517

```
20                  tcot        506
21                 maga         403
22                   ct         379
23                world         377
24                   az         367
25                   md         350
26             politics         329
27            wisconsin         327
28                   de         326
29           republican         319
30                   ut         298
31                   ri         273
32                ccot          241
33                 cnn          240
34                  ne          235
35                 usa          231
36                  wv          216
37                 nba          210
38                 nyc          195
39           leadership         190
40              foxnews         175
41                   2a         173
42           soundcloud         172
43            nevercruz         167
44              newyork         165
45                pjnet         163
46                  nra         163
47               recent         156
48               donald         151
49         womenfortrump        151
```

In [13]:
```
# Distribution of frequencies

# Create a figure of given size
fig = plt.figure(figsize=(12,9))
# Add a subplot
ax = fig.add_subplot(111)
# Remove grid lines (dotted lines inside plot)
ax.grid(False)
# Remove plot frame
ax.set_frame_on(False)

# limit axis so we can see more
matplotlib.pylab.xlim([0, 3000])


hashtags['frequency'].plot(kind='hist', bins=100, title='# (hashtag) Frequ
```

Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01de747d10>



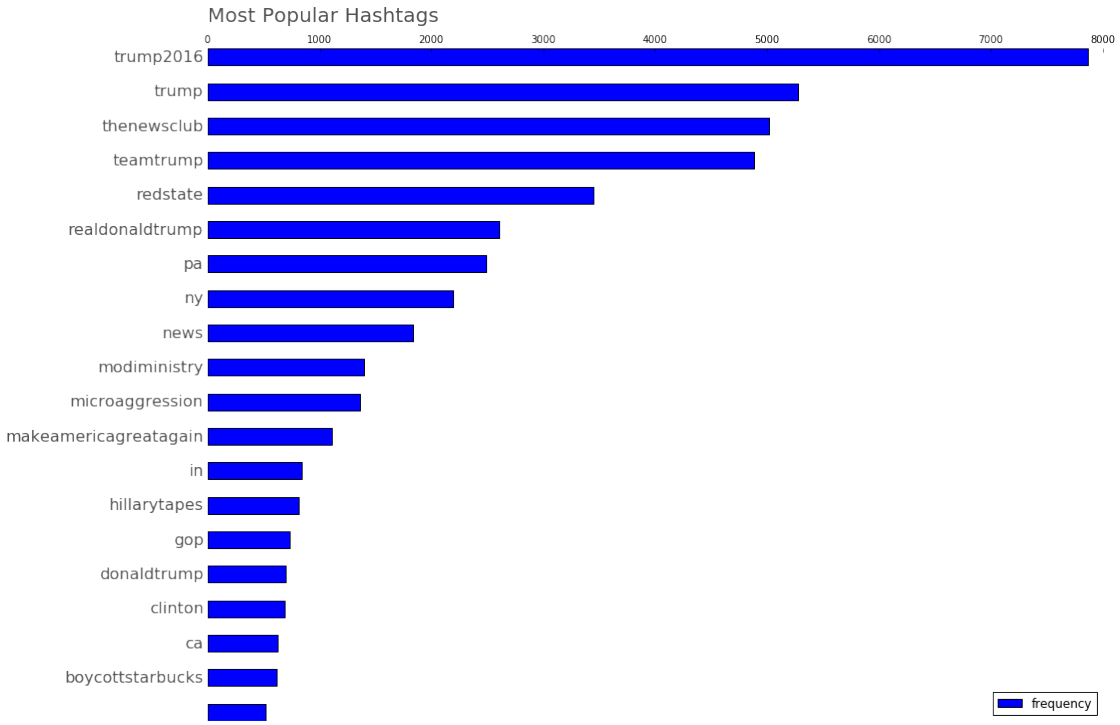
In [22]:
```
# Create a figure of given size
fig = plt.figure(figsize=(16,12))
# Add a subplot
ax = fig.add_subplot(111)
# Remove grid lines (dotted lines inside plot)
ax.grid(False)
# Remove plot frame
ax.set_frame_on(False)

# Set x axis label on top of plot, set label text
ax.xaxis.set_label_position('top')
ax.set_xlabel('Most Popular Hashtags', fontsize=20, alpha=0.7, ha='left')
# Position x tick labels on top
ax.xaxis.tick_top()
# A little above the axis
ax.xaxis.set_label_coords(0, 1.04)


def ord_to_char(v, p=None):
```

```
        return chr(int(v))

    ax.yaxis.set_major_formatter(FuncFormatter(ord_to_char))
    ax.yaxis.set_major_locator(MultipleLocator(1))

    # Remove tick lines in x and y axes
    ax.yaxis.set_ticks_position('none')


    # Y ticks
    # # for whatever reason 'trumptrain' raises:
    # # # ValueError: could not convert string to float: trumptrain
    hashtags['hashtags'].iloc[7] = u'trumptrain'

    yticks = [i.encode('unicode-escape') for i in hashtags['hashtags'].head(20
    ax.yaxis.set_ticks([x + 1 for x in range(21)])
    ax.set_yticklabels(yticks, fontsize=16, alpha=0.7)

    hashtags.head(20).sort_values('frequency').plot(kind='barh', ax=ax, align=
```

```
---------------------------------------------------------------------------
ValueError                                Traceback (most recent call last)

<ipython-input-22-aa7cf8ce124c> in <module>()
     36 ax.set_yticklabels(yticks, fontsize=16, alpha=0.7)
     37
---> 38 hashtags.head(20).sort_values('frequency').plot(kind='barh', ax=ax, ali


/usr/local/lib/python2.7/dist-packages/pandas/tools/plotting.pyc in __call_
   3771                            fontsize=fontsize, colormap=colormap, table=t
   3772                            yerr=yerr, xerr=xerr, secondary_y=secondary_y
-> 3773                            sort_columns=sort_columns, **kwds)
   3774     __call__.__doc__ = plot_frame.__doc__
   3775


/usr/local/lib/python2.7/dist-packages/pandas/tools/plotting.pyc in plot_fr
   2640                  yerr=yerr, xerr=xerr,
   2641                  secondary_y=secondary_y, sort_columns=sort_columns,
-> 2642                  **kwds)
   2643
   2644


/usr/local/lib/python2.7/dist-packages/pandas/tools/plotting.pyc in _plot(d
```

```
   2467         plot_obj = klass(data, subplots=subplots, ax=ax, kind=kind, **k
   2468
-> 2469     plot_obj.generate()
   2470     plot_obj.draw()
   2471     return plot_obj.result


    /usr/local/lib/python2.7/dist-packages/pandas/tools/plotting.pyc in generat
   1044         self._add_table()
   1045         self._make_legend()
-> 1046         self._adorn_subplots()
   1047
   1048         for ax in self.axes:


    /usr/local/lib/python2.7/dist-packages/pandas/tools/plotting.pyc in _adorn_
   1203         for ax in self.axes:
   1204             if self.yticks is not None:
-> 1205                 ax.set_yticks(self.yticks)
   1206
   1207             if self.xticks is not None:


    /usr/local/lib/python2.7/dist-packages/matplotlib/axes/_base.pyc in set_yti
   3106             Sets the minor ticks if *True*
   3107         """
-> 3108         ret = self.yaxis.set_ticks(ticks, minor=minor)
   3109         return ret
   3110


    /usr/local/lib/python2.7/dist-packages/matplotlib/axis.pyc in set_ticks(sel
   1598             xleft, xright = self.get_view_interval()
   1599             if xright > xleft:
-> 1600                 self.set_view_interval(min(ticks), max(ticks))
   1601             else:
   1602                 self.set_view_interval(max(ticks), min(ticks))


    /usr/local/lib/python2.7/dist-packages/matplotlib/axis.pyc in set_view_inte
   2260             if Vmin < Vmax:
   2261                 self.axes.viewLim.intervaly = (min(vmin, vmax, Vmin),
-> 2262                                                max(vmin, vmax, Vmax))
   2263             else:
   2264                 self.axes.viewLim.intervaly = (max(vmin, vmax, Vmin),


    /usr/local/lib/python2.7/dist-packages/matplotlib/transforms.pyc in _set_ir
```

```
    991
    992     def _set_intervaly(self, interval):
--> 993         self._points[:, 1] = interval
    994         self.invalidate()
    995     intervaly = property(BboxBase._get_intervaly, _set_intervaly)


ValueError: could not convert string to float: trumptrain
```



1.0.3 URLs

```
In [15]: urls = countEntities(entities, 'urls')
         urls.head(50)
```

Out[15]:

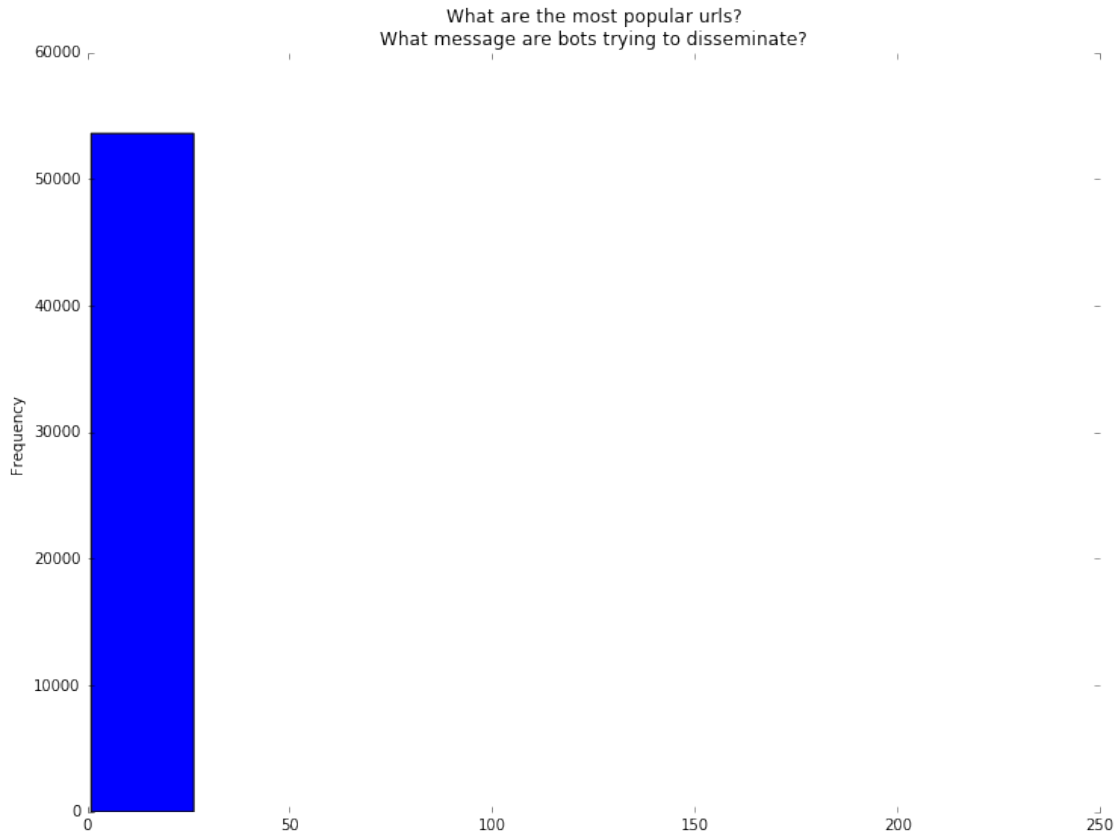
	urls	frequency
0	http://cnnpolitics.com	2535
1	http://cnn.it/20ndgad	2494
2	http://growapair.gq	1362
3	http://bit.ly/1rcvg6d	807
4	http://ift.tt/1jjvmta	760
5	http://on.fb.me/1l1mvpu	749
6	http://ift.tt/1h21yzd	565
7	http://ift.tt/1kfdkrh	432

8	http://www.apple.co/1h1p4do	309
9	https://youtu.be/gw8c2cq-vpg	301
10	http://dld.bz/eczmp	234
11	http://ebay.to/1pitmfy	170
12	http://dld.bz/ec2cw	167
13	http://ebay.to/1uizndx	166
14	http://ebay.to/1uizkyg	166
15	http://dcwhispers.com/global-elite-fear-donald...	166
16	http://ebay.to/1pisdoq	163
17	http://ebay.to/1uizmpo	161
18	http://greatagain.ml	157
19	http://tl.gd/n_1soe0sp	156
20	http://bit.ly/1l1mvpu	155
21	http://www.coachisright.com/why-the-dogs-of-he...	147
22	http://trumptrain.ml	120
23	http://ift.tt/1laz5qy	113
24	http://bit.ly/28vc4bv	97
25	http://rsbn.tv/live-stream-donald-trump-town-h...	78
26	http://cuttingedge.org/news/n1191.cfm	76
27	http://tl.gd/n_1son5ue	75
28	https://ballotpedia.org/closed_primary	74
29	http://nbcnews.com	63
30	http://ift.tt/1nmpcbc	61
31	http://ift.tt/1h0hfcl	56
32	http://ift.tt/1kb12cl	54
33	https://wp.me/p6uzrj-73o/	54
34	http://www.apple.co/1bdjqj2	49
35	https://wp.me/p6uzrj-7hj/	48
36	http://ift.tt/1hbcwlk	46
37	https://twibble.io	46
38	http://tl.gd/n_1sodgar	45
39	http://ift.tt/1dtm24y	44
40	https://www.therealstrategy.com/donald-trump-o...	43
41	http://ift.tt/1qnk1to	41
42	http://awe.sm/qab2c	40
43	http://www.cnn.com/2016/05/10/politics/gary-jo...	37
44	http://www.cuttingedge.org/news/n1191.cfm	31
45	http://tl.gd/n_1solcsi	31
46	http://openlettertodonaldtrump.com/	30
47	http://trumpdonald.org	30
48	http://news24.com	30
49	http://ift.tt/1izjoez	29

In [16]:
```
# # distribution of links

# Create a figure and subploy
fig = plt.figure(figsize=(12,9))
ax = fig.add_subplot(111)
```

```
        # Formatting
        ax.grid(False)
        ax.set_frame_on(False)

        # limit axis so we can see more
        matplotlib.pylab.xlim([0, 250])

        urls['frequency'].plot(kind='hist', bins=100,
                               title='What are the most popular urls?\nWhat messag
```

```
Out[16]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01d6838310>
```



1.0.4 Users

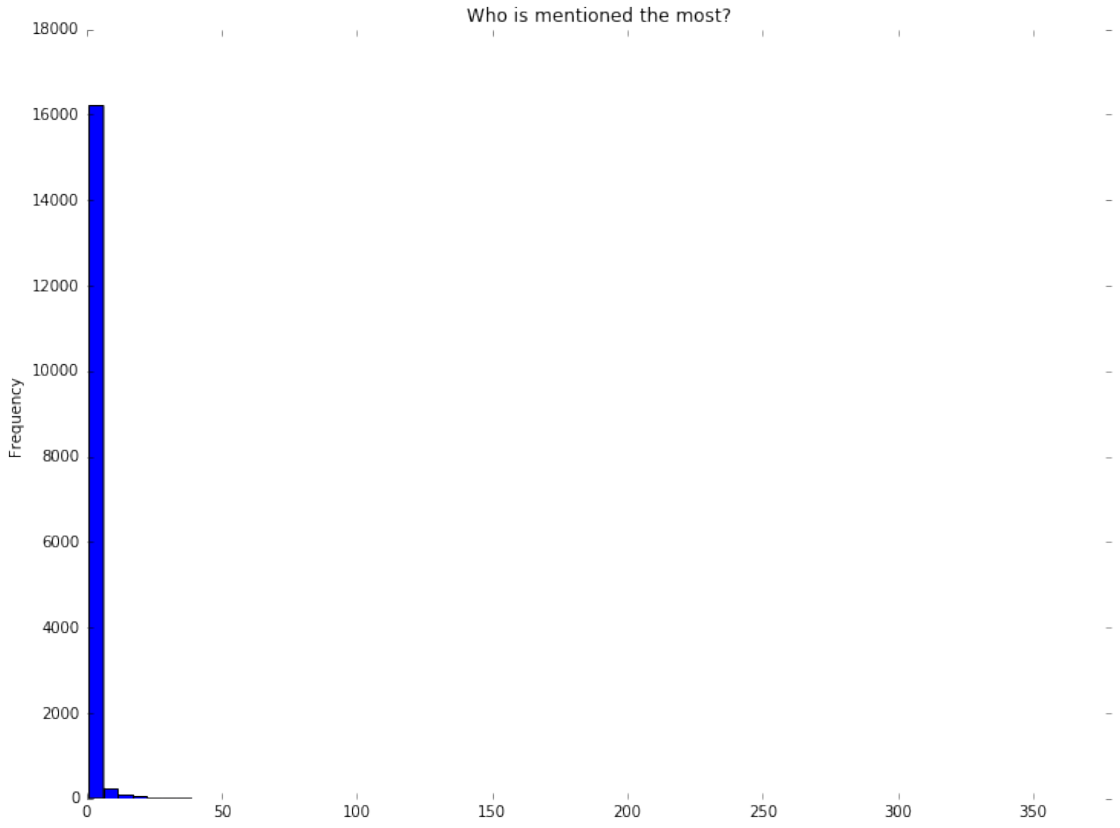
```
In [23]: users = countEntities(entities, 'user_mentions')
         users.head(50)
```

```
Out[23]:       user_mentions  frequency
         0   realdonaldtrump       1411
         1           youtube        657
```

2	samstwitch	538
3	wesleyrickard	522
4	californiagop45	509
5	lindasuhler	379
6	cnnpolitics	336
7	realcalvinhobbs	261
8	stylishrentals	255
9	danscavino	199
10	mitchellvii	194
11	endorsementsgop	190
12	huffpostpol	188
13	nytpolitics	163
14	dcexaminer	161
15	hillaryclinton	151
16	crowdfundgurus	146
17	tahquamenon70	139
18	breitbartnews	138
19	teamtrumpaz	138
20	cnn	132
21	gerfingerpoken	131
22	nytimes	127
23	tedcruz	120
24	slone	115
25	housecracka	104
26	immigrant4trump	99
27	risetoflyy	96
28	conceptgrp	94
29	wpjenna	93
30	foxnews	88
31	stophillarypac	85
32	dailycaller	83
33	abc	79
34	seanhannity	75
35	gop	72
36	phxken	72
37	weneedtrump	70
38	thelastrefuge2	64
39	usafortrump2016	64
40	jackdix03868724	62
41	uthornsrawk	60
42	braveconwarrior	59
43	donaldjohnted	57
44	washingtonpost	57
45	thepatriot143	56
46	gatewaypundit	54
47	theblaze	54
48	darren32895836	52
49	italians4trump	51

In [24]:
```
# # distribution of users

# Create a figure and subploy
fig = plt.figure(figsize=(12,9))
ax = fig.add_subplot(111)

# Formatting
ax.grid(False)
ax.set_frame_on(False)

# limit axis so we can see more
matplotlib.pylab.xlim([0, users['frequency'].iloc[5:].max()])

# use iloc to skip first two (donaldtrump and youtube)
users['frequency'].iloc[2:].plot(kind='hist', bins=100,
                                 title='Who is mentioned the most?')
```

Out[24]: <matplotlib.axes._subplots.AxesSubplot at 0x7f01dee34c90>



In [ ]:
```
def is_outlier(points, thresh=3.5):
    """
```

```
    Returns a boolean array with True if points are outliers and False
    otherwise.

    Parameters:
    -----------
        points : An numobservations by numdimensions array of observations
        thresh : The modified z-score to use as a threshold. Observations w
            a modified z-score (based on the median absolute deviation) gre
            than this value will be classified as outliers.

    Returns:
    --------
        mask : A numobservations-length boolean array.

    References:
    ----------
        Boris Iglewicz and David Hoaglin (1993), "Volume 16: How to Detect
        Handle Outliers", The ASQC Basic References in Quality Control:
        Statistical Techniques, Edward F. Mykytka, Ph.D., Editor.
    """
    if len(points.shape) == 1:
        points = points[:,None]
    median = np.median(points, axis=0)
    diff = np.sum((points - median)**2, axis=-1)
    diff = np.sqrt(diff)
    med_abs_deviation = np.median(diff)

    modified_z_score = 0.6745 * diff / med_abs_deviation

    return modified_z_score > thresh
```