chris-viz

October 23, 2017

Import visualization libraries and set `matplotlib` plot style:

```
In [1]: import pickle
        import seaborn as sns
        import matplotlib.pyplot as plt
        import pandas as pd
        from wordcloud import WordCloud
        import spacy
        from nltk.corpus import stopwords
        from collections import Counter
        from nltk.util import ngrams

        %matplotlib inline
        sns.set_style('whitegrid')
```

Read in `DataFrame` from serialized `pickle`:

```
In [2]: data = pickle.load(open("preprocessed-9-18.pkl", "rb"))
```

Look at first 5 rows:

```
In [3]: data.head()
```

```
Out[3]:                                                comment  hate         author  \
        1133                                               Kek   0.0    JenBenBoo92
        3550  Very relatable and Im not even in the US Who w...   0.0       DeXyDeXy
        5394  With the amount of cord cutters its not going ...   0.0      Bigly1821
        3371  Yeah its usually best when people who try to c...   0.0         jl2121
        7408  To the people who are defending this email if ...   0.0  Miles_Prower1

              subreddit  coder  id  binary_hate  duplicate  \
        1133  The_Donald      6   5            0      False
        3550  conspiracy      6   7            0      False
        5394  The_Donald      9   9            0      False
        3371  The_Donald      0  10            0      False
        7408  The_Donald      2  15            0      False

                                           tokenized_comment  \
```

```
1133                                                [Kek]
3550  [Very, relatable, and, Im, not, even, in, the,...
5394  [With, the, amount, of, cord, cutters, its, no...
3371  [Yeah, its, usually, best, when, people, who, ...
7408  [To, the, people, who, are, defending, this, e...

                                  sentencized_comment  \
1133                                              [Kek]
3550  [Very relatable and Im not even in the US Who ...
5394  [With the amount of cord cutters its not going...
3371  [Yeah its usually best when people who try to ...
7408  [To the people who are defending this email if...

                           ...                          num_capitals  \
1133                       ...                                     1
3550                       ...                                    20
5394                       ...                                     1
3371                       ...                                     3
7408                       ...                                     5

     proportion_capitals  num_all_caps_words  num_titlecase_words  num_words  \
1133            0.333333                   0                    1          1
3550            0.038023                   3                   18         95
5394            0.017857                   0                    1         12
3371            0.015075                   0                    3         40
7408            0.031056                   0                    5         31

      mean_word_length  num_sentences  mean_sentence_length  num_punctuation  \
1133          3.000000              1                   1.0                0
3550          5.536842              1                  95.0                0
5394          4.666667              1                  12.0                0
3371          4.975000              1                  40.0                0
7408          5.193548              1                  31.0                0

                              clean_tokenized_comment
1133                                          [kek]
3550  [relatable, im, even, us, would, vote, questio...
5394              [amount, cord, cutters, going, hard]
3371  [yeah, usually, best, people, try, change, sai...
7408  [people, defending, email, work, voted, trump,...

[5 rows x 22 columns]
```

Number of all comments:

```
In [4]: len(data)

Out[4]: 7619
```

Counts of hate and non-hate:

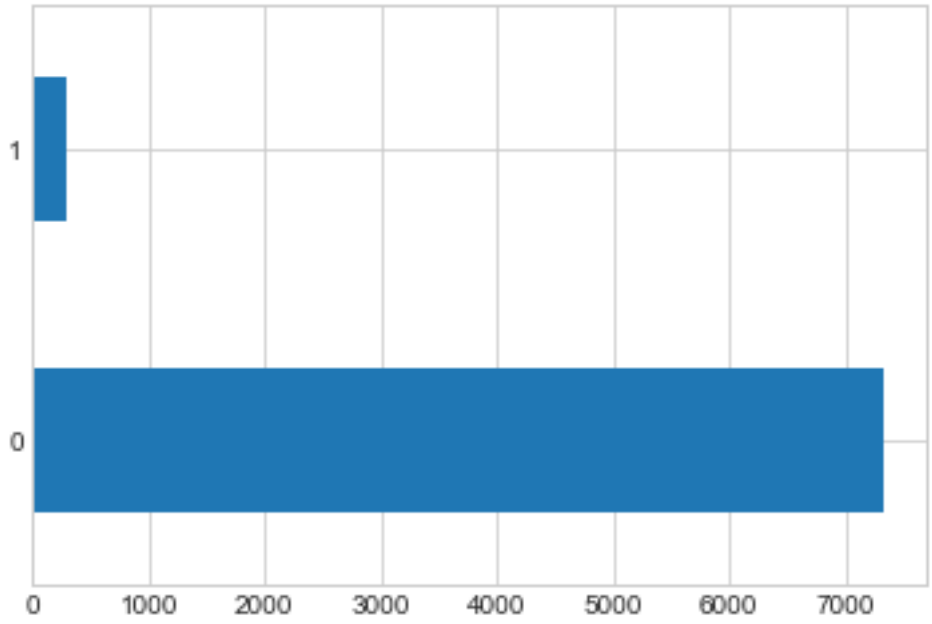
```
In [5]: data['binary_hate'].value_counts()
```

```
Out[5]: 0    7330
        1     289
        Name: binary_hate, dtype: int64
```

0.1 Basic bar plot of counts for hate and non-hate:

```
In [6]: data['binary_hate'].value_counts().plot.barh()
```

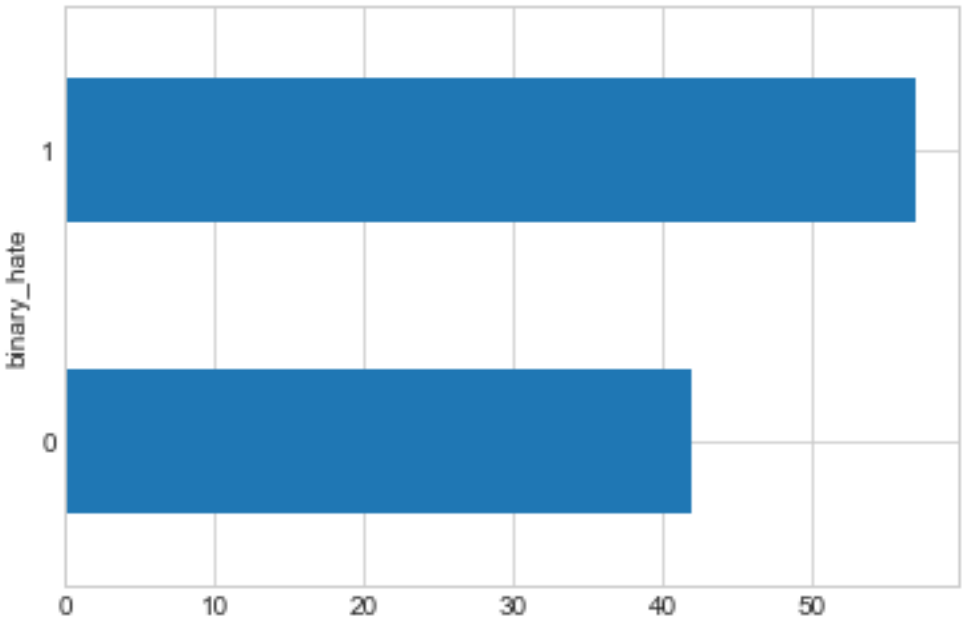
```
Out[6]: <matplotlib.axes._subplots.AxesSubplot at 0x11a356a58>
```



0.2 Mean number of words for hate vs non-hate comment:

```
In [7]: data.groupby("binary_hate").mean()['num_words'].plot.barh()
```

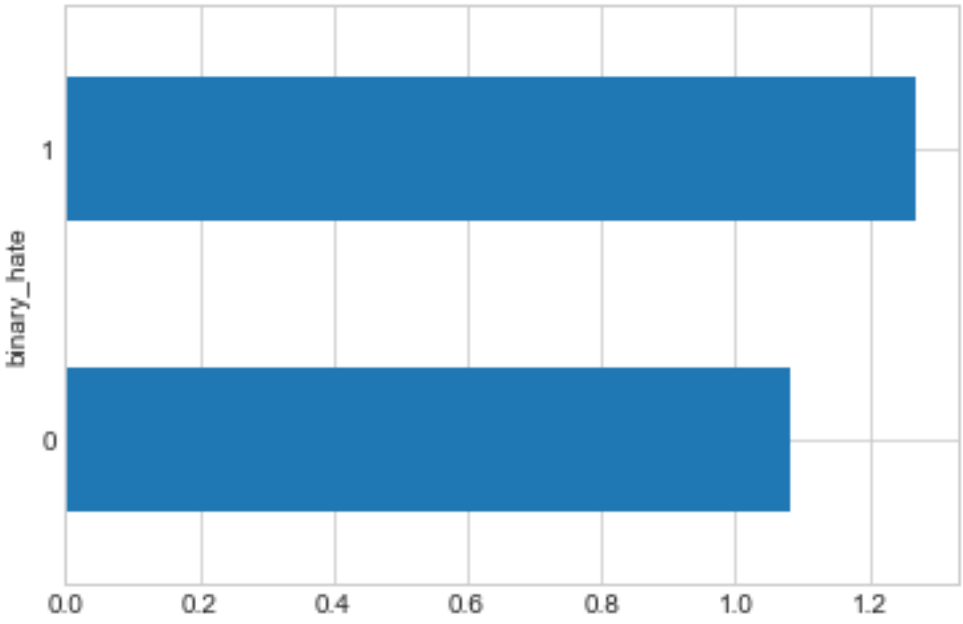
```
Out[7]: <matplotlib.axes._subplots.AxesSubplot at 0x11a504c50>
```



0.3 Mean number of all caps words for hate vs. non-hate comments:

```
In [8]: data.groupby("binary_hate").mean()['num_all_caps_words'].plot.barh()
```

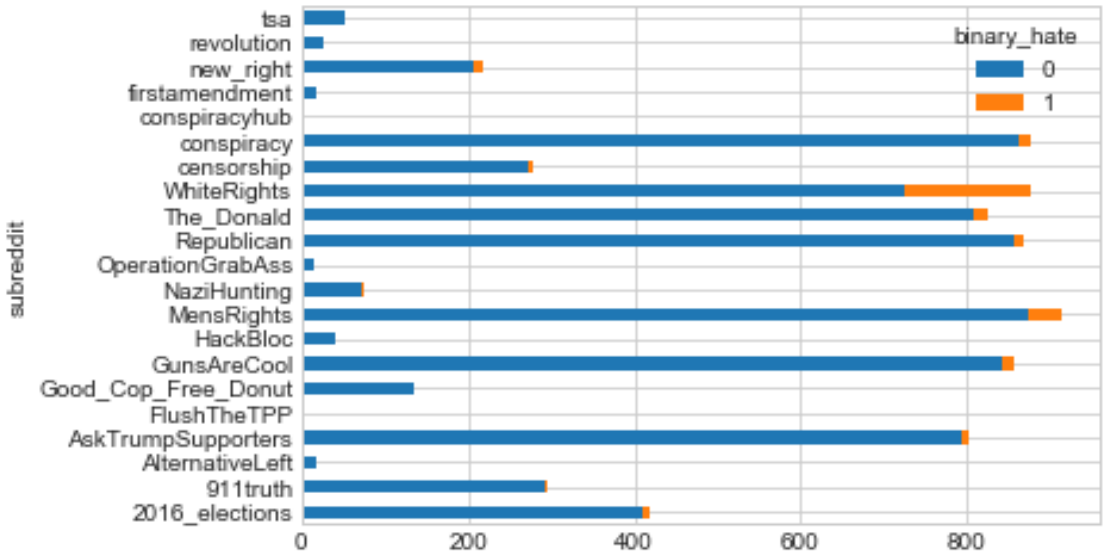
```
Out[8]: <matplotlib.axes._subplots.AxesSubplot at 0x11a721710>
```



0.4 Proportion of hate vs. non-hate by subreddit:

```
In [9]: data.groupby(["subreddit",'binary_hate'])['binary_hate'].count().unstack().plot.barh(sta
```

```
Out[9]: <matplotlib.axes._subplots.AxesSubplot at 0x11a9b5278>
```



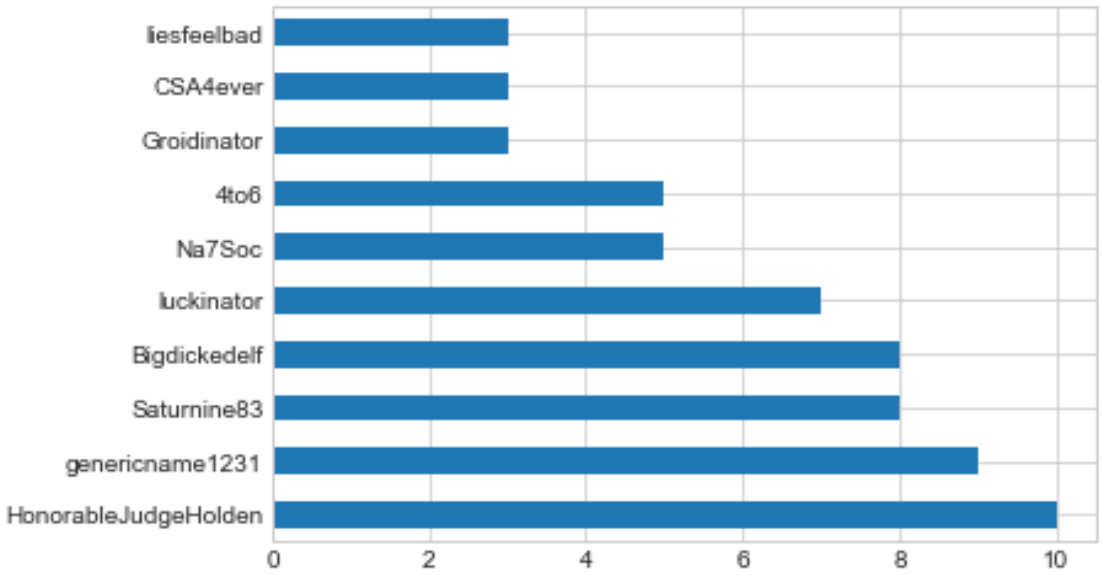
Subset `DataFrame` into hate and non-hate:

```
In [10]: hate_only = data[data['binary_hate'] == 1]
         not_hate = data[data['binary_hate'] == 0]
```

0.5 Author counts of hate comments:

```
In [11]: hate_only['author'].value_counts()[:10].plot.barh()
```

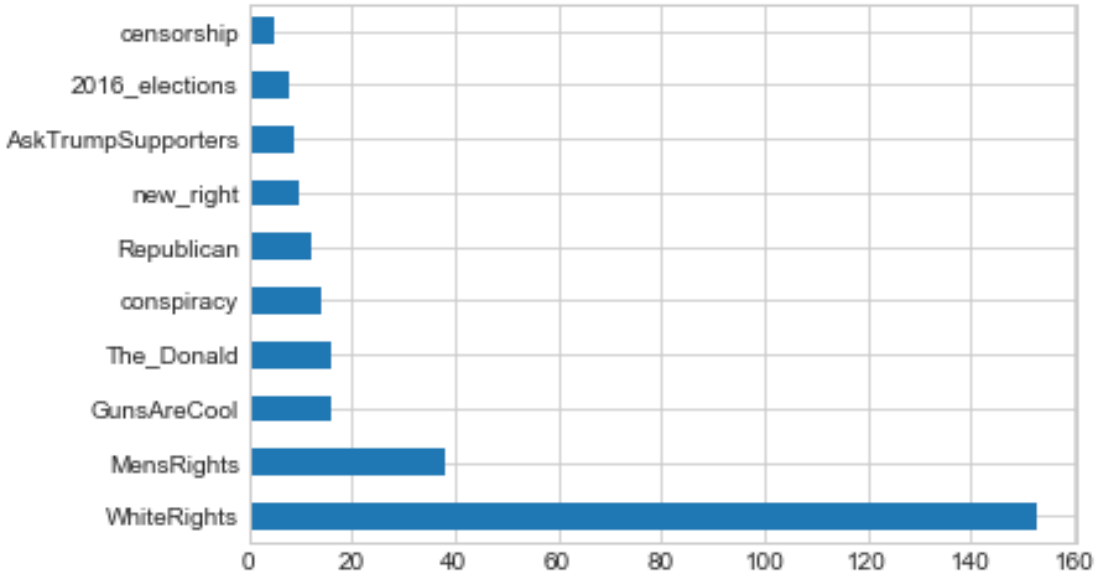
```
Out[11]: <matplotlib.axes._subplots.AxesSubplot at 0x11ab8c828>
```



0.6 Hate comment counts by subreddit:

```
In [12]: hate_only['subreddit'].value_counts()[:10].plot.barh()
```

```
Out[12]: <matplotlib.axes._subplots.AxesSubplot at 0x11abb07f0>
```



1 Text Viz

Run comments through NLP pipeline:

```
In [13]: nlp = spacy.load('en')
         hate_only['spacy'] = hate_only['comment'].apply(nlp)
```

```
/Users/chench/anaconda/lib/python3.5/site-packages/ipykernel/__main__.py:2: SettingWithCopyWarni
A value is trying to be set on a copy of a slice from a DataFrame.
Try using .loc[row_indexer,col_indexer] = value instead

See the caveats in the documentation: http://pandas.pydata.org/pandas-docs/stable/indexing.html#
  from ipykernel import kernelapp as app
```

1.1 Basic word cloud from only hate comments:

```
In [14]: raw = ' '.join([' '.join([t for t in c if t.lower() not in stopwords.words("english")])

         wordcloud = WordCloud().generate(raw)

         wordcloud = WordCloud(max_font_size=80).generate(raw)
         plt.figure()
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```


```
In [15]: Counter(raw.split()).most_common()[:20]
```

```
Out[15]: [('people', 98),
          ('white', 82),
```

```
 ('like', 65),
 ('dont', 47),
 ('women', 45),
 ('Jews', 41),
 ('one', 40),
 ('Im', 36),
 ('men', 35),
 ('Jewish', 34),
 ('want', 33),
 ('get', 33),
 ('think', 32),
 ('would', 29),
 ('need', 29),
 ('black', 27),
 ('know', 27),
 ('back', 27),
 ('whites', 26),
 ('even', 26)]
```

1.2 Bigram word cloud only hate comments:

```
In [16]: from nltk.util import ngrams

         hate_ngrams = ngrams(raw.split(), 2)
         hate_ngrams_string = ""
         for n in hate_ngrams:
             hate_ngrams_string += n[0] + "" + n[1] + " "

         print(hate_ngrams_string[:100])
         wordcloud = WordCloud(max_font_size=80).generate(hate_ngrams_string)
         plt.figure()
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```

```
FUCKINGWHITE WHITEMALES MALESDeflecting Deflectingquestion questionbitch bitchFucking Fuckingghe
```


```
In [17]: Counter(hate_ngrams_string.split()).most_common()[:20]
```

```
Out[17]: [('whitepeople', 9),
          ('blackpeople', 6),
          ('goback', 5),
          ('gonna', 5),
          ('hateniggers', 5),
          ('yearold', 5),
          ('whiterace', 5),
          ('dontwant', 4),
          ('WhiteHouse', 4),
          ('dontcare', 4),
          ('whitemen', 4),
          ('lookslike', 4),
          ('freespeech', 4),
          ('peoplelike', 4),
          ('dontlike', 4),
          ('isntgoing', 4),
          ('youresaying', 3),
          ('whitemale', 3),
          ('knowmany', 3),
          ('looklike', 3)]
```

1.3 Basic word cloud from non-hate comments:

```
In [18]: raw = ' '.join([' '.join([t for t in c if t.lower() not in stopwords.words("english")])
```

```
         wordcloud = WordCloud().generate(raw)
```

```
         wordcloud = WordCloud(max_font_size=80).generate(raw)
```

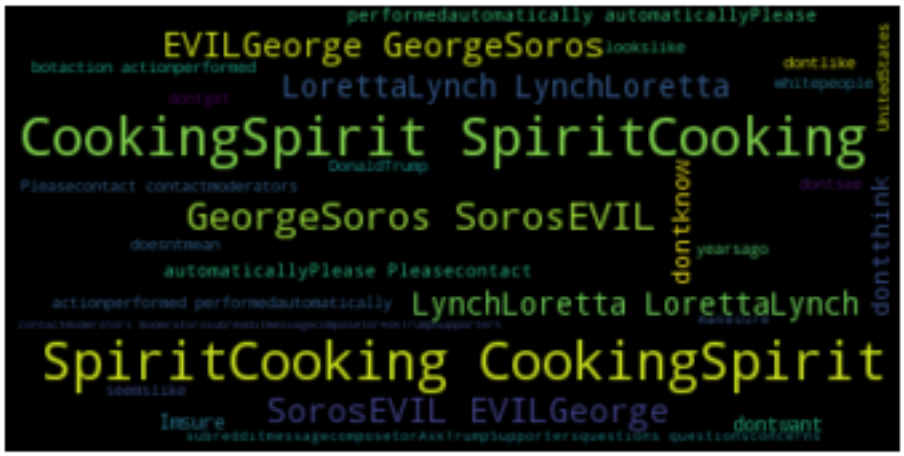
```
plt.figure()
plt.imshow(wordcloud, interpolation="bilinear")
plt.axis("off")
plt.show()
```

In [19]:
```
Counter(raw.split()).most_common()[:20]
```

Out[19]:
```
[('people', 1383),
 ('would', 1098),
 ('dont', 1055),
 ('like', 1054),
 ('think', 857),
 ('Trump', 806),
 ('one', 719),
 ('Im', 685),
 ('get', 615),
 ('know', 549),
 ('gt', 523),
 ('even', 521),
 ('want', 510),
 ('going', 472),
 ('really', 471),
 ('see', 467),
 ('time', 463),
 ('Spirit', 461),
 ('Cooking', 460),
 ('right', 446)]
```

1.4 Bigram word cloud non-hate comments:

```
In [20]: from nltk.util import ngrams

         non_ngrams = ngrams(raw.split(), 2)
         non_ngrams_string = ""
         for n in non_ngrams:
             non_ngrams_string += n[0] + "" + n[1] + " "

         wordcloud = WordCloud(max_font_size=80).generate(non_ngrams_string)
         plt.figure()
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```




```
In [21]: Counter(non_ngrams_string.split()).most_common()[:20]
```

```
Out[21]: [('SpiritCooking', 460),
          ('CookingSpirit', 459),
          ('GeorgeSoros', 198),
          ('SorosEVIL', 192),
          ('EVILGeorge', 191),
          ('LorettaLynch', 180),
          ('LynchLoretta', 179),
          ('dontthink', 106),
          ('dontknow', 105),
          ('Imsure', 83),
          ('dontwant', 68),
          ('UnitedStates', 55),
          ('Trumpsupporters', 52),
```

```
 ('dontsee', 50),
 ('contactmoderators', 49),
 ('automaticallyPlease', 49),
 ('questionsconcerns', 49),
 ('Pleasecontact', 49),
 ('botaction', 49),
 ('actionperformed', 49)]
```

1.5 Word cloud with lemmata, only hate comments

```
In [35]: lemmata_raw = ""

         for sp in hate_only["spacy"]:
             lemmata = [word.lemma_ for word in sp if word.lemma_[-1].isupper() == False]
             lemmata = [word.strip() for word in lemmata]
             lemmata = [word for word in lemmata if len(word) > 1 and word[1].isupper() == False
             lemmata_raw += ' '.join(lemmata).replace("jews", "jew").replace("jewish", "jew")

         wordcloud = WordCloud().generate(lemmata_raw)

         # lower max_font_size
         wordcloud = WordCloud(max_font_size=70).generate(lemmata_raw)
         plt.figure()
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```

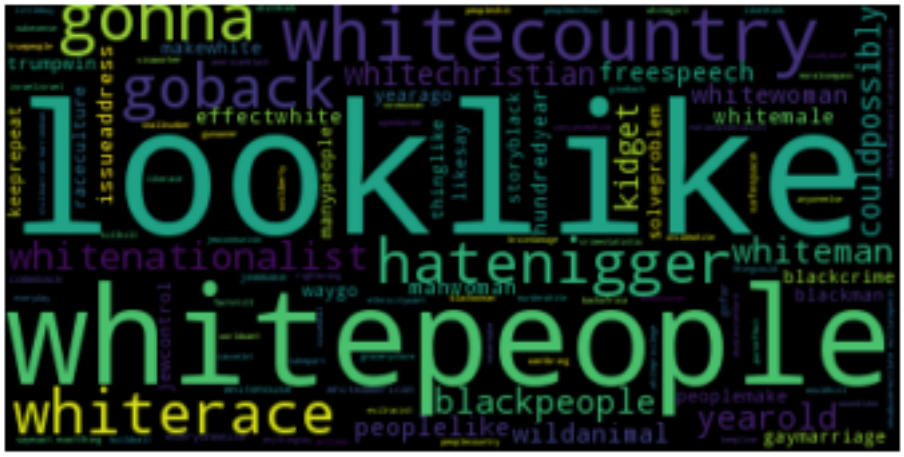

```
In [36]: Counter(lemmata_raw.split()).most_common()[:20]

Out[36]: [('white', 124),
          ('people', 94),
```

```
 ('jew', 87),
 ('like', 66),
 ('go', 58),
 ('get', 57),
 ('would', 52),
 ('woman', 52),
 ('make', 51),
 ('black', 50),
 ('say', 49),
 ('one', 46),
 ('man', 44),
 ('want', 42),
 ('race', 33),
 ('think', 32),
 ('even', 30),
 ('country', 30),
 ('back', 29),
 ('fuck', 29)]
```

1.6 Lemmata bigram word cloud, only hate comments:

```
In [37]: hate_ngrams = ngrams(lemmata_raw.split(), 2)
         hate_ngrams_string = ""
         for n in hate_ngrams:
             hate_ngrams_string += n[0] + "" + n[1] + " "

         print(hate_ngrams_string[:100])
         wordcloud = WordCloud(max_font_size=80).generate(hate_ngrams_string)
         plt.figure()
         plt.imshow(wordcloud, interpolation="bilinear")
         plt.axis("off")
         plt.show()
```

```
fuckingwhite whitemalesdeflect malesdeflectquestion questionbitchfuck bitchfuckghetto ghettopeop
```




```
In [38]: Counter(hate_ngrams_string.split()).most_common()[:20]
```

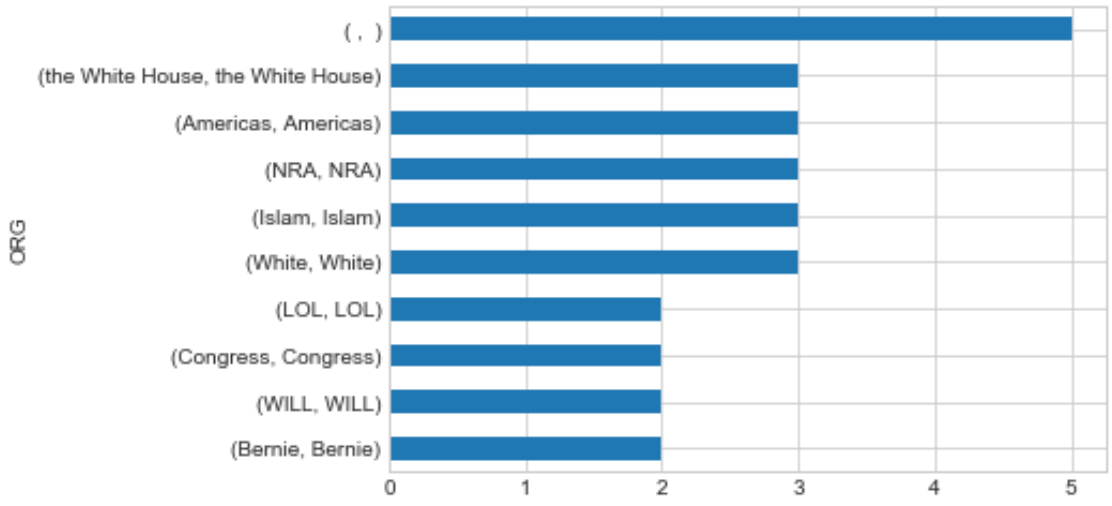
```
Out[38]: [('looklike', 8),
          ('whitepeople', 8),
          ('whitecountry', 6),
          ('gonna', 5),
          ('goback', 5),
          ('hatenigger', 5),
          ('whiterace', 5),
          ('blackpeople', 5),
          ('yearold', 5),
          ('whiteman', 5),
          ('kidget', 4),
          ('couldpossibly', 4),
          ('peoplelike', 4),
          ('whitenationalist', 4),
          ('wildanimal', 4),
          ('freespeech', 4),
          ('whitewoman', 4),
          ('whitemale', 3),
          ('trumpwin', 3),
          ('makewhite', 3)]
```

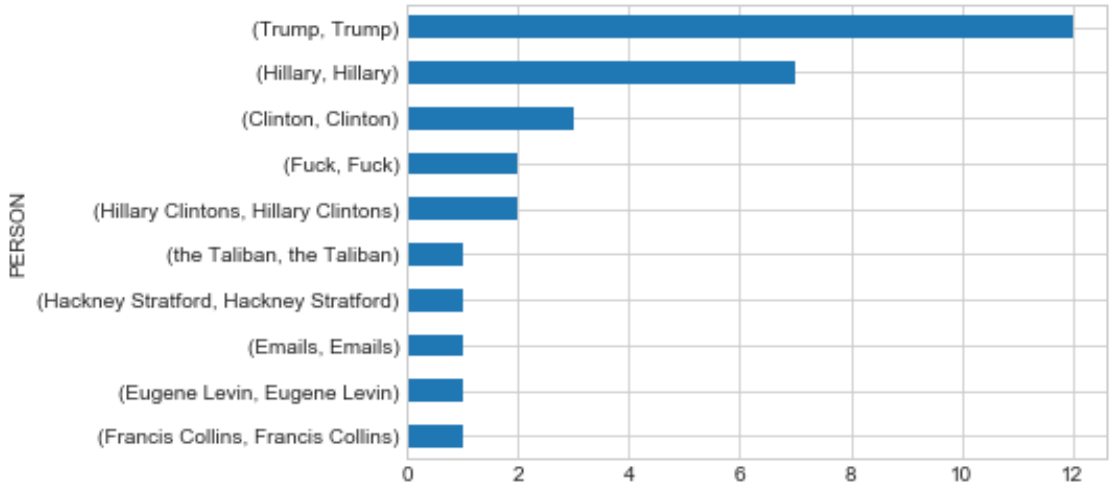
1.7 Extract entities

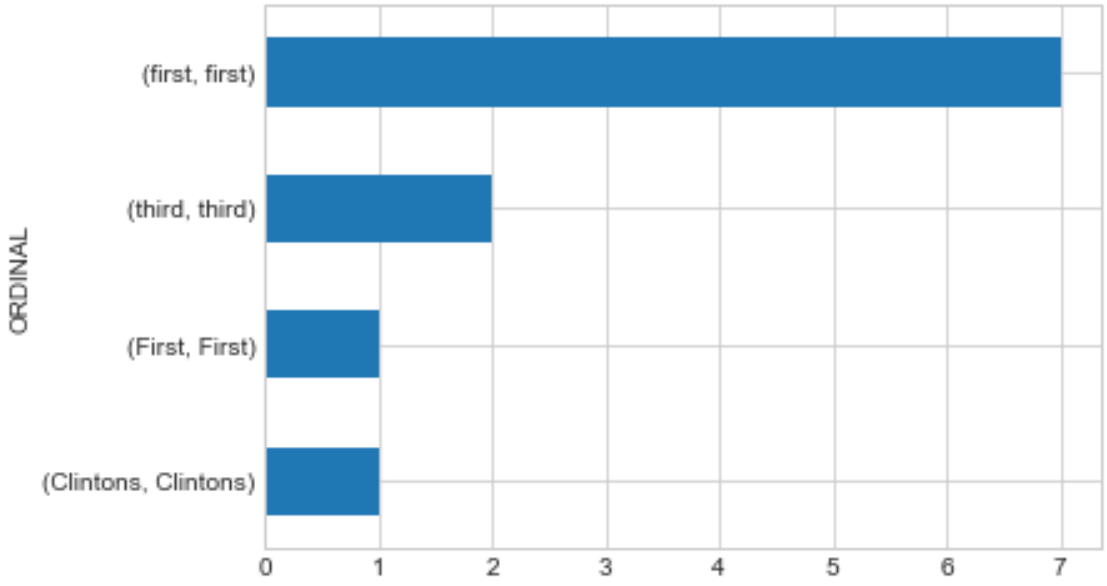
```
In [26]: all_ents = pd.DataFrame(columns=[0, 1])
         for sp in hate_only['spacy']:
             ents = [(ent.label_, ent.text) for ent in sp.ents]
             df = pd.DataFrame(ents)
             if len(df) > 0:
                 all_ents = pd.concat([all_ents, df])
```

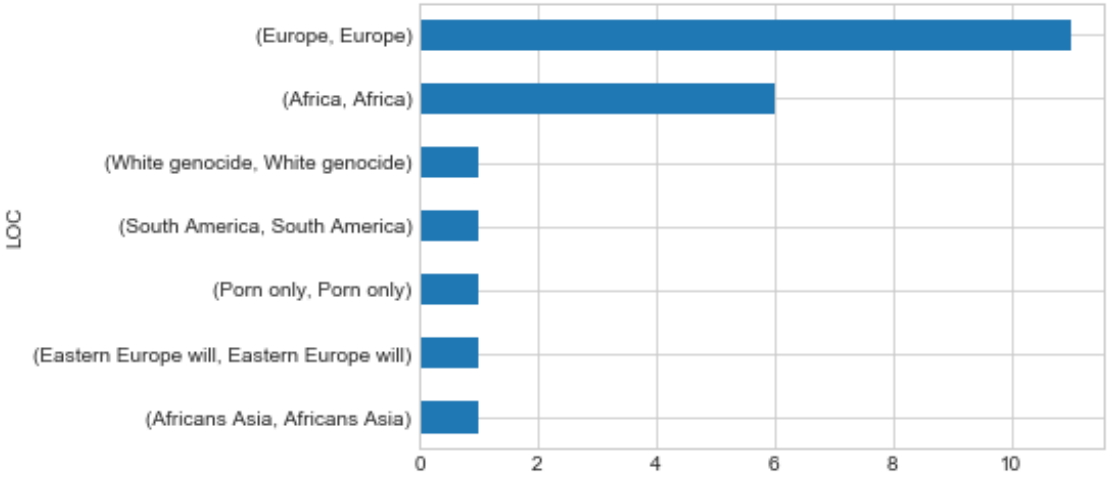
1.8 Plot entity counts by category:

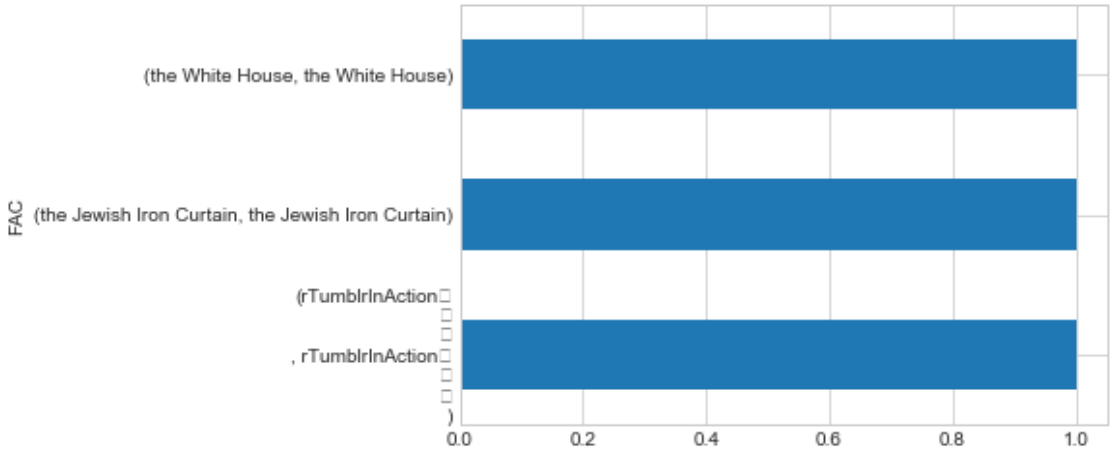
```
In [27]: all_ents.columns = ["NER", "Word"]

         for n in set(all_ents["NER"]):
             all_ents.groupby(["NER", "Word"])["Word"].value_counts()[n].sort_values()[-10:].plo
             plt.ylabel(n)
             plt.show()
```

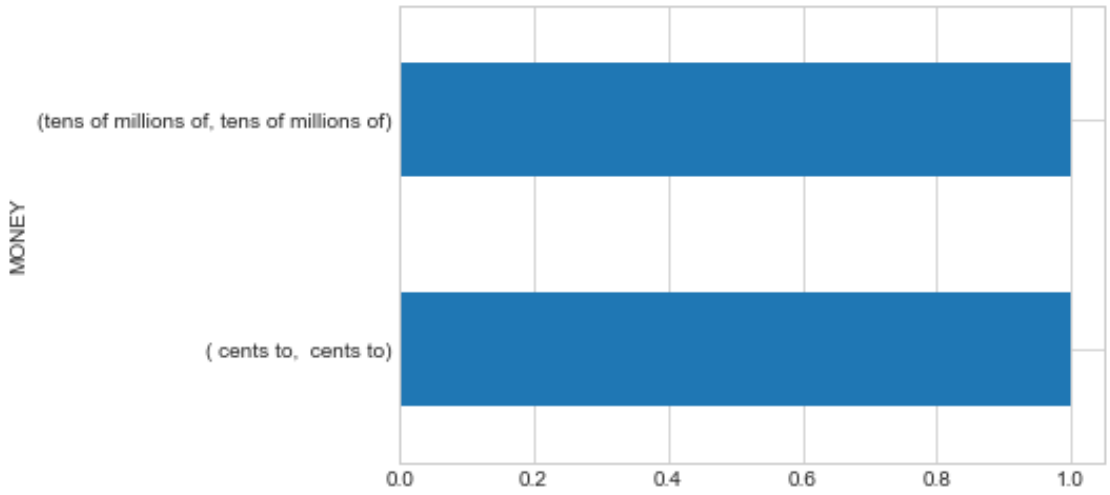


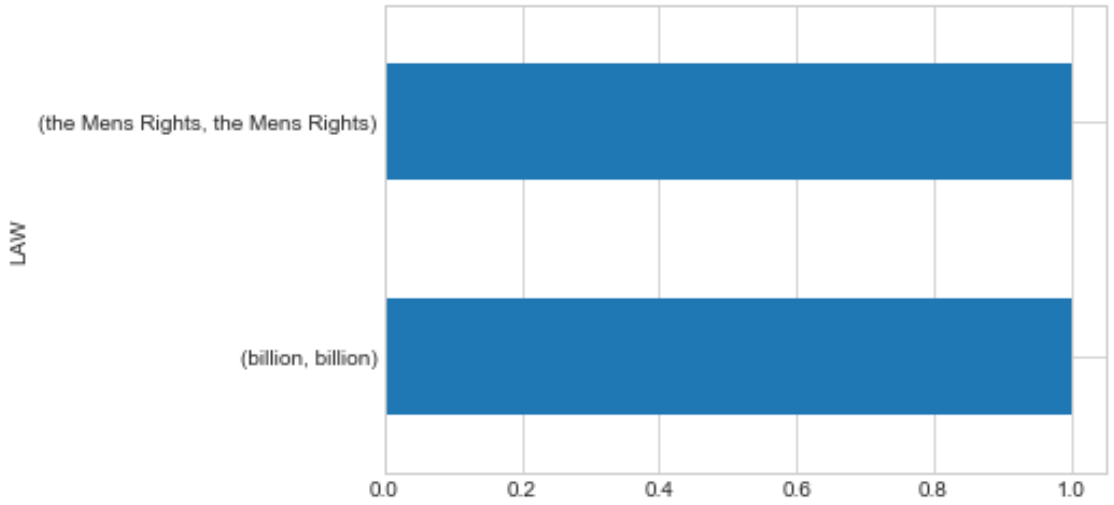


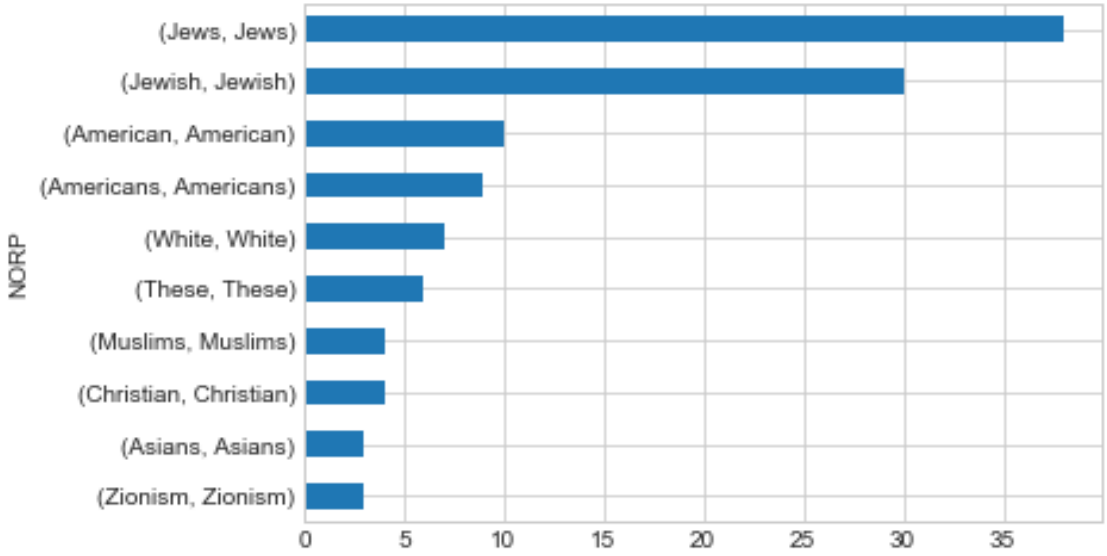

ORDINAL
(first, first)
(third, third)
(First, First)
(Clintons, Clintons)
0
1
2
3
4
5
6
7

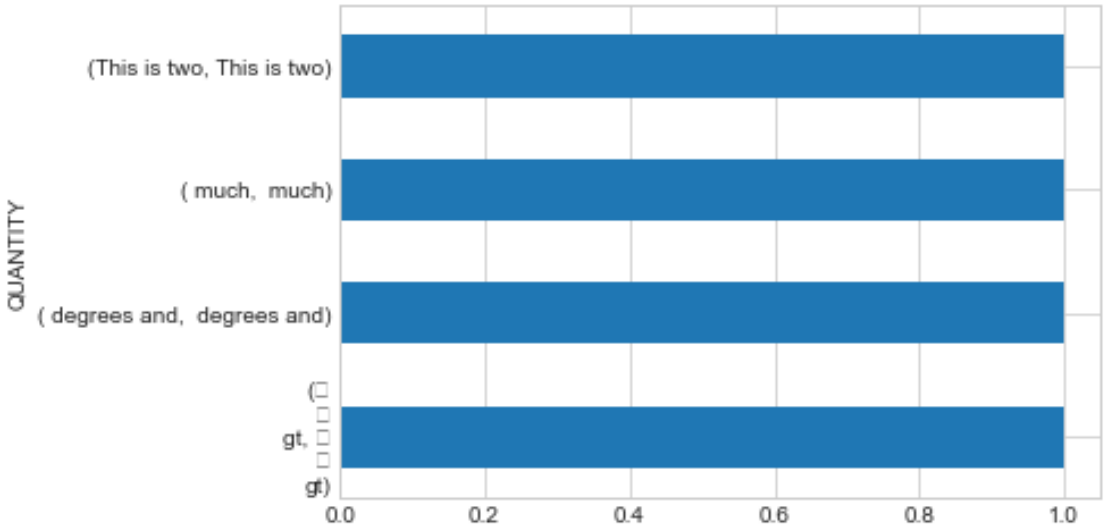

LOC
(Europe, Europe)
(Africa, Africa)
(White genocide, White genocide)
(South America, South America)
(Porn only, Porn only)
(Eastern Europe will, Eastern Europe will)
(Africans Asia, Africans Asia)
0
2
4
6
8
10

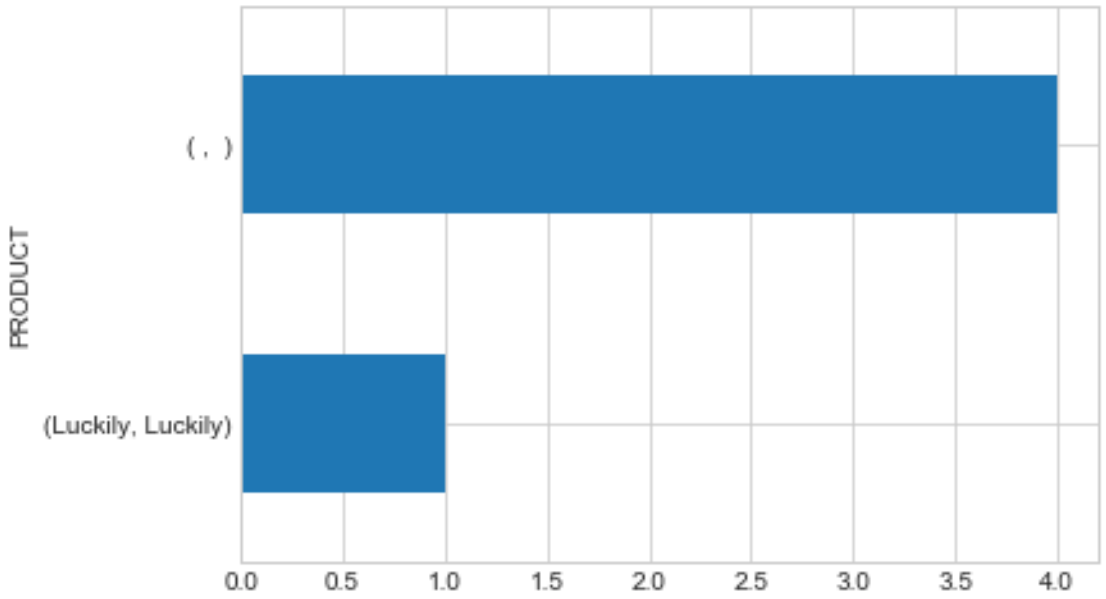

FAC
(the White House, the White House)
(the Jewish Iron Curtain, the Jewish Iron Curtain)
(rTumblrInAction□
, rTumblrInAction□
□□□□□
)
0.0
0.2
0.4
0.6
0.8
1.0

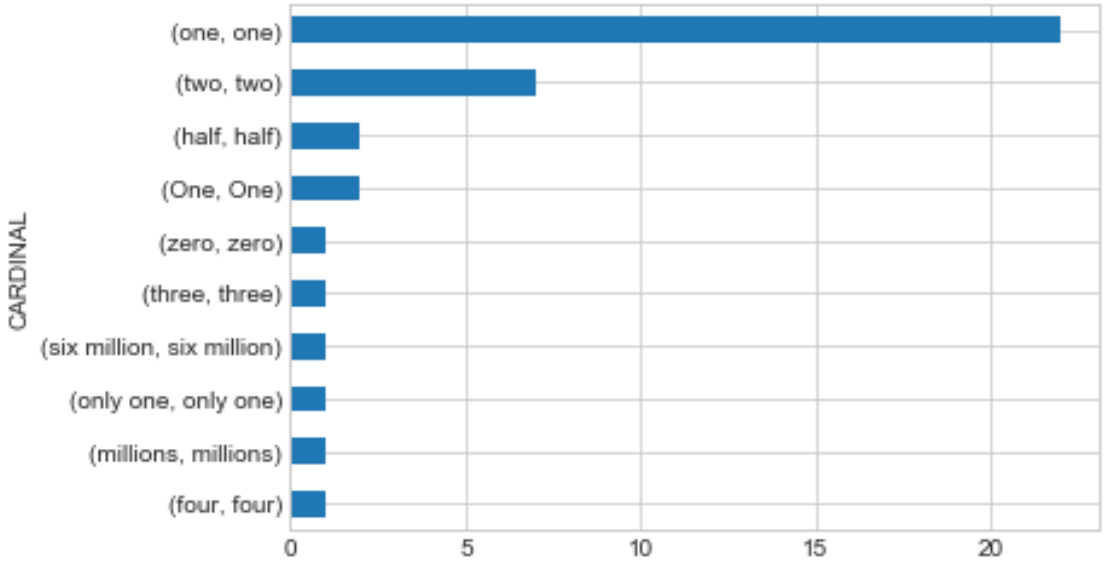

MONEY
(tens of millions of, tens of millions of)
(cents to, cents to)
0.0
0.2
0.4
0.6
0.8
1.0

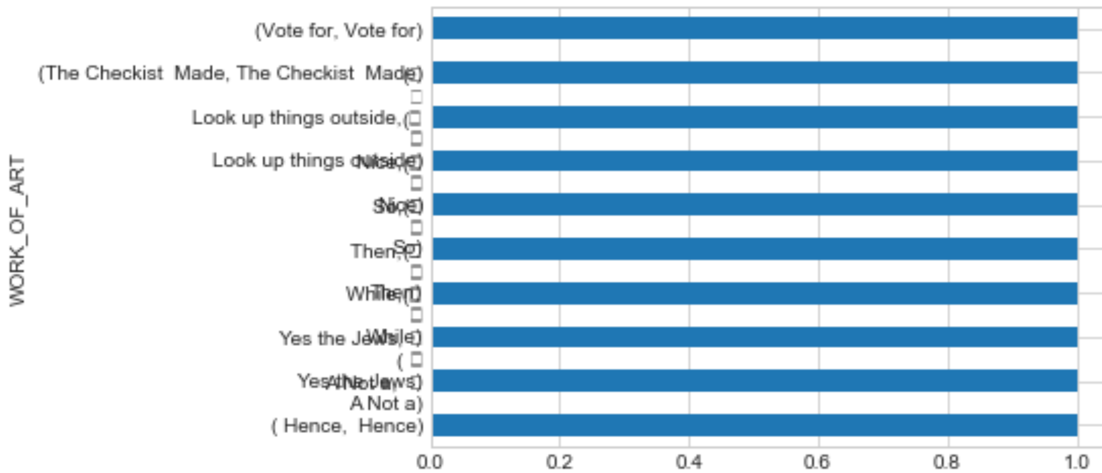

LAW
(the Mens Rights, the Mens Rights)
(billion, billion)
0.0
0.2
0.4
0.6
0.8
1.0

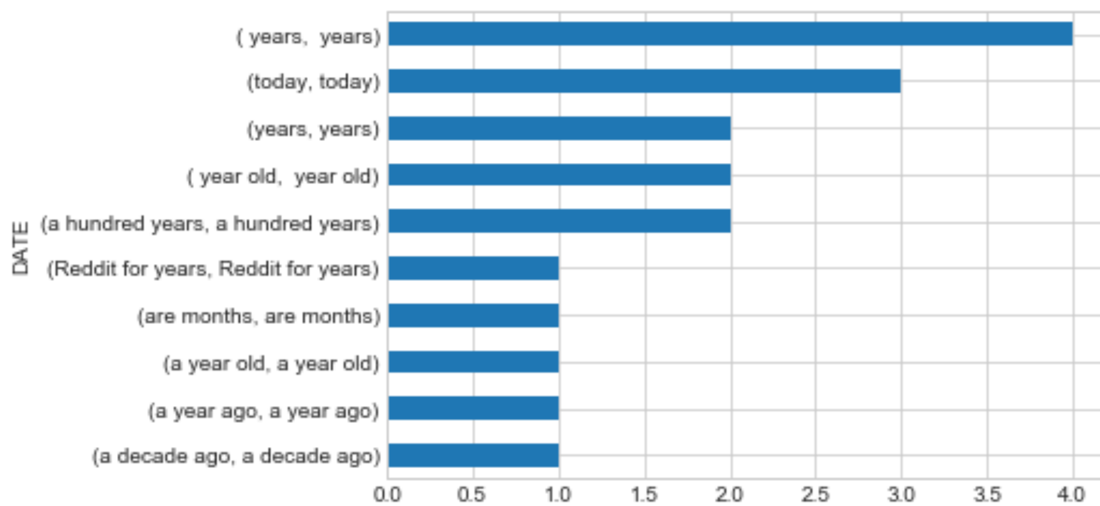

NORP
(Jews, Jews)
(Jewish, Jewish)
(American, American)
(Americans, Americans)
(White, White)
(These, These)
(Muslims, Muslims)
(Christian, Christian)
(Asians, Asians)
(Zionism, Zionism)
0
5
10
15
20
25
30
35

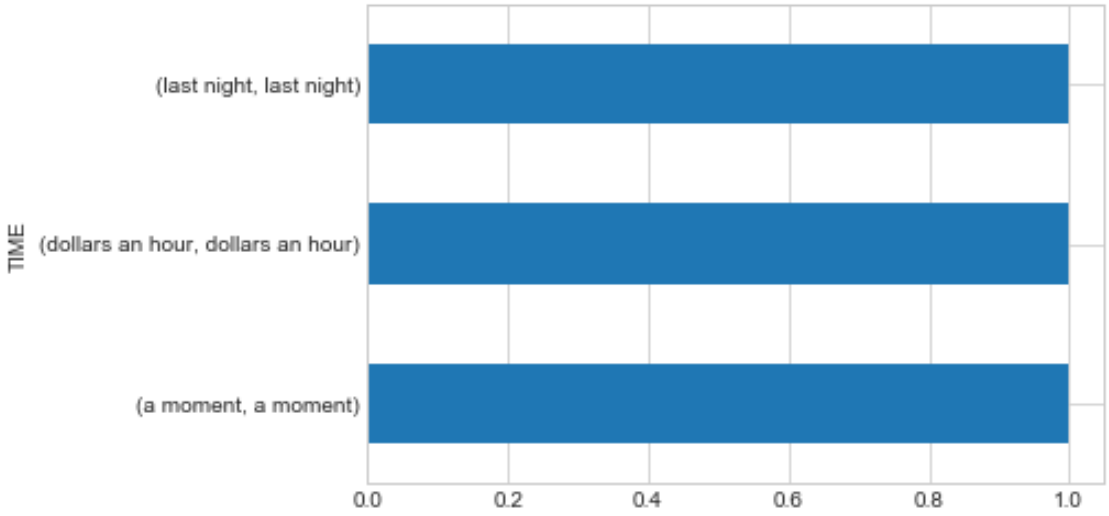

QUANTITY
(This is two, This is two)
(much, much)
(degrees and, degrees and)
(□ □ □ gt, □ □ gt)
0.0
0.2
0.4
0.6
0.8
1.0

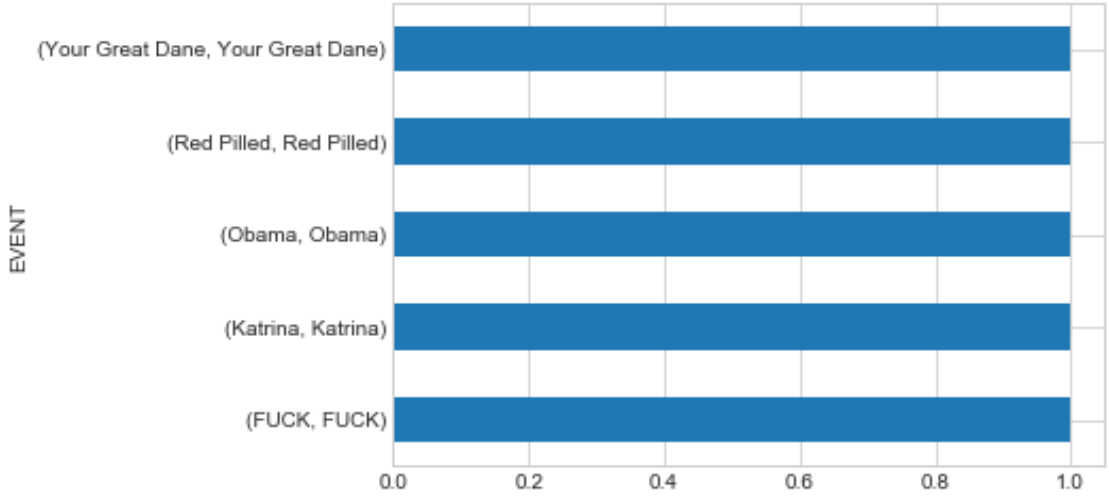

PRODUCT
(,)
(Luckily, Luckily)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0

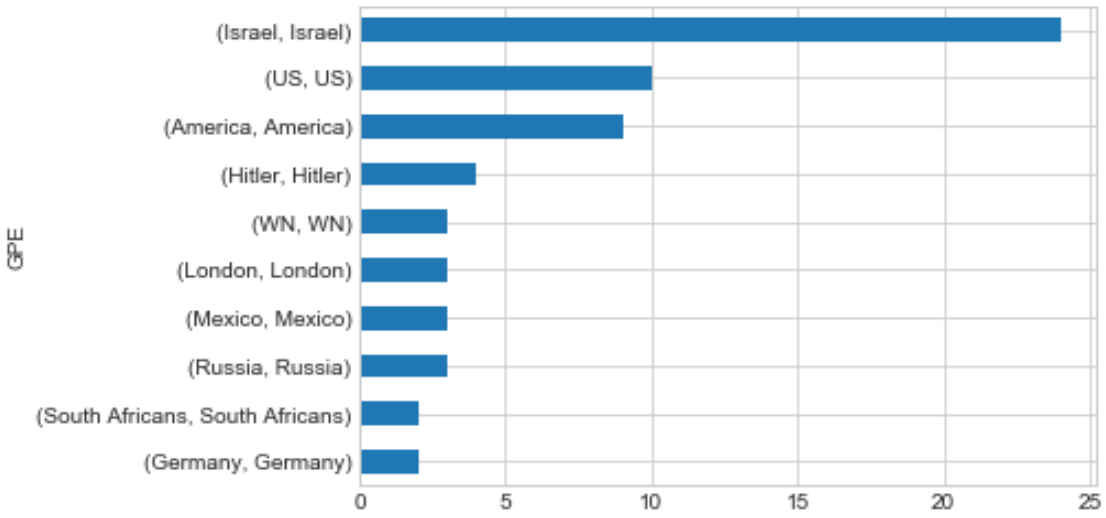

CARDINAL
(one, one)
(two, two)
(half, half)
(One, One)
(zero, zero)
(three, three)
(six million, six million)
(only one, only one)
(millions, millions)
(four, four)
0
5
10
15
20


WORK_OF_ART
(Vote for, Vote for)
(The Checkist Made, The Checkist Made)
Look up things outside,
A Not a)
(Hence, Hence)
0.0
0.2
0.4
0.6
0.8
1.0


DATE
(years, years)
(today, today)
(years, years)
(year old, year old)
(a hundred years, a hundred years)
(Reddit for years, Reddit for years)
(are months, are months)
(a year old, a year old)
(a year ago, a year ago)
(a decade ago, a decade ago)
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


TIME
(last night, last night)
(dollars an hour, dollars an hour)
(a moment, a moment)
0.0
0.2
0.4
0.6
0.8
1.0


EVENT
(Your Great Dane, Your Great Dane)
(Red Pilled, Red Pilled)
(Obama, Obama)
(Katrina, Katrina)
(FUCK, FUCK)
0.0
0.2
0.4
0.6
0.8
1.0

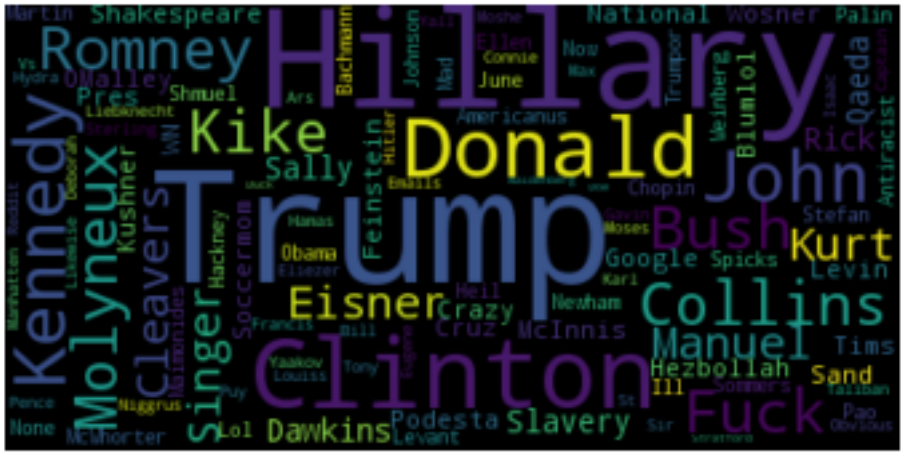


1.9 Word cloud for same entities

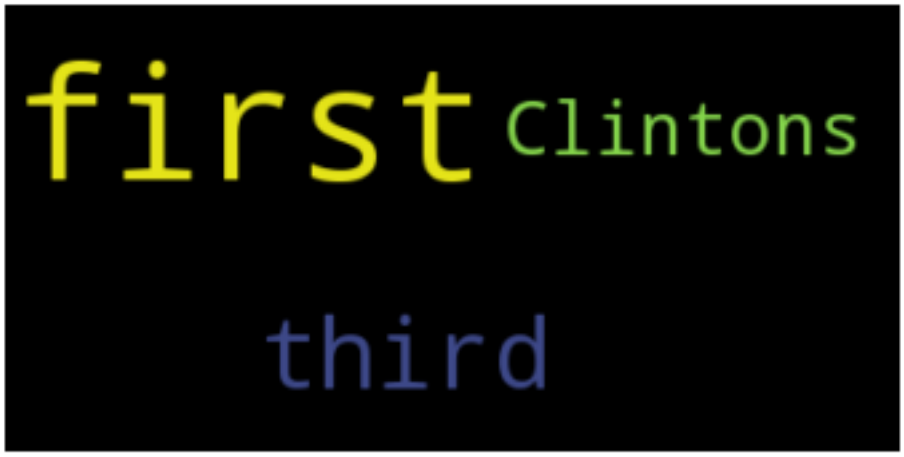
```
In [28]: for n in set(all_ents["NER"]):
             print(n)
             raw = ' '.join(list(all_ents[all_ents["NER"] == n]['Word']))
             try:
                 wordcloud = WordCloud().generate(raw)

                 # lower max_font_size
                 wordcloud = WordCloud(max_font_size=70).generate(raw)
                 plt.figure()
                 plt.imshow(wordcloud, interpolation="bilinear")
                 plt.axis("off")
                 plt.show()
             except:
                 pass
```

```
ORG
```

PERSON



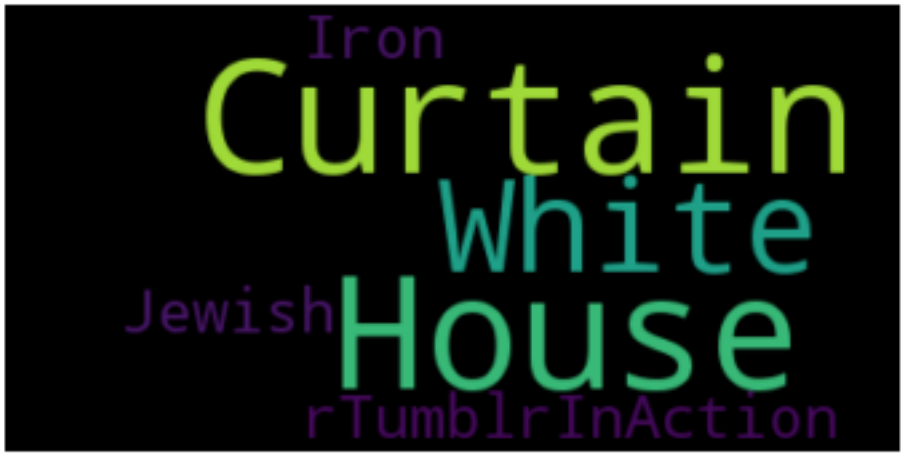
ORDINAL


first
Clintons
third

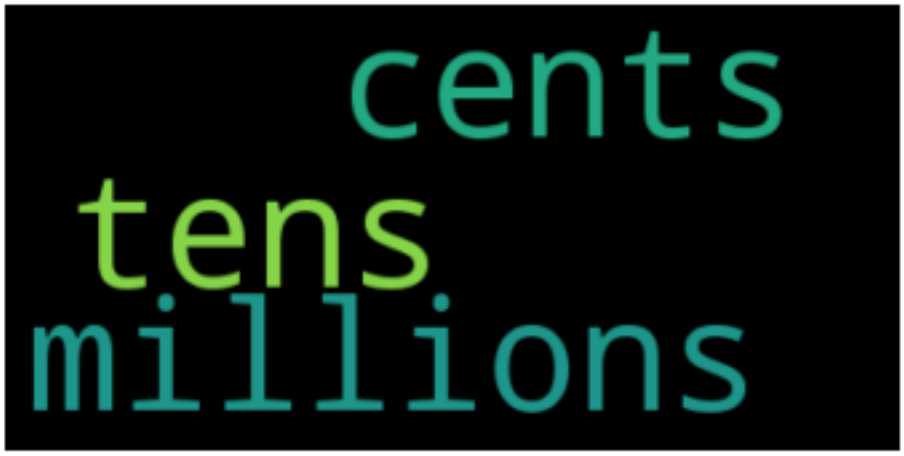
LOC


America
White
Asia
will
Europe
genocide
Eastern
Africans
South
Africa
Porn

FAC


Iron
Curtain
White
Jewish
House
rTumblrInAction

MONEY


cents
tens
millions

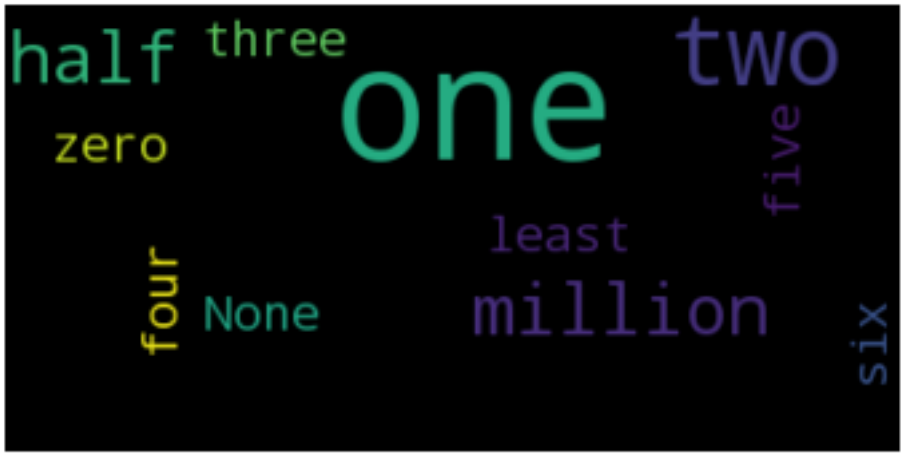
LAW

NORP

QUANTITY

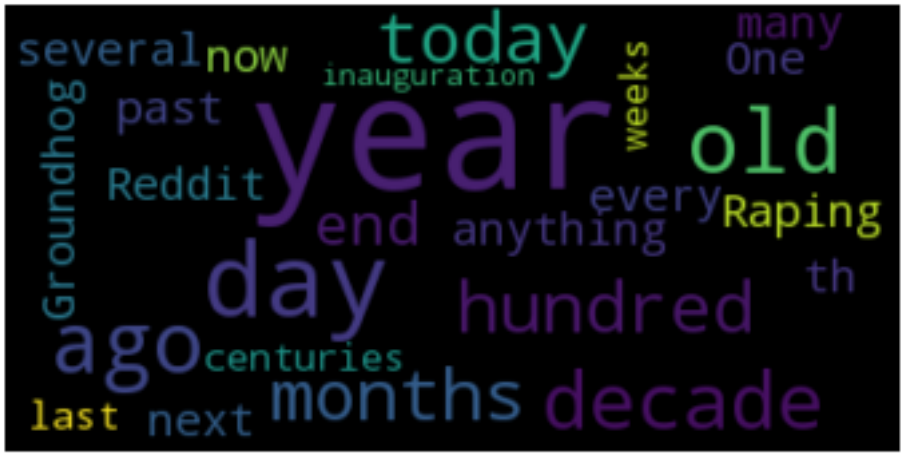
PRODUCT

CARDINAL



WORK_OF_ART

DATE


several
now
today
inauguration
many
One
Groundhog
past
year
weeks
old
Reddit
every
Raping
end
anything
th
day
hundred
ago
centuries
months
decade
last
next

TIME


moment
hour
last
night
dollars

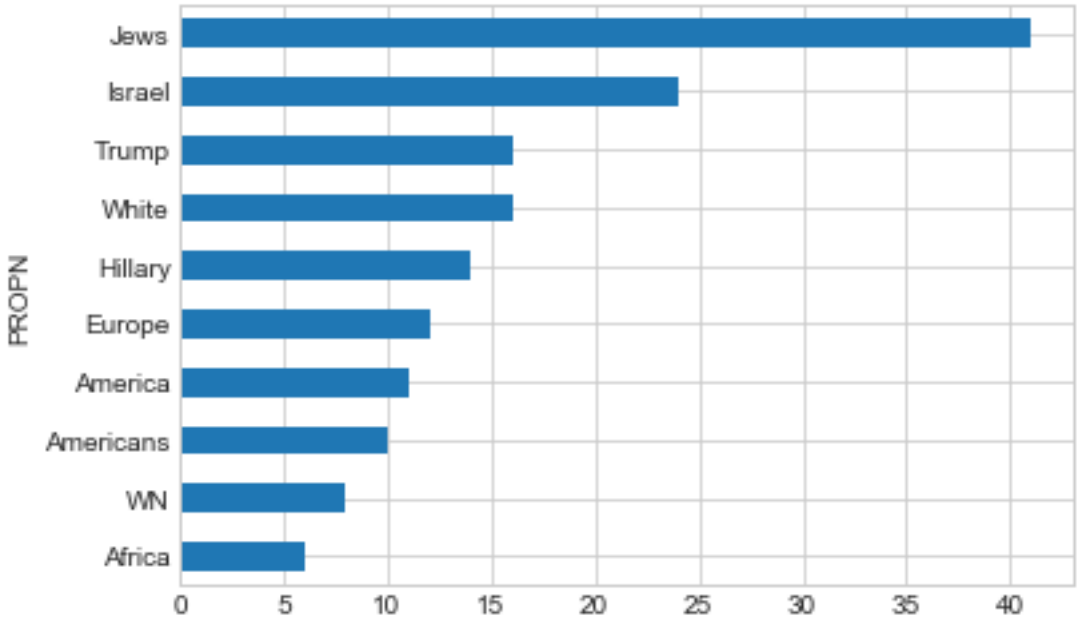
EVENT

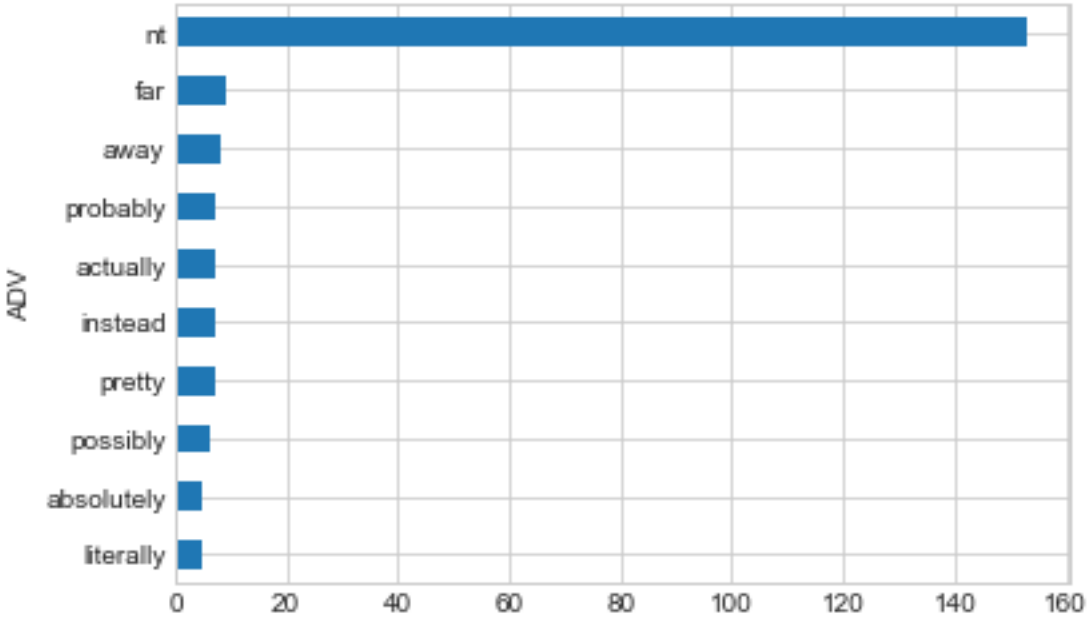
GPE

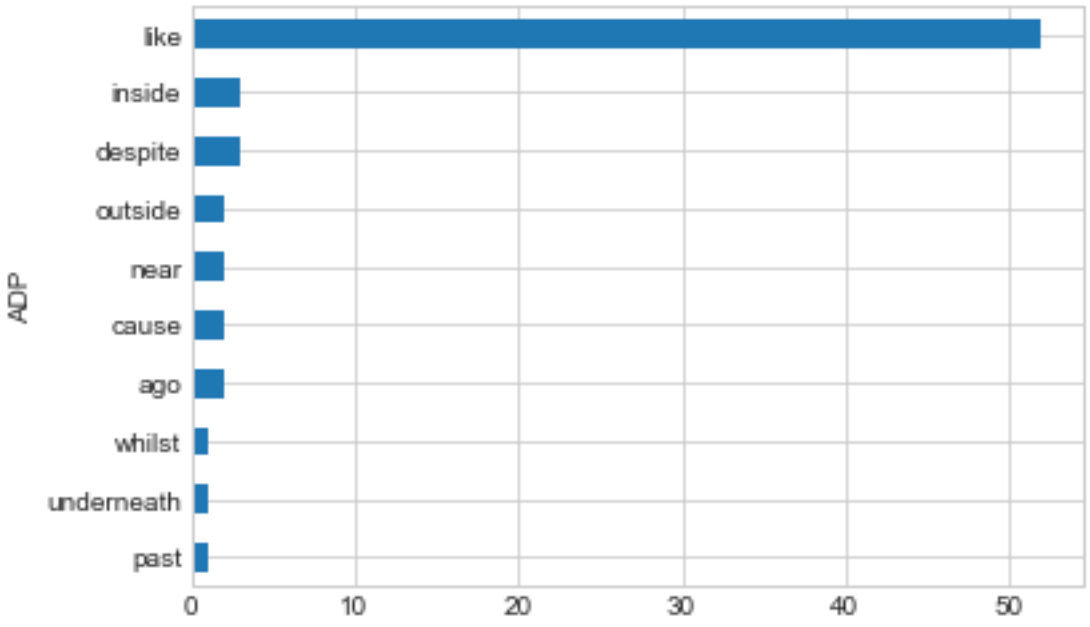
1.10 Part of Speech Frequencies

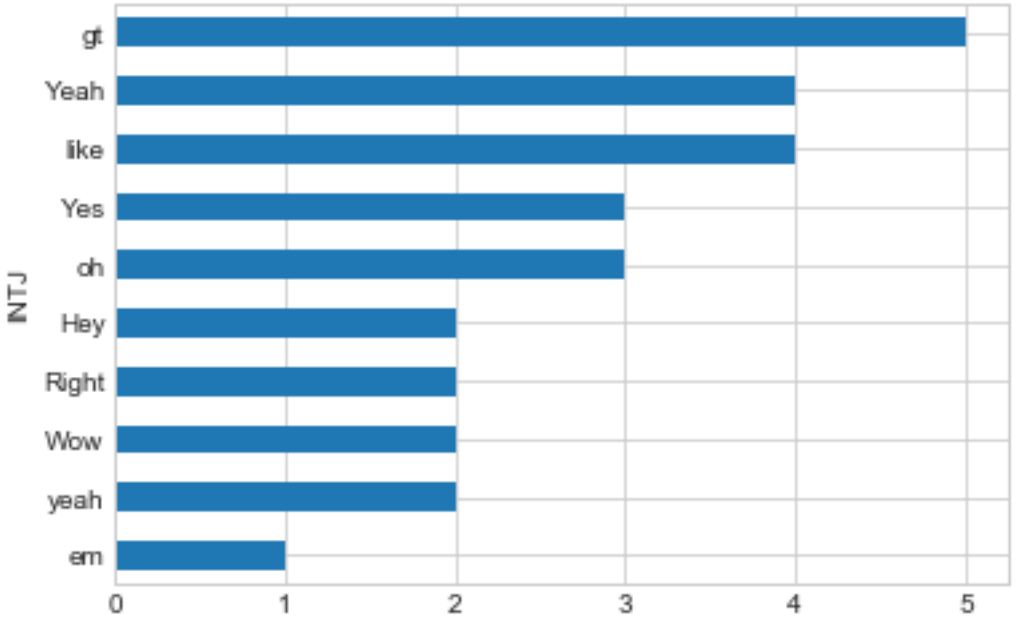
```
In [29]: all_pos = pd.DataFrame(columns=[0, 1])
         for sp in hate_only['spacy']:
             pos = [(word.pos_, word.text) for word in sp if word.is_stop == False]
             df = pd.DataFrame(pos)
             if len(df) > 0:
                 all_pos = pd.concat([all_pos, df])
```

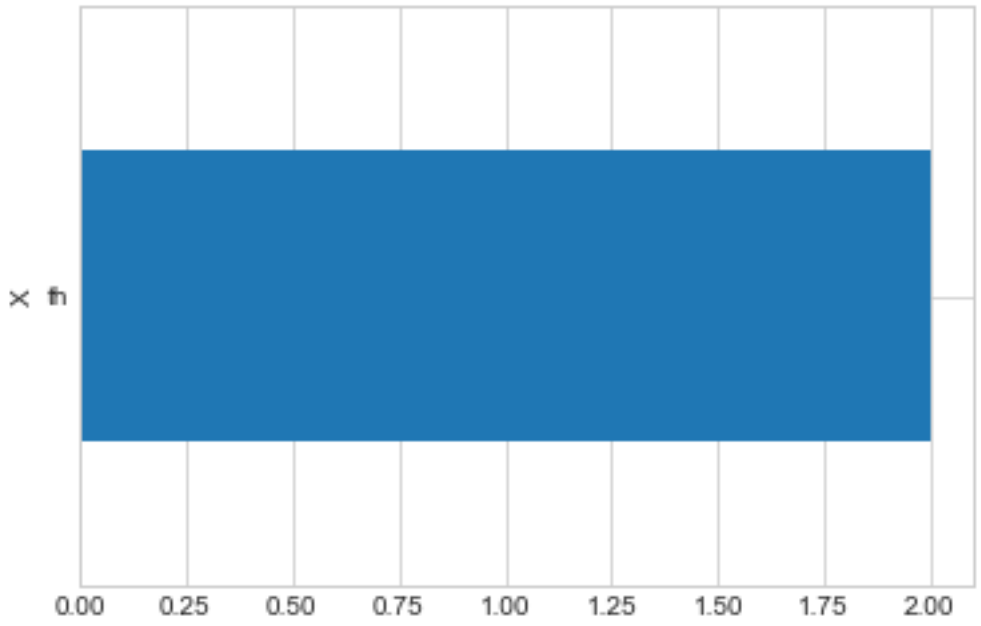
```
all_pos.columns = ["POS", "Word"]


for n in set(all_pos["POS"]):
    all_pos.groupby(["POS"])["Word"].value_counts()[n].sort_values()[-10:].plot.barh()
    plt.ylabel(n)
    plt.show()
```

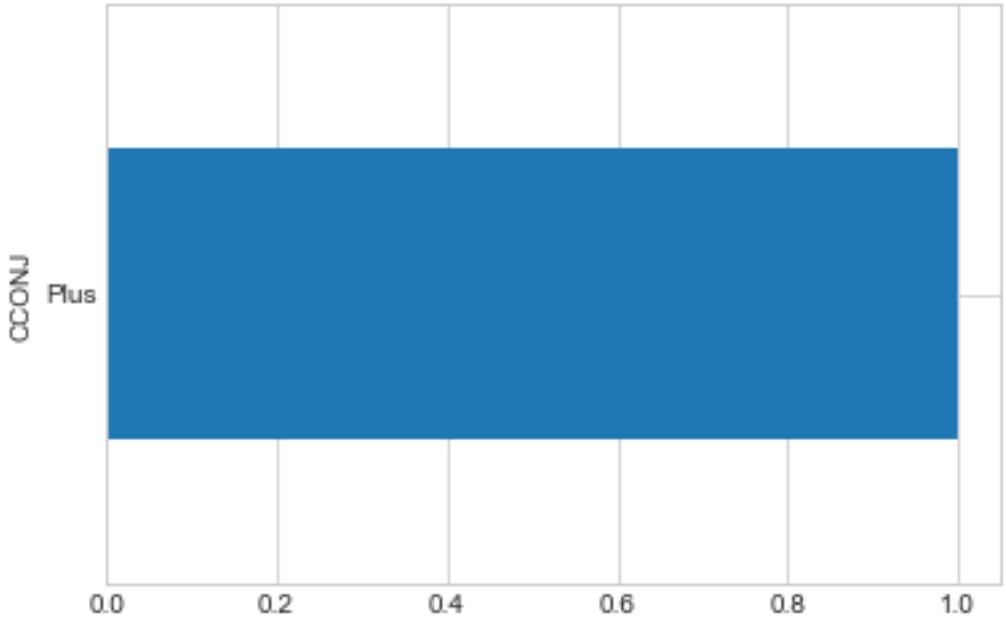


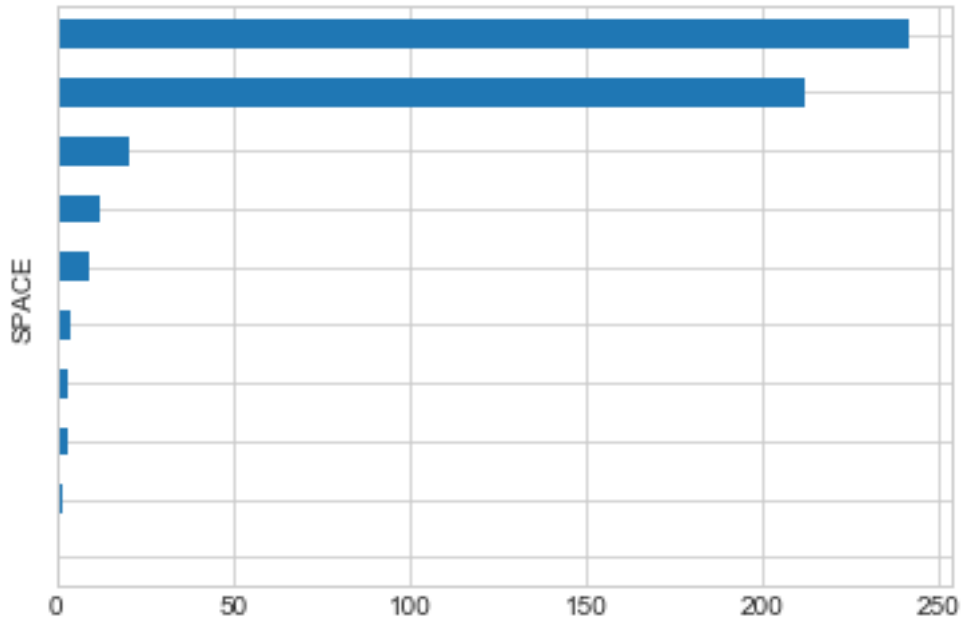


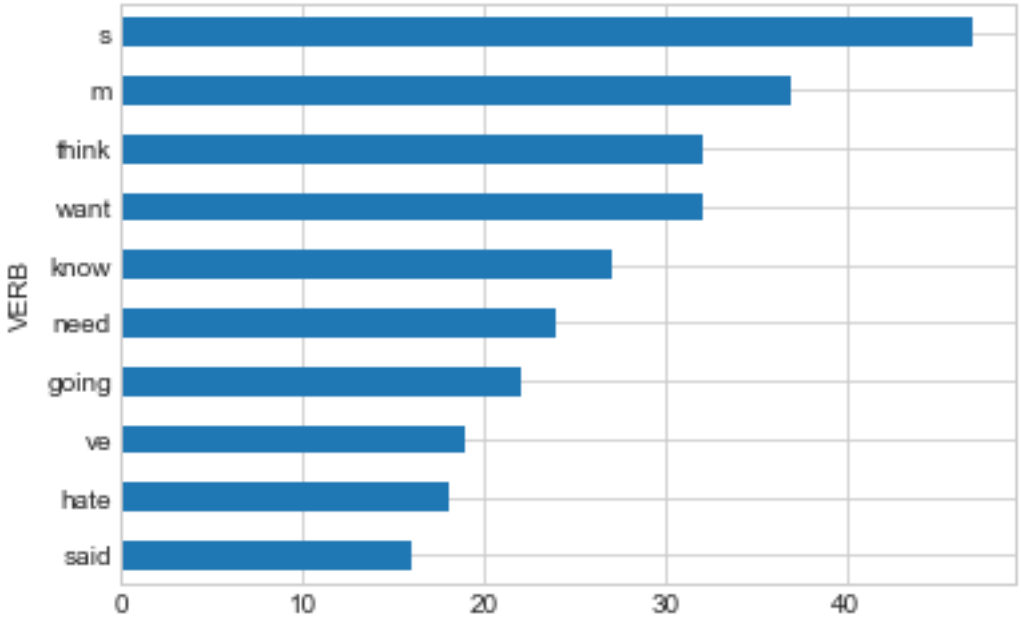

ADP
like
inside
despite
outside
near
cause
ago
whilst
underneath
past
0
10
20
30
40
50

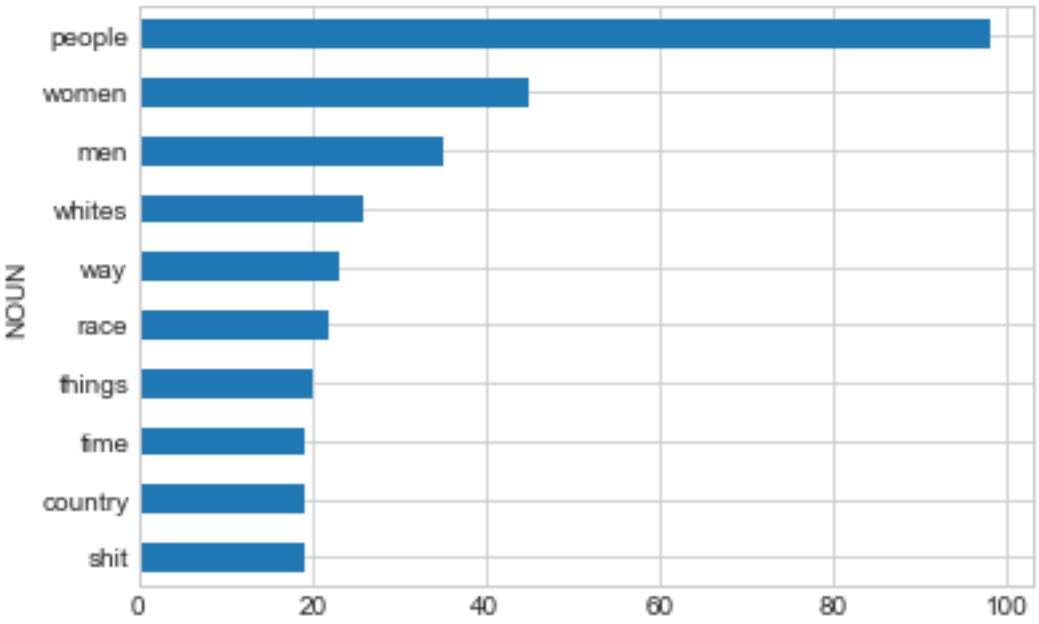

INTJ
gt
Yeah
like
Yes
oh
Hey
Right
Wow
yeah
em
0
1
2
3
4
5

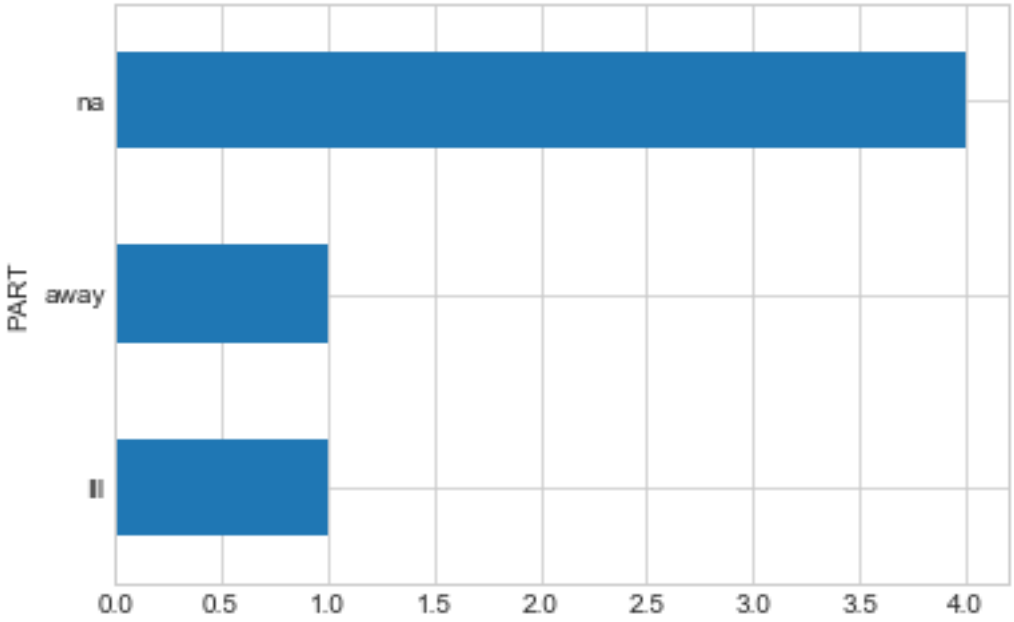

X fh
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00

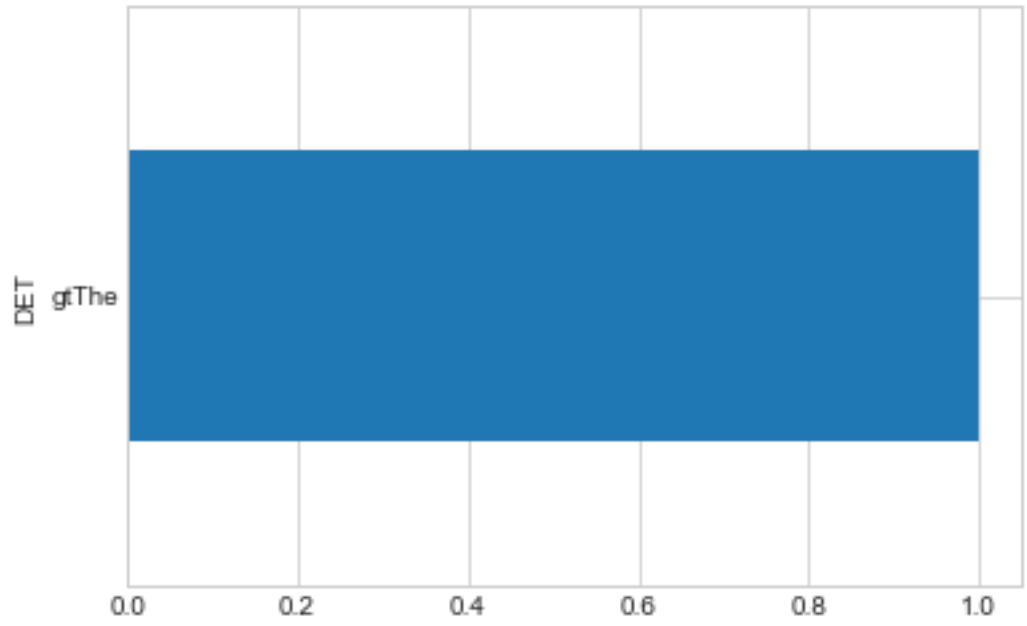

CCONJ
Plus
0.0
0.2
0.4
0.6
0.8
1.0

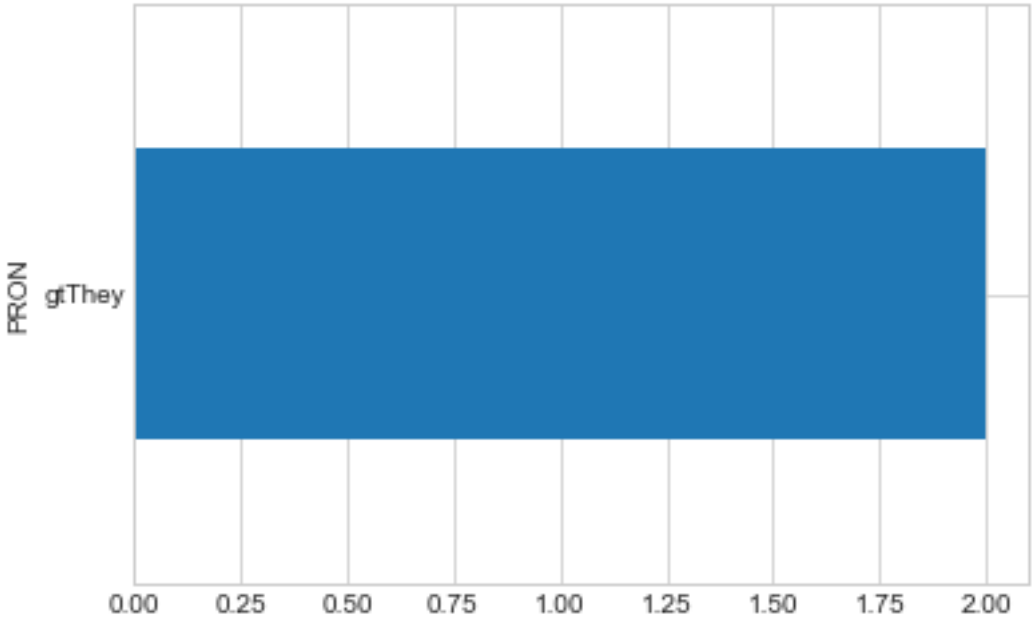

SPACE
0
50
100
150
200
250

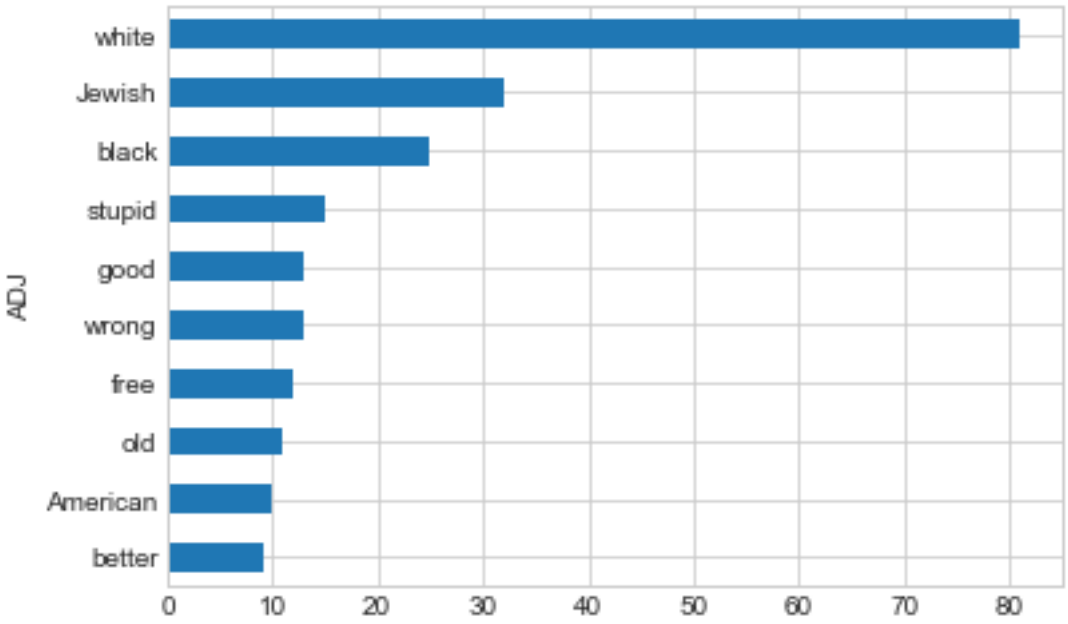

VERB
s
m
think
want
know
need
going
ve
hate
said
0
10
20
30
40

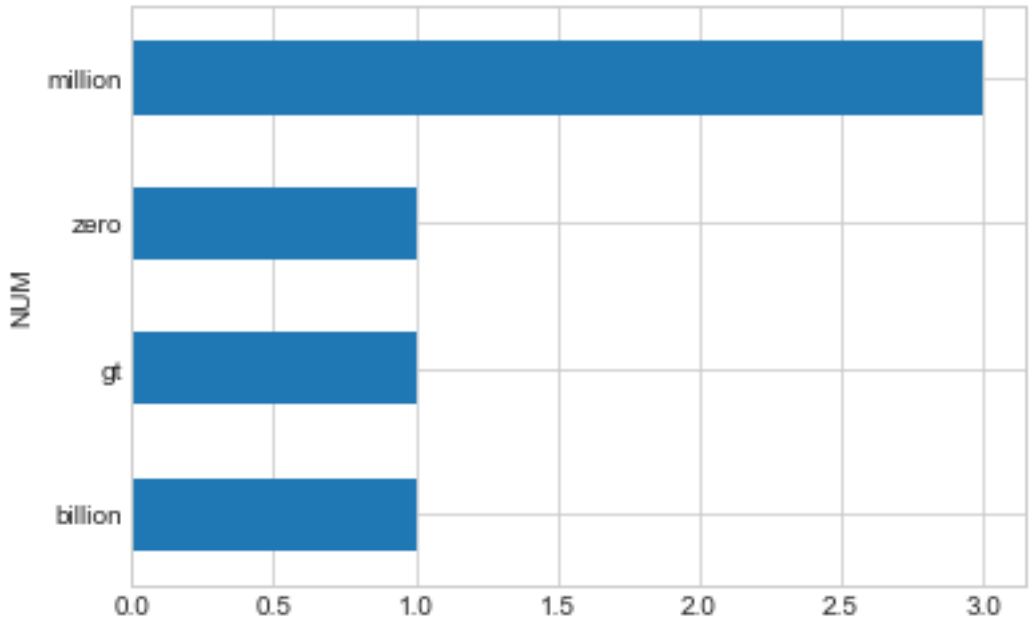

NOUN
people
women
men
whites
way
race
things
time
country
shit
0
20
40
60
80
100


PART
na
away
ll
0.0
0.5
1.0
1.5
2.0
2.5
3.0
3.5
4.0


DET
gtThe
0.0
0.2
0.4
0.6
0.8
1.0


PRON
gtThey
0.00
0.25
0.50
0.75
1.00
1.25
1.50
1.75
2.00





1.11 Part of Speech Word Clouds

```
In [30]: for n in set(all_pos["POS"]):
             print(n)
```

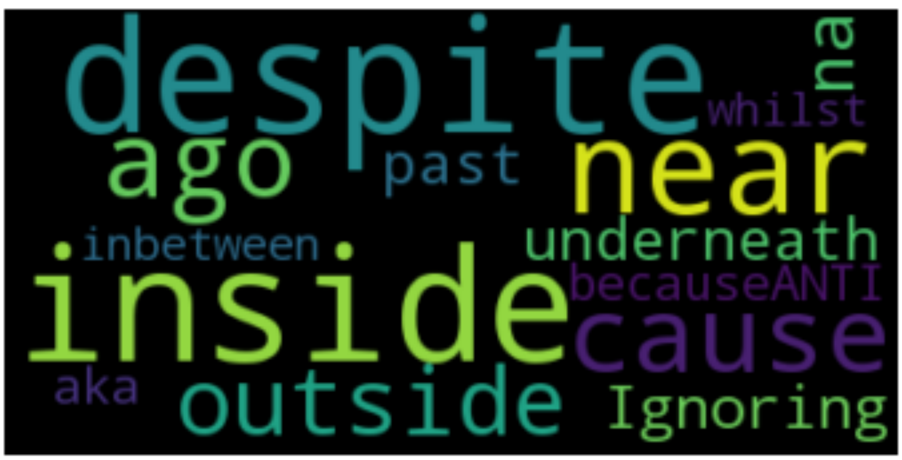
```
raw = ' '.join(list(all_pos[all_pos["POS"] == n]['Word']))
try:
    wordcloud = WordCloud().generate(raw)

    # lower max_font_size
    wordcloud = WordCloud(max_font_size=70).generate(raw)
    plt.figure()
    plt.imshow(wordcloud, interpolation="bilinear")
    plt.axis("off")
    plt.show()
except:
    pass
```

PROPN

ADV


nt
probably
possibly
seriously
actually
instead
pretty
Maybe
away
absolutely
exactly
soon
easily
far
literally
sexually
anymore
right
ahead
especially
long
ago
simply
truly
Eventually
constantly
unfortunately
naturally
Honestly
suddenly
obviously
completely
essentially
better
highly
apart
politically
specifically
freely
generally
deeply
fully
previously
likely
regardless
forever
currently
basically
longer
strongly
historically
matter
extremely
particularly
Sadly
hard
differently

ADP


despite
inside
near
ago
cause
outside
underneath
past
inbetween
whilst
na
becauseANTI
Ignoring
aka

INTJ

X
CCONJ

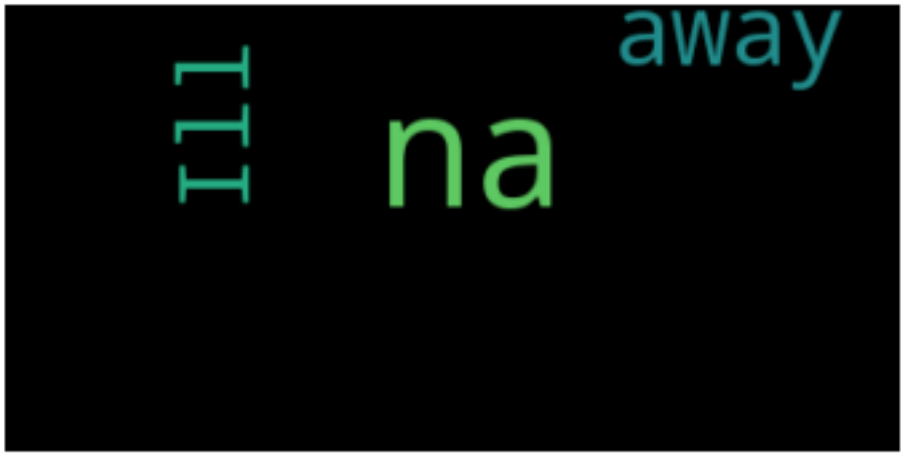
SPACE
VERB


think
hate
want
need
going
know
ve
look
come
said
fucking
believe
care
happen
got
ll
living
took
leave
exist
getting
stand
read
makes
stop
expect
agree
cause
lead
start
try
let
called
deny
trying
guess
hold
run
use
lost
given
turn
killed
live
deserve
kill
watch
coming
love
taken
saying
vote
hope
support
killing
repeating
went
play
allow
prove
forced
bring
refuse
blame
stuck
throw
feel
taught
wait
tell
spread
help
end
teach
destroy
created
considered
built
told
happened
understand
opposed
turned
enjoy
feed
left
says

NOUN

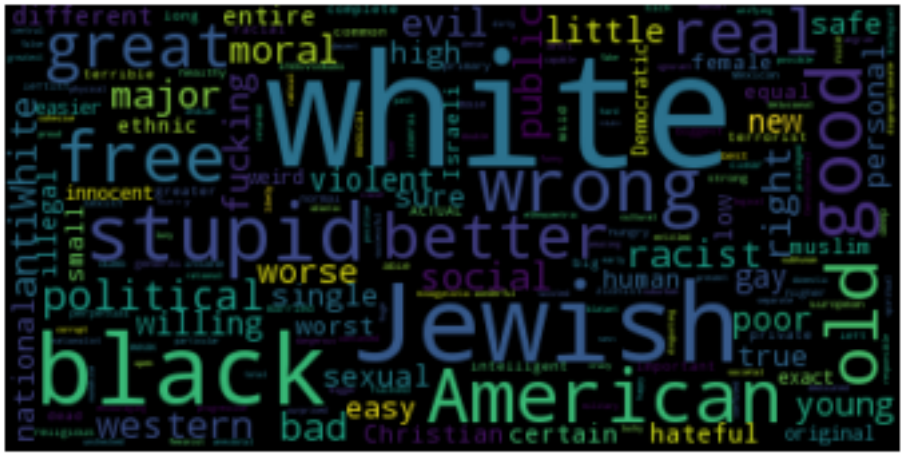

black
fact
country
word
porn
life
shit
feminist
immigration
society
way
people
jew
thing
money
day
vote
countries
culture
time
nigger
problem
gun
population
guy
trump
state
feminism
crime
race
women
lot
right
example
point
reason
year
kid
men
marriage
white
fuck
boy
issue
movement
violence
nation
world
religion
majority
media
war
stuff
speech
man
friend
immigrants
opinion
death
idea
result
government
truth
border
place
history
sex
interest
children
dog
internet

PART

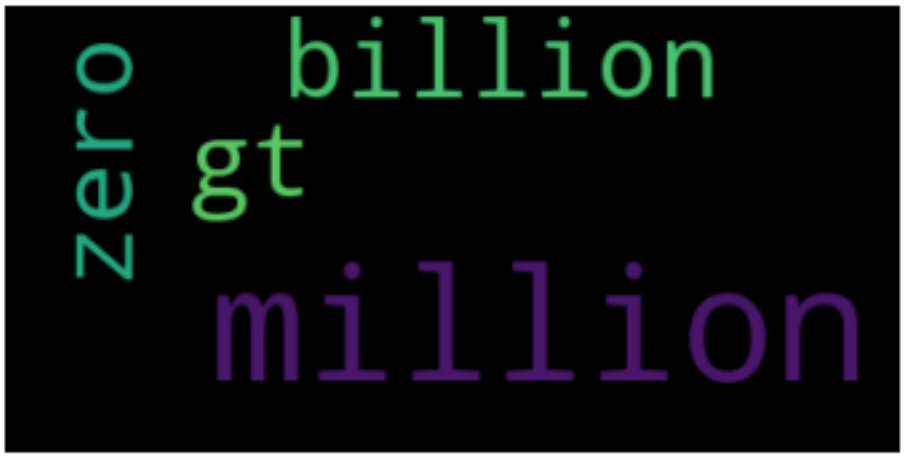


DET

PRON
ADJ



NUM



2 Variable Importance

```
In [31]: pd.read_csv('top-correlation-unigram.csv')
```

Out[31]:

	Rank	variable	corr	p_value	adj_p_value
0	1	gm1_jew	0.202	0.000000	0.000000
1	2	gm1_white	0.142	0.000000	0.000000
2	3	gm1_hate	0.119	0.000000	0.000000

3	4	gm1_women	0.107	0.000000	0.000000
4	5	gm1_black	0.106	0.000000	0.000000
5	6	gm1_fuck	0.103	0.000000	0.000000
6	7	gm1_problem	0.086	0.000000	0.000000
7	8	gm1_race	0.083	0.000000	0.000000
8	9	gm1_men	0.082	0.000000	0.000000
9	10	gm1_peopl	0.082	0.000000	0.000000
10	11	gm1_countri	0.081	0.000000	0.000000
11	12	gm1_societi	0.080	0.000000	0.000000
12	13	gm1_back	0.080	0.000000	0.000000
13	14	gm1_need	0.078	0.000000	0.000000
14	15	gm1_like	0.078	0.000000	0.000000
15	16	gm1_cultur	0.073	0.000000	0.000000
16	17	gm1_immigr	0.072	0.000000	0.000000
17	18	gm1_feminist	0.071	0.000000	0.000000
18	19	gm1_nation	0.071	0.000000	0.000000
19	20	gm1_old	0.068	0.000000	0.000000
20	21	gm1_around	0.066	0.000000	0.000000
21	22	gm1_crime	0.064	0.000000	0.000000
22	23	gm1_stupid	0.063	0.000000	0.000001
23	24	gm1_rape	0.062	0.000000	0.000001
24	25	gm1_come	0.062	0.000000	0.000001
25	26	gm1_american	0.061	0.000000	0.000001
26	27	gm1_ago	0.060	0.000000	0.000002
27	28	gm1_shit	0.060	0.000000	0.000002
28	29	gm1_wouldnt	0.060	0.000000	0.000003
29	30	gm1_year	0.057	0.000001	0.000008
30	31	gm1_just	0.056	0.000001	0.000015
31	32	gm1_instead	0.055	0.000001	0.000016
32	33	gm1_children	0.055	0.000002	0.000023
33	34	gm1_face	0.054	0.000003	0.000031
34	35	gm1_becaus	0.053	0.000003	0.000038
35	36	gm1_wors	0.051	0.000009	0.000097
36	37	gm1_will	0.051	0.000010	0.000097
37	38	gm1_one	0.051	0.000010	0.000097
38	39	gm1_left	0.050	0.000012	0.000115
39	40	gm1_want	0.050	0.000014	0.000137
40	41	gm1_never	0.049	0.000017	0.000156
41	42	gm1_alway	0.049	0.000019	0.000175
42	43	gm1_etc	0.047	0.000036	0.000319
43	44	gm1_major	0.047	0.000038	0.000330
44	45	gm1_thing	0.046	0.000059	0.000505
45	46	gm1_fact	0.046	0.000061	0.000506
46	47	gm1_kid	0.046	0.000063	0.000511
47	48	gm1_dont	0.045	0.000085	0.000679
48	49	gm1_theyr	0.044	0.000127	0.000994
49	50	gm1_talk	0.044	0.000140	0.001073

```
In [32]: pd.read_csv('top-correlation-bigram.csv')
```

Out[32]:

	Rank	variable	corr	p_value	adj_p_value
0	1	of them	0.062608	4.742833e-08	0.000010
1	2	and they	0.059344	2.272752e-07	0.000022
2	3	is just	0.058554	3.282093e-07	0.000022
3	4	they are	0.058068	4.104795e-07	0.000022
4	5	to be	0.057494	5.333764e-07	0.000023
5	6	need to	0.056955	6.806328e-07	0.000024
6	7	all the	0.056638	7.849197e-07	0.000024
7	8	we are	0.048805	2.088954e-05	0.000569
8	9	there are	0.047276	3.760382e-05	0.000911
9	10	peopl are	0.044692	9.771996e-05	0.002003
10	11	fact that	0.044598	1.010773e-04	0.002003
11	12	they can	0.044099	1.208220e-04	0.002026
12	13	is the	0.044099	1.208346e-04	0.002026
13	14	to get	0.043105	1.714744e-04	0.002670
14	15	becaus they	0.042249	2.305190e-04	0.003350
15	16	are a	0.041454	3.019636e-04	0.004114
16	17	their own	0.039521	5.710043e-04	0.007300
17	18	they have	0.039282	6.166813e-04	0.007300
18	19	that is	0.039185	6.362602e-04	0.007300
19	20	what they	0.038775	7.251394e-04	0.007904
20	21	out of	0.037482	1.086368e-03	0.010918
21	22	have a	0.037436	1.101846e-03	0.010918
22	23	no one	0.036605	1.419547e-03	0.013455
23	24	as a	0.036296	1.557723e-03	0.014149
24	25	the fact	0.036157	1.624187e-03	0.014163
25	26	have the	0.033066	3.951955e-03	0.033136
26	27	on the	0.032295	4.881705e-03	0.037923
27	28	and then	0.032165	5.056760e-03	0.037923
28	29	want to	0.032057	5.205982e-03	0.037923
29	30	can be	0.032048	5.218712e-03	0.037923
30	31	the world	0.031714	5.708791e-03	0.040146
31	32	i know	0.031473	6.086275e-03	0.041463
32	33	talk about	0.031048	6.810343e-03	0.043783
33	34	was a	0.031037	6.828477e-03	0.043783
34	35	of the	0.030743	7.374118e-03	0.045693
35	36	should be	0.030655	7.545670e-03	0.045693
36	37	and the	0.029759	9.498032e-03	0.055961
37	38	of it	0.029337	1.056304e-02	0.058881
38	39	you know	0.029332	1.057570e-02	0.058881
39	40	dont have	0.029213	1.089700e-02	0.058881
40	41	tri to	0.029148	1.107386e-02	0.058881
41	42	for a	0.028626	1.260191e-02	0.065410
42	43	to a	0.028258	1.378623e-02	0.069893
43	44	what is	0.027527	1.643972e-02	0.081451
44	45	of what	0.026898	1.906687e-02	0.092368

45	46	go to	0.026098	2.294097e-02	0.108720
46	47	and i	0.025922	2.387725e-02	0.110750
47	48	the onli	0.025622	2.555450e-02	0.116060
48	49	be a	0.025400	2.686017e-02	0.119500
49	50	look at	0.024862	3.026181e-02	0.129293