Table of Contents

Modern Text Mining with Python (Part 2 of 5): Data Exploration with Pandas

This notebook contains the code examples for the blog post on Medium.com.

To get it running, you will need the SQLite Database `selfposts.db`, which was created in the first notebook of this series.

Library Import & Settings

In [ ]:
```
# filter warnings on depreciation etc.
import warnings
warnings.filterwarnings("ignore")
```

In [ ]:
```
# import pandas, numpy
import pandas as pd
import numpy as np

# adjust pandas display
pd.options.display.max_columns = 30
pd.options.display.max_rows = 100
pd.options.display.float_format = '{:.2f}'.format
pd.options.display.precision = 2
pd.options.display.max_colwidth = -1
```

```
In [ ]: # Import matplotlib and seaborn and adjust some defaults
%matplotlib inline
%config InlineBackend.figure_format = 'svg'

from matplotlib import pyplot as plt
plt.rcParams['figure.dpi'] = 100

import seaborn as sns
sns.set_style("whitegrid")
```

Load Data into Pandas

```
In [ ]: import sqlite3

sql = """
SELECT p.*,
       c.category_1 as category, c.category_2 as subcategory
FROM posts_nlp p, categories c
WHERE p.subreddit = c.subreddit
"""
with sqlite3.connect('/home/deepcode/Downloads/selfposts.db') as con:
    df = pd.read_sql_query(sql, con)
```

Basic Properties of the Dataset

```
In [ ]: # list column names and datatypes
df.dtypes
```

```
Out[ ]: index                    int64
id                       object
subreddit                object
title                    object
selftext                 object
selftext_clean           object
selftext_lemma           object
selftext_nouns           object
selftext_adjectives      object
selftext_verbs           object
selftext_nav             object
no_tokens                float64
category                 object
subcategory              object
dtype: object
```

```
In [ ]: # select a sample of some data frame columns
df[['id', 'subreddit', 'title', 'selftext_clean']] \
  .sample(2, random_state=42)
```

Out[ ]:

	id	subreddit	title	selftext_clean
1501	4rwc67	AskLiteraryStudies	ideas for a tragedy survey	Hello folks! I am due to be teaching an undergrad (no pre-reqs) survey of European tragedy next semester, and I'm looking for some ideas of primary texts to put on the syllabus. The course has a pretty set component in the form of classical texts, but the rest of it is freer.I would especially welcome any suggestions or tips on 1) renaissance/baroque plays, 2) something from Calderon, 3) German classicism, 4) modern drama. If anyone can think of plays (the definition of "tragedy" can be fudged) by women or ethnic writers, that would be really helpful.This is the first time I'm teaching something like this, and it's been dropped in my lap rather suddenly, so I would be very grateful for suggestions from the more experienced teachers among you! Thanks in advance!
2586	7gdvsv	breastfeeding	Question about a very hungry child	Hey guys,My kid is 9 months old and still wakes up every 2 hours throughout the night to breastfeed. At this point, I co-sleep with her to get some sleep through the night and now I really like the arrangement. Is there a way or technique I could use where I could continue to co-sleep but the baby doesn't feed that much at night ? She is 10% weight and is EBFShe also started solids at 6 months and eats three meals a day at this point.

In [ ]:

```
# length of a dataframe
len(df)
```

Out[ ]:

```
5000
```

In [ ]:

```
# number of values per column
df.count()
```

Out[ ]:

```
index                  5000
id                     5000
subreddit              5000
title                  5000
selftext               5000
selftext_clean         5000
selftext_lemma         5000
selftext_nouns         5000
selftext_adjectives    5000
selftext_verbs         5000
selftext_nav           5000
no_tokens              5000
category               5000
subcategory            5000
dtype: int64
```

In [ ]:

```
# size info, including memory consumption
df.info(memory_usage='deep')
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 5000 entries, 0 to 4999
Data columns (total 14 columns):
 #   Column               Non-Null Count  Dtype  
---  ------               --------------  -----  
 0   index                5000 non-null   int64  
 1   id                   5000 non-null   object 
 2   subreddit            5000 non-null   object 
 3   title                5000 non-null   object 
 4   selftext             5000 non-null   object 
 5   selftext_clean       5000 non-null   object 
 6   selftext_lemma       5000 non-null   object 
 7   selftext_nouns       5000 non-null   object 
 8   selftext_adjectives  5000 non-null   object 
 9   selftext_verbs       5000 non-null   object 
 10  selftext_nav         5000 non-null   object 
 11  no_tokens            5000 non-null   float64
 12  category             5000 non-null   object 
 13  subcategory          5000 non-null   object 
dtypes: float64(1), int64(1), object(12)
memory usage: 20.0 MB
```

Exploring Column Summaries

The pandas `describe` method computes statistical summaries for each of the columns of a dataframe. The results are different for categorical and numerical features.

Summary for Categorical Features

In [ ]:
```
columns = [col for col in df.columns if not col.startswith('self')]
columns
```

Out[ ]:
```
['index', 'id', 'subreddit', 'title', 'no_tokens', 'category', 'subcategor
y']
```

In [ ]:
```
# describe categorical columns of type np.object
df[['category', 'subcategory', 'subreddit']] \
  .describe(include=object) \
  .transpose()
```

Out[ ]:

	count	unique	top	freq
category	5000	39	video_game	488
subcategory	5000	1003	logo	15
subreddit	5000	1003	logorequests	15

In [ ]:
```
df['subreddit'].value_counts()[:10]
```

```
Out[ ]: logorequests    15
        samuraijack     12
        RocketLeague    12
        Tinder          12
        backpacking     12
        Tekken          12
        solar           11
        gigantic        11
        DungeonWorld    11
        speedrun        11
        Name: subreddit, dtype: int64
```

Summary for Numerical Features

```
In [ ]: # describe numerical columns
        df.describe().transpose()
```

Out[ ]:

	count	mean	std	min	25%	50%	75%	max
index	5000.00	2499.50	1443.52	0.00	1249.75	2499.50	3749.25	4999.00
no_tokens	5000.00	163.90	132.00	20.00	79.00	117.00	193.00	943.00

Exploring Text Categories

```
In [ ]: # number of unique values = count distinct
        df['category'].nunique()
```

```
Out[ ]: 39
```

```
In [ ]: # group by category, count distinct subreddits and posts
        cat_df = df.groupby('category') \
                   .agg({'subreddit': pd.Series.nunique,
                         'id': pd.Series.count}) \
                   .rename(columns={'subreddit': 'num_subreddits',
                                    'id': 'num_posts'}) \
                   .sort_values('num_subreddits', ascending=False)

        # show top 5 records
        cat_df.head(5)
```

Out[ ]:

	num_subreddits	num_posts
category		
video_game	97	488
tv_show	68	323
health	57	290
profession	56	257
software	51	244

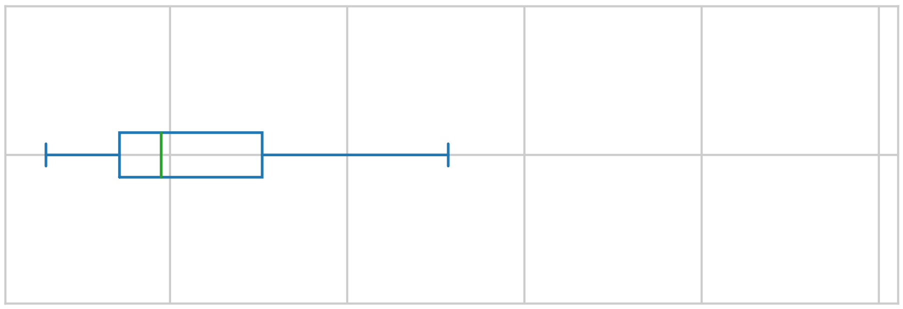
```
In [ ]: cat_df.describe()
```

Out[ ]:

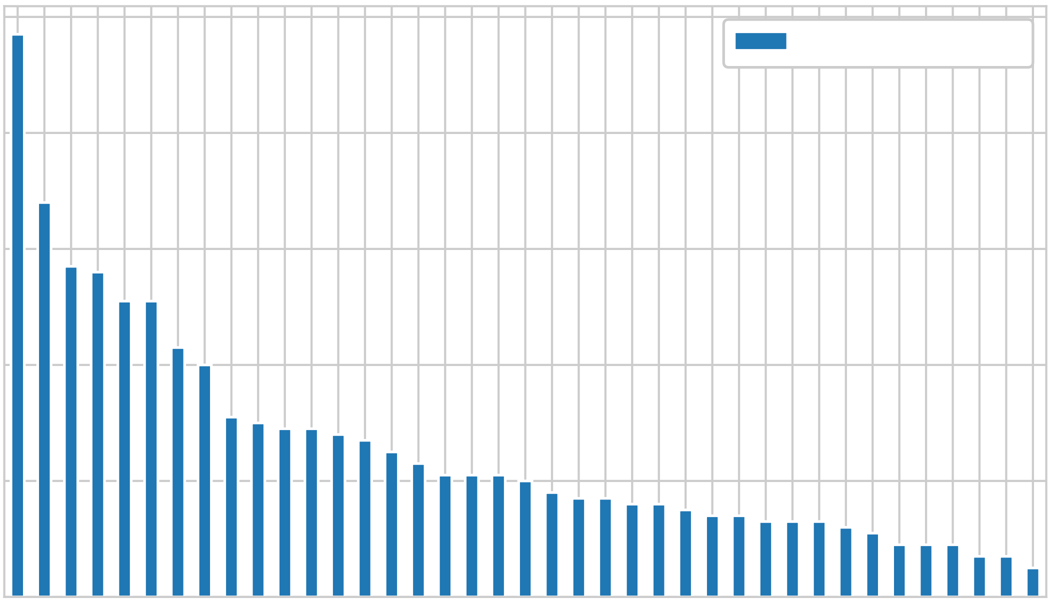
	num_subreddits	num_posts
count	39.00	39.00
mean	25.72	128.21
std	19.45	94.74
min	5.00	30.00
25%	13.00	71.50
50%	20.00	95.00
75%	29.50	152.00
max	97.00	488.00

Visualizing Frequency Distributions

```
In [ ]: # horizontal boxplot of a dataframe column
        cat_df[['num_posts']].plot(kind='box', vert=False, figsize=(6, 2));
```



```
In [ ]: # bar chart of a dataframe column
        cat_df[['num_subreddits']].plot(kind='bar', figsize=(7,4));
```



Exploring Word Frequencies

In [ ]:

```python
# create a data frame slice
sub_df = df[df['subreddit']=='TheSimpsons']

# sample cleaned text and tokens tagged as nouns
sub_df[['selftext_clean', 'selftext_nouns']].sample(2)
```

Out[ ]:

	selftext_clean	selftext_nouns
3098	Just saw on Youtube as a 3 min clip: Homer finds a antique green car in his garage (was green) Marge was upset that homer didn't do his chores so he did it on "laminated paper" he took car everywhere, that is where the episode had ended and could not find again help please	Youtube min clip Homer car garage Marge homer chore paper car episode
823	Okay, I've been researching for two hours and I can't find this episode that I remember. It pops in my head every once in a while and I've never been able to find it, you guys are my last hope. I'm pretty sure it was within the last... 6 years? maybe more. All that I remember is Lisa meets a new, possibly weird/creepy boy who I thiiink has been living at the school? He may have lost his parents. I'm pretty sure I remember him being into creepy stuff but it being a full episode and not a treehouse of horror or anything. If anyone has any idea what I'm talking about I'd be eternally grateful, I have no idea why but this one particular episode eludes me. (I also swear I remember him being voiced by Elijah Wood but apparently he's never been a guest on the show ¯\ (ツ) /¯)	hour episode head while guy hope year Lisa boy school parent stuff episode treehouse horror idea idea episode Elijah Wood guest show ¯\ ツ

Creating a List of Tokens from a List of Documents

In [ ]:
```
def my_tokenizer(text):
    return text.split() if text != None else []
```

In [ ]:
```
# transform list of documents into a single list of tokens
tokens = sub_df.selftext_nouns.map(my_tokenizer).sum()
```

In [ ]:
```
print(tokens[:200])
```

```
['episode', 'time', 'other', 'instance', 'proposal', 'Putlocker', 'week',
'tv', 'Bart', 'Thanksgiving', 'episode', 'other', 'show', 'girlfriend', 'ki
d', 'point', 'go', 'show', 'movie?there', 'thing', 'reference', 'continuit
y', 'quality', 'reference', 'back', 'movie', 'season', 'Mr.', 'Burns', 'jok
e', 'consequence', 'callback', 'order', 'problem', 'quality', 'matching',
'movie', 'lot', 'episode', 'show', 'point', 'season', 'dip', 'movie', 'en
d', 'cap', 'movie', 'cartooniness', 'vibe', 'show', 'hour', 'episode', 'hea
d', 'while', 'guy', 'hope', 'year', 'Lisa', 'boy', 'school', 'parent', 'stu
ff', 'episode', 'treehouse', 'horror', 'idea', 'idea', 'episode', 'Elijah',
'Wood', 'guest', 'show', '¯\\', 'ツ', 'post', 'time', 'episode', 'episode',
'Homer', 'City', 'New', 'York', 'season', 'episode', 'Youtube', 'min', 'cli
p', 'Homer', 'car', 'garage', 'Marge', 'homer', 'chore', 'paper', 'car', 'e
pisode', 'name', 'episode', 'Homer', 'bully', 'Kearney', 'joke', 'Homer',
'line', 'guy', 'thank', 'episode', 'Simpsons', 'episode', 'Lisa', 'Gaga',
'braindead', 'STUPID', 'paddling', 'episode', 'scene', 'Homer', 'Gill', 'ca
r', 'dog', 'way', 'show', 'minute', 'episode', 'court', 'minute', 'court',
'thing', 'Homer', 'dog', 'Springfield', 'dog', 'treehouse', 'horror', 'do
g', 'control', 'city', 'town', 'Gill', 'dog', 'Marge', 'Chihuahua', 'Deus',
'Ex', 'Machina', 'Marge', 'Chihuahua', 'death', 'moronic', 'joke', 'episod
e', 'reference', 'Turbo', 'WHY', 'REALLY', 'ASK', 'fucking', 'episode', 'tu
rbo', 'joke', 'month', 'production', 'time', 'hell', 'Turbo', 'month', 'tur
bo', 'episode', 'Treehouse', 'Horror', 'breakneck', 'pace', 'Treehouse', 'H
orror', 'heart', 'episode', 'GOD']
```

Counting Frequencies with a Counter

In [ ]:
```
from collections import Counter

counter = Counter(tokens)
counter.most_common(20)
```

Out[ ]:
```
[('episode', 19),
 ('show', 6),
 ('Homer', 6),
 ('dog', 5),
 ('movie', 4),
 ('joke', 4),
 ('time', 3),
 ('reference', 3),
 ('season', 3),
 ('car', 3),
 ('Marge', 3),
 ('other', 2),
 ('point', 2),
 ('thing', 2),
 ('quality', 2),
 ('guy', 2),
 ('Lisa', 2),
 ('treehouse', 2),
 ('horror', 2),
 ('idea', 2)]
```

In [ ]:
```python
df.category.unique()
```

Out[ ]:
```
array(['writing/stories', 'tv_show', 'autos', 'hardware/tools',
       'electronics', 'video_game', 'crypto', 'sports', 'hobby',
       'appearance', 'card_game', 'drugs', 'advice/question',
       'social_group', 'anime/manga', 'sex/relationships', 'software',
       'health', 'other', 'animals', 'arts', 'programming', 'rpg',
       'books', 'parenting', 'education', 'company/website', 'profession',
       'music', 'politics/viewpoint', 'stem', 'travel', 'geo',
       'religion/supernatural', 'board_game', 'movies', 'food/drink',
       'finance/money', 'meta'], dtype=object)
```

In [ ]:
```python
print([t[0] for t in counter.most_common(200)])
```

```
['episode', 'show', 'Homer', 'dog', 'movie', 'joke', 'time', 'reference',
'season', 'car', 'Marge', 'other', 'point', 'thing', 'quality', 'guy', 'Lis
a', 'treehouse', 'horror', 'idea', 'Gill', 'minute', 'court', 'Chihuahua',
'Turbo', 'turbo', 'month', 'Treehouse', 'Horror', 'instance', 'proposal',
'Putlocker', 'week', 'tv', 'Bart', 'Thanksgiving', 'girlfriend', 'kid', 'g
o', 'movie?there', 'continuity', 'back', 'Mr.', 'Burns', 'consequence', 'ca
llback', 'order', 'problem', 'matching', 'lot', 'dip', 'end', 'cap', 'carto
oniness', 'vibe', 'hour', 'head', 'while', 'hope', 'year', 'boy', 'school',
'parent', 'stuff', 'Elijah', 'Wood', 'guest', '¯\\', 'ツ', 'post', 'City',
'New', 'York', 'Youtube', 'min', 'clip', 'garage', 'homer', 'chore', 'pape
r', 'name', 'bully', 'Kearney', 'line', 'thank', 'Simpsons', 'Gaga', 'brain
dead', 'STUPID', 'paddling', 'scene', 'way', 'Springfield', 'control', 'cit
y', 'town', 'Deus', 'Ex', 'Machina', 'death', 'moronic', 'WHY', 'REALLY',
'ASK', 'fucking', 'production', 'hell', 'breakneck', 'pace', 'heart', 'GO
D']
```

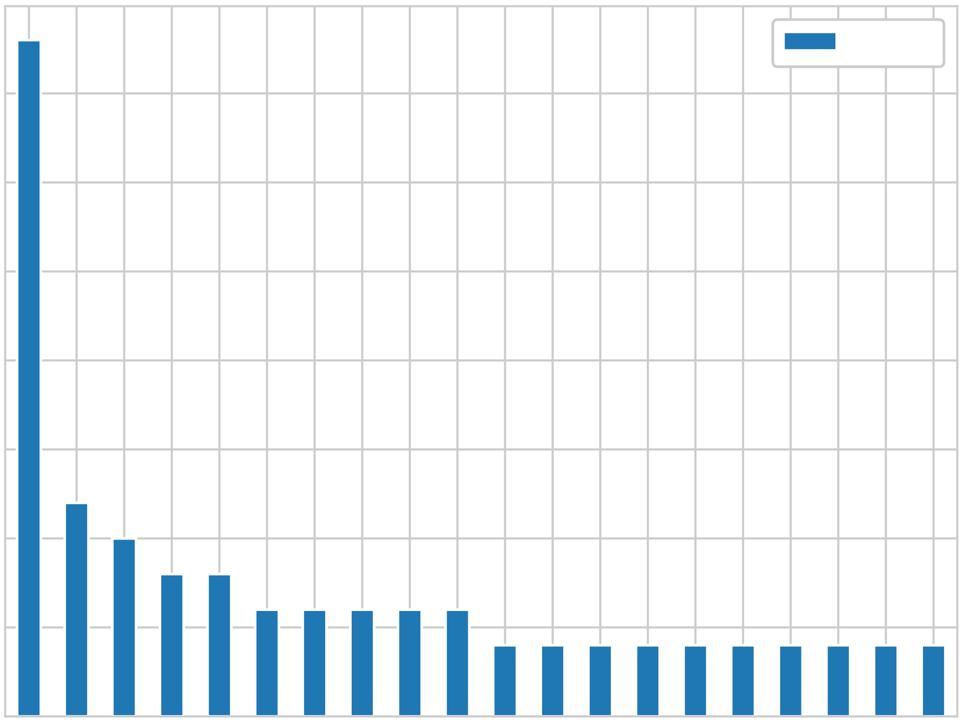
In [ ]:
```python
from spacy.lang.en.stop_words import STOP_WORDS

def remove_stopwords(tokens):
    """Remove stopwords from a list of tokens."""
    return [t for t in tokens if t not in STOP_WORDS]
```

```
# rebuild counter
counter = Counter(remove_stopwords(tokens))
```

```
2023-03-19 13:51:03.468638: I tensorflow/core/platform/cpu_feature_guard.c
c:193] This TensorFlow binary is optimized with oneAPI Deep Neural Network
Library (oneDNN) to use the following CPU instructions in performance-criti
cal operations:  AVX2 FMA
To enable them in other operations, rebuild TensorFlow with the appropriate
compiler flags.
2023-03-19 13:51:03.635675: W tensorflow/compiler/xla/stream_executor/platf
orm/default/dso_loader.cc:64] Could not load dynamic library 'libcudart.so.
11.0'; dlerror: libcudart.so.11.0: cannot open shared object file: No such
file or directory
2023-03-19 13:51:03.635710: I tensorflow/compiler/xla/stream_executor/cuda/
cudart_stub.cc:29] Ignore above cudart dlerror if you do not have a GPU set
up on your machine.
2023-03-19 13:51:04.316902: W tensorflow/compiler/xla/stream_executor/platf
orm/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer.s
o.7'; dlerror: libnvinfer.so.7: cannot open shared object file: No such fil
e or directory
2023-03-19 13:51:04.316998: W tensorflow/compiler/xla/stream_executor/platf
orm/default/dso_loader.cc:64] Could not load dynamic library 'libnvinfer_pl
ugin.so.7'; dlerror: libnvinfer_plugin.so.7: cannot open shared object fil
e: No such file or directory
2023-03-19 13:51:04.317004: W tensorflow/compiler/tf2tensorrt/utils/py_util
s.cc:38] TF-TRT Warning: Cannot dlopen some TensorRT libraries. If you woul
d like to use Nvidia GPU with TensorRT, please make sure the missing librar
ies mentioned above are installed properly.
2023-03-19 13:51:05.057663: W tensorflow/compiler/xla/stream_executor/platf
orm/default/dso_loader.cc:64] Could not load dynamic library 'libcuda.so.
1'; dlerror: libcuda.so.1: cannot open shared object file: No such file or
directory
2023-03-19 13:51:05.057688: W tensorflow/compiler/xla/stream_executor/cuda/
cuda_driver.cc:265] failed call to cuInit: UNKNOWN ERROR (303)
2023-03-19 13:51:05.057712: I tensorflow/compiler/xla/stream_executor/cuda/
cuda_diagnostics.cc:156] kernel driver does not appear to be running on thi
s host (dc-central): /proc/driver/nvidia/version does not exist
```

In [ ]:

```
# convert list of tuples into data frame
freq_df = pd.DataFrame.from_records(counter.most_common(20),
                                    columns=['token', 'count'])

# create bar plot
freq_df.plot(kind='bar', x='token');
```



Using Word Clouds

In [ ]:
```
%matplotlib inline
import matplotlib.pyplot as plt
```

In [ ]:
```
%%capture
%pip install wordcloud
```

In [ ]:
```
from wordcloud import WordCloud

def wordcloud(counter):
    """A small wordloud wrapper"""
    wc = WordCloud(width=1200, height=800,
                   background_color="white",
                   max_words=200)
    wc.generate_from_frequencies(counter)

    # Plot
    fig=plt.figure(figsize=(6, 4))
    plt.imshow(wc, interpolation='bilinear')
    plt.axis("off")
    plt.tight_layout(pad=0)
    plt.show()
```

In [ ]:
```
# create wordcloud
```

```
wordcloud(counter)
```

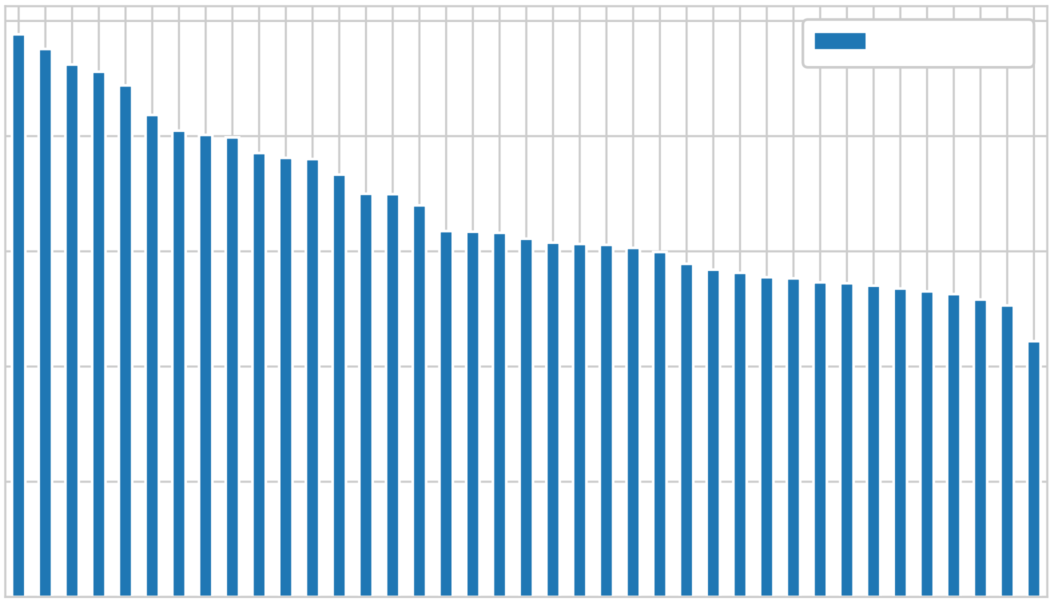
Comparing to Sopranos

In [ ]:
```
tokens2 = df[df['subreddit']=='thesopranos'].selftext_nouns \
         .map(my_tokenizer).sum()

counter2 = Counter(remove_stopwords(tokens2))
wordcloud(counter2)
```

Exploring Text Complexity

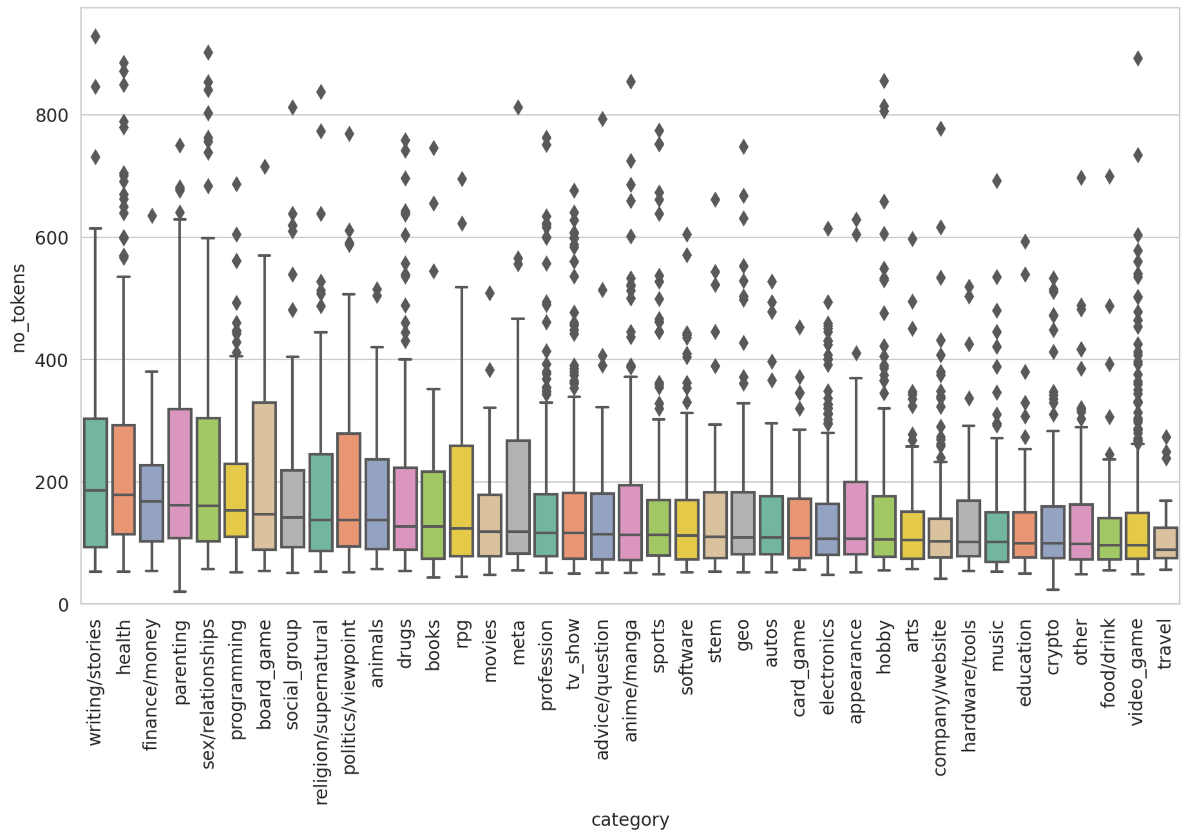
In [ ]:
```
df['no_tokens'] = df.selftext_lemma\
  .map(lambda l: 0 if l==None else len(l.split()))
```

In [ ]:
```
# mean number of tokens by category
df.groupby(['category']) \
  .agg({'no_tokens':'mean'}) \
  .sort_values(by='no_tokens', ascending=False) \
  .plot(kind='bar', figsize=(7,4));
```



```
In [ ]: # render plots as retina or png, because svg is very slow
        %config InlineBackend.figure_format = 'retina'

        import seaborn as sns

        def multi_boxplot(data, x, y, ylim = None):
            '''Wrapper for sns boxplot with cut-off functionality'''
            # plt.figure(figsize=(30, 5))
            fig, ax = plt.subplots()
            plt.xticks(rotation=90)

            # order boxplots by median
            ordered_values = data.groupby(x)[[y]] \
                                 .median() \
                                 .sort_values(y, ascending=False) \
                                 .index

            sns.boxplot(x=x, y=y, data=data, palette='Set2',
                        order=ordered_values)

            fig.set_size_inches(11, 6)

            # cut-off y-axis at value ylim
            ax.set_ylim(0, ylim)
```

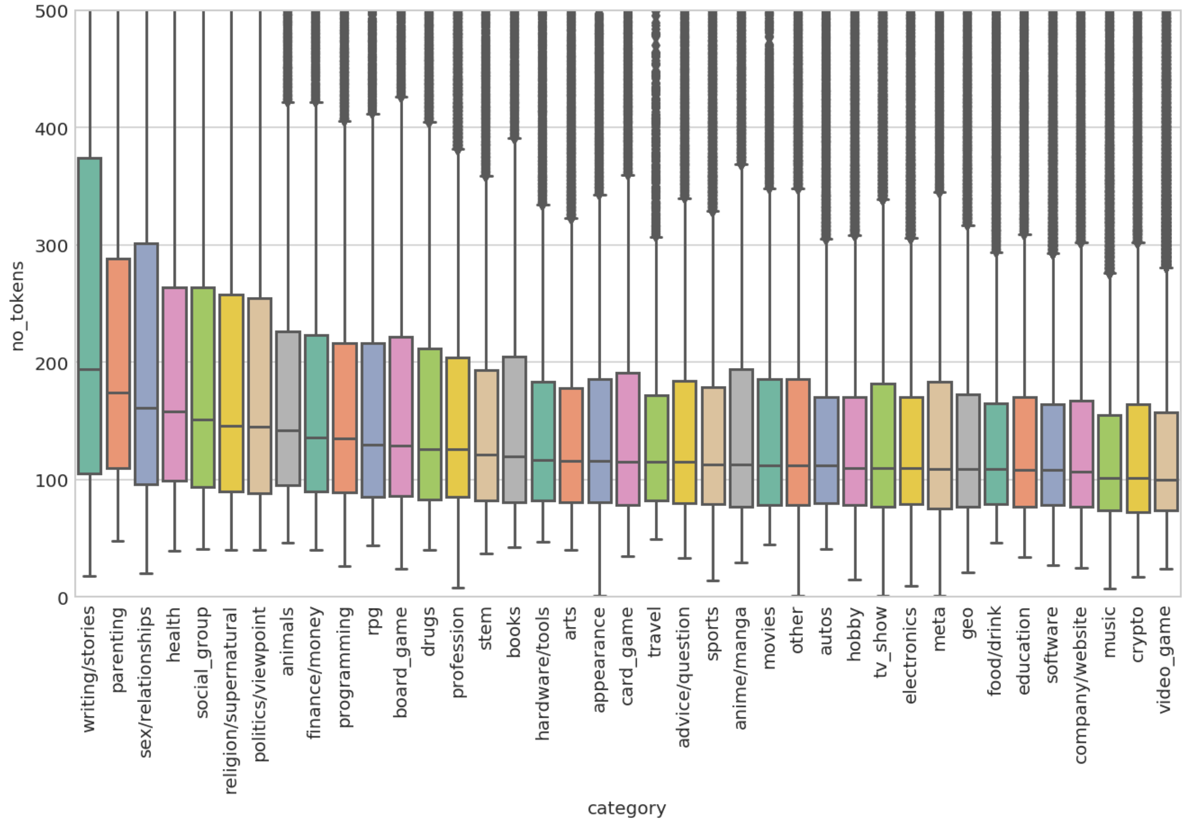
```
In [ ]: multi_boxplot(df, 'category', 'no_tokens');
```



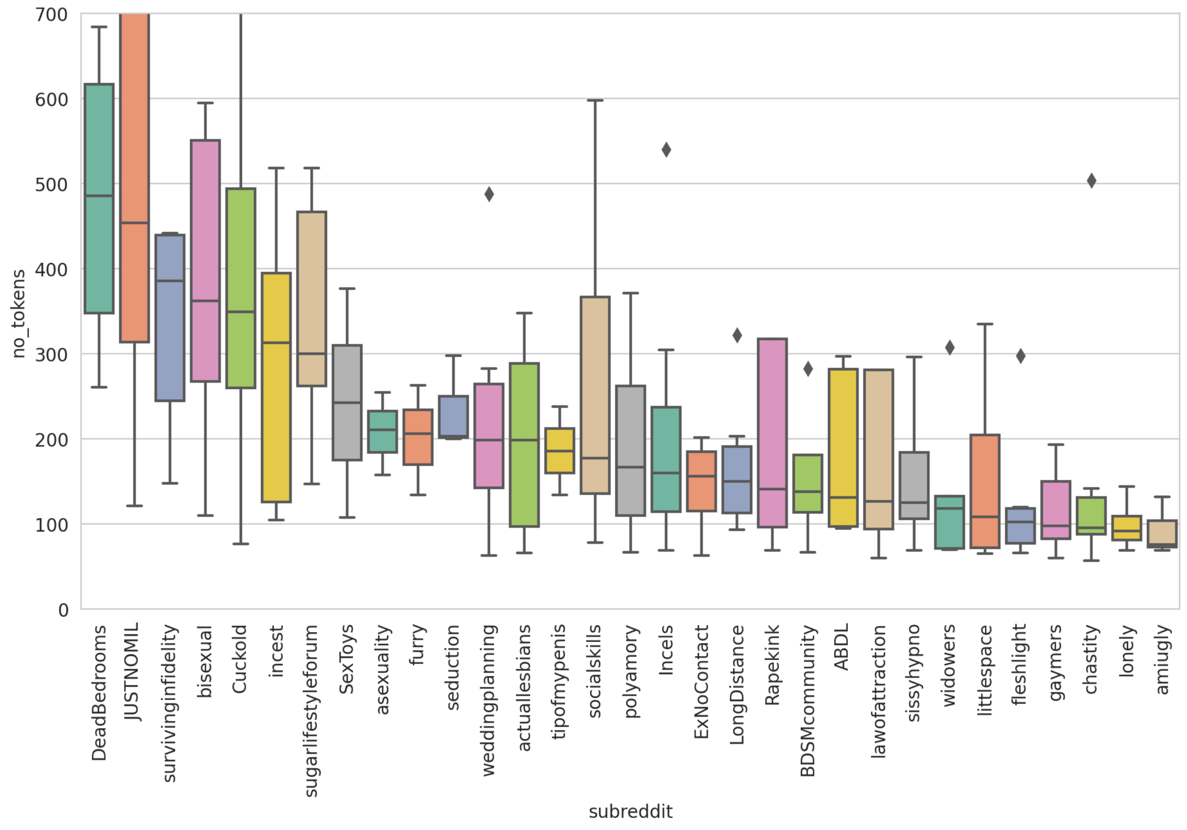
In [ ]:
```
# print text of outliers
df['selftext_lemma'][df.no_tokens > 1500]
```

Out[ ]:
```
Series([], Name: selftext_lemma, dtype: object)
```

In [ ]:
```
# cut-off diagram at y=500
multi_boxplot(df, 'category', 'no_tokens', ylim=500)
```



In [ ]:
```
# comparing subreddits within a single category
multi_boxplot(df[df.category=='sex/relationships'],
              'subreddit', 'no_tokens', ylim=700)
```


no_tokens
subreddit
DeadBedrooms
JUSTNOMIL
survivinginfidelity
bisexual
Cuckold
incest
sugarlifestyleforum
SexToys
asexuality
furry
seduction
weddingplanning
actuallesbians
tipofmypenis
socialskills
polyamory
Incels
ExNoContact
LongDistance
Rapekink
BDSMcommunity
ABDL
lawofattraction
sissyhypno
widowers
littlespace
fleshlight
gaymers
chastity
lonely
amiugly