archive_crawler

October 28, 2019

1 Initialization

The following code demonstrates a web crawler for a forum with multiple pages that contain blog posts. Different features are extracted from the parsed texts and the correlations between the features are investigated. With a larger database, the identified features could be used to train a machine learning algorithm to predict the popularity of new blog posts.

For legal reasons, the links to the forum in use cannot be shared. All visualizations that show names of authors or articles have been commented out.

1.0.1 Libraries

In [4]:
```
# DATA ACQUISITION
import urllib.request, urllib.parse, urllib.error
from urllib.request import Request, urlopen
import ssl
import re
ctx = ssl.create_default_context()
ctx.check_hostname = False
ctx.verify_mode = ssl.CERT_NONE

# DATA STORAGE AND PROCESSING
import numpy as np
import pandas as pd
import datetime

# VISUALIZATION
import matplotlib as mpl
import matplotlib.pyplot as plt
%matplotlib inline
import seaborn as sns
import IPython

# MACHINE LEARNING
from sklearn.model_selection import train_test_split
from sklearn.model_selection import cross_val_score
from sklearn.model_selection import KFold
from sklearn.preprocessing import StandardScaler
```

```
from sklearn.preprocessing import MinMaxScaler
from sklearn.pipeline import Pipeline

# NEURAL NETWORKS
from keras import regularizers
from keras.models import Sequential
from keras.layers import Dense
from keras.wrappers.scikit_learn import KerasRegressor
from keras.optimizers import Adam, Nadam
from keras.activations import softmax
from keras.losses import categorical_crossentropy, logcosh, \
    mean_squared_error
```

2 Data Acquisition

The data was acquired from a blog archive. The archive is subdivided into archives for each year since 2013. The yearly archives contain links to all posts from the respective year. Each blog post contains a header with the title, author, publication date, etc., followed by the main text with a number of graphics and hyperlinks. Also, there is a pop-up for each post that indicates the number of "claps" given by the reader.

2.0.1 Crawl archive for blog posts

The following code starts from the archive main page, crawls into the subpages to find all links to posts that have been made. While links to non-post pages are ignored, all relevant links are stored in a list.

```
In [5]: # Define criteria for relevant links
        exclude = ("/archive", "@", "responses", "tagged")

        # Create empty list of blog posts
        posts = []

        # Crawling the archive
        for year in range(2013, 2020):
            url = "ARCHIVE-URL%d" % (year)
            req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
            html = urlopen(req).read()

        # Parsing the archive to find all outgoing links
            links = re.findall(b'href="(ARCHIVE-URL.*?)"', html)

        # Find links to blog posts
            for link in links:
                if any(s in link.decode() for s in exclude):
                    continue
                posts.append(link.decode())
```

```
        # Delete dublicate links
        posts = list(set(posts))
```

2.0.2 Parse and Store Data

The html codes from the blog posts are parsed and a number of features are extracted using regular expressions. The features relate to the text, the images, as well as to the metadata of the articles. The retrieved informations are stored in a dataframe.

```
In [2]: # Create dataframe for information from blog posts
        columnnames = ["claps", "title", "author", "publication date", \
                       "number of graphics", "text length", \
                       "number of hyperlinks", "title length"]
        characteristics = pd.DataFrame(columns = columnnames)

        # Parse blog posts
        for url in posts:
            req = Request(url, headers={'User-Agent': 'Mozilla/5.0'})
            html = urlopen(req).read()

        # Regex-parse posts to find characteristics
            claps = re.search(r"([0-9]|\.|K)+(?= claps<\/button>)", \
                              html.decode()) # lookahead
            if claps:
                claps = claps.group(0)
            title = re.search(r"(?<=<title>).*(?=</title>)", html.decode())
            if title:
                title = title.group(0)
            author = re.search(r"(?<={\"@type\":\"Person\",\"name\":\")\
                .*?(?=\",\"url\":)", html.decode())
            if author:
                author = author.group(0)
            date = re.search(r"(?<=article:published_time\" content=\").\
                *?(?=T)", html.decode())
            if date:
                date = date.group(0)
            num_graphics = len(re.findall(b'<img class=', html))
            len_html = len(html)
            len_title = len(title)
            num_links = len(re.findall(b'href=', html))

        # Add identified properties to dataframe
            post_char = pd.DataFrame([[claps, title, author, date, \
                                       num_graphics, len_html, \
                                       num_links, len_title]], \
                                     columns = columnnames)
            characteristics = characteristics.append(post_char, \
                                                     ignore_index=True)
```

```
        # Initialize for next iteration
            claps = title = author = date = num_graphics = len_html = \
                num_links = len_title \
            = None

        # print(characteristics.head(10))
```

3 Data Cleaning

In order to enable further processing and to make the data more comparable, the data is cleaned.

```
In [3]: # Delete entries with no publication date
        characteristics.drop(characteristics[characteristics\
                                             ["publication date"].\
                                             isnull()]\
                             .index, inplace=True)

        # Adjust data types
        characteristics[["claps", "title", "author", \
                         "publication date"]] = \
            characteristics[["claps", "title", "author", \
                             "publication date"]].astype(str)

        # Claps: Replace None with 0
        characteristics.replace("None", 0, inplace=True)

        # Claps: Replace K with 1000s
        def thousands(x):
            if "K" in str(x):
                return 1000*float(str(x[:-1]))
            else:
                return x
        characteristics["claps"] = characteristics["claps"].apply(thousands)
        characteristics["claps"] = characteristics["claps"].astype(float).astype(int)

        # Create column for time (in days) since first blog post
        characteristics["publication date"] = characteristics["publication date"]\
            .apply(lambda x: datetime.datetime.strptime(x, '%Y-%m-%d'))
        characteristics["datedelta"] = characteristics["publication date"]\
            .apply(lambda x: (x-min(characteristics["publication date"])).days)

        # Create column for publication year
        characteristics["year"] = characteristics["publication date"].astype(str).\
            apply(lambda x: x[:4])

        # Sort Dataframe by Claps
```

```
characteristics = characteristics.sort_values(by=['claps'])

# print(characteristics)
```

3.0.1 Detection of Outliers

While the number of claps of most articles have been found to lie in a similar range, some articles have far more claps than the majority. In order to find correlations with and general trends in the data, these outliers are identified and ignored.

In [4]:
```
# Create Figure
f,(ax,ax2) = plt.subplots(1,2, sharey=True, figsize=(10, 6), dpi= 80\
                          , facecolor='w', edgecolor='k', gridspec_kw = \
                          {'width_ratios':[7, 1]})

# Plot data
ax.scatter(characteristics['claps'], characteristics['author'])
ax2.scatter(characteristics['claps'], characteristics['author'])

# Adjust axes
ax.set_xlim(-500,6500)
ax2.set_xlim(15500,16500)

ax.spines['right'].set_visible(False)
ax2.spines['left'].set_visible(False)

ax.set(xlabel='# of claps (in thousands)')
ax.xaxis.set_label_coords(0.65, -0.08)

ax.set_xticks(np.linspace(0,6000,7).astype(int))
ax.set_xticklabels(np.linspace(0,6,7).astype(int), fontdict=None, minor=False)

ax2.set_xticks([16000])
ax2.set_xticklabels([16], fontdict=None, minor=False)

# Create rectangle to mark outliers
rect = mpl.patches.Rectangle((2000,13),7000,9,linewidth=1,edgecolor='r',\
                             facecolor='none')
rect2 = mpl.patches.Rectangle((15000,13),1440,9,linewidth=1,edgecolor='r',\
                              facecolor='none')

ax.add_patch(rect)
ax2.add_patch(rect2)

ax.text(2080,12.2, 'Outliers', fontsize=10, color='red')

# plt.show()
# AUTHOR NAMES REMOVED
```

```
        # Drop Outliers
        # Only consider posts with less than or equal 1000 claps and less than
        # or equal 190000 words
        characteristics_reduced = characteristics.drop(characteristics\
                                                       [characteristics["claps"] > \
                                                        1000].index).copy()
        characteristics_reduced = characteristics_reduced\
            .drop(characteristics_reduced[characteristics_reduced["text length"] > \
                                          190000].index).copy()
```

Number of claps for posts sorted by author. In order to identify general trends, outliers are droped from database.

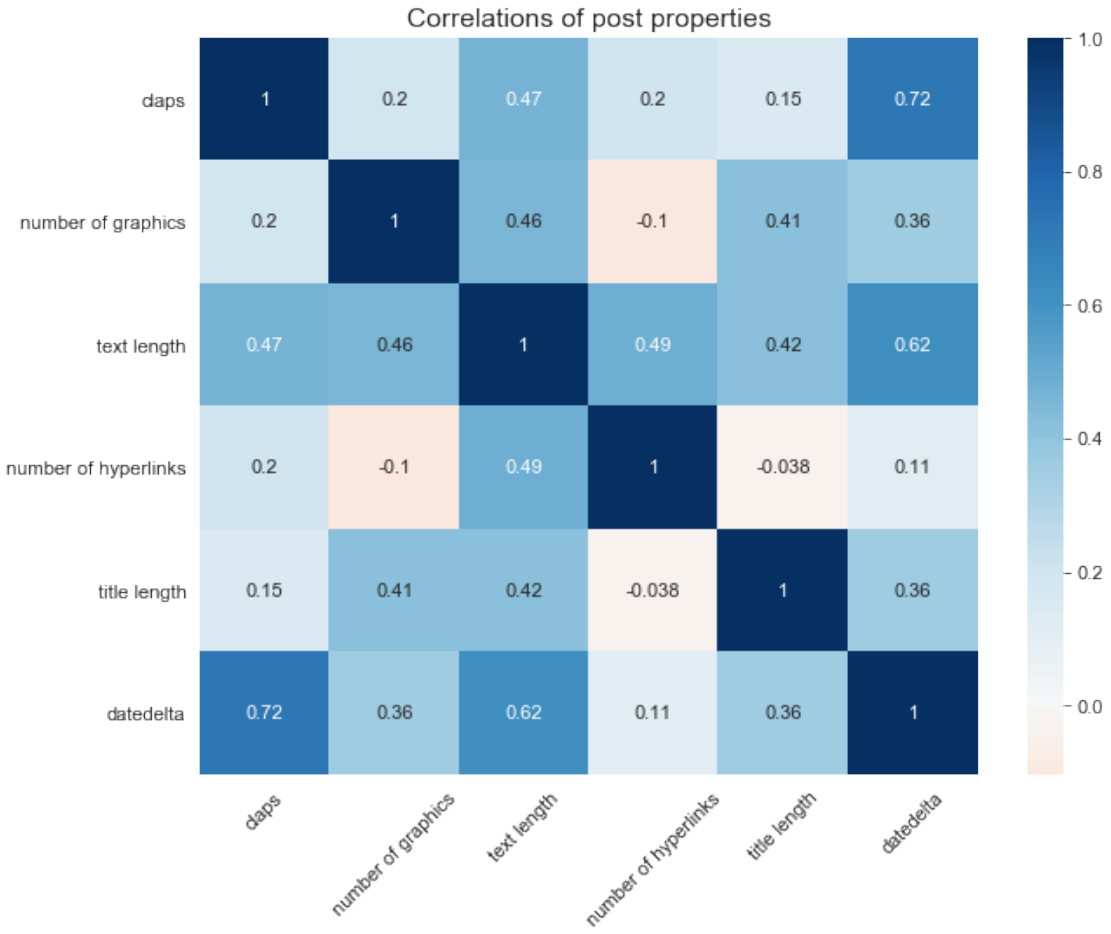
4 Interpretation

4.0.1 Investigation of correlations

A correlogram of the chosen features is plotted. In the first moment, a high correlation between text length and claps can be observed. Thus, it appears that readers tend to like long texts. However, a more careful consideration reveals more information: The strong correlation between text length and the publication date reveals that more recently published posts tend to be longer. Furthermore, the correlation between the date and the claps is very high. This indicates that the number of readers increased and thus the number of claps per post. Consequently, it seems that it is not the text length that yields higher numbers of claps, but rather the increased number of readers as a result of the growing popularity of the forum in general.

```
In [9]: # Create heatmap to visualize correlations
        plt.figure(figsize=(9,7), dpi= 80)
        h = sns.heatmap(characteristics_reduced.corr(), \
                        xticklabels=characteristics_reduced.corr().columns\
                      , yticklabels=characteristics_reduced.corr().columns, \
                        cmap='RdBu', center=0, annot=True)

        # Set decorations
        plt.title('Correlations of post properties', fontsize=14)
        plt.xticks(fontsize=10, rotation=45)
        plt.yticks(fontsize=10)
        plt.show()
```

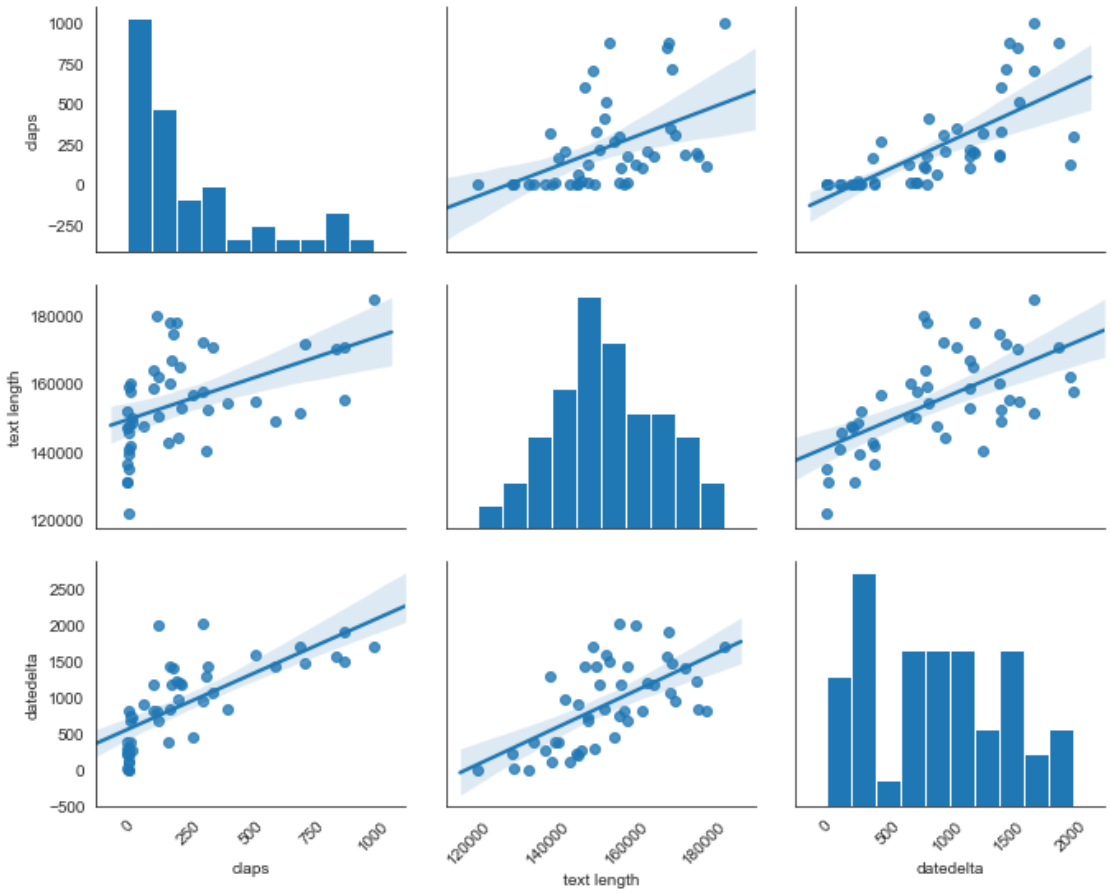


4.0.2 Distribution and Linear Regression of Data

Looking at a regression plot reveals interesting trends in the data. First of all, it is seen that most articles have few claps and an intermediate length. Furthermore it can be seen that the publishing frequency on the forum remained more or less constant. As discussed above, both the text length and the number of claps per article increase with the publishing date.

```
In [13]: # Create regression plot
         sns.set_style("white")
         p = sns.pairplot(characteristics_reduced[(['claps', 'text length', \
                                                    'datedelta'])]\
                          , height=2.5, aspect=1.25, kind='reg')

         # Adjust x-axis
         for ax in p.axes.flat:
             plt.setp(ax.get_xticklabels(), rotation=45)
```


claps
text length
datedelta
1000
750
500
250
0
−250
180000
160000
140000
120000
2500
2000
1500
1000
500
0
−500
0
250
500
750
1000
120000
140000
160000
180000
0
500
1000
1500
2000

In [ ]: