etl

July 6, 2021

```
[1]: import numpy as np
     import pandas as pd
```

1 Data Preprocessing for `articles` file

```
[2]: articles = pd.read_json('raw_data/articles/articles_07_04_2021.json')
```

```
[3]: articles.sample(5)
```

```
[3]:                 author                                 linkOfAuthorProfile  \
     6190         Mohan Gupta  https://towardsdatascience.com/@mohangupta13?s…
     31346    Charmaine Chui  https://towardsdatascience.com/@geek-cc?source…
     36150         German Osin  https://towardsdatascience.com/@gosin?source=c…
     8613   Federico Riveroll  https://towardsdatascience.com/@federicorivero…
     20745         Shai Ardazi  https://towardsdatascience.com/@shaiardazi?sou…

                                                articleTitle  \
     6190   A Review of Named Entity Recognition (NER) Usi…
     31346  Using Turf.js to Geocode coordinates with cust…
     36150  Feature Store as a Foundation for Machine Lear…
     8613   Outstanding results predicting Apple Stock app…
     20745                 Web scraping with Python-A to Z

                                                 articleLink   postingTime  \
     6190   https://towardsdatascience.com/a-review-of-nam…   Jul 9, 2018
     31346  https://towardsdatascience.com/using-turf-js-t…        Jun 22
     36150  https://towardsdatascience.com/feature-store-a…  Dec 10, 2020
     8613   https://towardsdatascience.com/making-a-contin…  Feb 13, 2020
     20745  https://towardsdatascience.com/web-scraping-wi…   Feb 7, 2019

              minToRead recommendations     responses
     6190   11 min read             724  11 responses
     31346   3 min read               1          None
     36150  12 min read             483   2 responses
     8613    9 min read            1.5K  18 responses
```

```
20745  13 min read             298   3 responses
```

1.1 Inspect missing columns

[4]:
```
 # Define the function which checks missing data and types of data
def missing_data(data):
    total = data.isnull().sum()
    percent = (data.isnull().sum()/data.isnull().count()*100)
    tt = pd.concat([total, percent], axis=1, keys=['Total', 'Percent'])
    types = []
    for col in data.columns:
        dtype = str(data[col].dtype)
        types.append(dtype)
    tt['Types'] = types
    return(np.transpose(tt))
```

[5]:
```
missing_data(articles)
```

[5]:
```
         author linkOfAuthorProfile articleTitle articleLink postingTime  \
Total         0                   0         1461           0           0
Percent     0.0                 0.0     3.307151         0.0         0.0
Types    object              object       object      object      object

        minToRead recommendations  responses
Total           1             268      22235
Percent  0.002264        0.606651  50.331621
Types      object          object     object
```

1.2 There's only one entry that doesn't have `minToRead`. It turns out to be a navigation article. So I decided to drop it.

[6]:
```
articles[articles['minToRead'].isnull()]
```

[6]:
```
             author                              linkOfAuthorProfile  \
40490  TDS Editors  https://towardsdatascience.com/@towardsdatasci…

      articleTitle                                      articleLink  \
40490   Navigation  https://towardsdatascience.com/navigation-1f82…

        postingTime minToRead recommendations responses
40490  Nov 14, 2020      None             298      None
```

[7]:
```
articles = articles.dropna(subset=['minToRead'])
```

1.3 Inspect `articleTitle` that has missing values

[8]:
```
articles[articles['articleTitle'].isnull()].sample(5)
```

[8]:
```
                    author  \
28876  Irfan Alghani Khalid
41025              Rose Day
20910          Jo Stichbury
24640      Oleksii Sheremet
40737           Sidney Kung

                                   linkOfAuthorProfile articleTitle  \
28876  https://towardsdatascience.com/@irfanalghani11…         None
41025  https://towardsdatascience.com/@rjday?source=c…         None
20910  https://towardsdatascience.com/@fluffymaccoy?s…         None
24640  https://towardsdatascience.com/@dynamic_phlox_…         None
40737  https://towardsdatascience.com/@sidneykung?sou…         None

                                           articleLink   postingTime  \
28876  https://towardsdatascience.com/this-is-how-i-w…  May 12, 2020
41025  https://towardsdatascience.com/understanding-t…   Nov 7, 2020
20910  https://towardsdatascience.com/anzograph-a-w3c…   Feb 8, 2019
24640  https://towardsdatascience.com/intersection-ov…  Jul 24, 2020
40737  https://towardsdatascience.com/adapting-data-s…  Nov 19, 2020

        minToRead recommendations    responses
28876  4 min read             294   1 response
41025  6 min read              48         None
20910  7 min read              77         None
24640  3 min read              43  2 responses
40737  6 min read             192         None
```

1.3.1 I inspected the 14376th entry and I decided to fill the column `articleTitle`'s null entries with segments from `articleLink`

[9]:
```
link = articles.loc[14376].articleLink
link
```

[9]:
```
'https://towardsdatascience.com/data-science-powered-segmentation-models-
ae89f9bd405f?source=collection_archive---------6-----------------------'
```

[10]:
```
pt1 = link.split("?")[0]
pt1
```

[10]:
```
'https://towardsdatascience.com/data-science-powered-segmentation-models-
ae89f9bd405f'
```

```
[11]: pt2 = pt1.split("/")[-1]
      pt2
```

```
[11]: 'data-science-powered-segmentation-models-ae89f9bd405f'
```

```
[12]: pt3 = pt2.split("-")[:-1]
      pt3
```

```
[12]: ['data', 'science', 'powered', 'segmentation', 'models']
```

```
[13]: title = " ".join(pt3)
      title
```

```
[13]: 'data science powered segmentation models'
```

1.3.2 Merge the above operations and fill null entries in `articleTitle`

```
[14]: articles['articleTitle'] = articles['articleTitle'].
       ↪fillna(articles['articleLink'].apply(lambda x: " ".join(x.split("?")[0].
       ↪split("/")[-1].split("-")[:-1])))
```

1.4 Add user_id column with the same technique as above(find segments from `linkOfAuthorProfile`)

```
[15]: # example link that contains `user_id`
      link = articles.loc[10].linkOfAuthorProfile
      link
```

```
[15]: 'https://towardsdatascience.com/@databeast?source=collection_archive---------
      3-----------------------'
```

```
[16]: link.split('?')[0].split('@')[-1]
```

```
[16]: 'databeast'
```

```
[17]: articles['user_id'] = articles['linkOfAuthorProfile'].apply(lambda x: x.split('?
       ↪')[0].split('@')[-1])
```

1.5 postingTime 's format is either like "Aug 25, 2018" for dates before 2021 or "Jan 13" for dates after 2021.

[18]:
```
# Convert this year's data format in "[Month] [day]" to "[Month] [day] [2021]"
def convert_date(x):
    if ',' not in x:
        x += ', 2021'
    return x

articles['postingTime'] = articles['postingTime'].apply(convert_date)

# Convert the data format in "[Month] [day] [year]" to datetime format
articles['postingTime'] = pd.to_datetime(articles['postingTime'], format='%b␣
 ↪%d, %Y')
```

1.6 recommendations columns have either under 1K(e.g. 221) or >=1K(e.g. 1.3K) or null values

[19]:
```
# Fill the null entries in "recommendations" with "0"
articles['recommendations'].fillna('0', inplace=True)
```

[20]:
```
# Format "3.4K" to "3400" and also transform from string to integer
def convert_recommendations(x):
    if x[-1] == 'K':
        x = int(float(x[:-1]) * 1000)
    else:
        x = int(x)
    return x

articles['recommendations'] = articles['recommendations'].
 ↪apply(convert_recommendations)
```

1.7 responses column has either null values or values' format like "2 responses"

[21]:
```
# Fill the null entries in `responses` with "0 response"
articles['responses'].fillna('0 response', inplace=True)
```

[22]:
```
# Extract the number and format from string to integer
articles['responses'] = articles['responses'].str.split(' ').str[0].astype(int)
```

1.8 `minToRead` column has format "3 min read"

```
[23]: # Extract the number and format from string to integer
      articles['minToRead'] = articles['minToRead'].str.split(' ').str[0].astype(int)
```

2 Data Preprocessing for users file

```
[24]: profiles = pd.read_json('raw_data/users/users_07_04_2021.json')
```

```
[25]: len(profiles)
```

```
[25]: 8000
```

2.1 There are 11684 unique number of user ids collected from `articles` file but there are only 8000 users' profiles are collected

```
[26]: len(set(articles.user_id))
```

```
[26]: 11684
```

2.2 Select unique `user_id` and corresponding `author`

```
[27]: users = articles[["user_id", "author", "linkOfAuthorProfile"]]
```

```
[28]: users = users.drop_duplicates(subset=["user_id"])
```

```
[29]: len(users)
```

```
[29]: 11684
```

2.3 Because of the duplicated names, after merging, there are 100 more wrong entries

```
[30]: df = pd.merge(users, profiles, how="left", left_on="author",␣
       ↪right_on="user_name")
```

```
[31]: df.sample(5)
```

```
[31]:             user_id               author  \
      10719     srees1988                 Sree
      5278    ivana-15022  Ivana Kotorchevikj
```

```
224     sethweidman        Seth Weidman
7476   samdenlepcha       Samden Lepcha
702    paulbradshaw       Paul Bradshaw

                                    linkOfAuthorProfile     user_name  \
10719  https://towardsdatascience.com/@srees1988?sour…           NaN
5278   https://towardsdatascience.com/@ivana-15022?so…           NaN
224    https://towardsdatascience.com/@sethweidman?so…  Seth Weidman
7476   https://towardsdatascience.com/@samdenlepcha?s…           NaN
702    https://towardsdatascience.com/@paulbradshaw?s…           NaN

                                                    desc      followers
10719                                                NaN            NaN
5278                                                 NaN            NaN
224    Became a data scientist to "use math to solve …  992 Followers
7476                                                 NaN            NaN
702                                                  NaN            NaN
```

[32]:
```
len(df)
```

[32]:
```
11748
```

2.4 I inspected all the duplicated user_name and deleted the wrong entries(I only showed the first inspection and omitted the output of all the other inspections)

[33]:
```
duplicated = profiles[profiles.duplicated(subset=['user_name'])].user_name
duplicated
```

[33]:
```
607       Gagandeep Singh
2337        Aditya Sharma
2944       Abhishek Kumar
3339      Harshdeep Singh
3888          Ofer Tirosh
4006               Gaurav
4015       Harshit Sharma
4320          Shen Huang
4343           An Nguyen
4726          Salil Jain
4766        Pranjal Gupta
4881        Shubham Gupta
4899         Bruno Santos
5573          Sahil Gupta
5932          Phoebe Wong
5989          Ravi Ranjan
6252                James
```

```
6562       Abhishek Kumar
7038        Nishant Sinha
7059         Vishal Singh
7329          Manu Sharma
7358      Shekhar Koirala
7411           Nick Jones
7421               Justin
7434           Wendy Wong
7551            Jason Lee
7578              Sue Liu
7725            Christina
7776        Mayank Mishra
7887    Benjamin Peterson
7949         Shikhar Gupta
Name: user_name, dtype: object
```

[34]:
```
df[df.author=='Gagandeep Singh']
```

[34]:
```
               user_id           author  \
3233      gaganmanku96  Gagandeep Singh
3234      gaganmanku96  Gagandeep Singh
4395  singh.gagandeep8  Gagandeep Singh
4396  singh.gagandeep8  Gagandeep Singh

                                 linkOfAuthorProfile        user_name  \
3233  https://towardsdatascience.com/@gaganmanku96?s…  Gagandeep Singh
3234  https://towardsdatascience.com/@gaganmanku96?s…  Gagandeep Singh
4395  https://towardsdatascience.com/@singh.gagandee…  Gagandeep Singh
4396  https://towardsdatascience.com/@singh.gagandee…  Gagandeep Singh

                                desc      followers
3233  Data Scientist at Zykrr. Geeky -  578 Followers
3234      Big Data Engineer at WooliesX   74 Followers
4395  Data Scientist at Zykrr. Geeky -  578 Followers
4396      Big Data Engineer at WooliesX   74 Followers
```

[35]:
```
df = df.drop(index=[3234, 4395])
```

[36]:
```
# df[df.author=='Aditya Sharma']
```

[37]:
```
df = df.drop(index=[2474, 8620])
```

[38]:
```
# df[df.author=='Abhishek Kumar']
```

[39]:
```
df = df.drop(index=[6811, 6813, 2828, 4363, 4364, 2827, 2828])
```

[40]:
```
# df[df.author=='Harshdeep Singh']
```

```
[41]: df = df.drop(index=[1180, 3498])
```

```
[42]: # df[df.author=='Ofer Tirosh']
```

```
[43]: df = df.drop(index=[3507, 7540])
```

2.5 During inspections I also found some profiles' description wasn't collected so I filled them manually

```
[44]: df.loc[3506].desc = "CEO and Founder of Tomedes, a professional services␣
       ↪provider to Fortune 500 companies around the world specializing in␣
       ↪localization and translation."
```

```
[45]: # df[df.author=='Gaurav']
```

```
[46]: df = df.drop(index=[256, 3993])
```

```
[47]: df.loc[257].desc = "Editor of TapTechie Publication and Tech@Breno"
```

```
[48]: # df[df.author=='Harshit Sharma']
```

```
[49]: df = df.drop(index=[384, 830])
```

```
[50]: # df[df.author=='Shen Huang']
```

```
[51]: df = df.drop(index=[3779, 5970])
```

```
[52]: # df[df.author=='An Nguyen']
```

```
[53]: df = df.drop(index=[955, 6053])
```

```
[54]: # df[df.author=='Salil Jain']
```

```
[55]: df = df.drop(index=[328, 7938])
```

```
[56]: # df[df.author=='Pranjal Gupta']
```

```
[57]: df = df.drop(index=[6699, 8601])
```

```
[58]: # df[df.author=='Shubham Gupta']
```

```
[59]: df = df.drop(index=[8120, 8820])
```

```
[60]: # df[df.author=='Bruno Santos']
```

```
[61]: df = df.drop(index=[2628, 4611])
```

```
[62]: # df[df.author=='Sahil Gupta']
```

```
[63]: df = df.drop(index=[3909, 8190])
```

```
[64]: # df[df.author=='Phoebe Wong']
```

```
[65]: df = df.drop(index=[4299, 5801])
```

```
[66]: # df[df.author=='Ravi Ranjan']
```

```
[67]: df = df.drop(index=[938, 3237])
```

```
[68]: # df[df.author=='James']
```

```
[69]: df = df.drop(index=[331, 8613])
```

```
[70]: # df[df.author=='Nishant Sinha']
```

```
[71]: df = df.drop(index=[213, 2663])
```

```
[72]: # df[df.author=='Vishal Singh']
```

```
[73]: df = df.drop(index=[1002, 1623])
```

```
[74]: df.loc[1001].desc = 'Medium member since August 2020'
```

```
[75]: # df[df.author=='Manu Sharma']
```

```
[76]: df = df.drop(index=[4068, 5522])
```

```
[77]: # df[df.author=='Shekhar Koirala']
```

```
[78]: df = df.drop(index=[894, 1652])
```

```
[79]: # df[df.author=='Nick Jones']
```

```
[80]: df = df.drop(index=[1054, 2063])
```

```
[81]: # df[df.author=='Justin']
```

```
[82]: df = df.drop(index=[2371, 5415])
```

```
[83]: df.loc[2372].desc="Hello, world! My name is Justin. I solve problems using data.
       ↪ Check me out at embracingtherandom.com and linkedin.com/in/justin-m-evans/"
```

```
[84]: # df[df.author=='Wendy Wong']
```

```
[85]: df = df.drop(index=[1519, 4695])
```

```
[86]: # df[df.author=='Jason Lee']
```

```
[87]: df = df.drop(index=[3501, 8042])
```

```
[88]: # df[df.author=='Sue Liu']
```

```
[89]: df = df.drop(index=[1537, 7131])
```

```
[90]: # df[df.author=='Christina']
```

```
[91]: df = df.drop(index=[932, 8556])
```

```
[92]: # df[df.author=='Mayank Mishra']
```

```
[93]: df = df.drop(index=[7293, 7555])
```

```
[94]: # df[df.author=='Benjamin Peterson']
```

```
[95]: df = df.drop(index=[3511, 6446])
```

```
[96]: # df[df.author=='Shikhar Gupta']
```

```
[97]: df = df.drop(index=[47, 6058])
```

2.6 Now there are no duplicated wrong entries!!! I also dropped the duplicated column `user_name`

```
[98]: df[df.duplicated(subset=["user_id"])]
```

```
[98]: Empty DataFrame
      Columns: [user_id, author, linkOfAuthorProfile, user_name, desc, followers]
      Index: []
```

```
[99]: len(df)
```

```
[99]: 11684
```

```
[100]: df = df.drop(columns="user_name")
```

2.7 followers column has null values or that format "552 followers"

```
[101]: # Change the 'null' entries to '0 follower'
       df['followers'].fillna('0 follower', inplace=True)
```

```
[102]: # Transform the format from "[num] follower(s)" to "num" in integer
       df['followers'] = df['followers'].str.split(' ').str[0]
```

```
[103]: # Format "3.4K" to "3400" and convert string to integer
       def convert_followers(x):
               if x[-1] == 'K':
                   x = int(float(x[:-1]) * 1000)
               else:
                   x = int(x)
               return x

       df['followers'] = df['followers'].apply(convert_followers)
```

```
[104]: df.sample(5)
```

```
[104]:              user_id          author  \
       9868   jeffrey-scholz  Jeffrey Scholz
       6702            dlite    Derek Haynes
       1253           leofle   Lio Fleishman
       3366    tolaniadekoya  Tolani Adekoya
       11435          colefp            Cole

                                           linkOfAuthorProfile  \
       9868   https://towardsdatascience.com/@jeffrey-scholz…
       6702   https://towardsdatascience.com/@dlite?source=c…
       1253   https://towardsdatascience.com/@leofle?source=…
       3366   https://towardsdatascience.com/@tolaniadekoya?…
       11435  https://towardsdatascience.com/@colefp?source=…

                                                         desc  followers
       9868                                               NaN          0
       6702                                       Working on         336
       1253   Partnership Solutions Engineer at Sisense , th…         22
       3366                                               NaN          0
       11435                                              NaN          0
```

3 Export the cleaned data to csv files

[105]:
```
df.to_csv("cleaned_data/users/users_07_04_2021.csv", index=False)
```

[106]:
```
articles = articles.drop(columns=["author", "linkOfAuthorProfile"])
articles.to_csv("cleaned_data/articles/articles_07_04_2021.csv", index=False)
```