CORD-19 Explore dataset

In general, this jupyter notebook is designated to explore the CORD19 dataset:

Wade, Alex D.; Williams, Ivana (2021), CORD-19 Software Mentions, Dryad, Dataset, https://doi.org/10.5061/dryad.vmcvdncs0 (https://doi.org/10.5061/dryad.vmcvdncs0)

First, relevant packages must be imported into the notebook.

In [1]:

```
import numpy as np
import pandas as pd
import csv
import ast
import collections
import matplotlib.pyplot as plt
import matplotlib.patches as mpatches
import Levenshtein as lev
from fuzzywuzzy import fuzz
import datetime
import matplotlib.pyplot as plt
import re
from urllib.parse import urlparse
from collections import Counter
```

Get the data and save it to a variable.

In [2]:

```
CORD19_CSV = pd.read_csv('../data/cord-19/CORD19_software_mentions.csv')
```

Show the head of the dataset to inspect all columns and obtain a broad overview.

In [3]:

```
CORD19_CSV
```

Out[3]:

	paper_id	doi	title	source_x	lice
0	00006903b396d50cc0037fed39916d57d50ee801	NaN	Urban green space and happiness in developed c...	ArXiv	a
1	0000fcce604204b1b9d876dc073eb529eb5ce305	10.1016/j.regg.2021.01.002	La Geriatría de Enlace con residencias en la é...	Elsevier; PMC	co
2	000122a9a774ec76fa35ec0c0f6734e7e8d0c541	10.1016/j.rec.2020.08.002	Impact of COVID-19 on ST-segment elevation myo...	Elsevier; Medline; PMC	no
			Absence of surface	Elsevier;	

General check of the DataFrame

First of all, it is interesting to investigate how many NaNs are contained within the DataFrame.

In [4]:

```
CORD19_CSV.isnull().sum().sum()
```

Out[4]:

```
12476
```

How are the NaNs distributed between the columns of the DataFrame?

In [5]:

```
def count_nan_per_column(df):
    """
    This function counts the NaNs per column of the received DataFrame
    and returns a DataFrame containing a column listing the number of NaNs..
    """
    col_num = []
    col_name = []
    for i in df:
        col_num.append(df[i].isnull().sum())
    col_names = df.columns
    df_joined =  pd.DataFrame(col_num, col_names)
    df_joined.columns = ['NaNs']
    return df_joined
```

In [6]:

```
df_NaNs = count_nan_per_column(CORD19_CSV)
df_NaNs
```

Out[6]:

	NaNs
paper_id	0
doi	3144
title	0
source_x	0
license	0
publish_time	0
journal	9332
url	0
software	0

For further analysis, it is important to obey that the columns "doi" and "journal" contain NaNs.

In [7]:

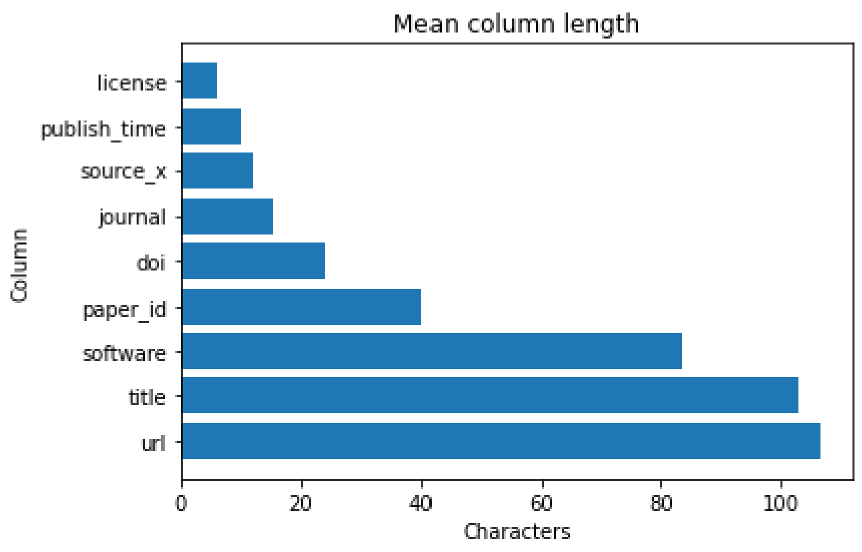
```
ser_col_length = CORD19_CSV.fillna('').astype(str).apply(lambda x:x.str.len()).mean()
ser_col_length_sorted = ser_col_length.sort_values(ascending=False)
ser_col_length_sorted
```

Out[7]:

```
url             106.520155
title           102.960064
software         83.598324
paper_id         40.000000
doi              24.133883
journal          15.316212
source_x         12.139758
publish_time      9.930508
license           5.899985
dtype: float64
```

In [8]:

```
x = ser_col_length_sorted.index
y = ser_col_length_sorted.values

plt.barh(x, y)
plt.title("Mean column length")
plt.xlabel("Characters")
plt.ylabel("Column")
plt.show()
```



For further analysis, the average length of titles and URLs is long and can be seen as descriptive.

The column publish_time is converted to the DateTime format.

In [9]:

```
CORD19_CSV['publish_time']=pd.to_datetime(CORD19_CSV.publish_time)
```

Dedicated analysis based on columns

Create own series for each column.

In [10]:

```
paper_id = CORD19_CSV.paper_id
doi = CORD19_CSV.doi
title = CORD19_CSV.title
source_x = CORD19_CSV.source_x
license = CORD19_CSV.license
publish_time= CORD19_CSV.publish_time
journal = CORD19_CSV.journal
url = CORD19_CSV.url
```

Checking the column "paper_id"

First, check the integrity of the series paper_id. It should be obvious that each row has a unique ID that is not shared with another row.

In [11]:

```
paper_id_counted = paper_id.value_counts()
paper_id_counted
```

Out[11]:

```
0ed3c6a5559cd73307184f51fc53ccc76da559bc    3
5d6678f81812464543b367e7de138e23b3483ed1    2
5d0d0bd116976e1412c10a84902894999df4a342    2
ff40e6b44e151e42a54227e255a88d0c0c104876    2
46b053c7126c1603101f46e4bb6e411f790a45fc    2
                                           ..
8193755d869fc3ecf58f4838fcd616163584cbdc    1
74165e4814a803cf72f62b9cad68c00659607c1a    1
ba32b2acc5045255ab8b9d17083b1196131dd700    1
9c1595012ce2d5e518584750984dfbb69771e0b5    1
f25f980338756b6535ec83c06ae483d56af3e085    1
Name: paper_id, Length: 77436, dtype: int64
```

The first finding linked to the column paper_id is that this column does not solely contain unique ID's. Some rows share ID's. For this, the exact number of shared ID's need to be found.

In [12]:

```
def find_shared_values(col):
    """ This function checks columns for shared values and deducts NaNs
    """
    col_shared = len(col) - len(col.value_counts())
    col_shared = col_shared - col.isnull().sum()
    return col_shared
```

In [13]:

```
paper_id_shared_ids_num = find_shared_values(paper_id)
paper_id_shared_ids_num
```

Out[13]:

```
12
```

The column contains 12 shared ID's with even one used three times.

In [14]:

```
paper_id_counted.head(paper_id_shared_ids_num)
```

Out[14]:

```
0ed3c6a5559cd73307184f51fc53ccc76da559bc    3
5d6678f81812464543b367e7de138e23b3483ed1    2
5d0d0bd116976e1412c10a84902894999df4a342    2
ff40e6b44e151e42a54227e255a88d0c0c104876    2
46b053c7126c1603101f46e4bb6e411f790a45fc    2
dd74a3a343529174fe7c6485723cf2d5911c18ed    2
d1dde1df11f93e8eae0d0b467cd0455afdc5b98c    2
0831fe32280e46ba8d5c1a9456111e1e009863ac    2
ec7d3038b8912a9fc92f4d02a2c30d566d4d0a93    2
36e2047d1674c3095617f3eb97f9f61e48989dfe    2
c89f86cdd9d41eeec127cc0b03990c52888a9635    2
b391f95092b4335dc9f80fa3a1d56ff9e1d3f8bf    1
Name: paper_id, dtype: int64
```

Check if the dataset contains only duplicates for the column paper_id or whole DataFrame rows. Therefore, the function collect_rows_of_df will support the process.

In [15]:

```
#Method receiving string and dataframe which returns double or tripple dataframe to append
def collect_rows_of_df(df,column,st):
    """This function receives a dataframe, a column contained within a dataframe
    and a string which can be found within the column.
    Then, the string is compared to the whole column and.
    When a match is found, the corresponding rows are returned as a dataframe.
    """
    subset = df[df[column] == st]
    return subset
```

Collecting rows that share their paper_id.

In [16]:

```
x = 0
shared_paper_id_df = pd.DataFrame(columns=['paper_id','doi','title','source_x','license','p
while x < paper_id_shared_ids_num:
    shared_paper_id_df = shared_paper_id_df.append(collect_rows_of_df(CORD19_CSV, 'paper_id
    x= x+1
shared_paper_id_df
```

Out[16]:

	paper_id	doi	title	source_x	license
4466	0ed3c6a5559cd73307184f51fc53ccc76da559bc	10.1016/j.jinf.2020.02.019	Simulating and forecasting the cumulative conf...	Elsevier; Medline; PMC	no-cc
4467	0ed3c6a5559cd73307184f51fc53ccc76da559bc	10.1016/j.jinf.2020.02.020	Novel coronavirus disease (Covid-19): The firs...	Elsevier; Medline; PMC	els-covid
4468	0ed3c6a5559cd73307184f51fc53ccc76da559bc	10.1016/j.jinf.2020.02.011	Chinese medical personnel	Elsevier; Medline;	els-

Besides software, the paper_id duplicates have variations among all other columns which means that the same paper_id was used for different publications.

Checking the column "doi"

Each paper in this dataset should have a unique "doi" because duplicates are skewing the analysis and potential outcomes.

In [17]:

```
doi_counted = doi.value_counts()
doi_counted
```

Out[17]:

```
10.1016/j.dsx.2020.04.012          2
10.31729/jnma.5498                 2
10.1097/im9.0000000000000035       1
10.1016/j.jmoldx.2014.12.002       1
10.1007/978-3-030-52237-7_31       1
                                  ..
10.1016/j.jairtraman.2020.101900   1
10.1101/2020.07.16.20155721        1
10.1093/cid/ciaa468                1
10.1016/j.jtumed.2020.06.005       1
10.1007/s12117-020-09401-y         1
Name: doi, Length: 74302, dtype: int64
```

In [18]:

```
doi_shared_dois_num = find_shared_values(doi)
doi_shared_dois_num
```

Out[18]:

```
2
```

The column "doi" contains two entries that share a "doi".

In [19]:

```
doi_counted.head(doi_shared_dois_num)
```

Out[19]:

```
10.1016/j.dsx.2020.04.012    2
10.31729/jnma.5498           2
Name: doi, dtype: int64
```

Based on rows, it needs to be explored how the shared doi's are affecting each other.

In [20]:

```
x = 0
shared_doi_df = pd.DataFrame(columns=['paper_id','doi','title','source_x','license','publis
while x < doi_shared_dois_num:
    shared_doi_df  = shared_doi_df.append(collect_rows_of_df(CORD19_CSV, 'doi', doi_counted
    x= x+1
shared_doi_df
```

Out[20]:

	paper_id	doi	title	source
16564	36e2047d1674c3095617f3eb97f9f61e48989dfe	10.1016/j.dsx.2020.04.012	Artificial Intelligence (AI) applications for ...	PI
16565	36e2047d1674c3095617f3eb97f9f61e48989dfe	10.1016/j.dsx.2020.04.012	Artificial Intelligence (AI) applications for ...	Elsev
38600	80273c63683cad57323802542cfdcfcd76c805bf	10.31729/jnma.5498	Mental Wellbeing during the Lockdown Period fo...	PI
42407	8cab7532249cedf3815d4dada6400390a1f8a28a	10.31729/jnma.5498	Interpersonal Violence during the COVID-19 Loc...	PI

The first matched doi is already known for sharing paper_id's. For the second shared doi, there is an interesting occurrence because the algorithm which created the dataset created two distinct outcomes for the column "software". The affected entries are presented below.

In [21]:

```
corrupted_software_mentions = pd.DataFrame()
corrupted_software_mentions['Publication 1'] = shared_doi_df.loc[38600]
corrupted_software_mentions['Publication 2'] = shared_doi_df.loc[42407]
corrupted_software_mentions
```

Out[21]:

	Publication 1	Public
paper_id	80273c63683cad57323802542cfdcfcd76c805bf	8cab7532249cedf3815d4dada6400390a1
doi	10.31729/jnma.5498	10.31729/jnm
title	Mental Wellbeing during the Lockdown Period fo...	Interpersonal Violence during the CC
source_x	PMC	
license	cc-by	
publish_time	2020-10-31 00:00:00	2020-10-31 0
journal	JNMA J Nepal Med Assoc	JNMA J Nepal Mec
url	https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7...	https://www.ncbi.nlm.nih.gov/pmc/articles/F
software	['SPSS']	['SPSS',

In comparison to publication 1, publication 2 is affiliated additionally to the software mention "WHO". This finding leads to the assumption that the algorithm which is creating software mentions does not work in a reproducible way.

Checking the column "title"

Next, the column title will be investigated to produce insights.

In [22]:

```
title_counted = title.value_counts()
title_counted
```

Out[22]:

```
Full Issue PDF
15
Posters
8
Poster Presentations
8
Poster Sessions
7
Poster Session Abstracts
7

..
Population Data-Driven Formulation of a COVID-19 Therapeutic
1
Metformin Reduces NGF-Induced Tumour Promoter Effects in Epithelial Ovarian
Cancer Cells
1
Viral infection detection using metagenomics technology in six poultry farms
of eastern China
1
Effects of work status changes and perceived stress on glycaemic control in
individuals with type 1 diabetes during COVID-19 lockdown in Italy
1
STRATEGY FOR THE CONTAINMENT, MITIGATION, AND SUPPRESSION OF THE COVID-19 PA
NDEMIC IN FRAGILIZED COMMUNITIES ON THE PERIPHERY OF A LARGE BRAZILIAN CITY.
1
Name: title, Length: 76578, dtype: int64
```

In [23]:

```
title_shared_titles_num = find_shared_values(title)
title_shared_titles_num
```

Out[23]:

```
870
```

Compared to the other duplicates, there are many more shared titles.

In [24]:

```
if title_shared_titles_num > 20:
    head_for_title = 40
else:
    head_for_title = title_shared_titles_num
title_counted.head(head_for_title)
```

Out[24]:

```
Full Issue PDF
15
Posters
8
Poster Presentations
8
Poster Sessions
7
Poster Session Abstracts
7
Scientific Abstracts
6
Physicians Poster Sessions
6
Mitteilungen der Deutschen Gesellschaft für Neurologie
5
NEWS
5
Infectieziekten
5
Abstracts cont.
5
Abstract
5
Virtual Meeting Announcment
4
Mitteilungen der ÖGKJ
4
Tracing the Origins of Agricultural Products with Barcoded Microbial Spores
4
Oral Abstracts
3
Mitteilungen des BDI
3
Mitteilungen der DGN
3
China
3
The EU versus the Automotive Industry
3
Oral presentations
3
Poster presentations
3
Oral Presentations
3
Introduction to Computer Network Vulnerabilities
3
Report
3
ISAR News
```

```
3
Highlights From the Current Issue – Audiovisual Summary
3
Forum
3
Using Morpholinos to Control Gene Expression
3
Computer Network Vulnerabilities
3
Reply
3
New approximations, and policy implications, from a delayed dynamic model of
a fast pandemic     3
Prediction of repurposed drugs for treating lung injury in COVID-19
3
Mitteilungen der DGIM
3
Symposium Summaries
3
Bildgebende Verfahren: Röntgen, Ultraschall, CT, Nuklearmedizin
3
In Response
3
Die Virologie von SARS-CoV-2
3
Corona, Immissionen und der Verbrennungsmotor
3
Poster
3
Name: title, dtype: int64
```

The root for such a high amount of title duplicates might be founded for various reasons. At least for this selection, the publications have poor naming which means that they might be published without a name and got assigned a placeholder. Furthermore, this selection has a weak linkage to COVID related research. Based on the titles, the shown selection contains more none COVID19 related publications than research about COVID19.

The column strings contain words that can be investigated by matching certain substrings. Therefore, the following function will support the process.

In [25]:

```python
def check_column_for_string(df, col, st):
    """Function which checks matches of substrings in the whole column and returns a datafr
    """
    df_match = df.loc[df[col].str.contains(st, case=False)]
    return df_match
```

To inspect the density of COVID related publications, the column title will be checked for common words:

- Covid
- Corona
- Lockdown
- SARS

In [26]:

```
matched_df_covid = check_column_for_string(CORD19_CSV, 'title', 'covid')
matched_df_corona = check_column_for_string(CORD19_CSV, 'title', 'corona')
matched_df_lockdown = check_column_for_string(CORD19_CSV, 'title', 'lockdown')
matched_df_sars = check_column_for_string(CORD19_CSV, 'title', 'sars')
```

In [27]:

```
sum_covid_related = len(matched_df_covid) + len(matched_df_corona) + len(matched_df_lockdow
sum_covid_related
```

Out[27]:

```
40233
```

The sum may contain an overlap of the checked substrings which needs to be deducted.

In [28]:

```
matched_df_covid_medical = matched_df_covid.append(matched_df_corona)
matched_df_covid_medical.append(matched_df_lockdown)
matched_df_covid_medical.append(matched_df_sars)
matched_df_covid_medical_len = len(matched_df_covid_medical)
matched_df_covid_medical_len
```

Out[28]:

```
30982
```

Furthermore, it is interesting to see the percentage of publications of the CORD19 dataset which match the chosen terms.

In [29]:

```
match_ratio = matched_df_covid_medical_len/len(CORD19_CSV)
match_ratio
```

Out[29]:

```
0.4000361532899494
```

To inspect the potential impacts of COVID on society, the column title will be checked for additional terms:

- Economic
- Social

In [30]:

```
matched_df_economic = check_column_for_string(CORD19_CSV, 'title', 'economic')
matched_df_social = check_column_for_string(CORD19_CSV, 'title', 'social')
```

In [31]:

```
len(matched_df_economic)
```

Out[31]:

```
465
```

In [32]:

```
len(matched_df_social)
```

Out[32]:

```
1542
```

Subsequently, the ratio between the term social and economic is presented.

In [33]:

```
len(matched_df_social)/len(matched_df_economic)
```

Out[33]:

```
3.3161290322580643
```

There might be an overlap between the covid related DataFrame and the DataFrame addressing covid impacts on society.

In [34]:

```
matched_df_society = matched_df_economic.append(matched_df_social)
matched_df_society_len = len(matched_df_society)
matched_df_society_len
```

Out[34]:

```
2007
```

Compared to the size of the dataset, potential covid impacts on society are more researched and published.

In [35]:

```
combined_df_fields = matched_df_society.append(matched_df_covid_medical)
dif_len_fields = len(matched_df_covid_medical) + len(matched_df_society) - len(combined_df_
dif_len_fields
```

Out[35]:

```
0
```

Both fields do not have any overlap. Potentially, both fields could not be linked to COVID related subjects and investigate distant fields.

Checking the column "source_x"

For publishing purposes, the source of the papers will be inspected.

```
In [36]:
```

```
source_x_counted = source_x.value_counts()
source_x_counted
```

```
Out[36]:
```

```
Medline; PMC                              32777
Elsevier; Medline; PMC                    15500
PMC                                       15103
MedRxiv; WHO                               4760
ArXiv                                      3115
Elsevier; PMC                              1831
BioRxiv; WHO                               1252
BioRxiv                                     796
MedRxiv                                     367
MedRxiv; Medline; PMC; WHO                  331
Medline; PMC; WHO                           285
BioRxiv; Medline; PMC; WHO                  278
MedRxiv; Medline; WHO                       152
BioRxiv; MedRxiv                            149
BioRxiv; Medline; WHO                       145
Elsevier                                    135
ArXiv; Medline; PMC                         123
Elsevier; Medline; PMC; WHO                 115
ArXiv; Elsevier; Medline; PMC                62
BioRxiv; MedRxiv; WHO                        51
BioRxiv; Medline; PMC                        27
ArXiv; Medline                               23
BioRxiv; Medline                             13
MedRxiv; Medline; PMC                        11
ArXiv; PMC                                    8
PMC; WHO                                      7
Elsevier; Medline                             6
BioRxiv; MedRxiv; Medline; WHO                6
ArXiv; Elsevier; PMC                          5
ArXiv; Elsevier                               3
BioRxiv; MedRxiv; Medline; PMC                3
BioRxiv; MedRxiv; Medline; PMC; WHO           3
Elsevier; PMC; WHO                            2
MedRxiv; Medline                              2
BioRxiv; MedRxiv; Medline                     1
ArXiv; Elsevier; Medline; PMC; WHO            1
Name: source_x, dtype: int64
```

Most of the publications have various and shared publishers.

```
In [37]:
```

```
len_source_x_counted = len(source_x_counted)
```

Thusly, it needs to be considered which sources are responsible for most publications.

In [38]:

```
i = 0
while i < len_source_x_counted:
    source_x_counted.index.values[i] = source_x_counted.index.values[i].replace(';', '')
    i= i+1
source_x_counted
```

Out[38]:

```
Medline PMC                          32777
Elsevier Medline PMC                 15500
PMC                                  15103
MedRxiv WHO                           4760
ArXiv                                 3115
Elsevier PMC                          1831
BioRxiv WHO                           1252
BioRxiv                                796
MedRxiv                                367
MedRxiv Medline PMC WHO                331
Medline PMC WHO                        285
BioRxiv Medline PMC WHO                278
MedRxiv Medline WHO                    152
BioRxiv MedRxiv                        149
BioRxiv Medline WHO                    145
Elsevier                               135
ArXiv Medline PMC                      123
Elsevier Medline PMC WHO               115
ArXiv Elsevier Medline PMC              62
BioRxiv MedRxiv WHO                     51
BioRxiv Medline PMC                     27
ArXiv Medline                           23
BioRxiv Medline                         13
MedRxiv Medline PMC                     11
ArXiv PMC                                8
PMC WHO                                  7
Elsevier Medline                         6
BioRxiv MedRxiv Medline WHO              6
ArXiv Elsevier PMC                       5
ArXiv Elsevier                           3
BioRxiv MedRxiv Medline PMC              3
BioRxiv MedRxiv Medline PMC WHO          3
Elsevier PMC WHO                         2
MedRxiv Medline                          2
BioRxiv MedRxiv Medline                  1
ArXiv Elsevier Medline PMC WHO           1
Name: source_x, dtype: int64
```

In [39]:

```
i = 0
df_source_holder = pd.DataFrame(columns=['source_x','counts'])
while i < len_source_x_counted:
    helper = source_x_counted.index[i].split()
    for x in helper:
        data = {'source_x': x, 'counts': source_x_counted[i]}
        df_source_holder = df_source_holder.append(pd.DataFrame([data]))
    i = i + 1

df_source_holder = df_source_holder.reset_index()
df_source_holder = df_source_holder.drop(labels=["index"], axis=1)
df_source_holder
```

Out[39]:

	source_x	counts
0	Medline	32777
1	PMC	32777
2	Elsevier	15500
3	Medline	15500
4	PMC	15500
...	...	...
91	ArXiv	1
92	Elsevier	1
93	Medline	1
94	PMC	1
95	WHO	1

96 rows × 2 columns

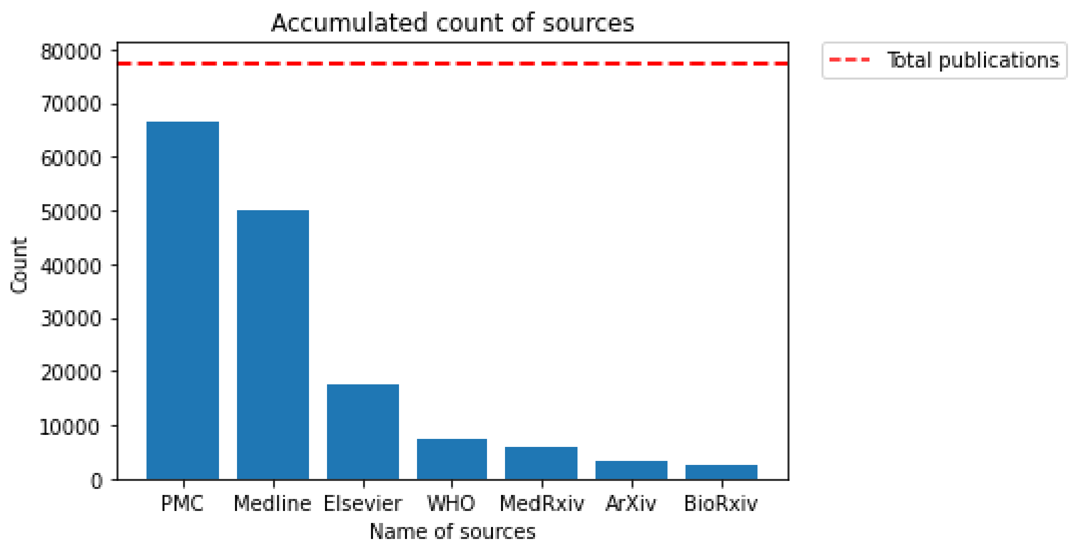
The DataFrame above contains the counts per each source which must be summed up properly.

In [40]:

```
len_df_source_holder = len(df_source_holder)
sources_series = df_source_holder['source_x'].value_counts()
len_sources_series = len(sources_series.index)

df_total_counts_source_x  = pd.DataFrame(columns=['total_counts'], index = sources_series.i
df_total_counts_source_x ['total_counts'] = 0

i = 0
while i < len_sources_series :
    x = 0
    while x < len_df_source_holder:
        if df_source_holder['source_x'][x] == sources_series.index[i]:
            df_total_counts_source_x ['total_counts'][sources_series.index[i]] = df_total_c
        x = x + 1
    i = i + 1

df_total_counts_source_x_sorted = df_total_counts_source_x.sort_values(by="total_counts", a
```

In [41]:

```
x = df_total_counts_source_x_sorted.index
y = df_total_counts_source_x_sorted['total_counts'].values

plt.bar(x,y)
plt.axhline(y=len(CORD19_CSV), linewidth=2,linestyle='--', color='r')


total_publications, = plt.plot(len(CORD19_CSV), label='Total publications', linestyle='--',
plt.legend(bbox_to_anchor=(1.05, 1), loc=2, borderaxespad=0.)

plt.title("Accumulated count of sources")
plt.xlabel("Name of sources")
plt.ylabel("Count")
plt.show()
```



Overall, PMC and Medline participate mostly as a source within this dataset. It is important to obey that several sources can contribute to one publication.

Checking the column "license"

For the column license, it might be interesting to create an overview of all used licenses.

In [42]:

```
license_counted = license.value_counts()
license_counted
```

Out[42]:

```
no-cc          29395
cc-by          23384
els-covid       7374
medrxiv         5401
arxiv           3141
cc-by-nc        2947
biorxiv         2325
cc-by-nc-nd     1860
green-oa         385
bronze-oa        352
cc-by-nc-sa      349
cc0              312
cc-by-nd         130
hybrid-oa         71
gold-oa           20
cc-by-sa           2
Name: license, dtype: int64
```

For the publications, there are several licenses used.

Checking the column "publish_time"

For the column publish_time, it might be interesting to present periods when most of the research was published.

In [43]:

```
publish_time_counted = publish_time.value_counts()
publish_time_counted
```

Out[43]:

```
2020-12-31    362
2020-09-30    346
2021-01-06    329
2020-09-18    327
2020-05-26    310
             ...
2010-10-22      1
2018-11-11      1
2008-07-13      1
2010-11-27      1
2012-12-22      1
Name: publish_time, Length: 5599, dtype: int64
```

Due to the given dates, it can be obtained that there is research published before COVID-19. For plotting purposes, the data is transformed into a DataFrame.

```
In [44]:
```

```
df_publish_time_counted = pd.DataFrame(columns=['publish_time', 'count'])
df_publish_time_counted['publish_time']= publish_time_counted.index
df_publish_time_counted['count']= publish_time_counted.values
df_publish_time_counted.sort_values(by=['publish_time'])
```

```
Out[44]:
```

	publish_time	count
4736	1962-09-01	1
4535	1964-01-01	1
4712	1964-02-01	1
5162	1968-07-01	1
4068	1970-01-01	2
...	...	...
4621	2021-06-15	1
2445	2021-06-30	4
4011	2021-07-13	2
2977	2021-11-30	3
2066	2021-12-31	5

5599 rows × 2 columns

```
In [45]:
```

```
test_def = df_publish_time_counted.sort_values(by=['publish_time'])
test_def
```

```
Out[45]:
```

	publish_time	count
4736	1962-09-01	1
4535	1964-01-01	1
4712	1964-02-01	1
5162	1968-07-01	1
4068	1970-01-01	2
...	...	...
4621	2021-06-15	1
2445	2021-06-30	4
4011	2021-07-13	2
2977	2021-11-30	3
2066	2021-12-31	5

5599 rows × 2 columns

To distinguish research between COVID-related and non-COVID-related publications, two timestamps are considered. For COVID-related research, publish dates after 2019/12/31 are taken into account. Whereas, publish dates for non-COVID-related research investigates publications before 2020/1/1.

In [46]:

```
date_before_2020 = pd.Timestamp(2020, 1, 1)
date_after_2020 = pd.Timestamp(2019, 12, 31)
df_before_2020 = CORD19_CSV[CORD19_CSV['publish_time'] < date_before_2020]
df_after_2020 = CORD19_CSV[CORD19_CSV['publish_time'] > date_after_2020]
```

To see which periods have more publications, the length of the corresponding DataFrame is shown.

As a result, most of the publications are published after 2019/31/12 which means that the dataset contains mostly research linked to COVID-19. Nevertheless, there is a considerable amount of publications before COVID-19.

In [47]:

```
count_before_2020 = len(df_before_2020)
```

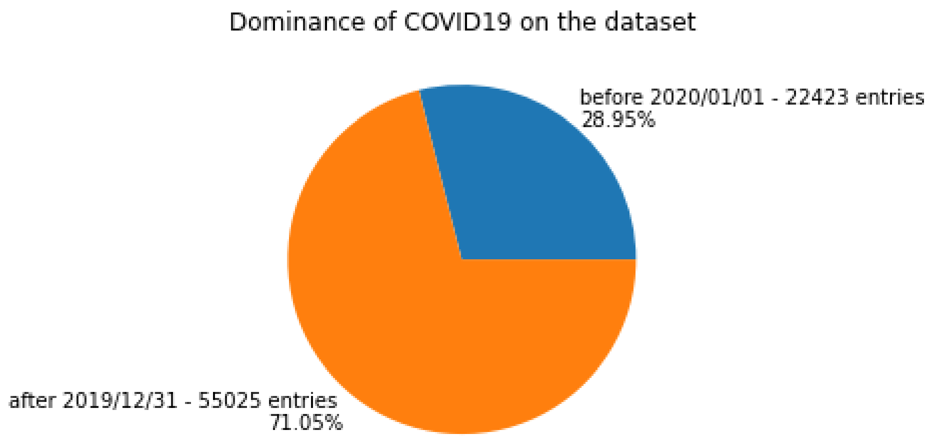
In [48]:

```
count_after_2020 = len(df_after_2020)
```

Therefore, a pie chart is plotted to present the ratio.

In [49]:

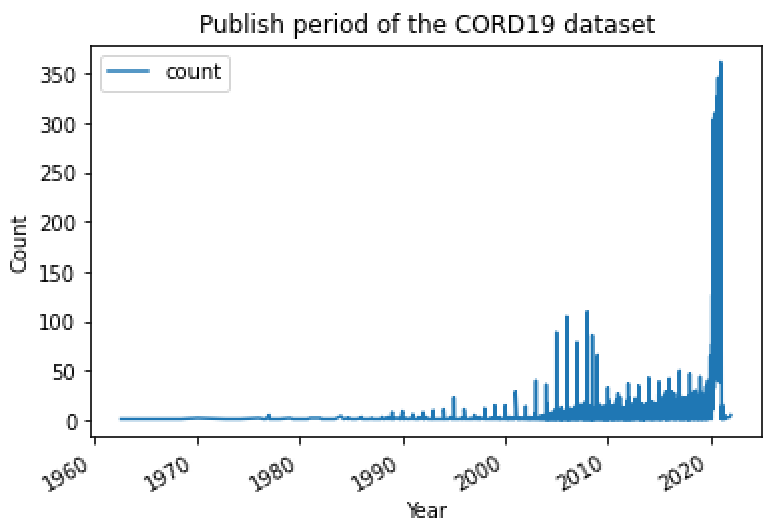
```
y = np.array([count_before_2020, count_after_2020])

plt.title("Dominance of COVID19 on the dataset")
pie_labels = ["before 2020/01/01 - " + str(count_before_2020) + " entries \n" + str(round(c
              "after 2019/12/31 - " + str(count_after_2020) + " entries \n" + str(round(cou
plt.pie(y, labels = pie_labels)
plt.show()
```



For this, all publication dates are shown within the next plot.

In [50]:

```
df_publish_time_counted.plot(x ='publish_time', y='count', kind = 'line')
plt.title("Publish period of the CORD19 dataset")
plt.xlabel("Year")
plt.ylabel("Count")
plt.show()
```



Due to this plot, several observations are created.

- The publications are starting from the 1960s which means that there were already other SARS viruses under the investigation of researchers.
- The plot shows four swings with varying extents. Therefore, the four major swings can be affiliated with corresponding SARS-pandemics.
  1. ~1995
  2. ~2001
  3. 2005-2010
  4. 2020
- The extent of the swings seems to have a relationship accordingly to the impact of the occurring pandemic. More widespread and global SARS outbreaks seem to indicate a high count of publications. Whereas, regional and smaller outbreaks are reflected with fewer publications.

In [51]:

```
threshold_source_x = len(CORD19_CSV)
threshold_source_x
```

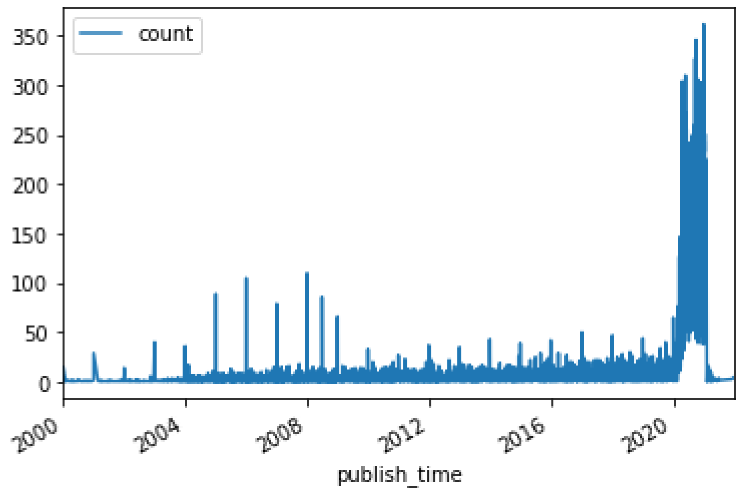
Out[51]:

```
77448
```

For further investigation, the next plot will target the last two decades.

In [52]:

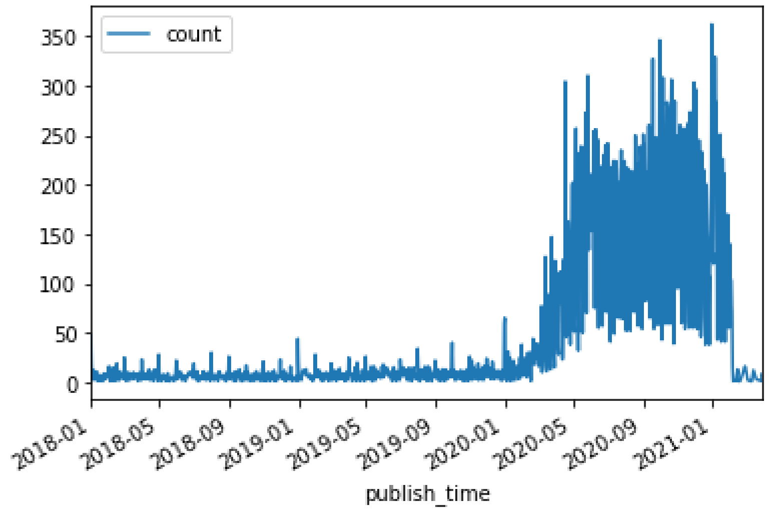
```
df_publish_time_counted.plot(x ='publish_time', y='count', kind = 'line', xlim=(pd.Timestam
plt.show()
```



In the plot above, two different SARS-outbreak can be seen. In the first decade of this century, the SARS outbreak seems to have less impact because less research was published. The latest SARS outbreak surpassed previous ones by impact and intensity which can be seen in the next plot.

In [53]:

```
df_publish_time_counted.plot(x ='publish_time', y='count', kind = 'line', xlim=(pd.Timestam
plt.show()
```



Checking the column "journal"

For this inspection, the NaN values of the column "journal" need to be considered.

```
In [54]:
```

```
find_shared_values(journal)
```

```
Out[54]:

60166
```

Most of the values of the column are shared ones. It is important to know that each row of the DataFrame is assigned to one journal or none. Thusly, the distribution of journals will be investigated.

```
In [55]:
```

```
journal_counted = journal.value_counts()
journal_counted.head(20)
```

```
Out[55]:

bioRxiv                                2726
PLoS One                               2268
Int J Environ Res Public Health        1154
Sci Rep                                 957
Viruses                                 613
Virology                                424
BMC Infect Dis                          421
J Med Virol                             394
Arch Virol                              386
Emerg Infect Dis                        383
PLoS Pathog                             356
BMJ Open                                352
medRxiv                                 344
Virol J                                 333
Front Psychol                           326
Clin Infect Dis                         316
Computational Science - ICCS 2020       296
Int J Mol Sci                           289
Front Immunol                           286
BMC Public Health                       271
Name: journal, dtype: int64
```

Compared to other columns, the distribution of journals is equally.

Checking the column "url"

The column "url" will be investigated to present contributing organisations by hostnames.

```
In [56]:
```

```
url_counted = url.value_counts()
url_counted
```

```
Out[56]:
```

```
https://api.elsevier.com/content/article/pii/S1386653220303589; (https://ap
i.elsevier.com/content/article/pii/S1386653220303589;) https://www.ncbi.nlm.
nih.gov/pubmed/32891938/; (https://www.ncbi.nlm.nih.gov/pubmed/32891938/;) h
ttps://www.sciencedirect.com/science/article/pii/S1386653220303589?v=s5; (ht
tps://www.sciencedirect.com/science/article/pii/S1386653220303589?v=s5;) htt
ps://doi.org/10.1016/j.jcv.2020.104616 (https://doi.org/10.1016/j.jcv.2020.1
04616)    1
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7686826/ (https://www.ncbi.nlm.
nih.gov/pmc/articles/PMC7686826/)
1
https://www.ncbi.nlm.nih.gov/pmc/articles/PMC7286207/ (https://www.ncbi.nlm.
nih.gov/pmc/articles/PMC7286207/)
1
http://medrxiv.org/cgi/content/short/2020.07.27.20159996v1?rss=1; (http://me
drxiv.org/cgi/content/short/2020.07.27.20159996v1?rss=1;) https://doi.org/1
0.1101/2020.07.27.20159996 (https://doi.org/10.1101/2020.07.27.20159996)
1
https://www.ncbi.nlm.nih.gov/pubmed/32324273/; (https://www.ncbi.nlm.nih.go
v/pubmed/32324273/;) https://doi.org/10.1002/cncr.32919 (https://doi.org/10.
1002/cncr.32919)
1

..
https://doi.org/10.1002/jdd.12364; (https://doi.org/10.1002/jdd.12364;) http
s://www.ncbi.nlm.nih.gov/pubmed/32813876/ (https://www.ncbi.nlm.nih.gov/pubm
ed/32813876/)
1
https://arxiv.org/pdf/2005.13754v1.pdf (https://arxiv.org/pdf/2005.13754v1.p
df)
1
https://doi.org/10.1371/journal.pone.0242474; (https://doi.org/10.1371/journ
al.pone.0242474;) https://www.ncbi.nlm.nih.gov/pubmed/33216795/ (https://ww
w.ncbi.nlm.nih.gov/pubmed/33216795/)
1
https://doi.org/10.1186/s13063-020-04997-6; (https://doi.org/10.1186/s13063-
020-04997-6;) https://www.ncbi.nlm.nih.gov/pubmed/33509275/ (https://www.ncb
i.nlm.nih.gov/pubmed/33509275/)
1
https://www.ncbi.nlm.nih.gov/pubmed/32238612/ (https://www.ncbi.nlm.nih.gov/
pubmed/32238612/)
1
Name: url, Length: 77448, dtype: int64
```

Some entries contain more than one URL pointing to a publication. Consequently, entries with more than one URL are split into a series containing one URL per tuple.

In [57]:

```
%%time
CORD19_CSV.url = CORD19_CSV.url.str.replace('[;]', '')
url = CORD19_CSV.url
url_list= []
for row in url:
    row_string = re.split('\s+', row)
    for i in row_string:
        url_list.append(urlparse(i).netloc)
url_list
```

```
Wall time: 9.09 s
```

Next, all URLs are counted to see which host publishes research and to which extent. For this, it is crucial to know that the same publication is accessible via different hosts.

In [58]:

```
url_series = pd.Series(url_list)
url_counted = url_series.value_counts()
url_counted.head(20)
```

Out[58]:

```
www.ncbi.nlm.nih.gov       64918
doi.org                    54410
api.elsevier.com           17660
www.sciencedirect.com      17660
medrxiv.org                 4995
arxiv.org                   3340
europepmc.org                313
academic.oup.com             118
www.cell.com                  46
link.springer.com             42
onlinelibrary.wiley.com       36
www.nature.com                23
jvi.asm.org                   16
journals.iucr.org             14
www.jbc.org                   14
www.thelancet.com              8
pubs.acs.org                   7
www.tandfonline.com            6
www.biorxiv.org                4
ajp.amjpathol.org              3
dtype: int64
```

As a result, the National Center for Biotechnology Information (NCBI) is responsible for almost all publications. This means that the publications are hosted by different organisations holding and securing the research as a possible backup.

Checking the column "software"

The analysis of the column software is conducted in the Jupyter Notebook named "CORD-19-software-counting-cs5099".