CORD-19 Software counting

This notebook is designated to count software mentions based on the CORD19 dataset from:

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 Levenshtein as lev
from fuzzywuzzy import fuzz
```

Get the data and save it to a variable.

In [2]:

```
CORD19_CSV = pd.read_csv('../data/cord-19/CORD19_software_mentions.csv' , converters={'soft
```

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

In [3]:

```
CORD19_CSV.head(20)
```

Out[3]:

	paper_id	doi	title	source_x	licer
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	cc
2	000122a9a774ec76fa35ec0c0f6734e7e8d0c541	10.1016/j.rec.2020.08.002	Impact of COVID-19 on ST-segment elevation myo...	Elsevier; Medline; PMC	nc
3	0001418189999fea7f7cbe3e82703d71c85a6fe5	10.1016/j.vetmic.2006.11.026	Absence of surface expression of feline infect...	Elsevier; Medline; PMC	nc
			Detection of		

The dataset contains nine different columns. Thusly, the next lines of this notebook explore the column "software". Therefore, the column software will be saved to an object.

In [4]:

```
software = CORD19_CSV.software
software
```

Out[4]:

```
0                    ['Google Street View']
1                                 ['SEGG']
2                     ['STATA',  'IAMCEST']
3                                 ['SPSS']
4        ['R',  'MassARRAY Typer Analyzer']
                        ...
77443                           ['UpToDate']
77444                   ['SALib',  'Panda']
77445                ['Prism',  'GraphPad']
77446    ['R package circular',  'R',  'R']
77447       ['GRAM',  'R studio',  'Stata']
Name: software, Length: 77448, dtype: object
```

In [5]:

```
len_software_multiple_entries = len(software)
len_software_multiple_entries
```

Out[5]:

```
77448
```

The software object contains 77448 rows. In each row, there are software mentions. Some rows contain more than one mention. For instance, row two and four have two mentions. Consequently, the object needs to be transformed into an object which contains solely one software entry per row.

In [6]:

```
software = software.explode(ignore_index = True)
```

Remove the brackets around the software mentions.

In [7]:

```
software = software.str.replace('\'', '')
```

Control the software object and inspect if each row contains only one entry.

In [8]:

```
software
```

Out[8]:

```
0         Google Street View
1                       SEGG
2                      STATA
3                    IAMCEST
4                       SPSS
                ...
558787                     R
558788                     R
558789                  GRAM
558790               R studio
558791                  Stata
Name: software, Length: 558792, dtype: object
```

In [9]:

```
len_software_single_entries = len(software)
len_software_single_entries
```

Out[9]:

```
558792
```

Now, the object contains solely one entry per row and has a length of 558792.

In [10]:

```
average_entries_per_row = len_software_single_entries/len_software_multiple_entries
average_entries_per_row
```

Out[10]:

```
7.215060427641773
```

Due to the alignment of the software object, it can be obtained that the dataset contains on average 7.2 software mentions per row. Furthermore, the value_counts function will be used to minimise the number of rows by checking identical duplicates.

In [11]:

```
software.value_counts(dropna=False)
```

Out[11]:

```
 R                8389
 SPSS             4738
SPSS              4472
 BLAST            3166
 Excel            2666
                  ...
 EFS Questback       1
AzureSpot            1
 Antigenic           1
AVL BOOST            1
Mystery Miner        1
Name: software, Length: 120264, dtype: int64
```

The function value_counts reduced the number of rows to 120279. Nevertheless, the dtype counts same software mentions as distinct. For instance, "SPSS" is listed twice with two varied numbers. For this, the datatype will be converted to a dictionary to check for possible empty spaces.

In [12]:

```
software_dict = software.to_dict()
software_dict
```

Out[12]:

```
{0: 'Google Street View',
 1: 'SEGG',
 2: 'STATA',
 3: ' IAMCEST',
 4: 'SPSS',
 5: 'R',
 6: ' MassARRAY Typer Analyzer',
 7: 'Wechat',
 8: ' SPSS Statistics',
 9: 'Statistical Package for Social Sciences (SPSS)',
 10: ' BD CBA',
 11: 'STATA',
 12: ' STATA',
 13: ' Statacorp',
 14: 'SPSS',
 15: ' SPSS',
 16: 'geNorm',
 17: ' GraphPad Prism',
```

The dictionary contains mentions with space at the first position. This means that the data is not cleaned properly for the value_counts() function. Therefore, the function remove_empty_spaces(d) takes a dictionary and removes spaces at the first position of a string.

In [13]:

```
def remove_empty_spaces(dic):
    """ Function removing an empty space at the first position of a string.
    """
    for i in dic:
        if dic[i][:1] == " ":
            dic[i] = dic[i].strip()
    return dic
```

In [14]:

```
software_dict = remove_empty_spaces(software_dict)
software_dict
```

Out[14]:

```
{0: 'Google Street View',
 1: 'SEGG',
 2: 'STATA',
 3: 'IAMCEST',
 4: 'SPSS',
 5: 'R',
 6: 'MassARRAY Typer Analyzer',
 7: 'Wechat',
 8: 'SPSS Statistics',
 9: 'Statistical Package for Social Sciences (SPSS)',
 10: 'BD CBA',
 11: 'STATA',
 12: 'STATA',
 13: 'Statacorp',
 14: 'SPSS',
 15: 'SPSS',
 16: 'geNorm',
 17: 'GraphPad Prism',
```

Now, the software mentions within the dictionary do not contain empty spaces at the first position of the string. For the use of value_counts(), the dictionary is converted to a pandas series.

In [15]:

```
software_series = pd.Series(software_dict)
software_series.value_counts()
```

Out[15]:

```
R                       10805
SPSS                     9210
GraphPad Prism           3986
Excel                    3856
BLAST                    3674
                        ...
Clarity integration         1
pathJC                      1
ANCOVA Global Test          1
sound                       1
Ophthalmology Match         1
Length: 102709, dtype: int64
```

Due to the removal of empty spaces, the length of the object decreased. To further minimise the length of the

object, all strings will be capitalized.

In [16]:

```
software_dict = software_series.to_dict()
```

In [17]:

```
def capitalize_mentions(dic):
    """ Function iterating a dictionary and capitalizing all strings.
    """
    for i in dic:
        dic[i] = dic[i].upper()
    return dic
```

In [18]:

```
software_dict = capitalize_mentions(software_dict)
software_dict
```

Out[18]:

```
{0: 'GOOGLE STREET VIEW',
 1: 'SEGG',
 2: 'STATA',
 3: 'IAMCEST',
 4: 'SPSS',
 5: 'R',
 6: 'MASSARRAY TYPER ANALYZER',
 7: 'WECHAT',
 8: 'SPSS STATISTICS',
 9: 'STATISTICAL PACKAGE FOR SOCIAL SCIENCES (SPSS)',
 10: 'BD CBA',
 11: 'STATA',
 12: 'STATA',
 13: 'STATACORP',
 14: 'SPSS',
 15: 'SPSS',
 16: 'GENORM',
 17: 'GRAPHPAD PRISM',
```

In [19]:

```
software_series = pd.Series(software_dict)
software_series = software_series.value_counts()
software_series
```

Out[19]:

```
R                              10805
SPSS                            9229
GRAPHPAD PRISM                  4461
EXCEL                           4054
BLAST                           3943
                               ...
FPC R / CRAN                       1
HCA BROWSER                        1
NETSTAP                            1
POCOVIDSCREEN                      1
NEUROINTERVENTIONAL SUITE          1
Length: 89462, dtype: int64
```

Due to the capitalization of software mentions, the length of the object could be decreased. Subsequently, the fuzzy-wuzzy compare algorithm will be introduced. This algorithm is based on Levenshtein which checks the similarity of strings by various aspects.

In [20]:

```
def fuzzy_ratio_compare(str1, str2, th):
    """ Function to compare to strings based on a given threeshold.
    """
    ratio = fuzz.ratio(str1, str2)
    if(ratio > th):
        return True
    else:
        return False

def fuzzy_partial_ratio_compare(str1, str2, th):
    """ Function to compare to strings based on a given threeshold.
    """
    ratio = fuzz.partial_ratio(str1, str2)
    if(ratio > th):
        return True
    else:
        return False

def fuzzy_token_sort_ratio_compare(str1, str2, th):
    """ Function to compare to strings based on a given threeshold.
    """
    ratio = fuzz.token_sort_ratio(str1, str2)
    if(ratio > th):
        return True
    else:
        return False

def fuzzy_token_set_ratio_compare(str1, str2, th):
    """ Function to compare to strings based on a given threeshold.
    """
    ratio = fuzz.token_set_ratio(str1, str2)
    if(ratio > th):
        return True
    else:
        return False
```

Due to performance reasons, further investigation will be conducted with a subset of the initial series based on a limit. For this, the successive analysis will focus on the 10% of the data that have the most mentions. Therefore, the corresponding amount of 1/10 of the dataset is calculated.

In [21]:

```
ten_percent_position = int(round(len(software_series)*0.1, 0))
ten_percent_position
```

Out[21]:

```
8946
```

The lowest mention within the 10% of most common software is considered for the limit.

In [22]:

```
lowest_mention_of_ten_percent = software_series[ten_percent_position]
lowest_mention_of_ten_percent
```

Out[22]:

```
9
```

As nine is the lowest mention within the 10% of most common software, all mentions with nine or more matches are picked for further analysis.

In [23]:

```
software_series_shaped = software_series[software_series >= lowest_mention_of_ten_percent]
selection_limit = len(software_series_shaped)
software_series_shaped
```

Out[23]:

```
R                10805
SPSS              9229
GRAPHPAD PRISM    4461
EXCEL             4054
BLAST             3943
                  ...
TESSERACT            9
3DRNA                9
PEPSEQ               9
TBC2TARGET           9
YELP                 9
Length: 9351, dtype: int64
```

Converting the series to a DataFrame for comparison purposes. The index of the DataFrame is required for selecting rows.

In [24]:

```
ts = software_series_shaped.to_frame()
list_soft = []
list_matches = [0]
for i in range(len(ts)):
    list_soft.append(software_series_shaped.index[i])
    list_matches.append(software_series_shaped[i])
df_shaped = pd.DataFrame()
#Insert the column software and matches
df_shaped['Software'] = list_soft
df_shaped['Matches'] = list_matches[1:]
#Sort the DataFrame by numeric by matches and then alphabetical by software
df_shaped = df_shaped.sort_values(by=['Matches', "Software"])
#Reverse the Dataframe to present the most common software at index position 0
df_shaped = df_shaped[::-1]
df_shaped.reset_index(drop = True)
```

Out[24]:

	Software	Matches
0	R	10805
1	SPSS	9229
2	GRAPHPAD PRISM	4461
3	EXCEL	4054
4	BLAST	3943
...	...	...
9346	3DRNA	9
9347	2DST	9
9348	- MASK	9
9349	- CNN	9
9350	- CAPS	9

9351 rows × 2 columns

Replacing special characters to prevent the error "unterminated subpattern" at a later stage of this notebook.

In [25]:

```
df_shaped['Software'] = df_shaped.Software.str.replace('(','')
df_shaped['Software'] = df_shaped.Software.str.replace(')','')
```

The following function compares software mentions based on the fuzzy-wuzzy method. As a result, a blacklist with identified duplicates and a modified DataFrame is returned.

In [26]:

```
def unify_dataframe(df, th):
    """Match software mentions based on fuzzywuzzy algorithm
    """
    df_holder = df
    blacklist = set()
    dict_matches = df.Matches
    for i in range(len(df)):
        for j in range(i + 1, len(df)):
            if df['Software'][i] not in blacklist:
                if(fuzzy_token_set_ratio_compare(df['Software'][i], df['Software'][j], th))
                    print("Position: "+str(i)+"/"+str(len(df))+" -> '"+df['Software'][i]+"'
                          " mentions matched with '" + df['Software'][j] + "' mentioned " +
                    df.iloc[i, df.columns.get_loc('Matches')] = int(df['Matches'].loc[i] +
                    blacklist.add(df['Software'][j])
    df_holder = df
    return df_holder, blacklist
```

In [27]:

```
%%time
threeshold = 84
df_returned = unify_dataframe(df_shaped, threeshold)
df_unified = df_returned[0]
blacklist = df_returned[1]
```

```
Position: 0/9351 -> 'R' with 10805 mentions matched with 'R PACKAGE' menti
oned 313 times
Position: 0/9351 -> 'R' with 11118 mentions matched with 'R FOUNDATION FOR
STATISTICAL COMPUTING' mentioned 280 times
Position: 0/9351 -> 'R' with 11398 mentions matched with 'R CORE TEAM' men
tioned 207 times
Position: 0/9351 -> 'R' with 11605 mentions matched with 'R STUDIO' mentio
ned 148 times
Position: 0/9351 -> 'R' with 11753 mentions matched with 'R DEVELOPMENT CO
RE TEAM' mentioned 86 times
Position: 0/9351 -> 'R' with 11839 mentions matched with 'R SCRIPT' mentio
ned 84 times
Position: 0/9351 -> 'R' with 11923 mentions matched with 'MASK R - CNN' me
ntioned 71 times
Position: 0/9351 -> 'R' with 11994 mentions matched with 'R FOUNDATION FOR
STATISTICAL' mentioned 63 times
Position: 0/9351 -> 'R' with 12057 mentions matched with 'R CORE' mentione
d 62 times
Position: 0/9351 -> 'R' with 12119 mentions matched with 'R PROJECT FOR ST
```

Next, the DataFrames presents the new aggregation numbers.

In [28]:

```
df_unified
```

Out[28]:

	Software	Matches
0	R	13163
1	SPSS	11290
2	GRAPHPAD PRISM	8499
3	EXCEL	4319
4	BLAST	6711
...	...	...
9347	3DRNA	9
8894	2DST	9
9108	- MASK	9
8868	- CNN	9
8883	- CAPS	9

9351 rows × 2 columns

The blacklist contains the matched duplicates which means that they need to be removed from the DataFrame.

In [29]:

```
for i in blacklist:
    name_of_index = df_unified[ df_unified['Software'] == i ].index
    df_unified.drop(name_of_index, inplace = True)
```

For comparison purposes, the DataFrame is sorted in descending order by matches and then alphabetical by software.

In [30]:

```
df_unified = df_unified.sort_values(by=['Matches', "Software"])
#Reverse the Dataframe to present the most common software at index position 0
df_unified = df_unified[::-1]
df_unified
```

Out[30]:

	Software	Matches
0	R	13163
1	SPSS	11290
2	GRAPHPAD PRISM	8499
4	BLAST	6711
3	EXCEL	4319
...	...	...
8611	7VINCUT	9
8856	6GCVAE	9
9085	4D	9
9347	3DRNA	9
8894	2DST	9

7168 rows × 2 columns

To verify the removal of duplicates, the length of the DataFrame is outputted.

In [31]:

```
len_df_unified = len(df_unified)
len_df_unified
```

Out[31]:

```
7168
```

For the highest percentile of software mentiones, the implemented algorithm leads to a reduction of approximately 23.4%.

In [32]:

```
reduction = round((1-len_df_unified/selection_limit) * 100, 2)
reduction
```

Out[32]:

```
23.35
```

To investigate the position change of software mentions, the following algorithm compares its index position to the sorted position by matches.

In [33]:

```
list_change = []
for i in range(len(df_unified)):
    dif = df_unified.index[i]-i
    if(dif > 0):
        list_change.append("+"+str(dif))
    else:
        list_change.append(df_unified.index[i]-i)
df_unified['Change'] = list_change
df_unified.head(20)
```

Out[33]:

	Software	Matches	Change
0	R	13163	0
1	SPSS	11290	0
2	GRAPHPAD PRISM	8499	0
4	BLAST	6711	+1
3	EXCEL	4319	-1
5	STATA	4048	0
10	MEGA	3428	+4
6	SAS	3399	-1
12	IMAGEJ	2779	+4
7	MATLAB	2710	-2
8	GOOGLE SCHOLAR	2485	-2
11	NET	2411	0
21	CLUSTALW	2162	+9
24	AUTODOCK VINA	2100	+11
13	SCOPUS	1845	-1
14	PYTHON	1620	-1
20	GOOGLE TRENDS	1538	+4
18	REDCAP	1464	+1
16	GISAID	1442	-2
60	CLUSTAL OMEGA	1381	+41

Assigning the outcome of this notebook to a new DataFrame for the classification notebook. The outcome is stored on an external file.

In [34]:

```
df_software_mentions = df_unified
df_software_mentions.to_pickle('software_mentions_CS5099.pkl')
```