Analysis of English data

This program imports the files generated by the parser (divided by month to put less load on the memory) and analyses them. It is **not language agnostic:** correct linguistic settings must be specified in **"setting up", "NLP" and "additional rules".**

First some additional rules for NER are defined. Some are general, some are language-specific, as specified in the relevant section.

The files are opened and preprocessed, then lemma frequency and NER frequency are calculated per each month and in the whole corpus. **important:** in case of empty months (so, when analysing less than one year of data) **remember to exclude them from the mean,** otherwise the mean will be distorted by the empty months.

All the dataframes are exported as CSV files for further analisys or for data visualization.

Setting up

Remember to check the folder paths.

In [1]:

```
%%capture
from tqdm.notebook import tqdm as tqdm #for progress bars
tqdm().pandas()
```

In [2]:

```
from pathlib import Path
import os
import pandas as pd
import spacy
from collections import Counter
from datetime import datetime

# Measure execution time
start_time = datetime.now()

# folder paths (1. containing a substet homogeneous by language and divided by date; 2.
for exports)
folder = Path("C://Users/copam/Desktop/jupyter test/exports_parser/EN")
export = Path("C://Users/copam/Desktop/jupyter test/exports_NLP/EN")

# month files (if need be, add other months here and in the list below).
january = open(os.path.join(folder, "1.txt"),encoding="utf8").read()
february = open(os.path.join(folder, "2.txt"),encoding="utf8").read()
march = open(os.path.join(folder, "3.txt"),encoding="utf8").read()
april = open(os.path.join(folder, "4.txt"),encoding="utf8").read()
may = open(os.path.join(folder, "5.txt"),encoding="utf8").read()
june = open(os.path.join(folder, "6.txt"),encoding="utf8").read()
july = open(os.path.join(folder, "7.txt"),encoding="utf8").read()
august = open(os.path.join(folder, "8.txt"),encoding="utf8").read()
september = open(os.path.join(folder, "9.txt"),encoding="utf8").read()
october = open(os.path.join(folder, "10.txt"),encoding="utf8").read()
november = open(os.path.join(folder, "11.txt"),encoding="utf8").read()
december = open(os.path.join(folder, "12.txt"),encoding="utf8").read()

months = [january,february,march, april, may, june, july, august, september, october, n
ovember, december]
```

NLP

Remember to check the language and the max_lenght. References on models here: https://spacy.io/models (https://spacy.io/models)

In [3]:

```
nlp = spacy.load('en_core_web_md')
spacy_stopwords = spacy.lang.en.stop_words.STOP_WORDS
nlp.max_length = 100000000
```

Additional rules for COVID19 NER

Remember to adapt for the specific language (below the comment). References here: https://spacy.io/usage/rule-based-matching#models-rules (https://spacy.io/usage/rule-based-matching#models-rules)

In [4]:

```
from spacy.pipeline import EntityRuler
ruler = EntityRuler(nlp)
ruler.overwrite_ents = True
patterns = [{"label": "COVID19", "pattern": "Covid"},
            {"label": "COVID19", "pattern": "covid"},
            {"label": "COVID19", "pattern": "Covid19"},
            {"label": "COVID19", "pattern": "covid19"},
            {"label": "COVID19", "pattern": "Covid 19"},
            {"label": "COVID19", "pattern": "Covid-19"},
            {"label": "COVID19", "pattern": "covid-19"},
            {"label": "COVID19", "pattern": "covid 19"},
            {"label": "COVID19", "pattern": "Corvid"},
            {"label": "COVID19", "pattern": "corvid"},
            {"label": "COVID19", "pattern": "Corvid19"},
            {"label": "COVID19", "pattern": "corvid19"},
            {"label": "COVID19", "pattern": "Corvid 19"},
            {"label": "COVID19", "pattern": "corvid 19"},
            {"label": "COVID19", "pattern": "Coronavirus"},
            {"label": "COVID19", "pattern": "coronavirus"},
            {"label": "COVID19", "pattern": "Corona virus"},
            {"label": "COVID19", "pattern": "Corona Virus"},
            {"label": "COVID19", "pattern": "corona virus"},
            {"label": "COVID19", "pattern": "COVID"},
            {"label": "COVID19", "pattern": "COVID19"},
            {"label": "COVID19", "pattern": "COVID 19"},
            {"label": "COVID19", "pattern": "2019-nCoV"},
            {"label": "COVID19", "pattern": "ncov"},
            {"label": "COVID19", "pattern": "nCoV"},
            {"label": "COVID19", "pattern": "sars"},
            {"label": "COVID19", "pattern": "SARS"},
            {"label": "COVID19", "pattern": "SARS-CoV2"},
## language-specific rules
## consider adding rules for scarce resources allocation, anxiety, ...
            {"label": "COVID19r", "pattern": "Wuhan virus"},
            {"label": "COVID19r", "pattern": "Wuhan Virus"},
            {"label": "COVID19r", "pattern": "virus from Wuhan"},
            {"label": "COVID19r", "pattern": "Virus from Wuhan"},
            {"label": "COVID19r", "pattern": "Chinese virus"},
            {"label": "COVID19r", "pattern": "chinese virus"}
           ]
ruler.add_patterns(patterns)
nlp.add_pipe(ruler)
```

In [5]:

```
file_doc = {}
for x in tqdm(months):
    file_doc[x] = nlp(x)
```

Preprocessing

In [6]:

```
# Definition of the preprocessing functions
def is_token_allowed(token):
    if (not token or not token.string.strip() or token.is_stop or token.is_punct or tok
en in spacy_stopwords):
        return False
    return True

def preprocess_token(token):
    if is_token_allowed:
        return token.lemma_.strip().lower()
```

In [7]:

```
# Actual preprocessing
complete_filtered_tokens = {}
for x in tqdm(months):
    complete_filtered_tokens[x] = [preprocess_token(token) for token in file_doc[x] if
is_token_allowed(token)]
```

Lemma frequency

calculates and exports lemma frequency, in general and per month.

In [8]:

```
lemmas_freq = {}
for x in tqdm(months):
    lemmas_freq[x] = Counter(complete_filtered_tokens[x]).most_common()
```

In [9]:

```
## january
lemmas_freq_january = lemmas_freq[january]
df_lemmas_freq_january = pd.DataFrame(lemmas_freq_january, columns={'Lemma':[1],'Count'
:[2]})
df_lemmas_freq_january.index += 1
df_lemmas_freq_january.to_csv(os.path.join(export, "lemmas\lemmas-frequency-1.csv"))

## february
lemmas_freq_february = lemmas_freq[february]
df_lemmas_freq_february = pd.DataFrame(lemmas_freq_february, columns={'Lemma':[1],'Coun
t':[2]})
df_lemmas_freq_february.index += 1
df_lemmas_freq_february.to_csv(os.path.join(export, "lemmas\lemmas-frequency-2.csv"))

## march
lemmas_freq_march = lemmas_freq[march]
df_lemmas_freq_march = pd.DataFrame(lemmas_freq_march, columns={'Lemma':[1],'Count':[2
]})
df_lemmas_freq_march.index += 1
df_lemmas_freq_march.to_csv(os.path.join(export, "lemmas\lemmas-frequency-3.csv"))

## april
lemmas_freq_april = lemmas_freq[april]
df_lemmas_freq_april = pd.DataFrame(lemmas_freq_april, columns={'Lemma':[1],'Count':[2
]})
df_lemmas_freq_april.index += 1
df_lemmas_freq_april.to_csv(os.path.join(export, "lemmas\lemmas-frequency-4.csv"))

## may
lemmas_freq_may = lemmas_freq[may]
df_lemmas_freq_may = pd.DataFrame(lemmas_freq_may, columns={'Lemma':[1],'Count':[2]})
df_lemmas_freq_may.index += 1
df_lemmas_freq_may.to_csv(os.path.join(export, "lemmas\lemmas-frequency-5.csv"))

## june
lemmas_freq_june = lemmas_freq[june]
df_lemmas_freq_june = pd.DataFrame(lemmas_freq_june, columns={'Lemma':[1],'Count':[2]})
df_lemmas_freq_june.index += 1
df_lemmas_freq_june.to_csv(os.path.join(export, "lemmas\lemmas-frequency-6.csv"))

## july
lemmas_freq_july = lemmas_freq[july]
df_lemmas_freq_july = pd.DataFrame(lemmas_freq_july, columns={'Lemma':[1],'Count':[2]})
df_lemmas_freq_july.index += 1
df_lemmas_freq_july.to_csv(os.path.join(export, "lemmas\lemmas-frequency-7.csv"))

## august
lemmas_freq_august = lemmas_freq[august]
df_lemmas_freq_august = pd.DataFrame(lemmas_freq_august, columns={'Lemma':[1],'Count':[
2]})
df_lemmas_freq_august.index += 1
df_lemmas_freq_august.to_csv(os.path.join(export, "lemmas\lemmas-frequency-8.csv"))

## september
lemmas_freq_september = lemmas_freq[september]
df_lemmas_freq_september = pd.DataFrame(lemmas_freq_september, columns={'Lemma':[1],'Co
unt':[2]})
df_lemmas_freq_september.index += 1
df_lemmas_freq_september.to_csv(os.path.join(export, "lemmas\lemmas-frequency-9.csv"))
```

```
## october
lemmas_freq_october = lemmas_freq[october]
df_lemmas_freq_october = pd.DataFrame(lemmas_freq_october, columns={'Lemma':[1],'Count'
:[2]})
df_lemmas_freq_october.index += 1
df_lemmas_freq_october.to_csv(os.path.join(export, "lemmas\lemmas-frequency-10.csv"))

## november
lemmas_freq_november = lemmas_freq[november]
df_lemmas_freq_november = pd.DataFrame(lemmas_freq_november, columns={'Lemma':[1],'Coun
t':[2]})
df_lemmas_freq_november.index += 1
df_lemmas_freq_november.to_csv(os.path.join(export, "lemmas\lemmas-frequency-11.csv"))

## december
lemmas_freq_december = lemmas_freq[december]
df_lemmas_freq_december = pd.DataFrame(lemmas_freq_december, columns={'Lemma':[1],'Coun
t':[2]})
df_lemmas_freq_december.index += 1
df_lemmas_freq_december.to_csv(os.path.join(export, "lemmas\lemmas-frequency-12.csv"))
```

Trends of the lemmas per month

"general" takes the data from the whole corpus. "mean" is the mean of the months.

Important: in case of empty months (so, when analysing less than one year of data) **remember to exclude them from the mean!**

In [10]:

```
# List of all lemma dataframes
df_lemmas_freq_all = [df_lemmas_freq_january,
                      df_lemmas_freq_february,
                      df_lemmas_freq_march,
                      df_lemmas_freq_april,
                      df_lemmas_freq_may,
                      df_lemmas_freq_june,
                      df_lemmas_freq_july,
                      df_lemmas_freq_august,
                      df_lemmas_freq_september,
                      df_lemmas_freq_october,
                      df_lemmas_freq_november,
                      df_lemmas_freq_december]

# Loop for index and series
L = []
for x in df_lemmas_freq_all:
    x = x.set_index('Lemma')
    L.append(pd.Series(x.values.tolist(), index=x.index))

# All together
df_lemmas_freq_all = pd.concat(L, axis=1, keys=('1','2','3','4','5','6','7','8','9','1
0','11','12'))
df_lemmas_freq_all = df_lemmas_freq_all.fillna('0')
for month in df_lemmas_freq_all:
    df_lemmas_freq_all[month] = df_lemmas_freq_all[month].str[0]

df_lemmas_freq_all = df_lemmas_freq_all.astype('int')

# Calculate the total
lemmasums = df_lemmas_freq_all.iloc[:, [0,1,2,3,4,5,6,7,8,9,10,11]].sum(axis=1)
df_lemmas_freq_all = pd.concat([df_lemmas_freq_all, lemmasums], axis = 1)
df_lemmas_freq_all = df_lemmas_freq_all.rename(columns={0: "total"})

# Calculate the mean of the months
lemmameans = df_lemmas_freq_all.iloc[:, [0,1,2,3,4,5,6]].mean(axis=1) ## In case of emp
ty months, exclude them from the mean here! Numbers are indices, where 0 is january and
11 is december
df_lemmas_freq_all = pd.concat([df_lemmas_freq_all, lemmameans], axis = 1)
df_lemmas_freq_all = df_lemmas_freq_all.rename(columns={0: "mean"})
df_lemmas_freq_all["mean"] = (df_lemmas_freq_all["mean"].astype('float')).round(2)

# Reorder and reindex
total_col = df_lemmas_freq_all.pop("total")
df_lemmas_freq_all.insert(0, "total", total_col)
df_lemmas_freq_all.reset_index(level=0, inplace=True)
df_lemmas_freq_all = df_lemmas_freq_all.sort_values(by=['total'], ascending=False)
df_lemmas_freq_all.index = pd.RangeIndex(len(df_lemmas_freq_all.index))
df_lemmas_freq_all.index += 1
df_lemmas_freq_all["lemma"] = df_lemmas_freq_all["index"]
df_lemmas_freq_all = df_lemmas_freq_all[['lemma','total','1','2','3','4','5','6','7',
'8','9','10','11','12','mean']]


# Export and display
df_lemmas_freq_all.to_csv(os.path.join(export, "lemmas\lemmas-frequency-timeseries.csv"
))
display(df_lemmas_freq_all.head(20))
```

```
<ipython-input-10-0372892eda72>:19: DeprecationWarning: The default dtype
for empty Series will be 'object' instead of 'float64' in a future versio
n. Specify a dtype explicitly to silence this warning.
  L.append(pd.Series(x.values.tolist(), index=x.index))
```

	lemma	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	say	10024	511	1183	2293	1948	1336	1525	1228	0	0	0	0	0	1432.00
2	|	6356	33	60	317	788	3489	547	1122	0	0	0	0	0	908.00
3	coronavirus	5696	246	634	1667	1197	596	902	454	0	0	0	0	0	813.71
4	2020	5388	110	169	713	1457	908	823	1208	0	0	0	0	0	769.71
5	covid-19	4452	0	102	657	1165	811	764	953	0	0	0	0	0	636.00
6	million	3937	32	129	538	981	720	608	929	0	0	0	0	0	562.43
7	year	3875	127	249	555	947	600	417	980	0	0	0	0	0	553.57
8	country	3755	125	330	858	786	567	557	532	0	0	0	0	0	536.43
9	people	3629	96	380	828	765	505	537	518	0	0	0	0	0	518.43
10	health	3620	129	409	715	753	588	525	501	0	0	0	0	0	517.14
11	new	3581	152	287	694	671	433	762	582	0	0	0	0	0	511.57
12	company	3507	70	194	656	959	486	448	694	0	0	0	0	0	501.00
13	case	3136	113	505	725	443	288	804	258	0	0	0	0	0	448.00
14	pandemic	2841	7	53	453	774	509	484	561	0	0	0	0	0	405.86
15	report	2722	77	237	465	589	297	594	463	0	0	0	0	0	388.86
16	virus	2699	174	490	590	431	378	311	325	0	0	0	0	0	385.57
17	swiss	2626	57	139	657	666	397	379	331	0	0	0	0	0	375.14
18	group	2594	31	96	422	764	340	367	574	0	0	0	0	0	370.57
19	include	2582	69	173	452	651	448	300	489	0	0	0	0	0	368.86
20	market	2579	135	209	518	593	259	373	492	0	0	0	0	0	368.43

NER

Calculates and exports named entity frequency, in general and per month. **Remember to check the export name.** References on NER tags here: https://spacy.io/api/annotation#named-entities (https://spacy.io/api/annotation#named-entities)

In [11]:

```
entity_list = {}
for x in tqdm(months):
    entity_list[x] = []
    for ent in file_doc[x].ents:
        entity_list[x].append((ent.text, ent.label_))

entity_counts = {}
for x in tqdm(months):
    entity_counts[x] = Counter(entity_list[x]).most_common()
    if not len(entity_counts[x]) == 0:
        enticat, count = zip(*entity_counts[x])
        entity, category = zip(*enticat)
        entity_counts[x] = tuple(zip(entity, category,count))

## january
entity_counts_january = entity_counts[january]
df_entity_counts_january = pd.DataFrame(entity_counts_january, columns={'Entity':[1],'C
ategory':[2],'Count':[3]})
df_entity_counts_january.index += 1
df_entity_counts_january.to_csv(os.path.join(export, "entities\entities-frequency-1.cs
v"))

## february
entity_counts_february = entity_counts[february]
df_entity_counts_february = pd.DataFrame(entity_counts_february, columns={'Entity':[1],
'Category':[2],'Count':[3]})
df_entity_counts_february.index += 1
df_entity_counts_february.to_csv(os.path.join(export, "entities\entities-frequency-2.cs
v"))

## march
entity_counts_march = entity_counts[march]
df_entity_counts_march = pd.DataFrame(entity_counts_march, columns={'Entity':[1],'Categ
ory':[2],'Count':[3]})
df_entity_counts_march.index += 1
df_entity_counts_march.to_csv(os.path.join(export, "entities\entities-frequency-3.csv"
))

## april
entity_counts_april = entity_counts[april]
df_entity_counts_april = pd.DataFrame(entity_counts_april, columns={'Entity':[1],'Categ
ory':[2],'Count':[3]})
df_entity_counts_april.index += 1
df_entity_counts_april.to_csv(os.path.join(export, "entities\entities-frequency-4.csv"
))

## may
entity_counts_may = entity_counts[may]
df_entity_counts_may = pd.DataFrame(entity_counts_may, columns={'Entity':[1],'Category'
:[2],'Count':[3]})
df_entity_counts_may.index += 1
df_entity_counts_may.to_csv(os.path.join(export, "entities\entities-frequency-5.csv"))

## june
entity_counts_june = entity_counts[june]
df_entity_counts_june = pd.DataFrame(entity_counts_june, columns={'Entity':[1],'Categor
y':[2],'Count':[3]})
df_entity_counts_june.index += 1
df_entity_counts_june.to_csv(os.path.join(export, "entities\entities-frequency-6.csv"))
```

```
## july
entity_counts_july = entity_counts[july]
df_entity_counts_july = pd.DataFrame(entity_counts_july, columns={'Entity':[1],'Categor
y':[2],'Count':[3]})
df_entity_counts_july.index += 1
df_entity_counts_july.to_csv(os.path.join(export, "entities\entities-frequency-7.csv"))

## august
entity_counts_august = entity_counts[august]
df_entity_counts_august = pd.DataFrame(entity_counts_august, columns={'Entity':[1],'Cat
egory':[2],'Count':[3]})
df_entity_counts_august.index += 1
df_entity_counts_august.to_csv(os.path.join(export, "entities\entities-frequency-8.csv"
))

## september
entity_counts_september = entity_counts[september]
df_entity_counts_september = pd.DataFrame(entity_counts_september, columns={'Entity':[1
],'Category':[2],'Count':[3]})
df_entity_counts_september.index += 1
df_entity_counts_september.to_csv(os.path.join(export, "entities\entities-frequency-9.c
sv"))

## october
entity_counts_october = entity_counts[october]
df_entity_counts_october = pd.DataFrame(entity_counts_october, columns={'Entity':[1],'C
ategory':[2],'Count':[3]})
df_entity_counts_october.index += 1
df_entity_counts_october.to_csv(os.path.join(export, "entities\entities-frequency-10.cs
v"))

## november
entity_counts_november = entity_counts[november]
df_entity_counts_november = pd.DataFrame(entity_counts_november, columns={'Entity':[1],
'Category':[2],'Count':[3]})
df_entity_counts_november.index += 1
df_entity_counts_november.to_csv(os.path.join(export, "entities\entities-frequency-11.c
sv"))

## december
entity_counts_december = entity_counts[december]
df_entity_counts_december = pd.DataFrame(entity_counts_december, columns={'Entity':[1],
'Category':[2],'Count':[3]})
df_entity_counts_december.index += 1
df_entity_counts_december.to_csv(os.path.join(export, "entities\entities-frequency-12.c
sv"))
```

Trends of the entities per month

"general" takes the data from the whole corpus. "mean" is the mean of the months.

Important: in case of empty months (so, when analysing less than one year of data) **remember to exclude them from the mean!**

In [12]:

```
# Merging entity and category (for better indexing)
df_entity_counts_january['Entity / Category'] = df_entity_counts_january['Entity'] + '
 // ' + df_entity_counts_january['Category']
df1 = df_entity_counts_january[['Entity / Category', 'Count']]

df_entity_counts_february['Entity / Category'] = df_entity_counts_february['Entity'] +
' // ' + df_entity_counts_february['Category']
df2 = df_entity_counts_february[['Entity / Category', 'Count']]

df_entity_counts_march['Entity / Category'] = df_entity_counts_march['Entity'] + ' // '
+ df_entity_counts_march['Category']
df3 = df_entity_counts_march[['Entity / Category', 'Count']]

df_entity_counts_april['Entity / Category'] = df_entity_counts_april['Entity'] + ' // '
+ df_entity_counts_april['Category']
df4 = df_entity_counts_april[['Entity / Category', 'Count']]

df_entity_counts_may['Entity / Category'] = df_entity_counts_may['Entity'] + ' // ' + d
f_entity_counts_may['Category']
df5 = df_entity_counts_may[['Entity / Category', 'Count']]

df_entity_counts_june['Entity / Category'] = df_entity_counts_june['Entity'] + ' // ' +
df_entity_counts_june['Category']
df6 = df_entity_counts_june[['Entity / Category', 'Count']]

df_entity_counts_july['Entity / Category'] = df_entity_counts_july['Entity'] + ' // ' +
df_entity_counts_july['Category']
df7 = df_entity_counts_july[['Entity / Category', 'Count']]

df_entity_counts_august['Entity / Category'] = df_entity_counts_august['Entity'] + ' //
' + df_entity_counts_august['Category']
df8 = df_entity_counts_august[['Entity / Category', 'Count']]

df_entity_counts_september['Entity / Category'] = df_entity_counts_september['Entity']
+ ' // ' + df_entity_counts_september['Category']
df9 = df_entity_counts_september[['Entity / Category', 'Count']]

df_entity_counts_october['Entity / Category'] = df_entity_counts_october['Entity'] + '
 // ' + df_entity_counts_october['Category']
df10 = df_entity_counts_october[['Entity / Category', 'Count']]

df_entity_counts_november['Entity / Category'] = df_entity_counts_november['Entity'] +
' // ' + df_entity_counts_november['Category']
df11 = df_entity_counts_november[['Entity / Category', 'Count']]

df_entity_counts_december['Entity / Category'] = df_entity_counts_december['Entity'] +
' // ' + df_entity_counts_december['Category']
df12 = df_entity_counts_december[['Entity / Category', 'Count']]


# List of all entity dataframes
df_ent_freq_all = [df1,df2,df3,df4,df5,df6,df7,df8,df9,df10,df11,df12]

# Loop for index and series
L = []
for x in df_ent_freq_all:
    x = x.set_index('Entity / Category')
    L.append(pd.Series(x.values.tolist(), index=x.index))
```

```
# All together
df_ent_freq_all = pd.concat(L, axis=1, keys=('1','2','3','4','5','6','7','8','9','10',
'11','12'))
df_ent_freq_all = df_ent_freq_all.fillna('0')
for month in df_ent_freq_all:
    df_ent_freq_all[month] = df_ent_freq_all[month].str[0]
df_ent_freq_all = df_ent_freq_all.astype('int')

# Calculate the total
entysums = df_ent_freq_all.iloc[:, [0,1,2,3,4,5,6,7,8,9,10,11]].sum(axis=1)
df_ent_freq_all = pd.concat([df_ent_freq_all, entysums], axis = 1)
df_ent_freq_all = df_ent_freq_all.rename(columns={0: "total"})

# Calculate the mean of the months
entymeans = df_ent_freq_all.iloc[:, [0,1,2,3,4,5,6]].mean(axis=1) ## In case of empty m
onths, exclude them from the mean here! Numbers are indices, where 0 is january and 11
 is december
df_ent_freq_all = pd.concat([df_ent_freq_all, entymeans], axis = 1)
df_ent_freq_all = df_ent_freq_all.rename(columns={0: "mean"})
df_ent_freq_all["mean"] = (df_ent_freq_all["mean"].astype('float')).round(2)

# Reorder and reindex
total_col_e = df_ent_freq_all.pop("total")
df_ent_freq_all.insert(0, "total", total_col_e)
df_ent_freq_all.reset_index(level=0, inplace=True)
df_ent_freq_all = df_ent_freq_all.rename(columns={"index": "enticat"})
df_ent_freq_all = df_ent_freq_all.sort_values(by=['total'], ascending=False)
df_ent_freq_all.index = pd.RangeIndex(len(df_ent_freq_all.index))
df_ent_freq_all.index += 1
df_ent_freq_all[['entity','category']] = df_ent_freq_all.enticat.str.split(" // ",expan
d=True,)
df_ent_freq_all = df_ent_freq_all[['entity','category','total','1','2','3','4','5','6',
'7','8','9','10','11','12','mean']]

# Export and display
df_ent_freq_all.to_csv(os.path.join(export, "entities\entities-frequency-timeseries.cs
v"))
display(df_ent_freq_all.head(20))
```

```
<ipython-input-12-de8688ba0d63>:46: DeprecationWarning: The default dtype
for empty Series will be 'object' instead of 'float64' in a future versio
n. Specify a dtype explicitly to silence this warning.
  L.append(pd.Series(x.values.tolist(), index=x.index))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	
1	coronavirus	COVID19	5227	205	584	1525	1105	556	841	411	0	0	0	0	0	7
2	China	GPE	1982	277	408	315	455	215	144	168	0	0	0	0	0	2
3	Switzerland	GPE	1965	34	153	519	476	292	255	236	0	0	0	0	0	2
4	Swiss	NORP	1490	38	94	404	369	214	205	166	0	0	0	0	0	2
5	first	ORDINAL	1218	36	174	219	234	177	156	222	0	0	0	0	0	1
6	US	GPE	1103	60	43	140	298	222	141	199	0	0	0	0	0	1
7	2020	DATE	1088	38	48	142	328	148	122	262	0	0	0	0	0	1
8	Italy	GPE	978	16	269	401	165	34	61	32	0	0	0	0	0	1
9	Covid-19	COVID19	973	0	39	111	224	116	295	188	0	0	0	0	0	1
10	one	CARDINAL	904	24	66	156	193	171	111	183	0	0	0	0	0	1
11	2019	DATE	886	15	43	136	281	109	88	214	0	0	0	0	0	1
12	U.S.	GPE	884	64	74	217	218	103	101	107	0	0	0	0	0	1
13	Europe	LOC	860	16	86	249	186	91	62	170	0	0	0	0	0	1
14	Germany	GPE	802	22	56	231	175	98	120	100	0	0	0	0	0	1
15	WHO	ORG	728	15	48	62	277	207	55	64	0	0	0	0	0	1
16	Geneva	GPE	610	14	126	113	113	110	75	59	0	0	0	0	0	
17	UK	GPE	578	27	45	155	121	93	93	44	0	0	0	0	0	
18	two	CARDINAL	561	24	56	117	79	88	115	82	0	0	0	0	0	
19	Thursday	DATE	557	46	86	166	81	46	74	58	0	0	0	0	0	
20	France	GPE	548	17	64	206	121	50	56	34	0	0	0	0	0	

Locations

Remember to change the category according to the linguistic model!

In [13]:

```
df_entity_counts_location = df_ent_freq_all[df_ent_freq_all["category"] == "GPE"]
df_entity_counts_location = df_entity_counts_location.reset_index(drop=True)
df_entity_counts_location.index += 1
df_entity_counts_location.to_csv(os.path.join(export, "entities\entities-frequency-0-general-locations.csv"))
display(df_entity_counts_location.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mea
1	China	GPE	1982	277	408	315	455	215	144	168	0	0	0	0	0	283.1
2	Switzerland	GPE	1965	34	153	519	476	292	255	236	0	0	0	0	0	280.7
3	US	GPE	1103	60	43	140	298	222	141	199	0	0	0	0	0	157.5
4	Italy	GPE	978	16	269	401	165	34	61	32	0	0	0	0	0	139.7
5	U.S.	GPE	884	64	74	217	218	103	101	107	0	0	0	0	0	126.2
6	Germany	GPE	802	22	56	231	175	98	120	100	0	0	0	0	0	114.5
7	Geneva	GPE	610	14	126	113	113	110	75	59	0	0	0	0	0	87.1
8	UK	GPE	578	27	45	155	121	93	93	44	0	0	0	0	0	82.5
9	France	GPE	548	17	64	206	121	50	56	34	0	0	0	0	0	78.2
10	Zurich	GPE	471	5	29	83	102	99	96	57	0	0	0	0	0	67.2
11	the United States	GPE	394	10	17	77	87	83	57	63	0	0	0	0	0	56.2
12	Spain	GPE	393	7	41	131	85	26	64	39	0	0	0	0	0	56.1
13	Iran	GPE	316	0	108	143	19	16	21	9	0	0	0	0	0	45.1
14	Wuhan	GPE	313	89	59	45	67	28	12	13	0	0	0	0	0	44.7
15	U.K.	GPE	306	29	41	55	60	40	45	36	0	0	0	0	0	43.7
16	Austria	GPE	298	0	41	140	28	35	34	20	0	0	0	0	0	42.5
17	India	GPE	272	7	6	29	64	25	106	35	0	0	0	0	0	38.8
18	Beijing	GPE	270	33	35	21	44	44	84	9	0	0	0	0	0	38.5
19	Japan	GPE	269	12	33	77	54	15	16	62	0	0	0	0	0	38.4
20	Australia	GPE	230	8	8	20	39	42	79	34	0	0	0	0	0	32.8

Persons

Remember to change the category according to the linguistic model!

In [14]:

```
df_entity_counts_person = df_ent_freq_all[df_ent_freq_all["category"] == "PERSON"]
df_entity_counts_person = df_entity_counts_person.reset_index(drop=True)
df_entity_counts_person.index += 1
df_entity_counts_person.to_csv(os.path.join(export, "entities\entities-frequency-0-general-persons.csv"))
display(df_entity_counts_person.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	Tedros	PERSON	253	2	38	43	79	38	27	26	0	0	0	0	0	36.14
2	Tedros Adhanom Ghebreyesus	PERSON	219	8	44	31	44	43	24	25	0	0	0	0	0	31.29
3	Donald Trump	PERSON	181	4	6	32	61	38	18	22	0	0	0	0	0	25.86
4	Leonteq	PERSON	161	0	0	0	62	0	19	80	0	0	0	0	0	23.00
5	Mnuchin	PERSON	153	15	0	38	0	0	1	99	0	0	0	0	0	21.86
6	Trump	PERSON	95	17	2	32	19	14	4	7	0	0	0	0	0	13.57
7	Ryan	PERSON	85	0	7	12	8	23	24	11	0	0	0	0	0	12.14
8	Julius Baer	PERSON	85	6	2	19	16	21	9	12	0	0	0	0	0	12.14
9	Boris Johnson	PERSON	68	0	4	13	17	21	11	2	0	0	0	0	0	9.71
10	Zur Rose	PERSON	67	0	0	28	30	0	1	8	0	0	0	0	0	9.57
11	John Miller	PERSON	63	1	2	20	20	9	4	7	0	0	0	0	0	9.00
12	Medacta	PERSON	59	0	0	1	42	1	0	15	0	0	0	0	0	8.43
13	Stephanie Nebehay	PERSON	54	0	10	8	18	3	8	7	0	0	0	0	0	7.71
14	Johnson	PERSON	46	0	1	2	17	23	3	0	0	0	0	0	0	6.57
15	Michael Shields	PERSON	46	0	1	15	15	8	2	5	0	0	0	0	0	6.57
16	Jefferies	PERSON	45	3	3	7	10	8	6	8	0	0	0	0	0	6.43
17	John Revill	PERSON	45	0	3	13	15	5	7	2	0	0	0	0	0	6.43
18	Sika	PERSON	44	0	0	0	34	0	0	10	0	0	0	0	0	6.29
19	Michael Ryan	PERSON	42	2	10	5	5	10	4	6	0	0	0	0	0	6.00
20	Xi Jinping	PERSON	39	4	9	3	10	6	3	4	0	0	0	0	0	5.57

Organizations

Remember to change the category according to the linguistic model!

In [15]:

```
df_entity_counts_organization = df_ent_freq_all[df_ent_freq_all["category"] == "ORG"]
df_entity_counts_organization = df_entity_counts_organization.reset_index(drop=True)
df_entity_counts_organization.index += 1
df_entity_counts_organization.to_csv(os.path.join(export, "entities\entities-frequency-0-general-organizations.csv"))
display(df_entity_counts_organization.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	WHO	ORG	728	15	48	62	277	207	55	64	0	0	0	0	0	104.00
2	Dow Jones Newswires	ORG	544	24	31	103	157	48	72	109	0	0	0	0	0	77.71
3	S&P	ORG	443	1	20	2	119	173	126	2	0	0	0	0	0	63.29
4	Reuters	ORG	442	14	31	107	94	50	105	41	0	0	0	0	0	63.14
5	EU	ORG	403	37	40	82	79	37	45	83	0	0	0	0	0	57.57
6	Roche	ORG	390	7	0	99	84	90	15	95	0	0	0	0	0	55.71
7	Trump	ORG	389	13	5	46	154	78	34	59	0	0	0	0	0	55.57
8	AP	ORG	344	16	54	63	49	34	119	9	0	0	0	0	0	49.14
9	Group	ORG	324	0	0	32	92	43	31	126	0	0	0	0	0	46.29
10	UN	ORG	323	6	19	78	81	49	24	66	0	0	0	0	0	46.14
11	COVID	ORG	263	0	2	42	77	43	47	52	0	0	0	0	0	37.57
12	the World Health Organization	ORG	259	6	49	51	65	35	41	12	0	0	0	0	0	37.00
13	EBITDA	ORG	244	0	11	19	49	32	28	105	0	0	0	0	0	34.86
14	Company	ORG	243	0	0	64	66	53	34	26	0	0	0	0	0	34.71
15	ABB	ORG	237	6	2	63	91	0	8	67	0	0	0	0	0	33.86
16	UBS	ORG	231	4	13	64	73	36	21	20	0	0	0	0	0	33.00
17	Dow Jones & Company,	ORG	194	2	3	40	57	24	26	42	0	0	0	0	0	27.71
18	FDA	ORG	193	0	0	24	52	51	18	48	0	0	0	0	0	27.57
19	the Board of Directors	ORG	172	0	0	24	53	8	79	8	0	0	0	0	0	24.57
20	EBIT	ORG	169	5	7	3	60	8	8	78	0	0	0	0	0	24.14

COVID19

Remember to change the category according to the linguistic model!

In [16]:

```
df_entity_counts_COVID19 = df_ent_freq_all[df_ent_freq_all["category"] == "COVID19"]
df_entity_counts_COVID19 = df_entity_counts_COVID19.reset_index(drop=True)
df_entity_counts_COVID19.index += 1
df_entity_counts_COVID19.to_csv(os.path.join(export, "entities\entities-frequency-0-general-COVID19.csv"))
display(df_entity_counts_COVID19.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	m
1	coronavirus	COVID19	5227	205	584	1525	1105	556	841	411	0	0	0	0	0	74
2	Covid-19	COVID19	973	0	39	111	224	116	295	188	0	0	0	0	0	13
3	SARS	COVID19	204	35	17	18	67	32	5	30	0	0	0	0	0	2
4	Coronavirus	COVID19	107	14	11	36	22	4	14	6	0	0	0	0	0	1
5	COVID	COVID19	49	0	0	11	11	9	12	6	0	0	0	0	0	
6	COVID19	COVID19	19	0	0	7	6	2	2	2	0	0	0	0	0	
7	SARS-CoV2	COVID19	14	0	0	0	0	5	2	7	0	0	0	0	0	
8	covid	COVID19	10	0	0	2	2	2	2	2	0	0	0	0	0	
9	Covid	COVID19	8	0	0	1	0	1	6	0	0	0	0	0	0	
10	covid-19	COVID19	6	0	0	2	0	0	0	4	0	0	0	0	0	
11	corona virus	COVID19	5	1	1	1	2	0	0	0	0	0	0	0	0	
12	COVID 19	COVID19	2	0	0	0	0	2	0	0	0	0	0	0	0	
13	covid19	COVID19	1	0	0	0	0	0	1	0	0	0	0	0	0	
14	Covid 19	COVID19	1	0	0	0	0	0	1	0	0	0	0	0	0	
15	Corona virus	COVID19	1	0	0	1	0	0	0	0	0	0	0	0	0	
16	nCoV	COVID19	1	0	1	0	0	0	0	0	0	0	0	0	0	

COVID19r

Remember to change the category according to the linguistic model!

In [17]:

```
df_entity_counts_COVID19r = df_ent_freq_all[df_ent_freq_all["category"] == "COVID19r"]
df_entity_counts_COVID19r = df_entity_counts_COVID19r.reset_index(drop=True)
df_entity_counts_COVID19r.index += 1
df_entity_counts_COVID19r.to_csv(os.path.join(export, "entities\entities-frequency-0-general-COVID19r.csv"))
display(df_entity_counts_COVID19r.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean

In [18]:

```
end_time = datetime.now()
print('Data elaborated in {}'.format(end_time - start_time))
```

```
Data elaborated in 0:04:49.584166
```