Analysis of Italian 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/IT")
export = Path("C://Users/copam/Desktop/jupyter test/exports_NLP/IT")

# 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('it_core_news_md')
spacy_stopwords = spacy.lang.it.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": "virus di Wuhan"},
            {"label": "COVID19r", "pattern": "Virus di Wuhan"},
            {"label": "COVID19r", "pattern": "virus cinese"},
            {"label": "COVID19r", "pattern": "Virus cinese"}
           ]
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	e	6726	490	1240	2498	1296	500	282	420	0	0	0	0	0	960.86
2	essere	3428	322	690	1188	644	248	144	192	0	0	0	0	0	489.71
3	il	3126	228	592	1118	688	188	140	172	0	0	0	0	0	446.57
4	coronavirus	1818	160	358	730	330	90	60	90	0	0	0	0	0	259.71
5	l’	1672	124	320	640	262	108	84	134	0	0	0	0	0	238.86
6	lo	1460	92	202	436	368	172	82	108	0	0	0	0	0	208.57
7	caso	1434	104	240	704	254	34	52	46	0	0	0	0	0	204.86
8	oms	1048	64	100	164	332	180	138	70	0	0	0	0	0	149.71
9	dell’	942	60	202	314	180	66	58	62	0	0	0	0	0	134.57
10	cina	872	152	218	176	156	78	56	36	0	0	0	0	0	124.57
11	della	802	60	122	160	214	130	50	66	0	0	0	0	0	114.57
12	virus	728	156	120	172	118	54	62	46	0	0	0	0	0	104.00
13	contagiare	664	64	186	282	98	8	14	12	0	0	0	0	0	94.86
14	italia	650	30	254	258	28	16	14	50	0	0	0	0	0	92.86
15	contagio	604	12	94	308	112	12	34	32	0	0	0	0	0	86.29
16	morto	592	26	72	282	136	28	12	36	0	0	0	0	0	84.57
17	epidemia	558	82	132	198	86	12	22	26	0	0	0	0	0	79.71
18	sanitario	542	64	110	150	144	26	30	18	0	0	0	0	0	77.43
19	l'	532	38	76	180	106	42	54	36	0	0	0	0	0	76.00
20	nuovo	522	22	80	244	110	10	32	24	0	0	0	0	0	74.57

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"))
```

```python
## 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	mea
1	coronavirus	COVID19	1412	138	282	574	260	62	42	54	0	0	0	0	0	201.7
2	Oms	ORG	882	50	74	128	288	154	124	64	0	0	0	0	0	126.0
3	Cina	LOC	854	144	212	176	154	76	56	36	0	0	0	0	0	122.0
4	Italia	LOC	598	30	238	234	26	12	14	44	0	0	0	0	0	85.4
5	Paese	LOC	536	12	66	296	98	30	8	26	0	0	0	0	0	76.5
6	Covid-19	MISC	402	0	46	142	108	30	36	40	0	0	0	0	0	57.4
7	Paesi	LOC	302	14	30	116	56	30	36	20	0	0	0	0	0	43.1
8	Coronavirus	MISC	300	16	58	104	60	22	16	24	0	0	0	0	0	42.8
9	Usa	LOC	298	22	22	92	96	28	22	16	0	0	0	0	0	42.5
10	Europa	LOC	298	22	38	126	52	12	32	16	0	0	0	0	0	42.5
11	Germania	LOC	284	8	44	166	40	10	4	12	0	0	0	0	0	40.5
12	Svizzera	LOC	250	2	56	84	54	26	18	10	0	0	0	0	0	35.7
13	Stati Uniti	LOC	246	14	18	82	72	30	8	22	0	0	0	0	0	35.1
14	Pechino	LOC	244	40	54	12	50	36	46	6	0	0	0	0	0	34.8
15	Spagna	LOC	230	0	42	142	40	2	2	2	0	0	0	0	0	32.8
16	Wuhan	LOC	224	110	42	20	20	12	20	0	0	0	0	0	0	32.0
17	Iran	LOC	212	2	76	120	12	2	0	0	0	0	0	0	0	30.2
18	Francia	LOC	192	8	48	100	16	12	4	4	0	0	0	0	0	27.4
19	Corea del Sud	LOC	172	16	58	66	22	4	0	6	0	0	0	0	0	24.5
20	Trump	MISC	170	0	2	22	120	14	12	0	0	0	0	0	0	24.2

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"] == "LOC"]
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-ge
neral-locations.csv"))
display(df_entity_counts_location.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	Cina	LOC	854	144	212	176	154	76	56	36	0	0	0	0	0	122.00
2	Italia	LOC	598	30	238	234	26	12	14	44	0	0	0	0	0	85.43
3	Paese	LOC	536	12	66	296	98	30	8	26	0	0	0	0	0	76.57
4	Paesi	LOC	302	14	30	116	56	30	36	20	0	0	0	0	0	43.14
5	Usa	LOC	298	22	22	92	96	28	22	16	0	0	0	0	0	42.57
6	Europa	LOC	298	22	38	126	52	12	32	16	0	0	0	0	0	42.57
7	Germania	LOC	284	8	44	166	40	10	4	12	0	0	0	0	0	40.57
8	Svizzera	LOC	250	2	56	84	54	26	18	10	0	0	0	0	0	35.71
9	Stati Uniti	LOC	246	14	18	82	72	30	8	22	0	0	0	0	0	35.14
10	Pechino	LOC	244	40	54	12	50	36	46	6	0	0	0	0	0	34.86
11	Spagna	LOC	230	0	42	142	40	2	2	2	0	0	0	0	0	32.86
12	Wuhan	LOC	224	110	42	20	20	12	20	0	0	0	0	0	0	32.00
13	Iran	LOC	212	2	76	120	12	2	0	0	0	0	0	0	0	30.29
14	Francia	LOC	192	8	48	100	16	12	4	4	0	0	0	0	0	27.43
15	Corea del Sud	LOC	172	16	58	66	22	4	0	6	0	0	0	0	0	24.57
16	Ginevra	LOC	138	10	56	22	16	18	4	12	0	0	0	0	0	19.71
17	Regno Unito	LOC	120	6	18	78	12	2	0	4	0	0	0	0	0	17.14
18	Giappone	LOC	120	22	52	12	26	4	0	4	0	0	0	0	0	17.14
19	Stato	LOC	104	2	16	44	30	4	4	4	0	0	0	0	0	14.86
20	Russia	LOC	96	6	2	32	34	2	10	10	0	0	0	0	0	13.71

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"] == "PER"]
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-gene
ral-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 Adhanom Ghebreyesus	PER	100	6	14	24	24	14	10	8	0	0	0	0	0	14.29
2	Donald Trump	PER	86	2	4	24	34	12	4	6	0	0	0	0	0	12.29
3	Ford	PER	44	0	0	42	0	0	0	2	0	0	0	0	0	6.29
4	Xi Jinping	PER	40	6	12	0	4	10	8	0	0	0	0	0	0	5.71
5	«	PER	36	8	8	2	8	4	4	2	0	0	0	0	0	5.14
6	Bolsonaro	PER	36	0	0	14	0	0	0	22	0	0	0	0	0	5.14
7	Masih Alinejad	PER	30	0	0	30	0	0	0	0	0	0	0	0	0	4.29
8	Esteri	PER	28	2	6	8	4	4	4	0	0	0	0	0	0	4.00
9	Tedros	PER	26	4	6	0	14	0	2	0	0	0	0	0	0	3.71
10	Spallanzani	PER	26	14	12	0	0	0	0	0	0	0	0	0	0	3.71
11	Roche	PER	24	8	16	0	0	0	0	0	0	0	0	0	0	3.43
12	Angela Merkel	PER	22	0	0	14	8	0	0	0	0	0	0	0	0	3.14
13	Jair Bolsonaro	PER	22	0	0	10	6	2	2	2	0	0	0	0	0	3.14
14	Boris Johnson	PER	18	0	0	14	2	0	0	2	0	0	0	0	0	2.57
15	Covid	PER	18	0	0	0	10	6	2	0	0	0	0	0	0	2.57
16	Shinzo Abe	PER	16	6	2	2	6	0	0	0	0	0	0	0	0	2.29
17	Alimonti	PER	16	0	0	0	16	0	0	0	0	0	0	0	0	2.29
18	Xi	PER	16	4	4	0	2	6	0	0	0	0	0	0	0	2.29
19	Racine	PER	16	0	0	16	0	0	0	0	0	0	0	0	0	2.29
20	Tedros Adhanom	PER	16	2	4	0	4	4	2	0	0	0	0	0	0	2.29

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	mea
1	Oms	ORG	882	50	74	128	288	154	124	64	0	0	0	0	0	126.0
2	Organizzazione mondiale della sanità	ORG	86	12	16	6	12	18	8	14	0	0	0	0	0	12.2
3	Ue	ORG	66	2	6	52	2	4	0	0	0	0	0	0	0	9.4
4	Onu	ORG	60	0	2	4	30	10	12	2	0	0	0	0	0	8.5
5	Organizzazione mondiale della Sanità	ORG	56	2	6	20	12	6	4	6	0	0	0	0	0	8.0
6	Sanità	ORG	48	0	18	18	4	0	0	8	0	0	0	0	0	6.8
7	ANSA	ORG	44	6	6	4	10	10	4	4	0	0	0	0	0	6.2
8	Organizzazione Mondiale della Sanità	ORG	42	6	6	12	16	2	0	0	0	0	0	0	0	6.0
9	Organizzazione	ORG	38	0	0	2	30	4	2	0	0	0	0	0	0	5.4
10	Johns Hopkins University	ORG	32	0	2	12	10	6	2	0	0	0	0	0	0	4.5
11	Fca	ORG	26	0	0	26	0	0	0	0	0	0	0	0	0	3.7
12	British Airways	ORG	26	4	18	2	0	0	0	2	0	0	0	0	0	3.7
13	Nazioni Unite	ORG	24	0	0	2	16	2	2	2	0	0	0	0	0	3.4
14	Parlamento	ORG	22	0	6	10	4	2	0	0	0	0	0	0	0	3.1
15	Renault	ORG	20	0	2	16	0	0	0	2	0	0	0	0	0	2.8
16	Brasile	ORG	20	0	0	2	0	2	2	14	0	0	0	0	0	2.8
17	Moody's	ORG	18	0	4	8	0	0	0	6	0	0	0	0	0	2.5
18	Bmw	ORG	18	0	2	16	0	0	0	0	0	0	0	0	0	2.5
19	Volkswagen	ORG	18	0	0	18	0	0	0	0	0	0	0	0	0	2.5
20	Commissione europea	ORG	18	0	0	16	2	0	0	0	0	0	0	0	0	2.5

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-gen
eral-COVID19.csv"))
display(df_entity_counts_COVID19.head(20))
```

	entity	category	total	1	2	3	4	5	6	7	8	9	10	11	12	mean
1	coronavirus	COVID19	1412	138	282	574	260	62	42	54	0	0	0	0	0	201.71
2	Coronavirus	COVID19	40	4	14	8	2	4	2	6	0	0	0	0	0	5.71
3	2019-nCoV	COVID19	16	12	4	0	0	0	0	0	0	0	0	0	0	2.29
4	Covid	COVID19	16	0	0	4	6	2	2	2	0	0	0	0	0	2.29
5	covid-19	COVID19	4	0	0	2	2	0	0	0	0	0	0	0	0	0.57
6	Covid-19	COVID19	4	0	0	0	0	2	2	0	0	0	0	0	0	0.57
7	COVID19	COVID19	2	0	0	2	0	0	0	0	0	0	0	0	0	0.29
8	covid	COVID19	2	0	0	0	0	0	0	2	0	0	0	0	0	0.29
9	SARS	COVID19	2	0	0	0	0	2	0	0	0	0	0	0	0	0.29

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-ge
neral-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
1	virus cinese	COVID19r	26	22	2	2	0	0	0	0	0	0	0	0	0	3.71
2	Virus cinese	COVID19r	4	4	0	0	0	0	0	0	0	0	0	0	0	0.57

In [18]:

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

```
Data elaborated in 0:00:55.014591
```