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

# 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('de_core_news_md')
spacy_stopwords = spacy.lang.de.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": "China-virus"},
            {"label": "COVID19r", "pattern": "China-Virus"},
            {"label": "COVID19r", "pattern": "chinesischer virus"},
            {"label": "COVID19r", "pattern": "Chinesischer 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	prozent	10196	175	724	1572	2164	3112	730	1719	0	0	0	0	0	1456.57
2	million	7784	28	126	759	1420	3535	511	1405	0	0	0	0	0	1112.00
3	coronavirus	7064	323	1000	2530	1309	936	470	496	0	0	0	0	0	1009.14
4	schweiz	7041	219	383	2178	1328	1204	868	861	0	0	0	0	0	1005.86
5	euro	5113	15	138	572	725	2803	119	741	0	0	0	0	0	730.43
6	unternehmen	4092	33	175	914	856	1192	319	603	0	0	0	0	0	584.57
7	milliarde	4066	20	127	630	842	1241	263	943	0	0	0	0	0	580.86
8	mensch	3959	126	261	1227	964	600	357	424	0	0	0	0	0	565.57
9	schweizer	3911	87	289	1018	769	621	442	685	0	0	0	0	0	558.71
10	land	3494	65	305	933	694	685	449	363	0	0	0	0	0	499.14
11	woche	3486	54	180	1125	904	562	320	341	0	0	0	0	0	498.00
12	zahl	3451	69	307	950	854	582	284	405	0	0	0	0	0	493.00
13	fall	3396	163	473	1156	574	390	297	343	0	0	0	0	0	485.14
14	virus	3333	157	570	1145	589	430	177	265	0	0	0	0	0	476.14
15	person	3258	91	213	999	613	481	342	519	0	0	0	0	0	465.43
16	kanton	3234	27	189	1073	657	445	315	528	0	0	0	0	0	462.00
17	quartal	3193	7	46	122	680	1698	99	541	0	0	0	0	0	456.14
18	|	3110	2	11	354	828	1194	420	301	0	0	0	0	0	444.29
19	2020	2994	21	101	504	724	690	384	570	0	0	0	0	0	427.71
20	stark	2988	32	196	718	648	772	270	352	0	0	0	0	0	426.86

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	m
1	Coronavirus	MISC	4855	219	741	1842	842	626	273	312	0	0	0	0	0	69
2	Schweiz	LOC	4442	166	262	1352	825	736	559	542	0	0	0	0	0	63
3	Schweizer	MISC	3193	64	245	825	641	494	379	545	0	0	0	0	0	45
4	Virus	MISC	2832	138	492	977	493	337	163	232	0	0	0	0	0	40
5	Deutschland	LOC	2038	46	147	674	373	439	207	152	0	0	0	0	0	29
6	China	LOC	1985	220	522	459	306	265	81	132	0	0	0	0	0	28
7	Corona-Krise	MISC	1696	4	5	400	436	463	148	240	0	0	0	0	0	24
8	Zürich	LOC	1562	35	67	233	359	414	115	339	0	0	0	0	0	22
9	Italien	LOC	1419	8	241	638	260	128	97	47	0	0	0	0	0	20
10	Coronavirus	COVID19	1394	86	218	444	295	174	103	74	0	0	0	0	0	19
11	BAG	ORG	1383	119	109	442	216	166	132	199	0	0	0	0	0	19
12	der Schweiz	LOC	1247	27	84	427	222	216	122	149	0	0	0	0	0	17
13	Covid-19	MISC	1246	6	52	268	294	300	139	187	0	0	0	0	0	17
14	Bern	LOC	1234	28	53	239	205	294	227	188	0	0	0	0	0	17
15	Europa	LOC	1115	18	123	291	215	262	93	113	0	0	0	0	0	15
16	Frankreich	LOC	870	13	40	276	174	209	88	70	0	0	0	0	0	12
17	Quarantäne	MISC	830	26	111	251	79	97	73	193	0	0	0	0	0	11
18	USA	LOC	813	16	74	136	176	200	76	135	0	0	0	0	0	11
19	Österreich	LOC	720	4	40	259	95	173	80	69	0	0	0	0	0	10
20	Corona-Krise	LOC	704	4	5	139	220	179	59	98	0	0	0	0	0	10

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-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	m
1	Schweiz	LOC	4442	166	262	1352	825	736	559	542	0	0	0	0	0	634
2	Deutschland	LOC	2038	46	147	674	373	439	207	152	0	0	0	0	0	29
3	China	LOC	1985	220	522	459	306	265	81	132	0	0	0	0	0	283
4	Zürich	LOC	1562	35	67	233	359	414	115	339	0	0	0	0	0	223
5	Italien	LOC	1419	8	241	638	260	128	97	47	0	0	0	0	0	202
6	der Schweiz	LOC	1247	27	84	427	222	216	122	149	0	0	0	0	0	178
7	Bern	LOC	1234	28	53	239	205	294	227	188	0	0	0	0	0	176
8	Europa	LOC	1115	18	123	291	215	262	93	113	0	0	0	0	0	159
9	Frankreich	LOC	870	13	40	276	174	209	88	70	0	0	0	0	0	124
10	USA	LOC	813	16	74	136	176	200	76	135	0	0	0	0	0	116
11	Österreich	LOC	720	4	40	259	95	173	80	69	0	0	0	0	0	102
12	Corona-Krise	LOC	704	4	5	139	220	179	59	98	0	0	0	0	0	100
13	Tessin	LOC	681	1	60	284	139	109	23	65	0	0	0	0	0	97
14	Corona-Pandemie	LOC	651	0	0	90	185	198	59	119	0	0	0	0	0	93
15	Genf	LOC	640	54	83	186	99	77	64	77	0	0	0	0	0	91
16	Berlin	LOC	571	9	29	143	116	188	21	65	0	0	0	0	0	81
17	Düsseldorf	LOC	564	0	5	19	83	381	2	74	0	0	0	0	0	80
18	Basel	LOC	548	13	20	172	115	96	55	77	0	0	0	0	0	78
19	den USA	LOC	520	9	33	110	110	105	56	97	0	0	0	0	0	74
20	Spanien	LOC	457	8	15	129	150	68	53	34	0	0	0	0	0	65

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-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	Daniel Koch	PER	274	21	18	118	56	48	10	3	0	0	0	0	0	39.14
2	Alain Berset	PER	260	1	28	74	64	43	36	14	0	0	0	0	0	37.14
3	Donald Trump	PER	259	3	29	100	46	39	23	19	0	0	0	0	0	37.00
4	Trump	PER	219	6	18	71	37	38	27	22	0	0	0	0	0	31.29
5	Berset	PER	215	4	16	65	38	50	38	4	0	0	0	0	0	30.71
6	Angela Merkel	PER	152	2	2	85	30	15	7	11	0	0	0	0	0	21.71
7	Jens Spahn	PER	138	3	26	51	42	8	3	5	0	0	0	0	0	19.71
8	Seehofer	PER	124	0	11	88	2	18	5	0	0	0	0	0	0	17.71
9	Emmanuel Macron	PER	118	0	8	59	14	12	11	14	0	0	0	0	0	16.86
10	Simonetta Sommaruga	PER	106	1	2	15	33	28	6	21	0	0	0	0	0	15.14
11	Guy Parmelin	PER	89	0	0	44	19	10	11	5	0	0	0	0	0	12.71
12	Ueli Maurer	PER	87	0	1	25	18	17	20	6	0	0	0	0	0	12.43
13	Horst Seehofer	PER	85	0	11	48	6	12	8	0	0	0	0	0	0	12.14
14	Parmelin	PER	82	0	0	34	9	20	18	1	0	0	0	0	0	11.71
15	Corona	PER	77	0	2	13	22	21	5	14	0	0	0	0	0	11.00
16	Sohn	PER	76	0	3	19	9	26	16	3	0	0	0	0	0	10.86
17	Bachmann	PER	73	0	0	21	30	1	17	4	0	0	0	0	0	10.43
18	Peter Altmaier	PER	70	0	15	20	31	4	0	0	0	0	0	0	0	10.00
19	Ursula von der Leyen	PER	68	1	0	37	14	5	9	2	0	0	0	0	0	9.71
20	Reuters	PER	67	1	11	16	12	4	6	17	0	0	0	0	0	9.57

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
1	BAG	ORG	1383	119	109	442	216	166	132	199	0	0	0
2	Bundesamt für Gesundheit	ORG	602	33	47	211	98	66	64	83	0	0	0
3	EU	ORG	574	15	14	177	107	109	101	51	0	0	0
4	WHO	ORG	543	72	124	105	72	107	33	30	0	0	0
5	Corona-Pandemie	ORG	444	0	0	69	83	161	55	76	0	0	0
6	Coronavirus-Pandemie	ORG	426	0	0	113	150	60	44	59	0	0	0
7	Reuters	ORG	329	3	22	59	54	123	10	58	0	0	0
8	SVP	ORG	313	0	7	88	55	44	74	45	0	0	0
9	CORONAVIRUS	ORG	287	1	1	79	73	59	41	33	0	0	0
10	EU-Kommission	ORG	273	6	11	63	61	70	27	35	0	0	0
11	Corona	ORG	268	2	1	63	65	42	46	49	0	0	0
12	UBS	ORG	266	7	15	91	63	17	14	59	0	0	0
13	http://www.nzz.ch	ORG	236	3	5	52	54	53	40	29	0	0	0
14	Redaktionen Berlin	ORG	229	1	4	13	58	77	1	75	0	0	0
15	Parlament	ORG	228	0	10	34	59	65	41	19	0	0	0
16	Frankfurt	ORG	228	2	4	10	58	77	2	75	0	0	0
17	Nationalrat	ORG	226	0	1	22	20	77	99	7	0	0	0
18	Weltgesundheitsorganisation	ORG	218	25	45	59	27	34	16	12	0	0	0
19	Roche	ORG	217	3	3	50	50	51	16	44	0	0	0
20	NZZ	ORG	209	1	3	37	41	51	35	41	0	0	0

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	mean
1	Coronavirus	COVID19	1394	86	218	444	295	174	103	74	0	0	0	0	0	199.14
2	Covid	COVID19	92	0	0	17	18	21	12	24	0	0	0	0	0	13.14
3	2019-nCoV	COVID19	43	19	22	2	0	0	0	0	0	0	0	0	0	6.14
4	Covid-19	COVID19	22	0	6	9	0	2	1	4	0	0	0	0	0	3.14
5	coronavirus	COVID19	9	2	1	2	2	2	0	0	0	0	0	0	0	1.29
6	SARS	COVID19	9	3	1	4	1	0	0	0	0	0	0	0	0	1.29
7	covid	COVID19	2	0	0	0	0	0	2	0	0	0	0	0	0	0.29
8	covid19	COVID19	2	0	0	1	1	0	0	0	0	0	0	0	0	0.29
9	Covid19	COVID19	1	0	0	0	0	1	0	0	0	0	0	0	0	0.14
10	COVID19	COVID19	1	0	0	0	0	1	0	0	0	0	0	0	0	0.14

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 0:08:49.866244
```