Copy_of_Assignment3_TDT4117_Google_Ngram

October 19, 2023

1 Installation

[ ]:
```
pip install google-pygram nltk gensim
```

```
Requirement already satisfied: google-pygram in ./lib/python3.8/site-packages
(0.0.1)
Requirement already satisfied: nltk in ./lib/python3.8/site-packages (3.8.1)
Requirement already satisfied: gensim in ./lib/python3.8/site-packages (4.3.2)
Requirement already satisfied: pandas in ./lib/python3.8/site-packages (from
google-pygram) (2.0.3)
Requirement already satisfied: requests in ./lib/python3.8/site-packages (from
google-pygram) (2.31.0)
Requirement already satisfied: joblib in ./lib/python3.8/site-packages (from
nltk) (1.3.2)
Requirement already satisfied: click in ./lib/python3.8/site-packages (from
nltk) (8.1.7)
Requirement already satisfied: regex>=2021.8.3 in ./lib/python3.8/site-packages
(from nltk) (2023.10.3)
Requirement already satisfied: tqdm in ./lib/python3.8/site-packages (from nltk)
(4.66.1)
Requirement already satisfied: scipy>=1.7.0 in ./lib/python3.8/site-packages
(from gensim) (1.10.1)
Requirement already satisfied: numpy>=1.18.5 in ./lib/python3.8/site-packages
(from gensim) (1.24.4)
Requirement already satisfied: smart-open>=1.8.1 in ./lib/python3.8/site-
packages (from gensim) (6.4.0)
Requirement already satisfied: pytz>=2020.1 in ./lib/python3.8/site-packages
(from pandas->google-pygram) (2023.3.post1)
Requirement already satisfied: python-dateutil>=2.8.2 in ./lib/python3.8/site-
packages (from pandas->google-pygram) (2.8.2)
Requirement already satisfied: tzdata>=2022.1 in ./lib/python3.8/site-packages
(from pandas->google-pygram) (2023.3)
Requirement already satisfied: six>=1.5 in ./lib/python3.8/site-packages (from
python-dateutil>=2.8.2->pandas->google-pygram) (1.16.0)
Requirement already satisfied: certifi>=2017.4.17 in ./lib/python3.8/site-
packages (from requests->google-pygram) (2023.7.22)
Requirement already satisfied: charset-normalizer<4,>=2 in ./lib/python3.8/site-
packages (from requests->google-pygram) (3.3.0)
```

```
Requirement already satisfied: urllib3<3,>=1.21.1 in ./lib/python3.8/site-
packages (from requests->google-pygram) (2.0.6)
Requirement already satisfied: idna<4,>=2.5 in ./lib/python3.8/site-packages
(from requests->google-pygram) (3.4)
WARNING: You are using pip version 21.1.1; however, version 23.3 is
available.
You should consider upgrading via the '/Users/havardnyboe/Documents/emner-
ntnu/TDT4117-InfGjenf/assignment3/bin/python -m pip install --upgrade pip'
command.
Note: you may need to restart the kernel to use updated packages.
```

[ ]:
```
import pandas as pd
import numpy as np
```

tdt4117-assist@idi.ntnu.no

2 Part 1

2.0.1 Visualizing the frequency of terms in google_pygram.

Assign the duration of search

[ ]:
```
search_strat_year = 1800
search_end_year = 2019
```

[ ]:
```
windows_phrases = ["Windows *"]
```

[ ]:
```
from google_pygram import GooglePyGram as gpg

# get the pygram
pygram = gpg(
    corpus='English',
    corpus_year=2019,
    start_year=search_strat_year,
    end_year=search_end_year,
    smoothing=3,
    case_sensitive=False,
    phrases=windows_phrases
)
```

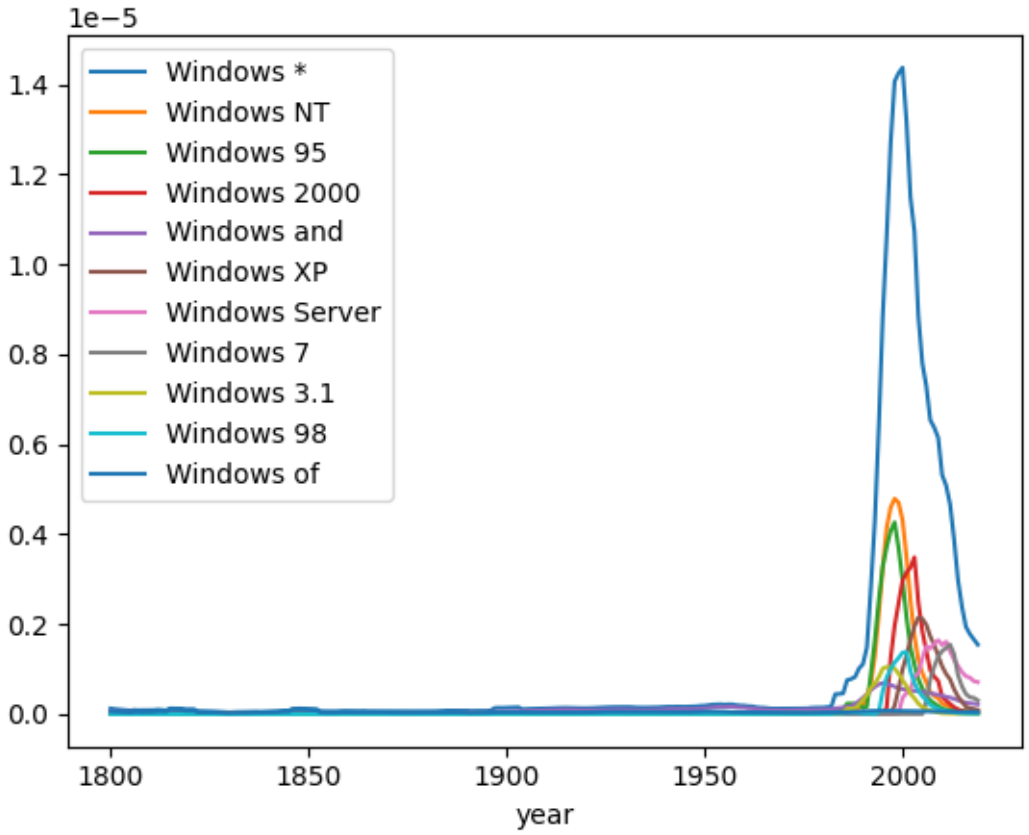
[ ]:
```
windows_ngram = pygram.to_df()
windows_ngram
```

[ ]:
```
        year     Windows *    Windows NT    Windows 95  Windows 2000  \
1800  1800.0  1.256992e-07  0.000000e+00  0.000000e+00  0.000000e+00
1801  1801.0  1.117113e-07  0.000000e+00  0.000000e+00  0.000000e+00
```

```
1802  1802.0  1.032439e-07  0.000000e+00  0.000000e+00  0.000000e+00
1803  1803.0  9.753901e-08  0.000000e+00  0.000000e+00  0.000000e+00
1804  1804.0  8.975180e-08  0.000000e+00  0.000000e+00  0.000000e+00
…      …           …             …             …             …
2015  2015.0  2.391613e-06  8.425371e-08  5.268687e-08  6.372359e-08
2016  2016.0  1.943714e-06  7.160036e-08  4.389111e-08  4.499392e-08
2017  2017.0  1.781299e-06  6.711888e-08  4.083445e-08  3.922459e-08
2018  2018.0  1.660342e-06  6.830725e-08  3.926611e-08  3.700710e-08
2019  2019.0  1.545920e-06  7.010812e-08  3.704839e-08  3.372727e-08

       Windows and    Windows XP  Windows Server     Windows 7   Windows 3.1  \
1800  6.618923e-08  0.000000e+00    0.000000e+00  0.000000e+00  0.000000e+00
1801  5.728825e-08  0.000000e+00    0.000000e+00  0.000000e+00  0.000000e+00
1802  5.064052e-08  0.000000e+00    0.000000e+00  0.000000e+00  0.000000e+00
1803  4.809578e-08  0.000000e+00    0.000000e+00  0.000000e+00  0.000000e+00
1804  4.676061e-08  0.000000e+00    0.000000e+00  0.000000e+00  0.000000e+00
…            …             …             …             …             …
2015  2.756791e-07  2.770391e-07    9.373562e-07  6.193418e-07  1.296385e-08
2016  2.545997e-07  1.641641e-07    8.442474e-07  4.473448e-07  9.541852e-09
2017  2.426956e-07  1.319128e-07    8.120625e-07  3.766601e-07  8.917743e-09
2018  2.338543e-07  1.151505e-07    7.358476e-07  3.607390e-07  8.515957e-09
2019  2.214090e-07  9.660044e-08    7.120567e-07  3.060421e-07  8.255860e-09

        Windows 98    Windows of
1800  0.000000e+00  5.951002e-08
1801  0.000000e+00  5.442309e-08
1802  0.000000e+00  5.260335e-08
1803  0.000000e+00  4.944323e-08
1804  0.000000e+00  4.299119e-08
…            …             …
2015  2.113779e-08  4.743138e-08
2016  1.581421e-08  4.751661e-08
2017  1.432118e-08  4.755145e-08
2018  1.316662e-08  4.848708e-08
2019  1.203536e-08  4.863677e-08

[220 rows x 12 columns]
```

[ ]:
```
# just plot
windows_ngram.plot(x='year')
```

[ ]:
```
<Axes: xlabel='year'>
```

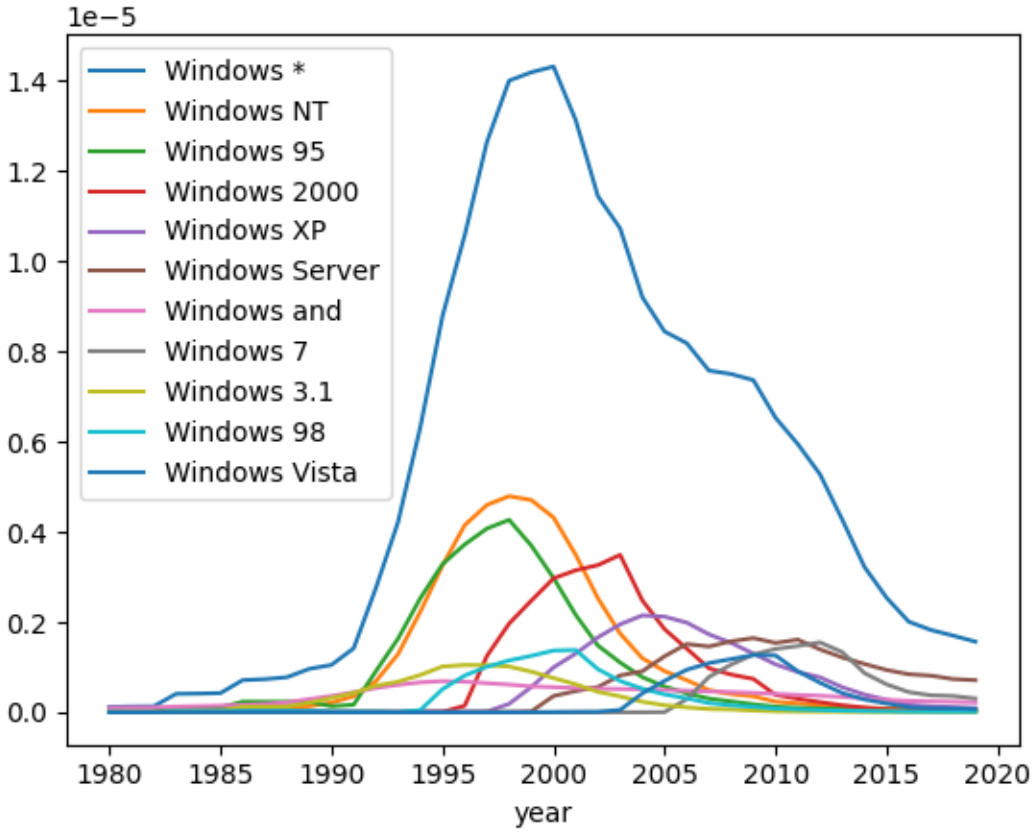


Now we get the time periods between 1980 to 2019

[ ]:
```
search_strat_year = 1980
search_end_year = 2019

# get the pygram
pygram = gpg(
    corpus='English',
    corpus_year=2019,
    start_year=search_strat_year,
    end_year=search_end_year,
    smoothing=3,
    case_sensitive=False,
    phrases=windows_phrases
)

# convert to dataframe
windows_ngram = pygram.to_df()
```

[ ]:
```
windows_ngram.plot(x='year')
```

[ ]: <Axes: xlabel='year'>

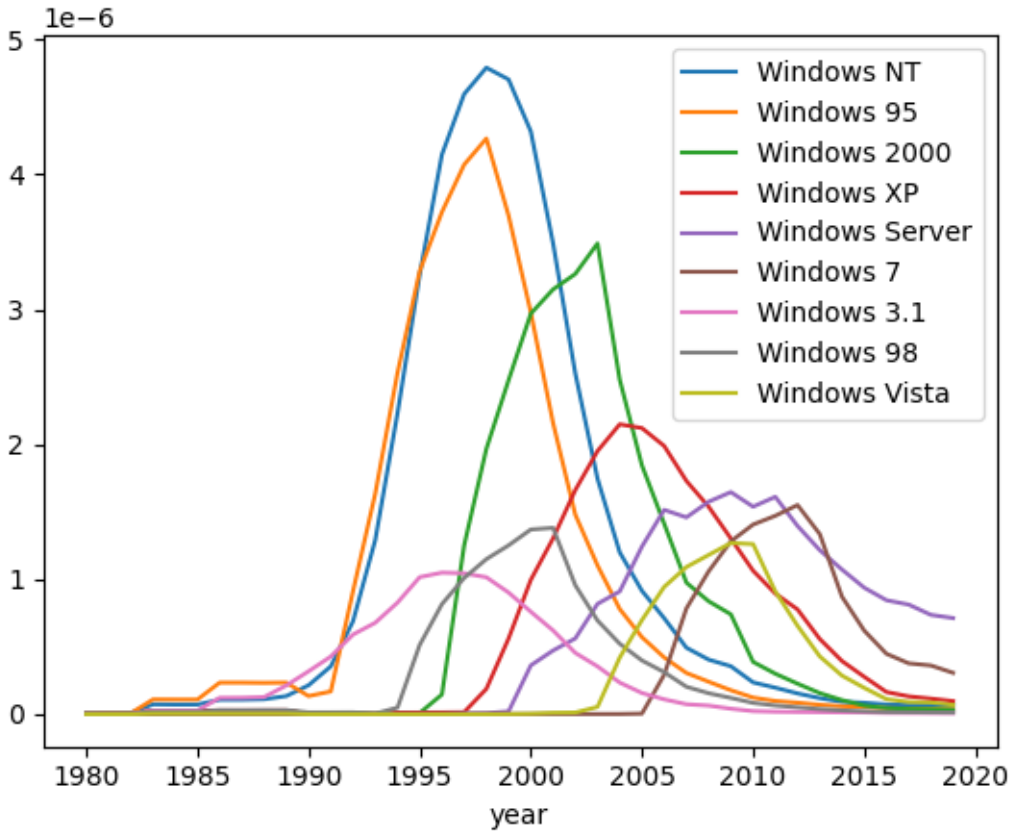


Pre-process the dataframe

[ ]:
```
# we drop the iPhone * and iPhone and to pre process the dataframe
windows_ngram = windows_ngram.drop(
    columns = ['Windows *', 'Windows and'])
```

[ ]:
```
windows_ngram.plot(x="year")
```

[ ]: <Axes: xlabel='year'>



3 Part 2

Visualizing the results to see the relevance

[ ]:
```
search_strat_year = 1990
search_end_year = 2019
```

Assign list the phrases to search

[ ]:
```
windows_phrases = ["Windows *"]
```

Now we get the frequency of the terms from GooglePygram. Then convert it to a dataframe.

[ ]:
```
from google_pygram import GooglePyGram as gpg

# get the pygram
pygram = gpg(
    corpus='English',
    corpus_year=2019,
    start_year=search_strat_year,
    end_year=search_end_year,
```

```
    smoothing=3,
    case_sensitive=False,
    phrases=windows_phrases
)
```

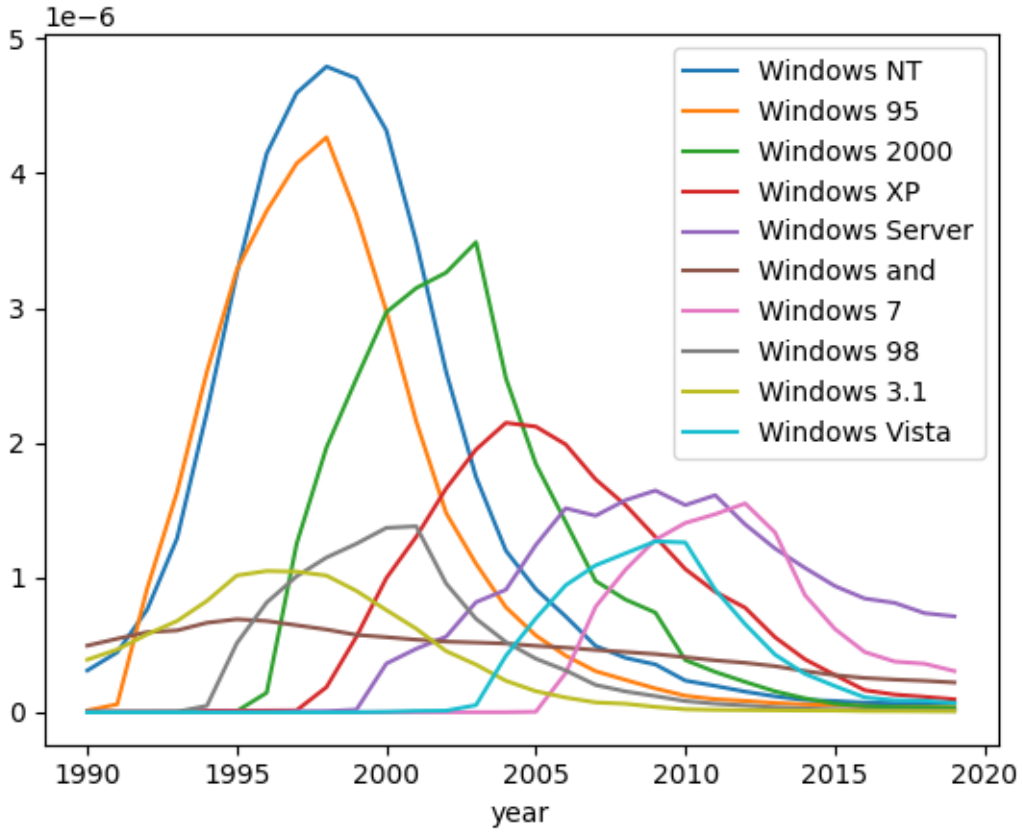
convert to the dataframe and pre process

[ ]:
```
windows_ngram = pygram.to_df()
```

[ ]:
```
windows_ngram = windows_ngram.drop(columns=[ 'Windows *'])
```

[ ]:
```
windows_ngram.plot(x="year")
```

[ ]:
```
<Axes: xlabel='year'>
```



4 Part 3

Computational approach:

We calculate the dissimilarities between Windows Vista and Windows NT

[ ]:
```
dissimilarity = np.log(windows_ngram['Windows Vista'] / windows_ngram['Windows␣
↪NT']) * windows_ngram['Windows Vista']
```

[ ]:
```
pd.DataFrame(dissimilarity)
```

[ ]:

	0
1990	-1.270044e-09
1991	-1.094934e-09
1992	-1.211158e-09
1993	-1.419752e-09
1994	-1.505830e-09
1995	-1.428798e-09
1996	-1.279464e-09
1997	-1.311528e-09
1998	-1.316729e-09
1999	-7.121829e-09
2000	-2.546276e-08
2001	-5.749398e-08
2002	-6.370110e-08
2003	-1.865361e-07
2004	-4.405314e-07
2005	-1.886365e-07
2006	2.664840e-07
2007	8.670628e-07
2008	1.263923e-06
2009	1.613737e-06
2010	2.114311e-06
2011	1.396137e-06
2012	9.483583e-07
2013	5.498358e-07
2014	3.246475e-07
2015	1.636135e-07
2016	4.871092e-08
2017	2.643035e-08
2018	2.069381e-08
2019	-1.153004e-09

Let us visualize the dataframe.

5 Part 4

5.1 Now use the computation method to calculate the time period of dissimilarities between two terms

5.1.1 French presidents:

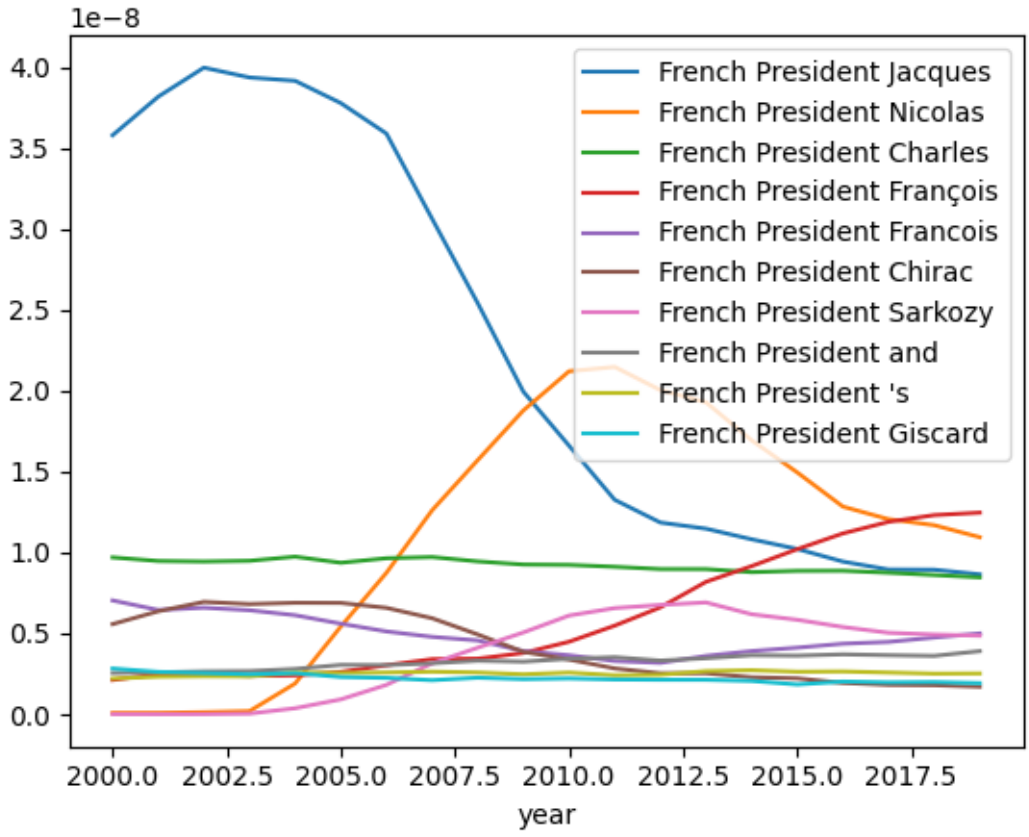
Tip: use start year of 2000 and end year of 2019

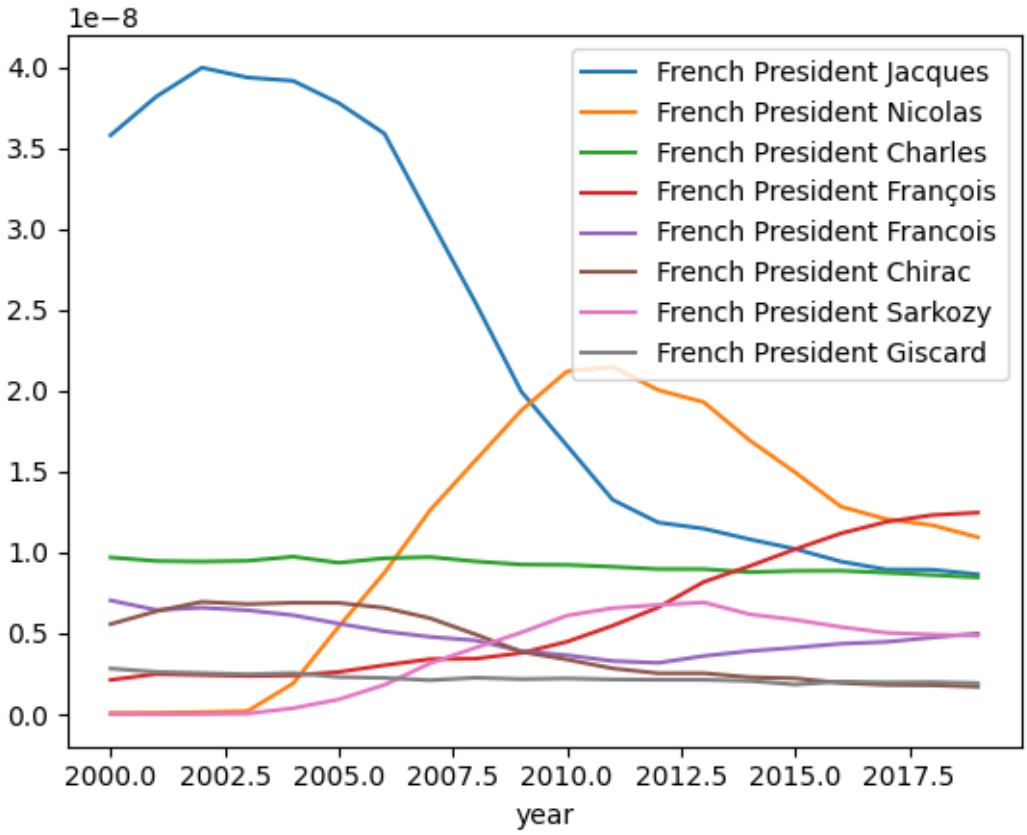
[ ]:
```
french_presidents_phrases = ["French President *"]

from google_pygram import GooglePyGram as gpg

# get the pygram
pygram = gpg(
    corpus='English',
    corpus_year=2019,
    start_year=2000,
    end_year=2019,
    smoothing=3,
    case_sensitive=False,
    phrases=french_presidents_phrases
)

french_ngram = pygram.to_df()
french_ngram = french_ngram.drop(columns=french_presidents_phrases)
french_ngram.plot(x="year")

french_ngram = french_ngram.drop(columns=["French President and","French␣
 ↪President 's"])
french_ngram.plot(x="year")

dissimilarity = np.log(french_ngram['French President Charles'] /␣
 ↪french_ngram['French President Nicolas']) * french_ngram['French President␣
 ↪Charles']
pd.DataFrame(dissimilarity)
```

[ ]:

	0
2000	4.596921e-08
2001	4.422452e-08
2002	4.042566e-08
2003	3.687774e-08
2004	1.589770e-08
2005	5.118903e-09
2006	9.438274e-10
2007	-2.528154e-09
2008	-4.801993e-09
2009	-6.548169e-09
2010	-7.673176e-09
2011	-7.808163e-09
2012	-7.209068e-09
2013	-6.864631e-09
2014	-5.765962e-09
2015	-4.632020e-09
2016	-3.277577e-09
2017	-2.805644e-09

```
2018 -2.632054e-09
2019 -2.178134e-09
```





5.2 German Chancellors:

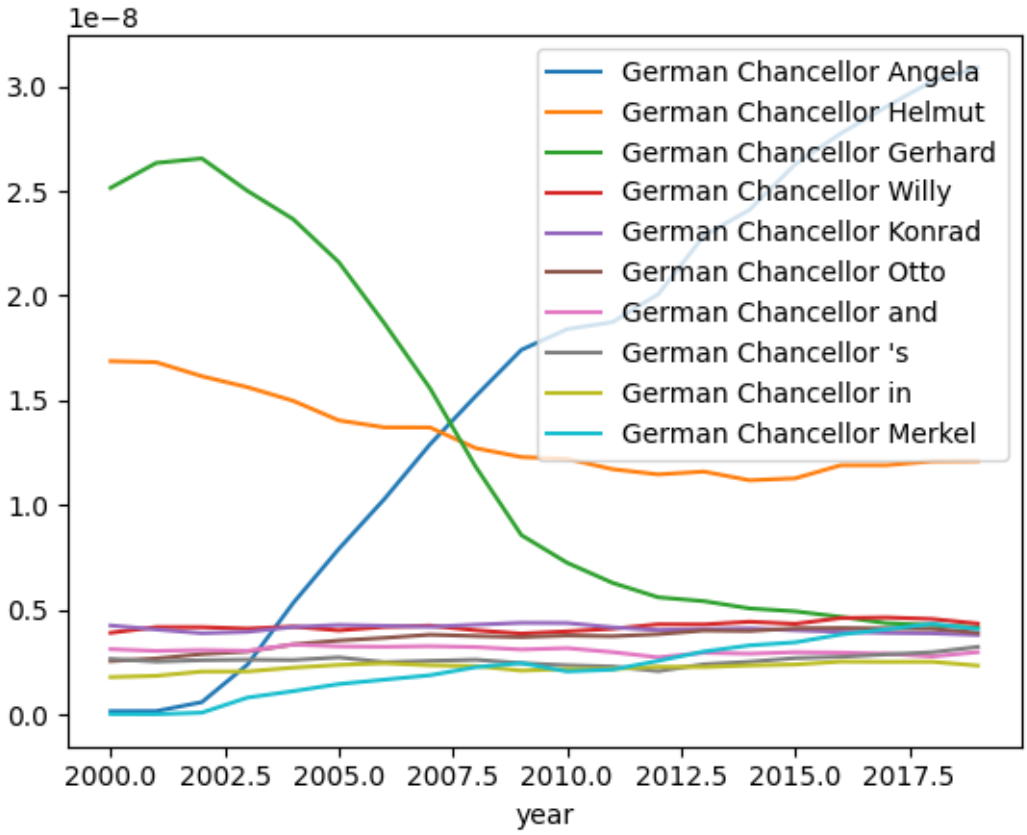
Tip: use start year of 2000 and end year of 2019

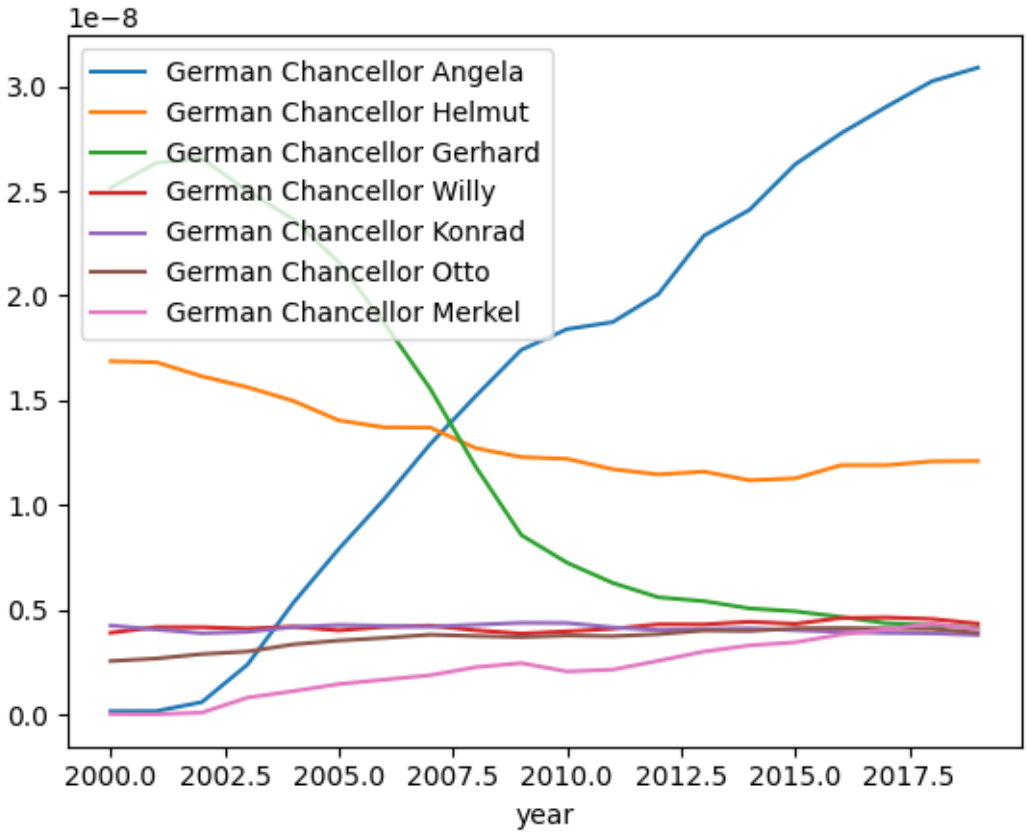
[ ]:
```
german_chancellor_phrases = ["German Chancellor *"]

from google_pygram import GooglePyGram as gpg

# get the pygram
pygram = gpg(
    corpus='English',
    corpus_year=2019,
    start_year=2000,
    end_year=2019,
    smoothing=3,
    case_sensitive=False,
    phrases=german_chancellor_phrases
)

german_ngram = pygram.to_df()
```

```
german_ngram = german_ngram.drop(columns=german_chancellor_phrases)
german_ngram.plot(x="year")

german_ngram = german_ngram.drop(columns=["German Chancellor and","German␣
 ↪Chancellor 's", "German Chancellor in"])
german_ngram.plot(x="year")

dissimilarity = np.log(german_ngram['German Chancellor Angela'] /␣
 ↪german_ngram['German Chancellor Helmut']) * german_ngram['German Chancellor␣
 ↪Angela']
pd.DataFrame(dissimilarity)
```

[ ]:

	0
2000	-7.570565e-10
2001	-7.668936e-10
2002	-1.965667e-09
2003	-4.486465e-09
2004	-5.507159e-09
2005	-4.542599e-09
2006	-2.950321e-09
2007	-7.778080e-10
2008	2.716661e-09
2009	6.059156e-09
2010	7.547481e-09
2011	8.798767e-09
2012	1.123382e-08
2013	1.550836e-08
2014	1.849094e-08
2015	2.221575e-08
2016	2.346860e-08
2017	2.585600e-08
2018	2.771484e-08
2019	2.892096e-08


1e−8
3.0
2.5
2.0
1.5
1.0
0.5
0.0
2000.0
2002.5
2005.0
2007.5
2010.0
2012.5
2015.0
2017.5
year
German Chancellor Angela
German Chancellor Helmut
German Chancellor Gerhard
German Chancellor Willy
German Chancellor Konrad
German Chancellor Otto
German Chancellor and
German Chancellor 's
German Chancellor in
German Chancellor Merkel



5.3 War in:

Tip: use start year of 1940 and end year of 2019

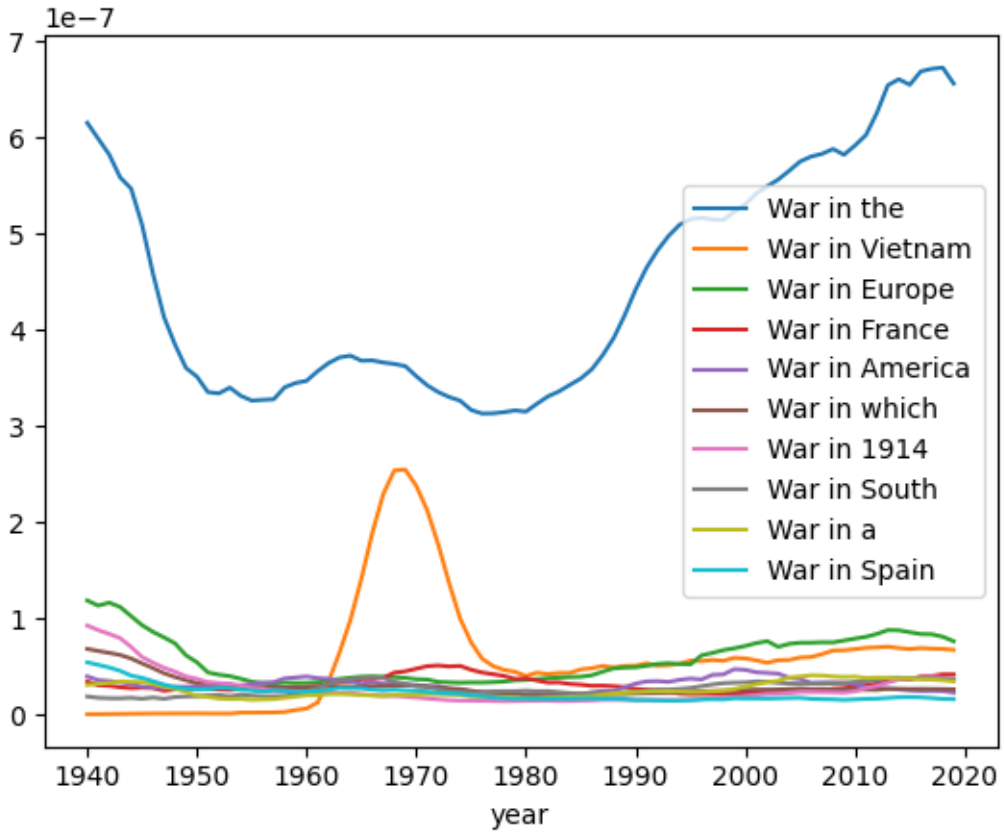
[ ]:
```
war_in_phrases = ["War in *"]

from google_pygram import GooglePyGram as gpg

# get the pygram
pygram = gpg(
    corpus='English',
    corpus_year=2019,
    start_year=1940,
    end_year=2019,
    smoothing=3,
    case_sensitive=False,
    phrases=war_in_phrases
)

war_ngram = pygram.to_df()
```

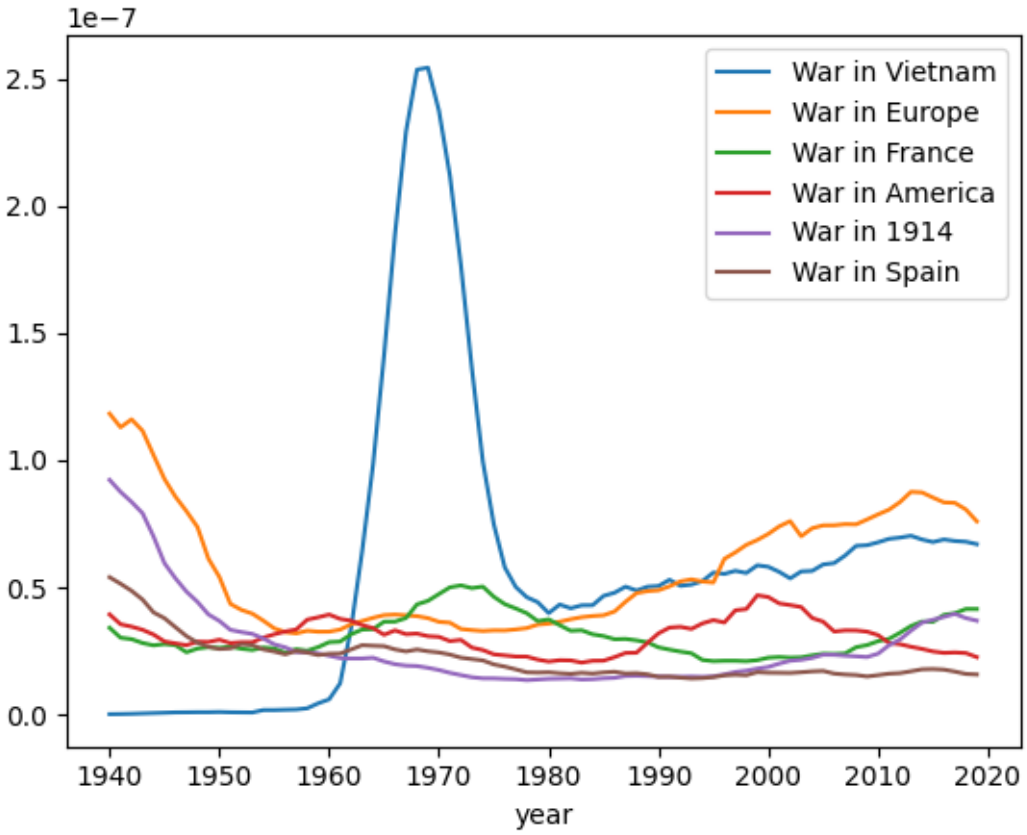
```
war_ngram = war_ngram.drop(columns=war_in_phrases)
war_ngram.plot(x="year")

war_ngram = war_ngram.drop(columns=["War in the","War in a", "War in which",␣
 ↪"War in South"])
war_ngram.plot(x="year")

dissimilarity = np.log(war_ngram['War in France'] / war_ngram['War in Europe'])␣
 ↪* war_ngram['War in France']
pd.DataFrame(dissimilarity)
```

[ ]:

	0
1940	-4.240686e-08
1941	-3.980469e-08
1942	-4.043116e-08
1943	-3.872882e-08
1944	-3.592997e-08
…	…
2015	-3.098925e-08
2016	-2.959656e-08
2017	-2.934489e-08
2018	-2.756165e-08
2019	-2.508949e-08

[80 rows x 1 columns]


1e−7
7
6
5
4
3
2
1
0
1940
1950
1960
1970
1980
1990
2000
2010
2020
year
War in the
War in Vietnam
War in Europe
War in France
War in America
War in which
War in 1914
War in South
War in a
War in Spain


1e−7
2.5
2.0
1.5
1.0
0.5
0.0
1940
1950
1960
1970
1980
1990
2000
2010
2020
year
War in Vietnam
War in Europe
War in France
War in America
War in 1914
War in Spain