Praca domowa 1 - warsztaty badawcze

Autorzy: Jakub Kozieł, Jan Skwarek, Tomasz Nocoń

Wstęp

```
! pip install swifter
! pip install pandas
! pip install textacy
! pip install spacy
```

```
Requirement already satisfied: swifter in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: parso>0.4.0 in /usr/local/lib/python3.7/dist
Requirement already satisfied: pandas>=1.0.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: cloudpickle>=0.2.2 in /usr/local/lib/python3
Requirement already satisfied: ipywidgets>=7.0.0 in /usr/local/lib/python3.
Requirement already satisfied: psutil>=5.6.6 in /usr/local/lib/python3.7/di
Requirement already satisfied: dask[dataframe]>=2.10.0 in /usr/local/lib/py
Requirement already satisfied: tqdm>=4.33.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: bleach>=3.1.1 in /usr/local/lib/python3.7/di
Requirement already satisfied: six>=1.9.0 in /usr/local/lib/python3.7/dist-
Requirement already satisfied: webencodings in /usr/local/lib/python3.7/dis
Requirement already satisfied: packaging in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: partd>=0.3.10 in /usr/local/lib/python3.7/di
Requirement already satisfied: toolz>=0.7.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: numpy>=1.13.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: fsspec>=0.6.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: jupyterlab-widgets>=1.0.0 in /usr/local/lib/
Requirement already satisfied: widgetsnbextension~=3.6.0 in /usr/local/lib/
Requirement already satisfied: nbformat>=4.2.0 in /usr/local/lib/python3.7/
Requirement already satisfied: ipython-genutils~=0.2.0 in /usr/local/lib/py
Requirement already satisfied: ipykernel>=4.5.1 in /usr/local/lib/python3.7
Requirement already satisfied: ipython>=4.0.0 in /usr/local/lib/python3.7/d
Requirement already satisfied: traitlets>=4.3.1 in /usr/local/lib/python3.7
Requirement already satisfied: jupyter-client in /usr/local/lib/python3.7/d
Requirement already satisfied: tornado>=4.0 in /usr/local/lib/python3.7/dis
Requirement already satisfied: prompt-toolkit<2.0.0,>=1.0.4 in /usr/local/l
Requirement already satisfied: pygments in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: simplegeneric>0.8 in /usr/local/lib/python3.
Requirement already satisfied: pexpect in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: pickleshare in /usr/local/lib/python3.7/dist
Requirement already satisfied: setuptools>=18.5 in /usr/local/lib/python3.7
Requirement already satisfied: decorator in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: jsonschema!=2.5.0,>=2.4 in /usr/local/lib/py
Requirement already satisfied: jupyter-core in /usr/local/lib/python3.7/dis
Requirement already satisfied: importlib-metadata in /usr/local/lib/python3
Requirement already satisfied: importlib-resources>=1.4.0 in /usr/local/lib
Requirement already satisfied: attrs>=17.4.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: pyrsistent!=0.17.0,!=0.17.1,!=0.17.2,>=0.14.
```

✓ 0 s ukończono o 00:03

```
Requirement already satisfied: python-dateutil>=2.7.3 in /usr/local/lib/pyt
Requirement already satisfied: pytz>=2017.3 in /usr/local/lib/python3.7/dis
Requirement already satisfied: locket in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: wcwidth in /usr/local/lib/python3.7/dist-pac
Requirement already satisfied: notebook>=4.4.1 in /usr/local/lib/python3.7/
Requirement already satisfied: Send2Trash in /usr/local/lib/python3.7/dist-
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: terminado>=0.8.1 in /usr/local/lib/python3.7
Requirement already satisfied: nbconvert in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pyzmq>=13 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: ptyprocess in /usr/local/lib/python3.7/dist-
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7
Requirement already satisfied: testpath in /usr/local/lib/python3.7/dist-pa
Requirement already satisfied: mistune<2,>=0.8.1 in /usr/local/lib/python3.
Requirement already satisfied: entrypoints>=0.2.2 in /usr/local/lib/python3
Requirement already satisfied: pandocfilters>=1.4.1 in /usr/local/lib/pytho
Requirement already satisfied: defusedxml in /usr/local/lib/python3.7/dist-
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/p
```

```
! python -m spacy download en_core_web_sm
```

```
Collecting en-core-web-sm==3.2.0
  Downloading https://github.com/explosion/spacy-models/releases/download/e
     |████████████████████████████████| 13.9 MB 379 kB/s
Requirement already satisfied: spacy<3.3.0,>=3.2.0 in /usr/local/lib/pythor
Requirement already satisfied: blis<0.8.0,>=0.4.0 in /usr/local/lib/python3
Requirement already satisfied: spacy-legacy<3.1.0,>=3.0.8 in /usr/local/lib
Requirement already satisfied: wasabi<1.1.0,>=0.8.1 in /usr/local/lib/pytho
Requirement already satisfied: pydantic!=1.8,!=1.8.1,<1.9.0,>=1.7.4 in /usr
Requirement already satisfied: typer<0.5.0,>=0.3.0 in /usr/local/lib/pythor
Requirement already satisfied: packaging>=20.0 in /usr/local/lib/python3.7/
Requirement already satisfied: langcodes<4.0.0,>=3.2.0 in /usr/local/lib/py
Requirement already satisfied: setuptools in /usr/local/lib/python3.7/dist-
Requirement already satisfied: preshed<3.1.0,>=3.0.2 in /usr/local/lib/pyth
Requirement already satisfied: srsly<3.0.0,>=2.4.1 in /usr/local/lib/pythor
Requirement already satisfied: numpy>=1.15.0 in /usr/local/lib/python3.7/di
Requirement already satisfied: catalogue<2.1.0,>=2.0.6 in /usr/local/lib/py
Requirement already satisfied: typing-extensions<4.0.0.0,>=3.7.4 in /usr/lo
Requirement already satisfied: pathy>=0.3.5 in /usr/local/lib/python3.7/dis
Requirement already satisfied: requests<3.0.0,>=2.13.0 in /usr/local/lib/py
Requirement already satisfied: jinja2 in /usr/local/lib/python3.7/dist-pack
Requirement already satisfied: murmurhash<1.1.0,>=0.28.0 in /usr/local/lib/
Requirement already satisfied: tqdm<5.0.0,>=4.38.0 in /usr/local/lib/pythor
Requirement already satisfied: spacy-loggers<2.0.0,>=1.0.0 in /usr/local/li
Requirement already satisfied: cymem<2.1.0,>=2.0.2 in /usr/local/lib/pythor
Requirement already satisfied: thinc<8.1.0,>=8.0.12 in /usr/local/lib/pytho
Requirement already satisfied: zipp>=0.5 in /usr/local/lib/python3.7/dist-p
Requirement already satisfied: pyparsing!=3.0.5,>=2.0.2 in /usr/local/lib/p
Requirement already satisfied: smart-open<6.0.0,>=5.0.0 in /usr/local/lib/p
Requirement already satisfied: chardet<4,>=3.0.2 in /usr/local/lib/python3.
Requirement already satisfied: certifi>=2017.4.17 in /usr/local/lib/python3
Requirement already satisfied: idna<3,>=2.5 in /usr/local/lib/python3.7/dis
Requirement already satisfied: urllib3!=1.25.0,!=1.25.1,<1.26,>=1.21.1 in /
Requirement already satisfied: click<9.0.0,>=7.1.1 in /usr/local/lib/pythor
Requirement already satisfied: MarkupSafe>=0.23 in /usr/local/lib/python3.7
✓ Download and installation successful
You can now load the package via spacy.load('en_core_web_sm')
```

```
import spacy
import pandas as pd
from tqdm.auto import tqdm
import swifter
import plotly.express as px
from wordcloud import WordCloud
from matplotlib import pyplot as plt
import textacy
from collections import Counter
import random

pd.options.plotting.backend = "plotly"
random.seed(123)
```

Preprocessing

As previously we work on data from https://dataverse.harvard.edu/dataset.xhtml?persistentId=doi:10.7910/DVN/6MZN76

```
!wget -O data.tar.gz https://dataverse.harvard.edu/api/access/datafile/:persiste
```

```
--2022-03-24 22:52:13--  https://dataverse.harvard.edu/api/access/datafile/
Resolving dataverse.harvard.edu (dataverse.harvard.edu)... 35.168.250.191,
Connecting to dataverse.harvard.edu (dataverse.harvard.edu)|35.168.250.191|
HTTP request sent, awaiting response... 303 See Other
Location: https://dvn-cloud.s3.amazonaws.com/10.7910/DVN/6MZN76/15de5b930dc
--2022-03-24 22:52:14--  https://dvn-cloud.s3.amazonaws.com/10.7910/DVN/6MZ
Resolving dvn-cloud.s3.amazonaws.com (dvn-cloud.s3.amazonaws.com)... 52.217
Connecting to dvn-cloud.s3.amazonaws.com (dvn-cloud.s3.amazonaws.com)|52.21
HTTP request sent, awaiting response... 200 OK
Length: 959382206 (915M) [application/x-gzip]
Saving to: 'data.tar.gz'

data.tar.gz         100%[===================>] 914.94M  48.6MB/s    in 20s

2022-03-24 22:52:34 (45.3 MB/s) - 'data.tar.gz' saved [959382206/959382206]
```

```
!tar -xf data.tar.gz
```

```
en = spacy.load("en_core_web_sm") #so far we keep on execution time
```

```
df = pd.read_table('Dail_debates_1919-2013.tab')
```

```
df.date = pd.to_datetime(df.date)
```

Filtrujemy ramkę danych tak, żeby pokazywała lata 2007-2010. Wtedy na świecie trwał kryzys ekonomiczny. Być może znajdziemy jakieś ciekawe dane pod tym kątem.

```
#we are going to use subsample from 2007 to 2010, being interested in the years
df = df[(df.date.dt.year>=2007)&(df.date.dt.year<=2010)]
```

```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 431261 entries, 3735544 to 4166804
Data columns (total 10 columns):
 #   Column       Non-Null Count   Dtype         
---  ------       --------------   -----         
 0   speechID     431261 non-null  int64         
 1   memberID     431261 non-null  int64         
 2   partyID      431261 non-null  int64         
 3   constID      431261 non-null  int64         
 4   title        431261 non-null  object        
 5   date         431261 non-null  datetime64[ns]
 6   member_name  431261 non-null  object        
 7   party_name   431261 non-null  object        
 8   const_name   431261 non-null  object        
 9   speech       431261 non-null  object        
dtypes: datetime64[ns](1), int64(4), object(5)
memory usage: 36.2+ MB
```

431 tys. wierszy to jednak nadal zbyt dużo. Wybierzemy z tego podzbiór, zachowując podobną ilość obserwacji z każdego roku

```
df["year"] = df.date.dt.year

df = df.groupby('year', group_keys=False).apply(lambda x: x.sample(3500, random_

df.drop("year", axis = 1)
```

	speechID	memberID	partyID	constID	title	date	member_n
3798325	3798425	656	6	39	Markets in Financial Instruments and Miscellan...	2007-10-31	Ms. Kathl Ly
3798478	3798578	1091	14	117	Written Answers - Social Welfare Benefits.	2007-10-31	Mr. Jack \
					Written Answers		Mr. N

3784062	3784162	309	8	148	Answers - Public Transport.	2007-09-26	Mr. I Demp
3764330	3764430	1091	14	117	Written Answers - Abbey Theatre.	2007-03-28	Mr. Jack \
3742209	3742309	253	8	123	Other Questions - Tax Code.	2007-02-07	Mr. B Co\

```
df.columns
```

```
Index(['speechID', 'memberID', 'partyID', 'constID', 'title', 'date',
       'member_name', 'party_name', 'const_name', 'speech', 'year'],
      dtype='object')
```

```
df["speech_en"] = df['speech'].swifter.apply(en)
```

Pandas Apply: 14000/14000 [06:18<00:00,

100%

Czas na powyższą operację <7min

Zależności

```
df.party_name.unique()
```

```
array(['Democratic Left', 'The Labour Party', 'Fianna Fáil', 'Fine Gael',
       'Green Party', 'Sinn Féin', "The Workers' Party",
       'Progressive Democrats', 'Independent', 'Socialist Party',
       "Sinn Féin the Workers' Party"], dtype=object)
```

```
df.nunique()
```

```
speechID       14000
memberID         195
partyID           11
constID           51
title           3055
date             371
member_name      195
party_name        11
const_name        51
speech         13550
year               4
```

```
speech_en      14000
dtype: int64
```

W analizowanych przez nas latach kolicję tworzyły partie FF-Green-PD

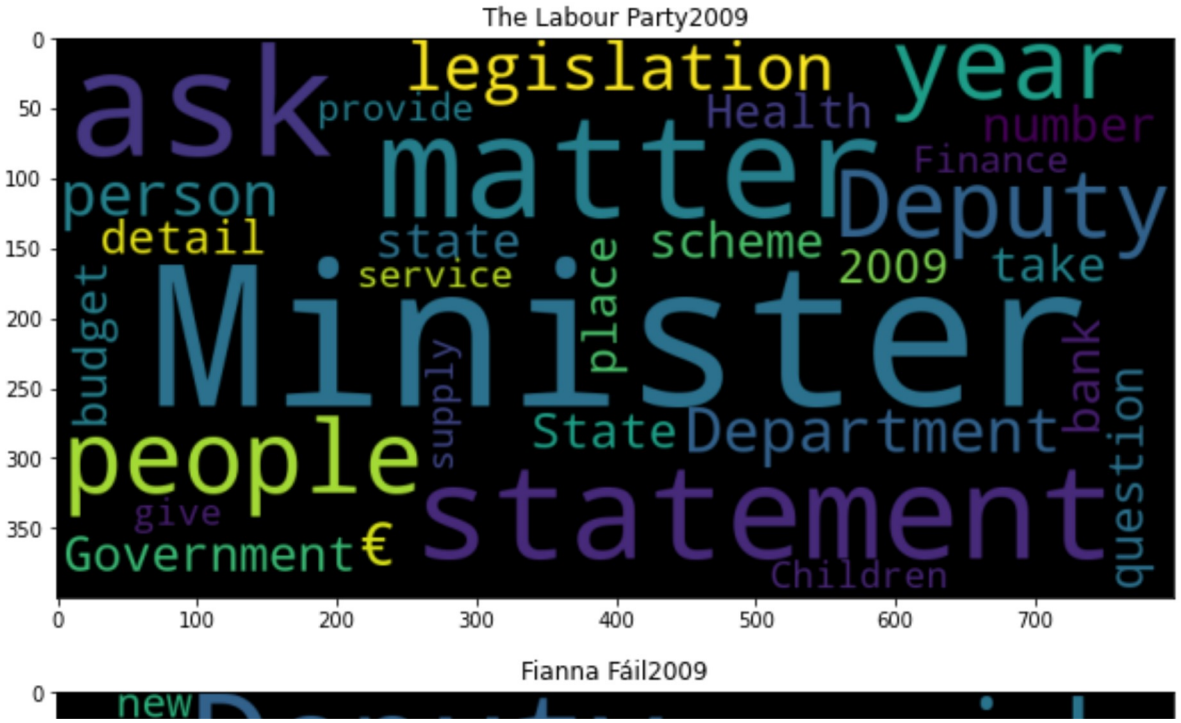
Naszym pomysłem na ten zbiór było zbadanie czy kryzys światowy z roku 2008 miał wpył na temtyke wypowiedzi w rządzie. W tym celu wzieliśmy okres 2007-2010, żeby zobaczyć czy faktycznie kryzys mial w latach 2008-2009 mial wpływ na to co było mówione

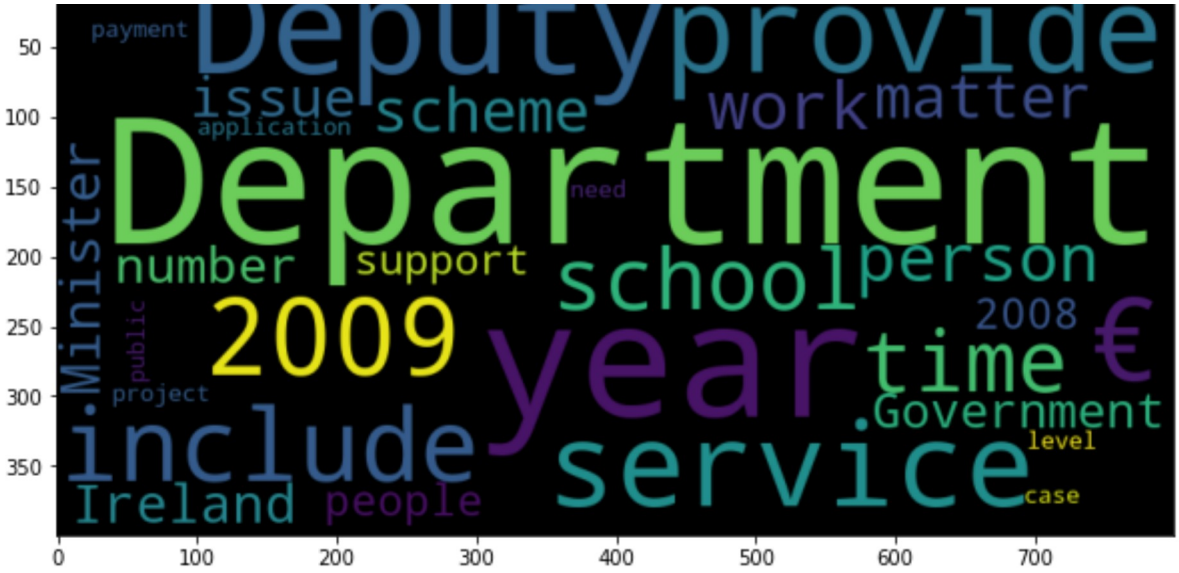
Skupiliśmy sie na charakterze wypowiedzi wewnątrz partii.

```
df.party_name.unique()
```

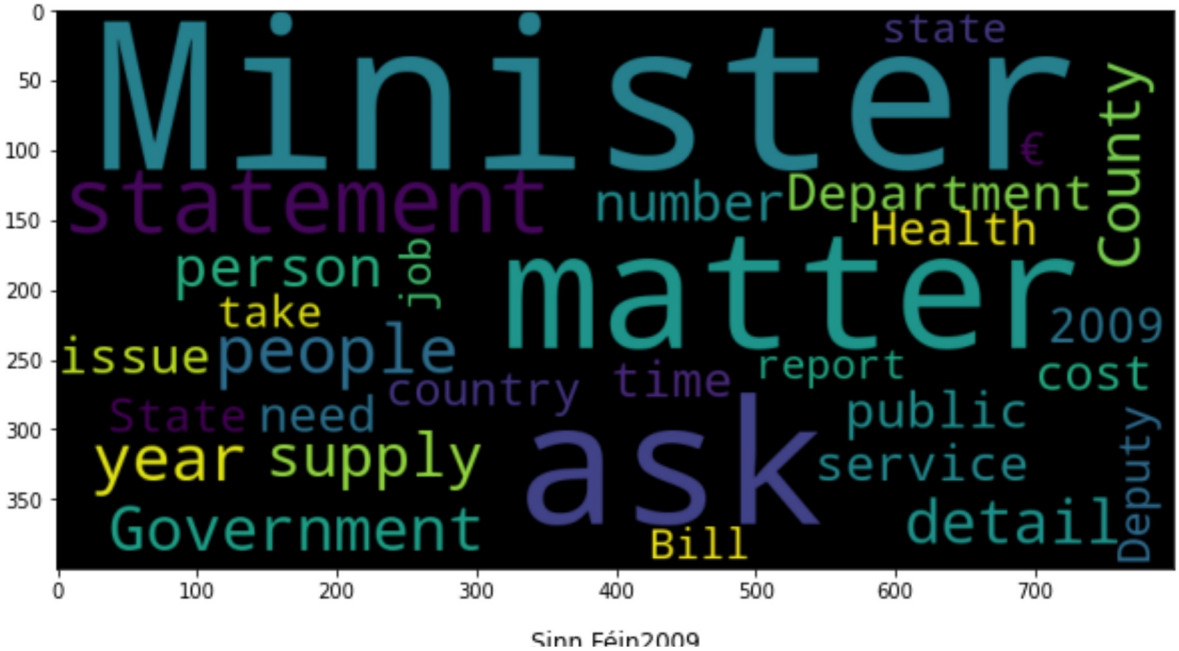
```
array(['Democratic Left', 'The Labour Party', 'Fianna Fáil', 'Fine Gael',
       'Green Party', 'Sinn Féin', "The Workers' Party",
       'Progressive Democrats', 'Independent', 'Socialist Party',
       "Sinn Féin the Workers' Party"], dtype=object)
```

```
year = 2009
df_year = df[df.date.dt.year == year]
parties = df_year['party_name'].unique()
for party in parties:
    temp = df_year[df_year['party_name'] == party]
    lemmas = temp.speech_en.apply(lambda doc: [token.lemma_ for token in doc if
    word_counts = dict(Counter(lemmas.sum()).most_common(30))
    wc = WordCloud(width=800, height=400)
    wc.generate_from_frequencies(frequencies=word_counts)
    plt.figure(figsize=(10,8))
    plt.title(party+str(year))
    plt.imshow(wc)
```





Fine Gael2009

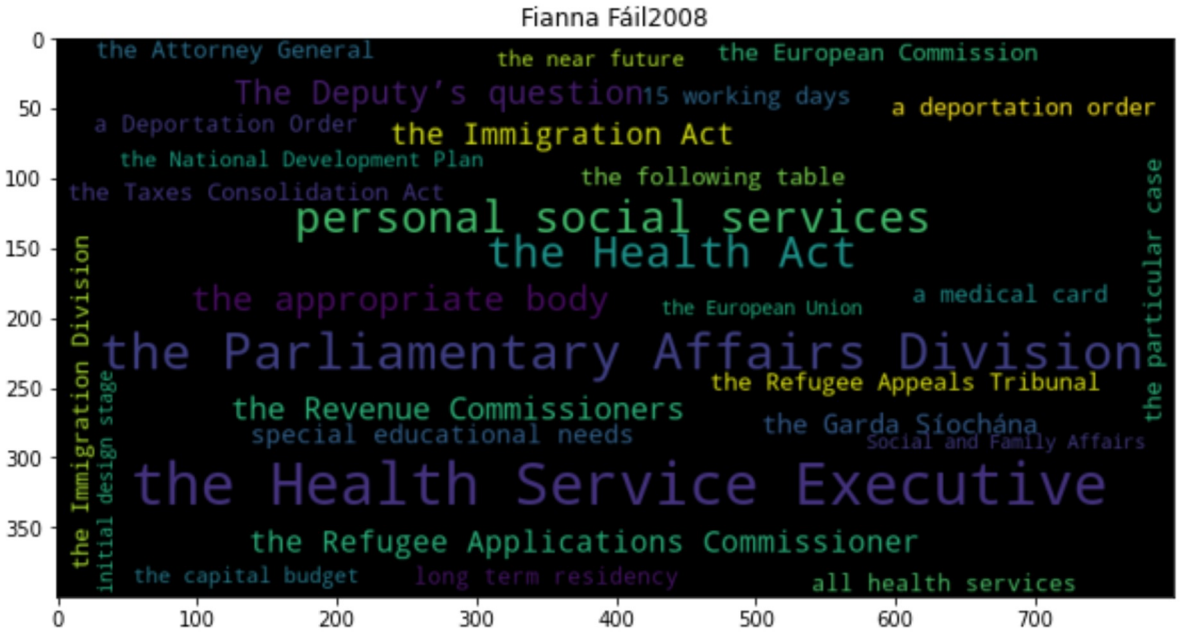


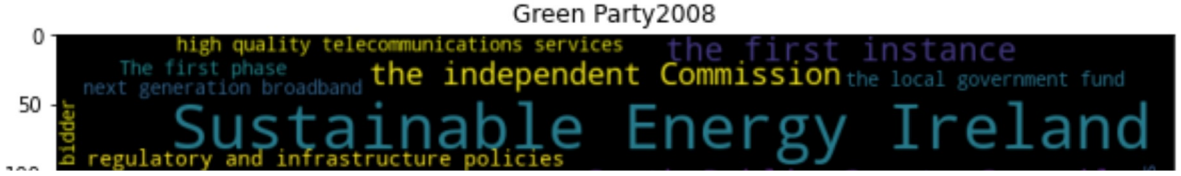
Sinn Féin2009

To co tutaj zrobiliśmy to podzieliliśmy wypowiedzi na osoby z partii. Dodotkowo manipulowalismy sobie latami po to, żeby zobaczyć czy faktycznie okres kryzysu spowoduje jakieś zmiany w kluczowych temtatch. Z przeprowadzonych eksperymentow wynika, że pojawia się wzmianki na ten temat, np lata 2008 czy 2009 sa cześciej pojawiaja się niż te pozstałe, jednak nie ma jakis oczywistych wniosków. To co możemy powiedziec, ze słowo Minister jest jednym z bardziej popularych słów, nie jest jednak to prawda dla partii socjalnych, np: Workers Party gdzie używaja oni irlandziekgo określenia na to stanowisko. To co mozemy powiedzieć, to ze mimo wszytko jednak teamty skupiaja sie na sprawch ważnych dla pogladów (zieloni mówiąo energi) partii czy dzijacych sie w kraju (healt care czy school). Ciekawe może być to, ze np w 2009 bardzo częstym słowem w wypowiedziach partii Progressive Democtrats było Art czy np wybieganie w przysłość do roku 2012.

Teraz podobna analiza tylko z wywołaniem funckji noun_chunks, czyli chcemy zobaczyć jakie grupy słów czesto się ze sobą pojawiały

```
year = 2008
df_year = df[df.date.dt.year == year]
parties = df_year['party_name'].unique()
for party in parties:
    temp = df_year[df_year['party_name'] == party]
    lemmas = temp.speech_en.apply(lambda doc: list(doc.noun_chunks))
    lemmas = lemmas.apply(lambda x: [''.join(str(el)) for el in x if len(el) > 2
    word_counts = dict(Counter(lemmas.sum()).most_common(30))
    wc = WordCloud(width=800, height=400)
    wc.generate_from_frequencies(frequencies=word_counts)
    plt.figure(figsize=(10,8))
    plt.title(party+str(year))
    plt.imshow(wc)
```





Faktycznie zestawienie ze sobą większej grupy słów daje ciekwsze rezulataty. Co wiecej lepiej się rozumie kontekst tych fraz oraz jest łatwiejsza ich interpretowalonść. (Przyczyny występowania danych słów ze sobą)

Praca z Dail_debates_1937-2011_ministers.tab

Wstępne informacje

```
df_ministers = pd.read_table('Dail_debates_1937-2011_ministers.tab')
```

```
df_ministers.head()
```

	govt_number	start_day	start_month	start_year	end_day	end_month	end_
0	18	9	3	1982	14.0	12.0	19
1	18	23	3	1982	14.0	12.0	19
2	20	10	3	1987	12.0	7.0	19

Widać, że będą w tej ramce ministrowie pełniący wiele funkcji, w różnych departamentach i z różnymi długościami kadencji. Trzeba o tym będzie pamiętać i uwzględniać to w analizie.

```
df_ministers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 1185 entries, 0 to 1184
Data columns (total 14 columns):
 #   Column        Non-Null Count  Dtype
```

```
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   govt_number  1185 non-null   int64
 1   start_day    1185 non-null   int64
 2   start_month  1185 non-null   int64
 3   start_year   1185 non-null   int64
 4   end_day      1148 non-null   float64
 5   end_month    1148 non-null   float64
 6   end_year     1148 non-null   float64
 7   position     1185 non-null   object
 8   department   1185 non-null   object
 9   name         1185 non-null   object
 10  memberName   1184 non-null   object
 11  memberID     1184 non-null   float64
 12  start_date   1185 non-null   object
 13  end_date     1148 non-null   object
dtypes: float64(4), int64(4), object(6)
memory usage: 129.7+ KB
```

Widać, że w ramce będą potencjalne nulle, głównie w dacie zkończenia kadencji, co mogło nie nastąpić przed rokiem 2011. Ograniczmy się do naszych lat i sprawdźmy, to ponownie.

```
df_ministers.start_date = pd.to_datetime(df_ministers.start_date)
df_ministers.end_date = pd.to_datetime(df_ministers.end_date)
```

```
df_ministers = df_ministers[( (df_ministers.start_date.dt.year<=2010) & (df_mini
```

```
df_ministers.info()
```

```
<class 'pandas.core.frame.DataFrame'>
Int64Index: 159 entries, 13 to 1181
Data columns (total 14 columns):
 #   Column       Non-Null Count  Dtype
---  ------       --------------  -----
 0   govt_number  159 non-null    int64
 1   start_day    159 non-null    int64
 2   start_month  159 non-null    int64
 3   start_year   159 non-null    int64
 4   end_day      159 non-null    float64
 5   end_month    159 non-null    float64
 6   end_year     159 non-null    float64
 7   position     159 non-null    object
 8   department   159 non-null    object
 9   name         159 non-null    object
 10  memberName   159 non-null    object
 11  memberID     159 non-null    float64
 12  start_date   159 non-null    datetime64[ns]
 13  end_date     159 non-null    datetime64[ns]
dtypes: datetime64[ns](2), float64(4), int64(4), object(4)
memory usage: 18.6+ KB
```

Jak się okazuje problem z nullami znika sam. Pozbądźmy się jeszcze kilku kolumn dla nas zbytecznych.

```
df_ministers = df_ministers.drop(["start_day", "start_month", "start_year", "end
```

```
df_ministers.head()
```

	govt_number	position	department	memberName	memberID	start_date	end
13	26	Taoiseach	Taoiseach	Mr. Bertie Ahern	5.0	2002-06-06	2007
14	27	Taoiseach	Taoiseach	Mr. Bertie Ahern	5.0	2007-06-14	2008
19	26	Minister	Foreign Affairs	Mr. Dermot Ahern	6.0	2004-09-29	2007
			Foreign	Mr. Dermot			

Utwórzmy kilka przydatnych zbiorków. Set z indeksami ministrów w interesujących nas latach oraz 2 słowniki, które będą wiązać pozycje lub departament z danym member ID.

```
ministers_ids = set(df_ministers.memberID.values)
#ministers_ids
```

```
dict_pos = df_ministers.groupby('position')['memberID'].apply(set).to_dict()
#dict_pos
```

```
dict_depart = df_ministers.groupby('department')['memberID'].apply(set).to_dict(
#dict_depart
```

```
len(ministers_ids)
```

```
52
```

Zatem w tym okresie było 52 osobnych ministrów róznego typu, w tym premierzy.

```
print(dict_depart.keys())
print(len(dict_depart.keys()))
```

```
dict_keys(['Agriculture and Food', 'Agriculture, Fisheries and Food', 'Arts
27
```

Powyżej nazwy 27 resortów, zawsze można sobie wypritnować ten słownik, aby dodatkowo zyskać intuicję co do potencjalnej ilości ministrów w danym obszarze w latach 2007-2010 włącznie.

```
print(dict_pos.keys())
print(len(dict_pos.keys()))
```

```
print(len(dict_pos.keys()))
```

```
dict_keys(['Minister', 'Minister of State', 'Taoiseach', 'Tánaiste'])
4
```

Budzić zainteresowanie mogą 2 obco brzmiące nazwy. Taoiseach to osoba premiera w Irlandzkim rządzie. W jego zakresie jest m.in. mianowanie osoby na urząd Tánaiste, czyli wicepremiera.

```
print("Liczba premierów:", len(dict_pos["Taoiseach"]), "Liczba wicepremierów:",
```

```
Liczba premierów: 2 Liczba wicepremierów: 3
```

Większa liczba wicepremirów mówi, że jeden z nich został odwołany lub zrezygnował z pełnienia funkcji. Dokonując analizy osobno dla tych, tak mało licznych grup, trzeba pamiętać, że na charakter wypowiedzi wpływ będzie mieć zarówno pełniona funkcja, ale co ważne, sposób wypowiadania się danej jednostki. (Dłuższe/krótsze wypowiedzi mogą nie być powiązane z funkcją premiera, a jedynie sposobem wygłaszania przemówień i prowadzenia polityki tego z obecnej kadencji.)

```
df_ministers.nunique()
```

```
govt_number     3
position        4
department     27
memberName     52
memberID       52
start_date     13
end_date       11
dtype: int64
```

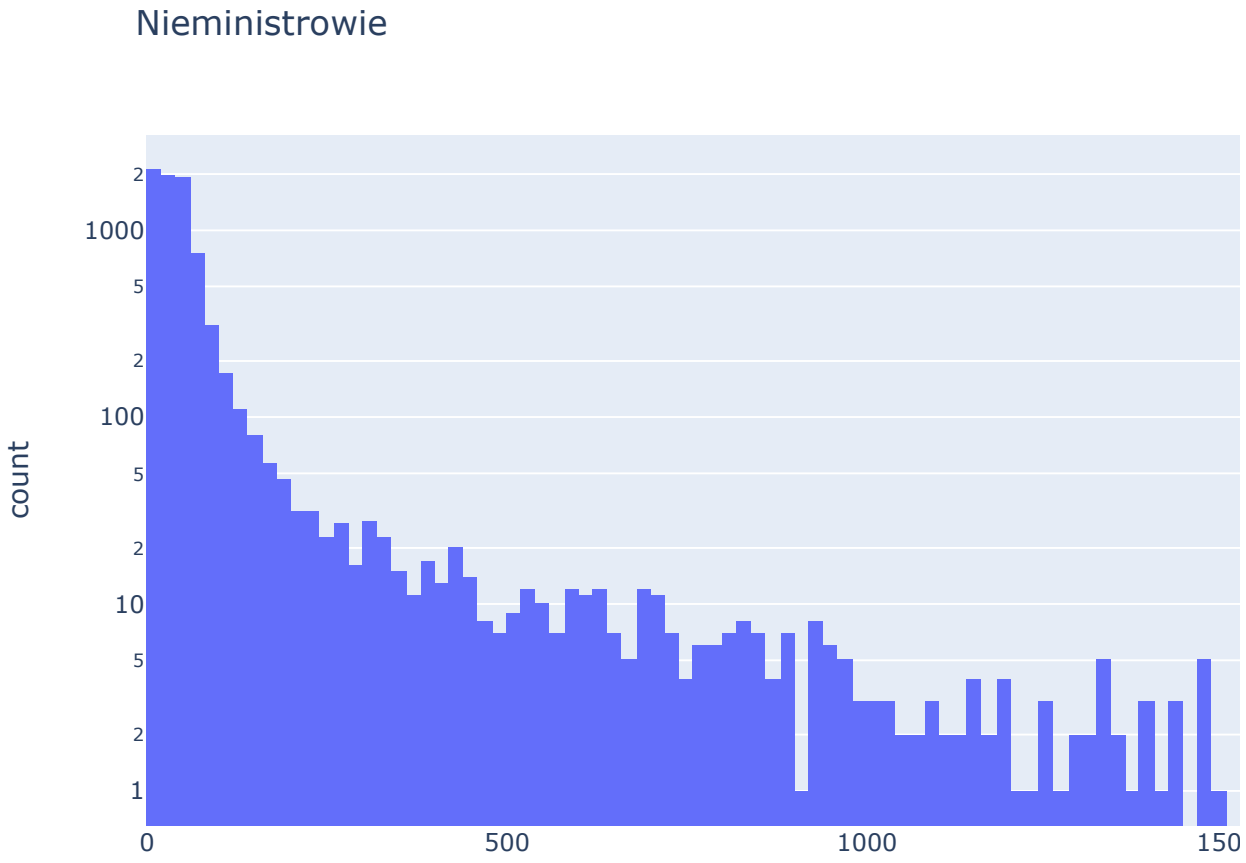
Luźna myśl: w 2008 roku było także referendum na temat retyfikacji traktatu lizbońskiego, pytanie czy mogło być to w jakikolwiek sposób odzwierciedlone w naszym zbiorku?? Ostatecznie ludzie opowiedzieli się przeciwko, co stoi w opozycji do wczesnych sądaży.

Zastanawiać może, skąd taka liczba urzędów, 3 przypadające na okres 2007-2010. Spodziewalibyśmy się 2: jeden przed wyborami w 2007 roku, drugi po nich. To może również tłumaczyć 3 wicepremirów wcześniej i przeczyć wnioskowi, że któryś został odwołany. Odpowiedzi dostarcza wikipedia. "There were two Governments of the 30th Dáil, which was elected at the 2007 general election on 24 May 2007. The 27th Government of Ireland (14 June 2007 – 7 May 2008) was led by Bertie Ahern as Taoiseach, and the 28th Government of Ireland (7 May 2008 – 9 March 2011) was led by Brian Cowen as Taoiseach." W obrędbie tej samej koalicji 2 partii nastąpiła wymiana rządu. Premierem nowego stał się Brian Cowen, uprzednio będący na funkcji wicepremiera. (Fakt wystąpowania tylko 2 premierów tłumaczy fakt, że Bertie Ahern pełnił te funkcje także przed rokiem 2007, bo od 1997)

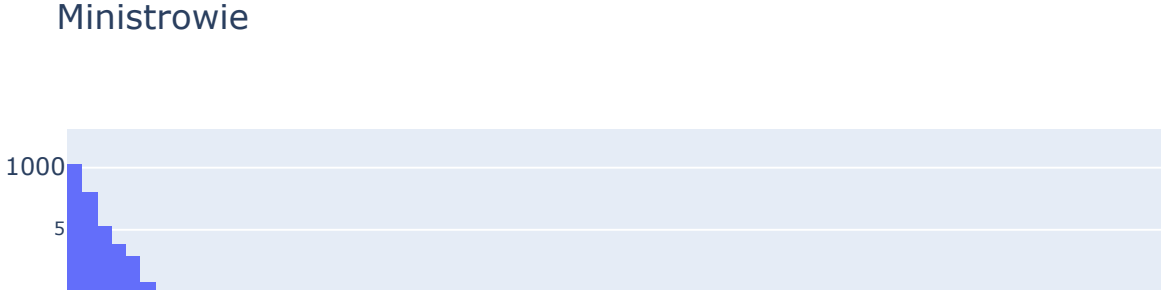
fakt, że Bertie Ahern pełnił tę funkcję także przed rokiem 2007, bo od 1997).

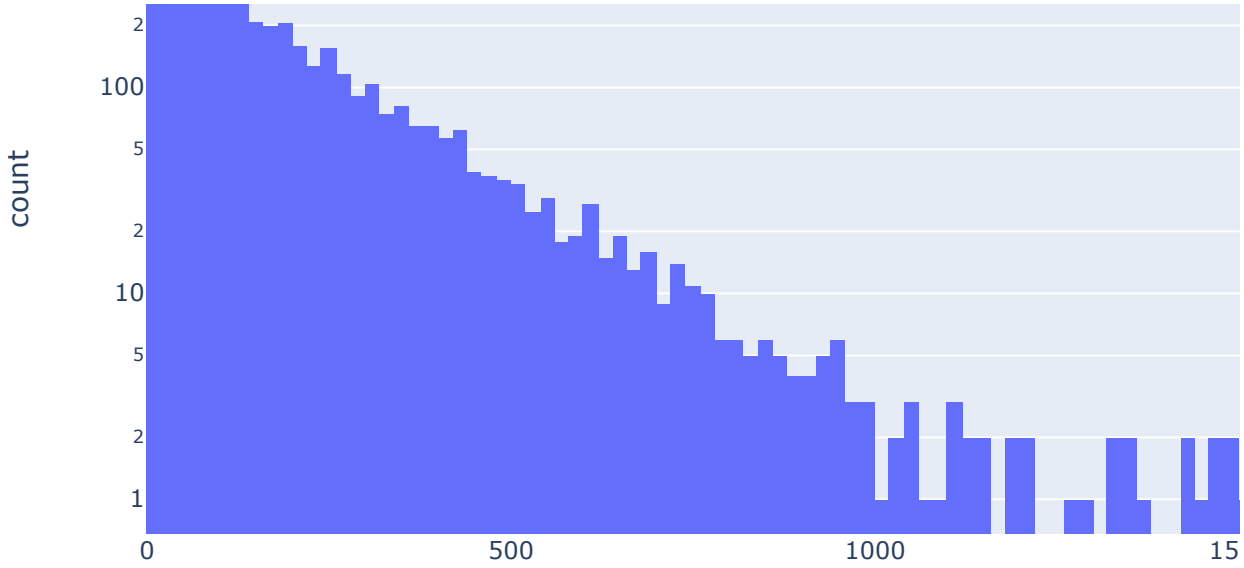
Długość tekstów ministrowie vs. nieministrowie

```
#non-ministers
doc_lens_nm = df.loc[~df['memberID'].isin(ministers_ids)].speech_en.str.len()
doc_lens_nm.hist(title = "Nieministrowie",log_y=True)
```




```
#ministers

doc_lens_m = df.loc[df['memberID'].isin(ministers_ids)].speech_en.str.len()
doc_lens_m.hist(title = "Ministrowie", log_y=True)
```





Można wyciągnąć wstępne wnioski co do długości speechy i rozkładów w poszczególnych grupach. Zastanawiający jest potencjalny brak dużej różnicy w ogólnej liczbie wystąpień. Sprawdźmy zatem ile jest nieministrów w kontrze do 52 ministrów.

```
print("Liczba speechy nieministrów:", len(df.loc[~df['memberID'].isin(ministers_
```

```
    Liczba speechy nieministrów: 8087 liczba speechy ministrów: 5913
```

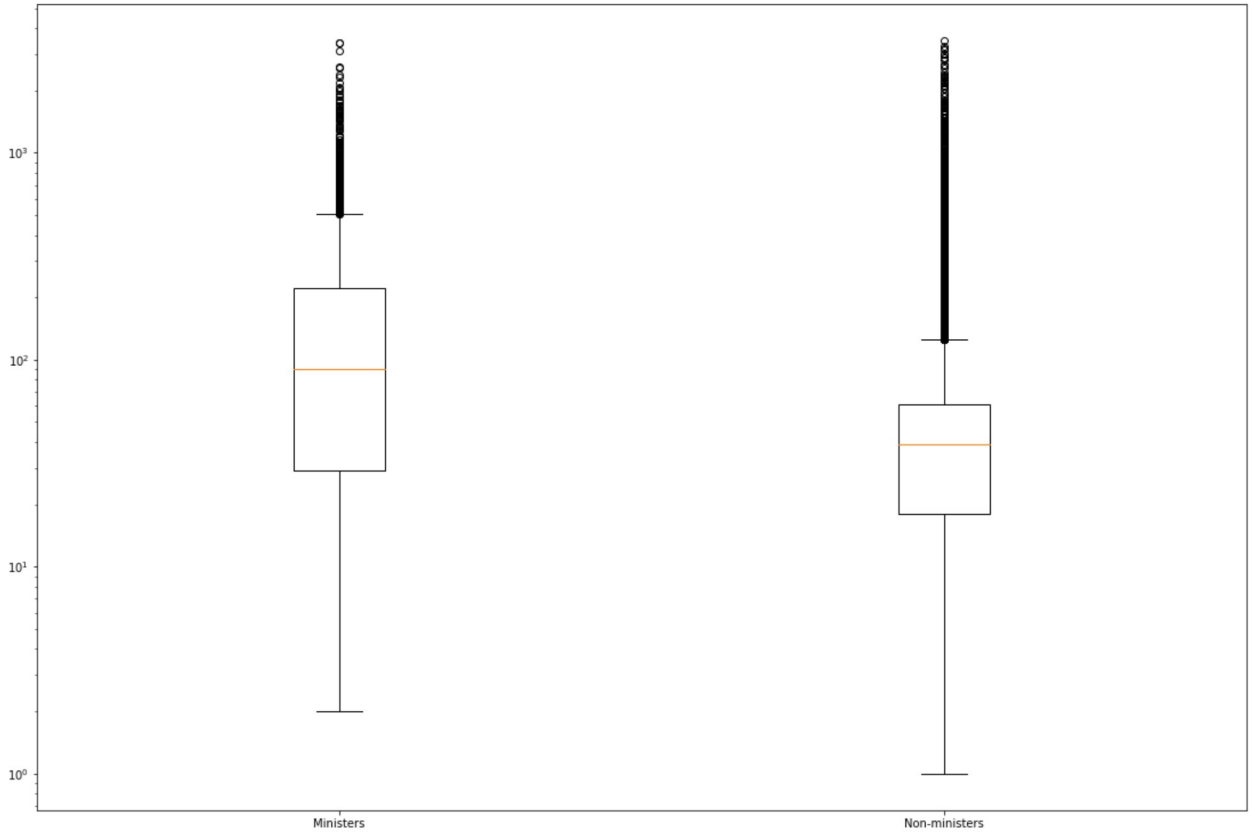
```
print("Liczba wypowiadających się nieministrów:", len(set(df.loc[~df['memberID']
```

```
    Liczba wypowiadających się nieministrów: 145
```

Dosyć zaskakująca dysproporcja, wielu wypowiadających się pełniło wcześniej lub poźniej funkcję ministra. Można to jednak uzasadniaż aż 3 rządami oraz faktem, że większość ze 160 miejsc będzie mieć rządząca koalicja. Istotniejsze jest, że można by rozważyć pomysł dokonywania analiz ze względu także czy dana wypowiedź padła z okresu pełnienia funkcji ministra.

```
data = [doc_lens_m, doc_lens_nm]

# Multiple box plots on one Axes
fig, ax = plt.subplots(figsize=(19, 13))
#fig = plt.figure(figsize=(19, 3))
ax.boxplot(data)
#ax.set_xticklabels["Ministers", "Non-ministers"]
plt.xticks([1, 2], ["Ministers", "Non-ministers"])
ax.set_yscale('log')
plt.show()
```

```
/usr/local/lib/python3.7/dist-packages/matplotlib/cbook/__init__.py:1376: V

Creating an ndarray from ragged nested sequences (which is a list-or-tuple
```



Ministrowie mają średnio dłuższe przemówienia, przesunięte są także odpowiednie kwantyle w kierunku większych wartości. Różnica mogłaby być jeszcze większa biorąc np. długość wypowiedzi tylko z okresu pełnienia funkcji.

Word clouds ministrowie vs. nieministrowie

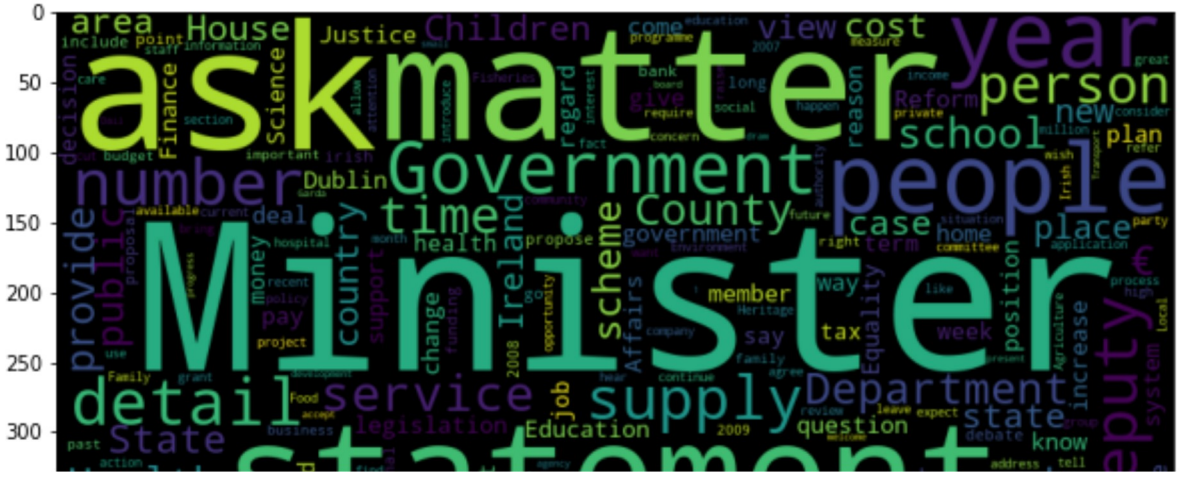
```
def cloud_from_lemmas(lemmas):
  word_counts = Counter(lemmas.sum())

  wc = WordCloud(width=800, height=400)
  wc.generate_from_frequencies(frequencies=word_counts)
  plt.figure(figsize=(10,8))
  plt.imshow(wc)
```

Słowa dla nieministrów.

```
lemmas_nm = df.loc[~df['memberID'].isin(ministers_ids)].speech_en.apply(lambda d

cloud_from_lemmas(lemmas_nm)
```

Słowa dla ministrów:

```
lemmas_m = df.loc[~df['memberID'].isin(ministers_ids)].speech_en.apply(lambda do

cloud_from_lemmas(lemmas_m)
```



Mimo faktu, że duże przecięcie będzie wynikać z faktu występowania słów charakterystycznych dla polityki, zacznijmy rozważać speeche ministrów jedynie za ich kadencji.

Odnotujmy już teraz fakt, że analizując słowa minstrów z tych lat, analizujemy jednocześnie słowa członków koalicji rządzącej FF-Green-PD. Słowa powiązane z wartościami partii, a nie tylko pozycją.

```
min_years_start = df_ministers.groupby('memberID')['start_date'].apply(list).to_
for k in min_years_start.keys():
  min_years_start[k] = min(min_years_start[k])

min_years_end = df_ministers.groupby('memberID')['end_date'].apply(list).to_dict
for k in min_years_end.keys():
  min_years_end[k] = max(min_years_end[k])

min_years_end
```

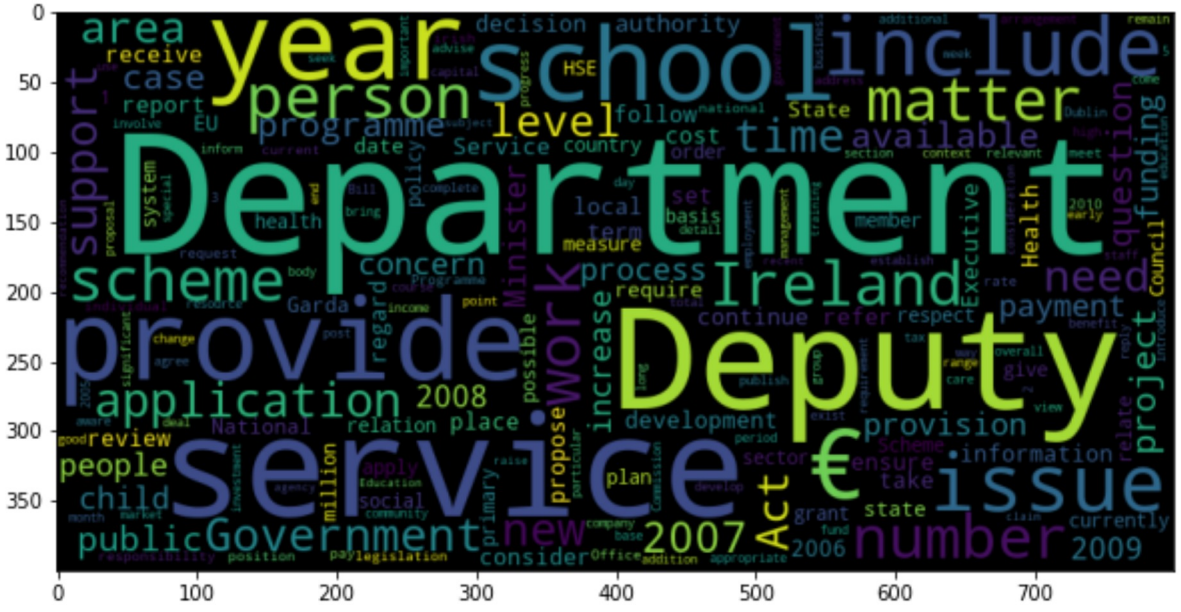
```
{5.0: Timestamp('2008-05-07 00:00:00'),
 6.0: Timestamp('2011-01-20 00:00:00'),
 9.0: Timestamp('2008-05-07 00:00:00'),
 10.0: Timestamp('2009-04-21 00:00:00'),
 108.0: Timestamp('2008-05-07 00:00:00'),
 160.0: Timestamp('2011-03-09 00:00:00'),
 247.0: Timestamp('2011-03-09 00:00:00'),
 253.0: Timestamp('2011-03-09 00:00:00'),
 274.0: Timestamp('2010-05-23 00:00:00'),
 309.0: Timestamp('2011-01-20 00:00:00'),
 373.0: Timestamp('2007-06-14 00:00:00'),
 445.0: Timestamp('2008-05-07 00:00:00'),
 472.0: Timestamp('2011-01-23 00:00:00'),
 484.0: Timestamp('2011-03-09 00:00:00'),
 486.0: Timestamp('2011-01-20 00:00:00'),
 492.0: Timestamp('2011-03-09 00:00:00'),
 565.0: Timestamp('2011-03-09 00:00:00'),
 594.0: Timestamp('2011-01-20 00:00:00'),
 602.0: Timestamp('2009-04-21 00:00:00'),
 604.0: Timestamp('2008-05-07 00:00:00'),
 628.0: Timestamp('2011-03-09 00:00:00'),
 630.0: Timestamp('2011-03-09 00:00:00'),
 693.0: Timestamp('2011-01-19 00:00:00'),
 719.0: Timestamp('2007-06-14 00:00:00'),
 741.0: Timestamp('2009-04-21 00:00:00'),
 770.0: Timestamp('2011-03-09 00:00:00'),
 852.0: Timestamp('2010-05-23 00:00:00'),
 854.0: Timestamp('2010-02-18 00:00:00'),
 868.0: Timestamp('2007-06-14 00:00:00'),
 899.0: Timestamp('2011-01-20 00:00:00'),
```

```
980.0: Timestamp('2009-04-21 00:00:00'),
1003.0: Timestamp('2011-03-09 00:00:00'),
1022.0: Timestamp('2010-02-23 00:00:00'),
1044.0: Timestamp('2011-03-09 00:00:00'),
1076.0: Timestamp('2007-06-14 00:00:00'),
1094.0: Timestamp('2009-04-21 00:00:00'),
1158.0: Timestamp('2008-05-07 00:00:00'),
1777.0: Timestamp('2011-03-09 00:00:00'),
1780.0: Timestamp('2009-04-21 00:00:00'),
1796.0: Timestamp('2011-03-09 00:00:00'),
1804.0: Timestamp('2009-04-21 00:00:00'),
1811.0: Timestamp('2011-03-09 00:00:00'),
1814.0: Timestamp('2011-01-23 00:00:00'),
1835.0: Timestamp('2007-06-14 00:00:00'),
1837.0: Timestamp('2007-06-14 00:00:00'),
1838.0: Timestamp('2011-03-09 00:00:00'),
2002.0: Timestamp('2011-03-09 00:00:00'),
2028.0: Timestamp('2011-01-23 00:00:00'),
2156.0: Timestamp('2011-03-09 00:00:00'),
2157.0: Timestamp('2011-01-23 00:00:00'),
2199.0: Timestamp('2011-03-09 00:00:00'),
2201.0: Timestamp('2011-03-09 00:00:00')}
```

```
df_only_curr_minister = df.loc[(df['memberID'].isin(ministers_ids)) & (df['date'
```

```
lemmas_curr_min = df_only_curr_minister.speech_en.apply(lambda doc: [token.lemma
```

```
cloud_from_lemmas(lemmas_curr_min)
```



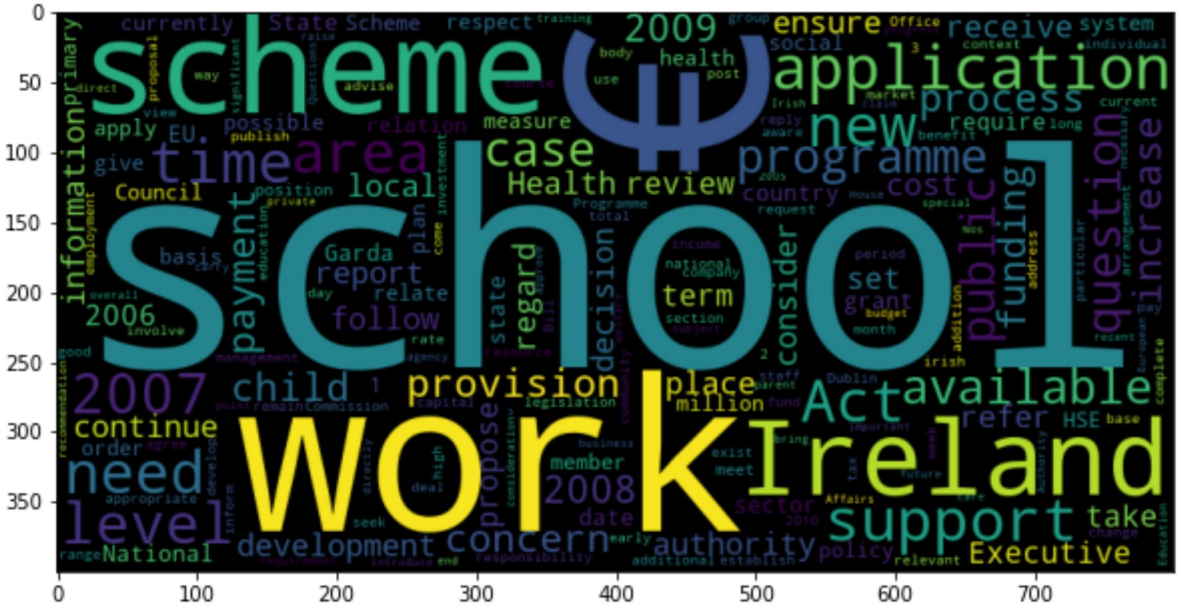
Taka analiza pozwala zauważyć, że słowa typowe dla polityki zmieniają się wraz z pozycją. Także używane zwroty grzecznościowe (z pewnością rozpoczynające część z wystąpień). Prócz słów typowych można doszukiwać się także innych, przykładowo symbol euro sugerujący podnoszenie kwestii wydatków lub planowania budżetu.

Teraz spróbujmy odflitrować niektóre słowa i powtórzyć tę samą chmurę.

```
not_interesting = set(["department", "deputy", "provide", "service", "include",
```

```
lemmas_specific = df_only_curr_minister.speech_en.apply(lambda doc: [token.lemma
```

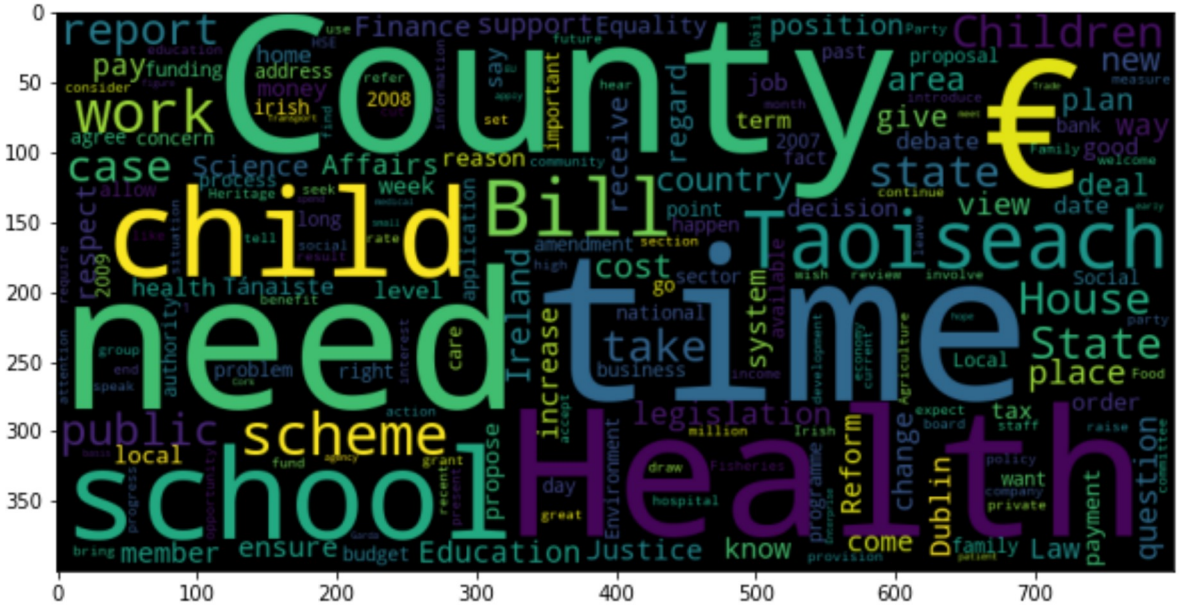
```
cloud_from_lemmas(lemmas_specific)
```



Teraz na lupę zbiór komplementarny do tego, czyli nieministrowie w trakcie wygłaszanie przemówień.

```
lemmas_specific = df.loc[~df["speechID"].isin(set(df_only_curr_minister["speechI
```

```
cloud_from_lemmas(lemmas_specific)
```



Można zauważyć różnice w danych tematach. W pierwszym wypadku istotne wydają się np. work, school, Ireland, euro, drugi to np. time, County, child, Health, Bill.

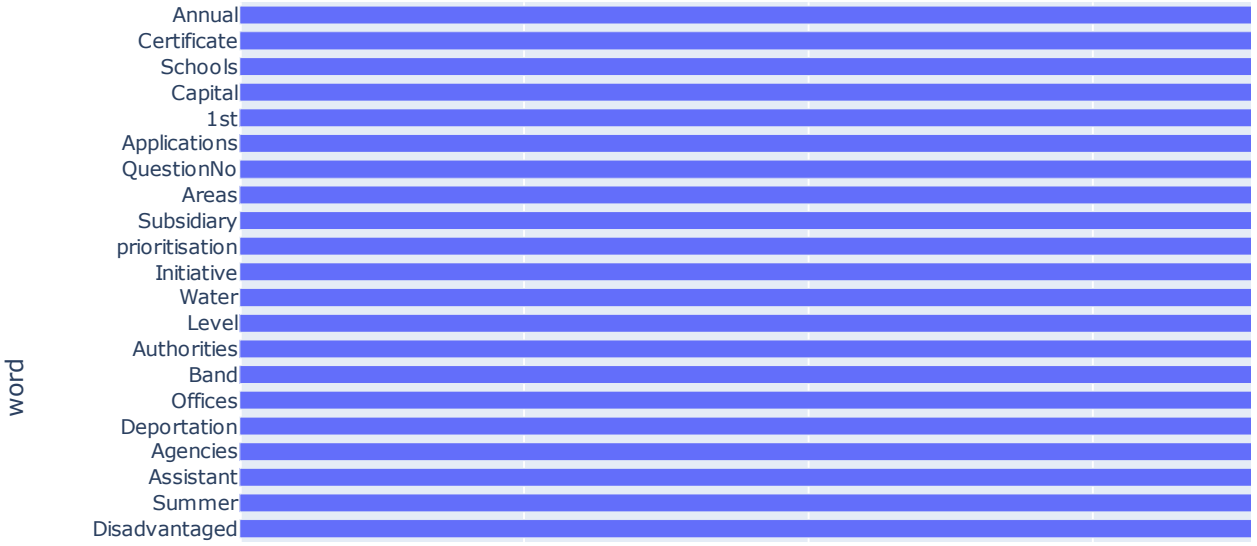
Barploty ministrowie vs. nieministrowie

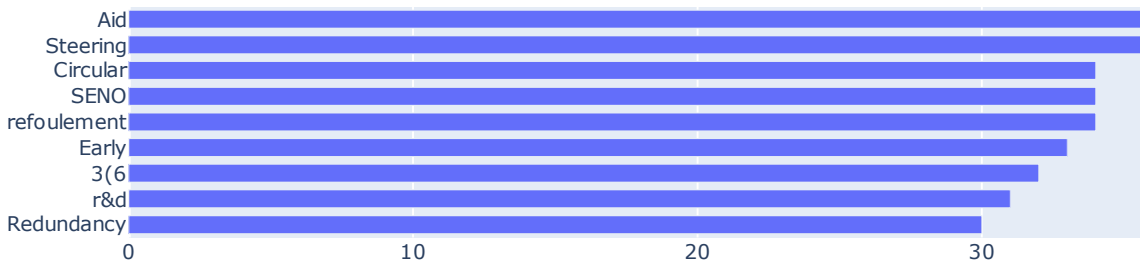
```
def plot_counts(counts):
  fig = px.bar(counts,orientation='h', y='word', x='count')

  fig['layout']['yaxis']['autorange'] = "reversed"
  fig.update_layout(bargap=0.30, font={'size':10})
  return fig


#word count for certain group
lemmas_only_ministers = df_only_curr_minister.speech_en.apply(lambda doc: [token
word_counts_only_ministers = Counter(lemmas_only_ministers.sum())

lemmas_only_non_ministers = df.loc[~df["speechID"].isin(set(df_only_curr_ministe
word_counts_only_non_ministers = Counter(lemmas_only_non_ministers.sum())
```

Sporządzimy wykres, gdzie będą znajdować się słowa najcześciej wypowiedziane przez daną grupę, nie występujące ani raz w drugiej grupie. Najpierw dla ministrów

Ministrowie:

```
counts = pd.DataFrame(Counter({k: v for k, v in word_counts_only_ministers.items
plot_counts(counts)
```






```
counts = pd.DataFrame(Counter({k: v for k, v in word_counts_only_non_ministers.i

plot_counts(counts)
```

Bardziej interesujący wydaje się wykres numer 2. Zestaw słow jak disgrace, scandal, shame, guillotine, outragous, coalition, jail, comfort, hang, fool, syndrome, savage. Są to słowa silnie nacechowane. Po pierwsze widzimy, że ministrowie mogą być o wiele bardziej zachowawczy w doborze słów. Po drugie takie słowa mogą służyć do formułowania zarzutów wobec rządzącej obecnie koalicji.

```
non_min_speeches = df.loc[~df["speechID"].isin(set(df_only_curr_minister["speech
```

```
non_min_speeches = df.loc[df['speechID'].isin(set(df_only_cuff_minister['speech
```

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "jail" in x)
```

```
3818076    asked the Minister for Justice, Equality and L...
3739718    I am amazed to hear that considering the war o...
3743888    A total of 33 people charged with homicide, 59...
3768625    I move amendmentNo. 84: In page 21, to delete ...
3752474    I have not prepared for this as much as I woul...
4110466    I join with the statement of the Minister of S...
4049927    I suggest that 2008 is normally seen as the st...
4101706    That would make it 40 people at any one time. ...
4152617    asked the Minister for Justice and Law Reform ...
Name: speech, dtype: object
```

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "jail" in x)
```

```
'A total of 33 people charged with homicide, 595 charged with assault and
97 charged with sexual offences were out on bail last year. Where is the w
ar on crime we have been promised? Where is the right of a judge to tag el
ectronically if necessary or the right to earn remission in jail without a
quarter of the sentence being completed before the criminal enters prison?
```

Można sobie poczytać kilka wybranych. Pierwsze nie zawierają gróźb więzenia dla "niekompetentnych rządzących", jednak i tak dosyć łatwo znaleźć zarzuty wobec rządu. Minister for Justice and Law Reform często jest wspominany w tych przemówieniach

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "guillotine"
```

```
3776115    This is a very bad start to the guillotine sea...
3762233    This highly important legislation, which has w...
3763831    We cannot have a full debate because the Bill ...
3750224    I am not too happy with the Government’s decis...
3820387    I support Deputy Jan O’Sullivan, whose amendme...
3759970    That is not agreed to for the same reason we o...
3881436    The Deputy has made his point. As this is Repo...
3893509    We are dealing with probably the most serious ...
4016823    That is that, a Leas-Cheann Comhairle. I expec...
4001076    I am delighted to speak on this Bill, represen...
4048243    There will be four common or garden Bills guil...
3992931    We are opposing the principle of guillotine an...
4004787    Will the Tánaiste indicate what is likely to h...
3951522    The Labour Party does not agree to the guillot...
3949324    The Financial Emergency Measures in the Public...
4115827    I have to object not only to the imposition of...
Name: speech, dtype: object
```

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "guillotine"
```

```
'We cannot have a full debate because the Bill is being guillotined.'
```

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "guillotine"
```

```
'I am not too happy with the Government’s decision to guillotine this Bil
l '
```

Allocation of time or 'guillotine' motions have been used by governments to limit the amount of time that MPs can spend debating a particular stage of a Bill in the House of Commons.

Wyrażanie niezadowolenia wobec rządu.

Teraz już ostatnie shame. Do analizy pozostałych zachęcamy samodzielnie.

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "scandal" in
```

```
3793098    In the Minister’s speech, one of the programme...
3820180    I wish to advise the House of the following ma...
3783355    The latest scandals in our crisis ridden healt...
3740595    Why has this matter been left until the eleven...
3757985    That is a great pity. We do not propose an ind...
3766251    I take this opportunity to extend my support a...
3750972    One of the most significant matters affecting ...
3855489    For many years Leopardstown Park Hospital has ...
3833268    I thank Deputy Penrose for sharing time. I sec...
3886475    Schools are now facing spiralling costs for fu...
3892653    That sounds daunting. I will begin my observat...
4039250    We were well warned about the impact of the Mu...
4019828    I welcome the opportunity to speak on this imp...
3965325    I thank the Acting Chairman for the opportunit...
4036082    I am pleased to speak on the Bill which I welc...
3957343    I join the Minister in extending my condolence...
4089408    I thank the Minister and his officials for the...
4109154    Since 2000, at least 508 migrant children plac...
Name: speech, dtype: object
```

```
non_min_speeches.loc[list(lemmas_only_non_ministers.apply(lambda x: "scandal" in
```

```
'The latest scandals in our crisis ridden health service and the failure o
f the Government to deliver radiotherapy show once and for all that the Ta
oiseach, Deputy Bertie Ahern, and the Minister for Health and Children, De
puty Harney, are unfit to govern. The ill-conceived, confused and mismanag
ed approach of the Government and the HSE, particularly in the provision o
f vital cancer treatment services, is a far bigger scandal than anything w
e have addressed in the Chamber [151]today. As Deputy O’Sullivan stated, I
```

Pierwsze zdanie z powyższego outputu.

TF-IDF

```
# !pip install scikit-learn
```

```
# from sklearn.feature_extraction.text import TfidfVectorizer
```

```
# lemmas_common = df.speech_en.apply(lambda doc: [token.lemma_ for token in doc
```

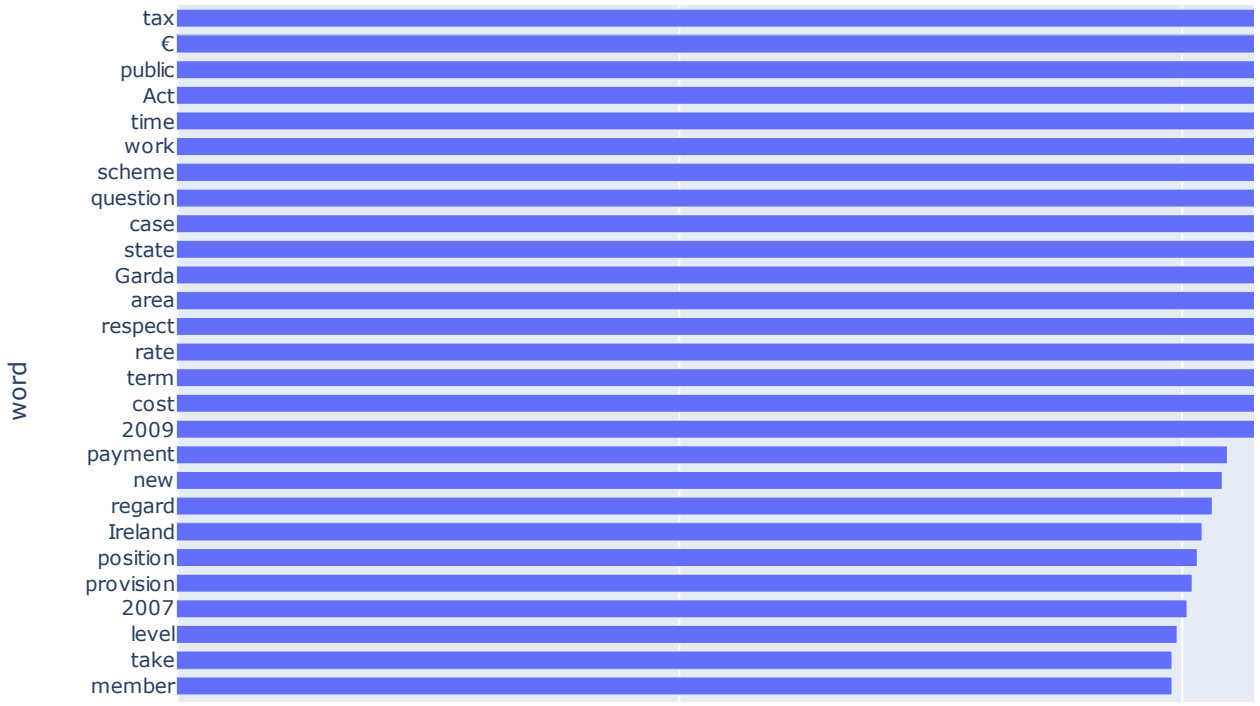
```
# vectorizer = TfidfVectorizer()
# X = vectorizer.fit_transform(lemmas_common.apply(lambda x: " ".join(x)))
# vectorizer.get_feature_names_out()

# tfidf_df = pd.DataFrame(X.toarray(), index = lemmas_common.keys(),columns=vect

# tfidf_df = tfidf_df.stack().reset_index()

# tfidf_df = tfidf_df.rename(columns={0:'tfidf','level_1': 'term', 'level_0': 's
# to_cos = tfidf_df.sort_values(by=['speech_id','tfidf'], ascending=[True,False]
# Counter(to_cos["term"]).most_common(30)
```

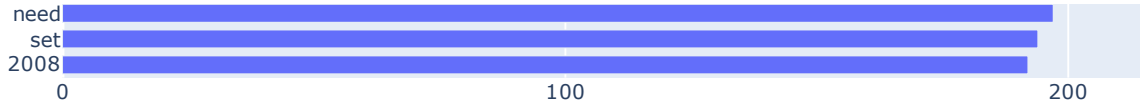
W końcu nie używamy go, był prototyp do wyciągnięcia np. 5 słow z najwyższym indexem z każdego speecha i potem grupowanie ich np. dla konkretnej partii i wyciąganie most common.

Względem departamnetów

Ten notebook i tak jest długi prezentujemy tylko resort finansów, można tak zrobić pozostałe i powyciągać wnioski zminiając nazwę departamentu.

```
lemmas_fin = df.loc[df['memberID'].isin(dict_depart["Finance"])].speech_en.apply
word_counts_fin = Counter(lemmas_fin.sum())
counts_fin = pd.DataFrame(word_counts_fin.most_common(30), columns=['word', 'cou
plot_counts(counts_fin)
```





Nie powinna dziwić np. popularność słowa tax czy euro w przemowach ministra finansów

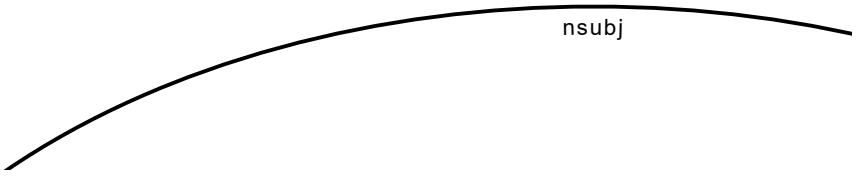
Institutional Grammar tagging algorithm

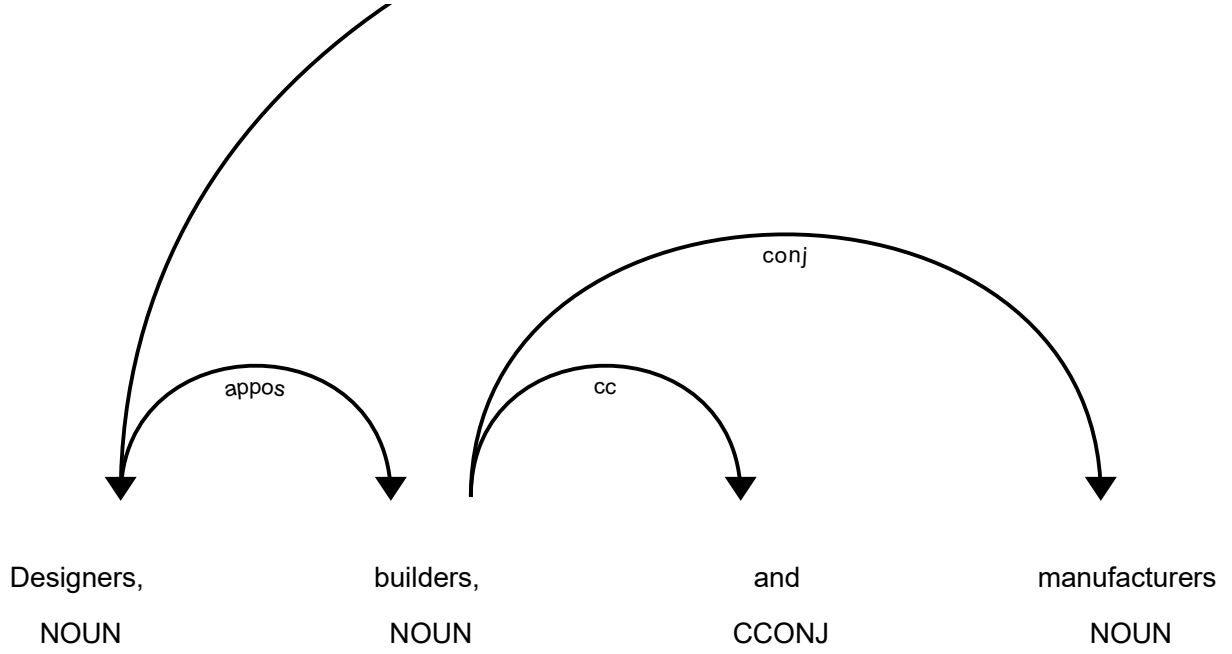
```
"""
Program to implement and demonstrate institutional grammar tagging algorithm. Du
library, coreference resolution does not work.
"""


# Library for coreference resolution
# !pip install neuralcoref

# Neuralcoref is not compatible with spaCy v3.0
# import neuralcoref

# neuralcoref.add_to_pipe(en)
from typing import Tuple, Union, List, Any, Optional

import spacy as spacy


def institutional_grammar_tagging_algorithm(sentence: spacy.tokens.Span) -> \
        Tuple[Union[List[Any], Any], Union[List[Any], Any], List[Optional[Any]]]
    """
    Implementation of institutional grammar tagging algorithm.
    :param sentence: deontic sentence
    :return: attributes, objects and verbs of the given deontic sentence
    """
    attributes = []
    objects = []
    verbs = []
    verb = sentence.root

    while verb is not None:
        attr = verb
        verb = None
        verbs.append(attr)

        newSubj = [c for c in attr.children if c.dep_ == "nsubj"]
        newPassiveSubj = [c for c in attr.children if c.dep_ == "nsubjpass"]

        if len(newSubj) == 0 and len(newPassiveSubj) == 0:
```

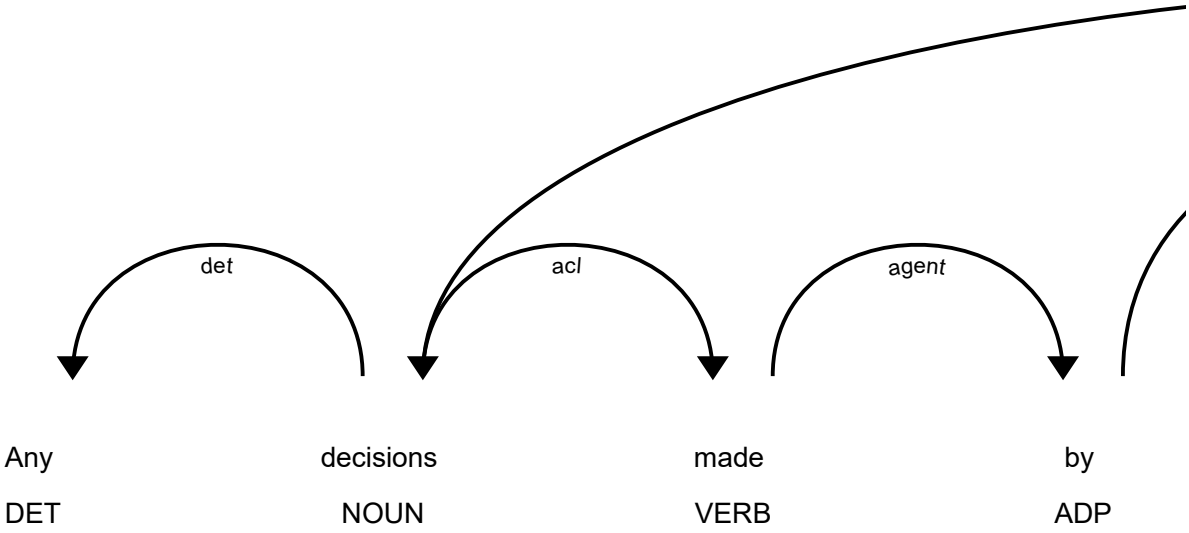
```
            attributes = [clausal for c in attr.children for clausal in c.childr

        attributes = attributes + newSubj
        objects = objects + newPassiveSubj + [c for c in attr.children if c.dep_

        if attr.dep_ == "conj" and attr.pos_ == "VERB":
            verb = attr.head

    for subject in attributes:
        for attr in attributes:
            if attr.dep_ == "conj":
                attributes.append(subject)

            attributes = attributes + [s for s in subject.children if s.dep_ ==

            if subject.pos_ == "PRONOUN":
                # coreference resolution does not work
                pass

    for objectx in objects:
        objects = objects + [s for s in objectx.children if s.dep_ == "conj"]

    return attributes, objects, verbs



testcase1 = en("Designers, builders, and manufacturers should submit details and
for sent in testcase1.sents:
  print(institutional_grammar_tagging_algorithm(sent))
print("\n")
for sent in testcase1.sents:
  print(sent.root)
```

```
([Designers], [details, documentation], [submit])
([], [decisions], [logged])


submit
logged
```

```
from spacy import displacy
for sent in testcase1.sents:
  displacy.render(sent, jupyter=True)
```








```
spacy.explain("appos")
```

```
'appositional modifier'
```