Analiza ilościowa przemówień inauguracyjnych prezydentów Stanów Zjednoczonych

Mateusz Wyszecki

17.01.2020
Komputerowe przetwarzanie dokumentów

Wstęp

Projekt ten ma na celu porównanie przemówień inauguracyjnych w celu analizy ewolucji języka w polityce. Wyniki tej analizy mogą mieć szereg zastosowań w dziedzinach socjologii i politologii, ponieważ ilustrują to, w jaki sposób zmiany polityczne, ekonomiczne i kulturowe wpływają na język.

Narzędzia

- **Biblioteka NLTK** (https://www.nltk.org/) wykorzystana do szczegółowej analizy języka
- **Biblioteka TextBlob** (https://textblob.readthedocs.io/) wykorzystana do prostej analizy sentymentu
- **Biblioteka MatPlotLib** (https://matplotlib.org/) wykorzystana do tworzenia wykresów

.

Dane do analizy

Wszystkie dane do analizy pochodzą z korpusów będących częścią biblioteki NLTK. Brany był pod uwagę korpus “Inaugural”, w skład którego wchodzą przemówienia inauguracyjne prawie wszystkich prezydentów Stanów Zjednoczonych. W celu porównania partii politycznych analizowany był tylko fragment korpusu (poczynając od przemowy Ulyssesa S. Granta).

1. Liczenie ilości wystąpień słowa w tekście

Ta funkcja zwraca wykres ilustrujący liczbę wystąpień danego słowa w tekście. Za argument może przyjąć dowolne słowo od użytkownika lub jedno z określonych słów kluczowych do analizy całego pola semantycznego.

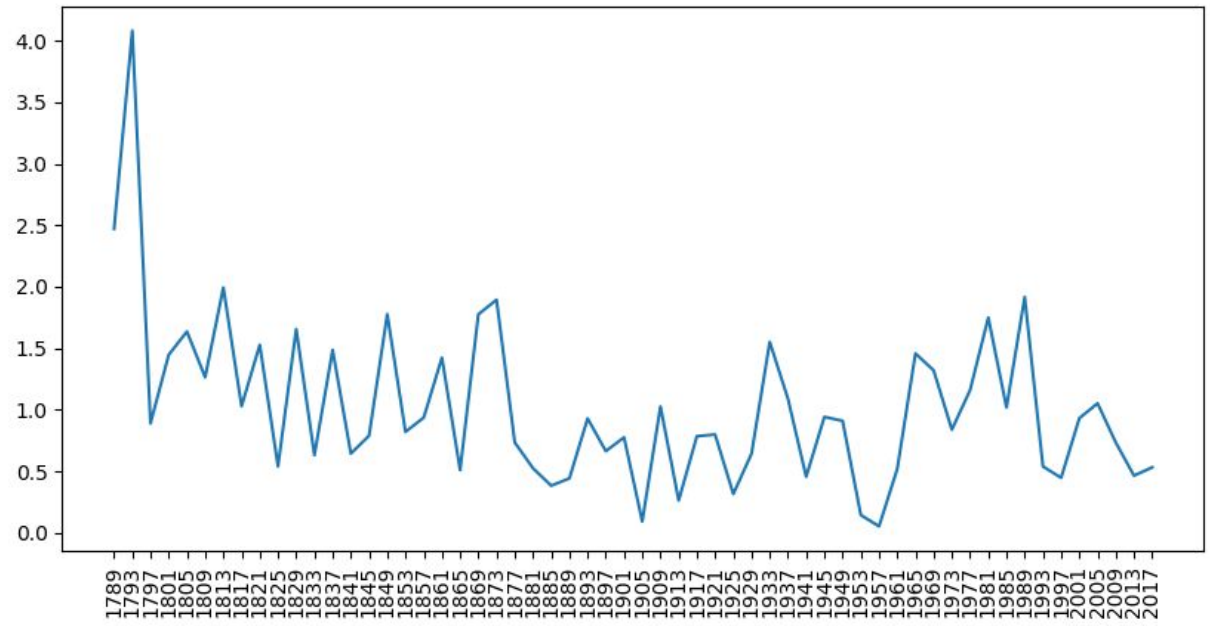
```
def CountWords(words):
    x = []
    y = []
    for fileid in inaugural.fileids():
        count = 0
        for w in inaugural.words(fileid):
            if w.lower() in words:
                count += 1
        per = (count/len(inaugural.words(fileid)))*100
        y.append(fileid[:4])
        x.append(per)

    plt.title('Liczba wystąpień:')
    plt.xticks(rotation=90)
    plt.plot(y, x)
    plt.show()
```

Słowa kluczowe i listy słów, które się w nich zawierają:

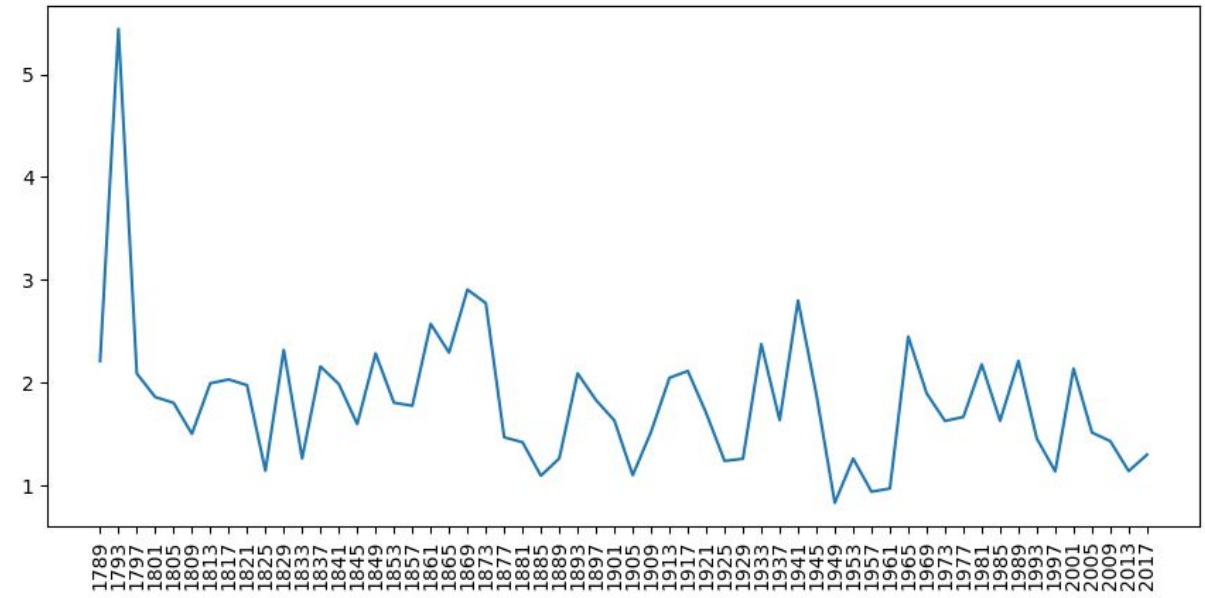
```
>>> politics = ["justice", "constitution", "democracy", "president",
"communism", "republic", "congress"]
>>> finance = ["tax", "money", "commerce", "economy"]
>>> emotions = ["love", "happiness", "dignity", "courage", "sacrifice",
"compassion", "loyalty", "patriotism", "morality", "generosity"]
>>> spirit = ["bible", "god", "spirit", "soul", "destiny", "prayer",
"faith", "truth", "wisdom", "conscience"]
```

Wyniki dla każdej domeny:

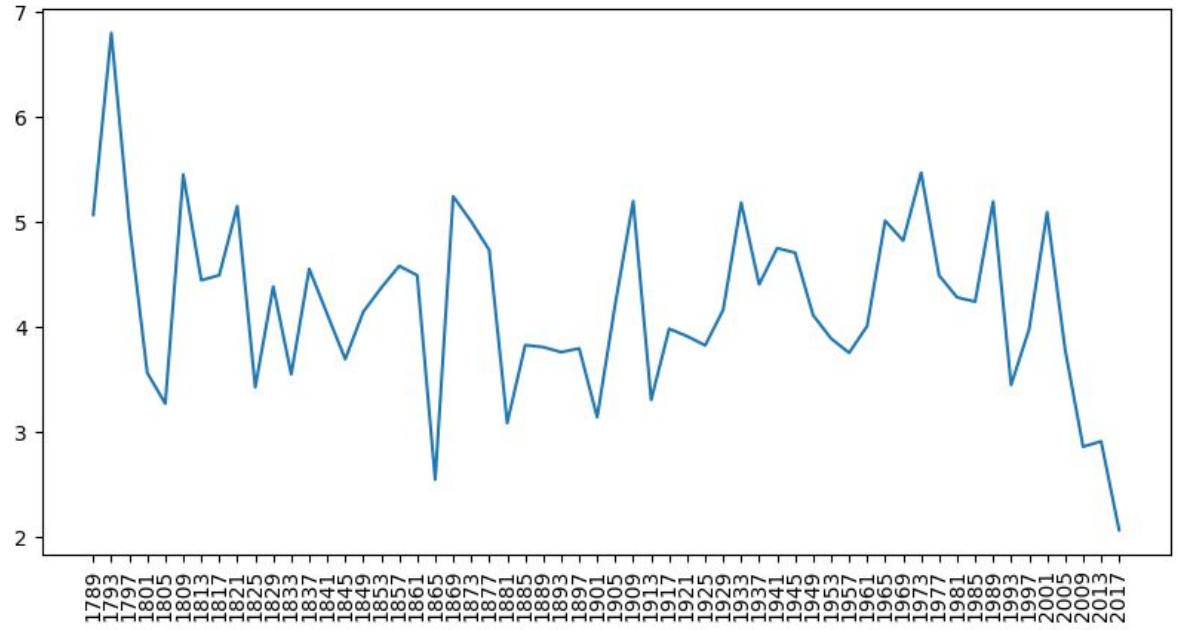
Emotions



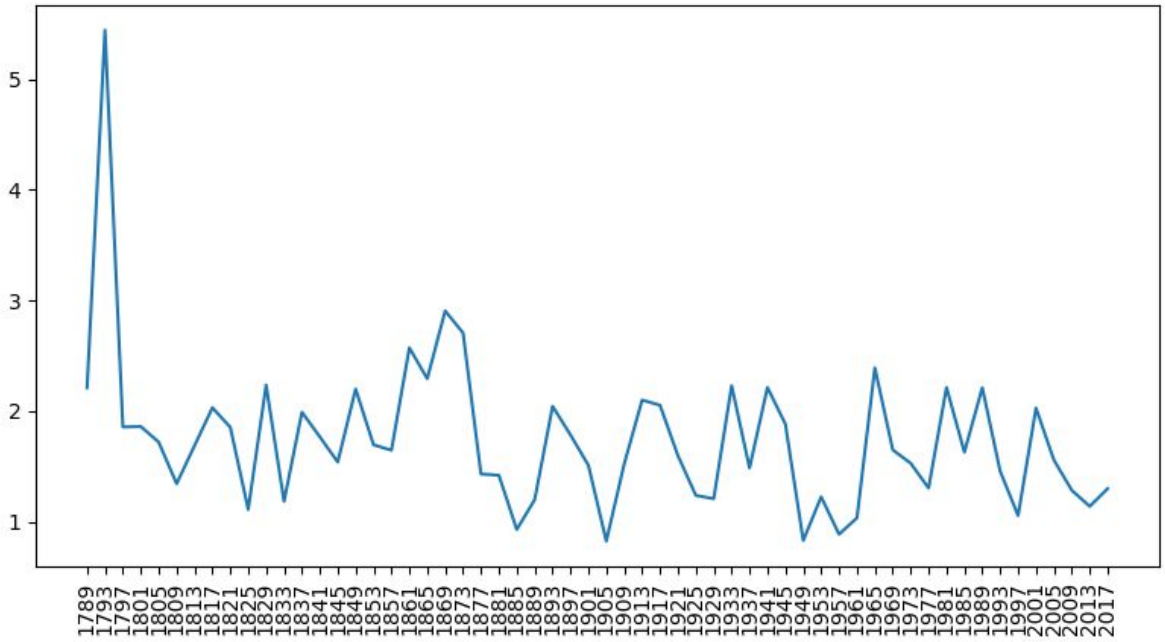
Spirit



Finance



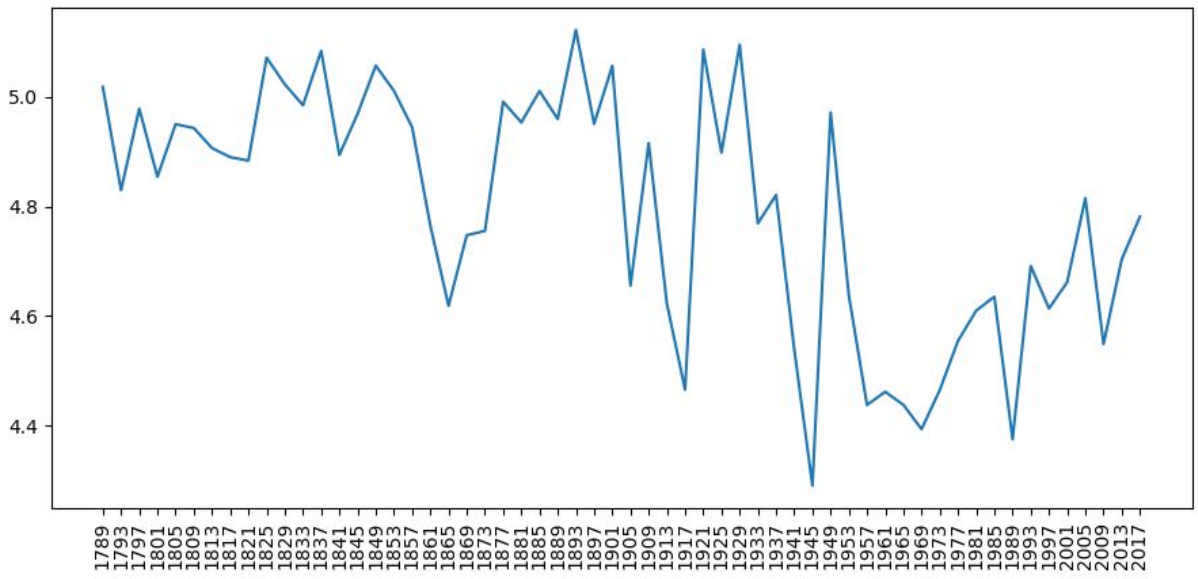
Politics


5
4
3
2
1
1789
1793
1797
1801
1805
1809
1813
1817
1821
1825
1829
1833
1837
1841
1845
1849
1853
1857
1861
1865
1869
1873
1877
1881
1885
1889
1893
1897
1901
1905
1909
1913
1917
1921
1925
1929
1933
1937
1941
1945
1949
1953
1957
1961
1965
1969
1973
1977
1981
1985
1989
1993
1997
2001
2005
2009
2013
2017

2. Średnia długość słowa

Ta funkcja zwraca średnią długość słowa dla każdego tekstu.

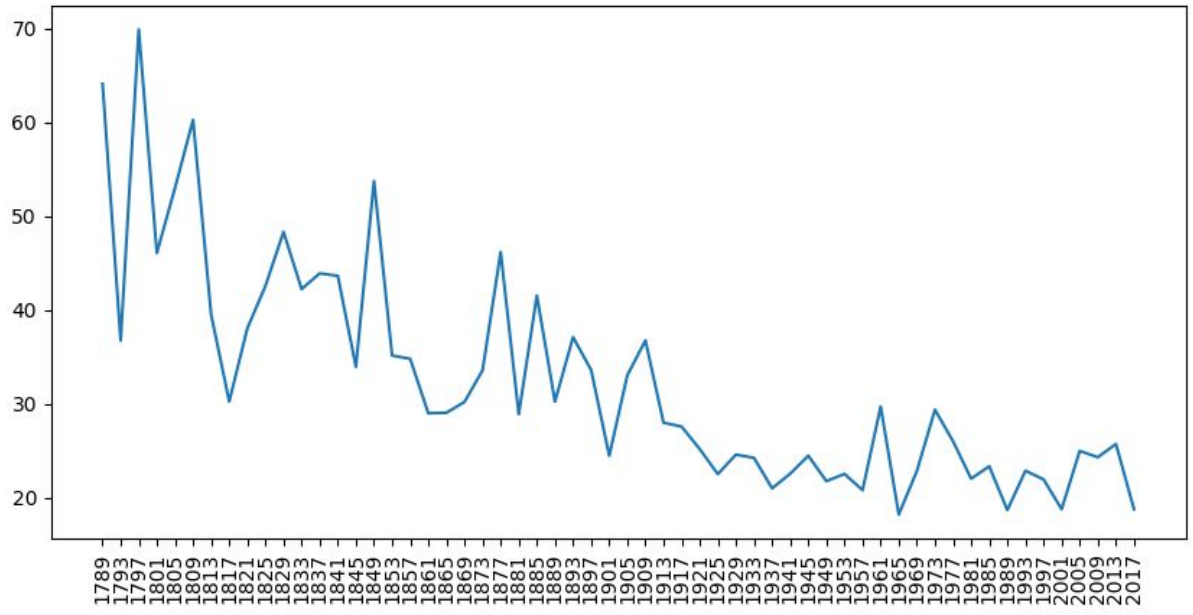
```
def avgWord():
    x1 = []
    y1 = []
    for fileid in inaugural.fileids():
        words = inaugural.raw(fileids=fileid)
        words = words.split()
        average = sum(len(word) for word in words) / len(words)
        print(fileid[:4], "-", average)
        y1.append(fileid[:4])
        x1.append(average)

    plt.title('Średnia długość słowa:')
    plt.xticks(rotation=90)4
    plt.plot(y1, x1)
    plt.show()
```



3. Średnia długość zdania

Ta funkcja zwraca średnią długość zdania dla każdego tekstu.

```
def avgSent():
    x2 = []
    y2 = []

    for fileid in inaugural.fileids():
        average = sum(len(sent) for sent in
inaugural.sents(fileids=[fileid])) / len(inaugural.sents(fileids=[fileid]))
        print(fileid[:4], "-", average)
        y2.append(fileid[:4])
        x2.append(average)

    plt.title('Średnia długość zdania:')
    plt.xticks(rotation=90)
    plt.plot(y2, x2)
    plt.show()
```



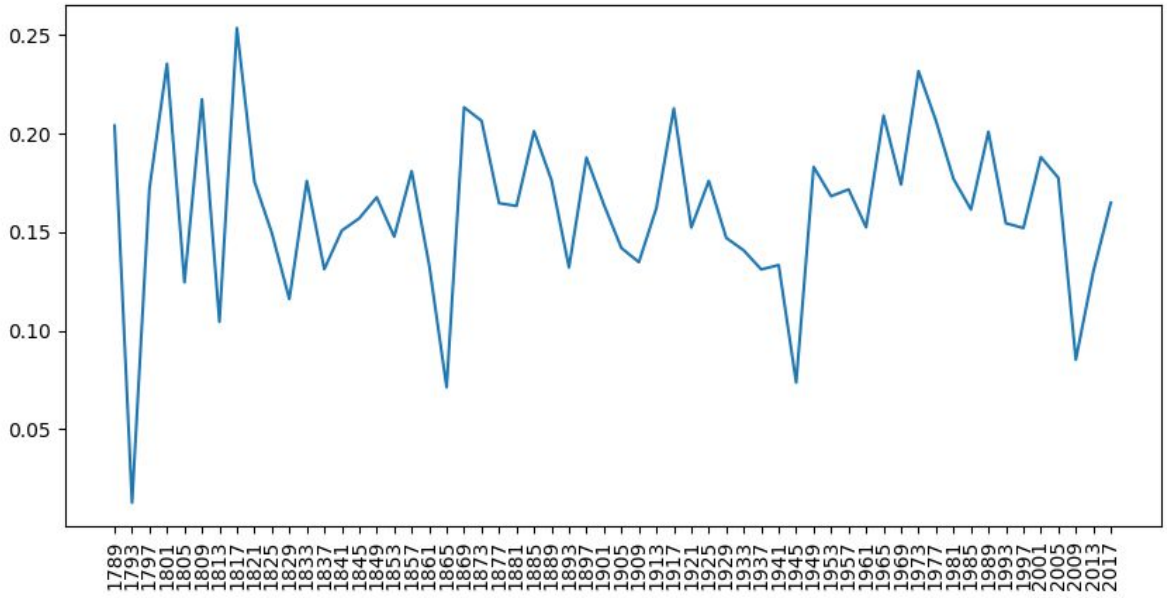
4. Analiza sentymentu

Ta funkcja wykorzystuje bibliotekę TextBlob, która zwraca dwie wartości dla każdego tekstu:

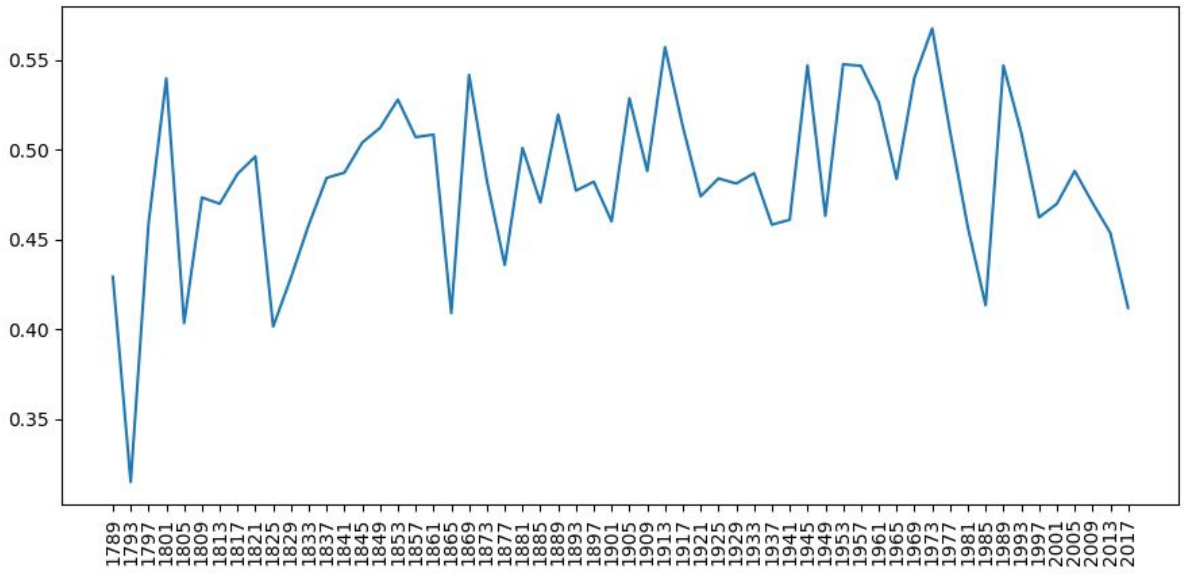
- **Polarity**, czyli uśrednione nacechowanie emocjonalne tekstu na skali od -1 do 1, (gdzie -1 to sentyment negatywny, a 1 -- pozytywny)
- **Subjectivity**, czyli uśredniona ilość słów nacechowanych emocjonalnie w tekście w skali od 0 (obiektywność) do 1 (subiektywność)

```
def senti():
    x3 = []
    y3 = []
    x31 = []

    for fileid in inaugural.fileids():
        text = inaugural.raw(fileids=fileid)
        senti = TextBlob(text)
        print(fileid[:4], "-", senti.sentiment)
        y3.append(fileid[:4])
        x3.append(senti.sentiment[0])
        x31.append(senti.sentiment[1])
    plt.title('Polarity')
    plt.xticks(rotation=90)
    plt.plot(y3, x3)
    plt.show()
    plt.title('Subjectivity')
    plt.xticks(rotation=90)
    plt.plot(y3, x31)
    plt.show()
```

Polarity:



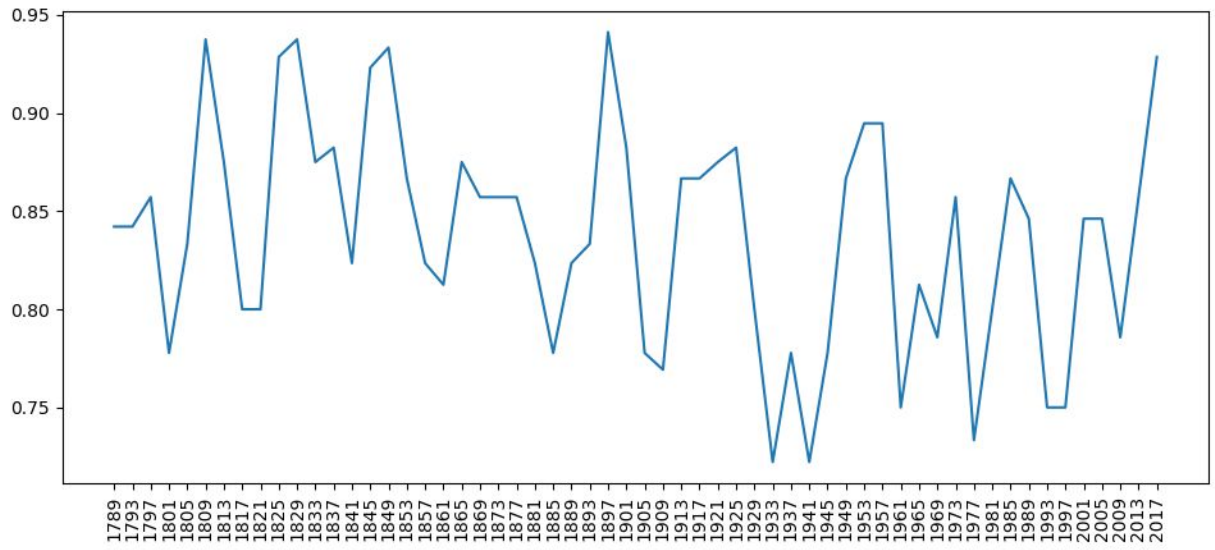
Subjectivity:



5. Zróżnicowanie słownictwa

Ta funkcja zwraca uśrednioną miarę “bogactwa” słownictwa, tj. Ilość unikatowych słów podzieloną przez ilość wszystkich słów w tekście.

```
def lexDiv():
    y4 = []
    x4 = []

    for fileid in inaugural.fileids():
        div = len(set(fileid)) / len(fileid)
        print(fileid[:4], "-", div)
        y4.append(fileid[:4])
        x4.append(div)

    plt.title('Różnorodność słownictwa')
    plt.xticks(rotation=90)
    plt.plot(y4, x4)
    plt.show()
```



6. Porównywanie ilości słów

Ta funkcja pozwala nam porównać ilość wystąpień w tekście dwóch wybranych słów.

```
def compare(word, word2):
    cfd = nltk.ConditionalFreqDist(
              (target, fileid[:4])
              for fileid in inaugural.fileids()
              for w in inaugural.words(fileid)
              for target in [word, word2]
              if w.lower().startswith(target))
    cfd.plot()
```

Przykładowe wyniki: