1. Podziel tekst wszystkich orzeczeń z określonego roku na słowa.

```
2. def tokenize(data):
        for text in data['items']:
            if not reader.in_2008(text):
                continue
            txt = re.sub(r'<[^>]*>', '', text['textContent'])
            txt = re.sub(r'-\n', '', txt)
            found = re.findall(r'\b[a-zA-ZąęłćżźóńśĄĘŁĆŻŹÓŃ]+\b', txt)
            found = [x.lower() for x in found]
            words.extend(found)
```

2. Oblicz statystykę występowania bigramów słów, pomijając w tekście wszystkie wyrazy, które nie stanowią słów.

```
unigrams = Counter(words)
bigrams = get_special_list()
def get_special_list():
    bigrams = []
    for i in range(0, len(words)-1):
        bigrams.append(words[i] + ' ' + words[i+1])
    return Counter(bigrams)
```

3.

Korzystając z wzoru na punktową informację wzajemną oblicz tę wartość dla wszystkich par słów. Wykorzystaj statystyki unigramów obliczone w poprzednim zadaniu.

```
def pmi(word1, word2, unigram_freq, bigram_freq, unigram_sum, bigram_sum):
    prob_word1 = unigram_freq[word1] / unigram_sum
    prob_word2 = unigram_freq[word2] / unigram_sum
    prob_word1_word2 = bigram_freq[" ".join([word1, word2])] / bigram_sum
    return math.log(prob_word1_word2/float(prob_word1*prob_word2), 2)


def calculate_pmi(unigrams, bigrams):
    d = {}
    unigram_sum = float(sum(unigrams.values()))
    bigram_sum = float(sum(bigrams.values()))
    for bigram in bigrams.keys():
        word1, word2 = bigram.split(' ')
        bigram_pmi = pmi(word1, word2, unigrams, bigrams, unigram_sum,
bigram_sum)
        d[bigram] = bigram_pmi
    return d
```

5.

odważne niepokorne, dystrybuującej kolektory, kolektory słoneczne, żonglera komika, komika cyrkowego, sociale verzekeringsbank, verzekeringsbank amsterdam, istituto nazionale, nazionale della, della providenza, providenza soziale, soziale inps, cordero alonso, firan obrusów, niezafałszowanemu współzawodnictwu, klaster niezawodnościowy, składkowo świadczeniowy, swarzewo władysławowo, ospr ipr, geraci responsabilita, responsabilita penale, gare automobilistiche, tajwanie taipei, taipei hsin, hsin tien, hesslewood country, hessle east, east yorkshire, yorkshire hu, hu opw, tuby rozgłoszeniowej

6. Korzystając z wzoru na statystykę logarytmiczną opartą o rozkład dwumienny (G2) sporządź analogiczną listę, jak dla punktowej informacji wzajemnej.

```
def H(k):
    N = np.sum(k)
    return np.sum((k/N)*np.log(k/N))


def llr(k):
    return 2*np.sum(k)*(H(k) - H(k.sum(axis=1) - H(k.sum(axis=0))))


def calculate_llr(bigrams, unigrams, bigrams_sum):
    d = {}
    k = np.array([[0, 0], [0, 0]])
    for bigram in bigrams:
        w1, w2 = bigram.split(' ')
        k[0][0] = bigrams[bigram]
        k[0][1] = unigrams[w2]*2 - k[0][0]
        k[1][0] = unigrams[w1]*2 - k[0][0]
        k[1][1] = bigrams_sum - k[0][0] - k[0][1] - k[1][0]
        d[bigram] = llr(k)
    return d
```

7.

odważne niepokorne, dystrybuującej kolektory, kolektory słoneczne, żonglera komika, komika cyrkowego, sociale verzekeringsbank, verzekeringsbank amsterdam, istituto nazionale, nazionale della, della providenza, providenza soziale, soziale inps, cordero alonso, firan obrusów, niezafałszowanemu współzawodnictwu, klaster niezawodnościowy, składkowo świadczeniowy, swarzewo władysławowo, ospr ipr, geraci responsabilita, responsabilita penale, gare automobilistiche, tajwanie taipei, taipei hsin, hsin tien, hesslewood country, hessle east, east yorkshire, yorkshire hu, hu opw, tuby rozgłoszeniowej

8.

W wynikach znajdują się jak np. east yorkshire, kolektory słoneczne. Część wyników jest dosyć dziwna jak np. „cordero Alonso”. Słowa te są ze sobą mocno powiązane. Są też takie jak np. „firan obrusów”, którego słowa raczej nie są ze sobą mocno związane. W obu przypadkach dostałem ten sam wynik.