# EDA

May 18, 2021

[1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
```

[2]:
```
religion_unlab = pd.read_csv('AllBooks_baseline_DTM_Unlabelled.csv')
religion_lab = pd.read_csv('AllBooks_baseline_DTM_Labelled.csv')

print(religion_lab.shape)
religion_lab.head()
```

```
(590, 8267)
```

[2]:

| | Unnamed: 0 | foolishness | hath | wholesome | takest | feelings | anger | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | Buddhism_Ch1 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 1 | Buddhism_Ch2 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 2 | Buddhism_Ch3 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 3 | Buddhism_Ch4 | 0 | 0 | 0 | 0 | 0 | 0 | |
| 4 | Buddhism_Ch5 | 0 | 0 | 0 | 0 | 0 | 0 | |

| | vaivaswata | matrix | kindled | … | erred | thinkest | modern | reigned | \ |
|---|---|---|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | |
| 1 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | |
| 2 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | |
| 3 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | |
| 4 | 0 | 0 | 0 | … | 0 | 0 | 0 | 0 | |

| | sparingly | visual | thoughts | illumines | attire | explains |
|---|---|---|---|---|---|---|
| 0 | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0 | 0 | 0 | 0 | 0 | 0 |

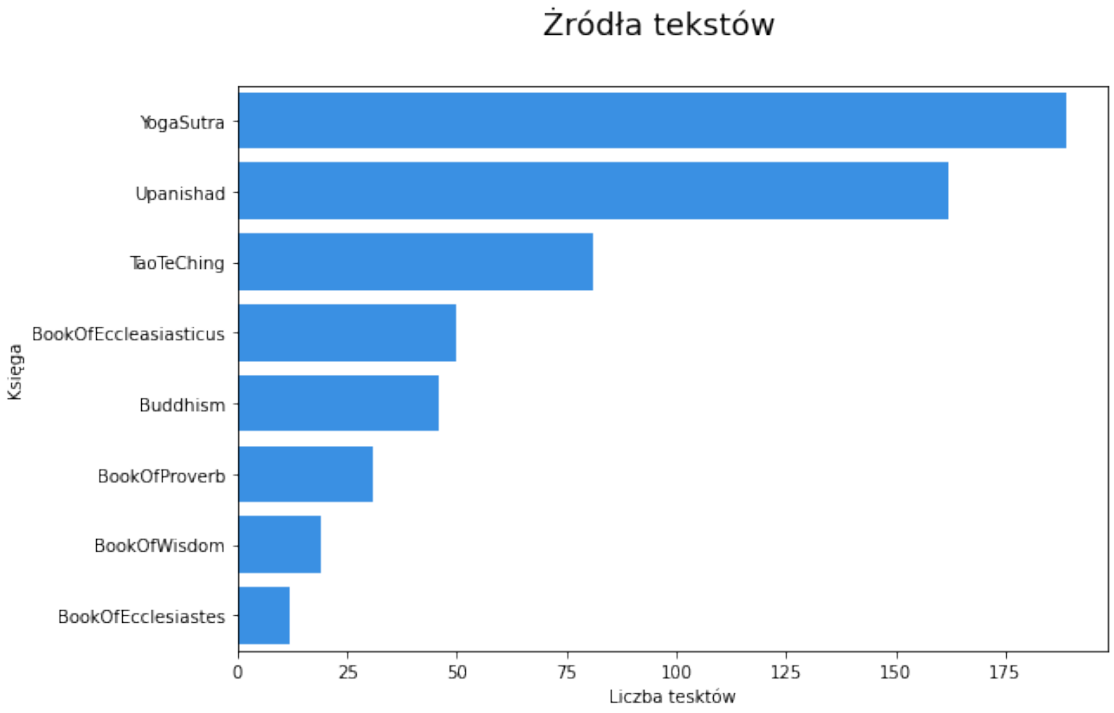
[5 rows x 8267 columns]

[3]:
```
religion_unlab.info()
```

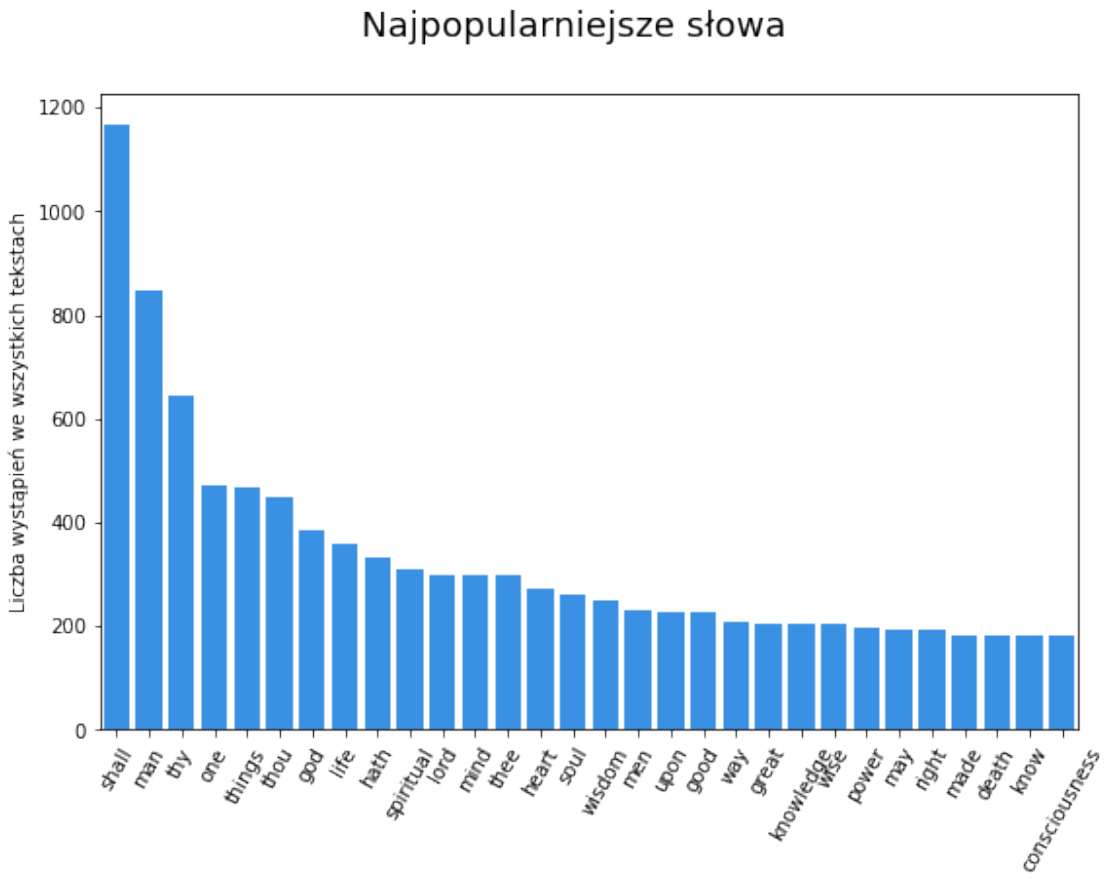
```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 590 entries, 0 to 589
Columns: 8266 entries, # foolishness to explains
dtypes: float64(8266)
memory usage: 37.2 MB
```

[7]:
```
religion_lab['label'] = religion_lab['Unnamed: 0'].apply(lambda x: x.
 ↪split('_')[0])

#for i in range(len(religion_lab)):
#    religion_lab['label'][i] = religion_lab['Unnamed: 0'][i].split('_')[0]


labels = religion_lab['label'].value_counts()
labels = pd.DataFrame(labels).reset_index()
labels.columns = ['label', 'count']
```

[5]:
```
fig = plt.figure(figsize = (9, 6))
sns.barplot(data = labels, y = 'label', x = 'count', color = 'dodgerblue')
fig.suptitle('Źródła tekstów', fontsize=18)
plt.xlabel('Liczba tesktów')
plt.ylabel("Księga")
plt.show()
```

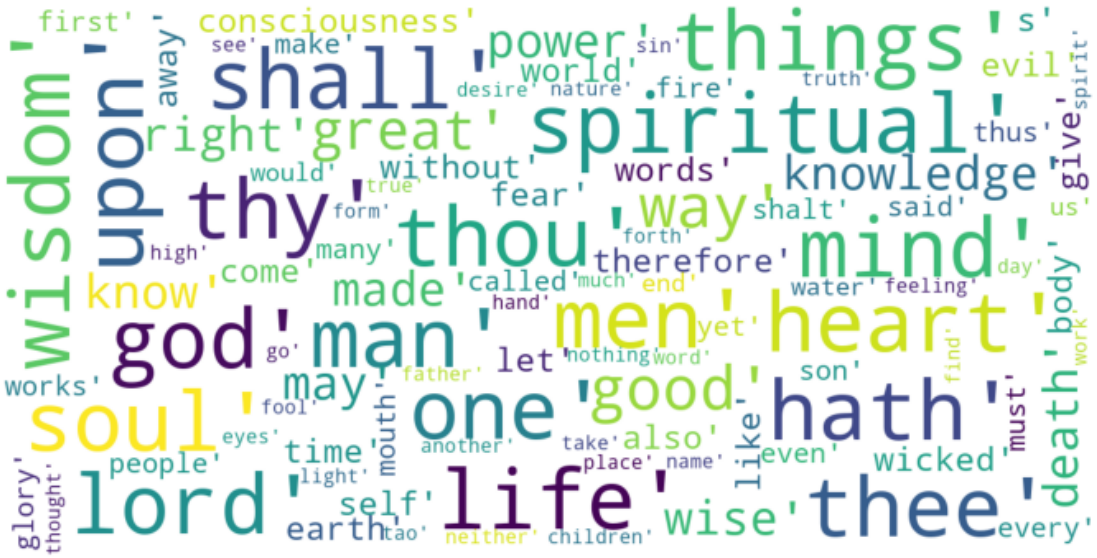


[6]:
```
most_pop_words = religion_unlab.sum().sort_values(ascending=False).head(30)
most_pop_words = pd.DataFrame(most_pop_words).reset_index()
most_pop_words.columns = ['word', 'count']

fig = plt.figure(figsize = (9, 6))
sns.barplot(data = most_pop_words, x = 'word', y = 'count', color =␣
↪'dodgerblue')
plt.xticks(rotation=60)
fig.suptitle('Najpopularniejsze słowa', fontsize=18)
plt.xlabel('')
plt.ylabel('Liczba wystąpień we wszystkich tekstach')
plt.show()
```



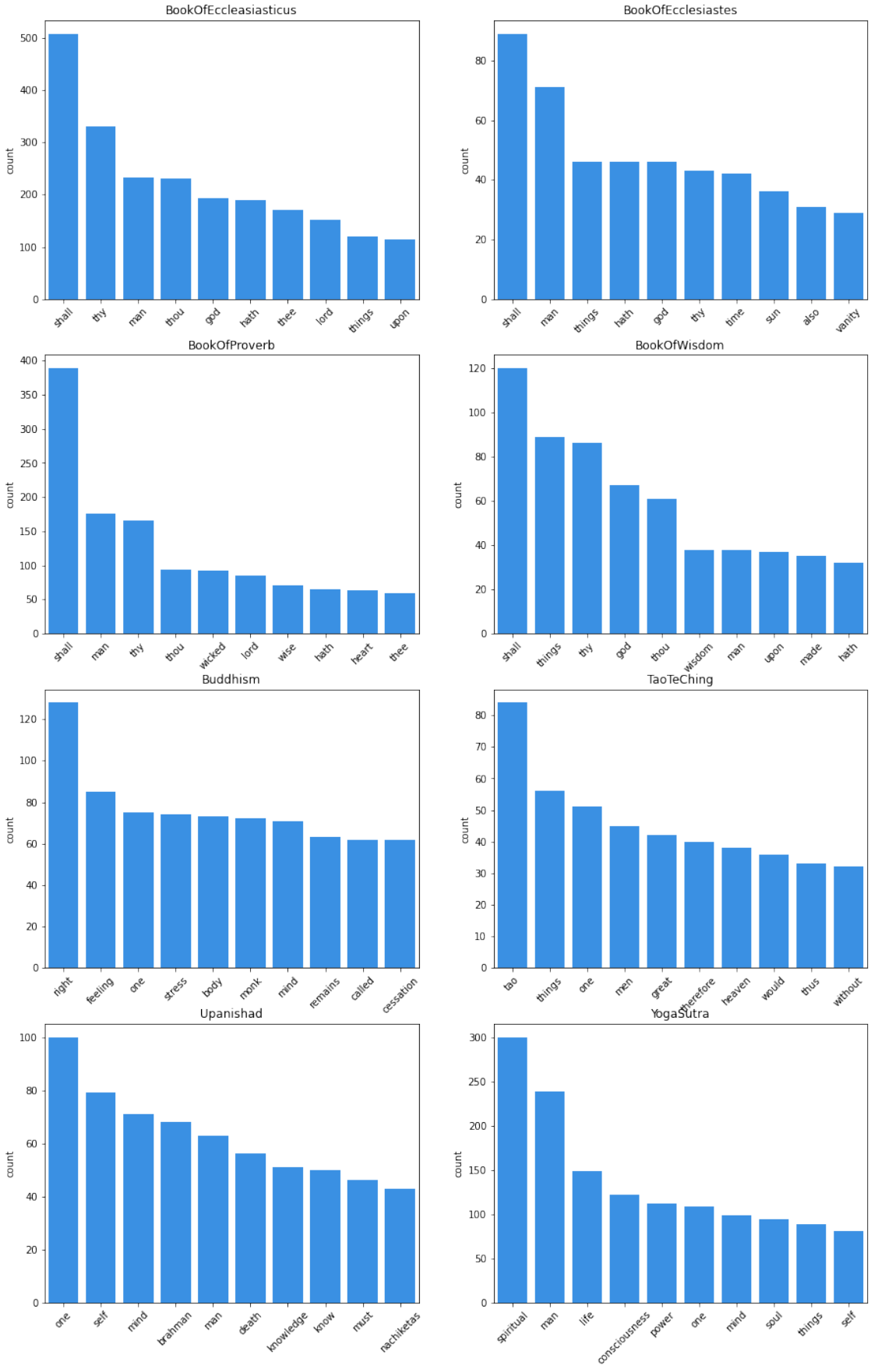

[7]:
```
from wordcloud import WordCloud, STOPWORDS
stopwords = set(STOPWORDS)

def show_wordcloud(data):
    wordcloud = WordCloud(
        background_color='white',
```

```
        stopwords=stopwords,
        max_words=100,
        max_font_size=30,
        scale=3,
        random_state=1)

    wordcloud=wordcloud.generate(str(data))

    fig = plt.figure(1, figsize=(12, 12))
    plt.axis('off')

    plt.imshow(wordcloud)
    plt.show()

show_wordcloud(religion_unlab.sum().sort_values(ascending=False).to_dict())
```




[8]:
```
religion_lab_stacked = religion_lab.drop('Unnamed: 0', axis = 1).
↪groupby('label').sum().transpose()

fig, ax = plt.subplots(4, 2, figsize = (15, 24))
i = 0

for c in religion_lab_stacked.columns:
    df = pd.DataFrame(religion_lab_stacked[c].sort_values(ascending = False)).
↪reset_index().head(10)
    df.columns = ['word', 'count']
    sns.barplot(data = df, x = 'word', y = 'count', color = 'dodgerblue', ax =␣
↪ax[int(np.floor(i/2)%4), int(i%2)])
```

```
    ax[int(np.floor(i/2)%4), int(i%2)].set_title(c)
    ax[int(np.floor(i/2)%4), int(i%2)].set_xlabel('')
    ax[int(np.floor(i/2)%4), int(i%2)].set_xticklabels(ax[int(np.floor(i/2)%4),␣
↪int(i%2)].get_xticklabels(), rotation=45)


    i+=1
```


BookOfEccleasiasticus
count
shall thy man thou god hath thee lord things upon
BookOfEcclesiastes
count
shall man things hath god thy time sun also vanity
BookOfProverb
count
shall man thy thou wicked lord wise hath heart thee
BookOfWisdom
count
shall things thy god thou wisdom man upon made hath
Buddhism
count
right feeling one stress body monk mind remains called cessation
TaoTeChing
count
tao things one men great therefore heaven would thus without
Upanishad
count
one self mind brahman man death knowledge know must nachiketas
YogaSutra
count
spiritual man life consciousness power one mind soul things self

[9]:
```
import re
from textstat import lexicon_count
from textstat import flesch_reading_ease
from textstat import flesch_kincaid_grade
from textstat import sentence_count
from textstat import lexicon_count
```

[10]:
```
file = open('Complete_data .txt', 'r')
file_content = file.read()
file.close()

content_list = re.split('\d+\.\d+', file_content)

text = []
for i in range(len(content_list)):
    stripped = content_list[i].strip()
    if stripped != '':
        text.append(content_list[i])

substrings_to_drop = ['\n', ' \n', '\n ', '  \n', '§', '§ ']

for i in range(len(corpus)):
    for j in substrings_to_drop:
        corpus[i] = corpus[i].replace(j, '')

data = pd.DataFrame(corpus).reset_index()
data.columns = ['index', 'text']

data.head()
```

[10]:

| | index | text |
|---|---|---|
| 0 | 0 | 1.The Buddha: "What do you think, Rahula: Wha… |
| 1 | 1 | 2.Once the Blessed One was staying at Kosambi… |
| 2 | 2 | 3."'Stress should be known. The cause by whic… |
| 3 | 3 | 4."Vision arose, clear knowing arose, discern… |
| 4 | 4 | 5.Sariputta: "There are these three forms of … |

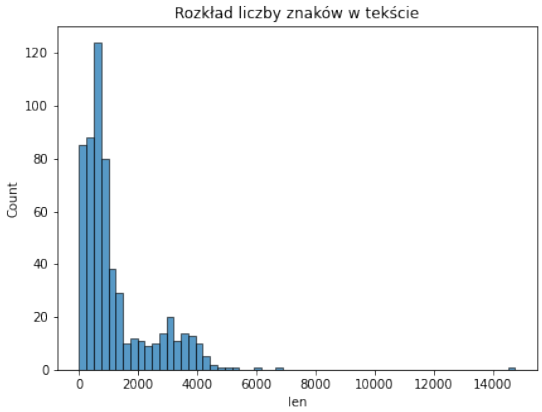
[11]:
```
#characters
data['len'] = data['text'].str.len()
#words
data['words'] = data['text'].apply(lambda x : lexicon_count(x,␣
 ↪removepunct=True))
#average sentence length
data['avg_sen'] = data['text'].str.split().apply(lambda x : [len(i) for i in␣
 ↪x]).map(lambda x: np.mean(x))
```

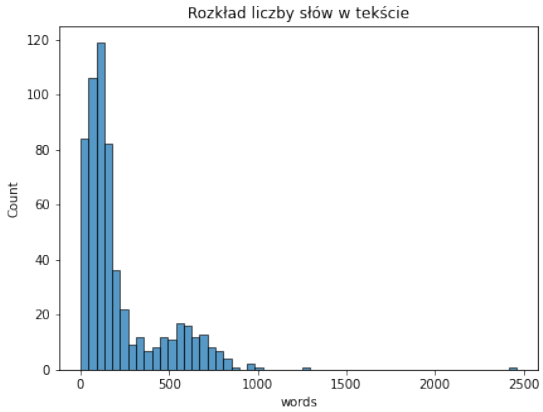
```
#reading ease
data['reading_ease'] = data['text'].apply(lambda x : flesch_reading_ease(x))
#flesch_kincaid_grade
data['grade'] = data['text'].apply(lambda x : flesch_kincaid_grade(x))
#sentences
data['sentences'] = data['text'].apply(lambda x : sentence_count(x))
```

[12]:
```
data
```

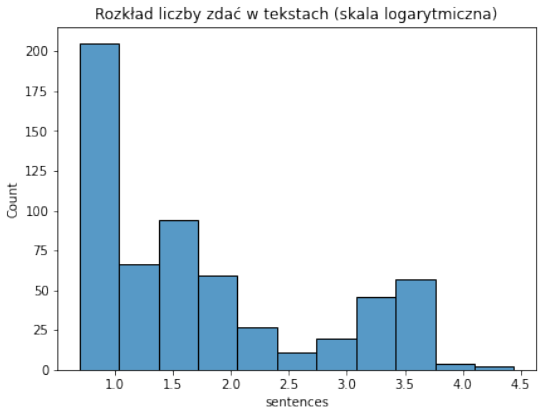
[12]:
```
     index                                                text   len  words  \
0        0   1.The Buddha: "What do you think, Rahula: Wha…  3631    587
1        1   2.Once the Blessed One was staying at Kosambi…  1512    265
2        2   3."'Stress should be known. The cause by whic…  2204    370
3        3   4."Vision arose, clear knowing arose, discern…  1584    267
4        4   5.Sariputta: "There are these three forms of …   216     29
..     …                                                …   …    …
586    586  condemning both the makers and the worshippers…  3014    560
587    587  worthily punished, and were destroyed by a mul…  4661    829
588    588   17:1. For thy judgments, O Lord, are great, a…  3167    551
589    589  intercession, in the sedition on occasion of C…  4357    770
590    590  All creatures obey God's orders for the servic…  3611    637

      avg_sen  reading_ease  grade  sentences
0    5.031561         38.39   16.0         18
1    4.705660         80.01    6.2         16
2    4.877333         71.34    7.5         22
3    4.823529         62.98    8.6         16
4    6.448276         56.76    8.9          2
..        …            …    …         …
586  4.344643         72.19    9.2         23
587  4.583836         72.80    9.0         35
588  4.709619         71.48    9.5         22
589  4.618182         73.21    8.8         33
590  4.632653         69.96   10.1         24

[591 rows x 8 columns]
```

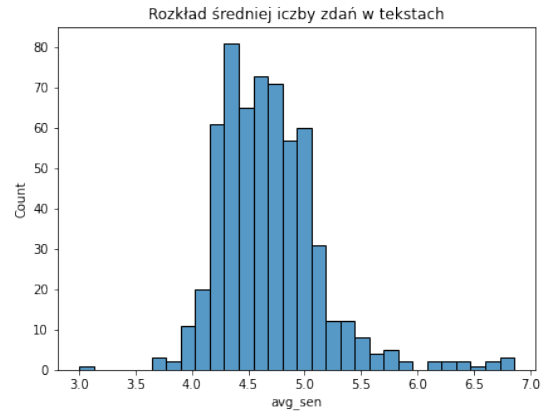
[13]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (15, 5))
sns.histplot(data = data, x = 'len', ax = ax1)
sns.histplot(data['words'], ax = ax2)
ax1.set_title('Rozkład liczby znaków w tekście')
ax2.set_title('Rozkład liczby słów w tekście')
plt.show()
```



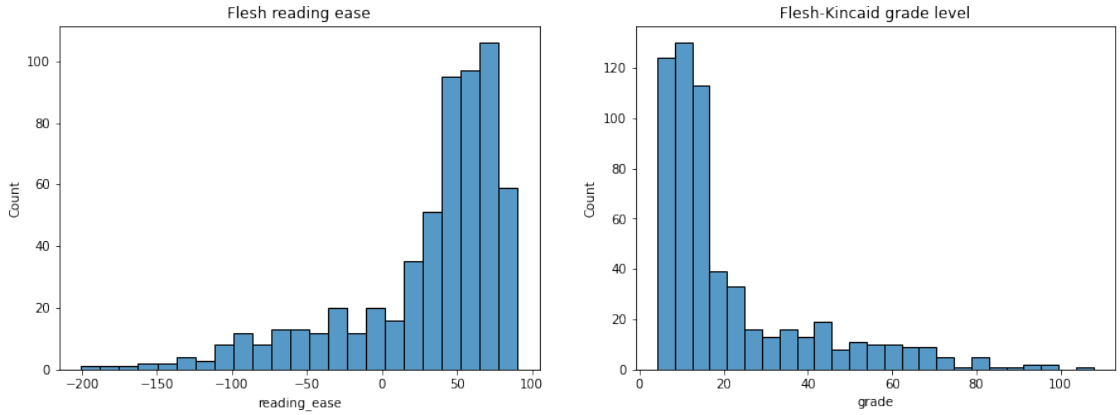




[14]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (15, 5))
sns.histplot(np.log1p(data['sentences']), ax = ax1)
sns.histplot(data = data, x = 'avg_sen', ax = ax2)
ax1.set_title('Rozkład liczby zdać w tekstach (skala logarytmiczna)')
ax2.set_title('Rozkład średniej iczby zdań w tekstach')
plt.show()
```




[15]:
```
fig, (ax1, ax2) = plt.subplots(1, 2, figsize = (15, 5))
sns.histplot(data = data, x = 'reading_ease', ax = ax1)
sns.histplot(data['grade'], ax = ax2)
ax1.set_title('Flesh reading ease')
ax2.set_title('Flesh-Kincaid grade level')
plt.show()
```




## 0.1 Druga część

[2]:
```
df = pd.read_csv("AllBooks_baseline_DTM_Unlabelled.csv")
print(f"Shape of data: {df.shape}.")
```

```
Shape of data: (590, 8266).
```

[3]:
```
df.head()
```

[3]:

| | # foolishness | hath | wholesome | takest | feelings | anger | vaivaswata | \ |
|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

| | matrix | kindled | convict | … | erred | thinkest | modern | reigned | sparingly | \ |
|---|---|---|---|---|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | … | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 1 | 0.0 | 0.0 | 0.0 | … | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 2 | 0.0 | 0.0 | 0.0 | … | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 3 | 0.0 | 0.0 | 0.0 | … | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |
| 4 | 0.0 | 0.0 | 0.0 | … | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 | |

| | visual | thoughts | illumines | attire | explains |
|---|---|---|---|---|---|
| 0 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 1 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 3 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 | 0.0 | 0.0 | 0.0 |

[5 rows x 8266 columns]

Mamy 8266 słów (kolumn), 590 rekordów. Wszystkie wartości są dodatnimi wartościami. Nie mamy braków w danych.

Na początek sprawdźmy czy mamy w naszej ramce danych tzw. skrótowce, czyli słówka typu “don’t”,“aren’t”, isn’t” itp.

[4]:
```
df.rename(columns = {"# foolishness":"foolishness"}, inplace = True)
for i in df.columns:
    if "'" in i: print(i)
```

Wniosek: nie mamy skrótowców, więc możemy pominąć punkt ich rozwijania.

Z naszych słów wyciągnijmy korzeń. Może się zdarzyć, że mamy jednocześnie 2 różne formy tego samego wyrazu, np. ‘play’, ‘playing’, ‘plays’. Dla naszego zadania jest to oczywiście jedno i to samo słowo. Zrobimy to ponownie wykorzystując bibliotekę Spacy.

[ ]:
```
import en_core_web_sm
nlp = en_core_web_sm.load()

listToStr = ' '.join([str(elem) for elem in df.columns])
doc = nlp(listToStr)

i=0
tokenDict = {}
for token in doc:
    if (str(token) != str(token.lemma_)):
        tokenDict[str(token)] = token.lemma_

print(tokenDict)
```

[6]:
```
df.rename(columns = tokenDict, inplace = True)
```

[7]:
```
print(f"Kolumny unikalne: {len(df.columns.unique())}.")
print(f"Wszystkie kolumny: {len(df.columns)}.")
print("Przyklad recznie znaleziony zduplikowanych kolumn")
df["oppose"]
```

```
Kolumny unikalne: 6277.
Wszystkie kolumny: 8266.
Przyklad recznie znaleziony zduplikowanych kolumn
```

[7]:

| | oppose | oppose |
|---|---|---|
| 0 | 0.0 | 0.0 |
| 1 | 0.0 | 0.0 |
| 2 | 0.0 | 0.0 |
| 3 | 0.0 | 0.0 |
| 4 | 0.0 | 0.0 |
| .. | … | … |
| 585 | 0.0 | 0.0 |

```
586     0.0     0.0
587     0.0     0.0
588     0.0     0.0
589     0.0     0.0

[590 rows x 2 columns]
```

Pozbadzmy sie duplikujacych kolumn

[8]:
```
df = df.sum(axis=1, level=0)
print(df.shape)
```

```
(590, 6277)
```

Sprawdźmy teraz czy mamy słówka zaliczane do grupy 'najpopularniejszych słówek języka'. W języku angielskim są to słówka typu "the", "is", "in", "for", "where", "when", "to", "at" etc. Ponownie wykorzystamy bibliotekę spaCy.

[9]:
```
from spacy.lang.en import English
from spacy.lang.en.stop_words import STOP_WORDS


nlp = English()
stopwords = []
for i in df.columns:
    lexeme = nlp.vocab[i]
    if lexeme.is_stop == True: stopwords.append(i)
print(stopwords)
print(len(stopwords))
```

```
['neither', 'something', 'I', 'elsewhere', 'thus', 'give', 'although',
'perhaps', 'well', 'take', 'out', 'keep', 'thence', 'full', 'nowhere', 'name',
'doing', 'move', 'part', 'nine', 'become', 'for', 'side', 'much', 'someone',
'whole', 'show', 'sometimes', 'third', 'still', 'empty', 'say', 'see', 'we',
'either', 'will', 'twelve', 'two', 'hereafter', 'might', 'whither', 'who', 'go',
'seem', 'mine', 'bottom', 'beyond', 'as', 'whatever', 'next', 'do', 'down',
'please', 'never', 'therefore', 'get', 'put', 'upon', 'amount', 'formerly',
'within', 'always', 'could', 'front', 'former', 'though', 'towards', 'once',
'often', 'whose', 'along', 'already', 'make', 'amongst', 'there', 'call',
'whereas', 'whether', 'behind', 'moreover', 'afterwards', 'in', 'anywhere',
'all', 'without', 'however', 'may', 'back', 'enough', 'many', 'five', 'ten',
'anyone', 'ever', 'he', 'last', 'other', 'besides', 'eleven', 'least', 'also',
'throughout', 'less', 'another', 'toward', 'everywhere', 'must', 'anything',
'quite', 'beside', 'hereby', 'almost', 'six', 'thereby', 'nothing', 'alone',
'rather', 'becoming', 'everything', 'top', 'wherever', 'whoever', 'first',
'together', 'wherein', 'this', 'due', 'among', 'namely', 'yet', 'nevertheless',
'beforehand', 'none', 'latter', 'three', 'would', 'eight', 'except', 'several',
'around', 'thereafter', 'even', 'unless', 'the', 'at', 'else', 'one', 'really',
```

```
'being', 'whereby', 'sometime', 'therein', 'hence', 'hundred', 'four', 'such',
'various', 'per', 'just', 'indeed', 'whence', 'otherwise', 'whenever', 'every',
'since', 'everyone']
166
```

[10]:
```
print(f"{len(stopwords)} słów z naszej ramki zostało zklasyfikowane jako słowa␣
↪o niskiej wartości dla całościowego znaczenia tekstu. Spośród ponad 6000␣
↪wszystkich słów, stanowią one niewielki procent więc możemy je usunąć.")
```

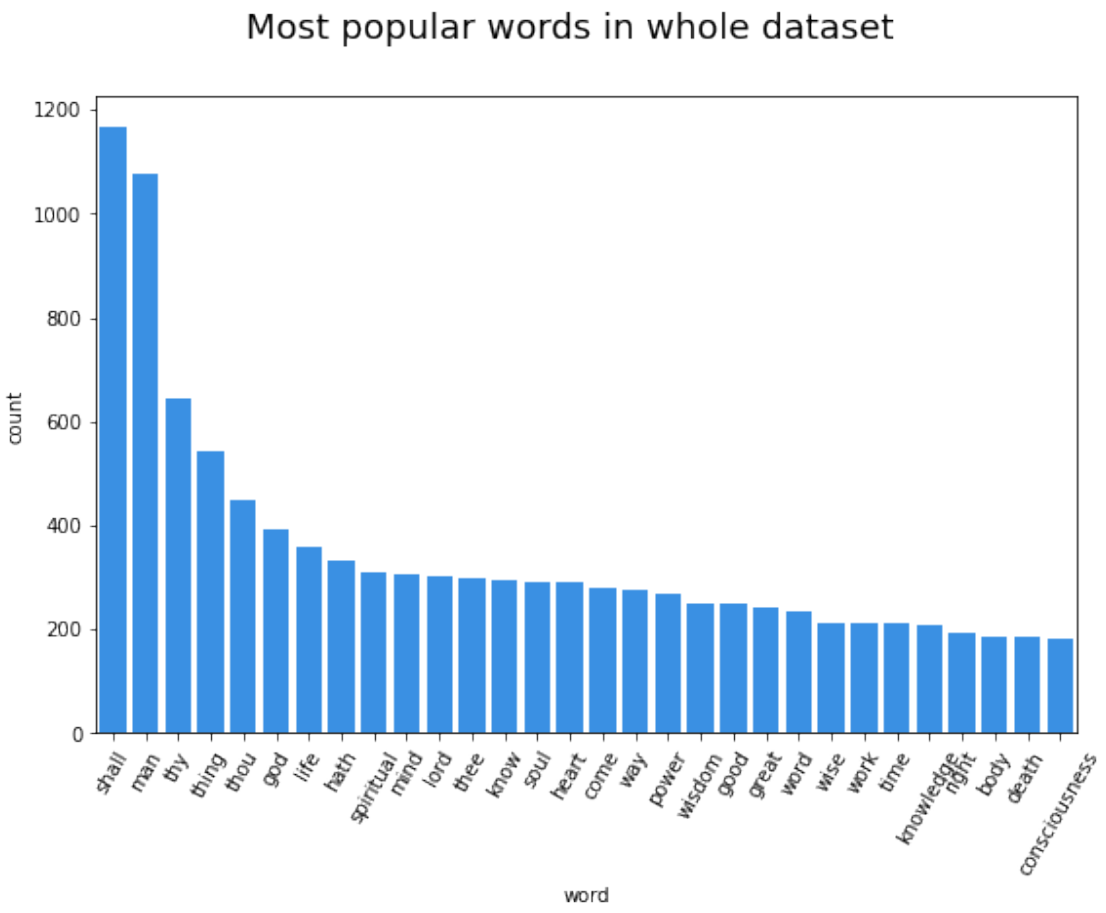
```
166 słów z naszej ramki zostało zklasyfikowane jako słowa o niskiej wartości dla
całościowego znaczenia tekstu. Spośród ponad 6000 wszystkich słów, stanowią one
niewielki procent więc możemy je usunąć.
```

[11]:
```
df = df.drop(columns = stopwords)
print(df.shape)
```

```
(590, 6111)
```

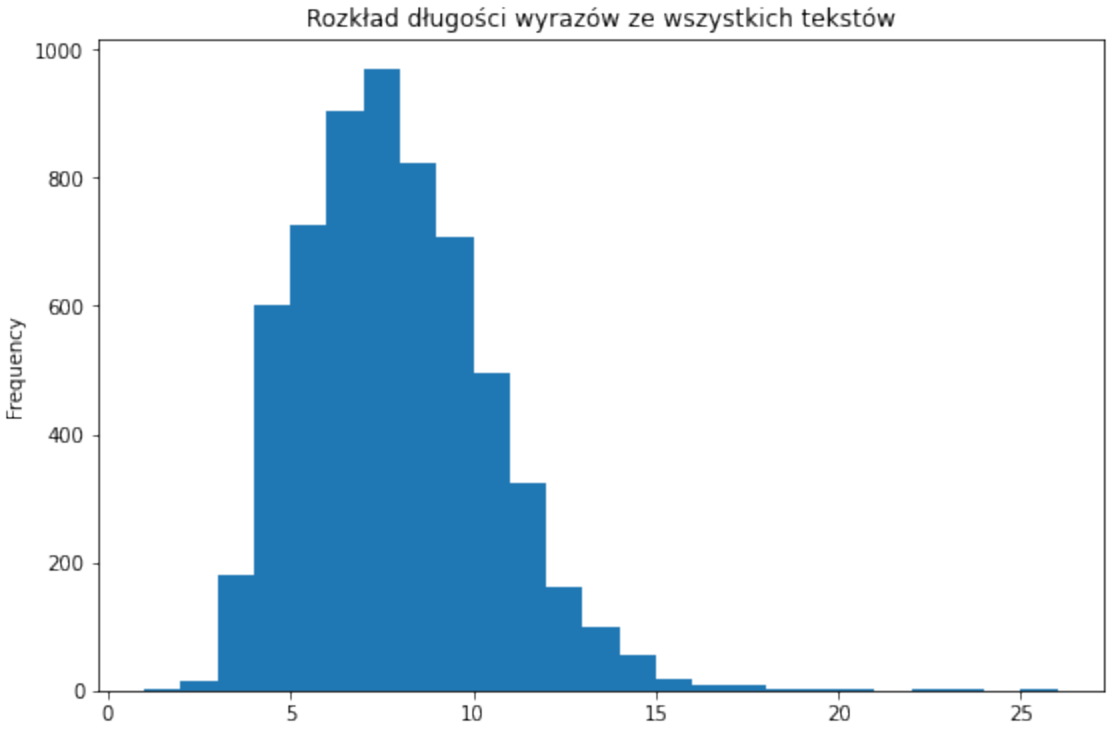
Teraz sprawdźmy najczęściej pojawiające się słówka

[12]:
```
most_pop_words = df.sum().sort_values(ascending=False).head(30)
most_pop_words = pd.DataFrame(most_pop_words).reset_index()
most_pop_words.columns = ['word', 'count']

fig = plt.figure(figsize = (9, 6))
sns.barplot(data = most_pop_words, x = 'word', y = 'count', color =␣
↪'dodgerblue')
plt.xticks(rotation=60)
fig.suptitle('Most popular words in whole dataset', fontsize=18)
plt.show()

print(f"Średnia liczba wystąpień jednego słowa: {np.
↪mean(most_pop_words['count']).round(2)}")
print(f"Odchylenie standardowe liczby wystąpień jednego słowa: {np.
↪std(most_pop_words['count']).round(2)}")
```




```
Średnia liczba wystąpień jednego słowa: 347.87
Odchylenie standardowe liczby wystąpień jednego słowa: 230.82
```

[13]:
```
d = {'word': df.columns}
word_len = pd.DataFrame(data = d)
word_len['nchars'] = word_len['word'].apply(lambda x: len(x))
word_len['occurences'] = word_len['word'].apply(lambda x: df[x].sum())


fig = plt.figure(figsize = (9, 6))
word_len['nchars'].plot(kind = 'hist', title = 'Rozkład długości wyrazów ze␣
 ↪wszystkich tekstów', bins = 25
                         , xlabel = "Liczba liter", ylabel = 'Liczba słów o␣
 ↪danej długości')
plt.show()
print(f"Średnia długość słowa: {np.mean(word_len['nchars']).round(2)}")
print(f"Odchylenie standardowe: {np.std(word_len['nchars']).round(2)}")
```




```
Średnia długość słowa: 7.38
Odchylenie standardowe: 2.59
```

[14]:
```
# Sprawdźmy te słowa, które sa bardzo długie albo krótkie
short_words = word_len.loc[word_len['nchars'] == 2]
short_words = short_words['word'].to_numpy()

long_words = word_len.loc[word_len['nchars'] >= 17]
long_words = long_words['word'].to_numpy()

print(word_len.loc[word_len['nchars'] == 2])
print(word_len.loc[word_len['nchars'] >= 17])
```

```
     word  nchars  occurences
1290   xi       2         1.0
1729   al       2         2.0
1959   iv       2         1.0
2500   ie       2         2.0
2578   ii       2         1.0
2967   st       2         2.0
3392   lo       2         2.0
3654   li       2         1.0
4204   nt       2        11.0
4356   ex       2         1.0
```

```
5185   ye        2        36.0
5348   ox        2         2.0
5982   om        2         2.0
5984   th        2         1.0
6018   em        2         1.0
```

| | word | nchars | occurences |
|---|---|---|---|
| 54 | noseconsciousness | 17 | 2.0 |
| 214 | neitherpainfulnorpleasant | 25 | 6.0 |
| 496 | contradistinction | 17 | 1.0 |
| 984 | clingingsustenance | 18 | 8.0 |
| 1494 | clingingaggregate | 17 | 29.0 |
| 2287 | selfcomprehension | 17 | 1.0 |
| 2705 | consciousnessconsciousness | 26 | 1.0 |
| 2758 | neitherpleasurenorpain | 22 | 2.0 |
| 2985 | bodyconsciousness | 17 | 2.0 |
| 3066 | argumentativethought | 20 | 1.0 |
| 3145 | stressfulsariputta | 18 | 1.0 |
| 3245 | selfconsciousness | 17 | 5.0 |
| 3469 | fabricationverbal | 17 | 1.0 |
| 4509 | fabricationsfabrication | 23 | 1.0 |
| 4717 | consciousnesshood | 17 | 1.0 |
| 4789 | intellectconsciousness | 22 | 2.0 |
| 5106 | neitherpleasantnorpainful | 25 | 3.0 |
| 5666 | lamentationlamentation | 22 | 1.0 |
| 5897 | soulconsciousness | 17 | 1.0 |
| 5952 | tongueconsciousness | 19 | 2.0 |
| 5962 | propertysariputta | 17 | 1.0 |
| 6055 | clingingclingingsustenance | 26 | 1.0 |
| 6058 | fabricationsmental | 18 | 1.0 |

[15]:
```
# Ponieważ liczby wystąpień tych słów są dużo niższe niż odchylenie␣
↪standardowe, również usuniemy je z ramki danych

df = df.drop(columns = (short_words), axis = 1)
df = df.drop(columns = (long_words), axis = 1)
```

[16]:
```
df.shape
```

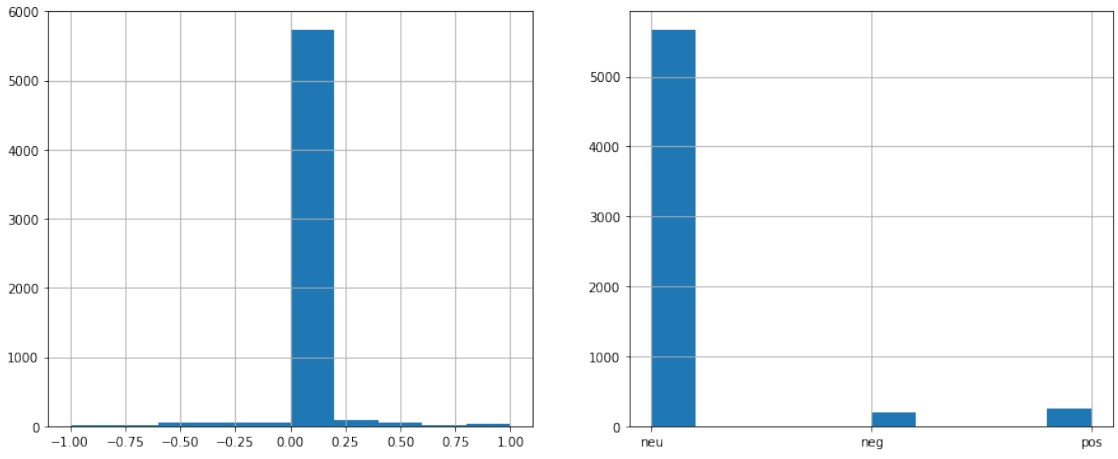
[16]:
```
(590, 6073)
```

Oceńmy teraz czy nasze słowa są nacechowane pozytywnie czy negatywnie.

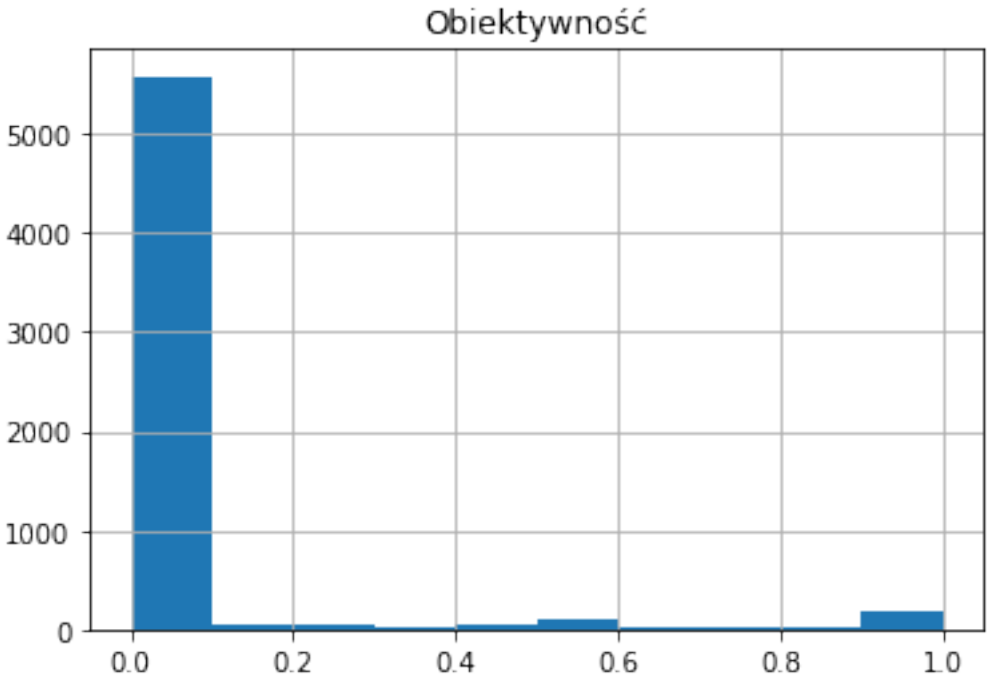
[17]:
```
from textblob import TextBlob

def polarity(text):
    return TextBlob(text).sentiment.polarity

def sentiment(x):
```

```
    if x<0:
        return 'neg'
    elif x==0:
        return 'neu'
    else:
        return 'pos'

def subjectivity(text):
    return TextBlob(text).sentiment.subjectivity

word_len['polarity_score']=word_len['word'].\
   apply(lambda x : polarity(x))

word_len['polarity']=word_len['polarity_score'].\
   map(lambda x: sentiment(x))

word_len['subjectivity']=word_len['word'].\
   map(lambda x: subjectivity(x))

fig, axs = plt.subplots(1, 2, figsize = (15, 6))
print("Nacechowanie emocjonalne słów:")
word_len['polarity_score'].hist(ax = axs[0])
word_len['polarity'].hist(ax = axs[1])

plt.show()

plot_sub = word_len['subjectivity'].hist()
plot_sub.set_title("Obiektywność")
```

```
Nacechowanie emocjonalne słów:
```



[17]: `Text(0.5, 1.0, 'Obiektywność')`




Wniosek: większość słów z naszej bazy ma neutralne nacechowanie emocjonalne. Z pozostałych nielicznych słów, większość jest nacechowana pozytywnie. Słowa są również raczej obiektywne.