Data Analysis

March 24, 2022

1 Klasyfikacja toksycznych komentarzy

1.0.1 *Marcin Drzewiecki, Patryk Świątek, Magdalena Szypulska*

1.1 Motywacja

Na tym etapie projektu zapoznamy się ze zbiorem danych, określimy podstawowe statystyki oraz przygotujemy dane, które wykorzystamy do budowy modelu klasyfikatora.

1.2 Zbiór danych

Zbiór danych pochodzi z *The Wikipedia Corpus*. Zbiór danych zawiera komentarze pochodzących z sekcji edytorskich artykułów na anglojęzycznej *Wikipedii*. W ramce danych znajdziemy 8 kolumn:

- `id` - unikalny ciąg znaków przypisany każdemu komentarzowi,
- `comment_text` - treść komentarza,

Dalsze kolumny będą mieć wartości binarne i będą nam określały, czy dany komentarz jest klasyfikowany jako:

- `toxic`,
- `severe_toxic`,
- `obscene`,
- `threat`,
- `insult`,
- `identity_hate`,

gdzie wartość 1 w danej kolumnie będzie oznaczała, że dany komentarz został otagowany dany typ toksycznego komentarza, w przeciwnym przypadku w kolumnie będzie znajdywała się wartość 0. W sytuacji, kiedy wszystkie wartości są równe 0, komentarz nie jest toksyczny.

[1]:
```
#importowanie pakietów

#pakiety podstawowe w pracy z danymi

import pandas as pd
import numpy as np
from os import path
import scipy.stats as stats

#pakiety do stworzenia wizualizacji
```

```
import matplotlib.pyplot as plt
import seaborn as sns
from PIL import Image
from wordcloud import WordCloud, STOPWORDS, ImageColorGenerator
```

[2]:
```
#from google.colab import files
#uploaded = files.upload()
```

[3]:
```
train = pd.read_csv('train.csv') #wczytanie danych
train_values = train.values #wartości do dalszej pracy
columns = np.array(train.columns) #nazwy kolumn
print('                                        Podstawowe informacje\n')
print('Nazwy kolumn:', columns)
print('Liczba rekordów w zbiorze treningowym to:', train.shape[0])
print('Liczba wartości brakujących w zbiorze treningowym to', train.isnull().
 ↪values.any().sum())
print('Liczba unikalnych wartości w kolumnie id:', train['id'].nunique())
print('Typy danych znajdujących się w kolumnach to:\n\n', train.dtypes)
```

```
                                        Podstawowe informacje

Nazwy kolumn: ['id' 'comment_text' 'toxic' 'severe_toxic' 'obscene' 'threat'
'insult'
 'identity_hate']
Liczba rekordów w zbiorze treningowym to: 159571
Liczba wartości brakujących w zbiorze treningowym to 0
Liczba unikalnych wartości w kolumnie id: 159571
Typy danych znajdujących się w kolumnach to:

 id               object
comment_text     object
toxic             int64
severe_toxic      int64
obscene           int64
threat            int64
insult            int64
identity_hate     int64
dtype: object
```

[4]:
```
train.head()
```

[4]:

	id	comment_text	toxic	\
0	0000997932d777bf	Explanation\nWhy the edits made under my usern…	0	
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s…	0	
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It…	0	
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on …	0	

```
4  0001d958c54c6e35  You, sir, are my hero. Any chance you remember…      0

   severe_toxic  obscene  threat  insult  identity_hate
0             0        0       0       0              0
1             0        0       0       0              0
2             0        0       0       0              0
3             0        0       0       0              0
4             0        0       0       0              0
```

[5]:
```
train.sample(7) #wybranie próby o liczności 7 z naszego zbioru danych
```

[5]:
```
                      id                                       comment_text  \
61986   a5ddcbe8225743aa  That is your opinion, and possibly an assertio…
141222  f394966a9aa4f60b  " August 2013 (UTC)\n Dank u wel! I didn't mak…
12318   20a6b1d69402d099  A new Oxbridge user box \n\nLord Charlton…I …
28936   4cac1e0f50786a33  Just call them, Unli-Night or Unli-Day, beside…
110490  4f2199c98953ad91  "\nPlease explain to me how anything in WP:SPA…
86195   e68988a6859e9549  Telos \n\nI have added Telos to the home plane…
93417   f9c3bd427da986d0  Obviously, JamesMLane and Dogru think a Critic…

        toxic  severe_toxic  obscene  threat  insult  identity_hate
61986       0             0        0       0       0              0
141222      0             0        0       0       0              0
12318       0             0        0       0       0              0
28936       0             0        0       0       0              0
110490      0             0        0       0       0              0
86195       0             0        0       0       0              0
93417       0             0        0       0       0              0
```

Sprawdźmy podstawowe statystyki dotyczące kolumn mających typ numeryczny za pomocą funkcji *describe*.

[6]:
```
train.describe()
```

[6]:
```
                toxic   severe_toxic        obscene         threat  \
count  159571.000000  159571.000000  159571.000000  159571.000000
mean        0.095844       0.009996       0.052948       0.002996
std         0.294379       0.099477       0.223931       0.054650
min         0.000000       0.000000       0.000000       0.000000
25%         0.000000       0.000000       0.000000       0.000000
50%         0.000000       0.000000       0.000000       0.000000
75%         0.000000       0.000000       0.000000       0.000000
max         1.000000       1.000000       1.000000       1.000000

              insult  identity_hate
count  159571.000000  159571.000000
mean        0.049364       0.008805
```

```
std          0.216627       0.093420
min          0.000000       0.000000
25%          0.000000       0.000000
50%          0.000000       0.000000
75%          0.000000       0.000000
max          1.000000       1.000000
```

Patrząc na średnią, która jest bardzo mała we wszystkich kategoriach, możemy mieć przypuszczenie, że większość danych w naszej bazie nie jest toksyczna (tj. średnia to suma 1 w danej kolumnie przez ilość wszystkich komentarzy, jeśli jest ona mała to znaczy, że proporcja 1 do ilości 0 i 1 jest również mała).

[42]:
```
#sprawdzenie hipotezy o tym, że większość danych zawartych w naszej bazie jest␣
 ↪nietoksyczna tj. wartości kolumn toxic, severe_toxic itd.
#są równe 0.

#W tym celu sprawdzimy, które z wierszy naszej ramki danych w kolumnach 2:, w␣
 ↪tym celu użyjemy funkcji sum z argumentem axis = 1, dzięki
#której będziemy zliczać względem wierszy. Jeśli suma w danym wierszu w␣
 ↪kolumnach 2: jest równa 0, to komentarz nie został otagowany jako
#żaden z typów toksycznych komentarzy.

print('Procent komentarzy, które nie są toksyczne to',
      (np.sum(train_values[:,2:], axis=1)==0).sum()/train.shape[0]*100)
```

```
Procent komentarzy, które nie są toksyczne to 89.83211235124176
```

Przykładowy tekst toksycznego komentarza:

[8]:
```
train[np.sum(train_values[:,2:], axis=1)>0]['comment_text'].sample(1)
```

[8]:
```
148650    "\n Look you moron, if the 1000 series is T-10…
Name: comment_text, dtype: object
```

Przykładowy tekst nietoksycznego komentarza:

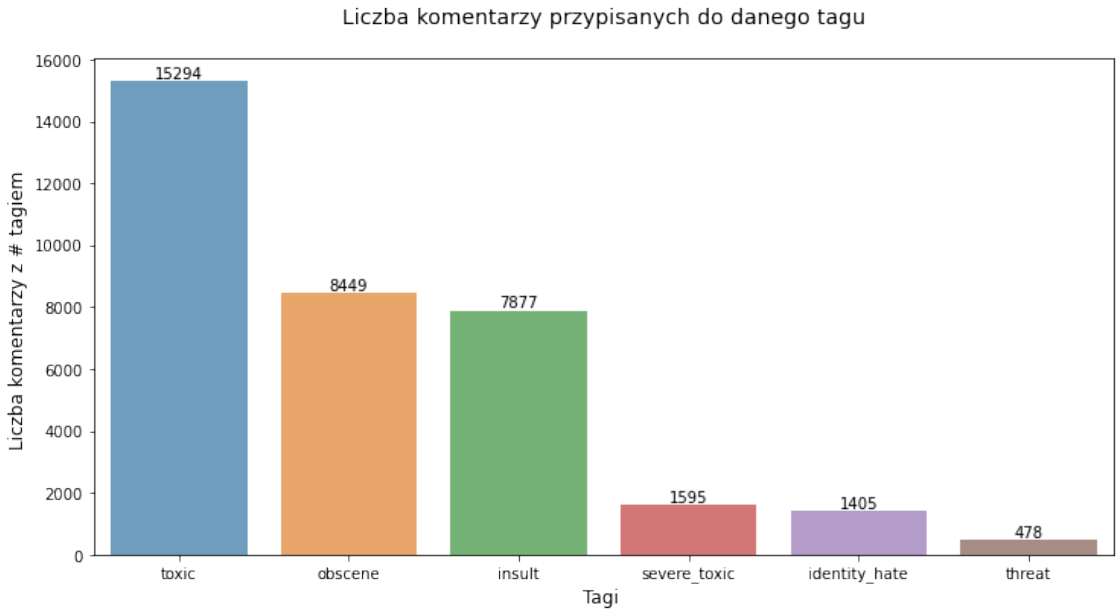
[9]:
```
train[np.sum(train_values[:,2:], axis=1)==0]['comment_text'].sample(1)
```

[9]:
```
159372    "\nThanks Black Kite. Talk "
Name: comment_text, dtype: object
```

Zauważmy, że w przypadku ~90% komentarzy nietoksycznych mamy do czynienia ze sporym niezbalansowaniem. Przedstawmy teraz liczbę komentarzy przypisanych do danej klasy za pomocą wykresu słupkowego.

[10]:
```
#Generowanie wykresu
x = train.iloc[:,2:].sum().sort_values(ascending = False) #przesumujmy liczbę 1␣
 ↪w kolumnach 2:
```

```
#wykres
plt.figure(figsize = (12,6))
ax = sns.barplot(x = x.index, y = x.values, alpha = 0.7)
plt.title('Liczba komentarzy przypisanych do danego tagu\n', fontsize = 14)
plt.xlabel('Tagi ', fontsize = 12)
plt.ylabel('Liczba komentarzy z # tagiem ', fontsize=12)
#dodanie dokładnej liczby
bars = ax.patches #pobieramy nasze słupki
labels = x.values
for bar, label in zip(bars, labels): #zip agreguje w tuple

    ax.text(bar.get_x() + bar.get_width()/2, bar.get_height() + 5, label,␣
 ↪ha='center', va='bottom')
    #dopisujemy tekst - (x + 1/2 szerokości słupka, wysokość słupka + 5),␣
 ↪label, który będzie wartością naszego słupka;
    #położenie poziome - horizontal - wyśrodkowowane, i tekst wertykalnie na␣
 ↪dole

plt.show()
```



Po wartościach możemy zauważyć, że mamy do czynienia z tym, że dany toksyczny komentarz może być otagowany więcej niż jednym tagiem. Ponadto zauważmy, że również klasy są między sobą niezbalansowane.

[84]:
```
#Podsumujmy:
print('Liczba komentarzy: ', len(train))
```

```
print('Liczba nietoksycznych komentarzy: ', (np.sum(train_values[:,2:],␣
 ↪axis=1)==0).sum())
print('Liczba toksycznych komentarzy: ', len(train)-(np.sum(train_values[:,2:],␣
 ↪axis=1)==0).sum())
print('Całkowita liczba tagów (tj. wartości w kolumnach 2: = 1):',train.iloc[:
 ↪,2:].sum().sum())
#stąd wniosek o tym, że toksyczne komentarze są otagowane nietylko jednym␣
 ↪tagiem
multitags = train.iloc[:,2:].sum(axis=1).value_counts() #sumujemy wartości w␣
 ↪wierszach, zliczamy ile jest danej wartości
multitags.index = multitags.index.astype(int) #konwertujemy float do int
df = pd.DataFrame(multitags).reset_index()
df.columns = ['liczba tagów', 'liczba toksycznych komentarzy z # ilością tagów']
#print('\n\n'+df.to_markdown())
display(df)
```

```
Liczba komentarzy:  159571
Liczba nietoksycznych komentarzy:  143346
Liczba toksycznych komentarzy:  16225
Całkowita liczba tagów (tj. wartości w kolumnach 2: = 1): 35098
```

	liczba tagów	liczba toksycznych komentarzy z # ilością tagów
0	0	143346
1	1	6360
2	3	4209
3	2	3480
4	4	1760
5	5	385
6	6	31

Mając do czynienia z multitagowanie warto byłoby przyjrzeć, które z tagów są ze sobą powiązane, mając jednak na uwadze, że tagowanie wykonywały różne osoby, co może rzutować na wyniki.

1.2.1 Histogramy długości komentarzy

[12]:
```
train_help = train.copy()
#train_help['length'] = train_help.apply(len(train_help['comment_text']))
train_help['length'] = train_help['comment_text'].apply(len)
train_help
```

[12]:

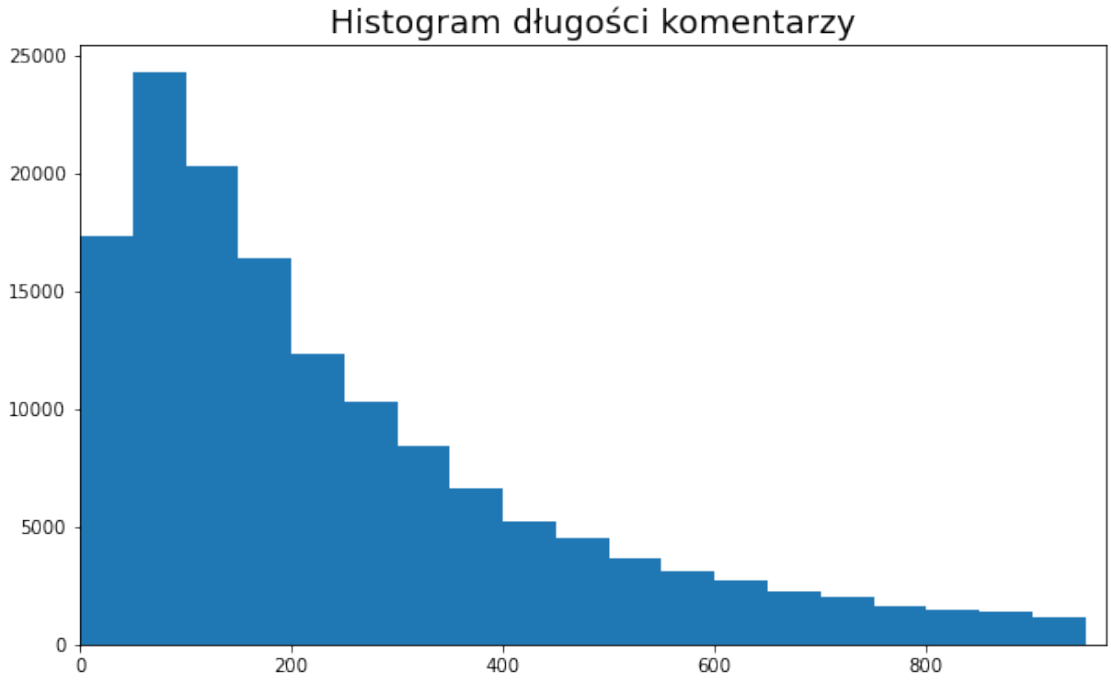
	id	comment_text	\
0	0000997932d777bf	Explanation\nWhy the edits made under my usern…	
1	000103f0d9cfb60f	D'aww! He matches this background colour I'm s…	
2	000113f07ec002fd	Hey man, I'm really not trying to edit war. It…	
3	0001b41b1c6bb37e	"\nMore\nI can't make any real suggestions on …	
4	0001d958c54c6e35	You, sir, are my hero. Any chance you remember…	
…	…	…	

```
159566  ffe987279560d7ff  ":::::And for the second time of asking, when …
159567  ffea4adeee384e90  You should be ashamed of yourself \n\nThat is …
159568  ffee36eab5c267c9  Spitzer \n\nUmm, theres no actual article for …
159569  fff125370e4aaaf3  And it looks like it was actually you who put …
159570  fff46fc426af1f9a  "\nAnd … I really don't think you understand…
```

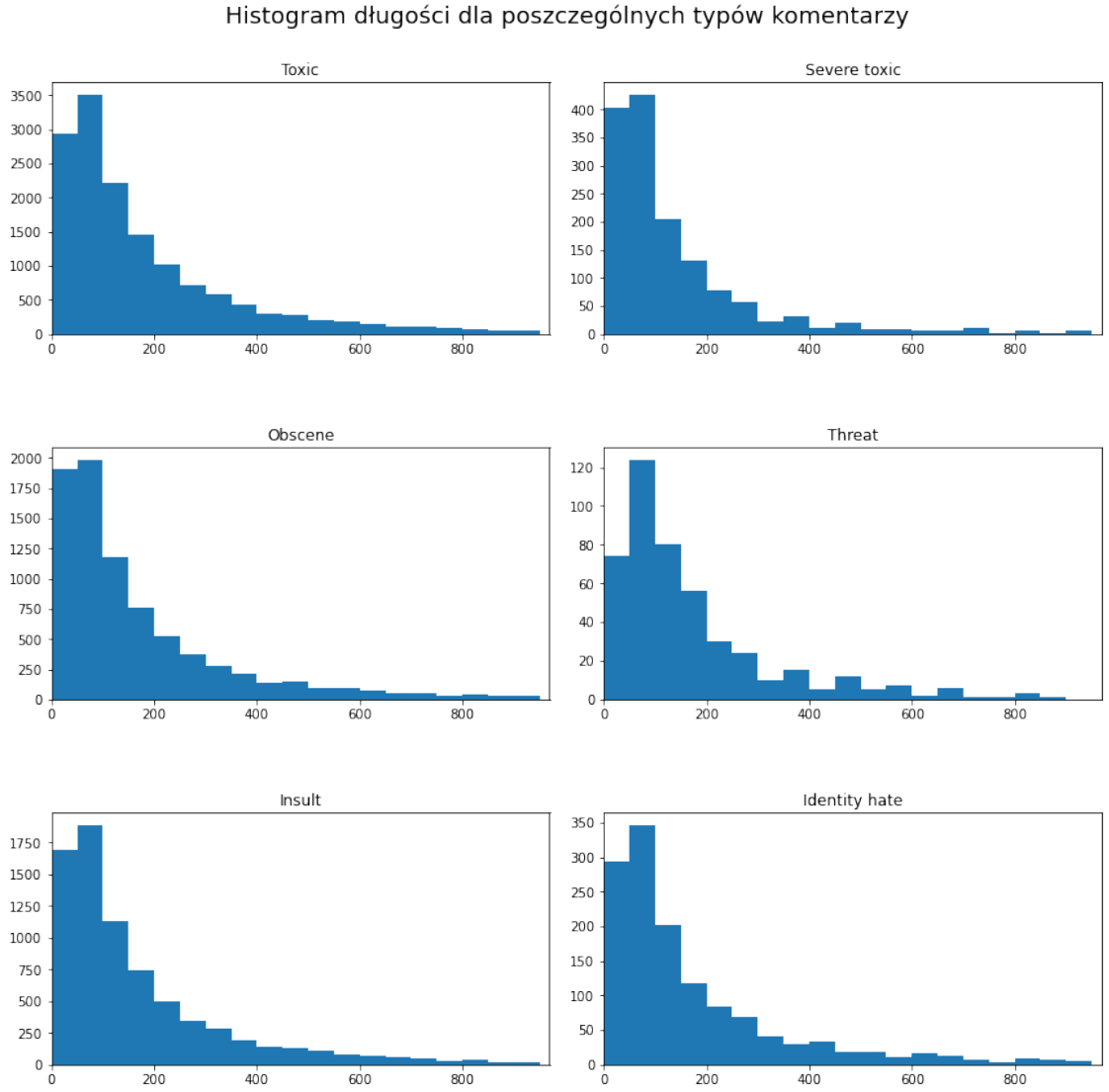
	toxic	severe_toxic	obscene	threat	insult	identity_hate	length
0	0	0	0	0	0	0	264
1	0	0	0	0	0	0	112
2	0	0	0	0	0	0	233
3	0	0	0	0	0	0	622
4	0	0	0	0	0	0	67
…	…	…	…	…	…	…	…
159566	0	0	0	0	0	0	295
159567	0	0	0	0	0	0	99
159568	0	0	0	0	0	0	81
159569	0	0	0	0	0	0	116
159570	0	0	0	0	0	0	189

```
[159571 rows x 9 columns]
```

[85]:
```
plt.figure(figsize=(10,6))
plt.hist(train_help['length'],  bins=np.arange(0,1000,50))
plt.xlim([0, 970])
plt.title("Histogram długości komentarzy", fontsize=18)
plt.show()
```

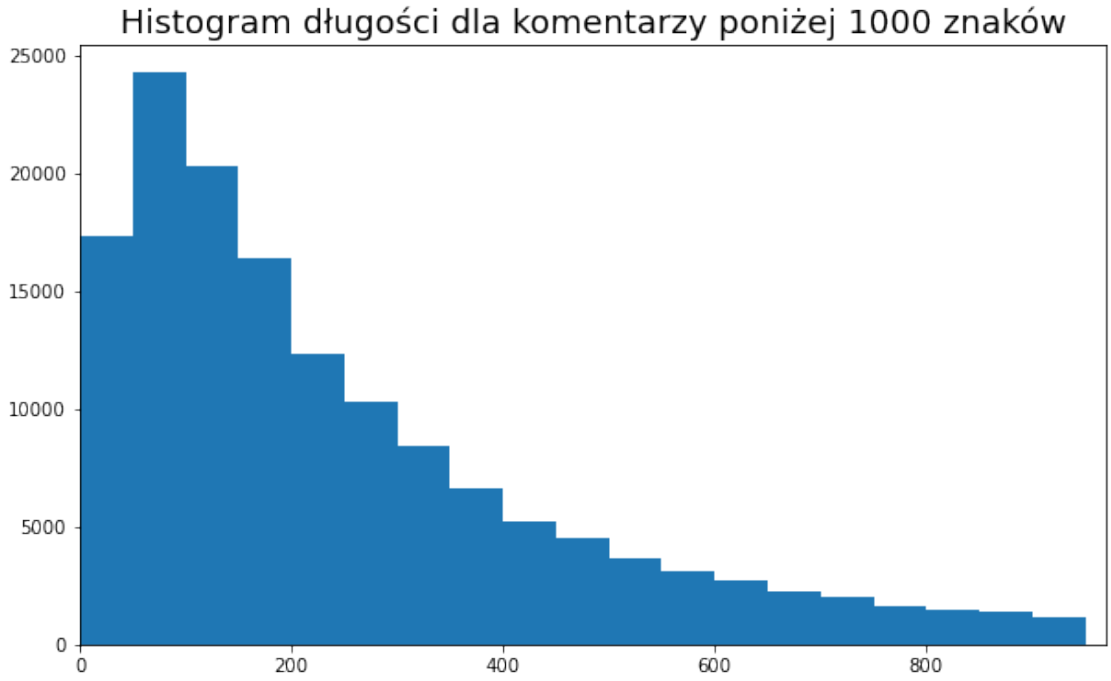


[80]:
```
kbins= 10

fig, axs = plt.subplots(3, 2, figsize=(12,12))
fig.tight_layout()

axs[0, 0].hist(train_help[train_help['toxic']==1]['length'], bins=np.
 ↪arange(0,1000,50))
axs[0, 0].set_title("Toxic", fontsize=12)
axs[0, 0].set_xlim([0, 970])

axs[0, 1].hist(train_help[train_help['severe_toxic']==1]['length'], bins=np.
 ↪arange(0,1000,50))
axs[0, 1].set_title("Severe toxic", fontsize=12)
axs[0, 1].set_xlim([0, 970])

axs[1, 0].hist(train_help[train_help['obscene']==1]['length'], bins=np.
 ↪arange(0,1000,50))
axs[1, 0].set_title("Obscene", fontsize=12)
axs[1, 0].set_xlim([0, 970])

axs[1, 1].hist(train_help[train_help['threat']==1]['length'], bins=np.
 ↪arange(0,1000,50))
axs[1, 1].set_title("Threat", fontsize=12)
axs[1, 1].set_xlim([0, 970])

axs[2, 0].hist(train_help[train_help['insult']==1]['length'], bins=np.
 ↪arange(0,1000,50))
axs[2, 0].set_title("Insult", fontsize=12)
axs[2, 0].set_xlim([0, 970])

axs[2, 1].hist(train_help[train_help['identity_hate']==1]['length'], bins=np.
 ↪arange(0,1000,50))
axs[2, 1].set_title('Identity hate', fontsize=12)
axs[2, 1].set_xlim([0, 970])

fig.suptitle("Histogram długości dla poszczególnych typów komentarzy",␣
 ↪fontsize=18)
fig.subplots_adjust(top=0.91, hspace=0.45)
```



Widzimy, że zdecydowanie najwięcej komentarzy ma długość mniejszą niż 1000 znaków. Zatem w celu bardziej dogłębnego spojrzenia na rozkład długości komentarzy spójrzmy tylko na te o długości nie większej niż 1000 znaków.

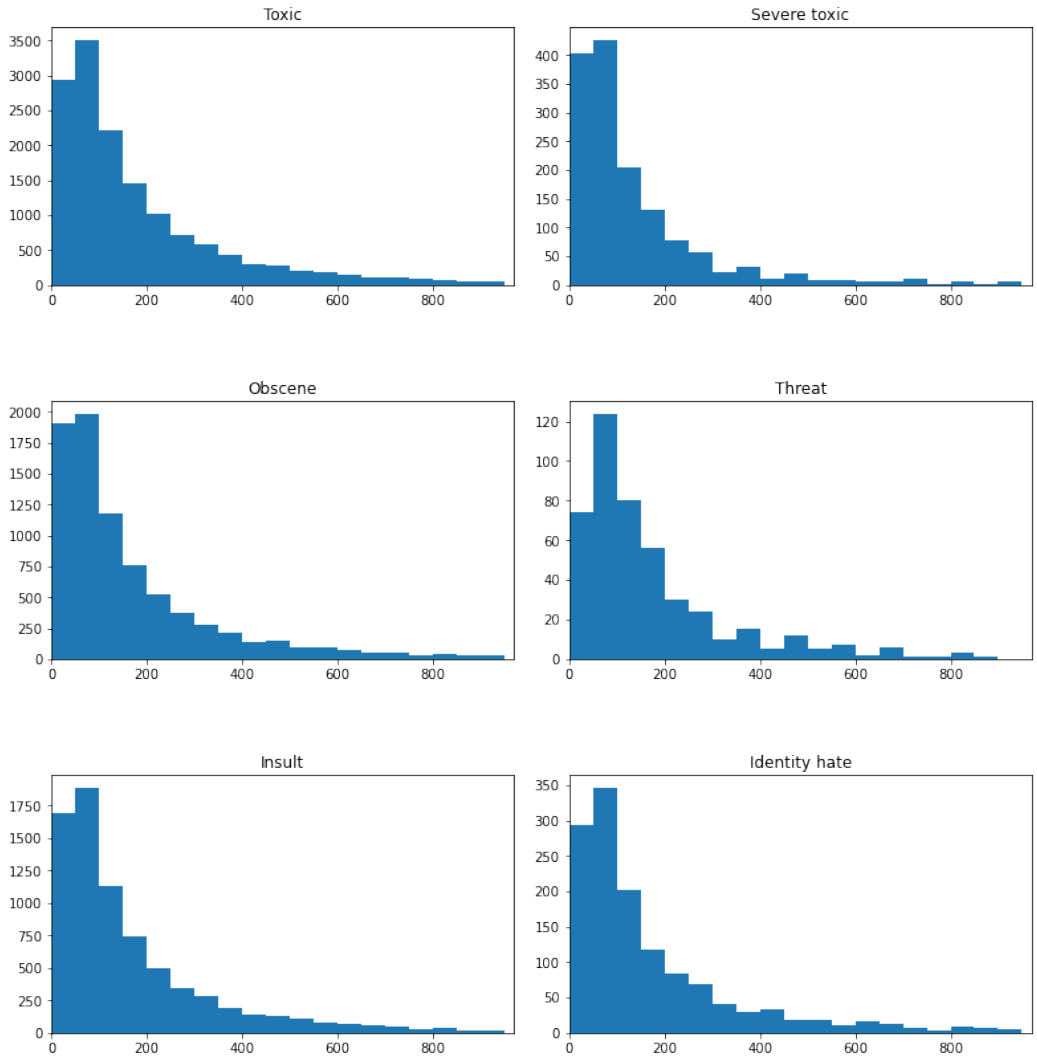
[81]:
```
plt.figure(figsize=(10,6))
plt.hist(train_help[train_help['length']<=1000]['length'], bins=np.
 ↪arange(0,1000,50))#, align='left')
plt.xlim([0, 970])
plt.title('Histogram długości dla komentarzy poniżej 1000 znaków', fontsize=18)
plt.show()
```



[95]:
```
kbins=15

fig, axs = plt.subplots(3, 2, figsize=(11,12))
fig.tight_layout()

axs[0, 0].hist(train_help[(train_help['toxic']==1) &␣
 ↪(train_help['length']<=1000)]['length'], bins=np.arange(0,1000,50))
axs[0, 0].set_title("Toxic", fontsize=12)
axs[0, 0].set_xlim([0, 970])

axs[0, 1].hist(train_help[(train_help['severe_toxic']==1) &␣
 ↪(train_help['length']<=1000)]['length'], bins=np.arange(0,1000,50))
axs[0, 1].set_title("Severe toxic", fontsize=12)
axs[0, 1].set_xlim([0, 970])

axs[1, 0].hist(train_help[(train_help['obscene']==1) &␣
 ↪(train_help['length']<=1000)]['length'], bins=np.arange(0,1000,50))
axs[1, 0].set_title("Obscene", fontsize=12)
axs[1, 0].set_xlim([0, 970])

axs[1, 1].hist(train_help[(train_help['threat']==1) &␣
 ↪(train_help['length']<=1000)]['length'], bins=np.arange(0,1000,50))
axs[1, 1].set_title("Threat", fontsize=12)
axs[1, 1].set_xlim([0, 970])
```

```
axs[2, 0].hist(train_help[(train_help['insult']==1) &␣
 ↪(train_help['length']<=1000)]['length'], bins=np.arange(0,1000,50))
axs[2, 0].set_title("Insult", fontsize=12)
axs[2, 0].set_xlim([0, 970])

axs[2, 1].hist(train_help[(train_help['identity_hate']==1) &␣
 ↪(train_help['length']<=1000)]['length'], bins=np.arange(0,1000,50))
axs[2, 1].set_title('Identity hate', fontsize=12)
axs[2, 1].set_xlim([0, 970])

fig.suptitle("Histogram długości dla poszczególnych typów komentarzy (o␣
 ↪długości poniżej 1000 znaków)", fontsize=18)
fig.subplots_adjust(top=0.91, hspace=0.45)
```

Histogram długości dla poszczególnych typów komentarzy (o długości poniżej 1000 znaków)



[17]:
```
train_corr = train_help[['toxic', 'severe_toxic', 'obscene', 'threat',␣
 ↪'insult', 'identity_hate']].corr(method='pearson')
train_corr
```

[17]:
```
                toxic  severe_toxic   obscene    threat    insult  \
toxic        1.000000      0.308619  0.676515  0.157058  0.647518
severe_toxic 0.308619      1.000000  0.403014  0.123601  0.375807
obscene      0.676515      0.403014  1.000000  0.141179  0.741272
threat       0.157058      0.123601  0.141179  1.000000  0.150022
insult       0.647518      0.375807  0.741272  0.150022  1.000000
identity_hate 0.266009      0.201600  0.286867  0.115128  0.337736

              identity_hate
toxic              0.266009
severe_toxic       0.201600
obscene            0.286867
threat             0.115128
insult             0.337736
identity_hate      1.000000
```

[18]:
```
import plotly.graph_objects as go

fig = go.Figure(data=[go.Table(header=dict(values=np.insert(np.array(train_corr.
 ↪columns), 0 ," ")),
                 cells=dict(values=np.transpose(np.append(np.array(train_corr.
 ↪columns).reshape(6,1),
                                                          np.round(train_corr.
 ↪values,4),axis=1)))])
fig.update_layout(width=900, height=450)
fig.show()
```

1.2.2 Współczynnik V Cramera

Ze względu na to, że zmienne określające typ komentarza to zmienne binarne, nie możemy użyć standardowego współczynnika korelacji Pearsona. Wyznaczymy więc współczynnik **V Cramera**. Określa on poziom zależności między dwiema zmiennymi nominalnymi.

Wzór:

$$\sqrt{\frac{\chi^2}{n \cdot min(c-1, r-1)}}$$

gdzie: * χ^2 - wynik testu zgodności chi-kwadrat * n - łączna liczba obserwacji * k – liczba kolumn w tabeli kontyngencji, * r – liczba wierszy w tabeli kontyngencji

[86]:
```
corr_matrix = np.ones(shape=(6,6))
a = 0
for i in train.columns[2:]:
    b = 0
    for j in train.columns[2:]:
        confusion_matrix = pd.crosstab(train[i], train[j])

        chi2 = stats.chi2_contingency(confusion_matrix, correction=False)[0]
        n = np.sum(confusion_matrix.values)
        minDim = min(confusion_matrix.shape)-1
        cram_V = np.round(np.sqrt((chi2/n) / minDim), 4)

        corr_matrix[a,b] = cram_V
        corr_matrix[b,a] = cram_V
        b += 1
    a += 1
```

[87]:
```
train_cram_cor = pd.DataFrame(corr_matrix, index = train.columns[2:], columns =␣
↪train.columns[2:])
train_cram_cor
```

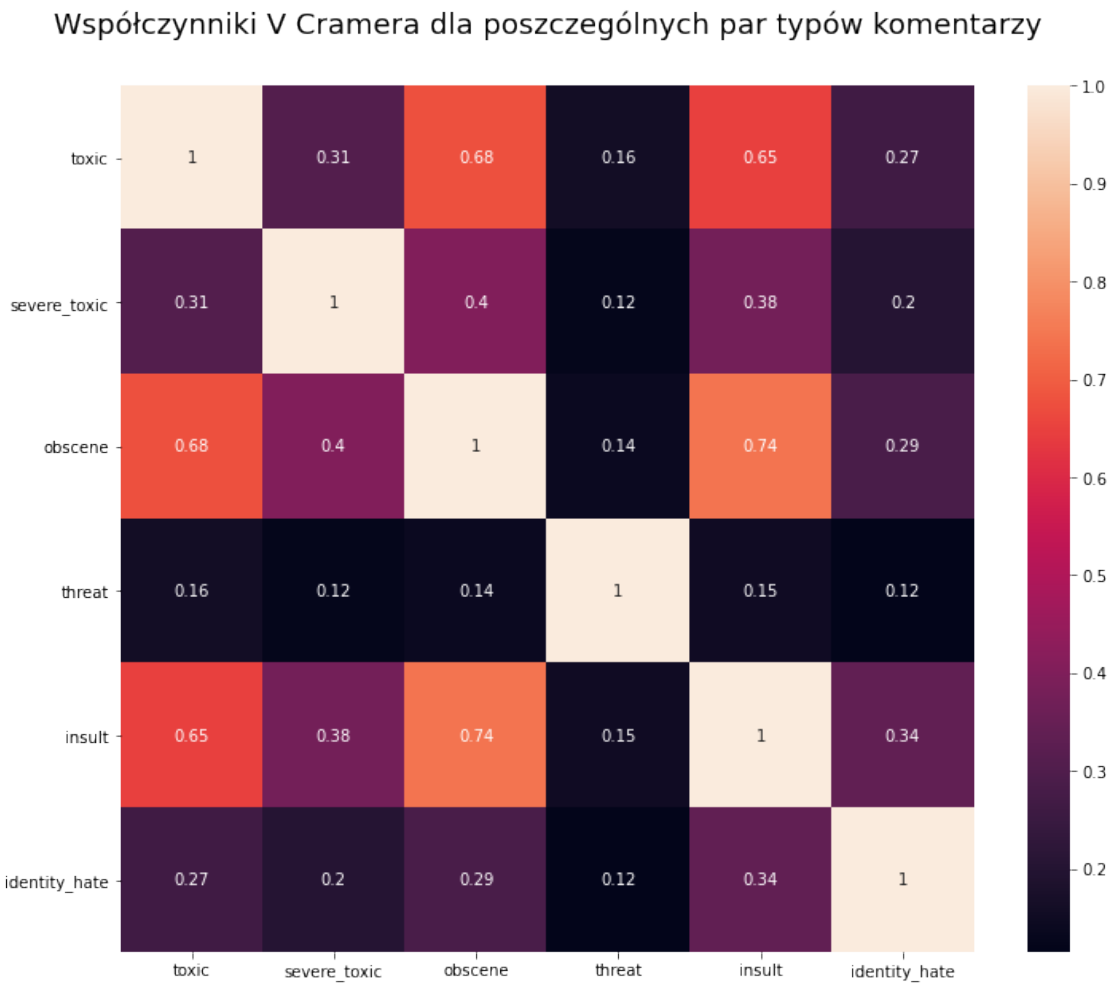
[87]:

	toxic	severe_toxic	obscene	threat	insult	identity_hate
toxic	1.0000	0.3086	0.6765	0.1571	0.6475	0.2660
severe_toxic	0.3086	1.0000	0.4030	0.1236	0.3758	0.2016
obscene	0.6765	0.4030	1.0000	0.1412	0.7413	0.2869
threat	0.1571	0.1236	0.1412	1.0000	0.1500	0.1151
insult	0.6475	0.3758	0.7413	0.1500	1.0000	0.3377
identity_hate	0.2660	0.2016	0.2869	0.1151	0.3377	1.0000

[89]:
```
import plotly.graph_objects as go

fig = go.Figure(data=[go.Table(header=dict(values=np.insert(np.
↪array(train_cram_cor.columns), 0 ," ")),
                 cells=dict(values=np.transpose(np.append(np.
↪array(train_cram_cor.columns).reshape(6,1),
                                                         np.
↪round(train_cram_cor.values,4),axis=1)))])
fig.update_layout(width=900, height=450)
fig.show()
```

[100]:
```
#heatmap

plt.figure(figsize=(12,10))
sns.heatmap(train_cram_cor,
            xticklabels=train_cram_cor.columns.values,
            yticklabels=train_cram_cor.columns.values, annot=True)
```

```
plt.yticks(rotation=0)
plt.title("Współczynniki V Cramera dla poszczególnych par typów komentarzy",␣
 ↪fontsize=18, y=1.05)
plt.show()
```



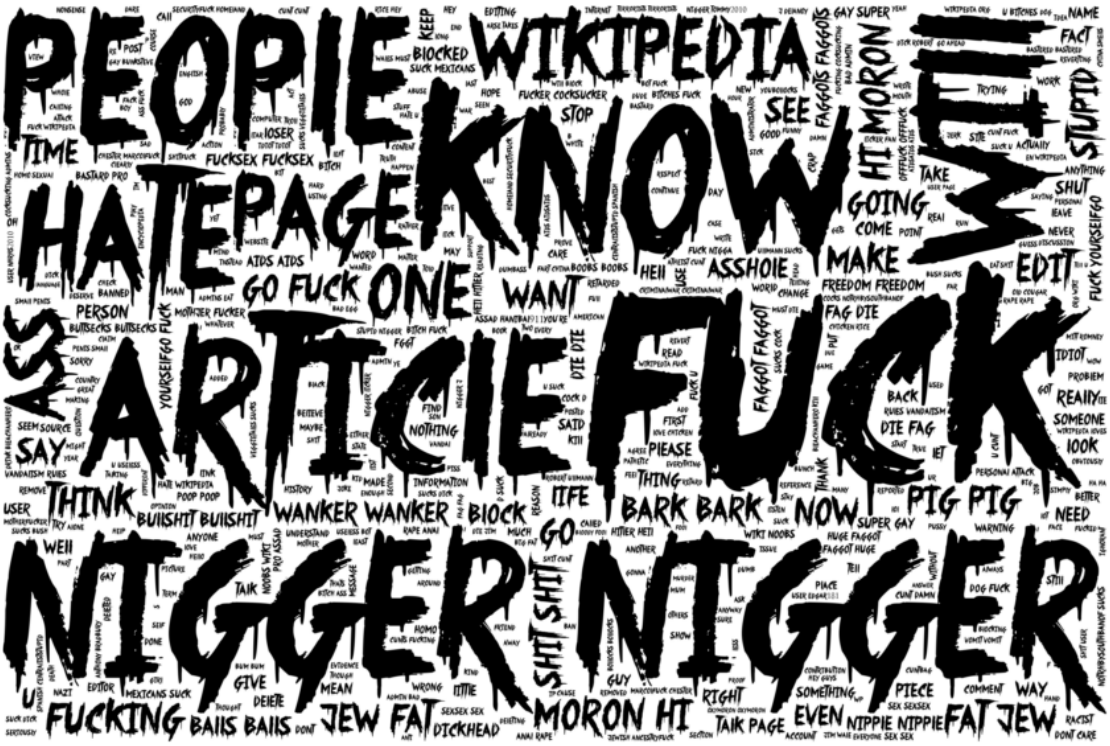
Możemy zauważyć, że komentarze typu *threat* występują niezależnie od innych typów. Z kolei najwyższy poziom zależności wykazują typy:

- *toxic* i *obscene*
- *toxic* i *insult*
- *obscene* i *insult*

2 Wizualizacja danych na podstawie WordClouds

[29]:
```
# Najczęstsze występujące słowa na podstawie WordClouds
text = " ".join(comment for comment in train[np.sum(train_values[:,2:],␣
 ↪axis=1)>0].comment_text)
# Create and generate a word cloud image:
wordcloud = WordCloud(background_color="white", font_path = 'Something Strange.
 ↪ttf', width = 3000, height = 2000,
           max_words=500, color_func=lambda *args, **kwargs: "black").
 ↪generate(text)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear') #argument interpolation␣
 ↪odpowiada za bardziej gładki obraz*
#*https://www.datacamp.com/community/tutorials/wordcloud-python
plt.axis("off")
plt.title("Najczęstsze Słowa występujące w Toksycznych Komentarzach",␣
 ↪fontsize=20, y=1.05)
fig = plt.gcf()
fig.set_size_inches(15, 13)
plt.show()
```

Najczęstsze Słowa występujące w Toksycznych Komentarzach



[26]:
```
# dla threat
text = " ".join(comment for comment in train[train['threat']==1].comment_text)

wordcloud = WordCloud(background_color="white", font_path = 'HelpMe.ttf', width␣
↪= 3000, height = 2000,
          max_words=500, color_func=lambda *args, **kwargs: "black").
↪generate(text)

# Display the generated image:
plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Najczęstsze Słowa występujące w Komentarzach Typu 'Threat'",␣
↪fontsize=20, y=1.05)
fig = plt.gcf()
fig.set_size_inches(15, 13)
plt.show()
```

Najczęstsze Słowa występujące w Komentarzach Typu 'Threat'

[27]:
```
# dla insult
text = " ".join(comment for comment in train[train['insult']==1].comment_text)
```

```
wordcloud = WordCloud(background_color="white", font_path = 'Scream Again.ttf',␣
 ↪width = 3000, height = 2000,
            max_words=500, color_func=lambda *args, **kwargs: "black").
 ↪generate(text)


plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Najczęstsze Słowa występujące w Komentarzach Typu 'Insult'",␣
 ↪fontsize=20, y=1.05)
fig = plt.gcf()
fig.set_size_inches(15, 13)
plt.show()
```

Najczęstsze Słowa występujące w Komentarzach Typu 'Insult'

[28]:
```
# dla identity_hate
text = " ".join(comment for comment in train[train['identity_hate']==1].
 ↪comment_text)

wordcloud = WordCloud(background_color="white", font_path = 'Remnant-Regular.
 ↪ttf', width = 3000, height = 2000,
```

```
            max_words=500, color_func=lambda *args, **kwargs: "black").
↪generate(text)


plt.imshow(wordcloud, interpolation='bilinear')
plt.axis("off")
plt.title("Najczęstsze Słowa występujące w Komentarzach Typu 'Identity-hate'",␣
↪fontsize=20, y=1.05)
fig = plt.gcf()
fig.set_size_inches(15, 13)
plt.show()
```

Najczęstsze Słowa występujące w Komentarzach Typu 'Identity-hate'