toxic_descriptive_statistics

December 5, 2019

In [16]:
```
import re
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
import seaborn as sns
import nltk
from nltk.corpus import stopwords
from nltk.stem.wordnet import WordNetLemmatizer
en_stop = set(nltk.corpus.stopwords.words('english'))
```

In [3]:
```
toxic_comments = pd.read_csv("toxic_comments.csv")
```

In [17]:
```
def preprocess_text(document):

        #now = datetime.datetime.now()

        # Remove all the special characters
        document = re.sub(r'\W', ' ', str(document))

        # remove all single characters
        document = re.sub(r'\s+[a-zA-Z]\s+', ' ', document)

        # Remove single characters from the start
        document = re.sub(r'\^[a-zA-Z]\s+', ' ', document)

        # Substituting multiple spaces with single space
        document = re.sub(r'\s+', ' ', document, flags=re.I)

        # Removing prefixed 'b'
        document = re.sub(r'^b\s+', '', document)

        # Converting to Lowercase
        document = document.lower()

        tokens = document.split()

        #### Remove stopwords
```

```
        words = [w for w in tokens if w not in stopwords.words('english')]
        words = [word for word in words if word not in en_stop]

        #### Lemmatize tokens obtained after removing stopwords
        wnl = WordNetLemmatizer()
        tagged = nltk.pos_tag(words)
        lem_list = []
        for word, tag in tagged:
            wntag = tag[0].lower()
            wntag = wntag if wntag in ['a', 'r', 'n', 'v'] else None
            if not wntag:
                lemma = word
            else:
                lemma = wnl.lemmatize(word, wntag)
            lem_list.append(lemma)

        preprocessed_text = ' '.join(lem_list)
        #lem_text = " ".join(lemma for lemma in lem_list)
        #print("Took %s"%(datetime.datetime.now()-now))

        return preprocessed_text
```

```
In [18]: # Clean all plot text summaries and append as a new column
         toxic_comments['clean_comment_text'] = toxic_comments['comment_text'].apply(lambda x:
```

```
In [19]: # Write prepared dataset to a csv for future use
         toxic_comments.to_csv("toxic_comments_cleaned_df.csv", index = False)
```

```
In [21]: df_toxic = toxic_comments.drop(['id', 'comment_text', 'clean_comment_text'], axis=1)
         counts = []
         categories = list(df_toxic.columns.values)
         for i in categories:
             counts.append((i, df_toxic[i].sum()))
         df_stats = pd.DataFrame(counts, columns=['category', 'number_of_comments'])
         df_stats
```

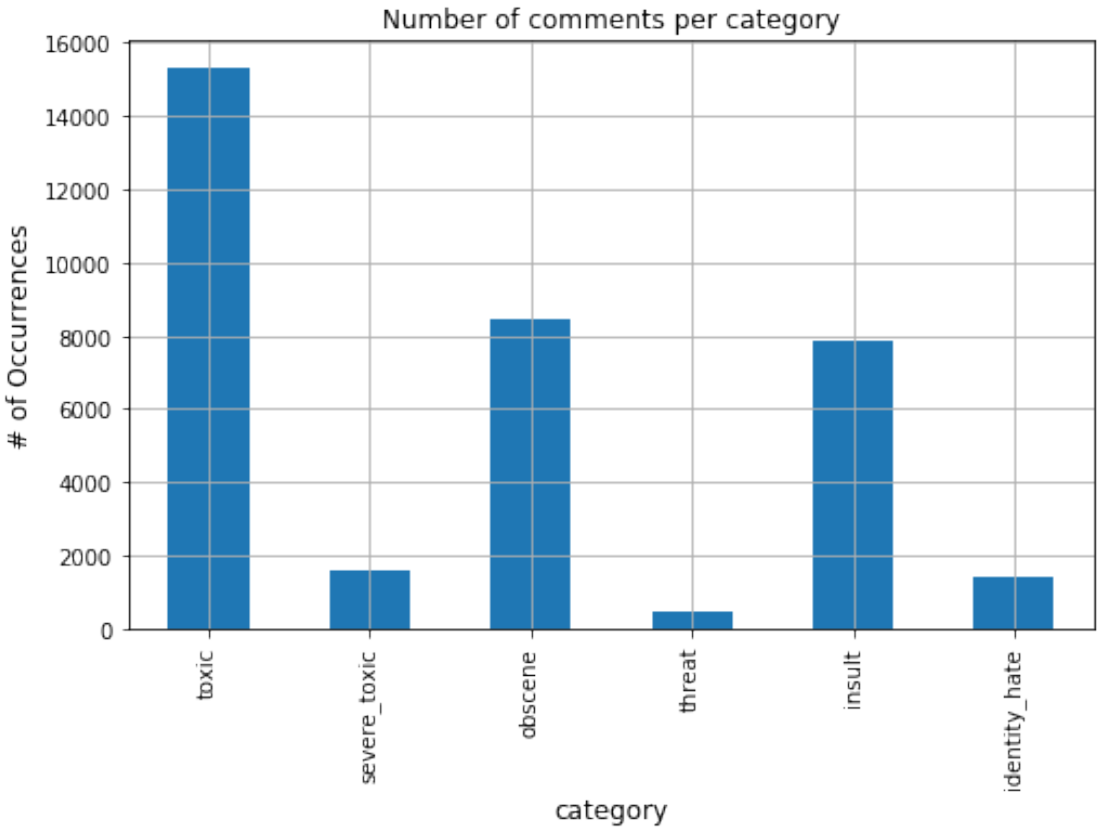
Out[21]:

	category	number_of_comments
0	toxic	15294
1	severe_toxic	1595
2	obscene	8449
3	threat	478
4	insult	7877
5	identity_hate	1405

0.0.1 Distribution of number comments per label

```
In [6]: df_stats.plot(x='category', y='number_of_comments', kind='bar', legend=False, grid=Tru
        plt.title("Number of comments per category")
        plt.ylabel('# of Occurrences', fontsize=12)
        plt.xlabel('category', fontsize=12)
```

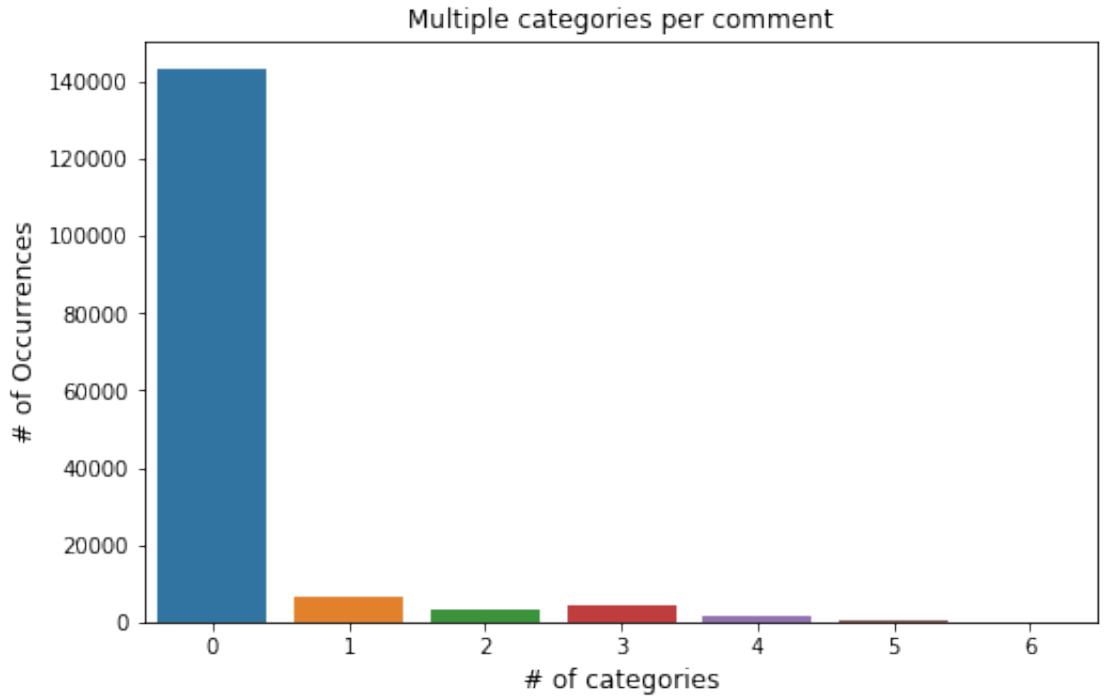
Out[6]: Text(0.5, 0, 'category')



0.0.2 Distribution of number of labels per movie

```
In [10]: rowsums = toxic_comments.iloc[:,2:].sum(axis=1)
         x=rowsums.value_counts()
         #plot
         plt.figure(figsize=(8,5))
         ax = sns.barplot(x.index, x.values)
         plt.title("Multiple categories per comment")
         plt.ylabel('# of Occurrences', fontsize=12)
         plt.xlabel('# of categories', fontsize=12)
```

Out[10]: Text(0.5, 0, '# of categories')



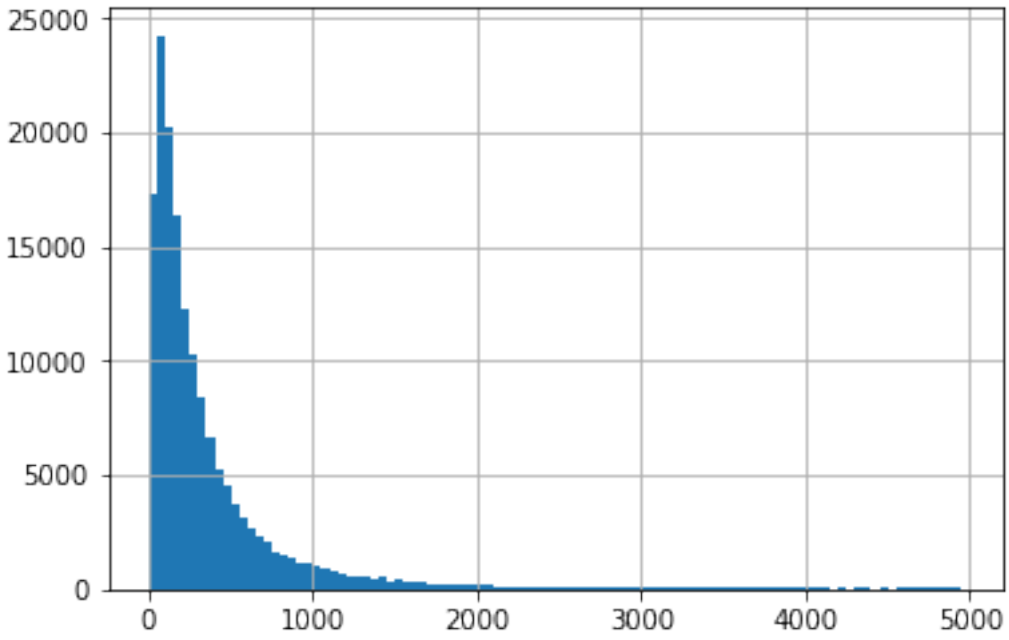

```
In [12]: print('Percentage of comments that are not labelled:')
         print(len(toxic_comments[(toxic_comments['toxic']==0) & (toxic_comments['severe_toxic
                                  (toxic_comments['obscene']==0) & (toxic_comments['threat']==
                                  (toxic_comments['insult']==0) & (toxic_comments['identity_ha
```

```
Percentage of comments that are not labelled:
0.8983211235124177
```

0.0.3 The distribution of the number of words in comment texts

```
In [13]: lens = toxic_comments.comment_text.str.len()
         lens.hist(bins = np.arange(0,5000,50))
```

```
Out[13]: <matplotlib.axes._subplots.AxesSubplot at 0x1245b02e8>
```



In [ ]: