toxic_comment_classification_original

July 1, 2023

DATA COLLECTION *KAGGLE*

[1]:
```
from google.colab import drive
drive.mount('/content/drive')
```

```
Mounted at /content/drive
```

[2]:
```
!kaggle -h
```

```
Traceback (most recent call last):
  File "/usr/local/bin/kaggle", line 5, in <module>
    from kaggle.cli import main
  File "/usr/local/lib/python3.10/dist-packages/kaggle/__init__.py", line 23, in
<module>
    api.authenticate()
  File "/usr/local/lib/python3.10/dist-
packages/kaggle/api/kaggle_api_extended.py", line 164, in authenticate
    raise IOError('Could not find {}. Make sure it\'s located in'
OSError: Could not find kaggle.json. Make sure it's located in /root/.kaggle. Or
use the environment method.
```

[3]:
```
!cp /content/drive/MyDrive/kaggle.json /root/.kaggle/kaggle.json
```

[4]:
```
!kaggle competitions download -c jigsaw-toxic-comment-classification-challenge
```

```
Downloading jigsaw-toxic-comment-classification-challenge.zip to /content
 99% 52.0M/52.6M [00:04<00:00, 18.5MB/s]
100% 52.6M/52.6M [00:04<00:00, 13.0MB/s]
```

[5]:
```
!unzip *.zip
```

```
Archive:  jigsaw-toxic-comment-classification-challenge.zip
  inflating: sample_submission.csv.zip
  inflating: test.csv.zip
  inflating: test_labels.csv.zip
  inflating: train.csv.zip
```

[6]:
```
!unzip train.csv.zip -d train
!unzip test.csv.zip -d test
```

```
!unzip test_labels.csv.zip -d testLabel
```

```
Archive:  train.csv.zip
  inflating: train/train.csv
Archive:  test.csv.zip
  inflating: test/test.csv
Archive:  test_labels.csv.zip
  inflating: testLabel/test_labels.csv
```

[7]:
```
import pandas as pd
dftrain=pd.read_csv('/content/train/train.csv')
dftrain
```

[7]:

	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
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
…	…	…	…	…	…	…
159566	0	0	0	0	0	0
159567	0	0	0	0	0	0
159568	0	0	0	0	0	0
159569	0	0	0	0	0	0
159570	0	0	0	0	0	0

[159571 rows x 8 columns]

The dataset is in csv format i am trying to use tensorflow and build my own neural network using **LSTM** layer.

Now data exploring

[8]:
```
#perform basic stats
dftrain.describe()
```

[8]:
```
                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
```

[9]:
```
dftrain.isnull().sum()
```

[9]:
```
id               0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
```

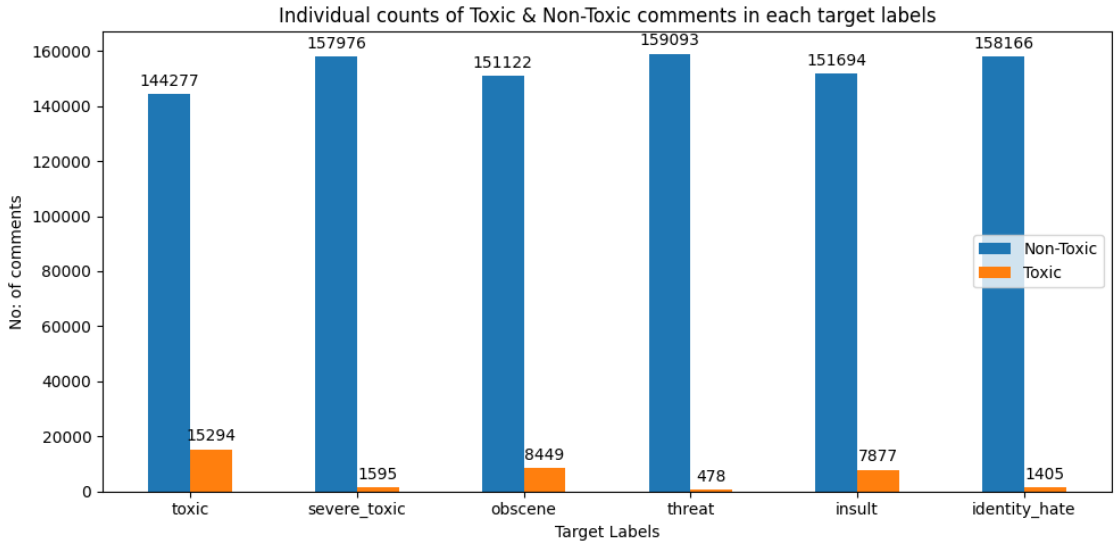
[10]:
```
# checking the value counts for each targets individually
for i in dftrain.columns.values[2:]:#skipping id and comment text columns
    print(dftrain[i].value_counts())
    print("-"*50)
```

```
0    144277
1     15294
Name: toxic, dtype: int64
--------------------------------------------------
0    157976
1      1595
Name: severe_toxic, dtype: int64
--------------------------------------------------
0    151122
1      8449
```

```
Name: obscene, dtype: int64
--------------------------------------------------
0    159093
1       478
Name: threat, dtype: int64
--------------------------------------------------
0    151694
1      7877
Name: insult, dtype: int64
--------------------------------------------------
0    158166
1      1405
Name: identity_hate, dtype: int64
--------------------------------------------------
```

[11]:
```
import matplotlib.pyplot as plt
import numpy as np

labels = dftrain.columns.values[2:]
class_0=[]
class_1=[]

for i in dftrain.columns.values[2:]:
    vc=dftrain[i].value_counts()
    class_0.append(vc[0])
    class_1.append(vc[1])

x = np.arange(len(labels))  # the label locations
width = 0.25  # the width of the bars

fig, ax = plt.subplots(figsize=(10,5))
rects1 = ax.bar(x - width/2, class_0, width, label='Non-Toxic')
rects2 = ax.bar(x + width/2, class_1, width, label='Toxic')

# Add some text for labels, title and custom x-axis tick labels, etc.
ax.set_ylabel('No: of comments')
ax.set_xlabel("Target Labels")
ax.set_title('Individual counts of Toxic & Non-Toxic comments in each target␣
 ↪labels')
ax.set_xticks(x)
ax.set_xticklabels(labels)
ax.legend(loc=7)

ax.bar_label(rects1, padding=3)
ax.bar_label(rects2, padding=3)

ax.bar
```

```
fig.tight_layout()

plt.show()
```



[12]:
```
!pip install venn
```

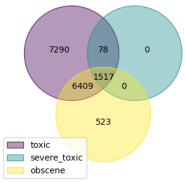
```
Collecting venn
  Downloading venn-0.1.3.tar.gz (19 kB)
  Preparing metadata (setup.py) … done
Requirement already satisfied: matplotlib in /usr/local/lib/python3.10/dist-
packages (from venn) (3.7.1)
Requirement already satisfied: contourpy>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->venn) (1.1.0)
Requirement already satisfied: cycler>=0.10 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->venn) (0.11.0)
Requirement already satisfied: fonttools>=4.22.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->venn) (4.40.0)
Requirement already satisfied: kiwisolver>=1.0.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->venn) (1.4.4)
Requirement already satisfied: numpy>=1.20 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->venn) (1.22.4)
Requirement already satisfied: packaging>=20.0 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->venn) (23.1)
Requirement already satisfied: pillow>=6.2.0 in /usr/local/lib/python3.10/dist-
packages (from matplotlib->venn) (8.4.0)
Requirement already satisfied: pyparsing>=2.3.1 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->venn) (3.1.0)
Requirement already satisfied: python-dateutil>=2.7 in
/usr/local/lib/python3.10/dist-packages (from matplotlib->venn) (2.8.2)
```

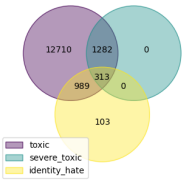
```
Requirement already satisfied: six>=1.5 in /usr/local/lib/python3.10/dist-
packages (from python-dateutil>=2.7->matplotlib->venn) (1.16.0)
Building wheels for collected packages: venn
  Building wheel for venn (setup.py) … done
  Created wheel for venn: filename=venn-0.1.3-py3-none-any.whl size=19699
sha256=5197ec5233950681fe62c7f2694677a7f4d91ed8bfbb502210cbf4a4256c425f
  Stored in directory: /root/.cache/pip/wheels/9c/ce/43/705b4a04cd822891d1d7a4c4
3fc444b4798978e72c79528c5f
Successfully built venn
Installing collected packages: venn
Successfully installed venn-0.1.3
```

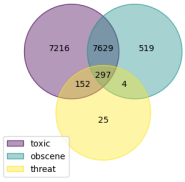
[13]:
```
import venn
from itertools import combinations
no_of_labels= np.arange(2,6)
rows_col=[(5,3),(5,4),(5,3),(2,3)] #The variable rows_col is a list of tuples␣
 ↪specifying the number of rows and columns for each subplot.

for i,rc in zip(no_of_labels,rows_col):
    comb = combinations(dftrain.columns.values[2:], i)
    fig, top_axs = plt.subplots(ncols=rc[1], nrows=rc[0],figsize=(20, 20))
    fig.suptitle("Venn diagram - considering "+str(i)+" Target␣
 ↪Labels",fontsize=24)
    fig.subplots_adjust(top=0.88)
    fig.tight_layout()
    top_axs=top_axs.flatten()
    for j,ax in zip(list(comb),top_axs):
        data_set=dict()
        for k in j:
            data_set[k]=set(dftrain[(dftrain[k]==1)].index)
        venn_dgrm=venn.venn(data_set,legend_loc="best",alpha=0.
 ↪4,fontsize=10,ax=ax)
```

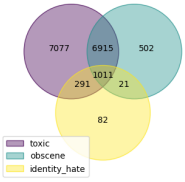
Venn diagram - considering 2 Target Labels

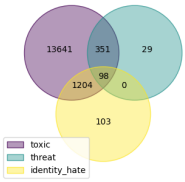
Venn diagram - considering 3 Target Labels

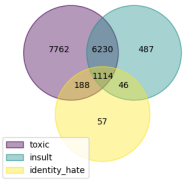


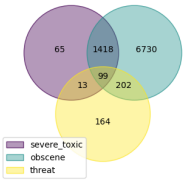


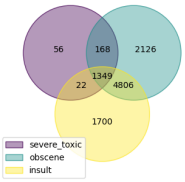


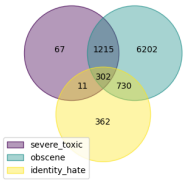


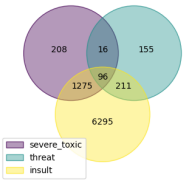


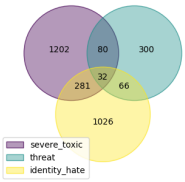


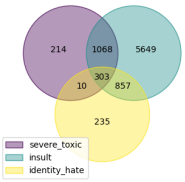


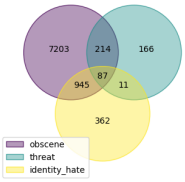


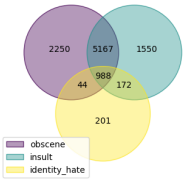




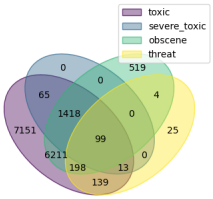


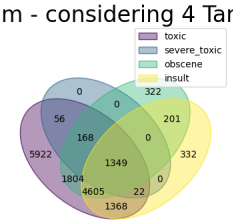


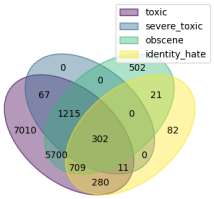


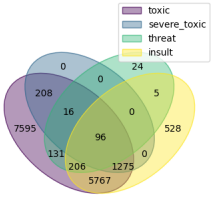


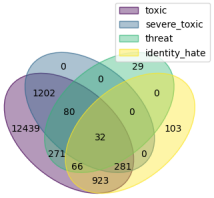
Venn diagram - considering 4 Target Labels

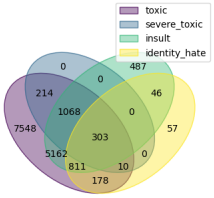


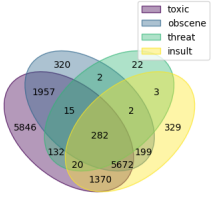


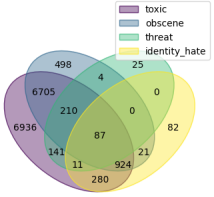


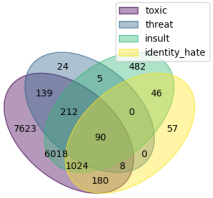


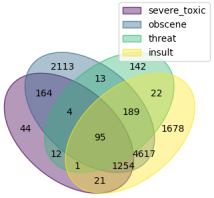


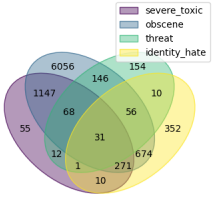


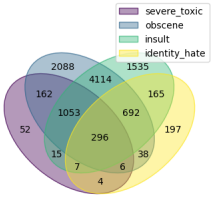


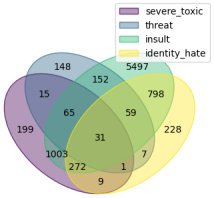


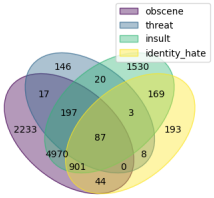




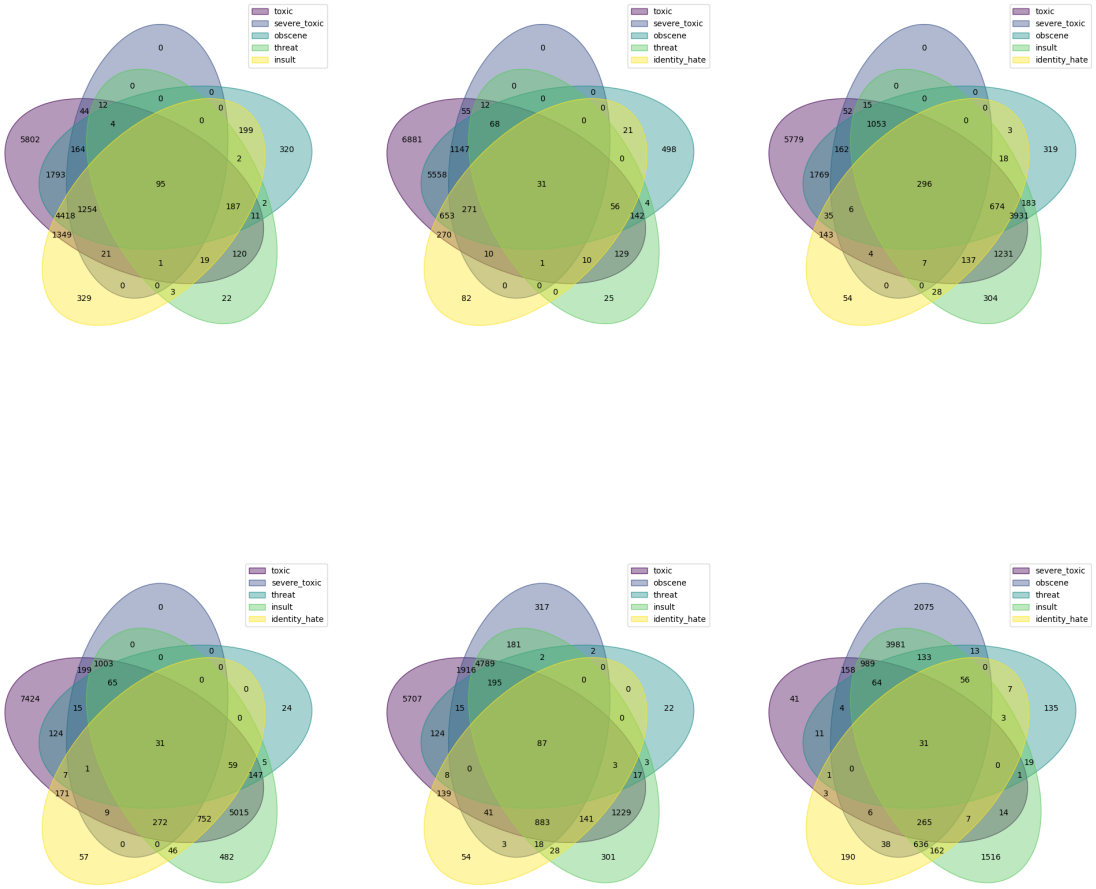






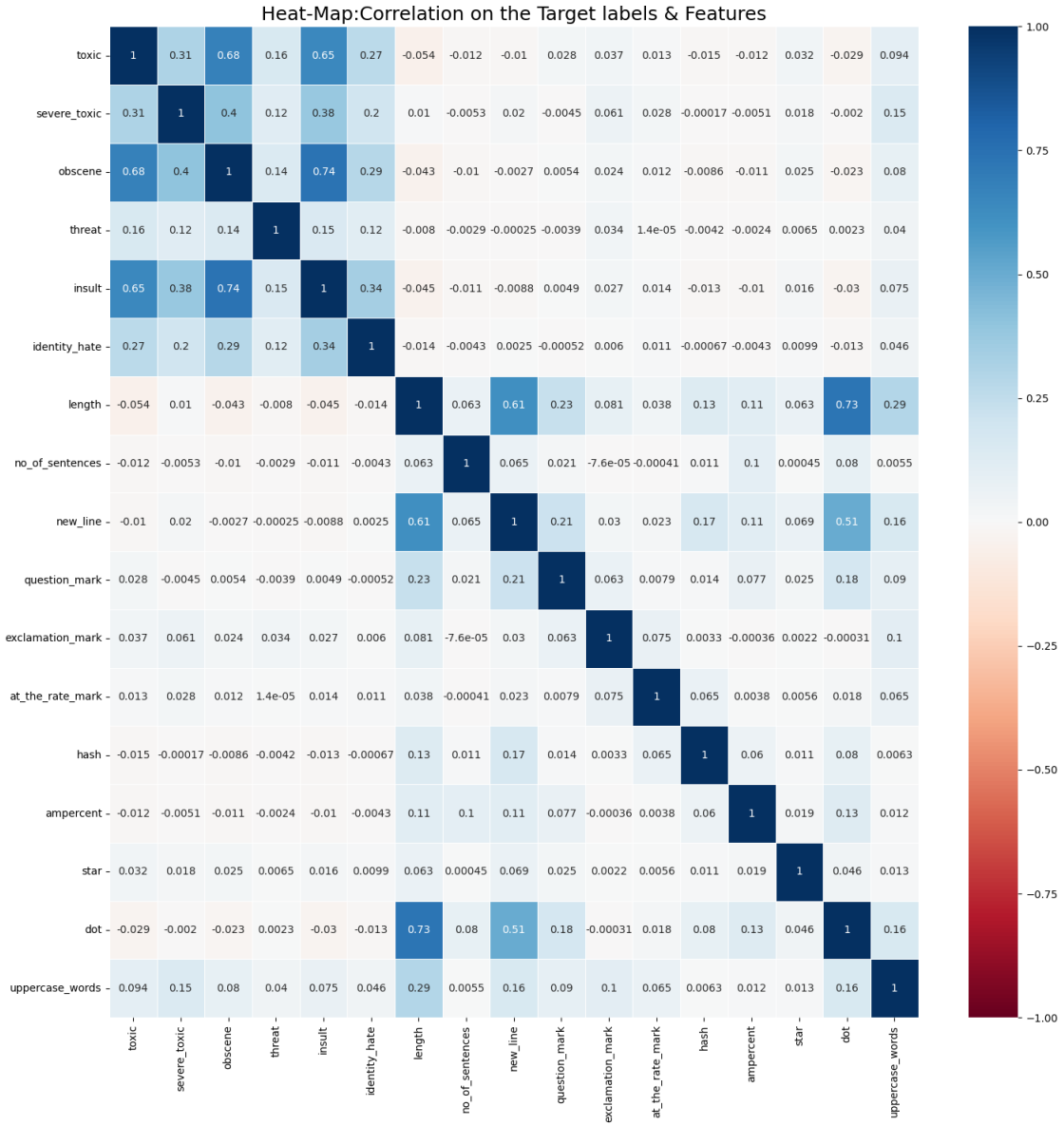


Venn diagram - considering 5 Target Labels



[14]:
```
import seaborn as sns
corr_df=dftrain.drop(columns=["id","comment_text"])
corr_df['length']=dftrain['comment_text'].str.len()
corr_df['no_of_sentences']=dftrain['comment_text'].str.split("/n").apply(len)
corr_df['new_line'] = dftrain['comment_text'].str.count('\n')
corr_df['question_mark'] = dftrain['comment_text'].str.count('\?')
corr_df['exclamation_mark'] = dftrain['comment_text'].str.count('!')
corr_df['at_the_rate_mark'] = dftrain['comment_text'].str.count('@')
corr_df['hash'] = dftrain['comment_text'].str.count('#')
corr_df['ampercent'] = dftrain['comment_text'].str.count('&')
corr_df['star']= dftrain['comment_text'].str.count('\*')
corr_df['dot'] = dftrain['comment_text'].str.count('\.')
corr_df['uppercase_words'] = dftrain['comment_text'].str.split().apply(lambda x:
 ↪ sum(map(str.isupper, x)))
```

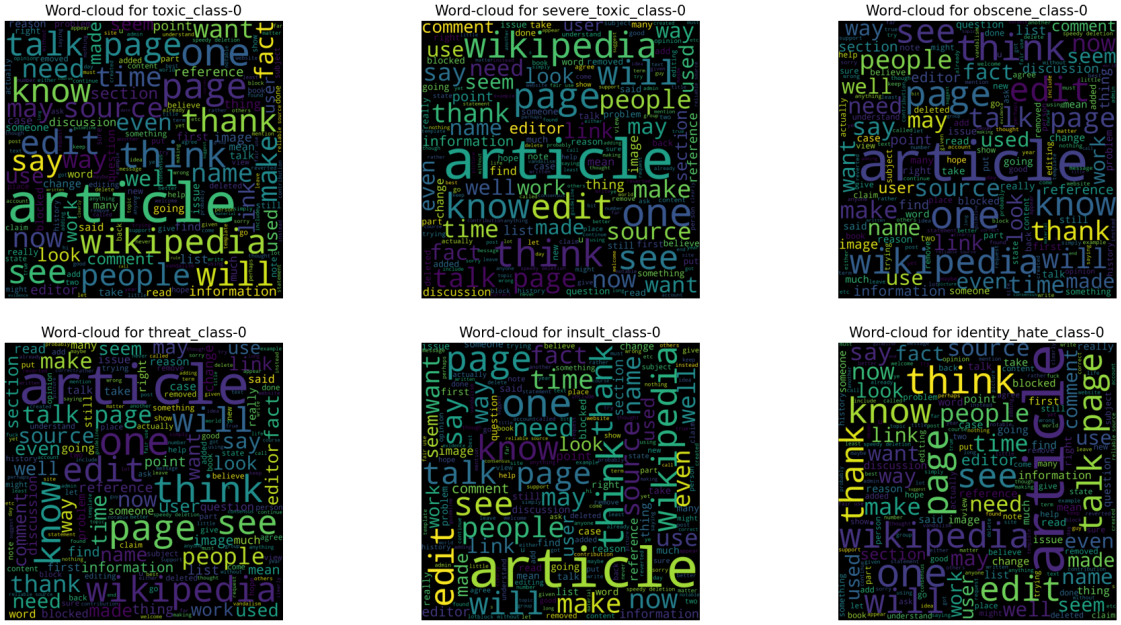
```
correlation=corr_df.corr()

plt.figure(figsize=(15,15))
sns.heatmap(correlation,vmin=-1,cmap='RdBu',annot=True,linewidths=.5)
plt.title("Heat-Map:Correlation on the Target labels & Features",fontsize=18)
plt.tight_layout()
```



[15]:
```
from wordcloud import WordCloud, STOPWORDS
```

[16]:
```
#doc: https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.
↪html#wordcloud.WordCloud
```

```
plt.figure(figsize=(20,10))
count=1
for col in dftrain.columns[2:]:
    toxic_class_0 = dftrain[dftrain[col]==0]['comment_text'].str.lower().values
    wordcloud = WordCloud(width=2000, height=2000,
                          background_color ='black', margin=1,
                          stopwords = STOPWORDS,
                          ).generate(" ".join(toxic_class_0))

    plt.subplot(2,3,count)
    plt.axis("off")
    plt.title("Word-cloud for "+col+"_class-0",fontsize=15)
    plt.tight_layout(pad=3)
    plt.imshow(wordcloud,interpolation='bilinear')
    count=count+1
plt.show()
```



[17]:
```
#doc: https://amueller.github.io/word_cloud/generated/wordcloud.WordCloud.
 ↪html#wordcloud.WordCloud
plt.figure(figsize=(20,10))
count=1
for col in dftrain.columns[2:]:
    toxic_class_0 = dftrain[dftrain[col]==1]['comment_text'].str.lower().values
    wordcloud = WordCloud(width=2000, height=2000,
                          background_color ='black', margin=1,
                          stopwords = STOPWORDS,
```

```
                        ).generate(" ".join(toxic_class_0))

    plt.subplot(2,3,count)
    plt.axis("off")
    plt.title("Word-cloud for "+col+"_class-1",fontsize=15)
    plt.tight_layout(pad=3)
    plt.imshow(wordcloud,interpolation='bilinear')
    count=count+1
plt.show()
```

[18]:
```
#now preprocessing the text
#text contains hyperlinks,emoji,ip address,html&css and extra punctuations and␣
 ↪special characters

dftrain['comment_text'].sample(10)
```

[18]:
```
52883    Hi Awadewit! As a new Wikipedian, it is reassu…
12367    have rich edit histories containing thousands …
98172    ":I just took a look at my user page on March …
14816                    Thank you for your encouragement.
5232     greatstory==                                   …
5972     "\n\nI concur ""tyrann"" ""terrible"" and ""ma…
59705    Note: It would not appear to be appropriate to…
37970                       ll fix eventually if you don't
95625    "\nSeveral things it says that it is the inter…
1296     Fuck you both, I'm not interested in what eith…
```

```
Name: comment_text, dtype: object
```

[19]:
```
dftrain.drop(['id'],axis=1,inplace=True)
dftrain
```

[19]:
```
                                                comment_text  toxic  \
0       Explanation\nWhy the edits made under my usern…      0
1       D'aww! He matches this background colour I'm s…      0
2       Hey man, I'm really not trying to edit war. It…      0
3       "\nMore\nI can't make any real suggestions on …      0
4       You, sir, are my hero. Any chance you remember…      0
…                                                     …    …
159566  ":::::And for the second time of asking, when …      0
159567  You should be ashamed of yourself \n\nThat is …      0
159568  Spitzer \n\nUmm, theres no actual article for …      0
159569  And it looks like it was actually you who put …      0
159570  "\nAnd … I really don't think you understand…      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
…                …       …     …     …             …
159566             0        0       0       0              0
159567             0        0       0       0              0
159568             0        0       0       0              0
159569             0        0       0       0              0
159570             0        0       0       0              0

[159571 rows x 7 columns]
```

[ ]:
```
import tensorflow as tf
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import pad_sequences
from sklearn.model_selection import train_test_split
from tensorflow.keras.utils import to_categorical
import re

# Define the preprocess function
def preprocess(text):
    # Remove URLs
```

```
    pattern = r"\b((?:https?://)?(?:(?:www\.)?(?:[\da-z\.-]+)\.(?:[a-z]{2,6})|(?
↪:(?:25[0-5]|2[0-4][0-9]|[01]?[0-9][0-9]?)\.){3}(?:25[0-5]|2[0-4][0-9]|[01]?
↪[0-9][0-9]?)|(?:(?:[0-9a-fA-F]{1,4}:){7,7}[0-9a-fA-F]{1,4}|(?:
↪[0-9a-fA-F]{1,4}:){1,7}:|(?:[0-9a-fA-F]{1,4}:){1,6}:[0-9a-fA-F]{1,4}|(?:
↪[0-9a-fA-F]{1,4}:){1,5}(?::[0-9a-fA-F]{1,4}){1,2}|(?:[0-9a-fA-F]{1,4}:
↪){1,4}(?::[0-9a-fA-F]{1,4}){1,3}|(?:[0-9a-fA-F]{1,4}:){1,3}(?::
↪[0-9a-fA-F]{1,4}){1,4}|(?:[0-9a-fA-F]{1,4}:){1,2}(?::
↪[0-9a-fA-F]{1,4}){1,5}|[0-9a-fA-F]{1,4}:(?:(?::[0-9a-fA-F]{1,4}){1,6})|:(?:(?
↪::[0-9a-fA-F]{1,4}){1,7}|:)|fe80:(?::[0-9a-fA-F]{0,4}){0,4}%[0-9a-zA-Z]{1,}|:
↪:(?:ffff(?::0{1,4}){0,1}:){0,1}(?:(?:25[0-5]|(?:
↪2[0-4]|1{0,1}[0-9]){0,1}[0-9])\.){3,3}(?:25[0-5]|(?:
↪2[0-4]|1{0,1}[0-9]){0,1}[0-9])|(?:[0-9a-fA-F]{1,4}:){1,4}:(?:(?:25[0-5]|(?:
↪2[0-4]|1{0,1}[0-9]){0,1}[0-9])\.){3,3}(?:25[0-5]|(?:
↪2[0-4]|1{0,1}[0-9]){0,1}[0-9])))(?::
↪[0-9]{1,4}|[1-5][0-9]{4}|6[0-4][0-9]{3}|65[0-4][0-9]{2}|655[0-2][0-9]|6553[0-5])?
↪(?:/[\w\.-]*)*/?)\b"
    text = re.sub(pattern, "", text)

    # Remove emoji
    pattern = "[" + u"\U0001F600-\U0001F64F" \
                  + u"\U0001F300-\U0001F5FF" \
                  + u"\U0001F680-\U0001F6FF" \
                  + u"\U0001F1E0-\U0001F1FF" \
                  + u"\U00002702-\U000027B0" \
                  + u"\U000024C2-\U0001F251" \
                  + "]+"
    text = re.sub(pattern, "", text, flags=re.UNICODE)

    # Remove IP addresses
    pattern = r"\b(?:\d{1,3}\.){3}\d{1,3}\b"
    text = re.sub(pattern, "", text)

    # Remove special characters
    pattern = r"[^\w\s]"
    text = re.sub(pattern, "", text)

    # Remove HTML tags
    pattern = r"<[^>]+>"
    text = re.sub(pattern, "", text)

    # Remove CSS syntax and inline styles
    pattern = r"(?i)<style([\s\S]*?)<\/style>|<script([\s\S]*?)<\/script>"
    text = re.sub(pattern, "", text)

    return text
```

```
#To speed up the data preprocessing and cleaning as the dataset is somewhat large
from multiprocessing import Pool

def preprocess_parallel(text):
    return preprocess(text)

with Pool() as pool:
    texts = pool.map(preprocess_parallel, dftrain['comment_text'])

print(len(texts),len(dftrain))
```

```
159571 159571
```

[ ]:
```
# Extract input (comment_text) and output (labels) from the DataFrame
input_texts = texts
output_labels = dftrain.iloc[:, 1:].values  # Assumes label columns are from index 1 to the end

# Split the dataset into train and test sets
train_texts, test_texts, train_labels, test_labels = train_test_split(
    input_texts, output_labels, test_size=0.2, random_state=42)

# Tokenize the input texts
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(train_texts)
train_sequences = tokenizer.texts_to_sequences(train_texts)
test_sequences = tokenizer.texts_to_sequences(test_texts)

# Pad the sequences to a fixed length
max_length = 6  # Adjust the maximum length as per your requirement
train_sequences = tf.keras.preprocessing.sequence.pad_sequences(train_sequences, maxlen=max_length)
test_sequences = tf.keras.preprocessing.sequence.pad_sequences(test_sequences, maxlen=max_length)

# Convert the output labels to categorical format
num_classes = output_labels.shape[1]  # Get the number of label columns
train_labels_categorical =train_labels
test_labels_categorical = test_labels
```

[ ]:
```
train_labels_categorical.shape,train_sequences.shape,test_sequences.shape
```

[ ]:
```
((127656, 6), (127656, 6), (31915, 6))
```

[ ]:
```
import tensorflow as tf
from tensorflow.keras.models import Sequential
```

```
from tensorflow.keras.layers import Embedding, LSTM, Dense
from tensorflow.keras.layers import Bidirectional, GlobalMaxPooling1D

# Define the model architecture
embedding_dim = 100  # Adjust the dimensionality of the word embeddings as per␣
 ↪your requirement

model = Sequential()
model.add(Embedding(input_dim=len(tokenizer.word_index) + 1,␣
 ↪output_dim=embedding_dim, input_length=max_length))
model.add(Bidirectional(LSTM(128, return_sequences=True)))
model.add(Dense(128, activation='relu'))
model.add(LSTM(64, return_sequences=True))
model.add(Dense(64, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(num_classes, activation='softmax'))

# Compile the model
model.compile(loss='binary_crossentropy', optimizer='adam',metrics=['accuracy'])

model.summary()
```

```
Model: "sequential_8"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_8 (Embedding)     (None, 6, 100)            20469200

 bidirectional_8 (Bidirectio  (None, 6, 256)           234496
 nal)

 dense_13 (Dense)            (None, 6, 128)            32896

 lstm_12 (LSTM)              (None, 6, 64)             49408

 dense_14 (Dense)            (None, 6, 64)             4160

 global_max_pooling1d_9 (Glo  (None, 64)               0
 balMaxPooling1D)

 dense_15 (Dense)            (None, 6)                 390

=================================================================
Total params: 20,790,550
Trainable params: 20,790,550
Non-trainable params: 0
_________________________________________________________________
```

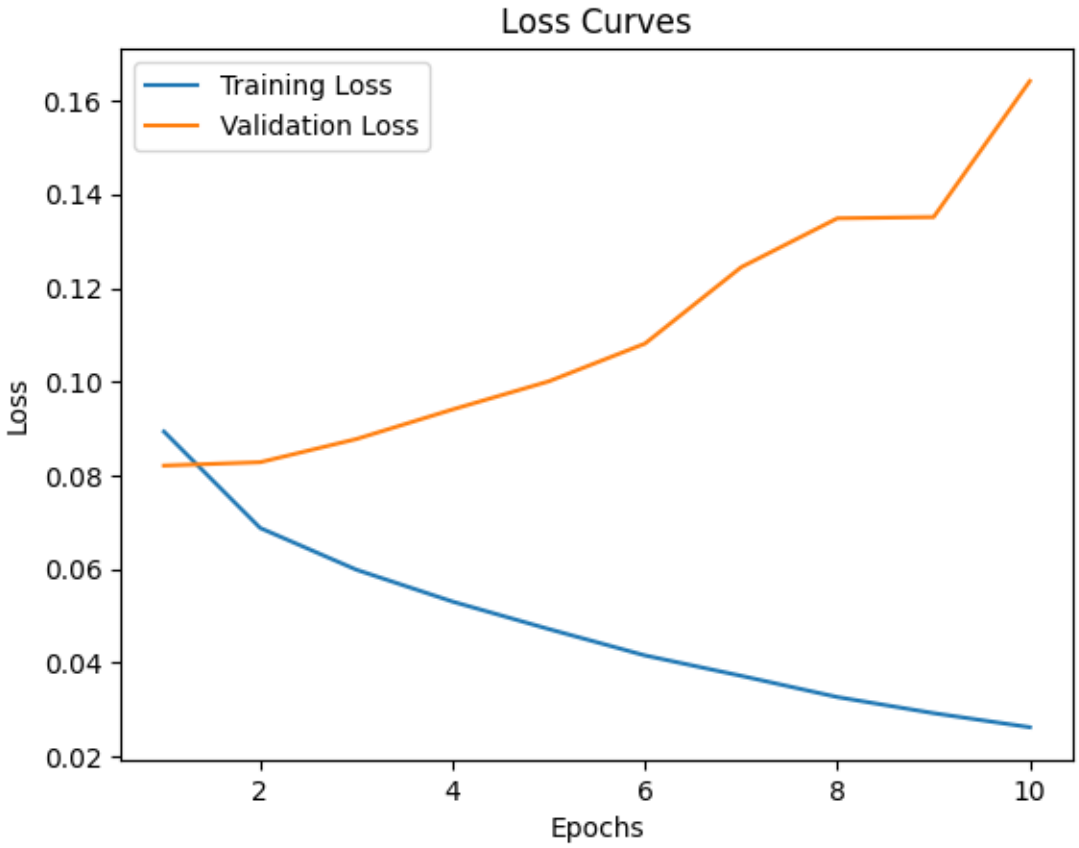
[ ]:
```
# Train the model
batch_size = 32  # Adjust the batch size as per your requirement
epochs = 10  # Adjust the number of epochs as per your requirement
```

[ ]:
```
history=model.fit(train_sequences, train_labels_categorical,␣
↪batch_size=batch_size, epochs=epochs, validation_data=(test_sequences,␣
↪test_labels_categorical))
```

```
Epoch 1/10
3990/3990 [==============================] - 77s 17ms/step - loss: 0.0894 -
accuracy: 0.9803 - val_loss: 0.0821 - val_accuracy: 0.9941
Epoch 2/10
3990/3990 [==============================] - 62s 15ms/step - loss: 0.0688 -
accuracy: 0.9898 - val_loss: 0.0828 - val_accuracy: 0.9941
Epoch 3/10
3990/3990 [==============================] - 62s 15ms/step - loss: 0.0599 -
accuracy: 0.9838 - val_loss: 0.0878 - val_accuracy: 0.9941
Epoch 4/10
3990/3990 [==============================] - 61s 15ms/step - loss: 0.0531 -
accuracy: 0.9884 - val_loss: 0.0941 - val_accuracy: 0.9941
Epoch 5/10
3990/3990 [==============================] - 61s 15ms/step - loss: 0.0472 -
accuracy: 0.9910 - val_loss: 0.1001 - val_accuracy: 0.9926
Epoch 6/10
3990/3990 [==============================] - 61s 15ms/step - loss: 0.0416 -
accuracy: 0.9714 - val_loss: 0.1081 - val_accuracy: 0.9818
Epoch 7/10
3990/3990 [==============================] - 62s 15ms/step - loss: 0.0372 -
accuracy: 0.9734 - val_loss: 0.1244 - val_accuracy: 0.9890
Epoch 8/10
3990/3990 [==============================] - 61s 15ms/step - loss: 0.0327 -
accuracy: 0.9688 - val_loss: 0.1349 - val_accuracy: 0.9843
Epoch 9/10
3990/3990 [==============================] - 60s 15ms/step - loss: 0.0293 -
accuracy: 0.9769 - val_loss: 0.1351 - val_accuracy: 0.9702
Epoch 10/10
3990/3990 [==============================] - 60s 15ms/step - loss: 0.0263 -
accuracy: 0.9645 - val_loss: 0.1642 - val_accuracy: 0.9695
```

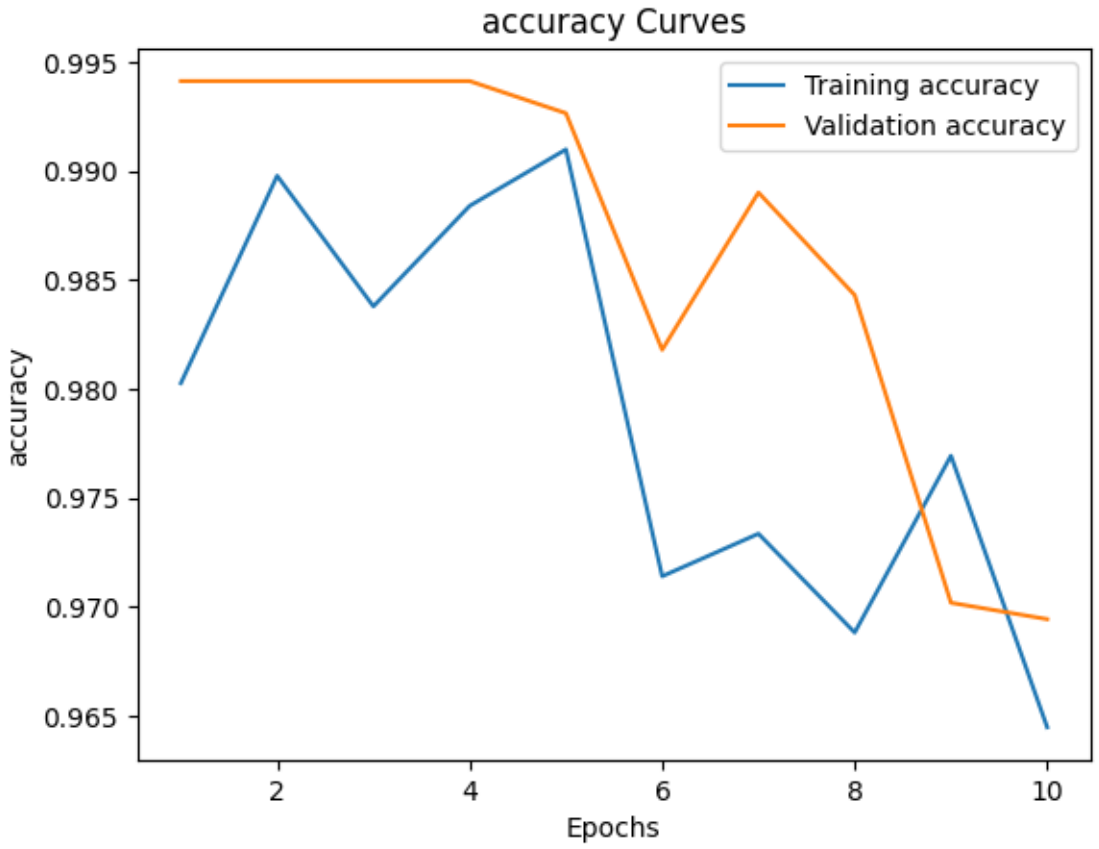
[ ]:
```
import matplotlib.pyplot as plt

# Train the model and obtain the history object
#history = model.fit(train_sequences, train_labels_categorical,␣
↪batch_size=batch_size, epochs=epochs, validation_data=(test_sequences,␣
↪test_labels_categorical))

# Get the loss values from the history object
```

```
train_loss = history.history['loss']
val_loss = history.history['val_loss']

# Plot the loss curves
epochs_range = range(1, epochs+1)
plt.plot(epochs_range, train_loss, label='Training Loss')
plt.plot(epochs_range, val_loss, label='Validation Loss')
plt.xlabel('Epochs')
plt.ylabel('Loss')
plt.title('Loss Curves')
plt.legend()
plt.show()
```



[ ]:
```
import matplotlib.pyplot as plt

# Get the loss values from the history object
train_accuracy = history.history['accuracy']
val_accuracy = history.history['val_accuracy']

# Plot the loss curves
```

```
epochs_range = range(1, epochs+1)
plt.plot(epochs_range, train_accuracy, label='Training accuracy')
plt.plot(epochs_range, val_accuracy, label='Validation accuracy')
plt.xlabel('Epochs')
plt.ylabel('accuracy')
plt.title('accuracy Curves')
plt.legend()
plt.show()
```



[ ]:
```
model.evaluate(test_sequences,test_labels_categorical)
```

```
998/998 [==============================] - 6s 6ms/step - loss: 0.1642 -
accuracy: 0.9695
```

[ ]:
```
[0.16415560245513916, 0.9694501161575317]
```

[ ]:
```
import time
name="final_model.h5"#'model_'+str(time.ctime()).replace(" ","_")+'.h5'
model.save(name)
```

Testing Model

[ ]:
```
df_test=pd.read_csv("/content/test/test.csv")
df_test_label=pd.read_csv("/content/testLabel/test_labels.csv")
# Perform the join based on the ID column
merged_df = pd.merge(df_test, df_test_label, on='id')

# Save the merged dataframe to a new CSV file
merged_df.to_csv('merged_file.csv', index=False)

merged_df.drop(['id'],axis=1,inplace=True)

merged_df.isna().sum()
```

[ ]:
```
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
```

[ ]:
```
#data preprocessing
with Pool() as pool:
    texts = pool.map(preprocess_parallel, merged_df['comment_text'])
input_texts = texts
output_labels = merged_df.iloc[:, 1:].values

# Tokenize the input texts
tokenizer = tf.keras.preprocessing.text.Tokenizer()
tokenizer.fit_on_texts(input_texts)
eval_sequences = tokenizer.texts_to_sequences(input_texts)

# Pad the sequences to a fixed length
max_length = 6  # Adjust the maximum length as per your requirement
eval_sequences = tf.keras.preprocessing.sequence.pad_sequences(eval_sequences,␣
 ↪maxlen=max_length)

# Convert the output labels to categorical format
num_classes = output_labels.shape[1]  # Get the number of label columns
eval_labels_categorical =output_labels
```

[ ]:
```
evaluate=model.evaluate(eval_sequences,eval_labels_categorical)
```

```
4787/4787 [==============================] - 25s 5ms/step - loss: -5.6686 -
accuracy: 0.9683
```

[ ]:
```
Accuracy = evaluate[1]*100
print(f"Accuracy : {Accuracy:.2f}")
```

```
Accuracy : 96.83
```