Toxic Comment Classification Challenge

PRESENT BY :

1. OMAR AHMED MOHAMED AHMED

2. ASMAA SAEED

3. MOHAMED SALAMA

4. ALAA SALAH MOHAMED

5. FATIMA SAMIR

Group :
15

Present TO :
Dr.Ahmed El-Sallab

Agenda

- Overview
- Data Analysis
- Text preprocessing pipeline
- BoW model
- LSTM, GRU, Bidirectional models

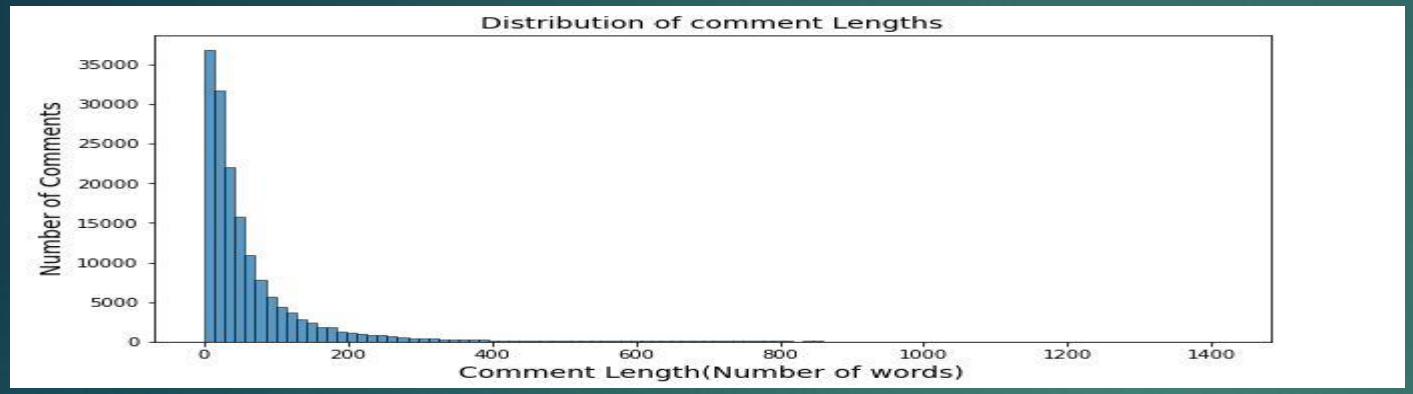
Overview

This is a Competition of Kaggle in NLP to classify Toxic comments of Comments and Classification of these Toxic comments into six classes (Toxic – Severe toxic – Obscene – Threat – Insult – Identity hate).

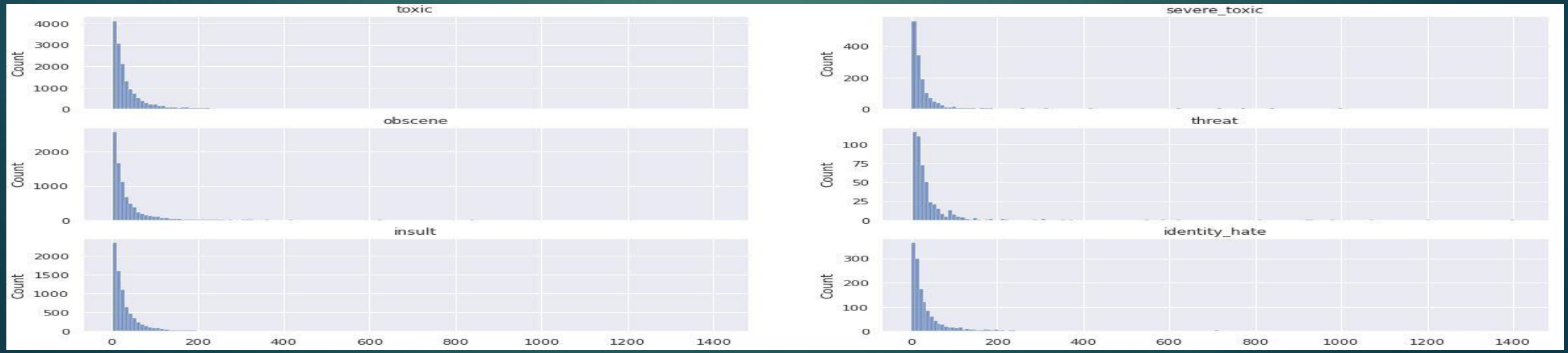
In this competition, you're challenged to build a multi-headed model that's capable of detecting different types of toxicity like threats, obscenity, insults, and identity-based hate better than Perspective's current models. You'll be using a dataset of comments from Wikipedia's talk page edits. Improvements to the current model will hopefully help online discussion become more productive and respectful.

Data Analysis

- ❑ Plot the distribution of tweet lengths
- ❖ We Looping in comment_text column and summation each comment length and plot Distribution of them

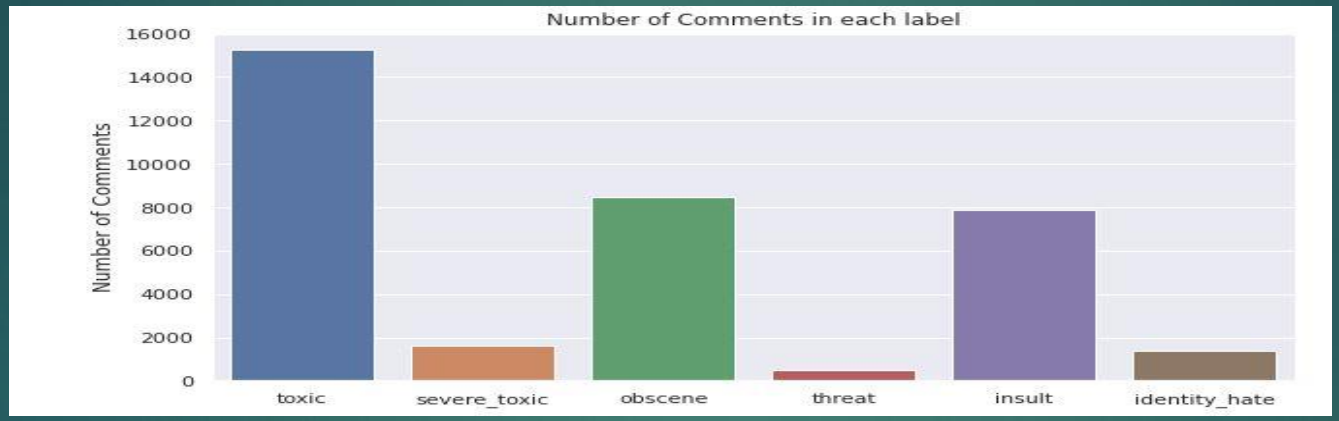


- ❑ We can see that most of the comments have the tweet lengths around 200 words. Few comments have very high length. We can further see the distribution of the comment lengths by each labels.

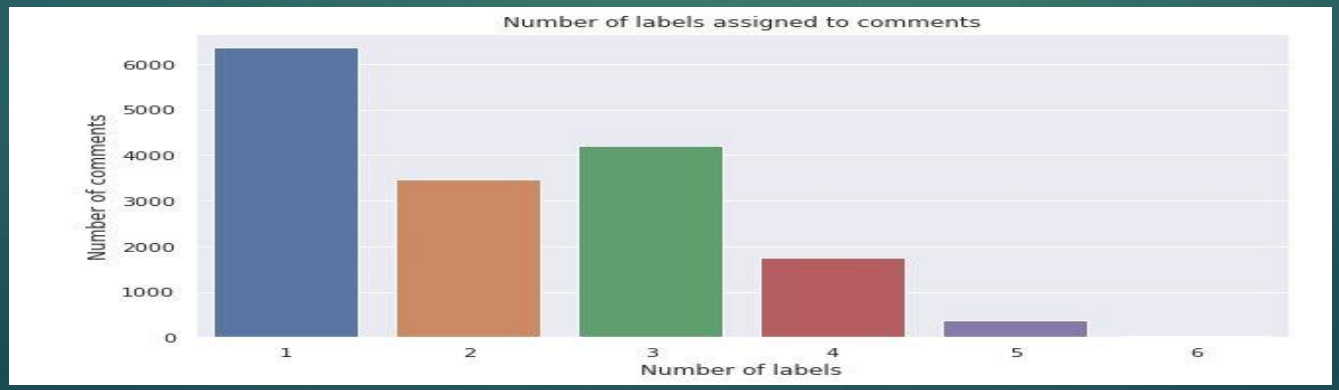


❑ Distribution of Labels

- Since this is a multi label classification task, each comment can have more than one label. Here, we will see number of comments belonging to a label. Also, we will see if comments are not classified to any labels.



- As we see, there are higher number of comments belonging to toxic labels, and comments in sever_toxic, threat and identity_hate are low. So, here we have the problem of imbalanced data and therfore we will have to be careful in selecting the correct evaluation metric later.



Text preprocessing pipeline

- We make a one Function including a lot of function for Text preprocessing pipeline and Data Cleaning for Comments:

- Remove special char such as (replace('\\"', "").replace('<unk>', 'u_n').replace(' @.@ ', '.'))
- Remove non-ASCII characters from list of tokenized words
- Apply Lowercase in text
- Remove punctuation from list of tokenized words
- Replace all integer occurrences in list of tokenized words with textual representation
- Remove Stopwords
- Lemmatize words and Verbs in text
- Apply Word tokenize in text

BoW model

- We split data into x_train , y_train , x_test and y_test.
- We Apply Tokenizer in Comment Before Applying model in dataset we Apply Tokenizer in x_train and y_test and use tokenizer.texts_to_matrix to put result of Tokenizer in text in matrix before modeling.
- We make modeling network process by Entering Dense layer , activation and input shape
- We Make modeling compile By Entering optimizer, loss and metrics
- We fit model By Entering tokenized_train, y_train, batch size , number of epochs and validation_data

```
model = Sequential()

model.add(Dense(512, activation = 'relu', input_shape = (vocab_sz,)))
model.add(Dense(128, activation = 'relu'))
model.add(Dense(6, activation='sigmoid'))

model.summary()
```

```
model.compile(optimizer='RMSProp',loss = 'binary_crossentropy',metrics=['AUC'])
```

```
batch_size = 128
epochs = 8

history = model.fit(tokenized_train, y_train,
          batch_size=batch_size,
          epochs=epochs,
          validation_data=(tokenized_test, y_test)
        )
```

LSTM, GRU, Bidirectional models

❑ LSTM

- ❖ We Make tokenizer.texts_to_sequences to x_train and x_test.
- ❖ We Call Function pad_sequences By Entering tokenizer_seq_train, maxlen, padding and truncating.
- ❖ We Call Function pad_sequences By Entering tokenizer_seq_test, maxlen, padding and truncating.
- ❖ We make modeling in Sequential and add Embedding layer in total number of tokenizer of words index.
- ❖ We add LSTM layer.
- ❖ We add Dense layer by Entering activation in this example sigmoid.
- ❖ We Make modeling compile By Entering optimizer, loss and metrics.
- ❖ We fit model By Entering tokenizer_seq_train, y_train, batch size , number of epochs and tokenizer_seq_test and y_test.

❑ Bidirectional

- We Make tokenizer.texts_to_sequences to x_train and x_test.
- We Call Function pad_sequences By Entering tokenizer_seq_train, maxlen, padding and truncating.
- We Call Function pad_sequences By Entering tokenizer_seq_test, maxlen, padding and truncating.
- We make modeling in Sequential and add Embedding layer in total number of tokenizer of words index.
- We add Dropout Layer
- We add Bidirectional layer.
- We add Dense layer by Entering activation in this example sigmoid.
- We Make modeling compile By Entering optimizer, loss and metrics.
- We fit model By Entering tokenizer_seq_train, y_train, batch size , number of epochs and tokenizer_seq_test and y_test.

❑ GRU

- We Make tokenizer.texts_to_sequences to x_train and x_test.
- We Call Function pad_sequences By Entering tokenizer_seq_train, maxlen, padding and truncating.
- We Call Function pad_sequences By Entering tokenizer_seq_test, maxlen, padding and truncating.
- We make modeling in Sequential and add Embedding layer in total number of tokenizer of words index.
- We add Dropout Layer
- We add Bidirectional layer By Entering GRU layer as input of Bidirectional layer ;

```
model_gru.add(layers.Bidirectional(layers.GRU(128,return_sequences=True)))
model_gru.add(layers.Bidirectional(layers.GRU(64)))
```

- We add Dense layer by Entering activation in this example sigmoid.
- We Make modeling compile By Entering optimizer, loss and metrics.
- We fit model By Entering tokenizer_seq_train, y_train, batch size , number of epochs and tokenizer_seq_test and y_test.