Toxic Comment Classification

Krishna Priya Gajula
Brian Hoang
Anjani Mallampati
Archana Yadawa

Instructor: Professor Sithu Aung

Introduction

Purpose: It is a common struggle and necessity for many social media platforms to be able to identify toxic comments

Kaggle Competition: Toxic Comment Classification Challenge

Dataset: comments from Wikipedia

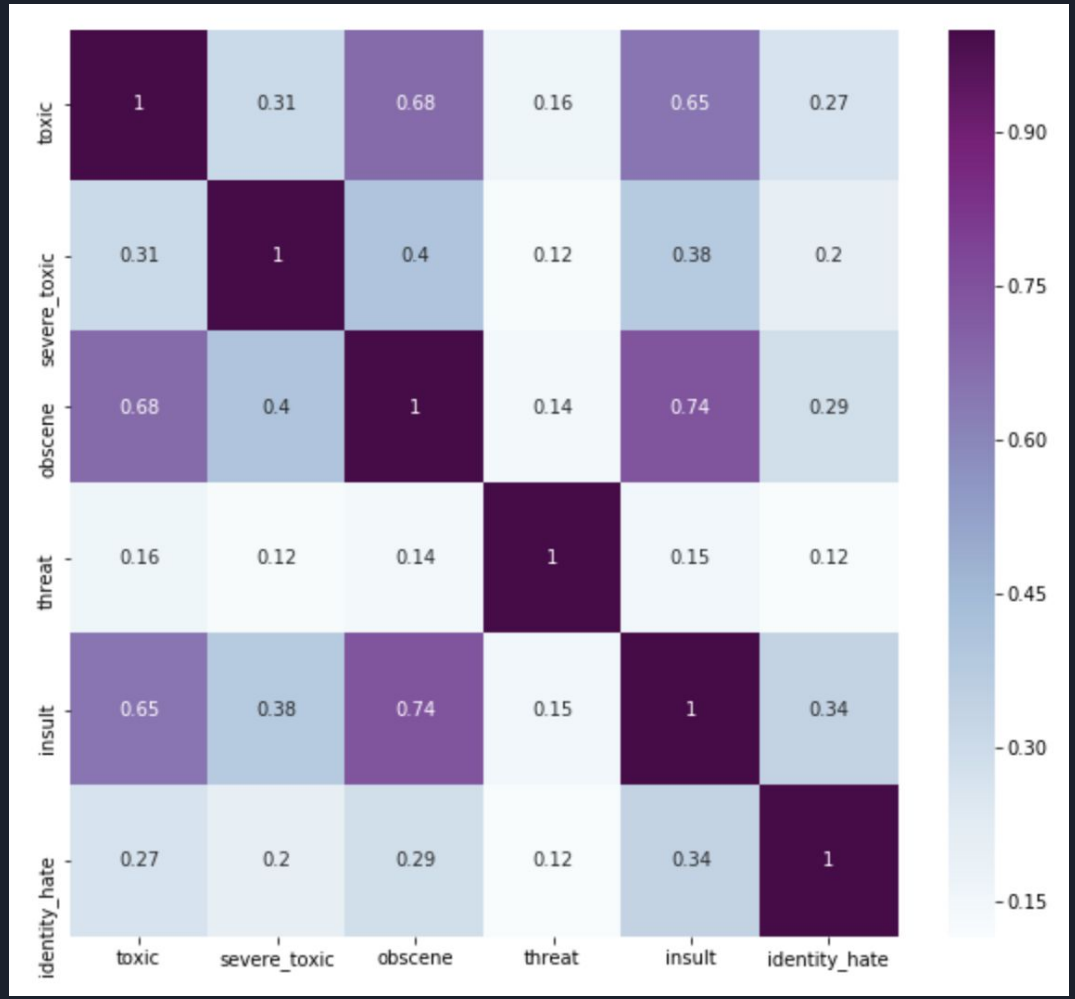
Overview

Part 1: Kaggle Competition

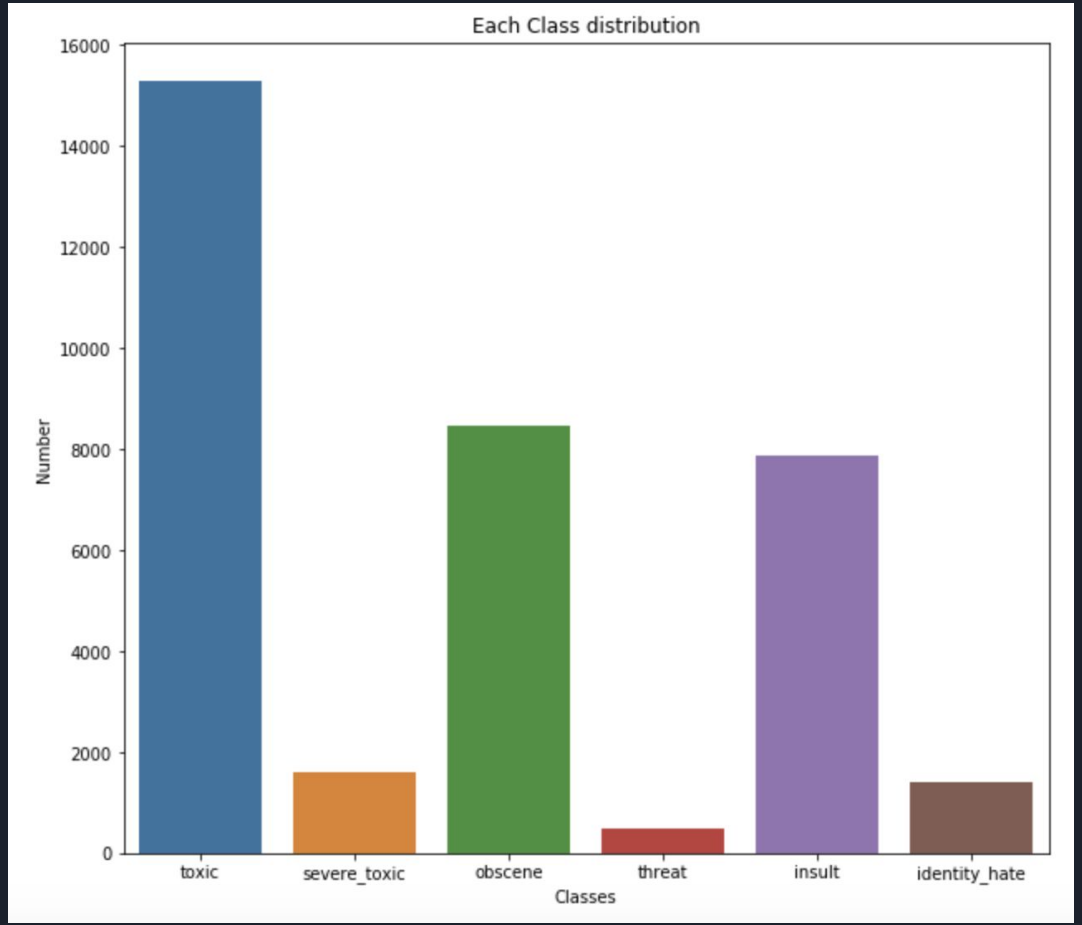
Part 2: Chrome Extension

Part 1:
Kaggle Competition

Correlation Matrix



Class Distribution



Data Preprocessing

(1) Check for null values and remove

(2) Convert into lowercase

```
#covert the text into lower case
X_train_lower = X_train["comment_text"].str.lower()
```

(3) Remove stop words

```
def clean_stopwords(text):
    text = ' '.join(word for word in text.split() if word not in STOPWORDS)
    return text
```

```
checking for missing values in train data
id               0
comment_text     0
toxic            0
severe_toxic     0
obscene          0
threat           0
insult           0
identity_hate    0
dtype: int64
checking for missing data in test data
id              0
comment_text    0
dtype: int64
```

(5) Tokenizer

This class allows to vectorize a text corpus, by turning each text into either a sequence of words or tokens.A filter is applied to remove punctuations.

(6) Keras Pad Sequence

pad_sequences is used to ensure that all sequences in a list have the same length. By default this is done by padding 0 in the beginning of each sequence until each sequence has the same length as the longest sequence.

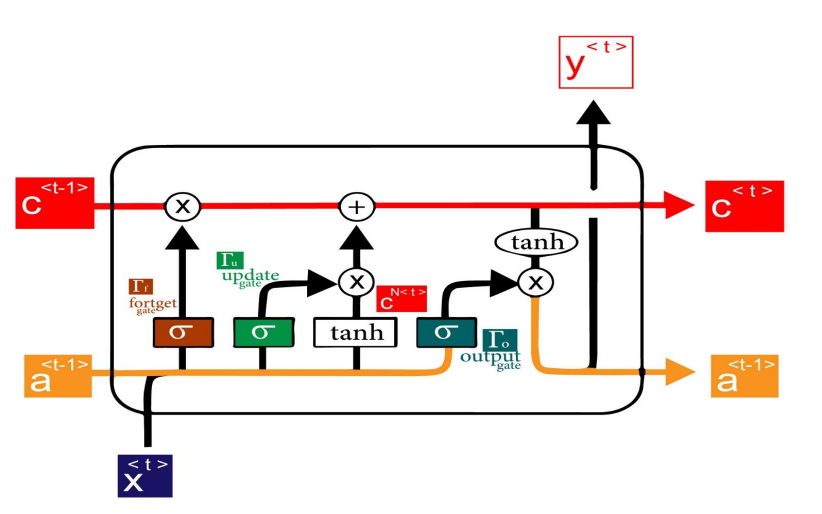
(7) GloVe Embedding(Global Vector for Word Representation)

- GloVe aims to achieve two goals:
  - (1) Create word vectors that capture meaning in vector space
  - (2) Takes advantage of global count statistics instead of only local information
- Unlike word2vec – which learns by streaming sentences – GloVe learns based on a co-occurrence matrix and trains word vectors so their differences predict co-occurrence ratios
- GloVe weights the loss based on word frequency

LSTM (Long Short Term Memory)

- Recurrent neural network that addresses machine learning's inability to evaluate words based on meaning of previous words.
  - Capable of learning long-term dependencies.
- Core structure relies on cell states. Information is regulated through gates.
  - Gates comprised of sigmoid and pointwise multiplication operation.
  - Sigmoid value (between 0-1) determines how much data can be added to cell state.

LSTM (Long Short Term Memory)



1. Sigmoid layer called the “input gate layer” decides which values will be updated.
2. tanh layer creates a vector of new candidate values, C~t, that could be added to the state.
3. Previous steps combined to create an update to the state.

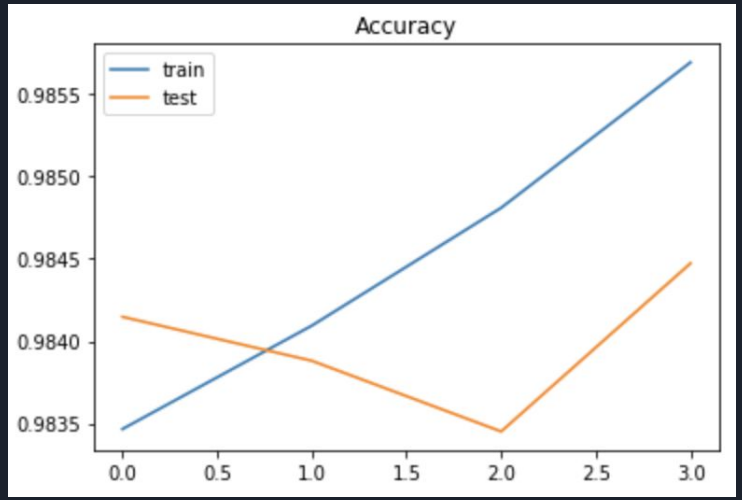
Model

Layer (type)	Output Shape	Param #	Connected to
input_1 (InputLayer)	(None, 100)	0	
embedding_1 (Embedding)	(None, 100, 300)	15000000	input_1[0][0]
spatial_dropout1d_1 (SpatialDro	(None, 100, 300)	0	embedding_1[0][0]
bidirectional_1 (Bidirectional)	(None, 100, 256)	329472	spatial_dropout1d_1[0][0]
conv1d_1 (Conv1D)	(None, 100, 64)	81984	bidirectional_1[0][0]
global_max_pooling1d_1 (GlobalM	(None, 64)	0	conv1d_1[0][0]
global_average_pooling1d_1 (Glo	(None, 64)	0	conv1d_1[0][0]
concatenate_1 (Concatenate)	(None, 128)	0	global_max_pooling1d_1[0][0] global_average_pooling1d_1[0][0]
dense_1 (Dense)	(None, 128)	16512	concatenate_1[0][0]
dropout_1 (Dropout)	(None, 128)	0	dense_1[0][0]
dense_2 (Dense)	(None, 64)	8256	dropout_1[0][0]
dropout_2 (Dropout)	(None, 64)	0	dense_2[0][0]
dense_3 (Dense)	(None, 6)	390	dropout_2[0][0]

Total params: 15,436,614
Trainable params: 436,614
Non-trainable params: 15,000,000

Results

```
Train on 129251 samples, validate on 14362 samples
Epoch 1/4
129251/129251 [==============================] - 2995s 23ms/step - loss: 0.0424 - acc: 0.9835 - val_loss: 0.0407 - va
l_acc: 0.9841
Epoch 2/4
129251/129251 [==============================] - 632s 5ms/step - loss: 0.0403 - acc: 0.9841 - val_loss: 0.0421 - val_
acc: 0.9839
Epoch 3/4
129251/129251 [==============================] - 670s 5ms/step - loss: 0.0378 - acc: 0.9848 - val_loss: 0.0416 - val_
acc: 0.9835
Epoch 4/4
129251/129251 [==============================] - 651s 5ms/step - loss: 0.0353 - acc: 0.9857 - val_loss: 0.0420 - val_
acc: 0.9845
```




```
Loss: 0.04627001273694052
Accuracy: 0.9835087821617776
```

Part 2:
Chrome Extension

Chrome Extension Basics

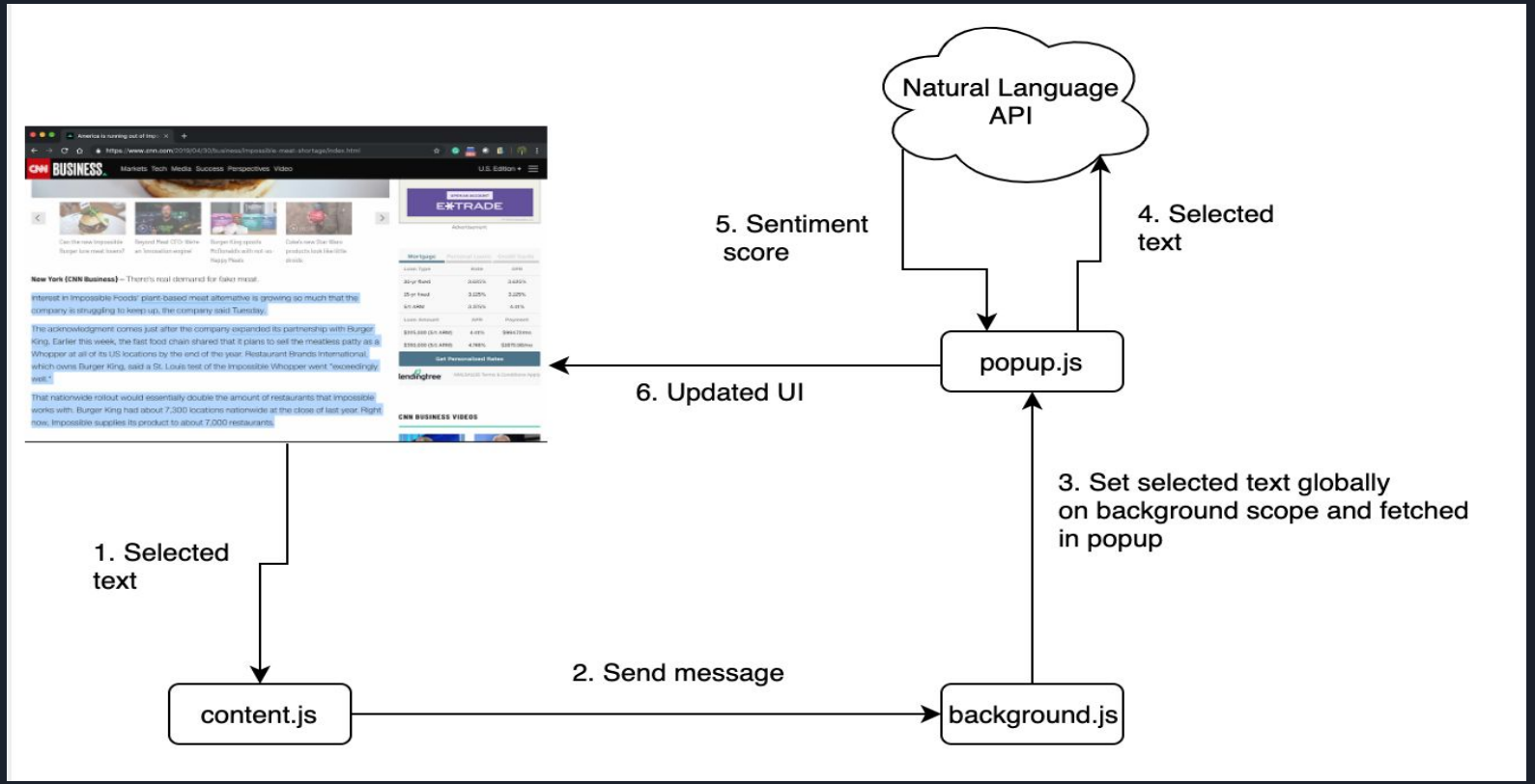
Languages: HTML, CSS, Javascript

HTML - Format of plugin

CSS - Styling

Javascript - used API key for http, calculation of toxic or harmless percentages

Chrome Extension Structure



Existing APIs

Google’s Cloud Natural Language API provides natural language understanding technologies to developers.

- sentiment analysis
- entity analysis
- entity sentiment analysis
- content classification
- syntax analysis

CNN US | World | Politics | Business | Opinion | Health | Entertainment | Style | Travel | Sports | Video Live TV U.S. Edition

Articles

Woman, 88, waved to students from her window for years. They gathered outside her home for one final goodbye

Adam Sandler returns to 'SNL' with a song about how he was fired

References

https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

https://thoughtbot.com/blog/how-to-make-a-chrome-extension

https://medium.com/@subodhgarg/how-to-build-chrome-extension-with-angularjs-googles-natural-language-api-370f9a4953e

https://cloud.google.com/docs/

Thank You!