Rakshak - Saving you from the Mean behind the Screen

TejasMandre[1], Shreyas Vaidya[2], Mehul Chaudhari[3], Jyoti Malhotra[4]
[1,2,3,4] *CSE Department MIT ADT University Pune, India*

Abstract— **With the evolution of technology, Internet has become one of widely used medium of communication. This mainly propagates through social media sites like Instagram, Facebook, etc. and forums like Reddit, Discord, etc. Pandemic has made a significant contribution to this given the increase in "work from home/remote" culture. With increasing convenience comes increasing levels of insecurity people face while using these mediums. Monitoring and detecting online conversations between people are very tough given millions of messages, comments being transmitted every second. Rakshak is a chrome extension being built to solve this problem using machine learning. It protects users from reading hate comments, abuses against them thereby maintaining their self-system and having proper contacts to authorities to prevent online harassment.**

Index Terms: **Machine Learning, Chrome Extension, DevOps, REST API**

I.INTRODUCTION

Social Media platforms have made it easier for offenders to spread hate and abuse while hiding behind their screens. These comments can have adverse effects on lives of people and can lead them to the extent of an unrecoverable mental state. Hate speech is on the rise. According to BBC, online hate speech in the UK and US has risen by 20 percent since the start of the pandemic. The current market situation has led to high unemployment. Online harassment is often ignored but it is a manifestation of real harassment. Everyone has started using the web more and more and this presents people withthe opportunity to spread hate with no accountability at all and that is the reason behind this meteoric rise. Many people on the web are falling prey to this hate and such comments get stuck in their heads. Haters very seldom have a personal reason for their behavior online, but the final aim being to publicly blackface and demean someone remains the same. Being students, our lives revolve around the world of social media and we have all experienced this first hand. For this purpose, machine learning approach is applied which employs several classification algorithms for recognizing hate speech. In this case, a classification recognizes the hate speech and with a script this identified text is made to stand out and the user can now easily distinguish between what has to be read and what is just hate, To address the problem of identifying hate speech, supervised learning algorithm as classification techniques were considered.

II. OBJECTIVE

Develop a chrome a chrome extension capable of detecting harassment in the text as it develops in real time. This increases online safety on highly trafficked social media platforms like Facebook, Twitter, YouTube, and other web-based platforms. Further to ensure coverage of other online services like video games and desktop applications make a PaaS and provide the API to end organizations with the help of which they can develop their own tools to ensure safety and security of their users.

III. DATASET

The Dataset for our problem could have been obtained from different sources like:

- Crowd-sourcing Platforms
- And available datasets

After researching through the methods, we concluded that the use of an available dataset was the way to go forward. A dataset by Jigsaw/ Conversation AI with the following description was chosen:
It contained many Wikipedia comments which had been labeled by human raters for toxic behavior. Each comment in the training set had a bit array of 6 items labelled
0 or 1 where each of them corresponded to the following categories:

- toxic

- severe toxic
- obscene
- threat
- insult
- identity hate

0 meant the absence of a particular category and 1 meant the presence of a particular category.

IV. DATAVISUALIZATION

For better understanding of the dataset, it was visualized using tools like Word Cloud and histograms for every category stated above. These provided us with an insight into the data that was being used. With the help of this information, we could plan the further steps better.

Fig. 1. Word cloud for toxic comments

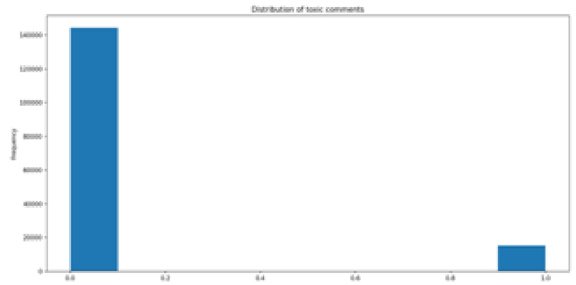


Fig. 2. Histogram for the distribution of Toxic Comments

V. MACHINELEARNING

A. Data Pre-processing

The first step was to fill missing fields with empty strings for consistency. Steps like removing non-ASCII characters, removing letters with numbers attached to them, removing new lines, and conversion to lower case were performed using regular expressions.

B. Vectorization

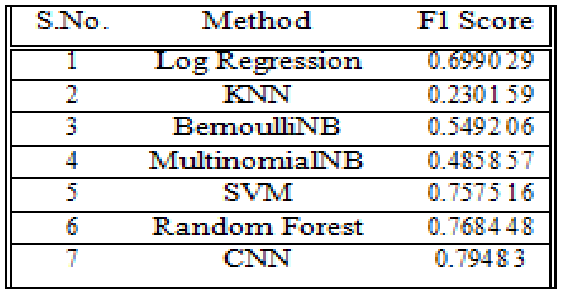
Vectorization helps greatly with feature extraction. Therefore, the text corpus is converted into vectors. Only a certain number of words are kept based on the frequency of their occurrence. The internal vocabulary is then updated with the word counts and the word indices. These indices are then used to define the various documents in the corpus as a sequence of indices. For consistency these sequences are further padded to a certain length.

C. Embedding

In the context of this application, embeddings are used to find out the co-occurrence of different words and the relation between them. Two methods were taken into consideration viz. Word2Vec and GloVe. GloVe was chosen because it not only considers the local context information of the words but also their co-occurrence through the corpus. Each entry in the text file contains a word and its probabilities of co-occurring with other words in the corpus. The thought that goes behind this is that more the probability of two words occurring together, more is the probability of the word pair implying the same meaning. The content of this file is converted in the form of a dictionary. Now the most commonly occurring words are converted into embedding vectors and an embedding matrix is formed. This embedding matrix is important during the training of the model.

D. Selection of Algorithms

We spent our time researching about the sort of algorithm which would be the most suitable for our objective. We calculated the F1 scores of different methods, the values found were as following:



S.No.	Method	F1 Score
1	Log Regression	0.699029
2	KNN	0.230159
3	BernoulliNB	0.549206
4	MultinomialNB	0.485857
5	SVM	0.757516
6	Random Forest	0.768448
7	CNN	0.79483

CNN gave us the highest F1 score and hence was used to build a model.

E. CNNModel

Six similar CNN models were made to get the probabilities of all the Six classes that have already been mentioned. The embedding matrix was added to the sequential model. Since we are predicting the probabilities initially the activation function best suited is Sigmoid. But using the sigmoid through multiple layers gives rise to the vanishing gradient problem and does not give the desired output. Hence, ReLU is used as it helps in back propagation without the vanishing gradient. Sigmoid is only used in the last step to obtain the probability. The model was further configured for training. The binary cross entropy loss function was used which works great when predicting probabilities. Adam optimizer is used, to iteratively update the weights based on the loss function, as it uses averages and converges to reach the minima quickly. Accuracy is used asametric.

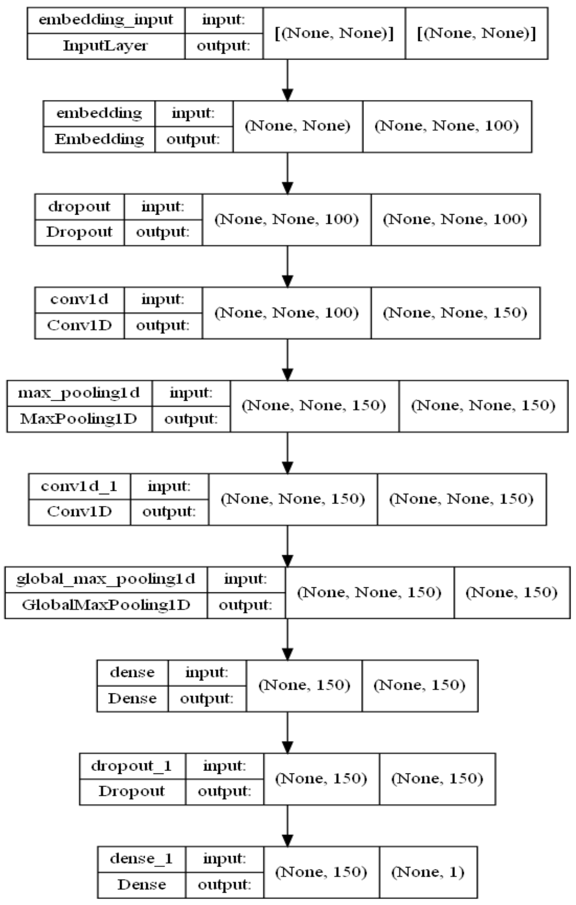

embedding_input | input: | [(None, None)] | [(None, None)]
InputLayer | output:
embedding | input: | (None, None) | (None, None, 100)
Embedding | output:
dropout | input: | (None, None, 100) | (None, None, 100)
Dropout | output:
conv1d | input: | (None, None, 100) | (None, None, 150)
Conv1D | output:
max_pooling1d | input: | (None, None, 150) | (None, None, 150)
MaxPooling1D | output:
conv1d_1 | input: | (None, None, 150) | (None, None, 150)
Conv1D | output:
global_max_pooling1d | input: | (None, None, 150) | (None, 150)
GlobalMaxPooling1D | output:
dense | input: | (None, 150) | (None, 150)
Dense | output:
dropout_1 | input: | (None, 150) | (None, 150)
Dropout | output:
dense_1 | input: | (None, 150) | (None, 1)
Dense | output:

Fig. 3. CNN model plot

F. Training and Saving

The model was trained over 10 epochs and 64 samples are processed before every model updation. After the training is complete the model is dumped into a pickle file. The pickle file is a serialized bit stream which stores the model in a very manageable size.

G. Performance of the model

The performance of the model was measured by plotting the graphs of accuracy and loss through the epochs during the training process.

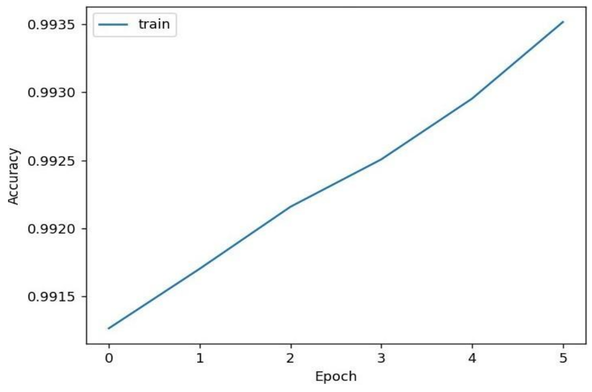


Fig. 4. Accuracy plot

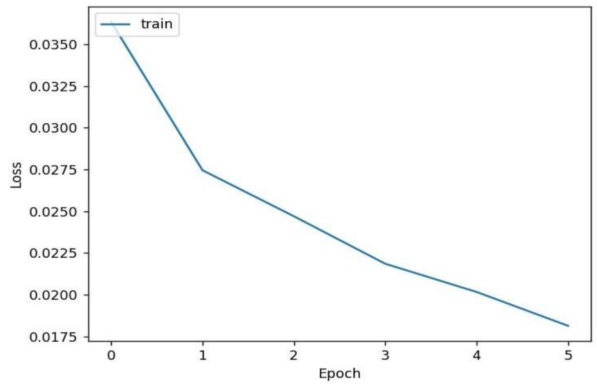


Fig. 5. Loss plot

VI. SERVING ML MODEL THROUGH THEAPI

The pickle file is deserialized by simply loading the file in the API. This gives back the original Python object which was obtained after the model was trained. This object can be used to predict the probability of a particular input belonging to a certain class. The input is first transformed into vectors before passing it to the predict function.

VII. USAGE OF REST API BETWEEN SERVER AND CHROME EXTENSION

Chrome extensions differ from the apps/web apps. The meta data related to the chrome extension resides in a manifest.json file. Meta data includes site-permissions needed to read the on screen data, description, version information. Sample man-ifest.jsonfile:

```
{
"name": "My extension",
...
"permissions": [ "activeTab"
],
"background": {
 "service_worker": "background.js"
}
}
```

The chrome extension works using the services workers de- fined by the developer that enables it to continue working the background. Content Scripts are used for DOM manipulation. Background file linked to the code has all the HTTP calls required for calling the API.

VIII.CONTAINERIZATION IN CLOUD

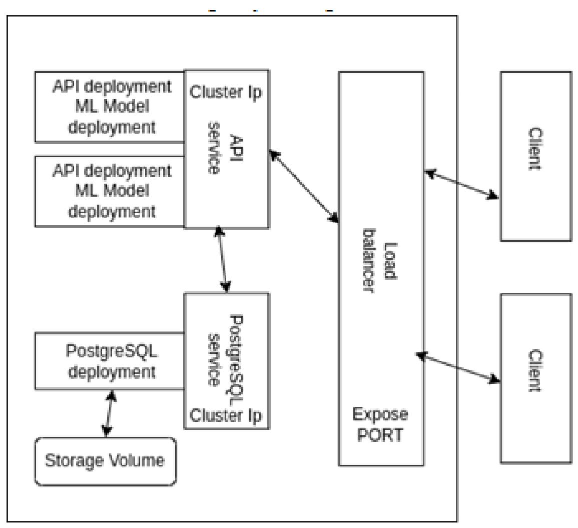


Fig. 6. System Diagram

A.We had targeted was different, we had to apply different logics based on the attributes of elements in the DOM of that website.

B. Filtering the Data

The content read is received as a collection of various HTML Tags. In the filtering process we discard the tags and the content inside these is the data which falls in the region of interest.

C. Sending the data to the back end

We went through various data structures to use to store and send our data efficiently to the back end. Data structures like array of objects, array of strings and JSON strings were considered, out of this we found out that using JSON strings was the most efficient method as the reading process was the fastest. The back end analyzes the data while the front end waits for a response. The response looks like this:

{

identity_hate: false insult: true obscene: true severe_toxic: false threat: false
toxic: true
}

Docker is a go to choice to build images out of the server side code and Kubernetes a container run time environment in which the image is deployed. Given that Rakshak's machine learning processes large number of inputs from multiple clients Kubernetes' pod replication strategy comes in handy when the server needs to scale. Two deployments are done one for the business logic containing the machine learning model instances and another one for the PostgreSQL database to store the user information. High security is needed as the server processes chats of people and any malicious attempt done to steal these needs to be prevented. The above-mentioned deployments run with an internal IP address. A load balancer (like Nginx) placed on top of this takes care of scaling the pods and migrating the traffic to respective services.

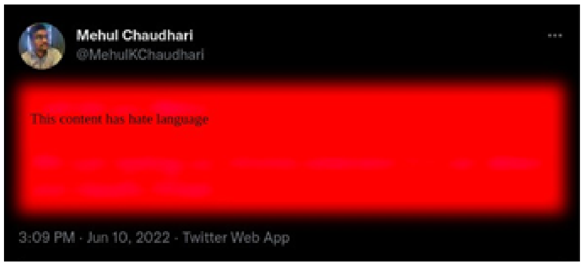
IX. CHROMEEXTENSION

A. Reading the content of a website

To read the data from a particular website, we went through options like web-scrapping and reading it through DOM elements. After researching, we found it was more feasible to use DOM elements than to use web-scrapping because the latter required permissions which could or could not have been procured in time. Since the DOM tree of every website that

D. Response and Application

The response from the backend is received in the form of an object containing the True or False for each of the categories of profanity. The changes are made to the styling of the analyzed data where even one of the values is found out to be true. The change in the styling of an element as a response is done locally. The output looks like this:



Mehul Chaudhari
@MehulKChaudhari

This content has hate language

3:09 PM · Jun 10, 2022 · Twitter Web App

Fig. 7. Output

ACKNOWLEDGMENT

Although a single sentence hardly suffices, we would like to thank Respected Prof. Dr. Shraddha Phansalkar, Head of Department, Computer Science and Engineering, for providing us the right ambience for carrying out this work. We are profoundly indebted to our guide, Prof. Dr. Jyoti Malhotra for her innumerable acts of timely advice, encouragement and we sincerely express our gratitude to her. We express our immense pleasure and thankfulness to all the teachers and staff of the Department of Computer Science and Engineering, MIT School Of Engineering for their cooperation and support.

REFERENCE

[1] Ashwin Geetd'Sa, Dominique Fohr, Irina Illina, Classification of Hate Speech Using Deep Neural Networks

[2] Debasmita Bhattacharya, Ingmar Weber, Thomas Davidson August 2019, Racial Bias in Hate Speech and Abusive Language Detection Datasets

[3] Yoon Kim, Convolutional Neural Networks for Sentence Classification

[4] AntalvandenBosch,BenMiller,FlorianKunneman, WesselStoop, Detecting harassment in real time as conversations develop Volume: Proceedings of the Third Workshop on Abusive Language Online, August 2019

[5] Muhammad Alghobiri, A Comparative Analysis of Classification Algorithms on Diverse Datasets

[6] ZulfadzliDrus, Haliyana Khalid, Sentiment Analysis in social media and Its Application

[7] Thomas Davidson, Debasmita Bhattacharya, Ingmar Weber, Racial Bias in HateSpeech and Abusive Language Detection Datasets

[8] Zeerak Waseem, Thomas Davidson, Dana Warmsley, Ingmar Weber, Understanding Abuse: A Typology of Abusive Language Detection Subtasks
[9] Prateek Mehta, Introduction to Google Chrome Extensions