Toxic Comment Classification (Kaggle Challenge)

Himanshu Upreti
Computer Science Department
IIT Bombay
Mumbai, India
173059004@iitb.ac.in

Digvijaysingh Gour
Computer Science Department
IIT Bombay
Mumbai, India
17305R001@iitb.ac.in

***Abstract*—Social Media has become a vital part of our life. Everyday, we would check messages or posts on Social Medias like WhatsApp, Facebook and many others. But sometimes, we come across an unpleasant entry by few people who are just expressing their distasteful views just to take a jab at you emotionally and it just ruins our day. This happens commonly with celebrities, bosses, instructors and people who are popular among crowd. So inorder to remove such unpleasantries we have come up with toxic comment classification so that user can use this as filter for those messages.**

I. Introduction

Comments are medium for people all over the world to express their opinions on Social Media. User may use it for good purpose like giving a genuine feedback, an intriguing insight about something new or a small jesting comment to make it a light heart moment.But these days , Social Media is becoming a platform for many use cases like Political Campaigning, Promotions of someone's organization or trolling some celebrity and so on. While doing these activities, user tend to post things which undermines one's opinion and sometimes its highly distasteful to give a long lasting impact.

[1]One area of study to tackle this problem is study of negative online behavior. Social Media Organizations are currently struggling to come up with a mechanism to solve this issue. Jigsaw in collaboration with Kaggle has come up with idea to host a contest to design a model to classify such comments and filter them out.

So primarily, dataset which was given on Kaggle had comments of following types :

- **Toxic** - Comments which contain cuss words of mild nature i.e degree of distastefulness is relatively lesser.
- **Severely toxic** - Comments which contain heavy cuss words i.e degree of distastefulness is relatively more.
- **Obscene** - Comments which have content for mature audience primarily consisting of words related to sexual nature or some body part. These are usually disgusting and morally distasteful.
- **Threat** - Comments which are targeted to a person or faction addressing that they will do something malicious to them for e.g Harassment, Rape or Killing etc.
- **Insult** - Comments which are targeted to individuals or organization disrespecting them in some way distastefully for e.g bodyshaming someone or tarnishing someone's image in public platform.
- **Identity Hate** - Comments which are targeted to a person or faction criticizing them in highly distasteful way on basis of race, caste, religion etc for e.g Islam is hated by most of people on world since they are normally associated with terrorism.

So our aim in this paper is to design a model which will calculate rough estimates of probability for each of classes as mentioned above.

II. Background

A. Multiclass Classification

In machine learning the task of classifying the input into more the two classes is known as mutliclass classification or multinomial classification

B. Mutliclass Multilabel Classification

In machine learning the task of classifying the input into more than two classes and an input can belongs to more than one class simultaneously is called mutliclass multilabel classification.

C. Natural Language Processing

The field of machine learning that deals with interaction between computers and human(natural) languages is known as Natural Language Processing. Generally, NLP task includes speech recognition, natural-language understanding, and natural language processing.

D. Neural Networks

Neural Networks are very powerful they can be used in almost every field, classification is one the application Neural Network.

E. Keras

[2] The project is fully implemented on Keras which a high-level neural networks API, written in Python. Keras runs on the top of TensorFlow, CNTK or Theano. The project mainly uses Keras and TensorFlow is used as background for it. TensorBoard is also used for visualizing the output and results.

F. DataSet

The dataset contains large number of Wikipedia comments which have been labeled by human raters for toxic behavior. The data set can be downloaded from https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data. Training data is provided in train.csv which is file of size 26.34MB, similarly test.csv and submission.csv are of size 23.44 MB and 1.4MB. The files should be present in input directory as :

```
"/input/train.csv"
"/input/test.csv"
"/input/sample_submission.csv"
```

directory.

III. Approach

A. Data Analysis

We have used datasets provided by Kaggle. Following is distribution of data w.r.t Toxic Classes

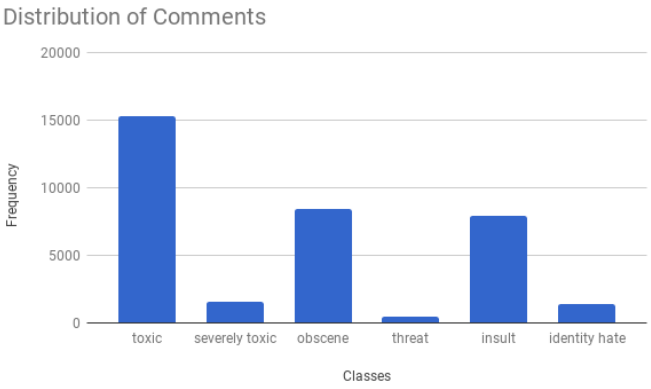

Figure 1. Distribution of Toxic Comments

But we observed that there were total to 157971 comments in all and highest number of toxic comments we got was about 15294 which was about ten times less than overall data. So that's why we added a new class for clean comments. This affects the distribution as visible in III-A

B. Preprocessing of Data

The preprocessing of data is one the time consuming task in our approach. So this is performed only once using

```
python3 preprocess.py
```

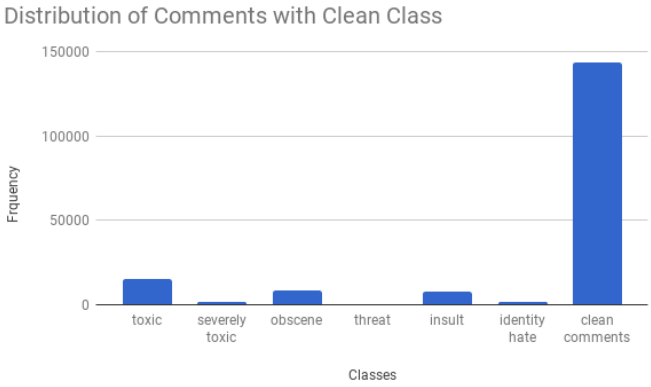

Figure 2. Distribution of Comments with Clean Class

1) Tokenization: Tokenization is process of breaking down a sentence or string into chunks of words. Tokens are used for easing the process of Text to Vector Classification. Tokenizer also makes provision for conversion of text into sequences of indices. Classical working of tokenizer is like,

"May the fourth be with you": Sentence (Input)
[4, 5, 1, 6, 3, 2]: Indices of Tokens (Output)

Keras provides module for Tokenizer.

```
from keras.preprocessing.text import Tokenizer
from keras.preprocessing import sequence
```

2) Embedding: Word Embeddings are set of language modeling and feature learning techniques. It involves mapping of one word in space of one dimension to continuous vector space in much higher dimensions. These mappings are generated using probabilistic models, Neural Networks and explicit representation in terms of context in which the word occurs usually. The project has used pretrained word embeddings of Glove. The GLove embeddings are needed to be downloaded from

```
$ wget http://nlp.stanford.edu/data/wordvecs
/glove.6B.zip
$ mkdir embeddings
$ unzip glove.6B.zip -d /embeddings/
```

The project uses 300d glove pre-trained embeddings, however experiments are also performed using 50d and 100d. The max length of sentences is limited to 200.

IV. Model Architecture

The figure below describe the overview architecture of our approach. The model is defined and trained using after preprocessing

```
python3 train.py
```

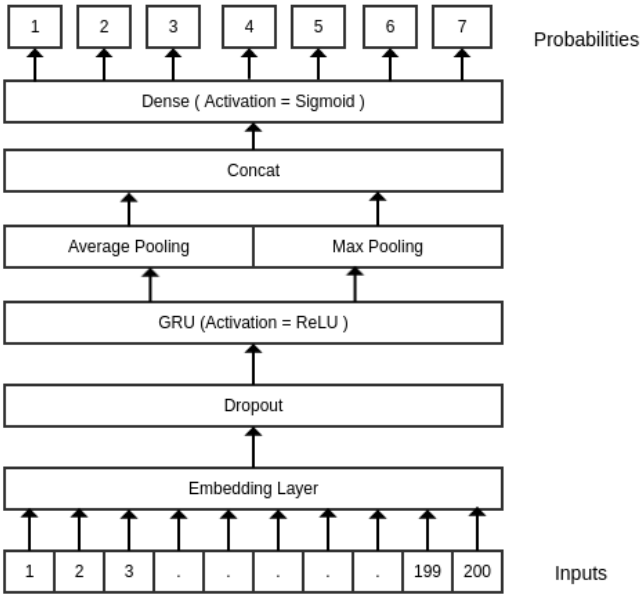

Figure 3. Architecture of Model

1) Gated Recurrent Units (GRU): Gated Recurrent Unit is a variation on LSTM, which is designed to solve the problem of Vanishing Gradient. Key parts of GRU are as follows :

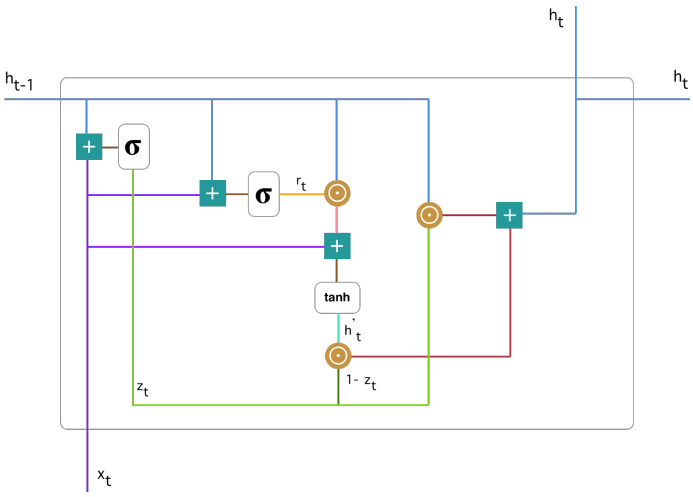

Figure 4. Architecture of GRU [3]

- **Update Gate**: It helps to decide on how much of past information(obtained from previous steps) needs to be propagated to future.
- **Reset Gate** : This gate helps model to decide on how much of past information to forget.

2) Dropouts: Dropout is a regularization technique which is used to reduce chances of over-fitting of Neural Network Models.It prevents Co-Adaptation phenomenon i.e multiple vectors during training co-adapt their weights for perfect fit. It skims off some amount of both hidden as well as visible units of Neural Network.

3) Pooling: When we huge number of features, its computationally intensive to classify and it may also lead to over-fitting. So to avoid this, we divide dataset is partitioned and take aggregate of data points in each partition. This process is known as Pooling. Pooling is useful when features in one region are more likely to be useful in other regions too (like boundary data points).

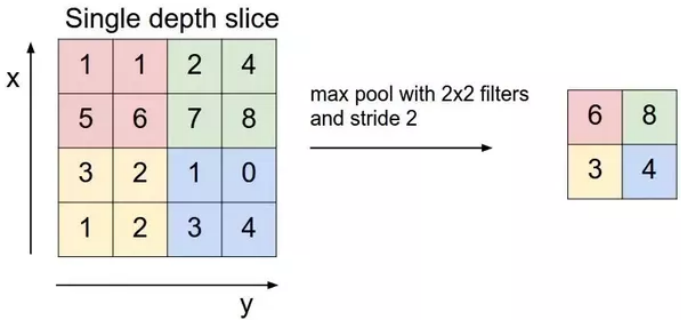

Figure 5. Example of Max Pooling [4]

4) Loss Function: We are using Binary Cross Entropy Loss for training in our project. Cross Entropy Loss is highly useful in use case where classification model is giving output values in between 0 and 1 which is what our model is exactly doing. This loss increases drastically as predicted label diverges from actual label data since it is log loss.

Cross Entropy Loss is defined by following function :

$$f(x) = -(y \log(p) + (1-y) \log(1-p)) \quad (1)$$

V. Prediction

After training the project uses saved model and trained weights to predict the output of comments on "test.csv" file. The output is saved to output directory as

```
$ mkdir output
$ python3 submission.py
```

To view live console view prediction of real world comments use

```
$ python3 predict.py
```

VI. WebDemo

In order to visualize, the live working of project a django app is prepared and present in system directory. The app can be accessed by

```
$ cd ./system/
$ python3 manage.py runserver 0.0.0.0:9000
```

The demo can be viewed by going to 0.0.0.0:9000.

VII. System setup - System Requirements

The project is developed on python 3.6 using anaconda. To install anaconda on your system please refer https://conda.io/docs/user-guide/install/linux.html

```
$ wget https://repo.anaconda.com/archive/
Anaconda3-5.1.0-Linux-x86_64.sh
$ bash Anaconda3-5.1.0-Linux-x86_64.sh
$ cconda install -c conda-forge tensorflow
$ conda install -c anaconda tensorflow-
tensorboard
$ conda install -c anaconda keras
$ conda install -c anaconda h5py
$ conda install -c anaconda django
```

To execute the whole code one can use

```
$ ./execute.sh
```

In order to skip the preprocessing task one can use

```
$ ./kag_train_submission.sh
```

VIII. Results

The live results can be viewed by the use of tensorboard:

```
$ tensorboard --logdir /logs 0.0.0.0
```

After execution of above command go to 0.0.0.0:6006. In our project, we have trained our Neural Network for about 10 Epochs and following are corresponding plots :

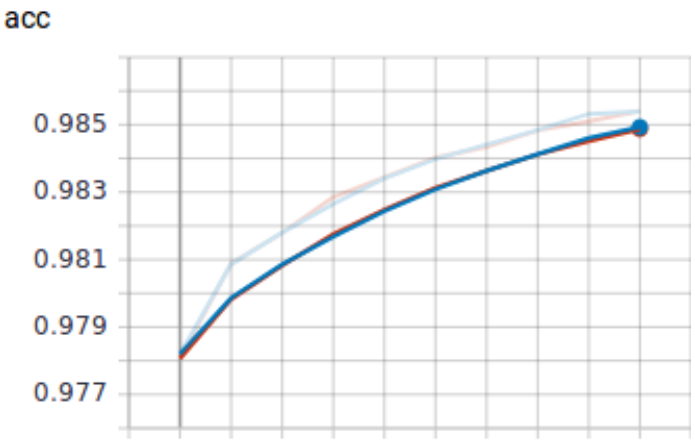

Figure 6. Accuracy plot vs number of epochs

For Validation purpose, we have made use of Train Test Split with 0.95 fraction of dataset for training purpose and 0.05 fraction for test purpose. Here are the corresponding plots,

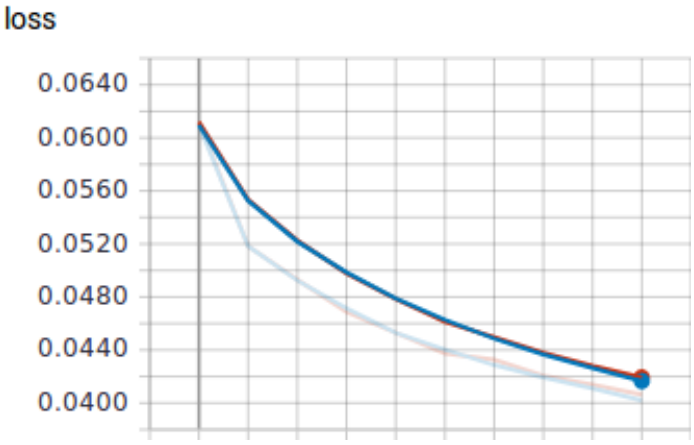

Figure 7. Loss plot vs number of epochs

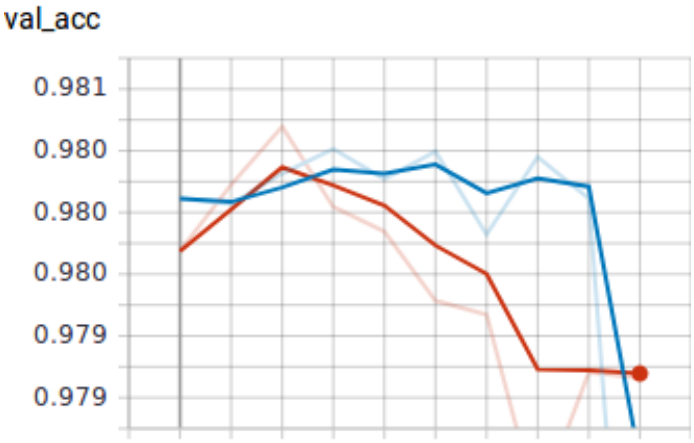

Figure 8. Validation Accuracy plot vs number of epochs

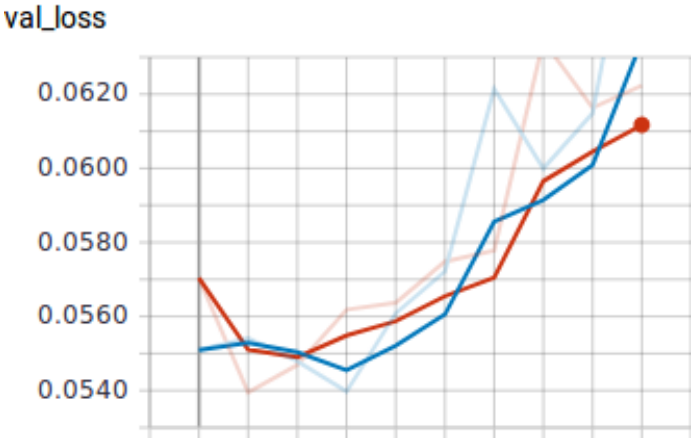

Figure 9. Validation Loss plot vs number of epochs

The training time for each epoch on just CPU powered system is about 1hr. The training time reduced efficiently if trained on GPU. Even with less powerful GPU GT710 the training of an epoch reduces to 35mins. Link to source https://github.com/erh94/toxic_

comment_classification_iml and the contribution by both member of the team is same. Every member was involved in reading paper for solving the problem, understanding keras, and coding. The report and presentation are also prepared by both the members in collaboration.

IX. Observations and Optimizations

- It is observed that the after 2-3 epochs the data is overfitting.
- The best model is saved after 2 epochs.
- The segmenting the preprocessing saves a lot of time, but tokenizer and processed train test data and model needed to be save in `/models` directory.

X. Other Approaches Used

- Multiclass Logistic Regression with TF-IDF Vectorizer with One vs All Approach.
- Neural Network Model with LSTM units as hidden layer that works upon Glove embeddings.

Table I
Comparison Between Different Approaches

Models	Accuracy on Kaggle Submission		
	Score 1	Score 2	Score 3
Logistic Regression	0.9676	0.9654	0.9645
Dense LSTM model	0.9685	0.9668	0.9656
GRU Model	0.9814	0.9820	0.9776

References

[1] Kaggle toxic comment challenge. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge.
[2] Kera documentation. https://keras.io.
[3] Gru - towards data science. https://towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.
[4] What is max pooling in convolutional networks - quora. https://www.quora.com/What-is-max-pooling-in-convolutional-neural-networks.