Identification and Classification Of Toxic Comments

Abhay Kumar Singh
Dept. of Computer Science & Engg.
Texas A&M University
College Station, Texas, USA
abhay@tamu.edu
UIN - 130004782

Mukund Srinath Heragu
Dept. of Computer Science & Engg.
Texas A&M University
College Station, Texas, USA
msh_tam@tamu.edu
UIN - 128003513

Rizu Jain
Dept. of Computer Science & Engg.
Texas A&M University
College Station, Texas, USA
rizujain@tamu.edu
UIN - 430000753

Vindhya Ningegowda
Dept. of Computer Science & Engg.
Texas A&M University
College Station, Texas, USA
vindhya.dn@tamu.edu
UIN - 930001925

***Abstract*—Due to the unprecedented growth of social media, the manner of communication has witnessed drastic changes. Although it has enabled a more connected and informed world, it has also engendered a new phenomenon: toxicity. An open platform to produce,comment and share content has enabled users to participate in hate speech: peddling, sexist, xenophobic, and all-around negative comments. To curb this challenge, we propose to build a model that identifies whether a comment on some online conversation platforms could be perceived as toxic to a discourse. We train our model using the data set provided by Wikipedia where comments are labelled using multiple toxic levels. Through the use of binary classification, we intend to detect whether a comment is toxic. We also perform multi-label classification using a few deep learning algorithms to label the comment with its corresponding toxicity level. Solving this issue is important for readers and moderators to remove a degree of profanity from online discussions.Using this approach, we hope to provide users with a toxic-free internet where social media platforms and other websites can be viewed by not only people from all over the world but also people of all ages of society.**

Keywords: ***Natural Language Processing, Binary Classification, Multi-label classification, Toxic Comment Detection***

I. Introduction

A. Motivation

With the onset of exclusive online platforms for communication and exchange of information the room for hate speech and online abuse and harassment has also been made. Repercussions can be witnessed in the form of decline in freedom of speech due to constant fear of abuse. This generation of toxicity on online communication platforms is the motivation behind the project to identify the type and level of toxicity in the user comments.

B. Problem Statement

The objective of the project is to implement various classification models to address the detection of hateful comments from the data-set and classify them into certain predefined levels of toxicity. We explore different preprocessing and NLP techniques to feed the data into the classifiers. We perform binary classification for detection of a toxic comment and multi-label classification to get the correct classification from pre-defined levels of toxicity.

II. Literature Review

Toxic comment detection is of grave importance, as it can curtail the adverse effects of harmful comments on users. Apart from detecting online toxic comments at scale (Wulczyn et al., 2017), related research includes the investigation on online harassment detection(Yin and Davison, 2009; Golbeck et al., 2017), cyberbullying(Zhong et al., 2016; Hee et al., 2015), offensive language (Chen et al., 2012; Xiang et al., 2012), abusive language (Park and Fung, 2017; Mehdad and Tetreault, 2016) and hate speech (Badjatiya et al., 2017; Davidson et al., 2017; Gamback and Sikdar, 2017; Vigna et al., 2017; Warner and Hirschberg, 2012). Similar methods can be applied for a variety of tasks even though every field has their own definition of classification. In our work we will focus on toxic comment detection.

Google and Jigsaw launched a project called Perspective (Hossein Hosseini et al., 2017), which automatically detects online insults, harassment, and abusive speech. A main limitation of these models is they are not able to determine degrees of toxicity. Besides traditional binary classification, a majority of related research deals with different aspects of toxic language “sexism” (Jha and Mamidi, 2017; Waseem and Hovy, 2016) and “racism” (Waseem, 2016; Kwok and Wang, 2013; Greevy and Smeaton, 2004). These tasks are multi-class problems, where each sample belongs to only one of the output classes. This is remarkable considering toxic comments seen in real world data can often be seen as multi-label problem with user comments fulfilling different predefined rules at the same time. Our work therefore

concentrates on multi-label classification containing six different forms of toxic language as specified in our dataset.

Toxic comment identification and detection is a supervised machine learning task which can be done using manual feature engineering (Robinson et al., 2018; Kennedy et al., 2017; Samghabadi et al., 2017) or using neural networks(Georgakopoulos et al., 2018; Ptaszynski et al., 2017; Pavlopoulos et al., 2017; Badjatiya et al., 2017; Vigna et al., 2017; Park and Fung, 2017; Gamb¨ack and Sikdar, 2017). While in the former method, manually extracted features are transformed into input vectors and directly used for classification. Whereas in the latter approaches are supposed to automatically learn features above those input features. Neural network approaches have been found to be more effective for learning (Zhang and Luo, 2018). Our implementation will focus more on neural networks(LSTM and CNN) and shallow learners such as Logistic regression on bag-of-words model.

III. Background Review

A. Data Preprocessing

Text Processing in Natural Language Processing is very crucial and is needed to convert the data set into such a form that is acceptable, predictable and analysable for the further modelling.
The data set at the first consists of text that will be in its natural human understandable format, i.e. in the form of sentences, paragraphs etc. Some cleaning is required to break the data in a machine understandable format.

1) Panda Data frames: To manipulate data with ease, special data structure available through python packages, Pandas DataFrames are available. This makes reshaping of the data set quite readily.

2) Case Conversion of Text: When the data set is not very large, to maintain consistency in the output expected, lower-casing of all of the data set is very effecting in text processing. Variation in different types of input capitalisation leads to unpredictable output because the models consider the different cases differently. This is popularly known as sparsity issue and lower-casing the entire data set solves it.

$$CANADA -> canada$$
$$CaNaDa -> canada$$
$$CanaDa -> canada$$

3) Tokenization: Tokens are words in a sentence, and sentences in a paragraph. Splitting a text into its tokens is the process of tokenization. This also removes punctuation. It must be noted that tokenization is language specific.

I am Good - ['I', 'am', 'Good']

4) Stop Words Removal: In the vocabulary that we build from the data set, we may find some extremely common words that might be insignificant for the task of the user. These should be excluded from the vocabulary entirely. Following are few examples of stop words in English language -

an, and, are, as, at, be, by, for, from, to, was, with
a, has, he, in, is, it, its, of, on, that, the, were, will

5) Lemmatization: To convert words in the vocabulary from their second or third form to their first form variants, lemmatization is used. In some situations, it might be useful to fetch for one of the different forms of words to get a document that contain another word in the set. Lemmatization removes the inflectional endings and returns the base or dictionary form of a word, which is known as the lemma.

	original_word	lemmatized_word
0	trouble	trouble
1	troubling	trouble
2	troubled	trouble
3	troubles	trouble

	original_word	lemmatized_word
0	goose	goose
1	geese	goose

Fig. 1. Lemmatization Example

B. Word Embedding Techniques

Giving a semantic meaning to the numerical representation of words is the crux of word embedding algorithms. A well-chosen representation of the input text might be more informative and may positively impact the overall model performance. The idea is that every semantic feature can be thought of as a single dimension in the broader, higher-dimensional semantic space and words be plotted based on these semantic feature values.

1) TF-IDF: In information retrieval, TF-IDF stands for Term Frequency - Inverse Document Frequency. It is a numerical value that evaluates the importance of a word in a document and how generic it is across all set of documents or corpus. It evaluates the weight of word *i* in document *j* based on following formula -

$$w_{i,j} = tf_{i,j} log(\frac{N}{df_i})$$

where, $w_{i,j}$ is weight of word *i* in document *j*, $tf_{i,j}$ is number of occurrences of word *i* in document *j*, df_i is number of documents containing word *i* and N is the total number of documents.
This indirectly also handles the stop words because stop words have really high frequency in a document, therefore *tf*

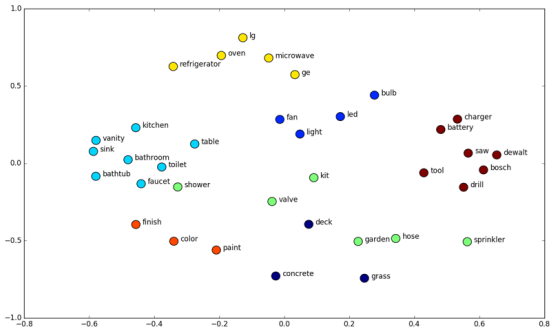

Fig. 2. Visualisation of Word Embedding of vectors

will be higher for it, but since it is also generic, it will be present in all the documents and thus making *idf* equals to 0.

2) Word2Vec: To learn the word vectors, a framework called word2vec can be used. From a large corpus of text, every word is represented by a vector in the fixed vocabulary. By going through each position in the text, which has a centre word and context words. The probability of word given a context or vice-versa can be calculated by using the similarity of the word vectors. In the model, by continuously adjusting the word vectors the probability can be maximised.

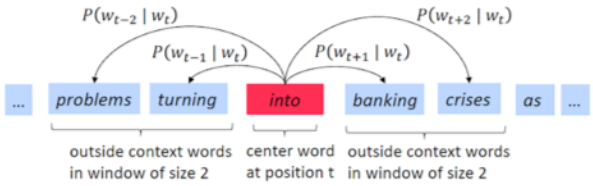

Fig. 3. Example windows and process for computing probability of a word in Word2Vec

3) GloVe: This vector set is provided by Stanford NLP team. Stanford University also provides models from 25, 50, 100, 200 to 300 dimensions of word vectors bases on 2, 6, 42, and 840 billion tokens. To build this embedding, the following idea is used: if two words are co-exist many time, both words may have similar meaning so the matrix will be closer.

4) FastText: It is a framework for learning word representations introduced by Facebook AI Research as an extension to Word2Vec model.
FastText considers morphological structure of each word and considers each word as a Bag of Character n-grams. Taking the word toxic and n=3(tri-grams) as an example, it will be represented by the character n-grams:

$$< to, tox, oxi, xic, ic >$$

and the special sequence

$$< toxic >$$

representing the whole world. The angular brackets indicate the beginning and end of the word. Leveraging n-grams from individual words based on their characters increases the chances for rare words representation since their character based n-grams might occur across other words of the corpus.

C. Binary Classification

We use certain classification techniques in data science to categorize our data into a desired and distinct number of classes where we can assign label to each class. For Binary classification, we do it for 2 classes.

1) Naïve Bayes Algorithm: It is one of the simplest algorithms. Inspired by the Bayes theorem, this belongs to the family of probabilistic classifiers with strong naive assumption of independence among different features.

$$p(c|x) = \frac{p(x|c)p(c)}{p(c)}$$

where *c* is a class, *x* is a word or document, $p(c|x)$ is Posterior probability, $p(x|c)$ is likelihood, $p(c)$ is class prior probability and $p(x)$ is the predictor probability. Ignoring the denominator term which is common for every probability, we can say that -

$$p(c|X) = p(x_1|c)p(x_2|c)p(x_3|c).....p(x_n|c)p(c)$$

It only counts the class distribution hence computational cost is greatly reduced.
Limitation: Naive Bayes can suffer from a problem called the zero probability problem. When the conditional probability is zero for a particular attribute, it fails to give a valid prediction.

2) Logistic Regression: Logistic Regression is actually a classification machine learning algorithm that evaluates the probability of an event existing to a particular class (binary) based on a linear regression model with sigmoid function (sigmoid(α) = $\frac{1}{1+e^{-\alpha}}$) applied over it. For binary classification problems, logistic regression is suitable because its predicted values are defined on a range from 0 to 1.
Figure 4. differentiates between how linear regression and logistics regression is different and applying sigmoid function over a linear regression model provide us the probability of belonging to a particular class.
In logistic regression, there is a threshold probability which defines a particular event belongs to which class and the default value of 0.5.

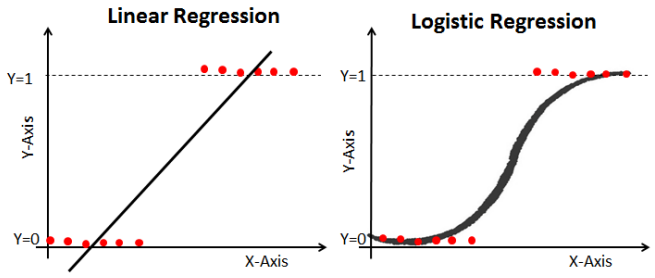

Fig. 4. Difference between Linear and Logistics Regression

3) Support Vector Machines: A Support Vector Machine (SVM) is a non-probabilistic binary linear supervised learning technique where a classifier is defined by a separating hyperplane. Given labelled training data, the algorithm tries to find the hyperplane that will separate the data into 2 parts and also tries to minimize the training error and maximize the distance of hyperplane from the classified points. If

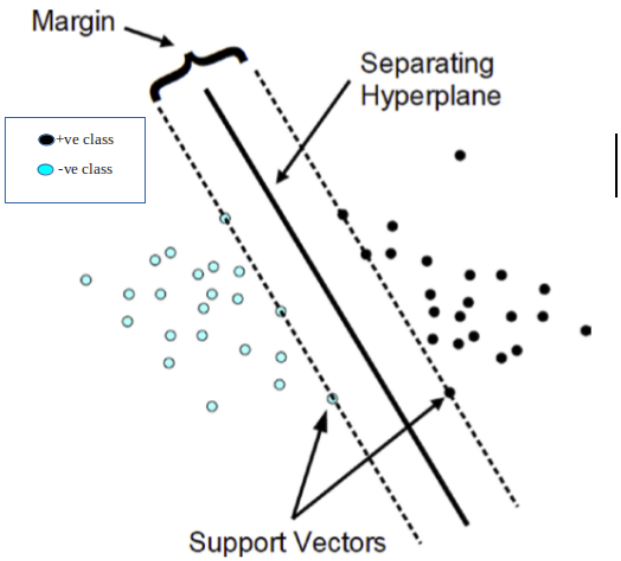

Fig. 5. Example of an SVM

the separating plane is non-linear, then we perform kernel trick where we take the input in the higher dimension and in that dimension, we are able to find a linearly separable hyperplane. We used Polynomial $K(x, y)$ and RBF $R(x, y)$ kernel in this project.

$$K(x, y) = (x^T y + c)^d$$
$$R(x, y) = \exp\left(-\gamma ||x - y||^2\right), \gamma > 0$$

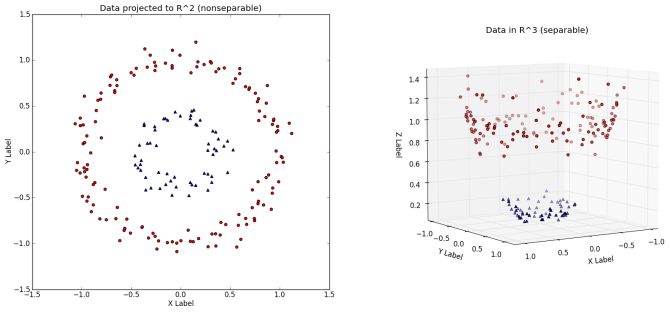

Fig. 6. Visualization of data points in higher dimension that have a separating hyperplane

D. Multi-label Classification

Multi-label classification and the strongly related problem of multi-output classification are variants of the classification problem where multiple labels may be assigned to each instance. Multi-label classification is a generalization of multi-class classification, which is the single-label problem of categorizing instances into precisely one of more than two classes; in the multi-label problem there is no constraint on how many of the classes the instance can be assigned to.
Multi-label classification can be considered as an alternative form derived from the multi-class classification where there are multiple classes (or labels) for all the instances (typical case of the given problem). Multi-class classification refers to the kind of classification problem wherein each instance is only classified into one of the multiple labels possible (mutually exclusive). But in case of Multi-label classification, each instance can be attributed to any number of the given classes (ranging from none, to all of them).

1) LSTM: Long short-term memory (LSTM) is an artificial recurrent neural network (RNN) architecture used in the field of deep learning. Unlike standard feed-forward neural networks, LSTM has feedback connections. It can not only process single data points (such as images), but also entire sequences of data (such as speech or video).
LSTM is a special artificial RNN capable of learning Long-term dependencies. A generic repeating module in LSTM with it's gates is given in figure 7. Below is the description of all 3 gates -

- Forget Gate: This gate Decides which information to be omitted in from the cell in that particular time stamp. It is decided by the sigmoid function.
- Input Gate: Decides how much of this unit is added to the current state. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of importance ranging from-1 to 1.
- Output Gate: Decides which part of the current cell makes it to the output. Sigmoid function decides which values to let through 0,1. and tanh function gives weightage to the values which are passed deciding their level of

importance ranging from-1 to 1 and multiplied with output of Sigmoid.

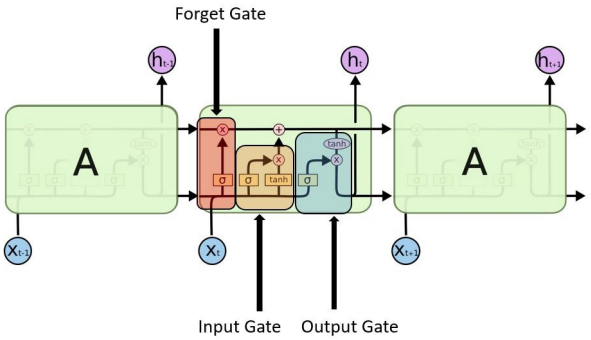

Fig. 7. LSTM with its Gates

2) Bidirectional LSTM and GRU: Bidirectional RNNs can compensate certain errors on long range dependencies. As compared to simple LSTM, bidirectional LSTM takes both directions into consideration. It means that input sequence is processed with both forward and reverse order of words. We average the output of these 2 layers. Similarly, we perform the same thing with GRU also and eventually concatenate both the models to create a dense network.

E. Cross Validation and Hyperparameter Tuning

Cross validation is a technique to identify how well our model performed without even testing it on the test data. It is a way to predict testing accuracy. It involves a step of dividing the training data into training and validation set and making multiple models for all combination of training and validation sets. Average score across all models tell us how well our model is.
Cross validation is also used for hyperparameter tuning. Most of the machine learning algorithms have hyperparameters and the right value of hyperparameter for a model improves its performance. Therefore, cross validation is performed with every combination of hyperparameter and the combination which gives the best validation score is chosen to create a new model on the entire dataset.

F. Performance Measures

There are many evaluation criteria available to evaluate performance of a classification problem. It is mainly evaluated using confusion matrix. Confusion matrix is a way to get true positives, false positives, true negatives and false negatives in both binary and multi-class classification problem performance evaluation. Our model should be able to minimize the false positives, false negatives and maximize the value of true positive and true negatives. Using these values, we calculate multiple score as explained below.

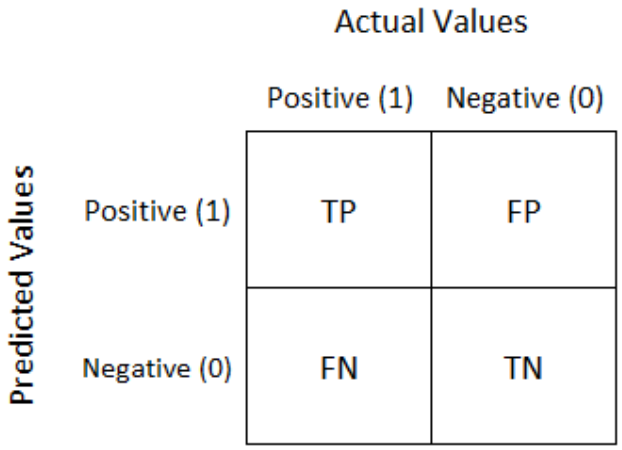

Fig. 8. Confusion Matrix - (TP: True Positive, FP: False Positive, FN: False Negative, TN: True Negative)

1) Accuracy: Accuracy is the number of correct predictions made by the model out of total predictions.

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN}$$

Numerator has correct predictions and denominator has all predictions including incorrect ones.
Accuracy is a good measure when the target variable class is almost balanced. In case of skewed data like having large negative samples, we may get a higher accuracy but our model is trying to learn more true negatives than true positives and thus it is performing good for opposite use case.

2) Precision and Recall or Sensitivity: Precision is the fraction of relevant instances among the retrieved instances. and recall is the fraction of relevant instances that have been retrieved over the total amount of relevant instances.

$$Precision = \frac{TP}{TP + FP}$$
$$Precision = \frac{TP}{TP + FN}$$

Recall provides classifier's performance information with respect to false negatives (how many did we miss), while precision is with respect to false positives(how many did we caught).

3) F1 Score: Our aim is to make precision max without affecting recall and maximize recall without affecting precision. Therefore, we meet at a middle ground and evaluated a new metric called F1 score.
F1 score is the harmonic mean of both precision and recall. Since it is a harmonic mean, if any of the metric is really small, the entire f1 score goes down. Therefore, it carries the

information of both precision and recall, and when both are high, then we get a higher f1 score.

$$F1Score = 2 * \frac{Precision * Recall}{Precision + Recall}$$

4) AUC-ROC: ROC (Receiver operating characteristic) curve is used for visual comparison of classification models. It shows the trade-off between the true positive rate ($\frac{TP}{TP+FN}$) and the false positive rate ($\frac{FP}{FP+TN}$). The area under the ROC curve (AUC-ROC) is a measure of the accuracy of the model. When a model is closer to the diagonal, it is less accurate and the model with perfect accuracy will have an area of 1.0.

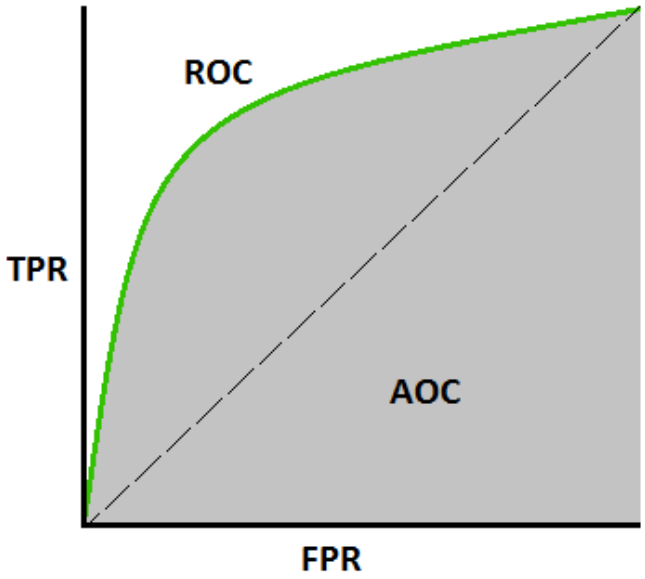

Fig. 9. ROC Curve

G. BERT

BERT, which stands for Bidirectional Encoder Representations from Transformers is, unlike recent language representation models, designed to pre-train deep bidirectional representations from unlabelled text by jointly conditioning on both left and right context in all layers. As a result, the pre-trained BERT model can be fine-tuned with just one additional output layer to create state-of-the-art models for a wide range of tasks, such as question answering and language inference, without substantial task-specific architecture modifications.
For our specific problem, very high ROC-AUC values have been obtained using BERT. This shows us just how powerful BERT really is, and since it is a generalized model, it can be used for a variety of tasks. Such high values indicate that it can be used in production in various industries, especially in customer service.

IV. DATASET

The dataset was taken from a kaggle competition conducted by Jigsaw/Conversation AI. Dataset consists of a comment text in each row with six labels namely - toxic, severe_toxic, obscene, threat, insult and identity_hate. Figure 10 contains few rows of dataset. As we can see in the figure, we comments have toxic multiplicity i.e. having more than 1 labels.

id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
000113f07ec002fd	hey man, i'm really not trying to edit war. it...	0	0	0	0	0	0
0002bcb3da6cb337	cocksucker before you piss around on my work	1	1	1	0	1	0
00040093b2687caa	alignment on this subject and which are contra...	0	0	0	0	0	0
0005c987bdfc9d4b	hey... what is it..\n@ | talk .\nwhat is it.....	1	0	0	0	0	0
0007e25b2121310b	bye! \n\ndon't look, come or think of comming ...	1	0	0	0	0	0

Fig. 10. Input Dataset

1) Data Distribution: Dataset consisted of 159571 comment texts out of which more than 90% of them were non-toxic. Figure 11 which is the distribution of comment texts with respect to each labels shows that out of all labels, maximum comments have toxic label. And due to this, for binary classification, we chose to train our model based on toxic label.

Figure 12 shows the distribution of first 115 comments

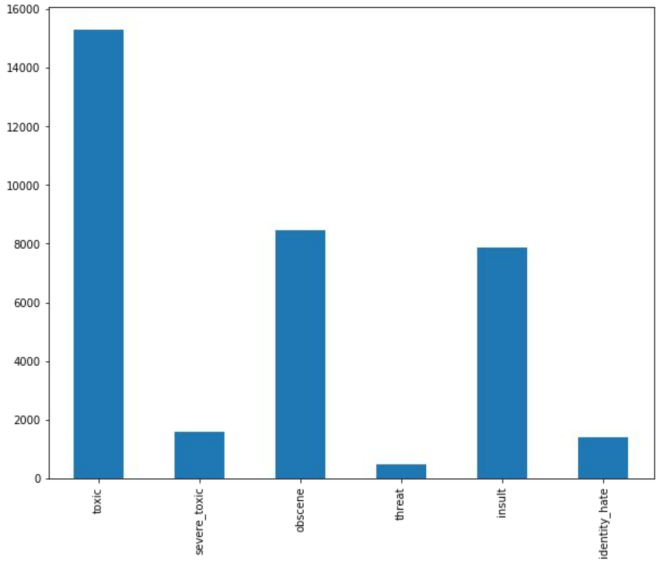

Fig. 11. Distribution of comments with different labels

with some toxic multiplicity. This figure shows that we have very data with higher multiplicity and more data with lower multiplicity.

2) Data Skewness: Since more than 90% of the input data is non-toxic (having 0 for each label), training on such skewed data will give us a model with high accuracy but low f1 score because our model will learn to classify non-toxic comments better than toxic ones. In order to combat this problem, we undersampled the dataset by randomly removing 100 thousand non-toxic comment texts from the dataset. Doing this gave us a balanced dataset which can be used to create our models. Figure 15 shows the before and after distribution with respect to toxic multiplicity.

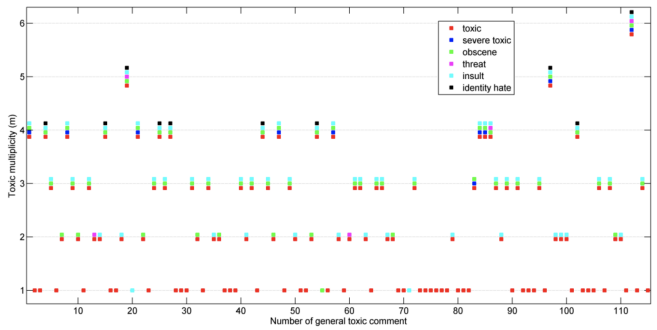

Fig. 12. Distribution of first 115 comments with toxicity greater than 0

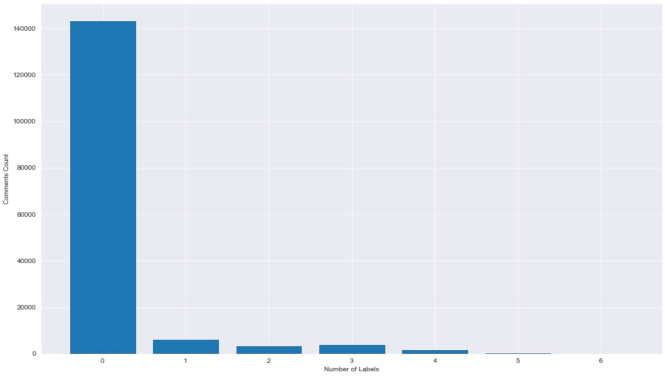

Fig. 13. Distribution before undersampling

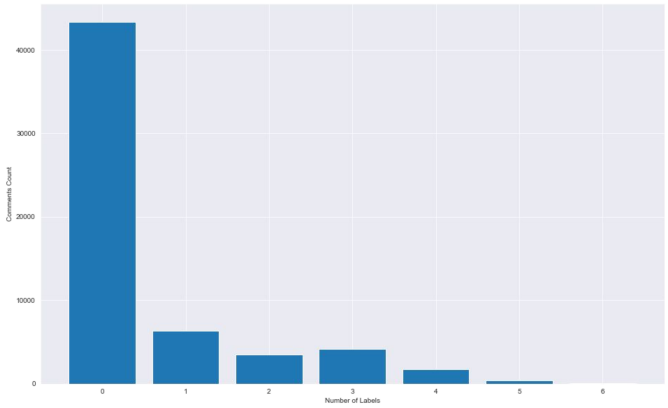

Fig. 14. Distribution after undersampling

Fig. 15. Before and after distribution with respect to Toxic multiplicity

V. Implementation

As part of this project, we implemented Binary and Multi-label classification for toxic comments. Both of the classification are separate and do not add up to the performance of each other. We tried to evaluate the performance of multiple word embedding techniques over binary and multi-label classification.

A. Preprocessing

The very first part of a natural language processing program is preprocessing of the data. Figure 16 shows all steps involved in the preprocessing. We used nltk libraries to perform all the preprocessing part on the text.

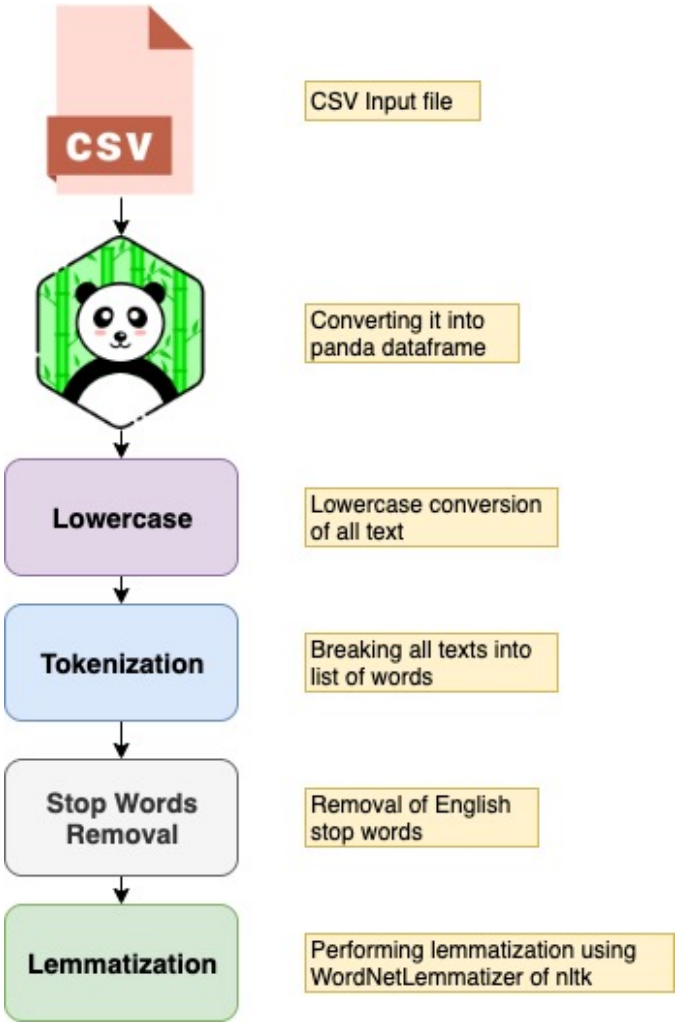

Fig. 16. Data Preprocessing Steps

B. Binary Classification

Since out of all labels, toxic label have highest frequency, we chose toxic label as the output for binary classification. For binary classification, we used TF-IDF and Word2Vec word embedding techniques and trained our model using Naive Bayes, Logistics Regression and SVM algorithms.

1) Training: Following are the tasks performed in training the model -

- We performed train-test split to use two third of data as training data and rest as testing data.
- TF-IDF of the dataset was calculated using nltk library.

Algorithm	Word Embedding	Accuracy	F1 Score	AUC-ROC
Naive Bayes	TF-IDF	0.885	0.776	0.851
	Word2Vec	*Naive Bayes can't take -ve input but Word2Vec has -ve values*		
Logistics Regression	TF-IDF	**0.912**	**0.823**	**0.874**
	Word2Vec	0.859	0.687	0.776
SVM	TF-IDF	0.906	0.799	0.849
	Word2Vec	0.876	0.722	0.796

TABLE I
BINARY CLASSIFICATION RESULTS

Algorithm	Word Embedding	Hyperparameter
Logistics Regression	TF-IDF	Lasso Regularization
	Word2Vec	Ridge Regularization
SVM	TF-IDF	Linear Kernel with C = 1
	Word2Vec	RBF Kernel with C = 10

TABLE II
BINARY CLASSIFICATION BEST HYPERPARAMETERS

- Word2Vec word embedding was created using a pre-trained word embedding data by Stanford.
- Sklearn implementation of Multinomial Naive Bayes was used for Naive Bayes.
- For Logistic Regression, we did hyperparameter tuning with Ridge and Lasso Regularization.
- For SVM, we did hyperparameter tuning with Polynomial and RBF kernels.

2) Testing and Result: For each word embedding technique, we created models using 3 above mentioned algorithms with best hyperparameters. We evaluated Accuracy, F1 score and AUC-ROC for each model. Table I shows the result.
For Logistics Regression, we performed Cross Validation over multiple values of C and regularization techniques. For TF-IDF, Lasso Regression gave us better metrics and for Word2Vec, Ridge Regularization gave us better metrics.
In SVM, we performed Cross Validation over Polynomial and RBF kernel with other hyper-parameters. For TF-IDF, SVM with Polynomial Kernel of degree 1 (Linear Kernel) with C = 1 performed best and for Word2Vec, SVM with RBF kernel. Scores mentioned in the Table I are for respective hyperparameter and Table II contains the best hyperparameters.

3) Analysis: Following are the analysis we did from binary classification -

- For TF-IDF, Logistic Regression with Lasso Regularization performed better than SVM with the F1 score of 0.823 whereas for Word2Vec, SVM with RBF kernel and C = 10 gave us the best F1 score of 0.72.
- Out of all combination of machine learning algorithms and word embedding techniques, Logistics Regression with TF-IDF word embedding performed the best when used with Lasso Regularization.
- While experimenting with the algorithm, we found that models with hyperparameter tuning performed much better than with no hyperparameter tuning. Even though Logistic Regression is a simple and interpretable algorithm as compared to SVM, it performed better than SVM for TF-IDF word embedding. A surprising result we noticed that TF-IDF word embedding performed better than word2vec for both of these algorithms even though it is just bag of words approach.
- We couldn't apply Naive Bayes algorithm to word2vec because naive bayes is a probability based learning algorithm, therefore it only expect positive values in it's input. But word2vec can contain negative values in it, therefore word2vec cannot be applied to Naive Bayes.

C. Multilabel Classification

In order to implement Multi-label classification, we use two methods as given below. Both methods use Deep Learning to classify comments into multiple labels. Using the knowledge from an external embedding can enhance the precision of your RNN because it integrates new information (lexical and semantic) about the words, information that has been trained and distilled on a very large corpus of data. We have used two of the pre-trained word embeddings, GloVe and FastText and made models using LSTM and Bidirectional LSTM and GRU.

- **LSTM:** Our model will have one input layer, one embedding layer, one LSTM layer with 128 neurons and one output layer with 6 neurons since we have 6 labels in the output. Once the words have been tokenized and lemmatized, we proved these sequence of words to an embedding layer (we do not need to one-hot encode the vectors as the output labels are already in the form of one-hot encoded vectors). Next, we use an LSTM layer to process the word embeddings. We use a single dense layer with six outputs with a sigmoid activation functions and binary cross entropy loss functions. Each neuron in the output dense layer will represent one of the six output labels. The sigmoid activation function will return a value between 0 and 1 for each neuron. If any neuron's output value is greater than 0.5, it is assumed that the comment belongs to the class represented by that particular neuron.

- **Bidirectional LSTM and GRU:** Our model will have one input layer, one embedding layer (with weights initialised to pre-trained FastText embeddings calculated on our pre-processed data) followed by a spatial dropout layer to reduce overfitting from correlated group of neurons. The output of dropout layer is fed parallely to bidirectional LSTM (128 neurons) and bidirectional GRU (128 neurons) followed by 1-D convolutional layer which has 64 filters with a kernel size of 2 so that each convolution will consider a window of 2 word embeddings. The four pooling layers (one global and one average for each LSTM and GRU respectively) are concatenated and fed to fully connected single dense output layer with six outputs activated by sigmoid

Algorithm	Word Embedding	Accuracy	AUC-ROC
LSTM	GloVe	0.950	0.927
Bidirectional GRU + LSTM	FastText	0.956	0.949

TABLE III
MULTILABEL CLASSIFICATION RESULTS

function.

1) Training: We used two different pre-trained word embedding files to perform word embedding on our dataset (GloVe and FastText). The GloVe word embedding file contains word vectors of 100 dimensions each and FastText word embedding used pre-trained 2 million word vectors on common crawl of 300 dimensions each. We divide the data using the train-test split and train our model on two-thirds of the data and train it on the remaining one-third of the data. We train the LSTM model on 5 epochs and the bi-directional GRU + LSTM model on 3 epochs. We initially used the mean squared error as our loss function, but we encountered issues that our models would converge to predicting near zero for every single class for every single input. Binary cross entropy loss avoided this problem by better handling each toxicity class independently instead of as a 6-dimensional vector.

2) Testing and Results: The table III shows us the best results obtained using the different word embedding techniques with each of the models.

3) Analysis: We explored multiple models and multiple word embedding techniques to research their applications possibilities and to study their behaviour. The following graphs show us the accuracy and loss values with each increasing epoch. This also ensures that no over-fitting occurs in the models we use. Our CNN model implemented bidirectional GRU-LSTM-pooling using pre-trained FastText embeddings. Bidirectional LSTM and GRU helped capture context in both directions of the comments and FastText embeddings performed better because of the n-gram sub-words information aiding in better classification of toxic words which are rare in general text corpus. The results table shows us that we achieved a high accuracy value for multi-label classification using the RNN and CNN models. We achieved higher accuracy and higher ROC-AUC score with the bi-directional GRU + LSTM model and with using FastText word embedding technique.
We also tried multi-label classification using the binary classifier created for Naive Bayes and Logistics Regression. We applied Binary Relevance and Chain Processing techniques for that, but both of them performed poorly because dataset size of other labels was really less and thus, the binary classifiers were not able to learn them properly.

VI. CONCLUSION

Through this project, we built a number of classifier for identification (binary) and classification (multilabel) of comment text we received from Wikipedia database using different shallow and deep learning algorithms. One of many challenges in this project was the data skewness and comments contains highly non-standard English vocabulary. For binary classification, surprisingly, a bag of word approach of word embedding i.e. TF-IDF performed much better than Word2Vec for both Logistic Regression and SVM. One hypothesis could be that only toxic word existence is enough to categorise a particular statement as toxic instead of understand the depth in the meaning of the sentence.
Bidirectional LSTM and GRU RNNs with pre-trained embeddings performed marginally better. Character level embedding offer a way to out of vocabulary problem predominantly found in online toxic comments.
A further avenue of research to pursue would be deploying BERT (Bidirectional Encoder Representations from Transformers) which shall be a more generic model trained through an extensive text corpus.

VII. CONTRIBUTIONS

Following are the contributions by each member on each of the tasks -

1) Abstract - Abhay, Mukund, Rizu, Vindhya
2) Motivation and Problem Statement - Rizu
3) Literature Survey - Abhay, Mukund, Rizu, Vindhya
4) Data Preprocessing - Abhay
5) TF-IDF - Abhay
6) Word2Vec - Rizu
7) GloVe - Mukund
8) FastText - Vindhya
9) Naive Bayes - Rizu
10) Logistic Regression & SVM - Abhay
11) Cross Validation & Hyperparameter Tuning - Abhay
12) LSTM - Mukund
13) Bidirectional LSTM + GRU - Vindhya
14) Performance Measures - Abhay, Rizu
15) Reference Management - Abhay, Mukund, Rizu, Vindhya

REFERENCES

[1] Dataset from Kaggle Competition: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/overview
[2] Ellery Wulczyn, Nithum Thain, and Lucas Dixon. 2017. Ex machina: Personal attacks seen at scale.
[3] Dawei Yin and Brian D. Davison. 2009. Detection of harassment on web 2.0.
[4] Jennifer Golbeck, Zahra Ashktorab, Rashad O. Banjo, Alexandra Berlinger, Siddharth Bhagwan, Cody Buntain, Paul Cheakalos, Alicia A. Geller, Quint Gergory, Rajesh Kumar Gnanasekaran, Raja Rajan Gunasekaran, Kelly M. Hoffman, Jenny Hottle, Vichita Jienjitlert, Shivika Khare, Ryan Lau, Marianna J. Martindale, Shalmali Naik, Heather L. Nixon, Piyush Ramachandran, Kristine M. Rogers, Lisa Rogers, Meghna Sardana Sarin, Gaurav Shahane, Jayanee Thanki, Priyanka Vengataraman, Zijian Wan, and Derek Michael Wu. 2017. A large labeled corpus for online harassment research.

[5] Haoti Zhong, Hao Li, Anna Cinzia Squicciarini, Sarah Michele Rajtmajer, Christopher Griffin, David J. Miller, and Cornelia Caragea. 2016. Content-driven detection of cyberbullying on the instagram social network.

[6] Cynthia Van Hee, Els Lefever, Ben Verhoeven, Julie Mennes, Bart Desmet, Guy De Pauw, Walter Daelemans, and Veronique Hoste. 2015. Detection and fine-grained classification of cyberbullying events.

[7] Ying Chen, Yilu Zhou, Sencun Zhu, and Heng Xu. 2012. Detecting offensive language in social media to protect adolescent online safety.

[8] Guang Xiang, Bin Fan, Ling Wang, Jason I. Hong, and Carolyn Penstein Rose. 2012. Detecting offensive tweets via topical feature discovery over a large scale twitter corpus.

[9] Ji Ho Park and Pascale Fung. 2017. One-step and twostep classification for abusive language detection on twitter.

[10] Yashar Mehdad and Joel R. Tetreault. 2016. Do characters abuse more than words?

[11] Pinkesh Badjatiya, Shashank Gupta, Manish Gupta, and Vasudeva Varma. 2017. Deep learning for hate speech detection in tweets.

[12] Thomas Davidson, Dana Warmsley, Michael W. Macy, and Ingmar Weber. 2017. Automated hate speech detection and the problem of offensive language.

[13] Bjorn Gamback and Utpal Kumar Sikdar. 2017. Using convolutional neural networks to classify hatespeech.

[14] Fabio Del Vigna, Andrea Cimino, Felice Dell'Orletta, Marinella Petrocchi, and Maurizio Tesconi. 2017. Hate me, hate me not: Hate speech detection on facebook.

[15] W. Lloyd Warner and Julia Hirschberg. 2012. Detecting hate speech on the world wide web.

[16] Hosseini, Hossein & Kannan, Sreeram & Zhang, Baosen & Poovendran, Radha. (2017). Deceiving Google's Perspective API Built for Detecting Toxic Comments.

[17] Akshita Jha and Radhika Mamidi. 2017. When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data.

[18] Zeerak Waseem. 2016. Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter.

[19] Zeerak Waseem and Dirk Hovy. 2016. Hateful symbols or hateful people? predictive features for hate speech detection on twitter.

[20] Irene Kwok and Yuzhou Wang. 2013. Locate the hate: Detecting tweets against blacks.

[21] Edel Greevy and Alan F. Smeaton. 2004. Classifying racist texts using a support vector machine.

[22] David Robinson, Ziqi Zhang, and Jonathan Tepper. 2018. Hate speech detection on twitter: Feature engineering v.s. feature selection.

[23] George W. Kennedy, Andrew W. McCollough, Edward Dixon, A. M. Parra Bastidas, J. Mark Ryan, and Chris Loo. 2017. Hack harassment : Technology solutions to combat online harassment.

[24] Niloofar Safi Samghabadi, Suraj Maharjan, Alan Sprague, Raquel Diaz-Sprague, and Thamar Solorio. 2017. Detecting nastiness in social media.

[25] Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, and Vassilis P. Plagianakos. 2018. Convolutional neural networks for toxic comment classification.

[26] Michal Ptaszynski, Juuso Kalevi Kristian Eronen, and Fumito Masui. 2017. Learning deep on cyberbullying is always better than brute force.

[27] John Pavlopoulos, Prodromos Malakasiotis, and Ion Androutsopoulos. 2017. Deeper attention to abusive user content moderation.

[28] Ji Ho Park and Pascale Fung. 2017. One-step and twostep classification for abusive language detection on twitter.

[29] Ziqi Zhang and Lei Luo. 2018. Hate speech detection: A solved problem? the challenging case of long tail on twitter.

[30] WWW - https://towardsdatascience.com/nlp-for-beginners-cleaning-preprocessing-text-data-ae8e306bef0f

[31] WWW - https://www.datacamp.com/community/tutorials/pandas-tutorial-dataframe-python

[32] WWW - http://blog.kaggle.com/2017/08/25/data-science-101-getting-started-in-nlp-tokenization-tutorial/

[33] WWW - https://towardsdatascience.com/why-do-we-use-embeddings-in-nlp-2f20e1b632d2

[34] WWW - http://web.stanford.edu/class/cs224n/slides/cs224n-2019-lecture01-wordvecs1.pdf

[35] WWW - https://medium.com/datadriveninvestor/classification-algorithms-in-machine-learning-85c0ab65ff4

[36] WWW - https://www.sciencedirect.com/topics/computer-science/logistic-regression-model

[37] WWW - https://towardsdatascience.com/machine-learning-classifiers-a5cc4e1b0623

[38] WWW - https://towardsdatascience.com/precision-vs-recall-386cf9f89488

[39] WWW - https://medium.com/@purnasaigudikandula/recurrent-neural-networks-and-lstm-explained-7f51c7f6bbb9

[40] Bert - https://arxiv.org/abs/1810.04805

[41] WWW - https://medium.com/huggingface/multi-label-text-classification-using-bert-the-mighty-transformer-69714fa3fb3d

[42] WWW - https://medium.com/jatana/report-on-text-classification-using-cnn-rnn-han-f0e887214d5f

[43] WWW - https://stackabuse.com/python-for-nlp-multi-label-text-classification-with-keras/

[44] Piotr Bojanowski and Edouard Grave and Armand Joulin and Tomas Mikolov, Facebook AI Research. Enriching Word Vectors with Subword Information.

[45] Betty van Aken, Julian Risch, Ralf Krestel, and Alexander Loser. Challenges for Toxic Comment Classification: An In-Depth Error Analysis