Analysis, Identification and Multi-label Classification of Toxic Online Comments with Machine Learning

Amrit Gurung

Department of Computer Engineering

Lumbini Engineering College, Pokhara University

Abstract

Discussing things you care about can be difficult. The threat of abuse and harassment online means that many people stop expressing themselves and give up on seeking different opinions. Platforms struggle to effectively facilitate conversations, leading many communities to limit or completely shut down user comments. So classifying the user comments or opinions for toxicity is important to make effective and abuse free online social platforms. In this report, we experiments with three models to classify online comments to various toxic categories. We evaluated our approaches in Wikipedia comments from Kaggle Toxic Comments Classification Challenge dataset. This is multi-labels classification problem. The three models used for experimentations are: 1) Multinomial Naïve Bayes ; 2) Support Vector Machine; 3) Long Short Term Memory(LSTM). Our experiment showed that LSTM outperformed Support Vector Machine slightly by few decimal points. Overall, the average model accuracy for all these models are above 90%.

1 Introduction

Over the decade, there has been a tremendous rise in social networking and social media platforms. People are now able to express themselves and their opinions and also discuss among other via these platforms. However, a conflicts may arise due to the difference in the opinions. Most of the times, this conflicts goes beyond simple differences and results in fight over the social media and often leads to the use of obscene and abusive language. Such conversational toxicity is an issue that can lead people both to stop genuinely expressing themselves and stop seeking others' opinions out of fear of abuse/harassments. The goal of this project will be to use machine learning and deep learning to identify toxicity in text, which could use to help deter users from posting potentially harmful messages, comments, craft more civil arguments when engaging in discourse with others, and gauge the toxicity of other users' comments.

This project aims to implement various machine learning and deep learning algorithms, to tackle the above task. As this is multi-label classification task, so any single comment can have more than one toxic category.

2 Related Work

There has been a lot of researches in toxicity classification and sentiment classification regarding toxicity in past few years, largely in the context of social media data where the Researchers have applied various machine leaning and deep learning algorithms to try and tackle this task.

3 Models

In this project, I implemented and tested three models, out of which two are machine learning model and one deep learning model. These models are discussed below in brief.

3.1 Naïve Bayes Model

Naïve Bayes model is based on Bayes Theorem and is particularly suited for when the dimensionality of the input is high. It simplifies computational complexity. Naïve Bayesian Classifier assumes that the effect of an attribute value on the given class is independent of the value of other attributes i.e. class conditional independent.

Let D be training set of tuples and each tuple is represented by n-dimensional vector, $X = (x_1, x_2, \ldots\ldots, x_n)$ depicting *n* measurements made on the tuple from *n* attributes, respectively, $A_1, A_2, \ldots. , A_n$. Suppose that there are *m* classes, $C_1, C_2, C_3, \ldots. , C_m$. Given a tuple *X,* the classifier will predict that *X* belongs to the class having higher posterior probability conditioned on *X* i.e. Naïve Bayes classifier predicts tuple *X* belongs to class C_i if and only if

$$P(Ci|X) > P(Cj|X) \qquad \text{for } i \le j \le m,\ i \ne j$$

$$i.e.\ P(Ci|X) = \frac{P(X|Ci)P(Ci)}{P(X)} \quad \text{must be maximum.}$$

$P(X) = constant$ for all classes , only $P(X|C_i)P(C_i)$ *need to be maximized.*

And if the class prior probabilities is not known, then it is commonly assumed that that the classes are equally likely,

$$\text{i.e. } P(Ci) = P(C_1) = P(C_2) = \ldots\ldots\ldots P(C_m)$$

and we would therefore maximize $P(X|C_i)$. Otherwise we maximize $P(X|C_i)P(C_i)$.

Naïve Bayesian assumes **Class conditional independence** because it would be extremely computationally extensive to calculate $P(X|C_i)$ given that there are many attributes. Class conditional independence presumes that the attribute's value are conditionally independent of one another, given the class label of the tuple. Thus,

$$P(X|C_i) = \prod_{k=1}^{n} P(x_k|C_i)$$
$$= P(x_1|C_i) * P(x_2|C_i) * \ldots\ldots * P(x_n|C_i))$$

We can easily estimate the probabilities $P(x_1|C_i) * P(x_2|C_i) * \ldots\ldots * P(x_n|C_i))$ from the training tuple. Recall that here x_k refers to the value of attribute A_k for tuple *X.*

To predict the class label for tuple *X*, $P(X|C_i)P(C_i)$ IS evaluated for each class C_I. The classifier predicts that the class label of tuple X is class C_I if and only if

$$P(X|C_i)P(C_i) > P(X|C_j)P(C_j) \quad \text{for } 1 \le J \le m,\ j \ne m$$

In theory, Bayesian classifier have minimum error rate compared to all other classifiers. However, in practice this is not always the case, owing inaccuracies in the assumptions made for its use, such as class-conditional independence, and lack of available probability data.

3.2 Support Vector Machine

Support Vector Machine (SVMs) is method for the classification of both linear and non-linear data. SVM can be used for both classification as well as regression tasks. They belongs to the family of generalized linear models. It uses nonlinear mapping to transform the original training data into

higher dimension and within this new dimension, it searches for linear optimal separating hyperplane (i.e., a 'decision boundary' separating the tuples of one classes from another). With appropriate nonlinear mapping to sufficiently high dimension, data from two classes can always be separated by hyperplanes. Two parallel hyperplanes are constructed on each side of the hyperplane that separate the data. The separating hyperplane is the hyperplane that maximize the distance between the two parallel hyperplanes. An assumption is made that the larger the margin or distance between these parallel hyperplanes the better the generalization error of the classifier will be.

Let the dataset D be given as (X_1,y_1), (X_2,y_2),……., $(X_{|D|},y_{|D|})$, where X_i is the set of training tuples with associated class labels, y_i . Each of the y_i can take one of the two values, either +1 or -1 corresponding to the two classes.

A separating hyperplane can be written as

$$W^T.X + b = 0 \qquad \text{……(1)}$$

Where W is the weight vector, namely $W = \{w_1, w_2, w_3, w_4, ..., w_n\}$; n is the number of attributes and b is a scaler, often refer to as bias. This bias allows us to increase the margin. In the absent of bias term, the hyperplane is forced to pass through the origin, restricting the solution.

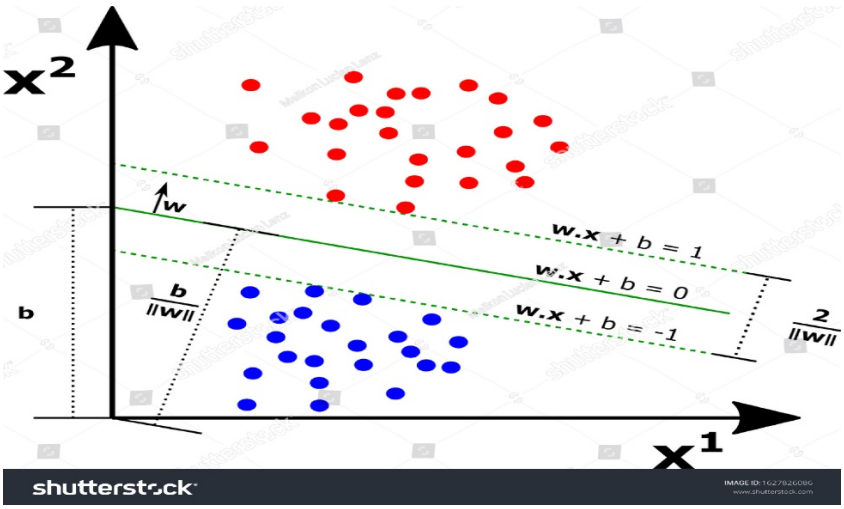

Figure 1 Maximum Marginal Hyperplanes for SVM trained with samples from two classes.

As we can see in the above figure, there are two dotted lines parallel to the hyperplane and passing through some points. These dotted lines are called marginal plane or marginal hyperplane and points through which it passes are called support vectors. We are interested in maximizing the marginal distance, the distance between two marginal plane, because the hyperplane with maximum margin seems to classify the unseen data more accurately hence reducing the classification error as compared to that with lower margin.

The given two parallel marginal planes or marginal hyperplanes can be described by the equations

$$H_1: W^T.X + b = 1 \qquad \text{……(2)} \quad \text{(upper margin plane)}$$

$$H_2: W^T.X + b = -1 \qquad \text{……(3)} \quad \text{(lower margin plane)}$$

That is, any tuples that falls on or above H_1 belongs to class +1, and any tuples that fall on or below H_2 belongs to class -1. They above equations can be combined together and let's write them in inequalities form, we get

$$y_i(W^T.X + b) \geq 1, \forall_i \quad \ldots\ldots\ldots\ldots(4)$$

Any training tuples that fall on marginal hyperplanes H_1 and H_2 satisfy the Eq. (5) and are called the ***support vectors***. If the training data are linearly separable, we can select these hyperplanes so that there is no points between them and then try to maximize their distance. By geometry, we find that the distance between these two marginal hyperplanes is $2/\|W\|$. So the distance between separating hyperplane and each marginal hyperplane is $1/\|W\|$. In order to find the optimal separating hyperplane having a maximal margin, a learning machine should minimize $\|W\|^2$ subject to the above inequality (5). (Note: This optimization problem is solved by the saddle points of Lagrange's Function.)

3.3 Long Short Term Memory

.One of the problems of LSTM is the problem of Long-Term Dependencies. In theory. RNN are absolutely capable of handling such ''long-term dependencies''. Sadly, in practice, they don't seems to be able to learn them. Long Short Term Memory is a special kind of RNN, capable of learning long-term dependencies [3]. LSTM architecture have gates that regulate the flow of information inside cell. Usually these gates are input, output and forget gate- but there are variations to this. The input gate control the flow of input activations, while the output gate controls the flow of output activations. The forget gate scale the cell internal state, and therefore forgetting and resetting the cell's memory.

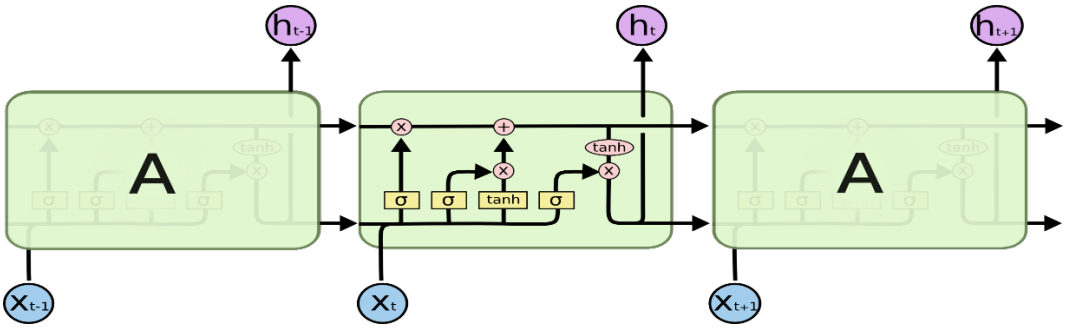

Figure 2. The repeating modules in LSTM contains four interacting layers

4 Experiments

4.1 Datasets

We trained and evaluated our models on Toxic Comment data available in Kaggle. The data contains 6 group of toxic categories. This is a multi-label classification problem, so each comment can belong to zero or more toxic categories. The 6 toxic categories given are

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

Our dataset contains almost 160k rows of online comments, all labeled. The distribution of comments of six toxic categories are shown in the table below.

As mentioned, this is multi-label classification problem so same comment can belong to zero or multiple categories.

Table 1: Distribution of toxic comments based on various categories

Toxic Categories	No. of data/comments with toxic remarks
Toxic	15294
Severe toxic	1595
Obscene	8449
Insult	7877
Threat	478
Identity hate	1405

4.2 Implementation Details

4.2.1 Preprocessing

4.2.1.1 Text Cleaning

Before start building models it is necessary to do some preprocessing on text data. Text data is well known for its unstructured property. It contains unusual texts and symbols that need to be cleaned so that machine learning model can grasp it. The reliability of your model is highly dependent on the quality of your data. There are many text cleansing and preprocessing techniques available but for our experiment we only performed some of them that includes removal of stop words, removing unnecessary symbols, numbers and characters, lowering text, striping extra spaces etc.

4.2.1.2 Padding

The algorithms requires that all the input text have same length. Many text data have varying degree of lengths. A padding length was decided on – and any text shorter than this was padded with zeros. The zeroes are appended at the end of each sentence. An y text longer than the decided length are discarded and removed from the text data. In this experiment padding is especially used in deep learning model and decided padded length is significantly high enough to include all data lengths.

4.2.1.3 Embedding

Word embedding is a popular techniques to convert text into word vectors (numeric vector). The basic requirement of word embedding or vectorized text is:

- It should not result into sparse, since it cost more computational costs.
- It should return most of the linguistic information.

For our models, we have used two word embedding techniques; TF-DF for first two models and Glove vector for last model.

4.2.1.3.1 TF-IDF

After text cleansing, we split our data into independent and target feature vectors. Then we used TF-IDF word embedding on our independent feature (comments). TF-IDF is a numerical statistics that is intended to reflect how important a word is to a document in a collection of corpus.

I used ***TfidfVectorizer*** provided by scikit-learn to vectorize the collections.

4.2.1.3.2 Glove Vectors

For our deep learning model, Glove library was chosen to embed the text in the data. Glove is a pre-trained word vectors which were trained on words from Wikipedia 2014– and Gigaword 5 corpus. The Glove package contains four different versions of dimensions – 50, 100, 200 and 300. The dimension refers to the length of vectors and larger vectors can store more information. The word vectors were trained on four hundred uncased words and were contained in a text file 989 megabytes.

The text file was parsed and each line are converted into numpy array and imputed into dictionary. Next, the dictionary was mapped to the words that were acquired from the tokenization process.

4.2.2 Model building

There are several supervised ML approaches for NLP multi-label classification tasks. There is no universal model to apply. I had to test different models and tune them but at end tuning wasn't necessary because model accuracy was already very high enough to not opt for hyperparameter tuning.

I used ***OnevsRest*** strategy to train our every classifier. The ***OnevsRest*** strategy is a strategy where you train N binary classifiers with one class at a time and leaving rest out. In other words, it consists on a multi-label algorithm that accepts a binary mask over multiple labels. The result for each prediction will be an array of 0s and 1s marking which class labels apply to each row input sample. In document classification One-Vs-Rest treats the classes as mutually exclusive and trains 6 different classifiers in our case, corresponding to each toxic category and converting this problem into a binary classification problem. We applied this method for Multinomial Naïve Bayes and Support Vector Machine. Similarly, for deep learning model, I used 300 dimensional Glove word vectors. I also used max sentence length of 1000

4.3 Results

4.3.1 Evaluation

I trained data on three machine learning models and one deep learning model. The mean accuracy obtained after training these models are given below in the table.

Table 2: Accuracy

Model	Mean Accuracy
MultinomialNB	97.7%
Logistic Regression	98.08%
Support Vector Machine	98.15%
LSTM	98.68%

It is seen that there is no significant difference between the models we trained our data on though Support Vector Machine and LSTM have highest accuracies.

The Confusion Matrix and Classification Report for SVM along with its test accuracy is shown below. There are six confusion matrices in ndarray form for six toxic categories.

```
Confusion matrix for SVM:
            [[[28566   293]
              [1028 2028]]

             [[31543    51]
              [245      76]]
```

```
[[30064   136]
 [525    1190]]

[[31829    12]
 [59       15]]

[[30071   230]
 [704    910]]

[[31593    28]
 [229    65]]]
```

```
Classification Report for SVM:
              precision    recall  f1-score   support

           0       0.87      0.66      0.75      3056
           1       0.60      0.24      0.34       321
           2       0.90      0.69      0.78      1715
           3       0.56      0.20      0.30        74
           4       0.80      0.56      0.66      1614
           5       0.70      0.22      0.34       294

   micro avg       0.85      0.61      0.71      7074
   macro avg       0.74      0.43      0.53      7074
weighted avg       0.84      0.61      0.70      7074
 samples avg       0.06      0.05      0.05      7074
```

Similarly, for our LSTM recurrent neural network, the validation loss and validation accuracies are obtained as below

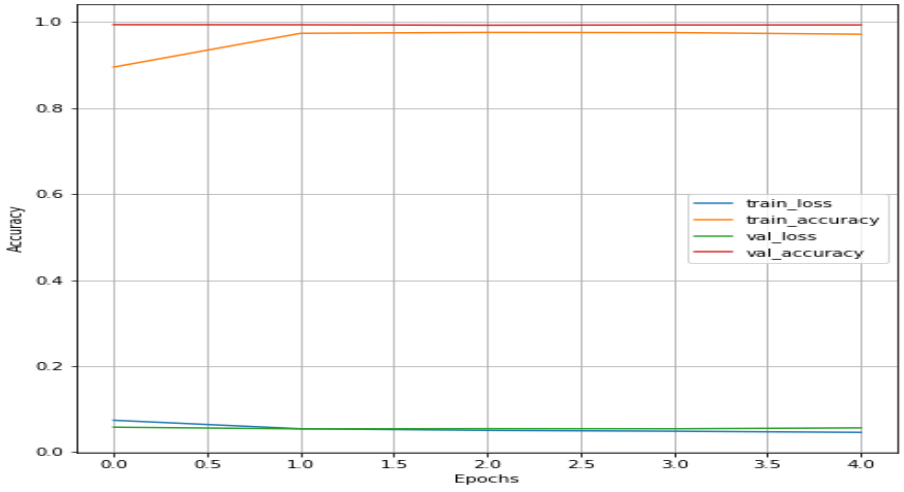

Fig 3. Validation accuracy vs Validation loss

The *val_loss* and *val_acc,* obtained are **0.9716** and **0.056** respectively. This result is already pretty impressive but turning the hyperparameters might still rise accuracy slightly.

4.3.3 Model comparison

Among machine learning algorithm support vector machine is the best model so far with an average accuracy of ~98.15% and LSTM outperforms SVM slightly. A table shows comparison of various machine learning models.

Table 3. Comparison of various algorithms.

Category	Acc NB	Acc LogReg	Acc SVM	SVM advantage to NB	SVM advantage to LogReg
Toxic	0.9499295002349992	0.9565094783017390	0.9586088046373179	0.008679304402318677	0.002099326335578855
severe_toxic	0.9904746984176720	0.9905060316465611	0.9907253642487859	0.000250665831138832	0.000219332602224731050
obscene	0.9721760927463575	0.9776594078019740	0.9792887357042144	0.007112642957856852	0.001629327902240351 7
Threat	0.9976813410621964	0.9977753407488642	0.9977753407488642	9.399968666778946e-05	0.0
Insult	0.9665674447751841	0.9705467648441172	0.9707347642174526	0.004167319442268558	0.000187999373354679
identity_hate	0.9911013629954567	0.9918533604887984	0.9919473601754660	0.000845997180009328	9.399968666767844e-05

There are many more algorithms for multi-label classification like decision trees or random forest. But the results so far are good enough to trust in a model predicting the correct class or classes of online toxic comments.

4.3.4 Testing Model

Now that the model is trained and we have obtained a decent accuracy on average, let's try out toxic online comment which is not part of our dataset, and compare its output label to the category that a human would give it. I am going to test our model on following toxic test.

"You little piece of shit do you really want me to smack on your fucking face goddamn moron "

Our best model, SVM predicted the following class labels for our test data:

```
Prediction for toxic is [1]
Prediction for severe_toxic is [0]
Prediction for obscene is [1]
Prediction for threat is [0]
Prediction for insult is [1]
Prediction for identity_hate is [0]
```

From the predicted class labels above, it is seen that our test data belongs to three categories out of sex available categories. These three categories are *toxic, obscene and insult.*

Similarly, I also used LSTM model to predict class labels on same test data and result of test came out to be very similar to SVM:

```
The given comment is: ['toxic', 'obscene', 'insult']
```

This shows that the toxic class/label/category prediction of our best performing models are comparable.

5 Conclusion

Our best performing models were LSTM and Support Vector Machine with mean accuracy of 98.68% and 98.15% respectively. The more complex and sophisticated model can be built by tuning hyperparameters but the results are good enough to trust our model.

In future, I am hoping to experiment with other deep learning algorithms and compare the performance with current results.

References

[1] Jiawei Han, Micheline kamber, Jian Pei. '' Data Mining concepts and Techniques''. Third Edition , Morgan kaufmann Publisher, 2012. (page no. 350 to 355 and 408 to 415)

[2] Durgesh Srivastava, Lekha Bhambhu. "Data Classification using Support Vector Machine". Article in Journal of Theoretical and Applied Information Technology · February 2010.

[3] Christopher Olah '' Understanding LSTM Networks''. Colah's blog, Posted on August 27, 2015.

[4] Sebastian Laverde, ''Automatic Classification of Sexual Harassment Cases''. A NLP based project by Omdena.

[5] Chu, T & Jue, K. ''Comment Abuse Classification with Deep Learning''.

[6] Kevin Khieu and Neha Narwal. ''Detecting and Classifying Toxic Comments''

[7] Stanford CS244N lectures series on Natural Language Processing with Deep Learning. (lecture 5 and 6)

[8] Analytics Vidhya's article on ''An Intuitive Understanding of Word Embedding: From Count Vector to Word2Vec''.