Inappropriate Question Detection

Chinmai Kaidabettu Srinivas
Department of Computer Science
North Carolina State University
ckaidab@ncsu.edu

Sayali Vinayak Dhavale
Department of Computer Science
North Carolina State University
sdhaval@ncsu.edu

Shrijeet Joshi
Department of Computer Science
North Carolina State University
sjoshi22@ncsu.edu

1 Abstract

Every social media platform and user forum faces the problem of derogatory and inappropriate questions and comments. Many users may feel hurt and insulted due to such actions. They may leave the platform and such a business will lose crucial user base. Hence, such websites and forums have to employ moderators to remove such questions. This is a human work overhead and we feel that it can be automated. Hence, we explore multiple Machine Learning methods to classify questions as appropriate and inappropriate. The methods discussed in the paper are Naive Bayes Classifier, Support Vector Machine, Convolutional Neural Network and Long Short Term Memory Neural Networks.

2 Introduction

2.1 Motivation and Problem Statement

Internet negativity has always been a hot topic. The anonymity and the sense of distance of people's internet presence have encouraged people to express themselves freely. This freedom can sometimes lead to extreme outtakes on others people or the particular topics. Issues like this happen almost all the time, across all platforms of discussion, and the modulators of these platforms have limited capabilities dealing with it. In today's era of internet there is a prevalence of user-driven forums where people learn from each other by asking and answering questions. These forums provide an excellent platform for knowledge sharing where people can ask questions and connect to others who contribute unique insights and quality answers. While this situation contributes significantly to the knowledge of human life, unfortunately it involves enormous dangers, since online texts with high toxicity can cause personal attacks, online harassment and bullying behaviors.Every online QnA platform and especially user driven forums face a challenge when inappropriate and divisive questions are posted. A question containing any of the below characteristics is termed as an inappropriate question:

- Targets specific group of people
- Contains sexual content
- Derogatory remarks

We wish to tackle this problem by automating the task of identifying toxic and misleading content. In this project we aim to experiment with various machine learning techniques to detect inappropriate questions so that they can be handled and removed. This way, the online forums remain a place where users can feel safe sharing their knowledge with the world. This project will be divided into four main tasks : data preprocessing, building different models, training and tuning models, evaluation of models.

2.2 Literature Survey

- Automated text classification is an essential domain of machine learning and in general consists of the following steps: reading a line of text, tokenization, stemming, vectorization, feature selection and machine learning algorithm [1].
- According to [3], tokenization refers to how the text is tokenized and tokens are recorded or annotated, by word or phrase. This is important because many down stream components need the tokens to be clearly identified for analysis. Stemming refers to reducing the words to their stems. This step is the process of conflating tokens to their root form, e.g. connection to connect, computing to compute etc. In feature selection, we construct vector space, which improves the scalability, efficiency and accuracy of a text classifier. In general, a good feature selection method should consider domain and algorithm characteristics. For text classification a major problem is the high dimensionality of the feature space. Hence feature selection is commonly used in text classification to reduce the dimensionality of feature space and improve the efficiency and accuracy of classifiers. In text classification, a text document may partially match many categories. We need to find the best matching category for the text document. The term (word) frequency/inverse document frequency (TF-IDF) approach is commonly used to weight each word in the text document according to how unique it is. In other words, the TF-IDF approach captures the relevancy among words and particular categories. To convert the text data into numerical form, tf-idf vectorizer is used. TF-IDF vectorizer converts a collection of raw documents to a matrix of Tf-idf features. Term Frequency summarizes how often a given word appears within a document. IDF downscales words that appear a lot across documents
- According to [2], CNN can outperform other well established methodologies providing enough evidence that their use is appropriate for toxic comment classification. CNNs consist of convolutional layers, pooling layers, embedding layer and fully-connected layer.
- According to [6], word2vec is a group of models to produce word embeddings that retains the context of words. Word2vec models are shallow, two-layer neural networks constructed based on the idea that similar words would appear in similar positions in the context. Long Short-term Memory (LSTM) models can avoid vanishing gradient problem, where long-term dependencies issues in RNN cause weights from the neural network cannot update its value because the gradient is becoming trivially small.
- According to [7], the usage of GloVe for word embedding improves performance of the model. GloVe is an unsupervised learning algorithm for obtaining vector representations for words. Training is performed on co-occurrence statistics from various corpus and the resulting representations exhibit interesting linear substructures of the word vector space. GloVe is essentially a log-bilinear model with a weighted least-squares objective. The main intuition underlying the model is the simple observation that ratios of word-word co-occurrence probabilities have the potential for encoding some form of meaning. The training objective of GloVe is to learn word vectors such that their dot product equals the logarithm of the words' probability of co-occurrence.

3 Dataset

Dataset was taken from the Kaggle competition. The dataset consists of the following fields:

- Toxic
- Severe_Toxic
- Obscene
- Threat
- Insult
- Identity_Hate

We added a new column called as Inappropriate whose value is 1 if the text belongs to one of the above categories else 0. The training set consisted of around 150k data points and test set consisted of around 100k data points. Since the dataset was huge, we considered only around 55k points from training and around 10k points for testing set.

4 Approach

- Data preprocessing:
  This is the first step in any classification task. We are eliminating the duplicate records or missing values. Next step in to perform tokenizaition. Tokenization is the process of splitting a sentence or text into a list of tokens. Next, we removed all stopwords like "a", "an", "the", etc. Finally we performed lemmatization. Lemmatization is the process of reducing inflectional forms and sometimes derivationally related forms of a word to a common base form.
- Encoding by word2vec:
  word2vec is used to produce word embeddings. These are two layer neural network models and are trained to reconstruct linguistic contexts of words. word2vec takes a large corpus of inputs and produces a vector space of some given large dimensions. Each word is converted into that large dimension vector in the space. The words are placed in such a way that share common contexts in the corpus are located in close proximity to one another in the space.
- Building models:
  We are planning to build different models to perform the detection of inappropriate question detection.
  1. Naive Bayes Classifier
  2. Support Vector Machines
  3. Convolutional Neural Networks
  4. LSTM Model
- Training and tuning models:
  We performed hyper parameter tuning for the neural network models. For CNN, we will be tuning the number of filters in the convolutional layer, kernel size, drop out rate in the drop out layers, batch size and epochs. For the LSTM models we will be performing the tuning on number of neurons in the LSTM layer, dropout rate in dropout layers.
- Performance Evaluation: We will compare the performance of various models that we have trained and choose the best model. We will be using the model with accuracy for performing this particular task.

5 Methodology

5.1 Naive Bayes Classifier

Naïve Bayes classifier is a simple probabilistic classifier based on applying Bayes' Theorem with strong independence assumptions. Due to the independence assumptions, when features are conditionally independent of each other, presence of one feature does not affect other features in classification tasks. Thus, these assumptions make the computation of Bayesian classification approach more efficient. The naive bayes model can be trained very efficiently depending on the precise nature of the probability, by requiring a relatively small amount of training data to estimate the parameters necessary for classification. Since independent variables are assumed, only the variances of the variables for each class need to be determined and not the entire covariance matrix.

An advantage of the naïve Bayes classifier is that it requires a small amount of training data to estimate the parameters necessary for classification. The decoupling of the class conditional feature distributions means that each distribution can be independently estimated as a one-dimensional distribution. This helps alleviate problems stemming from the curse of dimensionality, such as the need for data sets that scale exponentially with the number of features. Another advantage of naive bayes classifiers is that they are highly scalable. Naïve Bayes has been one of the popular machine learning methods for many years. Its simplicity makes the framework attractive in various tasks and reasonable performances are obtained. We have implemented a multinomial naive bayes classifier.

Table 1: Results for Naive Bayes Classifier

Accuracy	93.28
Precision	94.58
Recall	93.05
F-measure	93.81

5.2 Support Vector Machines

Support Vector machines are supervised learning models with associated learning algorithms that analyze data used for classification and regression analysis. A support-vector machine constructs a hyperplane or a set of hyperplanes in a high- or infinite-dimensional space, which can be used for classification tasks. Consider data point as a p-dimensional vector. We want to construct hyperplane to divide the points into two classes. We can intuitively say that the best hyperplane is the one that represents the largest separation, or margin, between the two classes. So we choose the hyperplane such that the distance from it to the nearest data point on each side is maximized. If such a hyperplane exists, it is known as the maximum-margin hyperplane. The larger the margin, the lower the generalization rate of the classifier.

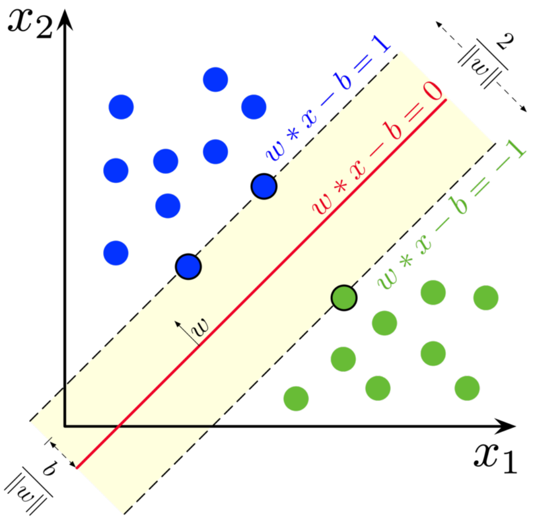


Figure 1: Maximum-margin hyperplane and margins for an SVM trained with samples from two classes. Samples on the margin are called the support vectors.

It is however possible that the original problem might not be linearly separable in the original finite-dimensional space. To make the data linearly separable, we often apply kernel transformation, i.e., we transform the original finite-dimensional space to a higher dimensional space such that the separation would be easier in the new space.

In SVMs we aim for two things:

- Set a larger margin
- Lower misclassification rate

If we increase the margin, we will get higher misclassification rate. If we decrease the margin, we will get lower misclassification rate. C Value indicates how wide the margin separating two classes is. Larger values of C indicate a smaller-margin hyperplane. Smaller values of C indicate larger-margin separating hyperplane. We have chosen C=0.1 in our implemetation.

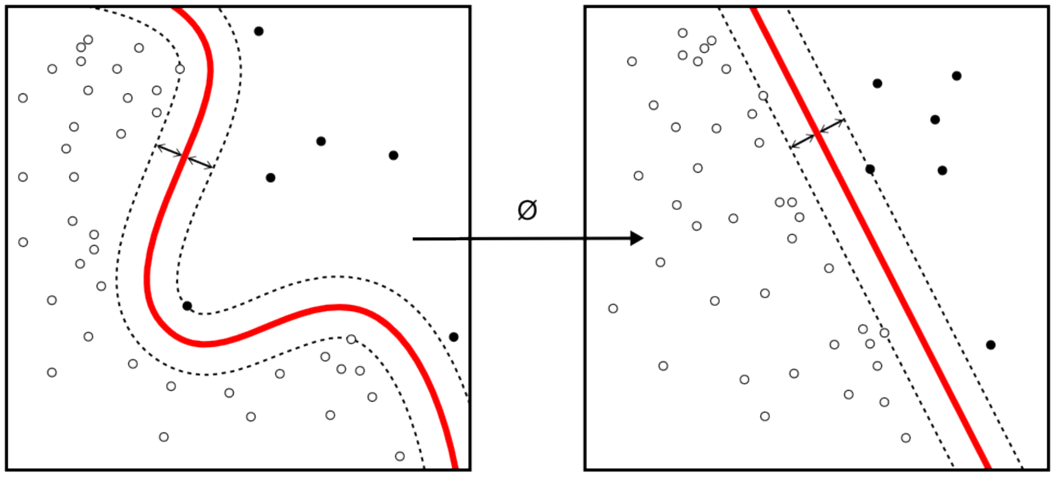


Figure 2: Kernel trick in SVM

Table 2: Results for SVM Classifier

Accuracy	93.9
Precision	93.6
Recall	94.6
F-measure	94.09

5.3 Convolutional Neural Networks

Convolution Nets were initially developed for image processing and acieved great results in terms of recognising and classifying an object. CNNs mainly involve two operations: convolution and pooling. The output of this operation is then passed to a fully connected layers and then modified as per the final task. The general architecture of CNN is as shown in the figure 3

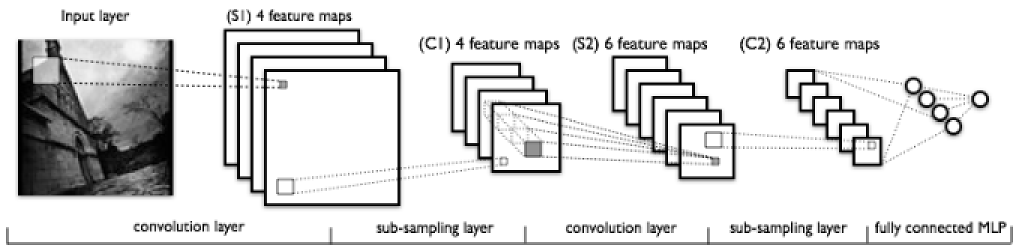
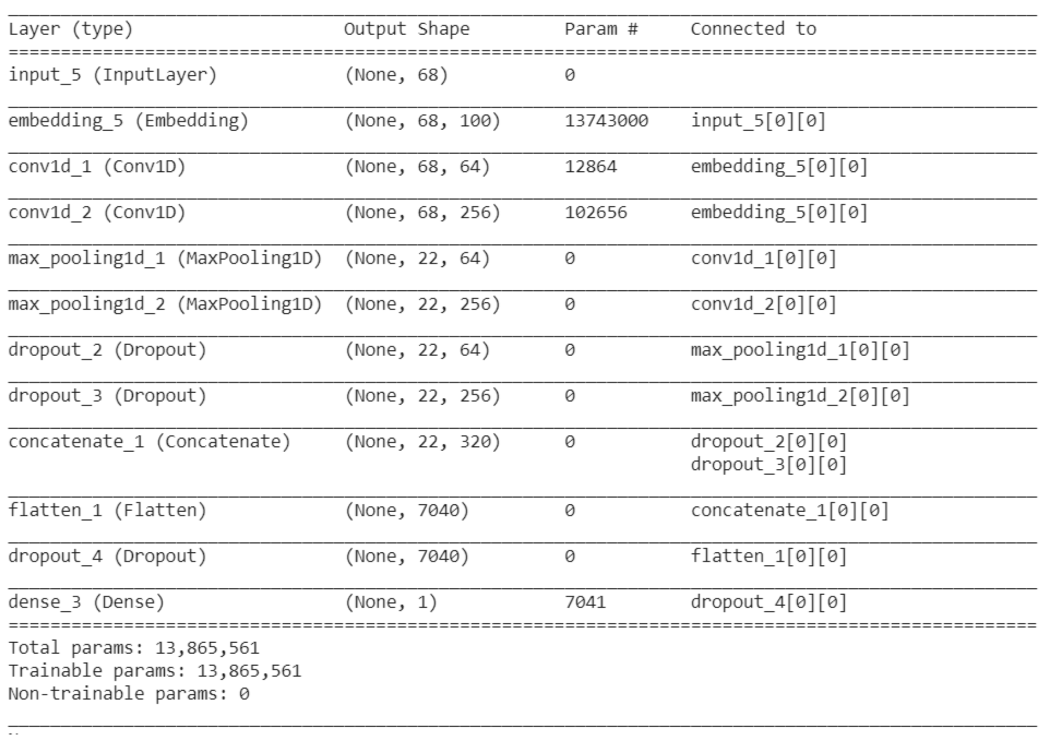


Figure 3: General Architecture of CNN

In case of Natural Language Processing, we have text which is represented in one dimension unlike images which are represented in the two dimensions. Hence in case of text data we will be using one dimensional convolution and pooling operations.

The input for the neural networks should be in the form of sequence and not English words or sentences. In order to convert the English sentences to sequences, we will make use of the glove embeddings which are of 100 dimensions. Once we have the sequence we fix a window size on which the convolutional filter is applied.

Next step is to perform pooling operation, in this step we combine the vector results from different convolution windows into a single dimensional vector. This can be done by taking either maximum or average value observed in resulting vector from the convolutions. This is done with a intuition that this vector will capture most relevant features present in the sentence.



```
____________________________________________________________________________________________________
Layer (type)                     Output Shape          Param #     Connected to
====================================================================================================
input_5 (InputLayer)             (None, 68)            0
____________________________________________________________________________________________________
embedding_5 (Embedding)          (None, 68, 100)       13743000    input_5[0][0]
____________________________________________________________________________________________________
conv1d_1 (Conv1D)                (None, 68, 64)        12864       embedding_5[0][0]
____________________________________________________________________________________________________
conv1d_2 (Conv1D)                (None, 68, 256)       102656      embedding_5[0][0]
____________________________________________________________________________________________________
max_pooling1d_1 (MaxPooling1D)   (None, 22, 64)        0           conv1d_1[0][0]
____________________________________________________________________________________________________
max_pooling1d_2 (MaxPooling1D)   (None, 22, 256)       0           conv1d_2[0][0]
____________________________________________________________________________________________________
dropout_2 (Dropout)              (None, 22, 64)        0           max_pooling1d_1[0][0]
____________________________________________________________________________________________________
dropout_3 (Dropout)              (None, 22, 256)       0           max_pooling1d_2[0][0]
____________________________________________________________________________________________________
concatenate_1 (Concatenate)      (None, 22, 320)       0           dropout_2[0][0]
                                                                   dropout_3[0][0]
____________________________________________________________________________________________________
flatten_1 (Flatten)              (None, 7040)          0           concatenate_1[0][0]
____________________________________________________________________________________________________
dropout_4 (Dropout)              (None, 7040)          0           flatten_1[0][0]
____________________________________________________________________________________________________
dense_3 (Dense)                  (None, 1)             7041        dropout_4[0][0]
====================================================================================================
Total params: 13,865,561
Trainable params: 13,865,561
Non-trainable params: 0

____________________________________________________________________________________________________
```

Figure 4: Our CNN architecture

Next step is to pass the output of pooling layer to the dropout layer. In this layer the neurons are randomly dropped and is done to avoid over-fitting. The output of this is passed to the fully connected layer and then to final output layer which classifies the text as appropriate or inappropriate.

Our model is as as shown in the figure 4

5.4 LSTM

LSTMs are special kinds of recurrent neural networks. LSTMs are capable of remembering information for long periods of time is the speciality of LSTMs.

Figure 5 shows the general architecture of the LSTMs.

LSTMs consists of forget gate, input gate and output gate.

- Forget Gate: This gate is responsible for deciding what information needs to be kept and what information needs to be forgotten. Knowledge from previous state and knowledge from current state is passed from the current input state is passed onto through the sigmoid function, if the output of the sigmoid function is closer to 1 it means to keep that particular information else forget the particular information.
- Input Gate: Input gate contributes to the current state of the system. previous hidden state and current input state is passed on to the sigmoid function and this decides which values will be updated to between 0 and 1.0. Here 1 means it is important else its not important. This gate controls the extent in which the new value flows in the cell.
- Output Gate: This gate decides the values for the next hidden state. This hidden state mainly contains knowledge from the previous inputs. Output of this is the hidden state and controls the extent to which the value in the cell is used to compute the output activation of the LSTM unit.

Figure 6 shows the summary of our LSTM model:

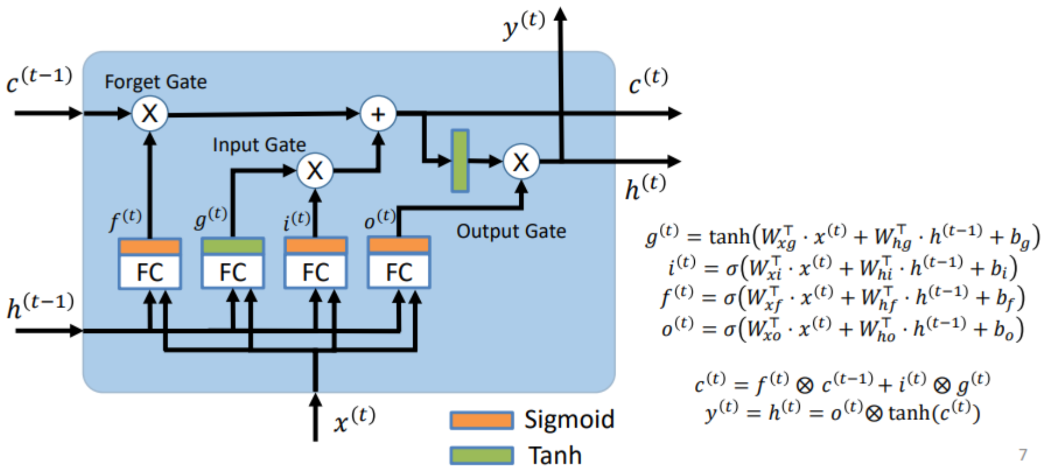
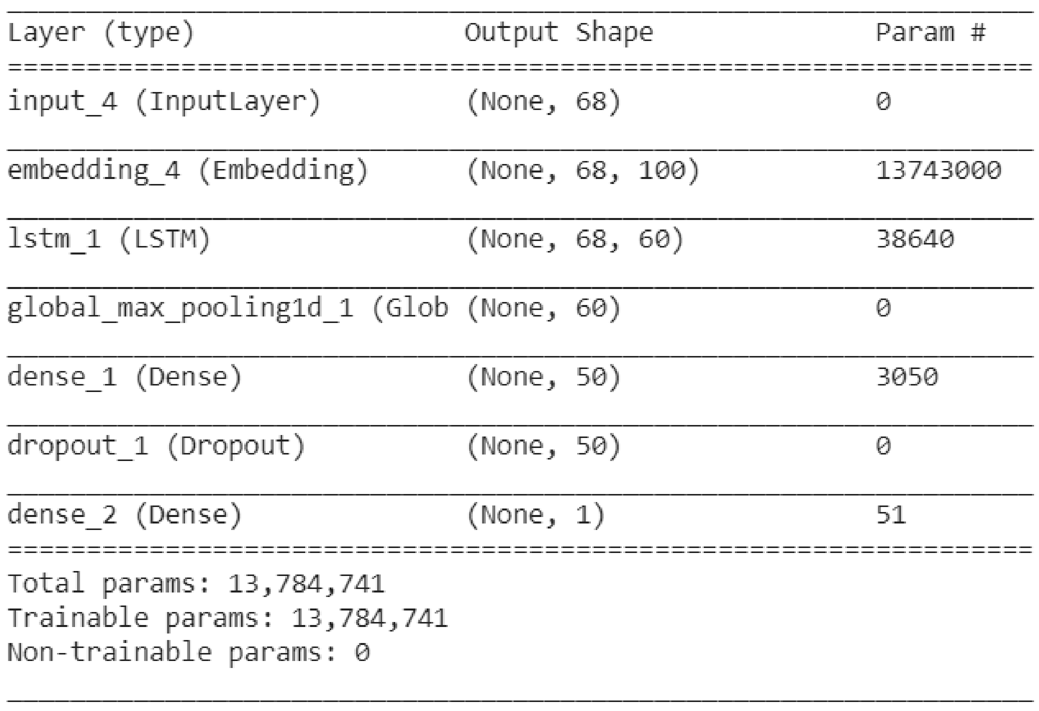

$y^{(t)}$
$c^{(t-1)}$
Forget Gate
$c^{(t)}$
Input Gate
$h^{(t)}$
$f^{(t)}$
$g^{(t)}$
$i^{(t)}$
$o^{(t)}$
Output Gate
FC
$h^{(t-1)}$
$x^{(t)}$
Sigmoid
Tanh
$g^{(t)} = \tanh(W_{xg}^{T} \cdot x^{(t)} + W_{hg}^{T} \cdot h^{(t-1)} + b_g)$
$i^{(t)} = \sigma(W_{xi}^{T} \cdot x^{(t)} + W_{hi}^{T} \cdot h^{(t-1)} + b_i)$
$f^{(t)} = \sigma(W_{xf}^{T} \cdot x^{(t)} + W_{hf}^{T} \cdot h^{(t-1)} + b_f)$
$o^{(t)} = \sigma(W_{xo}^{T} \cdot x^{(t)} + W_{ho}^{T} \cdot h^{(t-1)} + b_o)$
$c^{(t)} = f^{(t)} \otimes c^{(t-1)} + i^{(t)} \otimes g^{(t)}$
$y^{(t)} = h^{(t)} = o^{(t)} \otimes \tanh(c^{(t)})$
7

Figure 5: General architecture of LSTM



```
_________________________________________________________________
Layer (type)                 Output Shape              Param #
=================================================================
input_4 (InputLayer)         (None, 68)                0
_________________________________________________________________
embedding_4 (Embedding)      (None, 68, 100)           13743000
_________________________________________________________________
lstm_1 (LSTM)                (None, 68, 60)            38640
_________________________________________________________________
global_max_pooling1d_1 (Glob (None, 60)                0
_________________________________________________________________
dense_1 (Dense)              (None, 50)                3050
_________________________________________________________________
dropout_1 (Dropout)          (None, 50)                0
_________________________________________________________________
dense_2 (Dense)              (None, 1)                 51
=================================================================
Total params: 13,784,741
Trainable params: 13,784,741
Non-trainable params: 0
_________________________________________________________________
```

Figure 6: Our LSTM architecture

Since the LSTMs are capable of storing the long term dependencies we believe that this model is capable of keeping track of all the words responsible of contributing to the classification. We believe that this model will outperform all the other models mentioned before.

6 Hyper Parameter Tuning

Parameters which define the model architecture are called as Hyper Parameters and the process involved in finding an ideal value for the hyper parameter such that the model's performance is at its best is called as hyper parameter tuning.

6.1 Convolutional Neural Networks

Below are the Hyper parameters for CNNs:

- Number of filters in the convolutional layer
- Drop out rate for dropout layer
- Activation Function
- Batch Size

We will be tuning all the parameters mentioned above. To begin with we will be keeping the filter size as 64 for convolution layer, number of epochs as 20, activation function as relu. drop out rate as 0.5. We will be changing the batch size by keeping all the other parameters constant. We got the best accuracy for batch size of 256 as shown in Table 3

Table 3: Comparison of Accuracy for different batch sizes

Batch Size	**Accuracy**
128	94.01
256	95.55
512	95.22

Next we will be tuning number of filters in convolutional layers. Table 4 shows the accuracy for the different filters.

Table 4: Comparison of Accuracy for different filters

Filters	**Accuracy**
64	95.01
128	94.55
256	94.22

Next, we will tune the drop out rates of dropout layer in the network. Table 5 shows the accuracy for different drop out rates. We obtained the best accuracy for a drop out rate of 0.5

Table 5: Comparison of Accuracy for different drop out rates

Dropout	**Accuracy**
0.2	94.01
0.5	95.55
0.6	94.22

Next we tuned the activation function for the convolution layers. Table 9 shows the accuracy values for different activation functions.

We trained a model using those hyper parameters which gave us the best results during the tuning process, i.e.

- Activation function: relu

Table 6: Comparison of Accuracy for different activation functions

Activation	**Accuracy**
relu	95.35
sigmoid	92.76
tanh	93.02

- Drop out rate: 0.5
- Batch size: 256
- Filters: 64

6.2 LSTM

We performed the hyper parameter tuning for LSTM model as well. Below are the hyper parameters that we tuned for LSTM network:

- Number of units in LSTM layer
- Activation function

Table 7 shows the accuracy obtained for different activation functions in LSTM network. We obtained best accuracy for relu activation function.

Table 7: Comparison of Accuracy for different activation functions for LSTM network

Activation	**Accuracy**
relu	98.35
sigmoid	97.76
tanh	96.02

Table 8 shows the accuracy obtained for different number of units functions in a LSTM layer in LSTM network. We obtained best accuracy for 64 number of units in LSTM layer in LSTM network.

Table 8: Comparison of Accuracy for different number of units for the LSTM layer in LSTM network

Filters	**Accuracy**
64	98.12
128	97.32
256	97.89

We obtained best accuracy by considering activation function as relu and choosing 64 as number of units in the LSTM layer in the LSTM network.

7 Results and Conclusion

Below is the comparison of accuracy, precision, recall and F1-Measure values for the different model we trained.

Table 9: Comparison of performance of various classifiers

Classifier	**Accuracy**	**Precision**	**Recall**	**F-measure**
Naive Bayes	93.28	94.58	93.05	93.81
SVM	93.9	93.6	94.6	94.09
CNN	95.68	94.4	94.9	94.65
LSTM	98.23	98.8	98.2	98.5

In this project we experimented with different Machine Learning Models. LSTM model performed the best. Since our dataset is skewed, Deep learning Models perform better than the traditional ML models

References

[1] Text Classification Using Machine Learning Techniques M. IKONOMAKIS ,S. KOTSIANTIS, V. TAMPAKAS

[2] Convolutional Neural Networks for Toxic Comment Classification; Spiros V. Georgakopoulos ,Sotiris K. Tasoulis, Aristidis G. Vrahatis, Vassilis P. Plagianakos

[3] A Review of Machine Learning Algorithms for Text-Documents Classification; Aurangzeb Khan, Baharum Baharudin, Lam Hong Lee, Khairullah khan

[4] Toxic Comment Classification; Pallam Ravi, Hari Narayana Batta, Greeshma S, Shaik Yaseen

[5] https://www.kaggle.com/c/quora-insincere-questions-classification

[6] Application of Recurrent Neural Networks In Toxic Comment Classification; Li, Siyuan

[7] Toxic Comment Classification; Deepan Das

[8] Toxic Comment Classification - An Empirical Study; Manikandan R, Sneha Mani

[9] https://en.wikipedia.org/wiki/Naive_Bayes_classifier

[10] https://en.wikipedia.org/wiki/Support-vector_machine