Abstract—The emancipation of communication systems implies that bullying is no longer limited to schoolyards or street corners. This harassment has also infiltrated victims’ homes, via email, cell phones, and social media websites, in the form of cyber-bullying. For those who suffer such harassment, the effects can be devastating. The aim of our experimentation therefore, is to create a model using natural language processing to automate the flagging of cyber-aggressive comments with greater efficiency than the state-of-the-art models. In this task, this Paper proposes the application of Recurrent Neural Network Architectures to effectively analyze and flag such cyber aggressive comments on social media. The semantic document embeddings learnt by Doc2Vec [1] were fed into three recurrent neural network architectures for sequence classification. Our extensive experimentation indicates that an LSTM model using document embeddings outperforms the state-of-the-art models using standard feature extraction techniques for machine learning or other deep learning approaches.

Index Terms—cyber-aggressive comments; document embeddings; recurrent neural networks.

I. INTRODUCTION
Cyber-bullying refers to bullying of a person, typically by sending messages of an intimidating or threatening nature, by means of electronic communication. Various surveys indicate that this kind of online harassment, when too severe, not only lowers the self esteem of the victims, thereby inculcating bad habits among them, leaving them scarred for their whole life, but also results in suicides very often. According to NoBullying [2], an international authority on bullying, over 95% of teens use social media to communicate with their peers, of which 43% have been bullied online at least once and 1 among 4 teenagers have been bullied more than once. Twenty five countries were surveyed under a project, recently commissioned by Microsoft Corporation, to understand the global pervasiveness of online bullying [3].

According to this, the top three countries in which children reported online bullying were China, Singapore and India respectively. Due to the sudden explosion of text data with extensive use of social media nowadays, it is not feasible for people to manually flag cyber-aggressive comments, which has created the need for the existence of an automated system to efficiently perform such a classification task. Though few such automated models are existent, cyber-bullies often find means to defy such systems through usage of special characters, misspellings, sarcasm, etc. due to which the models specifically aimed at insult-detection or sentiment analysis for cyber-bullying detection become ineffective in such cases. Through our experimentation, we aim to determine whether the learning of long term dependencies by some rnn architectures coupled with word order dependent document embeddings, can prove effective to flag cyber-aggressive comments.

The remaining part of the Paper has been organized using the methodology discussed as follows. A survey of various related research experiments and literatures has been discussed in Section II. Section III comprises of the methodology proposed by this Paper. In Section IV, the results obtained from the experiment have been studied. Section V provides a conclusion to the Paper followed by a brief insight about future work which may be involved.

II. RELATED WORK
A recent research experiment was conducted in the field of flagging cyber-aggressive comments using various deep learning techniques to learn semantic embeddings for the purpose of identifying hate speech among tweets by Pinkesh Badjatiya et al, 2017 [4]. The performed extensive experimentation with these semantic embeddings and performed an analysis and comparison of tweet semantic embeddings from standard feature extraction techniques over Global Vectors for Word Representation or GloVe, and embeddings learnt by other models such as FastText, Convolutional Neural Networks and Long Short term Memory Unit Networks, and have shown the application of such deep learning frameworks to be considerably efficient in flagging tweets with hate speech.

With the objective of detecting cyber aggressive comments, Vikas S Chavan and Shylaja S S, 2015 [5] proposed the incorporation of two additional features like capturing of pronouns

† authors contributed equally
and skip grams in addition to standard techniques such as n-grams and TF-IDF. In accordance to their hypothesis, their model proved to be efficient in classifying comments with an accuracy of approximately 86%.

In another effort to curb cyber-bullying by Rekha Sugandhi et al., 2015 [6], a comparative study was made among algorithms like Support Vector Machine (SVM), Naive Bayes and J48 to detect cyber-bullying. In addition to standard techniques for data pre-processing, case folding and replacement of special symbols were studied.

Vinita Nahar et al., 2013, [7] presented a new way of feature selection, namely semantic features using the Latent Dirichlet Allocation and weighted features or the bullying-like features which included a static badword vocabulary. They employed a ranking algorithm which detected the people who bullied most and their victims. Their proposed approach constructs a cyber-bullying detection matrix and an associated graph representation.

Recent research involved in the usage of document embeddings or paragraph vectors or deep learning techniques like recurrent neural networks, in related classification problems have proven to be quite effective. In this direction, Parinya Sanguansat, 2016 [8] proposed the application of unsupervised deep learning feature extraction for text, called Paragraph2Vec or Doc2Vec with machine learning algorithms for sentiment analysis on social media. The experiments conducted indicated his proposal to be more effective than standard baseline methods.

In the field of sentiment analysis, another work in this direction by Xin Wang et al., 2015 [9], was directed at predicting polarities of tweets using word embeddings with Long Short-Term Memory (LSTM) networks. Through their experiments on a public and noisy labelled data, they showed that the model gave better results than several feature-engineering approaches.

It is observed that the application of deep learning through recurrent neural networks for sequence learning and document embeddings are state-of-the-art architectures in the fields of sentiment classification and other text classification problems. Considering the proximity of our task with sentiment analysis, both being text classification problems, our experiments are aimed at developing an efficacious model for identification of cyber-bullying comments through the combination of these architectures. In this attempt, our experimentation incorporates document embeddings generated using Doc2Vec with various RNN architectures, which include Simple RNN, LSTM and GRU networks and perform hyper-parameter tuning for certain parameters. We evaluate the models based on certain metrics in order to find the most accurate model among these for the task involved.

### III. Proposed Method

The methodology involved in the classification method as proposed by this paper includes creation of a feature matrix as the precursory step which numerically represent comments taking their semantics into consideration.

#### A. Datasets

The datasets used for training and testing purposes in our experiment have been collected from various repositories in Kaggle [10] and Github [11] websites. They comprise of comments which have been scraped from various social media sites and have been labelled manually. The total data was accumulated and split into training and testing data with 20645 and 8817 comments respectively. Each dataset comprises of two columns, in which each row consists of a comment or a group of comments from a conversation and a corresponding label, which is either a 0 or 1, indicating whether the corresponding comment is non-cyber-aggressive or cyber-aggressive respectively. These contain a variety of comments of both classes, with variations like special characters, sarcasm, insults, etc. used by cyber-bullies on various social media sites.

#### B. Pre-Processing and Feature-Extraction

The datasets used consist of raw text with labels, and such sequences of text cannot be processed directly by the learning algorithms. Therefore, the text data has to be converted into a feature matrix before they are used as input. Such a feature matrix should have a fixed optimum dimension, even though the comments in the data may be of variable length. Such a matrix is typically built using standard feature extraction techniques like bag-of-words, n-grams, word2vec or one hot encodings for words to name a few. The major drawback of traditional approaches like bag-of-words and n-grams often result in sparsely populated feature matrices with a very high dimension and semantics of the words are not retained. Bag-of-words approach, also suffer from the same problem, of not retaining the semantics of the words. Other approaches like word2vec enable the generation of dense low dimensional features which can be used to find analogous words, which is more suitable for classifying topics in a sentence by aggregating similar words. However, such a method is not suitable as it does not take into consideration, the order in which the words or phrases are present in case of long paragraphs of text, which is an important parameter in detecting cyber-aggressive comments. Since we are dealing even with multiple comments in a single instance, we generate document embeddings using Doc2Vec, as paragraph vectors to be learnt sequentially by various RNN architectures.

Doc2vec [12] (also known as paragraph2vec) is an algorithm for unsupervised learning of continuous representations for larger blocks of text, such as sentences, paragraphs or entire documents. The architecture of Doc2Vec comprises of the distributed memory and the distributed bag of words algorithms. However, for our experiments, the default distributed memory model has been employed since it outperformed DBOW as per the report in [13]. This model takes word order into account and produces dense fixed length feature vectors of any suitable dimension.

The procedure for obtaining the feature matrix with the document embeddings is as follows.

1. **Dataset Rearrangement:** The training and testing datasets are rearranged such that the comments are segregated.
NCB indicating tokens. The tags in our experimentation consist of a prefix indicating whether the comment is for training or testing purpose and whether it is cyber-aggressive or non cyber-aggressive. This prefix is followed by a unique index number to identify each comment. In this figure, the prefix is TRAIN_NCB indicating that it is from training data and non cyber-aggressive and 152 indicates the comment number in the file.

into two separate files based on their labels. Therefore, we have a pair of datasets each for training and testing, with the pair comprising of one dataset of cyber-aggressive comments and another has non cyber-aggressive comments. This rearrangement has been made to facilitate convenient tagging of sentences and feature retrieval from Doc2Vec as explained in the following subsections, in order to serve as the additional document context for training the Doc2Vec model.

2) Text Normalization: In order to ensure that our text data is consistent, we normalize it by removing of unnecessary characters like leading or trailing newline characters, tabs or spaces. Stop words have not been eliminated in our experimentation with the intention of preserving the long range semantic meaning of the sentences. We do not remove punctuation symbols and other special characters from the comments as cyber-bullies may use such means to construct cyber-aggressive terms as such as !id!ot or @$$. We also convert the entire text data to unicode representation to ensure a single canonical form for all comments.

3) Text Rearrangement: The sentences have to be formatted into a list of tokenized words and a corresponding tag or label for this list, to give a unique identification to the sentence, as shown in figure 1. The words are tokenized with respect to spaces only as other punctuation symbols may be used in certain cyber-aggressive words. Though there is no generalized convention for generating tags, we use this method for conveniently generating the document context for training the model for generating document embeddings. This formatting is performed on all comments from both training and testing datasets, resulting in an aggregation of such labelled lists of tokens.

4) Feature Extraction: With the aggregation of labelled sentences obtained in the previous step, the Doc2Vec model initially builds a vocabulary table. Once the vocabulary is generated, we train the model such that in each training epoch, the sequence of sentences fed to the model is randomized. This is done to ensure better training and to remove dependence on the order in which sentences are fed for training. The dimensions are set to 100 features, which has been found to be suitable among various values tested, being an optimum value such that the feature matrix is neither dimensionally too small to be learnt nor too large or sparse. Thus, using Doc2Vec, we obtain a dense feature matrix with an optimum dimension. The document embeddings are then retrieved into separate training and testing feature matrices. We also simultaneously generate separate label vectors for the training and testing matrices, as was provided originally in the dataset. The vectors are then standardized by scaling them for faster training of the model. For this purpose, the shape or dimension of the distribution is not taken into consideration. Transformation of the data is done by scaling data by dividing non-constant features by their standard deviation after ensuring that the data is centered first with the removal of the mean value of each feature.

C. Recurrent Neural Network Models

The first layer of the networks we implemented in our experiments comprises of the simple RNN, LSTM layer or GRU layer with 100 units since are feature vectors of document embeddings are 100 dimension vectors. As in typical neural network architectures, we use tanh as the activation function for these neurons. A dropout layer with a dropout of 20%, which has been found to be optimum for our problem through hyper-parameter tuning, has been employed. This ensures the prevention of over-fitting of the model. Finally, since this is a classification problem we use a Dense output layer with one neuron to obtain a single predicted probability for deciding the class label and this neuron is assigned with sigmoid activation function for this purpose. The sigmoid function has been suitably chosen for our model as the output can be conveniently tuned to be either 0 or 1. This is useful as we are dealing with a binary classification problem, to make 0 or 1 predictions for the two classes (non cyber-aggressive or cyber-aggressive). The training is performed for 100 epochs, which has been found experimentally to be optimum for maximum accuracy and avoiding over-fitting of the model. We have used adam as the optimizer and binary cross entropy for the loss function, found to be suitable for our model. The deep learning models through RNNs which have been experimented on, are as follows.

1) Simple Recurrent Neural Networks (simple RNN): The usage of sequential information is the chief principle that governs a recurrent neural network. Unlike the traditional neural networks, RNNs perform the same task for every single element of a sequence, with the output being dependent on the computations made previously, because of which these networks are termed as recurrent. The diagram in figure 2. shows the unrolling of a RNN into a complete network.

The processes and equations that govern the computations in a simple RNN have been discussed as follows: $x_t$ is assumed as the input while $s_t$ is assumed to be the hidden state at $t$, where $t$ represents a certain time step. $s_t$ is computed based on the preceding hidden state and the current step input.

$$s_t = f(Ux_t + Ws_{t-1})$$ (1)
The figure shows the unfolding of the computations involved during forward propagation of a simple recurrent neural network. The figure has been adapted from “Deep learning” by Yann LeCun, Yoshua Bengio & Geoffrey Hinton, Nature (International weekly journal of science) 521, pages 436 to 444, 28 May 2015.

The function $f$ in eq 1. is generally taken as a non linearity, such as tanh or a rectified linear unit (ReLU). $s_{-1}$, which is necessary to compute the first hidden state, is generally initialized with zeroes.

$$o_t = \tanh(Vs_t)$$  
(2)

When predicting whether a sentence is cyber-aggressive, we may only care about the final output and not the aggressive or non-aggressive nature of each word. We have also defined our loss, or error function to be the binary cross entropy loss (eq 3), since we are dealing with a binary classification problem.

$$E_t(y_t, \hat{y}_t) = -y_t \log \hat{y}_t$$  
(3)

$$E(y, \hat{y}) = \sum_t E_t(y_t, \hat{y}_t)$$  
(4)

Here, $y_t$ is the correct value at time step $t$, and $\hat{y}_t$ the predicted value. We typically treat the full sequence as one training example, so the final error is just the sum of the errors at each time step (word). We then calculate the gradients of the error with respect to the parameters involved ($U, V$ and $W$) and then learn good parameters using Stochastic Gradient Descent. Just like we total up the errors, we also total up the gradients at each time step for one training example:

$$\frac{\partial E}{\partial W} = \sum_t \frac{\partial E_t}{\partial W}$$  
(5)

However the states at distant steps have been observed to have no contribution to the learning process due to the fact that the gradient contributions become zero at such steps. Therefore a simple RNN due to this vanishing gradient problem becomes incapable of learning long-range dependencies [14].

2) Long Short Term Memory Networks (LSTM): Long short term memory units or LSTMs employ a gating mechanism in their design to tackle the vanishing gradients. The internal operations involved in the computation of a hidden state $s_t$ is observed through the figure below and the following equations.

$$i = \sigma(x_t U^i + s_{t-1} W^i)$$  
(6)

$$f = \sigma(x_t U^f + s_{t-1} W^f)$$  
(7)

$$o = \sigma(x_t U^o + s_{t-1} W^o)$$  
(8)

$$g = \tanh(x_t U^g + s_{t-1} W^g)$$  
(9)

$$c_t = c_{t-1} \circ f + g \circ i$$  
(10)

$$s_t = \tanh(c_t) \circ o$$  
(11)

where $\sigma(x)$ represents the sigmoid function, $\circ$ has been used to indicate element-wise multiplication and where the input, forget and output gates are representation by $i, f$ an $o$ gates, respectively. The dimensions possessed by all of these gates $d_s$ are equal and are the same as the size of the hidden state.

The present input and the previous hidden state are needed in the estimation of the candidate hidden state, $g$. The unit in the system acting as the internal memory is $c_t$, which is a fusion of the previous memory given as $c_{t-1}$ multiplied by the forget gate and the freshly obtained hidden state $g$, multiplied by the input gate. The output state $s_t$ is calculated, provided the memory $c_t$ is given, by computing its product with the output gate.

The chief operating mechanism which enables LSTM networks to explicitly model long term dependency is the gating mechanism. The network becomes capable of learning how the memory is to operate using the parameters learnt by the network for its gates.
3) Gated Recurring Unit Networks (GRU): The chief principle of employing a gating mechanism in a GRU network to overcome the vanishing gradient problem to enable capturing of long-term dependencies is similar to a LSTM, as seen by the following equations.

\[
\begin{align*}
    z &= \sigma(x_t U^z + s_{t-1} W^z) \\
    r &= \sigma(x_t U^r + s_{t-1} W^r) \\
    h &= \tanh(x_t U^h + (s_{t-1} \circ r) W^h) \\
    s_t &= (1 - z) \circ h + z \circ s_{t-1}
\end{align*}
\]

GRU’s gating system consists of two gates, namely, r or the reset gate, and z or the update gate. A gated recurring unit differs from a LSTM unit with respect to certain aspects related to their gating mechanism. A distinct internal memory \((c_t)\) apart from the exposed hidden state and the output gate as present in a LSTM unit is not possessed by a GRU.

The input and forget gates paired by an update gate \(z\), together with the reset gate \(r\) is directly applied to the preceding hidden state. Thus, the function of the reset gate in a LSTM is really split up into both \(r\) and \(z\). A second non-linearity when computing the output, is not applied in case of gated recurring unit networks.

D. LSTM vs Simple RNN and GRU Networks

The idea involved in the development of recurrent neural networks was to retain the data computed in a previous step and used in later processes. However, a simple RNN, due to the vanishing gradient problem mentioned earlier is often unable to handle long term dependencies, and fit only in problems where the gap between relevant data and the place of its need is small. On the other hand, LSTM and GRU networks are capable of effectively capturing long term dependencies through its gating mechanism through the concept of layers and gates, thereby tackling the vanishing gradients in a simple RNN.

Research indicates that one of the primary features of LSTM which is lacking in a GRU network is the controlled exposure of memory content. While a GRU network exposes its entire content without imposing any control, LSTM networks use the output gate to control the quantity of memory content visible for use by other units in the network.

Apart from this, it is not necessary for a GRU unit to employ a memory unit in order to regulate the flow of information, as in a LSTM unit. Possessing lesser number of parameters, a GRU unit may train faster or require lesser data for the purpose of generalization. However, when handling large numbers of data instances, an LSTM network with greater expressiveness may lead to better results as seen in various experiments.

IV. Results

Various models created by incorporating document embeddings as sequential input to the recurrent neural network architectures were tested with comments from the test dataset consisting of 8817 comments. The end result of such an experiment is to create an effective model, for the generation of predicted labels for the test data, which are probable values to indicate whether the comments being cyber-aggressive towards peers on social media. We apply various evaluation metrics to find the model with maximum precision and accuracy among those which have been tested.

A. Evaluation Metrics on various Models

The various evaluation parameters that we have applied to compare the models when applied to the test dataset are as follows:

1) Learning curves and training performance: The Learning curves referred to in our results portray the plots of training and validation testing accuracy or the loss incurred by during training of the model with number of epochs. The slope of the curve reflects the speed with which a model approaches convergence. The convergence of the models is indicated by the plateau region of the curves.

The training accuracy and loss incurred in 100 epochs have also been specified in the results given. These parameters are a measure of how well a model is trained.

2) Accuracy score: This is a parameter used in multi-label classification, in order to compute subset accuracy. It is a measure of the number of positive match of the true labels and the predicted ones.

3) K-Fold Cross Validation: K-fold cross-validation refers to an evaluation metric in which the original sample is divided randomly into k samples of equal size, of which only one sub-sample is maintained for testing the model evaluated, known as the validation set, while the remaining sub-samples are used as training set. The cross-validation process is repeated for k iterations, so as to ensure that each of the k sub-samples are used once as the validation data. The mean of the k results obtained from the process may be used to produce a single evaluation parameter. The value of K is typically taken as 10.
4) Precision: The precision is an evaluation parameter which is a measure of the ability of the classifier not to falsely label a negative sample as positive. It can be defined as follows:

\[
\text{Precision} = \frac{TP}{TP + FP}
\]  
\[\text{(16)}\]

where TP is the total number of true positives and FP the total number of false positives.

5) Recall: The recall value for a model, represents the ability of the classifier to find all the positive samples. Recall may be written as:

\[
\text{Recall} = \frac{TP}{TP + FN}
\]  
\[\text{(17)}\]

where TP refers to the total number of true positives and FN corresponds to the total number of false negatives for both classes.

6) F-score: The F-score is a metric which may be thought of as a weighted harmonic mean of the precision and recall values for a model, with the best possible value for F-score being 1 and the worst being 0.

7) Area under ROC Curve: From the confusion matrices, two other metrics, the False Positive Rate (FPR) and the True Positive Rate (TPR) values are computed, which are subsequently combined into a single metric, by first computing the two former metrics with many different thresholds and subsequently plotting them on a single graph, with the FPR values taken as the abscissa and the TPR values taken as the ordinate. The resulting curve is referred to as a Receiver Operating Characteristic (ROC) curve, and the metric we consider for evaluating of the models is the area under the ROC curve or AUROC. A model is said to be ideally perfect if it obtains an AUROC score of 1.

8) Confusion matrices: A confusion matrix, also referred to as an error matrix is a significant metric for evaluating models involving machine learning, deep learning and more specifically, problems involving statistical classification. It is a specific table layout that helps in visualization of the performance of an algorithm, generally involving supervised learning.

The summarization of the results obtained have been portrayed in Figures 4-8 and in Tables I and II.
RESULTS

<table>
<thead>
<tr>
<th>Model</th>
<th>Training Accuracy after 100 epochs</th>
<th>Training loss after 100 epochs</th>
<th>Testing Accuracy Score</th>
<th>10-Fold Cross Validation Score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple RNN</td>
<td>88.48 %</td>
<td>0.2696</td>
<td>89.21 %</td>
<td>79.87 %</td>
</tr>
<tr>
<td>LSTM</td>
<td>97.29 %</td>
<td>0.0739</td>
<td>93.84 %</td>
<td>85.92 %</td>
</tr>
<tr>
<td>GRU</td>
<td>93.94 %</td>
<td>0.1491</td>
<td>92.28 %</td>
<td>84.20 %</td>
</tr>
</tbody>
</table>

TABLE I: EVALUATION METRICS

<table>
<thead>
<tr>
<th>Model</th>
<th>AUROC Score</th>
<th>Precision</th>
<th>Recall</th>
<th>F-score</th>
</tr>
</thead>
<tbody>
<tr>
<td>Simple RNN</td>
<td>0.79</td>
<td>0.88</td>
<td>0.88</td>
<td>0.88</td>
</tr>
<tr>
<td>LSTM</td>
<td>0.92</td>
<td>0.94</td>
<td>0.94</td>
<td>0.94</td>
</tr>
<tr>
<td>GRU</td>
<td>0.90</td>
<td>0.92</td>
<td>0.92</td>
<td>0.92</td>
</tr>
</tbody>
</table>

TABLE II: EVALUATION METRICS

B. Inference

The end result of our experiments with the tested models is obtaining labels for the unseen test comments indicating whether they are cyber-aggressive/non cyber-aggressive.

Depending on the performance of the models based on the evaluation metrics used, we have summarized and compared the models to identify the most suitable model for efficiently classifying and labelling unseen comments.

Table I consists of a summary of the evaluation results for the tested models, based on training accuracy, loss, accuracy score and k-fold cross validation whereas Table II portrays the AUROC Score, precision, recall and f-score. The learning curves have been displayed in Figures 5 and 6 whereas the ROC curves have been presented in Fig 7 and the confusion matrices in Fig 8.

Our evaluation of the tested models indicate that the highest accuracy achieved is that of the LSTM model, which is approximately 93.84%, with an f-score of 0.94. LSTM has also surpassed other tested models in effectively labelling the test dataset as evaluated by other metrics summarized above, thereby proving it to be an effective model to flag cyber-aggressive comments.

V. CONCLUSION AND FUTURE WORK

In this paper, we have proposed the usage of document embeddings generated by Doc2Vec [12] for sequential learning by recurrent neural network architectures as a model for effective classification of comments as cyber aggressive/non-cyber aggressive on social media. As consequence of such an experiment, we found that the Doc2Vec embeddings coupled with an LSTM network gives a considerable accuracy of approximately 93.84% and an f-score of 0.94 in labelling test comments as cyber aggressive/non-cyber aggressive with a precision and recall value, which has been found to be greater than the simple RNN and the GRU network models. Such a model can also be used to compare and evaluate human performance and accuracy in classifying comments as cyber aggressive or not.
Further future work may be directed towards finding an optimized model which can more accurately flag cyber-aggressive comments with less training time complexity, which may be incorporated into an application programming user interface to monitor real time cyber-bullying by peers on social media.

REFERENCES


