International Journal of Scientific Research in Engineering and Management (IJSREM)
Volume: 06 Issue: 05 | May - 2022 Impact Factor: 7.185 ISSN: 2582-3930

# TOXIC COMMENT CLASSIFICATION USING DEEP LEARNING

Aakash Garg
Maharaja Agrasen Institute of Technology (CSE)

GGSIPU
Delhi, India

Himanshu Goyal
Maharaja Agrasen Institute of Technology (CSE)

GGSIPU
Delhi, India

Ishaan Garg
Maharaja Agrasen Institute of Technology (CSE)

GGSIPU
Delhi, India

## ABSTRACT

With the waxing leverage in the comment section of online platforms, many users show some intolerable behavior by posting toxic comments. The comments should be classified in such a way that no one is bullied, some users are also encouraged to write the comments. And this can be a great help to the community. we present an expert model for toxic comments classification, using deep learning tools. We identify major components in an online toxic comment and determine the toxicity level of the comment.

## II. KEYWORDS

Comment classification, LSTM (Long-Short Term Memory), Machine Learning, LSTM-CNN, toxicity, Talos, Fast-Text, RNN

## I. INTRODUCTION

The global internet usage arising from the online interactive communications over the internet among users produces on a daily basis a vast amount of information. While this situation contributes substantially to the quality of human life, sadly, it involves colossal insecurity, since online content with huge toxicity can cause intimate attacks, online persecution, and bullying attitude. Comments containing explicit language can be classified into countless classes such as Toxic, Severe Toxic, Obscene, Threat, Insult, and Identity Hate. This has brought about both the industrial and research community in the latest few years while there are several endeavors to determine an adept model for online toxic comment forecasting. The hazard of misconduct and harassment is the means for many people to stop expressing themselves and give up on seeking contrasting ideas.

## III. LITERATURE SURVEY

P. Vidyullatha, Satya Narayan Padhy, Javvaji Geetha Priya, Kakarlapudi Srija, Sri Satyanjani Koppisetti in their research paper, Identification and Classification of Toxic Comment Using Machine Learning Methods [1] talks about the growing menace of hatredness and negativism has been witnessed in the online platforms especially social media like Instagram, twitter, facebook as such, many government agencies around the world have seen the insignificant rise of cases related to cyber bullying that has led to increase of hate and violence. The main and the most important task of every online platform provider is to keep the conversations and deals constructive, comprehensive and inclusive. They conclude by taking the hamming loss as a measure of identifying the most suitable algorithm to classify toxic comments.

Nayan Banik in his research paper, Toxicity Detection on Bengali Social Media Comments using Supervised Models [2] talks about the apathetic behavior and nature of the commentators which may hamper and disturb the online environment or atmosphere. They are very clear that Researchers have tried statistical machine learning models to find the toxicity of a sentence. But these models require frequency based featured engineering or probabilistic

phenomena and hence do not scale up well with unstructured comments of social media. He found that a part of Sentiment Analysis is constituted in classification of such toxic comments by Automated machine learning based models and they are majorly used for the English language, showing more better results than any other statistical models. Major problems in comments of the social media platforms include grammar errors, useless punctuation, spelling errors, random duplication and emojis.

Riccardo Mulas, in his research paper, A Supervised Multiclass Multilabel Word Embeddings Approach for Toxic Comment Classification [3] found that four out of ten Americans have experienced the synonymous online harassment, and most considered this as a genuine problem. It should be noted that the last report stipulated that most of the people who have faced this kind of problem asked for new technologies, although they are in dilemma about how to balance free speech and online security or safety. He found during his research that comments that have been shared within digital platforms can conceal hazards, such as harassment, insults, fake news, and more in general, comments that may hurt the feelings of others. Threatened by the dangers related to personal attacks that lead most of the persons leave the discussion in which they were participating or playing any role. Such type of problem is related to the so-called toxic comments, i.e., verbal bullying, personal attacks, and, more generally, an violent way in which many people participate in a discussion, which brings some participants to throw over it.

Revati Sharma and Meetkumar Patel in their research paper, Toxic Comment Classification Using Neural Networks and Machine Learning [4] found that these type of comments involves collosal dangers as online text conversation with immense toxicity quality cause personal assaults, online insult, and harassing attitude. They came to a very symbolic conclusion that using the word embedding methods and also analyzing the elementary level neural network algorithms results with complex Convolutional Neural Networks and Recurrent Neural Network compose: LSTM (LongShort Term Memory) results, the collected scrutiny show that the LSTM perform better than the CNNs in terms of both the efficiency and time performance given the identical number of epoch and hence are desirable to use as compared to CNN with wordlevel embeddings.

Marwan Torki and Mai Ibrahim in their research paper, Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning [5] found that the current CNN-based model only predicts whether or not a comment is harmful. Their solution also averts the problem of data class imbalance by using a fair subgroup of the data for creating the model. They came to know that cyber-blustering and digital harassment have become two of the most deliberate issues in many civil digital communities.

## IV. PROPOSED ARCHITECTURE

1. Incorporating data (Comments)
2. Data pre-processing and feature engineering (Lemmatization, Padding, Stopword Removal)
3. Model creation and assessment
4. Setting up basic LSTM and CNN Model
5. Plotting Actual, Benchmark Predicted Values, and LSTM-CNN Predicted Values
6. Evaluating model accuracy and model loss.
7. Calculating model accuracy using test set.

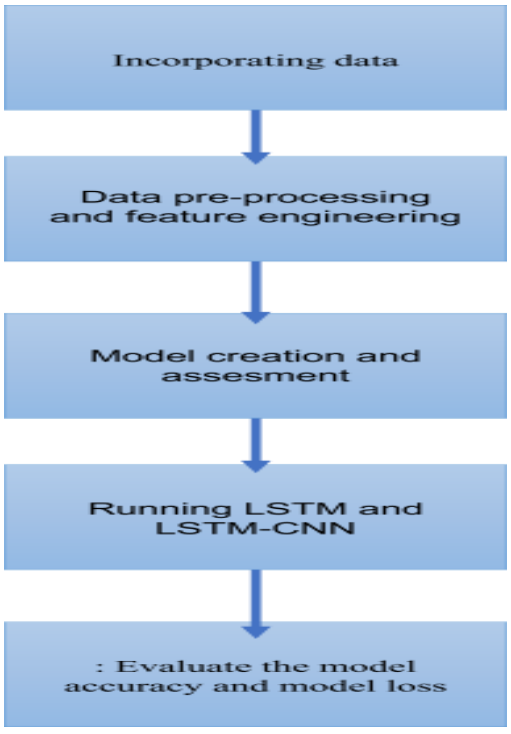



Fig 1: Proposed Architecture

## V. APPROACH

To protect the community and ensure toxic comments are not displayed on the platform, we are building a baseline model that's capable of detecting and classifying different types of toxicity like, identity-based hate, obscenity, insults, and threats. The project is created to help improve community on the internet. It uses two types of algorithms – Long Short-Term Memory networks (usually just called LSTMs) and convolutional neural network CNNs. They are used to recognize toxic comments based on the context of the words.

For this project measure of performance is the accuracy calculated by predicting the toxicity level of the comments in the dataset. A 10-fold cross validation was carried out to calculate the accuracy of the model. The dataset is split into training set and validation set. The models are trained on training set and accuracy is measured over testing dataset.

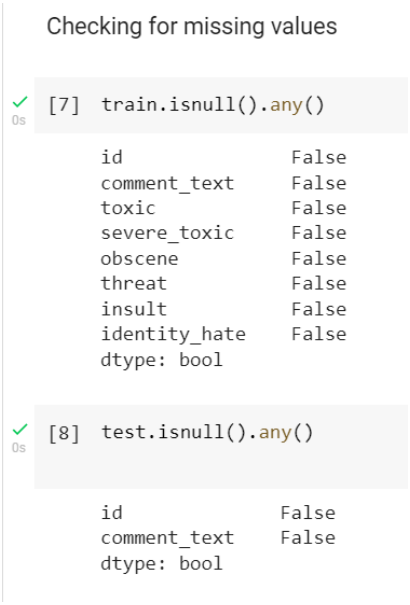



Fig 3: Checking for null values.

### A. HANDLING MISSING VALUES

After importing the training and test data into a pandas dataframe, the data is checked for missing data. Using the 'isnull' function on both the data, it is discovered that there are no missing records. Once this has been determined, another check is made to determine whether there is any duplicated records in the set. With pandas's unique methods of discovering duplicate indexes within a dataframe, it can use unique methods to eliminate duplicates and be left with a single record per unique index value.

| | id | comment_text | toxic | severe_toxic | obscene | threat | insult | identity_hate |
|---|---|---|---|---|---|---|---|---|
| 0 | 0000997932d777bf | Explanation\nWhy the edits made under my usern... | 0 | 0 | 0 | 0 | 0 | 0 |
| 1 | 000103f0d9cfb60f | D'awww! He matches this background colour I'm s... | 0 | 0 | 0 | 0 | 0 | 0 |
| 2 | 000113f07ec002fd | Hey man, I'm really not trying to edit war. It... | 0 | 0 | 0 | 0 | 0 | 0 |
| 3 | 0001b41b1c6bb37e | "\nMore\nI can't make any real suggestions on ... | 0 | 0 | 0 | 0 | 0 | 0 |
| 4 | 0001d958c54c6e35 | You, sir, are my hero. Any chance you remember... | 0 | 0 | 0 | 0 | 0 | 0 |

Fig 2: Training Data

### B. NORMALIZING TEXT

In order to focus on the content of the comment and not the language, a morphological analysis is performed to normalize the text. The Regex tool will be used to perform the analysis and to tackle the inconsistency in data.

To effectively classify toxic comments and isolate them from the rest of the dataset, we create a dictionary that contains common representations of cuss words that can be ordinarily found over online forums or social media platforms. Then we create a program that performs a series of programs and gives clean data as output.

```
def clean_text(text,remove_repeat_text=True, remove_patterns_text=True,
               is_lower=True):

  if is_lower:
    text=text.lower()

  if remove_patterns_text:
    for target, patterns in RE_PATTERNS.items():
      for pat in patterns:
        text=str(text).replace(pat, target)

  if remove_repeat_text:
    text = re.sub(r'(.)\1{2,}', r'\1', text)

  text = str(text).replace("\n", " ")
  text = re.sub(r'[^\w\s]',' ',text)
  text = re.sub('[0-9]',"",text)
  text = re.sub(" +", " ", text)
  text = re.sub("([^\x00-\x7F])+"," ",text)
  return text
```

Fig 4: Function for cleaning the comment text.

## C. REMOVING DUPLICITY

Before moving to the next step we need to train out machine to eliminate duplicity in posts. To do this, 'set's check_duplicates function' is used to identify and remove duplicates. Now our data is clean and consistent, it's time to perform Lemmatization. Lemmatization is the process of grouping together the different inflected forms of a word so they can be analyzed as a single item. For example, it is very inefficient of a machine learning algorithm to treat playing, plays, and play as separate words because, in fact they are not. Lemmatization match 'playing' and 'plays' to its root word, i.e. play. To carry out Lemmatization, 'WordNetLemmatizer' from the "nltk" library is imported, a function 'lemma' performs Lemmatization, and it is then applied to the clean data.

```
[17] def lemma(text, lemmatization=True):
       output=""
       if lemmatization:
         text=text.split(" ")
         for word in text:
            word1 = wordnet_lemmatizer.lemmatize(word, pos = "n")
            word2 = wordnet_lemmatizer.lemmatize(word1, pos = "v")
            word3 = wordnet_lemmatizer.lemmatize(word2, pos = "a")
            word4 = wordnet_lemmatizer.lemmatize(word3, pos = "r")
            output=output + " " + word4
       else:
         output=text

       return str(output.strip())
```

Fig 5: Function to lemmatize the comments.

## D. REMOVAL OF STOPWORDS

Using stopwords to remove uninformative words from a sentence is one of the most important components in text pre-processing for use cases that involve text classification. By removing stopwords, a more informative document can be created for use-cases that involve text classification.

Spacy library is used to remove stopwords from the text data. It has a list of common stopwords called STOP_WORDS which can be used to remove stopwords from any textual data or statements.

This classifier was specifically designed to learn language patterns in customer reviews for detecting toxic comments. The model was trained on approximately 1,57,000 unique messages from customer reviews scraped from Kaggle. While training the model, additional stopwords are searched that might be unique to the dataset.

While skimming through dataset, there were several comments which has single-letter or two-letter words existed without any meaning or context, (e.g. I study in University of liverpool w!! or You got this right bb.) To ensure such single-letter or two-letter words do not hamper the performance of our model, these are pushed to the list of stopwords. While adding to the list we need to make sure that words like as, or, am, me letters like I and A are not added as they make sense on their own.

Once this was completed, words in dataset are searched for that could be possible stopwords, (the criteria being that they appear in the dataset very frequently, and second, they do not have a significant contribution towards the classification task). With the help of stopword list and advanced algorithms, it will remove the unwanted stopwords from the dataset. After removing them, it will produce a clean data set that is free from all inconsistencies.

## E. SPLITTING, INDEXING AND REPRESENTATION

To train a deep-learning model to detect toxic comments, we must first convert the data into its equivalent machine-readable form. This requires that we separate the text into multiple components: words, punctuation, numerals and symbols. Once each word has been converted into a word vector, additional processing is required so strings are represented with numbers (i.e., 'HAT' will be represented as 0 and 'BAT' as 1).

1. Tokenization: Tokenization is used break or split down sentences into unique words. e.g. "I am here to groove" will become ["I", "am", "here", "to", "groove"].

2. Indexing: Each word is mapped to a specific index as a dictionary-like fashion e.g. {1: "I",2: "am",3: "here",4: "to",5: "groove"}.

3. Index Representation: It is use to represent a statement with the help indexing done in the above step. A statement is fed as a chain of index. For example. [1,7,2,1,4,19].

Using the 'Tokenizer' class from the 'Keras library, the above-mentioned steps can be easily performed. This class allows vectorizing a text corpus, by turning each text into either a sequence of integers (each integer being the index of a token in a dictionary) or into a vector where the coefficient for each token could be binary, based on word count, based on tf-idf, etc.

## F. PADDING

Padding is a technique that is used to ensure that training examples and test examples of a fixed length have comparable lengths. Padding involves the addition of extra values which are not in the domain. This is done to short or wide sequences to make them longer or narrower respectively, so that they can fall into compatibility with a fixed-length sequence but they are not representative of what they claim to represent. Hence, Padding adds additional information as long as it comes in handy.

The "pad_sequences" function from the "Keras" library is used and the sequence vector length is fixed at 200 words. As soon as the padding of sequence vectors is completed, creation of deep-learning models can be started.

## G. SPLIT TRAINING DATA INTO TRAIN-SET AND VALIDATION-SET

In this part of the project, we try to fit a deep learning model on the training data. The validation set accounts for 20%. Before trying to fit a deep learning model on the training data, the data is split into train-set and validation-set.

## H. IMPORT FASTTEXT'S PRE-TRAINED WORD EMBEDDINGS

The pre-trained word embeddings from fastText are used to harness the power of transfer learning. An embedding matrix is created by assigning the vocabulary with the pre-trained word embeddings and then learning a new representation while predicting words that have no frequency statistics.

```
[ ] embeddings_index_fasttext = {}
    f = open('drive/MyDrive/wiki-news-300d-1M.vec', encoding='utf8')
    for line in f:
        values = line.split()
        word = values[0]
        embeddings_index_fasttext[word] = np.asarray(values[1:], dtype='float32')
    f.close()
```

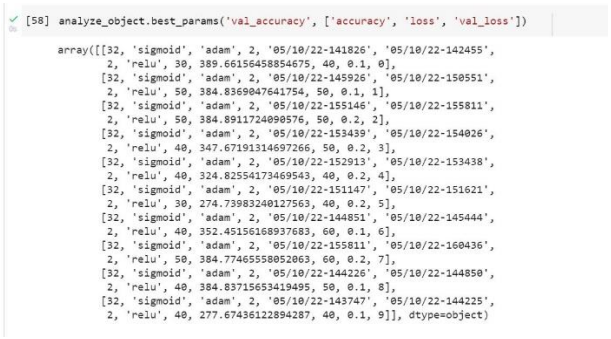
Fig 6: Importing Fast Text

.

## I. LSTM MODEL CREATION

We will be working on a Natural Language Processing use-case. Long Short Term Memory (LSTM) networks are similar to Recurrent Neural Networks (RNN) but with one major difference that hidden layer updates are replaced by memory cells. This makes them better at finding and exposing long-range dependencies in data which is imperative for sentence structures.

We import the "Talos" library to begin. This is used to perform hyperparameter tuning as well as model evaluation. Using the "Scan" function, a GridSearchCV is performed and the best parameters are found that would give the highest accuracy.



```
[58] analyze_object.best_params('val_accuracy', ['accuracy', 'loss', 'val_loss'])

array([[32, 'sigmoid', 'adam', 2, '05/10/22-141826', '05/10/22-142455',
        2, 'relu', 30, 389.66156458854675, 40, 0.1, 0],
       [32, 'sigmoid', 'adam', 2, '05/10/22-145926', '05/10/22-150551',
        2, 'relu', 50, 384.8369047641754, 50, 0.1, 1],
       [32, 'sigmoid', 'adam', 2, '05/10/22-155146', '05/10/22-155811',
        2, 'relu', 50, 384.8911724090576, 50, 0.2, 2],
       [32, 'sigmoid', 'adam', 2, '05/10/22-153439', '05/10/22-154026',
        2, 'relu', 40, 347.67191314697266, 50, 0.2, 3],
       [32, 'sigmoid', 'adam', 2, '05/10/22-152913', '05/10/22-153438',
        2, 'relu', 40, 324.82554173469543, 40, 0.2, 4],
       [32, 'sigmoid', 'adam', 2, '05/10/22-151147', '05/10/22-151621',
        2, 'relu', 30, 274.73983240127563, 40, 0.2, 5],
       [32, 'sigmoid', 'adam', 2, '05/10/22-144851', '05/10/22-145444',
        2, 'relu', 40, 352.45156168937683, 60, 0.1, 6],
       [32, 'sigmoid', 'adam', 2, '05/10/22-155811', '05/10/22-160436',
        2, 'relu', 50, 384.77465558052063, 60, 0.2, 7],
       [32, 'sigmoid', 'adam', 2, '05/10/22-144226', '05/10/22-144850',
        2, 'relu', 40, 384.83715653419495, 50, 0.1, 8],
       [32, 'sigmoid', 'adam', 2, '05/10/22-143747', '05/10/22-144225',
        2, 'relu', 40, 277.6743612894287, 40, 0.1, 9]], dtype=object)
```

Fig 7: Talos Grid Scan Analysis (LSTM)

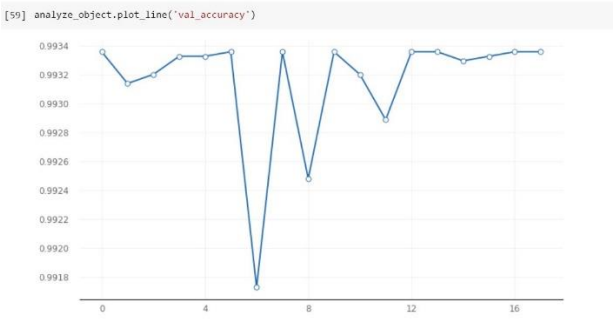



Fig 8: Validation accuracy found by talos scan for LSTM model

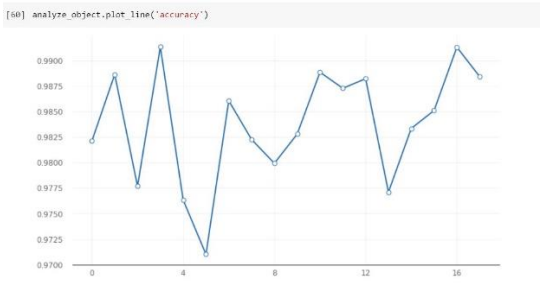
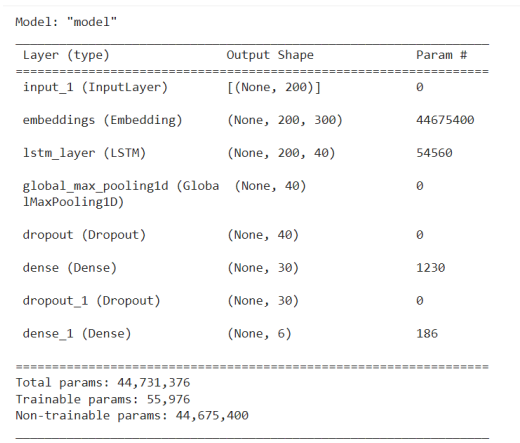



Fig 9: Overall accuracy for different hyperparameters for LSTM Model

The next step is to define the number of layers for the LSTM model. Then, using the best hyper-parameters defined in the previous step, we compile the model and train it using the train-set and validation-set.



```
Model: "model"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 200)]             0

 embeddings (Embedding)      (None, 200, 300)          44675400

 lstm_layer (LSTM)           (None, 200, 40)           54560

 global_max_pooling1d (Globa  (None, 40)               0
 lMaxPooling1D)

 dropout (Dropout)           (None, 40)                0

 dense (Dense)               (None, 30)                1230

 dropout_1 (Dropout)         (None, 30)                0

 dense_1 (Dense)             (None, 6)                 186

=================================================================
Total params: 44,731,376
Trainable params: 55,976
Non-trainable params: 44,675,400
_________________________________________________________________
```

Fig 10: LSTM Model Parameters

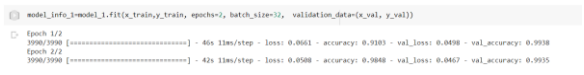



Fig 11: Training Result of LSTM Model

## J. LSTM-CNN MODEL CREATION

There are two main approaches to tackle the problem of language translation: neural networks and deep learning. These methods learn from examples and make it possible to translate entire sentences instead of just words or phrases. LSTM can effectively preserve the characteristics of historical information in long text sequences whereas CNN can extract the local features of the text. Combining the two traditional neural network architectures will help to harness their combined capabilities. The goal is to compare the performance of both the deep-learning architectures and ascertain the best deep-learning model for the project.

The process of training the hybrid model starts with evaluating different hyperparameters for the hybrid model. Following this, the models that give the highest accuracy are selected for training. At this step, a validation-set is also created to assess how good the model is after training. This helps us determine any improvements required in our model by comparing it with other models, thus enabling us to improve our existing models

```
[59] analyze_object.best_params('val_accuracy', ['accuracy', 'loss', 'val_loss'])

array([[50, 'relu', 2, 'sigmoid', '05/11/22-064647', 0.1, 40, 2, 32, 3,
        'adam', 64, 40, 96.26771974563599, '05/11/22-064823', 3, 0],
       [50, 'relu', 2, 'sigmoid', '05/11/22-065856', 0.2, 40, 2, 32, 3,
        'adam', 64, 30, 144.34663724899292, '05/11/22-070121', 3, 1],
       [60, 'relu', 2, 'sigmoid', '05/11/22-070121', 0.2, 40, 2, 32, 3,
        'adam', 64, 30, 144.3636441230774, '05/11/22-070346', 3, 2],
       [50, 'relu', 2, 'sigmoid', '05/11/22-070347', 0.2, 40, 2, 32, 3,
        'adam', 64, 40, 144.36036109924316, '05/11/22-070611', 3, 3],
       [60, 'relu', 2, 'sigmoid', '05/11/22-070612', 0.2, 40, 2, 32, 3,
        'adam', 64, 40, 144.33738350868225, '05/11/22-070836', 3, 4],
       [60, 'relu', 2, 'sigmoid', '05/11/22-071421', 0.2, 50, 2, 32, 3,
        'adam', 64, 40, 101.67614769935608, '05/11/22-071603', 3, 5],
       [60, 'relu', 2, 'sigmoid', '05/11/22-071015', 0.2, 50, 2, 32, 3,
        'adam', 64, 30, 100.21928095817566, '05/11/22-071155', 3, 6],
       [50, 'relu', 2, 'sigmoid', '05/11/22-065004', 0.1, 50, 2, 32, 3,
        'adam', 64, 30, 144.35121059417725, '05/11/22-065228', 3, 7],
       [50, 'relu', 2, 'sigmoid', '05/11/22-070837', 0.2, 50, 2, 32, 3,
        'adam', 64, 30, 97.6816759109497, '05/11/22-071014', 3, 8],
       [50, 'relu', 2, 'sigmoid', '05/11/22-071156', 0.2, 50, 2, 32, 3,
        'adam', 64, 40, 144.36582851409912, '05/11/22-071421', 3, 9]],
      dtype=object)
```

Fig 12: Talos Grid Scan Analysis (LSTM-CNN)

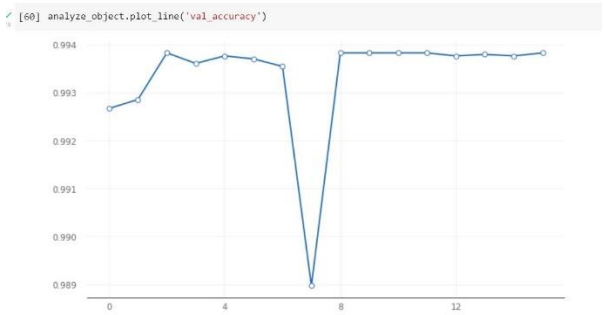



Fig 13: Validation accuracy of various hyperparameters by talos scan for LSTM-CNN

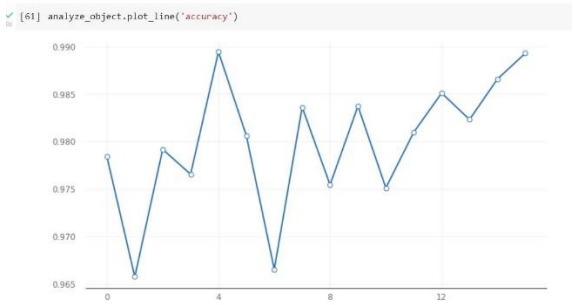



Fig 14: Accuracy of various hyperparameters by talos scan for LSTM-CNN

Next, we define the number of layers needed in the LSTM-CNN model, compile it and train the model using the train-set and validation-set.

```
Model: "model_1"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_2 (InputLayer)        [(None, 200)]             0

 embeddings (Embedding)      (None, 200, 300)          44675400

 lstm_layer (LSTM)           (None, 200, 50)           70200

 conv1d (Conv1D)             (None, 200, 64)           9664

 max_pooling1d (MaxPooling1D  (None, 66, 64)           0
 )

 global_max_pooling1d_1 (Glo  (None, 64)               0
 balMaxPooling1D)

 batch_normalization (BatchN  (None, 64)               256
 ormalization)

 dense_2 (Dense)             (None, 40)                2600

 dropout_2 (Dropout)         (None, 40)                0

 dense_3 (Dense)             (None, 30)                1230

 dropout_3 (Dropout)         (None, 30)                0

 dense_4 (Dense)             (None, 6)                 186

=================================================================
Total params: 44,759,536
Trainable params: 84,008
Non-trainable params: 44,675,528
_________________________________________________________________
```

Fig 15: LSTM-CNN Model Parameters

```
[ ] model_info_2=model_2.fit(x_train,y_train, epochs=2, batch_size=32,  validation_data=(x_val, y_val))

    Epoch 1/2
    3990/3990 [==============================] - 49s 12ms/step - loss: 0.0741 - accuracy: 0.8897 - val_loss: 0.0542 - val_accuracy: 0.9938
    Epoch 2/2
    3990/3990 [==============================] - 47s 12ms/step - loss: 0.0554 - accuracy: 0.9697 - val_loss: 0.0515 - val_accuracy: 0.9938
```

Fig 16: Training Result of LSTM-CNN

## K. EVALUATE THE MODEL ACCURACY AND MODEL LOSS DURING THE TRAINING PHASE.

A good indication for the quality of the deep-learning model is the accuracy and the loss value during every epoch. Ideally, the loss value for any deep-learning model should decrease as the number of epochs increases, on the other hand, the accuracy should increase as the number of epochs increases. This gives a fairly decent idea about the quality of your model and whether it has been appropriately trained.

**LSTM:**

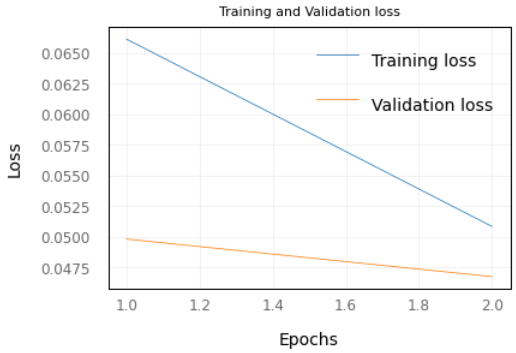



Fig 17: Training and Validation Loss of LSTM Model

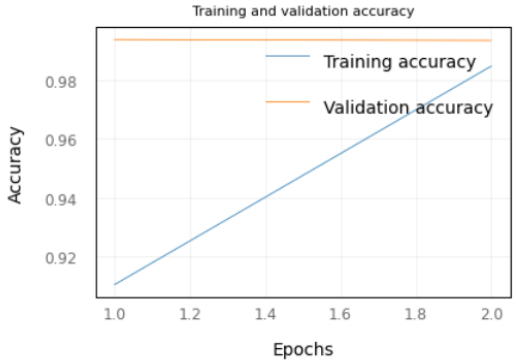



Fig 18: Training and Validation Accuracy of LSTM Model

**LSTM-CNN:**


Fig 19: Training and Validation Loss of LSTM-CNN Model

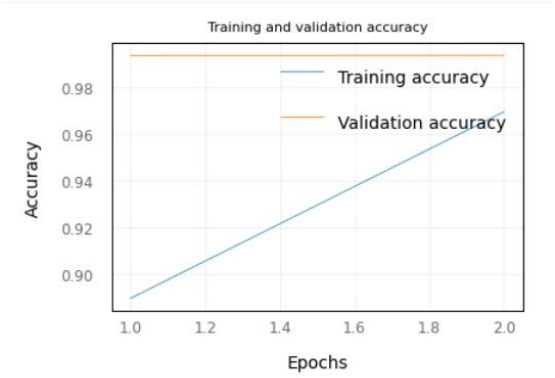



Fig 20: Training and Validation Accuracy of LSTM-CNN Model

## L. CALCULATING MODEL ACCURACY USING TEST-SET

To further evaluate the model, cross-validation is used to estimate the standard error of the model. After obtaining a good estimation on the standard error, we then assess the performance of our deep learning models using test set accuracy and loss values. We can obtain this information by executing the "predict" function for all inputs present in our test set data.

The maximum accuracy for the LSTM model is found to be 97.70% and that of the LSTM-CNN model is 97.10%.

## VI. RESULT

Given that the LSTM deep-learning model has a higher accuracy (as compared to LSTM-CNN model), this project entails that the LSTM deep-learning model is a better choice as compare to LSTM-CNN model as evident from the accuracy. LSTM model gives an accuracy of 97.70% and LSTM-CNN gives an accuracy of 97.10%.

## VII. CONCLUSION

The traditional LSTM Model has the edge over the hybrid LSTM-CNN Model. In order to provide a better understanding of how deep learning models are trained, it was decided to evaluate the training phase of both models. After performance evaluation on both models, it can be claimed that the hybrid model loses marginally. The training phase generates a great deal of insights into how neural network models are trained and can be leveraged for further improving Deep Learning Models in future projects.

## VIII. FUTURE SCOPE

This project classifies toxic comments in various categories with high accuracy. There are many things that can be incorporated in the project some of which include:

- Before posting a comment, the model can check whether it is toxic or not. So, at the time of posting only we can filter out the comments which are toxic in nature.
- We can make it multi-lingual. Classification can be done on various languages.

## IX. REFERENCES

[1]. P. Vidyullatha, Satya Narayan Padhy, Javvaji Geetha Priya, Kakarlapudi Srija, Sri Satyanjani Koppisetti (2019, November) : "Identification and Classification of Toxic Comment Using Machine Learning Methods", Turkish Journal of Computer and Mathematics Education Vol.12 No.9 (2021), 70-74, Available:

https://www.turcomat.org/index.php/turkbilmat/article/download/2798/2425/5341

[2]. Nayan Banik (2019, November): "Toxicity Detection on Bengali Social Media Comments using Supervised Models", International Conference on Innovation in Engineering and Technology (ICIET) 23-24 December, 2019, Available: https://www.researchgate.net/publication/337317824_Toxicity_Detection_on_Bengali_Social_Media_Comments_using_Supervised_Models

[3]. Roberto Saia, Andrea Corriga, Riccardo Mulas, Diego Reforgiato Recupero (September, 2019) : "A Supervised Multi-class Multi-label Word Embeddings Approach for Toxic Comment Classification", : 11th International Joint Conference on Knowledge Discovery, Knowledge Engineering and Knowledge Management (KDIR-2019), Available:
(PDF) A Supervised Multi-class Multi-label Word Embeddings Approach for Toxic Comment Classification (researchgate.net)

[4]. Revati Sharma , Meetkumar Patel (September 2018) : "Toxic Comment Classification Using Neural Networks and Machine Learning", International Advanced Research Journal in Science, Engineering and Technology Vol. 5, Issue 9, September 2018, Available : IARJSET.2018.597.pdf

[5]. Marwan Torki. Mai Ibrahim A (December, 2018) : "Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning", Conference ICMLA, Available: (PDF) Imbalanced Toxic Comments Classification Using Data Augmentation and Deep Learning (researchgate.net)

[6]. T. Cooper, C. Stavros, and A. R. Dobele, "Domains of influence:exploring negative sentiment in social media," Journal of Product &Brand Management, 2019.

[7]. "What are the 10 most spoken languages in the world? — babbel,"https://www.babbel.com/en/magazine/the-10-most-spoken-languages-in-the-world/, (Accessed on 09/30/2019).

[8] B. van Aken, J. Risch, R. Krestel, and A. L¨oser, "Challenges for toxiccomment classification: An in-depth error analysis," in Proceedings ofthe 2nd Workshop on Abusive Language Online (ALW2), 2018, pp. 33–42.

[9] B. Vidgen, A. Harris, D. Nguyen, R. Tromble, S. Hale, and H. Margetts,"Challenges and frontiers in abusive content detection." Associationfor Computational Linguistics, 2019.

[9] S. V. Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P.Plagianakos, "Convolutional neural networks for twitter text toxicityanalysis," in INNS Big Data and Deep Learning conference. Springer,2019, pp. 370–379.

[10] A. Obadimu, E. Mead, M. N. Hussain, and N. Agarwal, "Identifyingtoxicity within youtube video comment," in International Conferenceon Social Computing, Behavioral-Cultural Modeling and Prediction andBehavior Representation in Modeling and Simulation. Springer, 2019,pp. 214–223.

[11] S. C. Eshan and M. S. Hasan, "An application of machine learning todetect abusive bengali text," in 2017 20th International Conference ofComputer and Information Technology (ICCIT). IEEE, 2017, pp. 1–6.

[12] M. G. Hussain, T. Al Mahmud, and W. Akthar, "An approach to detectabusive bangla text," in 2018 International Conference on Innovation inEngineering and Technology (ICIET). IEEE, 2018, pp. 1–5.

[13] Spiros V. Georgakopoulos, Sotiris K. Tasoulis, Aristidis G. Vrahatis, Vassilis P. Plagianakos," Convolutional Neural Networks for Toxic Comment Classification" ,arXiv:1802.09957v1 [cs.CL] 27 Feb 2018.

[14] Mujahed A. Saif, Alexander N. Medvedev, Maxim A. Medvedev, TodorkaAtanasova, "Classification of Online

Toxic Comments Using the Logistic Regression and Neural Networks Models",2018

[15] Fahim Mohammad, "Is preprocessing of text really worth your time for toxic comment classification", Int'l Conf. Artificial Intelligence | ICAI'18,2018, pg no: 447-480.

[16] Pooja Parekh, Hetal Patel," Toxic Comment Tools: A Case Study", Volume 8, No. 5, May-June 2017, pg no: 964 – 967

[17] NavoneelChakrabarty ," A Machine Learning Approach to Comment Toxicity Classification", 2016

[18] Pallam Ravi, Hari Narayana Batta, Greeshma S, Shaik Yaseen," Toxic Comment Classification",Volume: 3, Issue: 4, 2019

[19] Aboujaoude, Elias, et al. "Cyberbullying: Review of an old problemgone viral." Journal of Adolescent Health 57.1 (2015): 10-18.

[20] Georgakopoulos, Spiros V., et al. "Convolutional Neural Networksfor Toxic Comment Classification." arXiv preprint arXiv:1802.09957(2018).