Toxic Comment Classification Using Hybrid Deep Learning Model

Rohit Beniwal and Archna Maurya

Abstract With the increasing availability of affordable data services and social media presence, our life is not untouched with 'cyber,' i.e., electronic technology. With it, various challenges and issues are faced, and the most sensitive among them is Cyberbullying. Cyberbullying is the form of 'abusive,' 'offensive,' 'inappropriate,' and 'toxic' comments that are present on the platforms. With the fear of online abuse and bullying, many people give-up on perceiving different opinions and stop expressing themselves. Nowadays, various online platforms like Quora, Wikipedia, Twitter, and Facebook have become part and parcel of everybody's life. These stages battle to viably encourage discussions, driving numerous networks to restrict or shutdown client remarks. Unfortunately, online comments with toxicity cause online badgering, bullying, and personal attacks. Therefore, toxic comment classification problem has attracted the attention of many organizations from the past few years. Hence, in this paper, we present a hybrid Deep Learning model that will detect such toxic comments and classify them according to the type of toxicity. As an outcome, we achieved the best results with an accuracy of 98.39% and an f1 score of 79.91%.

Keywords Bidirectional-gated recurrent unit · Convolution · Deep neural network · Multi-label classification · Toxic comments

1 Introduction

In today's digital era, social media provides a common platform that let users express their opinion in the form of online comments. People consider it their freedom of expression; however, many users use this fundamental right in a negative way, such

R. Beniwal · A. Maurya (✉)
Department of Computer Science and Engineering, Delhi Technological University, New Delhi 110042, India
e-mail: ms.archnamaurya96@gmail.com

R. Beniwal
e-mail: rohitbeniwal@yahoo.co.in

© The Author(s), under exclusive license to Springer Nature Singapore Pte Ltd. 2021

P. Karuppusamy et al. (eds.), *Sustainable Communication Networks and Application*, Lecture Notes on Data Engineering and Communications Technologies 55,
https://doi.org/10.1007/978-981-15-8677-4_38

as disrespecting other users, threatening other users, spreading fake news, cyberbullying, personal comments, toxic comments, etc. on online discussion platforms. Those comments, which are disrespectful and rude, and that force users to leave the conversation or online discussion are called "Toxic Comments." Nowadays, users face issues like abuse, harassment, cyberbullying online threats, and hate speeches, which can be classified as toxic comments. Therefore, such comments need to be recognized as quickly as possible and should be removed from the internet, but it is not that simple. It is a tedious task to filter and ban such comments. According to the Pew survey (2014) [1] about online harassment, some key findings are that every four in ten, i.e., 40% of internet users are victims of online harassment, purposeful embarrassment, and stalking. Both men and women experience a different kind of online harassment where women face it more frequently. Men experience fewer instances of verbal abuse, embarrassment, and threats, which are "less severe." In contrast, women experience badgering, such as being followed, inappropriate behavior, and threats on a more severe level.

Various online platforms are taking different initiatives to make their platform free from problems such as toxic comments, online harassment, and provide a safe online environment for their users. A few of the platforms even turn off comments for such posts based on crowdsourcing votes (upvotes/downvotes). Manually identifying such comments and flagging them is a time-consuming and challenging exercise. However, such an exercise is inefficient and not scalable. Comment classification is a classic example of Natural Language Processing (NLP) and a fundamental part of numerous applications such as web search, text mining, and sentiment analysis, etc. Hence, a wide scope of machine learning strategies has also been applied for comment classification.

The Deep Learning model identifies whether or not a comment is toxic. In the case of toxic, it further categorizes the comment in six different labels, namely toxic, severe toxic, obscene, threat, insult, and identity hate. All the listed labels are not mutually exclusive. Comment classification problems are generally also known as multi-class classification or multi-label classification [2]. Multi-class classification means data or comment belongs to only one out of the six labels. In contrast, Multi-label classification comment belongs to more than one label simultaneously. For example, a comment can be both insulting and threatening simultaneously.

In the recent past, Google and Jigsaw started a venture called "Perspective", which uses AI to distinguish toxic comments naturally [3]. The perspective API [4] score represents the impact of the comment in the discussion so that platforms can use this score to provide real-time feedback to users. In most of the cases, this model is not reliable, inclined to blunders, and the degree of toxicity is not determined.

Therefore, in this paper, we are using a hybrid Deep Learning model for improving the performance of toxic comment classification. To be particular, we analyze the dataset to understand how to process the data. Preprocessing and word embedding layer form a matrix, then we feed the matrix to Convolutional Neural Network (CNN) and Bidirectional Gated Recurrent Unit (Bi-GRU) layer respectively. Noise and important features are filtered out through CNN. After this, dense and dropout layers further perform the classification. Hence, we provide a multi-label classification

model that is capable of recognizing different types of toxicity, such as severe toxic, threats, obscenity, insults, and identity-based hate. Moreover, we are providing probability estimates for each sub-type, which is conclusively strong enough to outperform 'Perspective' API's current models.

The rest of the paper is organized as follows: Sect. 2 deliberates the related work; Sect. 3 defines the proposed methodology followed by Sect. 4, which describes its implementation; Sect. 5 examines the result and analysis; finally, Sect. 6 concludes the research paper and provides direction for future work.

2 Related Work

Toxic comments have a profound impact on a user's health online as well as offline. There have been several research papers on detecting toxic comments in online discussions. Most of the work is based on machine learning, text classification, sentiment analysis, and Deep Learning neural network. Abusive comment classification work started with Yin et al. [5] paper in which they used Support Vector Machine (SVM) and Term Frequency-Inverse Document Frequency (TF-IDF) features and compared the performance with a simple TF-IDF model on a chat-style database. Nguyen [6] proposed a model for sentiment label distribution using a hybrid model of bidirectional Long Short Term Memory cell (LSTM) model with word-level embedding and Convolutional Neural Network (CNN) model with character embedding technique on Stanford Twitter sentiment corpus. This hybrid model achieved an accuracy of 86.63%. Chu and Jue [7] compared Deep Learning models such as Recurrent Neural Network (RNN) models as LSTM with word embedding, CNN model with character embedding, and CNN model with word embedding. CNN, with the character embedding model, performed best between them with an accuracy of 94%. This paper also specified that character level embedding has improved performance than word-level embedding for CNN. In the real-life for practical applications such as automatic comment moderation, CNN with word embedding was suggested.

Georgeakopoulos [3] proposed a Deep Learning approach using CNN for toxicity classification in the text classification and compare the performance with SVM, K-nearest neighbors (KNN), Naïve Bayes (NB) and Linear discriminated analysis (LDA). NB and KNN had the lowest precision and recall scores. It means that they classify some non-toxic comments to toxic comments and vice versa. CNN had the best precision and recall score. CNN also attained the best performance with an accuracy score of 92.7%. Khieu and Narwal [8] used different Deep Learning models for toxic comment classification. They used SVM, LSTM, CNN, multilayer perceptron in combination with word and character-level embedding models for toxicity detection. They evaluated their model on the Kaggle toxic comment classification challenge dataset. LSTM model achieved the best performance with an accuracy of 92.7% and an f1 score of 70.6%.

As far as the above models or approaches are concerned, we provide a model in this paper, combining both CNN and Bi-LSTM Deep Learning models with word

embedding that increases the accuracy of toxic comments classification along with the F1 score.

3 Methodology

Our methodology for detecting toxic comments and classifying them according to the type of toxicity is divided into the following ten phases, namely dataset used, data preprocessing, embedding layer, convolution layer, max-pooling layer, Bi-GRU layer, global max-pooling layer, dropout layer, and two dense layers. Figure 1 represents the proposed methodology.

3.1 Dataset Used

In this research paper, we will be using the dataset available from the Kaggle Competition [9] known as the "Toxic Comment Classification Challenge". This dataset is a collection of comments from "Wikipedia's talk page edit". The dataset contains 159,571 comments that have been rated by humans for six sorts of toxicity labels such as toxic, severe toxic, obscene, threat, insult, and identity hate.

3.2 Data Preprocessing

Dataset is a collection of real-world data; however, such data is generally inconsistent and incomplete that requires data preprocessing. Data preprocessing helps us to clean, format, and organize the raw data. To achieve the same, firstly, we will remove Stopwords from the dataset. "Stop words are common English words such as, the, am, there; which do not influence the semantic of the review and removing them can reduce noise" [10]. Secondly, Tokenization will be performed. "Tokenization is the process of splitting the input into meaningful pieces" [11]. These pieces are called tokens of words. At last, the padding sequence will be used to make each comment of the dataset into the same length.

3.3 Embedding Layer

In the third phase, the Embedding layer will be used for mapping the words of comment on a vector of real numbers. For each unique word, the corresponding vector will be assigned in the space. There are various methods for creating word embeddings such as Glove, Word2vec, and FastText.

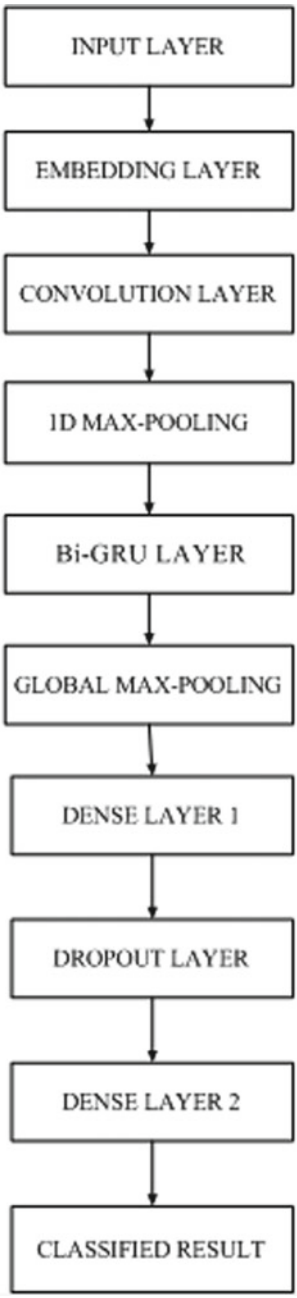

Fig. 1 Proposed hybrid deep learning model for toxic comment classification

3.4 Convolution Layer

In the fourth phase, the embedding layer output will feed into the 1D convolution layer. The Convolution layer is the center structure of the Convolutional Neural Network (CNN). The primary target of convolution will be to extract features from the input and pass its outcomes to the next layer.

3.5 Max-Pooling Layer

In the fifth phase, the yield of convolution will be transferred to the 1D Max pooling layers. The pooling layer will be used for reducing the dimension of processed data and only keeps important information. This layer will reduce the computation cost of the network. The pooling layer will diminish the features that decline the likelihood of overfitting.

3.6 Bidirectional-GRU Layer

The yield of Max pooling will be transferred into the bidirectional GRU layer. Bi-GRU is a kind of bidirectional recurrent neural network. It is almost similar to the bi-LSTM model. "GRU is faster than LSTM because it requires less calculation to refresh its concealed state" [12, 13]. GRU also overcomes the problem of vanishing gradient.

3.7 Global Max-Pooling Layer

In this phase, we will be using the 1D global max-pooling layer. The global constraint will yield the absolute most significant feature of the feature map rather than a feature window. The global max-pooling layer will reduce the dimension of input to one.

3.8 Dense Layer 1

In this phase, the dense layer will utilize the Relu (Rectified linear unit) activation function. A dense is only a normal layer of neurons in a neural system. This dense layer will receive the output from all the neurons of previous layers and help in refining the flow of gradient.

3.9 Dropout Layer

In this phase, we will use the dropout layer to dodge the issue of overfitting in the system. This layer will remove extra neurons from the neural network during the training phase and reduce the complexity of the model.

3.10 Dense Layer 2

Lastly, in the tenth phase, the last dense layer will utilize the sigmoid activation function for multi-label classification. The number of neurons in the last layer will be the number of classes in our dataset.

4 Implementation

The following sub-sections elaborate phase-wise implementation details of the proposed methodology. We used the Python programming language for the implementation of our model. TensorFlow and Keras libraries were used for building the neural network. To provide GPU support, we implemented our model on the Kaggle platform. After data preprocessing, we started building our model. We set up our input layer. As mentioned in the Keras documentation, we have to include the shape for the very first layer, and then Keras will automatically derive the shape for the rest of the layers.

4.1 Dataset Used

Exploratory Data Analysis (EDA) was performed on the dataset that gives us important information regarding the dataset and provides a way to handle the data for the model. Dataset was split into preparation and validation set into the 90:10 ratios. Table 1 defines the distribution of labels in the training set. Figure 2 shows the distribution of the number of comments vs. the length of comments that represent that the vast majority of comments were inside 500 characters length. Figure 3 shows sample instance of the dataset schema and database schema was represented as <id, comment, toxic, severe toxic, obscene, threat, insult, identity hate> that helps in understanding the dataset for further use in the model.

Table 1 Distribution of labels in the training set

Comment label	Number of comments
Toxic	15,294
Severe toxic	1595
Obscene	8449
Threat	478
Insult	7877
Identity hate	1405

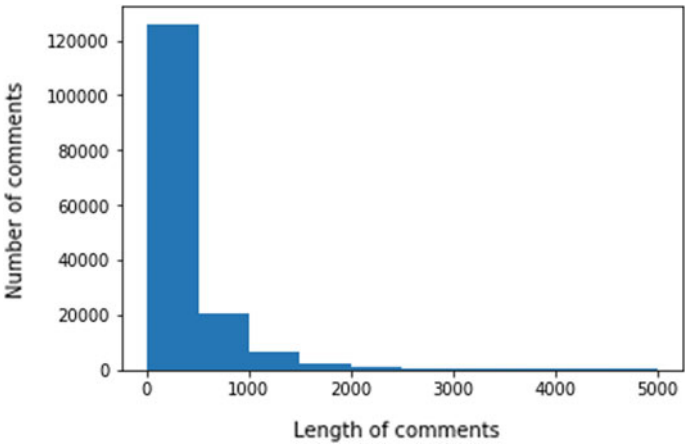

Fig. 2 Distribution of Comment length

	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'aww! 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. 3 Sample instance of the dataset

4.2 Data Preprocessing

Data preprocessing was used to increases the quality of the dataset and making it ready for model implementation. Firstly, we removed Stopwords using Python built-in dictionary of stopwords Nltk.corpus. Secondly, we performed Tokenization using keras.preprocessing.text() function. At last, we performed the padding sequence using keras.preprocessing.sequence() function. Padding sequence in Deep Learning was used to make each comment of the dataset into the same length. We assume the maximum length of a comment to be 500 and then add padding sequences

at the end of shorter comments to make their length equal to 500. Beautiful soup Python library was utilized for hauling information out of HTML and XML records that exist in the dataset. This phase was mainly used for converting the dataset into the standard form for further implementation.

4.3 Embedding Layer

In our model, embedding will be made dependent on the tokenized text. TF-IDF tokenization will make a lattice of features which would be utilized to develop the embedding. Each comment of the dataset was changed over to a one-dimensional vector of numerical components paying little mind to the tokenization strategy. To handle the comment of variable length padding was used and the vector was filled with zeros at the end so that all the comments have equal length. We assume the size of the embedding word vector to be 240d.

4.4 Convolution Layer

In the 1D convolution layer, we used 100 filters with length 4. By 1D convolution, we understand that "the kernel used here for convolution was a one-dimensional vector" [13]. Adding CNN on the top of the GRU helps in the sense that CNN combines with the polling layer brings out the important temporal features devoid of any noise which bidirectional GRU can use more effectively. The yield of this layer is conveyed to the 1D Max pooling layers.

4.5 Max-Pooling Layer

Various types of pooling techniques could use, e.g., 1D max pooling, global, max, average, and sum that depends on the architecture of the model. We passed the output of convolution to the 1D max-pooling layer that applied the max pool operation on the window of every four characters. As the output, we get a matrix of size = number of sentences * 125 * 100, which was also called Extracted Features. These extracted features were then transferred into the bidirectional GRU layer.

4.6 Bidirectional-GRU Layer

Bidirectional GRU has only two gates, the reset and update gate. "The reset gate(r) was utilized to choose how much past state data was required to keep and to overlook" [12].

$$r^{(t)} = \sigma\left(W^{(r)}x^{(t)} + U^{(r)}h^{(t-1)}\right) \text{ (Reset gate)} \quad (1)$$

The update gate (z) worked similarly to the forget gate and input gate of the LSTM model. "The update signal $z^{(t)}$ is responsible for determining how much of the hidden state should be carried forward to the next state" [12].

$$z^{(t)} = \sigma\left(W^{(z)}x^{(t)} + U^{(z)}h^{(t-1)}\right) \text{ (Update gate)} \quad (2)$$

Bi-GRU had two units of the recurrent network, one unit to move the data in a forward way, and the second unit moved the data in a backward way with the help of reset and update gate.

4.7 Global Max-Pooling Layer

The next global max-pooling layer reduced the dimension of input to one. For example, if we had data [2,3,4,5,6,6,7] with pool length 3 yield was 4,5,6,6,7 respectively; however, if 1D global max-pooling was used, then yield equal to 7. We performed 1D global max-pooling using Keras.layers() function.

4.8 Dense Layer 1

This layer produced the yield of dimension 50. We performed this dense layer using Keras.layers() function and utilized the Relu (Rectified linear unit) activation function.

4.9 Dropout Layer

The yield of the dense layer passed to the Dropout layer, which impaired a few neurons in the following layer so that the entire system could conclude better. The dropout rate was set at 20% and performed using Keras.layers() function.

Table 2 Performance of hybrid deep learning model

Accuracy	Precision	Recall	F1 score
98.39	86.05	74.59	79.91

4.10 Dense Layer 2

The last dense layer utilized the sigmoid activation function for multi-label classification using Keras.layers() function that created six-dimensional vectors, defined as the labels of toxicity. The final output file contained the probability of labels occurring on the dataset.

We used binary cross-entropy loss function because this function is more effective on classification tasks compared to other loss functions. Adam optimizer is designed to improve the classic Stochastic Gradient Descent (SGD) optimizer. This model minimized the log-loss function using the SGD optimization algorithm. The model was trained using batch size 32 and run for 20 epochs. A 10% size validation dataset was likewise passed on along.

5 Result and Analysis

A confusion matrix is utilized to calculate the performance of the characterized model. We report the result of our model using standard Precision, Recall, Accuracy, and F1 measure [14]. "F1 score is defined as the harmonic mean of precision and recall score." Accuracy is defined as the ratio of exactly matched instances to total instances.

We partitioned the dataset into training and testing sets. The training set contains 143,613 comments and the testing set contains 15,958 comments. After training and testing, we got the best results as shown in Table 2.

6 Conclusion and Future Work

There have been ceaseless trials of experiments to detect the presence of toxic comments of various kinds on online platforms. This holds importance in the research field due to the tremendously growing online interactive communication among users. Toxic comment classification is used for detecting the toxicity in social media platforms. Our work is dedicated to finding the best possible solution for toxic comment classification. This paper has implemented a deep learning based model using convolution and bidirectional gated recurrent units that successfully performs the multi-label classification of different sorts of toxic comments. As an outcome, we

achieved the best results with an accuracy of 98.39%, a precision score of 86.05%, and a recall score of 74.59%, and the computed F1 score of 79.91%.

"Common challenges for toxic comment classification among different datasets that contain out-of-jargon words with its long-range dependencies, and multi-word phrases" [15]. The limitation of this model is that it is trained on Google Jigsaw's toxic comment dataset and achieved a high accuracy on our test set. However, the proposed model may not be able to achieve the same level of performance on other datasets like twitter dataset. Another limitation is for handling the out of vocabulary words and our dataset is mostly a collection of English language comments, so our model works on the English language only. Google Jigsaw's toxic comment dataset is a collection of comments from "Wikipedia's talk page" and comments have been labeled by human raters [3, 7] because there is no standard definition of toxic labels, human raters rate the comments on their personal beliefs and therefore this dataset is skewed.

For future work, the proposed work will be extended to experiment with our model and the pre-trained word embedding techniques like Glove, Word2vec, and FastText trained on toxic comment dataset. Many enhancements may be possible to our model by adding consideration based instruments for better detection of toxic comments. Different users use different number of online platforms for discussions, so developing different models for each platform is not an efficient way to handle this problem; therefore, it is required to build a solitary framework that works over various platforms.

Reference

1. Duggan, M.: Online harassment (2017)
2. Risch, J., Krestel, R.: Toxic comment detection in online discussions. In: Deep Learning-Based Approaches for Sentiment Analysis, pp. 85–109. Springer, Singapore (2020)
3. Georgakopoulos, S.V., Tasoulis, S.K., Vrahatis, A.G., Plagianakos, V.P.: Convolutional neural networks for toxic comment classification. In: Proceedings of the 10th Hellenic Conference on Artificial Intelligence, pp. 1–6 (2018)
4. Perspective. https://perspectiveapi.com/
5. Yin, D., Xue, Z., Hong, L., Davison, B.D., Kontostathis, A., Edwards, L.: Detection of harassment on web 2.0. In: Proceedings of the Content Analysis in the WEB, vol. 2, pp.1–7 (2009)
6. Nguyen, H., Nguyen, M.L.: A deep neural architecture for sentence-level sentiment classification in twitter social networking. In: International Conference of the Pacific Association for Computational Linguistics, pp. 15–27. Springer, Singapore (2017)
7. Chu, T., Jue, K., Wang, M.: Comment abuse classification with deep learning. Von https://web.stanford.edu/class/cs224n/reports/2762092.pdf abgerufen (2016)
8. Khieu, K., Narwal, N.: Detecting and classifying toxic comments. Web: https://web.stanford.edu/class/archive/cs/cs224n/cs224n, 1184
9. Toxic Comment Classification ChallengeKaggle. https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge/data
10. Maalej, W., Nabil, H.: Bug report, feature request, or simply praise? On automatically classifying app reviews. In: IEEE 23rd International Requirements Engineering Conference (RE), pp. 116–125 (2015)

11. Mullen, L.A., Benoit, K., Keyes, O., Selivanov, D., Arnold, J.: Fast, consistent tokenization of natural language text. J. Open Source Softw. **3**(23), 655 (2018)
12. Dey, R., Salemt, F.M.: Gate-variants of gated recurrent unit (GRU) neural networks. In: IEEE 60th International Midwest Symposium on Circuits and Systems (MWSCAS), pp. 1597–1600 (2017)
13. Saeed, H.H., Shahzad, K., Kamiran, F.: Overlapping toxic sentiment classification using deep neural architectures. In: IEEE International Conference on Data Mining Workshops (ICDMW), pp. 1361–1366 (2018)
14. Precision and recall. https://en.wikipedia.org/wiki/Precision_and_recall
15. Van Aken, B., Risch, J., Krestel, R., Löser, A.: Challenges for toxic comment classification: an in-depth error analysis. arXiv preprint arXiv:1809.07572 (2018)