Combating against Toxicity using NLP and Deep Learning Models

Mahima Mali[1], Ankita Jena[2], Jensy Doshi[3], Archana Nanade[4]
[1,2,3]*Department of Computer Science, MPSTME, Mumbai*
[4]*Assistant Professor, Department of Computer Science, MPSTME, Mumbai*

***Abstract*- Social media and online communities struggle to deal with increasing hate speech, abuse, and cyberbullying, toxic text detection has become a crucial Natural language processing challenge. In this paper, we propose and compare two deep learning models for better text toxicity detection: Long Short-Term Memory (LSTM) and Bi-Directional Gated Recurrent Unit (GRU) neural networks, along with GloVe and FastText embeddings. We also implemented ensemble techniques on the two top performing models. We evaluated the performance of our proposed models on the widely used dataset, the Jigsaw Toxic Comment Classification Challenge. We achieved an accuracy of 92.41% after combining both Bi-GRU with GloVe embeddings and Bi-GRU with FastText embeddings using model averaging ensemble techniques. We also encountered some research gaps while training the models: Biases affecting the models out of which algorithmic bias has a strong influence on what is considered abuse or slurs and even making inaccurate and unfair predictions, another gap is the lack of proper training data affecting the detection of toxicity in conversations where the language is dynamic and changing. We proposed four research questions based on the identified gaps and tried to answer them with experimentation and literature review.**

Index Terms: Bi-GRU, Classification, Deep Learning, FastText, GloVe, LSTM, NLP, Toxicity

1. INTRODUCTION

Microblogging websites today offer an open and public space for people to express their ideas and opinions, but the sheer volume of posts, comments and messages shared makes it almost impossible to control the content being uploaded. Furthermore, because of the broad range of cultures, customs, and beliefs many people tend to use aggressive and hateful language while conversing with those who do not share the same backgrounds. While such hostility may seem normal, it in fact has an insidious influence on our mental health and in extreme cases incites acts of riots and terrorism [12]. A rise in hate speech on social media correlates to an increase in crimes against minorities in the real world, according to a study by Cardiff University. Former Facebook product manager Frances Haugen alleged in testimony before a Senate Commerce Committee in October 2021 that Facebook was fully aware social media was harming people's mental health and that the spread of misleading information was having an impact on society (Bartz, 2022) [45]. Platforms such as Twitter have struggled to effectively facilitate dialogues. According to statistics, in 2016 anywhere from 9% and 25% of people on Instagram claim to have experienced bullying, with the issue being much more prevalent on Twitter and Facebook [12]. As a result, many communities have started to restrict or completely disable user comments [48]. It is important to be able to protect online communities and conversation threads from turning hostile. Therefore, we see it as a necessity to be able to identify such speech and censor any information that contains toxic words or phrases.

Though Internet content has now evolved into formats that did not exist before such as GIFs, Infographics, and carousel posts, most of the content is text-based. Projections from IDC show that 80% of worldwide data will be unstructured data by 2025 typically text-heavy and not following a predefined data model [67]. Natural Language Processing (NLP) transforms such unstructured data into usable formats. NLP provides the context to analyze and better manage such data. It has become an essential tool for detecting abusive, disrespectful, and hateful words on social media. Finally leading to better content moderation through early detection of hate and abuse online.[3] NLP is

important because it works to resolve ambiguity in language and adds useful analytical structure to the data. Existing work in this area has approached the problem of detecting toxic speech as a classification task.

In the age of technology and the Internet, online conversation quality is vital. Focusing on constructive feedback and being able to classify it distinct from online hate raises the standard of online discourse and educates users. Additionally, there are an increasing number of comments that are so poisonous that they must be completely disabled. As a result, filtering harmful comments based on their severity enhances online dialogues [24].

While traditional machine learning methods have achieved notable accomplishments in knowledge discovery, they may prove inadequate in achieving satisfactory performance when confronted with intricate data types, such as imbalanced datasets, high-dimensional data, or data with a high level of noise. The reason behind this is that it is difficult for these methods to capture multiple characteristics and underlying structure of data [62]. Hence, ensemble frameworks are being used to effectively manage the efficiency and performance of the model. Ensemble method is a technique in which multiple models are being created and then combined to produce improved results. More accurate solutions are usually produced with the help of ensemble methods than a single model would. This has been the case in several machine learning competitions, where the winning solutions used ensemble methods [63].

The aim of our research paper is to build and evaluate a toxic speech classifier using NLP and deep learning approaches: LSTM with GloVe embeddings and Bi-GRU with GloVe and Bi-GRU with FastText embeddings and finally applying ensemble technique on the best two performing models using model averaging to improve the performance of the models. We then identified some research gaps like biases affecting the performance of the models and various methods to mitigate the biases. We have attempted to answer the four research questions proposed in Section __. We have implemented and evaluated the ability of various relevant NLP techniques to classify and detect toxicity efficiently. Our analysis classifies the comments to a conversation to determine whether it belongs to any of the various types of toxic classes such as, obscene, identity hate, threats, toxic, insults, or severe toxic. The input to our proposed model is using a dataset of conversational comments from the largest online encyclopedia, Wikipedia.

The remaining paper is divided into following sections. Section 2 discusses the related work in the field of NLP and deep learning approaches to detect toxicity. Section 3 proposes the research questions to be answered. Section 4 presents the proposed methodology being used to answer the proposed research questions, the dataset used and the proposed algorithms. Section 5 is devoted to the results of the experimentation done for RQ1 and RQ2. Section 6 is a discussion in which the answers to RQ3 and RQ4 are being discussed. Section 7 and 8 concludes the paper as well as proposes the future work which can be done to take our research further.

2. LITERATURE REVIEW

We have reviewed 66 papers in total to get a better understanding of the topic and the methods and models used to detect the toxicity of comments.

Detecting toxic comments and controlling them is a major and difficult task. The existing work has implemented the NLP, ML as well as DNN approaches with various transformers, embeddings and almost every possible combination of models and transformers. Hate speech is a particular form of offensive language where the person using it is basing his opinion either on segregate, racist or extremist background or on stereotypes. With the help of various newly proposed approaches, the classification of toxicity in comments has been an important research field. The research and analysis of the paper [2] provided a novel usage of the NLP approach to classify the type of toxicity in comments such as obscene, identity hate, threat, toxic, insult, and severely toxic by using the LSTM model which gave an accuracy of 94%.

Kohli, M., Kuehler, E., & Palowitch, J. [23] examined the approaches to classify online abuse using deep learning approaches by building RNN models and introducing variants of existing models based on TF-

IDF sentence vectors that performed well. The authors trained their own GloVe model. Alshamrani, S. et. al. [9] studied various toxic behaviours such as hate and obscenity from various news topics posted on mainstream media and news channels on youtube. Zhao, Z., Zhang, Z., & Hopfgartner, F. [13] carried out a learning curve analysis to perform a comparative study between CNN, SVM and LSTM amongst which in terms of F1 score CNN outperformed whereas SVM outperformed in terms of Precision.

Another interesting approach was taken by Jain, S., Kaushik, G., Prabhu, P., & Godbole, A. [5] in which a euphemistic substitution approach was developed by using NLP and ML to detect toxic texts and provide a polite substitute to replace the toxicity. In fact, C. Cross, S. Munukutla, and T. S. Yong [15] built a censorship filter at Stanford in 2019 utilizing CNN-LSTM Model trained on the Sina Weibo dataset.

The research done in paper [4] examined current popular transformer models and examined the effects of various pre-processing methods and text representations, such as the standard TD-IDF and pre-trained word embeddings. The Kaggle toxic comment classification dataset was utilised for experiments, and the best-performing model was compared with related techniques using common metrics for data analysis. The bare BERT model outperformed amongst all the other models with an average accuracy of 97.41%. The paper [14] aims to filter out toxicity from social media by text mining and making use of deep learning models constructed using LSTM neural networks. The model classifies a given sentence as toxic or non-toxic and also gives the percentage of toxicity or non-toxicity of the given sentence.

Detecting toxicity can be biased by various factors which many researchers pointed out in their research work. To understand the differences in conceptions of offense between men and women the authors Reuben Binns, Michael Veale, Max Van Kleek and Nigel Shadbolt [6] provided some exploratory methods and found that female annotators were less likely to agree with each other's offense scores than males. The authors of [8] proposed a methodology able to detect the toxicity of a comment based on the text as well as the emojis within that comment. The GloVe which is an open-source text-based toxicity detector was used which converts input text into vectors and then trains a bidirectional Long Short-Term Memory model (bi LSTM). In this work [3] the authors investigated the context of abusive language detection on Twitter and thus recommended that the context of the messages should also be provided to the annotators which would have a positive impact on the implementation of classification systems.

The authors of [7] combined two new datasets-ALONE- based on youth's toxic conversations and HASOC'20, and carried out preprocessing and ML algorithms, among which LR and XGBoost outperformed, and carried out DNN, among which CNN gave the best results. The paper [10] aims to employ several deep-learning architectures for estimating the toxicity of Georgian comments, among which CNN (88.8%) and bi-Bi-GRU-CNN (88.6%) outperformed. An unsupervised method was applied in another paper [11] called the Local Interpretable Model-Agnostic Explanations (LIME) with LSTM (GloVe) classifier to test toxic spans. The models were evaluated using the F1 score, accuracy, precision, and recall.

According to the authors Aggarwal and Zhai, 2012; Xia et al., 2011, ensemble methods for text classification, such as 2546 stacking and bagging, are commonly used approaches. In social media, simple but effective ensemble approaches have been used for the sentiment classification of Tweets [64]. The use of numerous models combined into one is known as an ensemble method, which leads in better results. When compared to a single model, ensemble approaches typically gives more precise results. In several machine learning as well as deep learning competitions, the winning solutions incorporated ensemble techniques [63].

After carrying out the literature review, we examined whether deep learning algorithms performed or obtained better results and accuracy than ML algorithms. In deep learning approaches, LSTM with GloVe and Bi-GRU with GloVe embeddings as well as with FastText embeddings were the ones that outperformed. We also found that ensemble methods and approaches can even increase the performance of the models. There are still some drawbacks and more research to carry out in this area. The drawbacks

which we found were: Inaccurate or poor-quality datasets, and biases affecting the accuracy of the model and only classifying the comments into binary classes instead of multilabel classes accurate.

3. RESEARCH QUESTIONS

After reviewing the existing work, we came up with some research questions which we have tried to answer through our research paper with the help of a literature survey and experimentation.

RQ1: How can we build a model that can detect different types of toxicity like threats, obscenity, insults, and identity-based hate? (Experimentation)

RQ2: Which combination of model and embeddings worked the best on the dataset, LSTM with GloVe, or Bi-GRU with GloVe embeddings and FastText Embeddings? (Experimentation)

RQ3: What are the biases which would affect the accuracy of the toxicity detection model? (Literature review)

RQ4: What are the methods that deal with the lack of proper training data or 'biases'? (Literature review)

4. METHODOLOGY

In this paper we have implemented LSTM with GloVe embeddings and Bi-GRU with GloVe embeddings as well as Bi-GRU with FastText Embeddings.

LSTM: LSTM is a type of neural network designed specifically for sequential data processing. It solves the problem of vanishing gradients in traditional RNNs by incorporating memory cells that can retain information over time and gates that control the flow of information. This allows LSTMs to effectively process long sequences of data and learn from them [2]. LSTMs are commonly used in NLP applications and time series forecasting, and can be trained with standard optimization algorithms. They can also be combined to create deep recurrent networks for more complex tasks [24].

Bi-GRU: A Bi-GRU model uses two GRU layers that process the input sequence in opposite directions - one layer processes the sequence forwards, while the other processes it backwards. This allows the model to capture both the past and future context of the input sequence, which can be useful for NLP tasks where context is important [24]. In addition to the Bi-GRU layers, the model also uses GloVe embeddings. By using pre-trained GloVe embeddings, the model can leverage this semantic information to improve its performance on NLP tasks. To use GloVe embeddings in a Bi-GRU model, the embeddings are first loaded into memory as a dictionary where the keys are the words and the values are the corresponding embedding vectors [4].

GloVe: GloVe is a technique for representing words in NLP as dense vectors of real numbers, learned through an optimization process based on co-occurrence matrices that measure the frequency of word pairs in a corpus [23]. The resulting vectors capture both semantic and syntactic information of words and have been shown to perform well in various NLP tasks, making GloVe a popular choice among researchers and practitioners due to its computational efficiency and ability to capture context and meaning [11].

FastText: FastText is an extension of Mikolov's embedding technique that utilizes the skip-gram model. In FastText, words are represented as a collection of character n-grams, and each n-gram is assigned a vector representation [16]. The word representation is then obtained by summing up these vector representations. This approach allows FastText to generate embeddings even for misspelled words, uncommon words, or words that were not part of the training corpus. Unlike Mikolov's embeddings, FastText employs character n-gram word tokenization, enabling it to handle such cases effectively [65].

LSTM (Long Short-Term Memory) with GloVe, Bi-GRU with GloVe and B-GRU with FastText embeddings uses pre-trained word embeddings to improve the performance of a text classification or generation task. Bi-GRU with GloVe and FastText is computationally efficient compared to LSTM and can still capture the sequential information in the text [23].

Ensemble methods can be used in LSTM and Bi-GRU models to improve their performance and increase their accuracy. We propose the following steps that can be followed:

1. Training individual LSTM and Bi-GRU models with GloVe and FastText embeddings on the selected dataset.
2. After training the individual models, we would evaluate their performance and select the two best-performing models.
3. We will then combine the best-performing models using an averaging method which is an ensemble method.
4. Finally, we use the ensemble model to make predictions on the validation dataset.

4.1 Dataset

The following dataset has been used: Jigsaw Comment Toxicity Challenge Dataset: The dataset contains Wikipedia comments that are annotated into six classes of toxicity type. It contains 159,571 comments, classified into the 6 labels (obscene, identity hate, threats, toxic, insults, to severe toxic). The dataset was created to identify and classify toxic online comments into multiple labels instead of just binary classes [1].

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
15	00078f8ce7eb276d	"\n\nJuelz Santanas Age\n\nIn 2002, Juelz Sant...	0	0	0	0	0	0
16	0007e25b2121310b	Bye! \n\nDon't look, come or think of comming ...	1	0	0	0	0	0
17	00089748926 8bc93	REDIRECT Talk:Voydan Pop Georgiev- Chernodrinski	0	0	0	0	0	0
18	0009801bd85e5806	The Mitsurugi point made no sense - why not ar...	0	0	0	0	0	0
19	0009eaea3325de8c	Don't mean to bother you \n\nI see that you're...	0	0	0	0	0	0

Figure 1: Sample Data

4.1.1 Data Fields

- id: id of the comment
- comment_text: the text of the comment
- toxic: the value of 0 (non-toxic) or 1 (toxic)
- severe_toxic: the value of 0 (non-severe_toxic) or 1 (severe_toxic)
- obscene: the value of 0 (non-obscene) or 1 (obscene)
- threat: value of 0 (non-threat) or 1 (threat)
- insult: value of 0 (non-insult) or 1 (insult)
- identity_hate: value of 0 (non-identity_hate) or 1 (identity_hate)

4.2 Exploratory Data Analysis

We carried out the exploratory data analysis for the dataset to analyze the data visually.

Training Data Comment Breakdown

A total of 159571 comments are there in the dataset out of which 16225 or 10.17%, are classified as toxic.

15294 toxic comments. (9.58% of all data.)
- 1595 or 10.43% were also severe_toxic.
- 7926 or 51.82% were also obscene.
- 449 or 2.94% were also threat.
- 7344 or 48.02% were also insult.
- 1302 or 8.51% were also identity_hate.
- 15294 or 100.00% were also any_label.

1595 severe_toxic comments. (1.00% of all data.)
- 1595 or 100.00% were also toxic.
- 1517 or 95.11% were also obscene.
- 112 or 7.02% were also threat.
- 1371 or 85.96% were also insult.
- 313 or 19.62% were also identity_hate.
- 1595 or 100.00% were also any_label.

8449 obscene comments. (5.29% of all data.)
- 7926 or 93.81% were also toxic.
- 1517 or 17.95% were also severe_toxic.
- 301 or 3.56% were also threat.
- 6155 or 72.85% were also insult.
- 1032 or 12.21% were also identity_hate.
- 8449 or 100.00% were also any_label.

478 threat comments. (0.30% of all data.)
- 449 or 93.93% were also toxic.
- 112 or 23.43% were also severe_toxic.
- 301 or 62.97% were also obscene.
- 307 or 64.23% were also insult.
- 98 or 20.50% were also identity_hate.
- 478 or 100.00% were also any_label.

7877 insult comments. (4.94% of all data.)
- 7344 or 93.23% were also toxic.
- 1371 or 17.41% were also severe_toxic.
- 6155 or 78.14% were also obscene.
- 307 or 3.90% were also threat.
- 1160 or 14.73% were also identity_hate.
- 7877 or 100.00% were also any_label.

1405 identity_hate comments. (0.88% of all data.)
- 1302 or 92.67% were also toxic.
- 313 or 22.28% were also severe-toxic.
- 1032 or 73.45% were also obscene.
- 98 or 6.98% were also threat.
- 1160 or 82.56% were also insult.

- 1405 or 100.00% were also any_label.

16225 any_label comments. (10.17% of all data.)
- 15294 or 94.26% were also toxic.
- 1595 or 9.83% were also severe_toxic.
- 8449 or 52.07% were also obscene.
- 478 or 2.95% were also threat.
- 7877 or 48.55% were also insult.
- 1405 or 8.66% were also identity_hate.

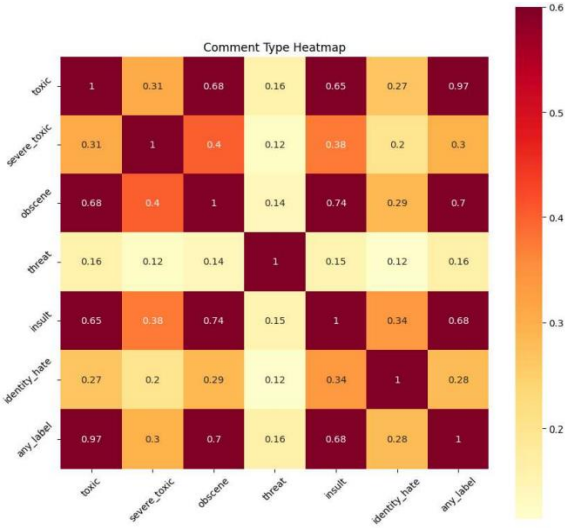

Figure 2: Heatmap showing correlation among variables

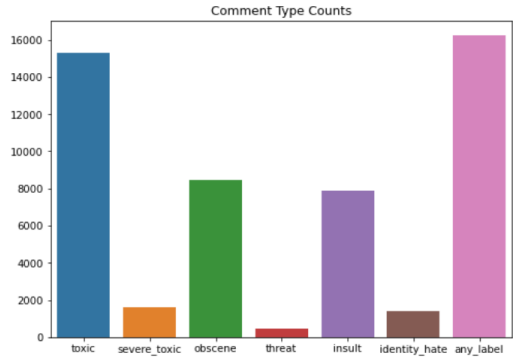

Figure 3: Count of each type of toxic comment

4.3 Pre-processing

For pre-processing of the data, we first set the parameters. max_len=120 which is the maximum number of words for the tokenizer. The final Vocabulary size: is 20000 which refers to the total number of unique words or tokens in a corpus of text, number of tokens = vocabulary_size+1 and embedding dimension as 300.

4.3.1 Cleaning of the dataset

Removing stopwords, converting the comment to lowercase, changing the contracted words into possible non-contracted form, and removing the non-English words. The function performs the following operations:

- Replaces newline characters with an empty string.
- It removes the punctuation and converts the text to lowercase.
- Strips white spaces and returns the cleaned text.

4.3.2 Tokenization

Tokenization is a process of breaking down a large piece of text into smaller units called tokens. Tokens are usually words, but they can also be phrases, numbers, symbols, or any other meaningful unit of text. Tokenization is an essential step in many natural language processing (NLP) tasks, such as text classification, sentiment analysis, and information retrieval. By breaking down the text into tokens, we can easily analyze and manipulate the text to extract useful information [15]. When performing tokenization, each unique word or token is assigned a unique identifier or index, which is used to represent the word in a numerical form that can be processed by machine learning algorithms [13].

```
tokenizer = Tokenizer(num_words = vocabulary_size+1,\
                      filters='!"#$%&()*+,-./:;<=>?@[\\]^_`{|}~\t\n0123456789',\
                      lower=True, split=' ')
```

Figure 4: Tokenization of the comments

4.3.3 Sequencing

Sequencing refers to the process of splitting a long sequence of data into smaller, more manageable chunks that can be processed by the LSTM. In the case of natural language processing, sequences are usually split into individual words or tokens [16].

Max_len=120 which is the maximum length of each sentence including padding.

4.3.4 Padding

Padding is a technique used in Natural Language Processing (NLP) to ensure that all sequences in a dataset have the same length. When training a recurrent neural network like LSTM, it is important that all input sequences have the same length so that they can be batched together and processed efficiently [15].

Padding involves adding extra tokens (usually zeros) to the end of shorter sequences to make them the same length as the longest sequence in the dataset. Once the sequences have been padded, they can be batched together and processed efficiently by the LSTM. When processing the padded sequences, the LSTM will ignore the zeros added for padding since they do not contain any meaningful information [16].

4.4 Embeddings

We used a combination of two-word embedding techniques as input to the deep learning classification model. We chose a 300-dimensional vector to represent each word in the vocabulary. The initialization was done using two pre-trained word embeddings. The word embedding techniques used for this work are:

1. GloVe [66] learns word embeddings by dimensionality reduction of the co-occurrence count matrix. Instead of learning raw co-occurrence probabilities, it learns ratios of co-occurrence probabilities to distinguish relevant words from irrelevant words. EMBEDDINGS_DIM=300 which is the dimensions for the GloVe embeddings is being used to train the model.

```
!unzip glove*.zip
!ls
!pwd
```

```
Archive:  glove.6B.zip
  inflating: glove.6B.50d.txt
  inflating: glove.6B.100d.txt
  inflating: glove.6B.200d.txt
  inflating: glove.6B.300d.txt
drive             glove.6B.200d.txt  glove.6B.50d.txt  sample_data
glove.6B.100d.txt  glove.6B.300d.txt  glove.6B.zip
/content
```

Figure 5: Unzipping GloVe Embedding

2. FastText Embeddings also called FT [65] is an algorithm created by Facebook that assumes every word to be n-grams of character. It helps to give vector representations for out-of-vocabulary words. For the current work, FastText embeddings5 is used for generating token vectors of dimension 300.

```
1  !unzip crawl*.zip
2  !ls
3  !pwd
```

```
Archive:  crawl-300d-2M.vec.zip
  inflating: crawl-300d-2M.vec
crawl-300d-2M.vec  crawl-300d-2M.vec.zip  drive  sample_data
/content
```

Figure 6: Unzipping FastText Embedding

4.5 Design Diagram

In this section we have discussed the pseudocode to detect the toxicity of the comments along with the design diagrams of all the proposed models.

4.5.1 LSTM with GloVe

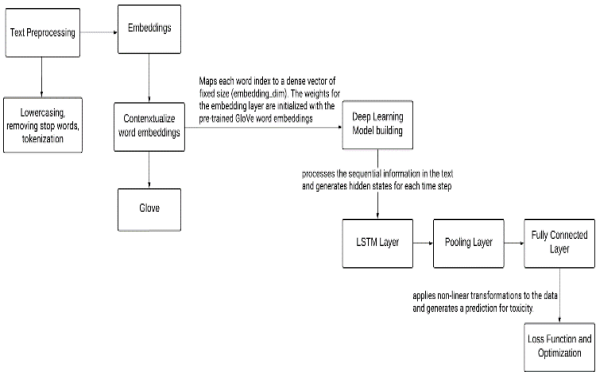

Figure 7: Design diagram of LSTM with GloVe

4.5.2 Bi-GRU with GloVe Embeddings

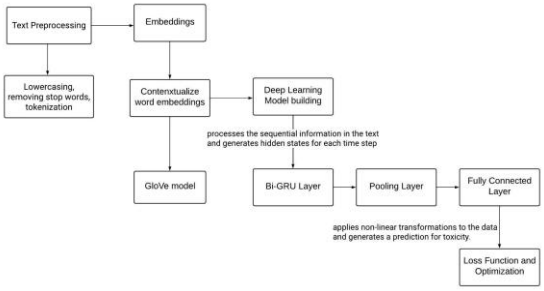

Figure 8: Design diagram of Bi-GRU with GloVe

4.5.3 Bi-GRU with FastText Embeddings

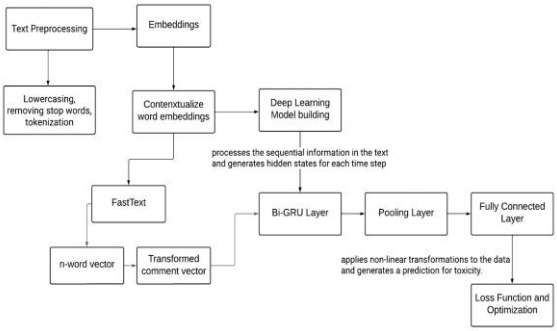

Figure 9: Design diagram of Bi-GRU with FastText

Pseudo Code to detect the toxicity of a comment:

if i is a comment:

then:

1. Import the libraries (numpy, pandas, Tensorflow, embeddings, LSTM, Bi-GRU)
2. Load the dataset
3. Data pre-processing

- Clean_Text (remove stopwords, lowercase)
- Tokenizer.fit_on_text
- Define vocabulary length len(word_index))
- Pad_sequences

4. Load GloVe as well as FastText embeddings
   embedding_dict = {}
   with open("GloVe.6B.100d.txt", "r")and
   ("crawl-300d-2M.vec", "r") as f:
   for line in f:
   values = line.split()
   word = values[0]
   vector = np.asarray(values[1:], dtype="float32")
   embedding_dict[word] = vector

5. Define and create embedding matrix
   embedding_matrix =
   np.zeros((vocabulary_size, 100))
   for word, i in tokenizer.word_index.items():
   embedding_vector = embedding_dict.get(word)
   if embedding_vector is not None:
   embedding_matrix[i] = embedding_vector

6. Split dataset (90-10 ratio or 80-20 ratio)
7. Define the model (this step is explained in detail in section 4.5)

8. Compile the model
   model.compile(optimizer="adam",
   loss="binary_crossentropy",
   metrics=["accuracy";_____])

9. Fit the model
   history = model.fit(x_train, y_train,
   validation_data=(x_test, y_test), epochs=2)

10. Visualize the results
    import matplotlib.pyplot as plt
    plt.show()

11. Select the 2 best-performing models to apply the ensemble approach using the model averaging method.

12. Finally test the model on the test dataset and classify the comments in the 6 classes mentioned providing the percentage of toxicity detected for the comment.

4.6 Training and Testing

In this section we have discussed the layers and hyperparameters used in each of the four models proposed.

4.6.1 LSTM with GloVe

Train and validation split for the LSTM with GloVe model was kept at an 80 to 20 ratio.

```
Model: "sequential_7"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 embedding_5 (Embedding)     (None, None, 300)         6000300

 lstm_3 (LSTM)               (None, None, 50)          70200

 global_max_pooling1d_6 (Glo  (None, 50)               0
 balMaxPooling1D)

 dense_5 (Dense)             (None, 50)                2550

 dense_6 (Dense)             (None, 6)                 306

=================================================================
Total params: 6,073,356
Trainable params: 6,073,356
Non-trainable params: 0
_________________________________________________________________
```

Figure 10: LSTM model layers

1. First, the input layer is created which specifies the shape of the input data, which in this case is (max_len,). The max_len parameter specifies the maximum length of the input sequence, which is a hyperparameter chosen based on the length of the longest comment in the dataset.
2. Then embedding layer takes in the input sequence and outputs a dense vector representation for each word. It maps each word index to a dense vector of fixed size (embedding_dim). The weights for the embedding layer are initialized with the pre-trained GloVe word embeddings (embedding_weights_GloVe), which capture the semantic relationships between words. This layer is made trainable to allow fine-tuning of the embeddings during training.
3. The LSTM layer that is used to model the sequential data. It processes the input sequence one element at a time while maintaining an internal state, which allows it to capture long-term dependencies in the data. The LSTM layer has 50 units and returns the output sequence of the same length as the input sequence.
4. We then added the GlobalMaxPooling1D layer which takes the maximum value across the entire output sequence of the LSTM layer, which allows it to capture the most salient features of the sequence.

5. A Dense layer is added to the model, it is a fully connected layer that takes in the output of the previous layer and applies a linear transformation followed by a non-linear activation function. In this case, the first Dense layer has 50 units and uses the relu activation function. The relu function applies the element-wise rectified linear function, which is known for its ability to learn non-linear relationships in the data.
6. The output Dense layer has 6 units and uses the sigmoid activation function, which is commonly used for multi-label classification problems. The sigmoid function applies the element-wise logistic function, which outputs a probability value between 0 and 1 for each label.
7. Finally, the model is compiled with optimizer='adam', loss='binary_crossentropy', and metrics= ['AUC', 'accuracy']. The built model is returned and a summary is printed.

4.6.2 Bi-directional GRU with GloVe

Train and validation split for the LSTM with GloVe model was kept at a 90 to 10 ratio.

```
Model: "GRU_model_glove"
_________________________________________________________________
 Layer (type)                Output Shape              Param #
=================================================================
 input_1 (InputLayer)        [(None, 120)]             0

 embedding (Embedding)       (None, 120, 300)          6000300

 spatial_dropout1d (SpatialD  (None, 120, 300)         0
 ropout1D)

 bidirectional (Bidirectiona  (None, 120, 84)          86688
 l)

 global_max_pooling1d (Globa  (None, 84)               0
 lMaxPooling1D)

 dense (Dense)               (None, 6)                 510

=================================================================
Total params: 6,087,498
Trainable params: 6,087,498
Non-trainable params: 0
_________________________________________________________________
```

Figure 11: Layers in GRU with GloVe model

1. The input sequences are first passed through an embedding layer, which maps each word index to a dense vector representation based on pre-trained GloVe embeddings.
2. A SpatialDropout1D layer is added to prevent overfitting by randomly dropping out entire 1D feature maps.
3. Two GRU layers are stacked on top of each other, one going forward and one going backwards (i.e., bidirectional GRU). Each GRU layer has 42 units and returns sequences (as opposed to just the final hidden state).
4. A GlobalMaxPooling1D layer is used to aggregate the sequence outputs from the bidirectional GRU layers into a fixed-length vector.
5. A Dense layer with sigmoid activation is added to produce binary outputs for each of the 6 categories.
6. Finally, the model is compiled with the Adam optimizer, binary cross-entropy loss, and AUC metric.
7. The function returns the compiled model, which is stored in the variable GRU_model_GloVe. A summary of the model architecture can be obtained by calling the summary() method on the model object.

```
def GRU_model_glove():
    global max_len,num_tokens,embedding_weights_glove

    inputs = layers.Input(shape=(max_len,))

    x = layers.Embedding(input_dim=num_tokens,\
                         output_dim=embedding_dim,\
                         embeddings_initializer=keras.initializers.Constant(embedding_weights_glove),\
                         trainable=True)(inputs)

    x = layers.SpatialDropout1D(0.3)(x)

    forward_layer = layers.GRU(42,return_sequences=True)
    backward_layer = layers.GRU(42,activation="relu",dropout=0.1,return_sequences=True,go_backwards=True)
    x = layers.Bidirectional(forward_layer,backward_layer=backward_layer)(x)

    x = layers.GlobalMaxPooling1D()(x)

    outputs = layers.Dense(units=6,activation='sigmoid')(x)

    model = keras.models.Model(inputs=inputs, outputs=outputs, name="GRU_model_glove")

    model.compile(optimizer=tf.optimizers.Adam(),\
                  loss=tf.losses.BinaryCrossentropy(),\
                  metrics=['AUC'])

    return model

GRU_model_glove = GRU_model_glove()
GRU_model_glove.summary()
```

Figure 12: Bi-GRU with GloVe model architecture

4.6.3 Bi-directional GRU with FastText

```
Model: "GRU_model"
__________________________________________________________________________________________________
 Layer (type)                Output Shape              Param #   Connected to
==================================================================================================
 input_5 (InputLayer)        [(None, 120)]             0
__________________________________________________________________________________________________
 embedding_4 (Embedding)     (None, 120, 300)          6000300   input_5[0][0]
__________________________________________________________________________________________________
 spatial_dropout1d (SpatialDropo (None, 120, 300)      0         embedding_4[0][0]
__________________________________________________________________________________________________
 bidirectional (Bidirectional)  (None, 120, 128)       140544    spatial_dropout1d[0][0]
__________________________________________________________________________________________________
 global_average_pooling1d (Globa (None, 128)           0         bidirectional[0][0]
__________________________________________________________________________________________________
 global_max_pooling1d (GlobalMax (None, 128)           0         bidirectional[0][0]
__________________________________________________________________________________________________
 concatenate (Concatenate)   (None, 256)               0         global_average_pooling1d[0][0]
                                                                 global_max_pooling1d[0][0]
__________________________________________________________________________________________________
 dense (Dense)               (None, 6)                 1542      concatenate[0][0]
==================================================================================================
Total params: 6,142,386
Trainable params: 142,086
Non-trainable params: 6,000,300
__________________________________________________________________________________________________
```

Figure 13: Layers in Bi-GRU with FastText

1. We have defined the GRU_model_fasttext function, which builds the model architecture using Keras layers. The input to the model is a sequence of integers representing the tokens in the text, with the sequence length set to max_len. The input is passed through an embedding layer

initialized with pre-trained FastText embeddings, with trainable=False to prevent the embeddings from being updated during training. The output of the embedding layer is passed through a SpatialDropout1D layer to reduce overfitting.

2. The core of the model consists of a Bidirectional GRU layer with 64 units in each direction, followed by GlobalAveragePooling1D and GlobalMaxPooling1D layers. The outputs of these layers are concatenated and passed through a Dense layer with sigmoid activation, which outputs a probability score for each of the six categories. The model is compiled with the Adam optimizer, binary cross-entropy loss, and AUC metric.
3. The model is then trained on a training set (x_train, y_train) for two epochs with a batch size of 32, and evaluated on a validation set (x_val, y_val). The performance of the model is evaluated using several metrics, including accuracy, F1-score, precision, recall, and a classification report.

```
def GRU_model_fasttext():
    global max_len,num_tokens,embedding_weights_fasttext

    inputs = layers.Input(shape=(max_len,))

    x = layers.Embedding(input_dim=num_tokens,\
                         output_dim=embedding_dim,\
                         embeddings_initializer=keras.initializers.Constant(embedding_weights_fasttext),\
                         trainable=False)(inputs)

    x = layers.SpatialDropout1D(0.3)(x)

    forward_layer = layers.GRU(64,return_sequences=True)
    backward_layer = layers.GRU(64,activation="relu",dropout=0.3,return_sequences=True,go_backwards=True)
    x = layers.Bidirectional(forward_layer,backward_layer=backward_layer)(x)

    avg_pool = layers.GlobalAveragePooling1D()(x)
    max_pool = layers.GlobalMaxPooling1D()(x)
    x = layers.concatenate([avg_pool,max_pool])

    outputs = layers.Dense(units=6,activation='sigmoid')(x)

    model = keras.models.Model(inputs=inputs, outputs=outputs, name="GRU_model")

    model.compile(optimizer=tf.optimizers.Adam(),\
                  loss=tf.losses.BinaryCrossentropy(),\
                  metrics=['AUC'])

    return model

GRU_model_fasttext = GRU_model_fasttext()
GRU_model_fasttext.summary()
```

Figure 14: GRU with FastText model architecture

4.6.4 Ensemble model

```
model_nums = 2
size1 = x_train.shape[0]

y_train_pred = np.zeros((model_nums,size1,6),dtype="float32")
y_train_pred[0] = GRU_model_fasttext.predict(x_train)
y_train_pred[1] = GRU_model_glove.predict(x_train)

size2 = X_test.shape[0]
y_test_pred = np.zeros((model_nums,size2,6),dtype="float32")
y_test_pred[0] = GRU_model_fasttext.predict(X_test)
y_test_pred[1] = GRU_model_glove.predict(X_test)

y_pred_ensemble = np.zeros((size2,6),dtype="float32")

for i in range(6):
    lg = LogisticRegression()
    temp = np.zeros((size1,model_nums),dtype="float32")
    for j in range(model_nums):
        temp[:,j] = y_train_pred[j,:,i]
    lg.fit(temp,y_train[bad_comment_cat[i]])

    temp = np.zeros((size2,model_nums),dtype="float32")
    for  j in range(model_nums):
        temp[:,j] = y_test_pred[j,:,i]
    y_pred[:,i] = lg.predict_proba(temp)[:,1]
```

Figure 15: Ensemble model architecture

1. First, we initialized y_train_pred and y_test_pred numpy arrays of size (model_nums, size1, 6) and (model_nums, size2, 6) respectively, where model_nums is the number of models being ensembled, size1 and size2 are the number of examples in the training and test sets, and 6 represents the number of output classes.
2. Then, we used the two models to predict the output probabilities for each training and test example and saves the predictions in the corresponding y_train_pred and y_test_pred arrays.
3. Next, we initialize y_pred numpy array of size (size2,6) where size2 is the number of examples in the test set and 6 is the number of output classes.
4. Finally, for each output class i, we initialize a LogisticRegression object lg and train it on the output probabilities of the i-th output class predicted by the two models in y_train_pred. It then uses the trained model to predict the probabilities of the i-th output class for the test set in y_test_pred and saves the predicted probabilities in the i-th column of y_pred.
5. This code essentially performs a simple ensemble by averaging the output probabilities of two models and then training a logistic regression model on the averaged probabilities to make the final predictions.

5. RESULTS

In this section we have discussed the results we have achieved from all the proposed models.

5.1 LSTM with GloVe Embeddings

Figure 16: Training and Validation Accuracy and Loss

```
Accuracy of the model :  0.91928813134478
F1-score:  0.7708327410668265
Precision Score :  0.8216045038705138
Recall Score :  0.671362852213916
AUC score :  0.6912231890921551
```

Figure 17: Metrics of LSTM Model

	precision	recall	f1-score	support
Toxic	0.85	0.73	0.78	1554
severe_toxic	0.50	0.32	0.39	146
obscene	0.86	0.79	0.82	824
threat	0.00	0.00	0.00	47
insult	0.77	0.70	0.73	768
identity hate	0.69	0.08	0.14	139
micro avg	0.82	0.68	0.75	3478
macro avg	0.61	0.44	0.48	3478
weighted avg	0.80	0.68	0.73	3478
samples avg	0.06	0.06	0.06	3478

Figure 18: Classification report

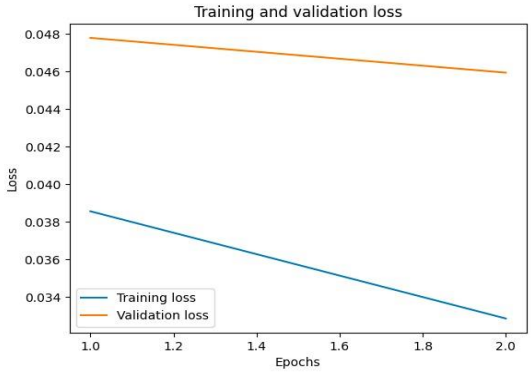

Figure 19: Training vs Validation Loss

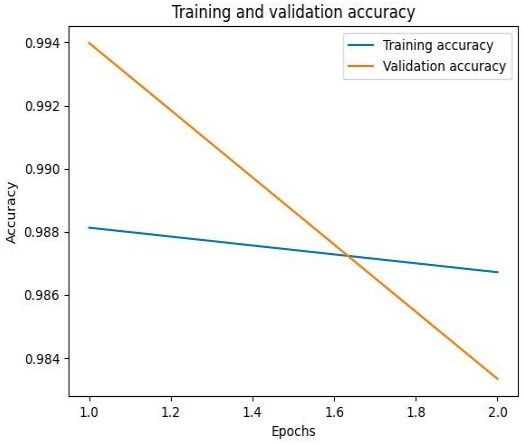

Figure 20: Training vs Validation Accuracy

5.2 Bi-GRU with GloVe

Figure 21: Training and Validation Accuracy and Loss

```
Accuracy of the model :  0.9233613234741196
F1-score:  0.7476814838503357
Precision Score :  0.8422190201729106
Recall Score :  0.672225416906268
```

Figure 22: Metrics for Bi-GRU with GloVe

	precision	recall	f1-score	support
Toxic	0.85	0.76	0.80	1554
severe_toxic	0.45	0.43	0.44	146
obscene	0.83	0.83	0.83	824
threat	0.50	0.19	0.28	47
insult	0.78	0.71	0.74	768
identity hate	0.67	0.35	0.45	139
micro avg	0.80	0.73	0.76	3478
macro avg	0.68	0.54	0.59	3478
weighted avg	0.80	0.73	0.76	3478
samples avg	0.07	0.07	0.06	3478

Figure 23: Classification report

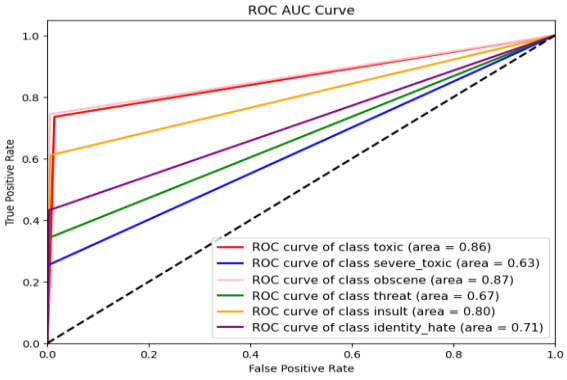

Figure 24: ROC curve

5.3 Bi-GRU with FastText

Figure 25: Training and Validation Accuracy and Loss

```
Accuracy of the model :  0.9239879684170949
F1-score:  0.7683632586437157
Precision Score :  0.7939282428702852
Recall Score :  0.7443933294997125
```

Figure 26: Metrics for Bi-GRU with FastText

	precision	recall	f1-score	support
Toxic	0.86	0.76	0.81	1554
severe_toxic	0.53	0.35	0.42	146
obscene	0.86	0.80	0.83	824
threat	0.49	0.43	0.45	47
insult	0.74	0.78	0.76	768
identity hate	0.45	0.57	0.50	139
micro avg	0.79	0.74	0.77	3478
macro avg	0.65	0.61	0.63	3478
weighted avg	0.80	0.74	0.77	3478
samples avg	0.07	0.07	0.06	3478

Figure 27: Classification report

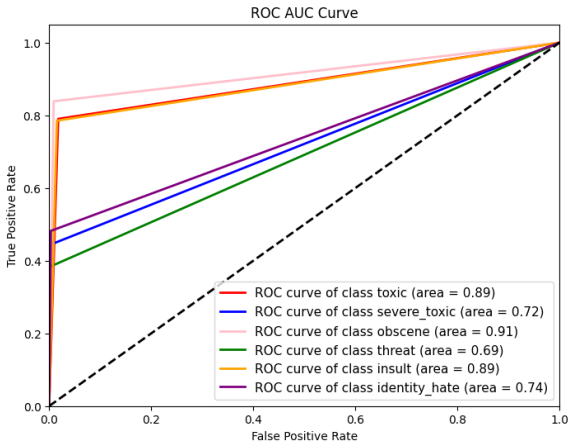

Figure 28: ROC curve

5.4 Ensemble Model

```
Accuracy of the model :  0.9241132974056899
F1-score:  0.7575331772053083
Precision Score :  0.8288349846258968
Recall Score :  0.6975273145485912
```

Figure 29: Metrics for Ensemble method

	id	comment_text	toxic	severe_toxic	obscene	threat	insult	identity_hate
0	00001cee341fdb12	Yo bitch Ja Rule is more succesful then you'll...	0.985916	0.036378	0.970185	0.007426	0.960260	0.178641
1	0000247867823ef7	== From RfC == \n\n The title is fine as it is...	0.009413	0.002286	0.004302	0.001052	0.006008	0.002150
2	00013b17ad220c46	" \n\n == Sources == \n\n * Zawe Ashton on Lap...	0.010316	0.002306	0.004516	0.001059	0.006141	0.002165
3	00017563c3f7919a	If you have a look back at the source, the in...	0.009503	0.002286	0.004315	0.001054	0.006019	0.002149
4	00017695ad8997eb	I don't anonymously edit articles at all.	0.009599	0.002289	0.004351	0.001054	0.006030	0.002152
5	0001ea8717f6de06	Thank you for understanding. I think very high...	0.009582	0.002289	0.004334	0.001062	0.006056	0.002151
6	00024115d4cbde0f	Please do not add nonsense to Wikipedia. Such ...	0.009555	0.002287	0.004319	0.001054	0.006223	0.002151
7	000247e83dcc1211	,Dear god this site is horrible.	0.257307	0.002362	0.005678	0.001077	0.010059	0.002282
8	00025358d4737918	" \n Only a fool can believe in such numbers. ...	0.026626	0.002298	0.004788	0.001062	0.007985	0.002189
9	00026d1092fe71cc	== Double Redirects == \n\n When fixing double...	0.009422	0.002286	0.004298	0.001051	0.005997	0.002148

Figure 30: Predicted output from ensemble model

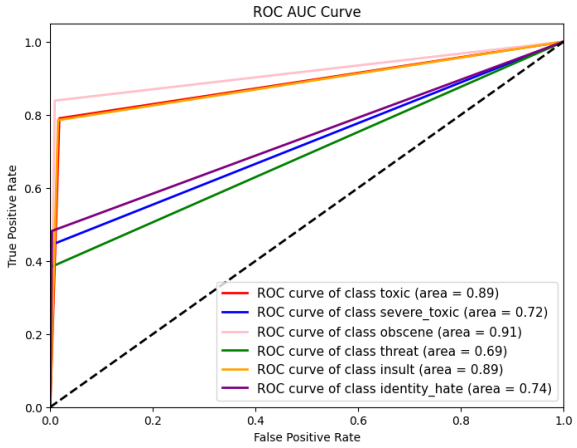

Figure 31: ROC curve

6. DISCUSSION

After reviewing the existing work, we were able to build 4 models the LSTM with GloVe embeddings, the Bi-GRU with GloVe embeddings, Bi-GRU with FastText embeddings and an ensemble method combining both Bi-GRU with GloVe as well as Bi-GRU with FastText embeddings as those were the two models which performed better from all the other models in terms of accuracy and other metrics. And therefore, can better detect different types of toxicity like threats, obscenity, insults, and identity-based hate (RQ1).

Answering our second research questions (RQ2) which was to compare the proposed models using different types of embeddings we found that Bi-GRU performs marginally better than LSTM, thus we applied ensemble method of both the models which are Bi-GRU with GloVe and Bi-GRU with FastText. For the ensemble method, we achieved a predicted accuracy of 92.41%. For the test data comments we finally got the submission file consisting of the probabilities of the comments against the 6 classes of toxicity.

Table I: Comparing all the proposed models

Models	Training Accuracy	Validation Accuracy	Predicted Accuracy	F-1 Score	Precision (TP/TP+FP)	Recall (TP/TP+FN)
LSTM with GloVe	98.93%	97.46%	91.92%	77.08%	82.16%	67.13%
Bi-GRU with GloVe	98.74%	98.48%	92.33%	74.76%	84.22%	67.22%
Bi-GRU with FastText	98.51%	98.75%	92.39%	76.83%	79.39%	74.43%
Ensemble Model	-	-	92.41%	75.75%	82.88%	69.75%

It was identified that due to its inherent subjectivity, identifying internet toxicity has always been difficult. The context, location, socio-political climate, and background of the authors and readers of the posts all play a significant role in determining whether the content can be classified as toxic [39]. There are various biases which may affect the accuracy of the toxicity detection of comments which we will be discussing answering our third research question (RQ3).

Types of biases affecting the accuracy:

- Data bias

Data bias refers to the mistake that arises when components of a dataset are given too much weight or prominence. Such datasets are not an accurate representation of the use case of machine learning

models, leading to imbalanced results, systematic discrimination, and reduced precision. Having millions of data points is necessary. Inaccurate predictions are likely to result from poor or incomplete data, as well as biased data collection and analysis techniques since the quality of the outputs is influenced by the quality of the inputs.

- Systemic bias

Occurs when certain social groups are favored and others are devalued.[61] The reason behind it is institutional and stems from the underrepresentation of disabled people in studies. The biggest drawback of systemic bias is that it is surreptitiously hidden in the world and thus overlooked.

- Selection bias

Selection bias occurs when you have data that are not properly randomized. If your dataset is not properly randomized, it means the sample isn't representative - it doesn't truly reflect the analyzed population. Randomization is the process that balances out the effects of uncontrollable factors - variables in a data set that are not specifically measured and can compromise results.[59]

- Reporting Biases

A reporting bias is the inclusion of only a subset of results in an analysis, which typically only covers a small fraction of evidence.[58] Reporting bias can take many forms. An instance of reporting bias would involve analyzing the data primarily based on studies referenced in citations of other studies (citation bias), disregarding reports not written in the scientist's native language (language bias), or selectively choosing studies with positive findings over those with negative findings (publication bias).

- Database bias

The bias [60] in which the dataset used for training to develop a model may not accurately reflect the diversity of opinions, language, and context in real-world use cases [33].

- Gender bias

Bias in which the toxicity detection models may be biased towards certain genders or language used by certain genders, leading to false positive or negative results [6].

- Cultural bias

Bias due to which the models may not accurately detect toxicity in different languages and cultures, leading to cultural insensitivity and false results [39].

- Historical bias

Bias in which toxicity detection models may be trained on historical data that is no longer reflective of current social norms and expectations, leading to outdated results [33].

- Algorithmic bias

Bias due to which the algorithm used for toxicity detection may have biases based on the design choices and the mathematical model used, leading to biased results.

- Annotator bias

Bias in which human annotators used to label data for training the model may have their own biases and beliefs that affect their labelling decisions (toxic language detection) TLD tasks, leading to biased results [38].

Lastly, in order to justify work on various bias control methods referring to the last research question (RQ4) the work done so far by existing research is mentioned. So far, many debiasing methods have been developed to mitigate biases in learned models, such as data re-balancing (Dixon et al., 2018), residual fitting (He et al., 2019; Clark et al., 2019), adversarial training [54] and data filtering approach. While most of these works are successful on other natural language processing (NLP) tasks, their performance on debasing this are unsatisfactory [55]. One potential explanation is that the evaluation of language toxicity is more subjective and intricate compared to other general natural language processing (NLP) tasks, which often have clearly defined unambiguous labels [54]. As current debiasing techniques reduce the biased behavior of models by correcting the training data or measuring the difficulty of modelling them, which prevents models from capturing spurious data and thus due to the absence of non-linguistic correlation between input texts and labels, the subtle nature of toxicity annotations can render these techniques inadequate for the task of toxic language detection (TDL).

There are many methods proposed to mitigate the biases in NLP tasks other than TLD [55] train a robust classifier in an ensemble with a bias-only model to learn the more generalizable patterns in training dataset, which are difficult to be learned by the naive bias-only model.[56]

To assess the normative biases of algorithmic content moderation systems, the authors of [6] offer some exploratory techniques. A case study using an existing

dataset of comments marked as offensive is used in the paper. The authors used comments labeled by various demographic subsets (men and women) to train classifiers in order to understand how variations in these groups' conceptions of offense may influence the effectiveness of the performing models on various test datasets. They wrapped up by talking about some of the moral decisions algorithmic moderation system developers must make considering the various levels of viewpoint variety that are wanted among discussion participants.

Every year a considered number of novel slang words or phrases emerge or are uncovered from the internet, eventually making their way into the dictionary. These additions include abbreviations as well as trendier version of existing words. And to deal with such changes in the language and to detect toxicity of a text accurately the dataset needs to be updated regularly which is a time taking task. To train the model for accurate toxicity detection the dataset needs to be large enough as well as updated [13]. While detecting toxicity as discussed earlier various biases are also to be dealt with.

There are various transfer learning methods like multi-task learning to deal with the issue of lack of training dataset and biases. Multi-task learning is used to train a model that provides output for multiple tasks based on a single input which is common to all the multiple tasks [49].

To deal with the biases while detecting toxicity of the text the authors of [50] proposed an invariant rationalization (INVRAT) which is a framework consisting of rational generators and predictors which is used to rule out the relation of syntactic patterns to toxicity labels and achieved a lower false rate than already existing debiasing methods. Another proposed method is the empirical analysis of multi-task learning for reducing the identity bias in detection of toxicity by the authors of [36]. They proposed the MTL method with an attention layer that together learns to predict the comment's toxicity as well as the identities present in the comments in order to reduce the identity base bias. The proposed model in [36] outperformed all the other baseline models.

The Jigsaw dataset by Kaggle contains a large number of comments from Wikipedia, and mitigating algorithmic and annotator bias in the dataset is crucial to ensure that machine learning models trained on the dataset do not perpetuate or amplify existing biases. Methods that can be effectively used to mitigate algorithmic and annotator bias in the Jigsaw dataset have been considered. After examining the existing research to mitigate bias, we were able to conclude that for our dataset, the following could be adopted in the future:

- Data augmentation: One of the most effective ways to mitigate bias in the Jigsaw dataset is to augment it with additional data. This can involve adding more comments from diverse perspectives to ensure that the dataset reflects a wide range of viewpoints. This can help reduce algorithmic and annotator bias by ensuring that the dataset is more balanced.
- Adversarial training: Adversarial training is a method used to improve the robustness of machine learning models by adding perturbations to the input data. This method can be used to mitigate algorithmic bias by creating adversarial examples that are designed to fool the model. By training the model to correctly classify these examples, it can be made more robust to bias in the input data.
- Fairness constraints: Fairness constraints can be used to ensure that machine learning models trained on the Jigsaw dataset are fair and unbiased. These constraints can be added to the training process to enforce fairness criteria such as demographic parity, which requires that the model's predictions are consistent across different demographic groups.
- Human-in-the-loop annotation: One of the major sources of bias in the Jigsaw dataset is annotator bias. Human-in-the-loop annotation can help mitigate this bias by involving humans in the annotation process. This can include using multiple annotators to label each comment and then using a consensus-based approach to determine the final label. Additionally, annotators can be trained on how to recognize and mitigate bias in their labelling.
- Bias audit: Bias audit involves analyzing the dataset to identify and quantify any biases that may be present. This can be done by analyzing the distribution of comments and labels across different demographic groups. Once biases have been identified, they can be corrected by removing

biased comments or labels or by adding additional comments to the dataset to address any gaps.

- Regularization: Regularization is a technique used to prevent overfitting by adding a penalty term to the loss function during training. Regularization can be used to mitigate bias in the Jigsaw dataset by adding a penalty term that encourages the model to make unbiased predictions.

In conclusion, mitigating algorithmic and annotator bias in the Jigsaw dataset requires a combination of techniques, including data augmentation, adversarial training, fairness constraints, human-in-the-loop annotation, bias audit, and regularization. By using these methods, it is possible to ensure that machine learning models trained on the Jigsaw dataset are fair, unbiased, and able to generalize to diverse populations.

7. CONCLUSION

Social Media has two sides, one it acts as a powerful channel of communication and a global audience giving opportunity to even the most isolated groups to express opinions and grievances on the other side, minority opinions are oppressed and hate groups formed. Online experiences of users are mediated by algorithms designed to maximize their engagement, which promotes extreme content inadvertently. YouTube's autoplay feature, which causes the player to start playing a related video after one end, can be particularly harmful. According to a Wall Street Journal investigation, the algorithm directs users to videos that support conspiracies.

With the increasing importance of online communication, toxicity detection in text is likely to remain an important area of research in the years to come. In this paper, we have presented a comparative study of deep learning approaches with NLP in the detection and classification of toxic comments into six labels. GloVe embeddings and FastText embeddings were used to train the LSTM and Bi-GRU models to get a better accuracy than the existing baseline models. An ensemble method combining Bi-GRU with GloVe and Bi-GRU with FastText using the averaging method was proposed which gave improved results than individual models. The paper also identifies several research gaps using the existing work such as the types of biases affecting the accuracy of the toxicity detection model and lack of proper training data for such tasks. Methods such as multi-task learning (MTL), MTL with attention layer and invariant rationalization were studied. It was found that such methods can help deal with problems of lack of proper training data and inherent bias and noise in the data that the model will learn from each individual dataset. However, the scope of improvement of the model quality due to existing loopholes clearly stated in this section shall be compensated by our proposed future work.

8. FUTURE WORK

Toxicity detection is an area of research which is becoming highly essential and is likely to see continued advancements in technology. The implementation carried out in this paper so far proposed the models for toxic text detection. The dataset is imbalanced as there are more positive comments than the negative ones and thus a balanced dataset with each class having equal number of comments can be used to further test the models. We would also experiment with the discussed methodologies for RQ3 and RQ4 and try implementing them in near future to deal with inaccurate and inappropriate data to improve the performance of the toxicity detection models to improve the F1-score and provide accurate results.

REFERENCES

[1] Toxic comment classification challenge: https://www.kaggle.com/c/jigsaw-toxic-comment-classification-challenge

[2] Garlapati, A., Malisetty, N., & Narayanan, G. (2022, March). Classification of Toxicity in Comments using NLP and LSTM. In *2022 8th International Conference on Advanced Computing and Communication Systems (ICACCS)* (Vol. 1, pp. 16-21). IEEE.

[3] Menini, S., Aprosio, A. P., & Tonelli, S. (2021). Abuse is contextual, what about nlp? the role of context in abusive language annotation and detection. *arXiv preprint arXiv:2103.14916*.

[4] Maslej-Krešňáková, V., Sarnovský, M., Butka, P., & Machová, K. (2020). Comparison of deep learning models and various text pre-processing techniques for the toxic comments classification. *Applied Sciences*, *10*(23), 8631.

[5] Jain, S., Kaushik, G., Prabhu, P., & Godbole, A. (2021, July). Detox: NLP Based Classification And Euphemistic Text Substitution For Toxic Comments. In *2021 12th International Conference on Computing Communication and Networking Technologies (ICCCNT)* (pp. 1-5). IEEE.

[6] Binns, R., Veale, M., Van Kleek, M., & Shadbolt, N. (2017). Like trainer, like bot? Inheritance of bias in algorithmic content moderation. In *Social Informatics: 9th International Conference, SocInfo 2017, Oxford, UK, September 13-15, 2017, Proceedings, Part II 9* (pp. 405-415). Springer International Publishing.

[7] Malik, P., Aggrawal, A., & Vishwakarma, D. K. (2021, April). Toxic speech detection using traditional machine learning models and bert and fasttext embedding with deep neural networks. In *2021 5th International Conference on Computing Methodologies and Communication (ICCMC)* (pp. 1254-1259). IEEE.

[8] Aquino, M., Ortiz, Y., Rashid, A., Tumlin, A. M., Artan, N. S., Dong, Z., & Gu, H. (2021, October). Toxic Comment Detection: Analyzing the Combination of Text and Emojis. In *2021 IEEE 18th International Conference on Mobile Ad Hoc and Smart Systems (MASS)* (pp. 661-662). IEEE.

[9] Alshamrani, S., Abuhamad, M., Abusnaina, A., & Mohaisen, D. (2020). Investigating Online Toxicity in Users Interactions with the Mainstream Media Channels on YouTube. In *CIKM (Workshops)*.

[10] Lashkarashvili, N., & Tsintsadze, M. (2022). Toxicity detection in online Georgian discussions. *International Journal of Information Management Data Insights*, *2*(1), 100062.

[11] Taleb, M., Hamza, A., Zouitni, M., Burmani, N., Lafkiar, S., & En-Nahnahi, N. (2022, May). Detection of toxicity in social media based on Natural Language Processing methods. In *2022 International Conference on Intelligent Systems and Computer Vision (ISCV)* (pp. 1-7). IEEE.

[12] Zhong, H., Li, H., Squicciarini, A. C., Rajtmajer, S. M., Griffin, C., Miller, D. J., & Caragea, C. (2016, July). Content-Driven Detection of Cyberbullying on the Instagram Social Network. In *IJCAI* (Vol. 16, pp. 3952-3958).

[13] Zhao, Z., Zhang, Z., & Hopfgartner, F. (2019). Detecting toxic content online and the effect of training data on classification performance. *EasyChair*.

[14] Dubey, K., Nair, R., Khan, M. U., & Shaikh, S. (2020, December). Toxic comment detection using lstm. In *2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)* (pp. 1-8). IEEE.

[15] Cross, C., Munukutla, S., & Yong, T. S. Bypassing Censorship.

[16] Almerekhi, H., Kwak, H., Jansen, B. J., & Salminen, J. (2019, September). Detecting toxicity triggers in online discussions. In *Proceedings of the 30th ACM conference on hypertext and social media* (pp. 291-292).

[17] A. G. D'Sa, I. Illina and D. Fohr, "BERT and fastText Embeddings for Automatic Detection of Toxic Speech," 2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA), Tunis, Tunisia, 2020, pp. 1-5, doi: 10.1109/OCTA49274.2020.9151853.

[18] Zaheri, S., Leath, J., & Stroud, D. (2020). Toxic comment classification. *SMU Data Science Review*, *3*(1), 13.

[19] Xia, M., Field, A., & Tsvetkov, Y. (2020). Demoting racial bias in hate speech detection. *arXiv preprint arXiv:2005.12246*.

[20] Bhat, M. M., Hosseini, S., Hassan, A., Bennett, P., & Li, W. (2021, November). Say 'YES'to positivity: Detecting toxic language in workplace communications. In *Findings of the Association for Computational Linguistics: EMNLP 2021* (pp. 2017-2029).

[21] Venkit, P. N., & Wilson, S. (2021). Identification of bias against people with disabilities in sentiment analysis and toxicity detection models. *arXiv preprint arXiv:2111.13259*.

[22] Jeoung, S., & Diesner, J. (2022). What Changed? Investigating Debiasing Methods using Causal Mediation Analysis. *arXiv preprint arXiv:2206.00701*.

[23] Kohli, M., Kuehler, E., & Palowitch, J. Paying attention to toxic comments online. *Web: https://web. stanford. edu/class/archive/cs/cs224n/cs224n*, *1184*.

[24] Nguyen, L. T., Van Nguyen, K., & Nguyen, N. L. T. (2021). Constructive and toxic speech detection for open-domain social media comments in vietnamese. In *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices: 34th International Conference on Industrial, Engineering and Other Applications of Applied Intelligent Systems, IEA/AIE 2021, Kuala Lumpur, Malaysia, July 26–29, 2021, Proceedings, Part I 34* (pp. 572-583). Springer International Publishing.

[25] Schmidt, A., & Wiegand, M. (2017, April). A survey on hate speech detection using natural language processing. In *Proceedings of the fifth international workshop on natural language processing for social media* (pp. 1-10).

[26] Pavlopoulos, J., Malakasiotis, P., & Androutsopoulos, I. (2017). Deep learning for user comment moderation. *arXiv preprint arXiv:1705.09993*.

[27] Fortuna, P., & Nunes, S. (2018). A survey on automatic detection of hate speech in text. *ACM Computing Surveys (CSUR)*, *51*(4), 1-30.

[28] Arango, A., Pérez, J., & Poblete, B. (2019, July). Hate speech detection is not as easy as you may think: A closer look at model validation. In *Proceedings of the 42nd international acm sigir conference on research and development in information retrieval* (pp. 45-54).

[29] Arango, A., Pérez, J., & Poblete, B. (2022). Hate speech detection is not as easy as you may think: A closer look at model validation (extended version). *Information Systems*, *105*, 101584.

[30] Davidson, T., Bhattacharya, D., & Weber, I. (2019). Racial bias in hate speech and abusive language detection datasets. *arXiv preprint arXiv:1905.12516*.

[31] Kiritchenko, S., & Mohammad, S. M. (2018). Examining gender and race bias in two hundred sentiment analysis systems. *arXiv preprint arXiv:1805.04508*.

[32] Noever, D. (2018). Machine learning suites for online toxicity detection. *arXiv preprint arXiv:1810.01869*.

[33] Pavlopoulos, J., Sorensen, J., Dixon, L., Thain, N., & Androutsopoulos, I. (2020). Toxicity detection: Does context really matter?. *arXiv preprint arXiv:2006.00998*.

[34] Park, J. H., Shin, J., & Fung, P. (2018). Reducing gender bias in abusive language detection. *arXiv preprint arXiv:1808.07231*.

[35] Sap, M., Card, D., Gabriel, S., Choi, Y., & Smith, N. A. (2019, July). The risk of racial bias in hate speech detection. In *Proceedings of the 57th annual meeting of the association for computational linguistics* (pp. 1668-1678).

[36] Vaidya, A., Mai, F., & Ning, Y. (2020, May). Empirical analysis of multi-task learning for reducing identity bias in toxic comment detection. In *Proceedings of the International AAAI Conference on Web and Social Media* (Vol. 14, pp. 683-693).

[37] Aroyo, L., Dixon, L., Thain, N., Redfield, O., & Rosen, R. (2019, May). Crowdsourcing subjective tasks: the case study of understanding toxicity in online discussions. In *Companion proceedings of the 2019 world wide web conference* (pp. 1100-1105).

[38] Excell, E., & Moubayed, N. A. (2021). Towards equal gender representation in the annotations of toxic language detection. *arXiv preprint arXiv:2106.02183*.

[39] Garg, T., Masud, S., Suresh, T., & Chakraborty, T. (2022). Handling bias in toxic speech detection: A survey. *ACM Computing Surveys*.

[40] Tatman, R. (2017, April). Gender and dialect bias in YouTube's automatic captions. In *Proceedings of the first ACL workshop on ethics in natural language processing* (pp. 53-59).

[41] Cheng, L., Mosallanezhad, A., Silva, Y. N., Hall, D. L., & Liu, H. (2022, July). Bias mitigation for toxicity detection via sequential decisions. In *Proceedings of the 45th International ACM SIGIR Conference on Research and Development in Information Retrieval* (pp. 1750-1760).

[42] Ross, B., Rist, M., Carbonell, G., Cabrera, B., Kurowsky, N., & Wojatzki, M. (2017). Measuring the reliability of hate speech annotations: The case of the european refugee crisis. *arXiv preprint arXiv:1701.08118*.

[43] Mathew, B., Dutt, R., Goyal, P., & Mukherjee, A. (2019, June). Spread of hate speech in online social media. In *Proceedings of the 10th ACM conference on web science* (pp. 173-182).

[44] Graumas, L., David, R., & Caselli, T. (2019, September). Twitter-based polarised embeddings

for abusive language detection. In *2019 8th International Conference on Affective Computing and Intelligent Interaction Workshops and Demos (ACIIW)* (pp. 1-7). IEEE.

[45] https://www.bu.edu/bhr/2022/06/14/social-media-and-wellbeing-cause-for-concern/

[46] Gupta, S., Lee, S., De-Arteaga, M., & Lease, M. (2023). Same Same, But Different: Conditional Multi-Task Learning for Demographic-Specific Toxicity Detection. *arXiv preprint arXiv:2302.07372*.

[47] Sheth, A., Shalin, V. L., & Kursuncu, U. (2022). Defining and detecting toxicity on social media: context and knowledge are key. *Neurocomputing*, *490*, 312-318.

[48] H. Fan *et al.*, "Social Media Toxicity Classification Using Deep Learning: Real-World Application UK Brexit," *Electronics*, vol. 10, no. 11, p. 1332, Jun. 2021, doi: 10.3390/electronics10111332

[49] Nelatoori, K. B., & Kommanti, H. B. (2022). Multi-task learning for toxic comment classification and rationale extraction. *Journal of Intelligent Information Systems*, 1-25.

[50] Chuang, Y. S., Gao, M., Luo, H., Glass, J., Lee, H. Y., Chen, Y. N., & Li, S. W. (2021). Mitigating biases in toxic language detection through invariant rationalization. *arXiv preprint arXiv:2106.07240*.

[51] Lucas Dixon, John Li, Jeffrey Sorensen, Nithum Thain,and Lucy Vasserman. 2018. Measuring and mitigating unintended bias in text classification. In Proceedings of the 2018 AAAI/ACM Conference on AI Ethics, and Society, pages 67–73.

[52] Christopher Clark, Mark Yatskar, and Luke Zettle-moyer. 2019. Don't take the easy way out: Ensemble based methods for avoiding known dataset biases. In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4060–4073.

[53] Mengzhou Xia, Anjalie Field, and Yulia Tsvetkov.2020. *Demoting racial bias in hate speech detection.* In Proceedings of the Eighth International Workshop on Natural Language Processing for Social Media, pages 7–14.

[54] Xuhui Zhou, Maarten Sap, Swabha Swayamdipta,Yejin Choi, and Noah A Smith. 2021. *Challenges in automated debiasing for toxic language detection.*In Proceedings of the 16th Conference of the European Chapter of the Association for Computational Linguistics: Main Volume, pages 3143–3155.

[55] Christopher Clark, Mark Yatskar, and Luke Zettle-moyer. 2019. *Don't take the easy way out: Ensemble based methods for avoiding known dataset biases.* In Proceedings of the 2019 Conference on Empirical Methods in Natural Language Processing and the 9th International Joint Conference on Natural Language Processing (EMNLP-IJCNLP), pages 4060–4073.

[56] Antigoni Founta, Constantinos Djouvas, Despoina Chatzakou, Ilias Leontiadis, Jeremy Blackburn, Gianluca Stringhini, Athena Vakali, Michael Siriv ianos, and Nicolas Kourtellis. 2018. *Large scale crowdsourcing and characterization of twitter abusive behavior.*

[57] Ashwin Geet d'Sa, Irina Illina, and Dominique Fohr. 2020. Bert and fasttext embeddings for automatic detection of toxic speech. In 2020 International Multi-Conference on:"Organization of Knowledge and Advanced Technologies"(OCTA), pages 1–5.

[58] Ioannidis, J. P., Munafo, M. R., Fusar-Poli, P., Nosek, B. A., & David, S. P. (2014). Publication and other reporting biases in cognitive sciences: detection, prevalence, and prevention. *Trends in cognitive sciences*, *18*(5), 235-241.

[59] Mandel, I., Farr, W. M., & Gair, J. R. (2019). Extracting distribution parameters from multiple uncertain observations with selection biases. *Monthly Notices of the Royal Astronomical Society*, *486*(1), 1086-1093.

[60] Borrajo, L., & Cao, R. (2022). Bias Testing for Big Data Analytics. *Unpublished manuscript available at http://dm. udc. es/preprint/Bias_Testing_for_Big_Data_Analytics. Pdf*.

[61] Wamburu, J., Tadesse, G. A., Cintas, C., Oshingbesan, A., Akumu, T., & Speakman, S. (2022, December). Systematic Discovery of Bias in Data. In *2022 IEEE International Conference on Big Data (Big Data)* (pp. 4719-4725). IEEE.

[62] Dong, X., Yu, Z., Cao, W., Shi, Y., & Ma, Q. (2020). A survey on ensemble learning. *Frontiers of Computer Science*, *14*, 241-258.

[63] Zhou, Z. H. (2012). *Ensemble methods: foundations and algorithms*. CRC press.

[64] Zimmerman, S., Kruschwitz, U., & Fox, C. (2018, May). Improving hate speech detection with deep learning ensembles. In *Proceedings of the eleventh international conference on language resources and evaluation (LREC 2018)*.

[65] Joulin, A., Grave, E., Bojanowski, P., Mikolov, T., 2016. Bag of tricks for efficient text classification. arXiv preprint arXiv:1607.01759 .

[66] Pennington, J., Socher, R., Manning, C.D., 2014. GloVe: Global vectors for word representation, in: Proceedings of the 2014 conference on empirical methods in natural language processing (EMNLP), pp. 1532–1543.

[67] https://solutionsreview.com/data-management/80-percent-of-your-data-will-be-unstructured-in-five-years/