Am I Being Bullied on Social Media? An Ensemble Approach to Categorize Cyberbullying

Tasnim Ahmed*, Mohsinul Kabir†, Shahriar Ivan‡, Hasan Mahmud§ and Dr. Kamrul Hasan¶
*†‡§¶ *Department of Computer Science and Engineering, Islamic University of Technology, Dhaka, Bangladesh*
{**tasnimahmed, †mohsinulkabir, ‡shahriarivan, §hasan, ¶hasank*} *@iut-dhaka.edu*

2021 IEEE International Conference on Big Data (Big Data) | 978-1-6654-3902-2/21/$31.00 ©2021 IEEE | DOI: 10.1109/BigData52589.2021.9671594

***Abstract*—People can easily reveal their aggressive remarks on social media platforms using the anonymity it provides. During the COVID-19 pandemic, the usage of social media has been increased several times according to surveys and people are vulnerable to cyber attacks now more than ever. Prevention of cyberbullying needs careful monitoring and identification. Most of the existing works on cyberbullying detection employed traditional machine learning classifiers with handcrafted features, and deep learning-based models have made their way in this domain very recently. Categorizing cyberbullying based on traits is a complex task and needs contextual consideration. In this work, we have proposed a new approach to detect cyberbullying on social media platforms using a neural ensemble method of transformer-based architectures with attention mechanism. Our proposed architecture is trained on one balanced and one imbalanced dataset and outperforms the given ML and DNN baselines by a significant margin in both cases. We achieved an average F1-score of 95.59% for five classes and 90.65% for six classes on the Fine-Grained Cyberbullying Dataset (FGCD), and 87.28% on Twitter parsed dataset. Our in-depth results provide great insights into the effectiveness of transformer-based models in cyberbullying detection and paves the way for future researches to combat this serious online issue. We have released our models and code.**[1]

***Index Terms*—Cyberbullying Detection, Social Media Data Mining, Natural Language Processing, Transformers, Ensemble Learning**

1. Introduction

During the COVID-19 pandemic, internet usage has increased about 50%-70% and 50% of that increased time was spent on different social media in 2020 according to a survey[2]. This increased time spent on social media has made people extremely vulnerable to hate speech and cyber-attacks. Online platforms allow users anonymity and people dispose of their racist, sexist, and homophobic remarks towards different groups taking advantage of it. The victims of these attacks often suffer from anxiety, misery and attempt suicide in more tragic cases [1].

To prevent cyberbullying, detecting which specific traits of the victims and which minority groups the cyberbully is attacking is important. Different sociolinguistic researches [2], [3] have studied different types of cyberbullying and compared them with offline bullying. In computer science, the majority of the previous researches [4], [5], [6] mainly focused on only detecting cyberbullying, which can be briefly denoted as a binary classification problem. Recent researches [7], [8] in this domain have explored classifying different types of cyberbullying leveraging modern machine learning classifiers. Classifying the type of cyberbullying is a multi-class classification problem that has a number of challenges- (1) understanding the context of a conversation is important to classify cyberbullying because a seemingly positive verse might be sarcastic and may contain subtle negativity thus making it really difficult for the classifier to detect it as cyberbullying, (2) Annotation or labeling the dataset is tough and needs expert intervention in annotating the correct cyberbullying type. Wang et al. [9] proposed the work of improving the classification of traits of victims targeted by the cyberbully such as age, gender, religion, ethnicity, or other. To our knowledge, this was the first work that investigated victim's trait-based cyberbullying detection. We observed some limitations in their research as they did not consider 'Not Cyberbullying (Not CB)' as a separate class while training and evaluating their classification model which may misclassify a positive comment as a type of cyberbullying. In this work, we seek to improve this trait-based investigation by mitigating their limitations and propose a better classification architecture and analyze the common errors in this domain, which can be pioneering for future researchers. We chose to work with a set of transformer-based classifiers which uses the attention mechanism to better extract the context of sentences in classification tasks [10].

We validated our results by applying our classifiers on a different dataset with different class labels. Finally, by evaluating the metrics, we propose an ensemble of classifiers that outperforms all baseline results of the two datasets. Therefore, our contributions are as follows-

1) To the best of our knowledge, we are the first

1. Our models and code are available at: github.com/tasnim7ahmed/Cyberbullying-Detection-with-Transformers

2. https://www.forbes.com/sites/markbeech/2020/03/25/covid-19-pushes-up-internet-use-70-streaming-more-than-12-first-figures-reveal/

to apply and compare a range of transformers to classify cyberbullying from a fine-grained balanced dataset of around 48,000 tweets. In earlier works, we had already seen the use of single transformer-based models like BERT, or ensemble of traditional models such as XGBoost, SVM, Regression etc. for detection of cyberbullying from social media comments. The novelty factor of our work was to apply an ensemble of transformer-based models in this domain, where each transformer, namely BERT [11], DistilBERT [12], RoBERTa [13], and XLNet [14] was chosen to tackle different challenges.

2) The same set of transformers were applied to another Twitter dataset with a heavy imbalance to validate our result.
3) The individual classifier's performance was evaluated based on several evaluation metrics. Based on this, we built two different ensemble approaches to improve the overall accuracy of classification. Finally one specific ensemble method was proposed that outperforms all the individual classifiers as well as the other ensemble method, capable of producing excellent results on both balanced and imbalanced datasets.
4) We conducted an in-depth error and performance analysis on the result of our proposed architecture. Our analysis generalizes the common errors in current cyberbullying researches thus can be utilized in future researches to tackle the current challenges.

The remainder of this paper is structured as follows: the next section describes the related and prominent literature in this domain. The *Dataset* section describes the two text corpora used in our study followed by our proposed methodology. In the *Results and Discussion* section we illustrate and analyze our experimental results. Finally, the *Conclusion* section concludes this paper by summarizing our work.

2. Related Works

Cyberbullying detection, as defined in the literature, involves identifying aggressive, inhumane, and vicious comments towards an individual or a group of people on any digital platform. Identifying commentators, supporters, and victims of cyberbullying are also topics of research in this domain. Previous literature [15], [16], [17] have tried to classify hate speech and aggressive comments in different social media platforms like Twitter, Reddit, YouTube, etc. leveraging different feature extraction techniques as bag of words, word-level and character-level n-grams, etc. For the classification task, different algorithms as Support Vector Machines, Naive Bayes, and Logistic Regression, etc. have been applied [18], [19]. These works followed a manual feature engineering approach, while some recent works [20], [21], [22] used (deep) neural network-based architecture where the network is supposed to automatically learn abstract features from the input corpus. Neural networks based approaches have been appeared to be more effective in classifying hate speech [23], while feature-based approaches carry some sort of explainability.

Though most of the works related to cyberbullying are traditional binary classification tasks, some works considered different types of cyberbullying such as sexism [24], racism [25], the severity of toxicity [26] etc. The great majority of related research works considered this type of problem where each sample is labeled with exactly one class and are usually framed as 'multiclass classification problem'. We, therefore, choose two multi-class cyberbullying datasets and investigate how our proposed architecture works on these datasets with varying numbers of classes and samples.

Sarracén et al. [27] proposed an attention-based LSTM network to identify hate speech in Twitter and Facebook where the attention layer is applied on a bidirectional LSTM to generate a context vector for each word embedding (the representation of words for text analysis). Wang et al. [9] proposed a graph convolutional neural network model with eight different tweet embedding methods and six different classification models to compare the performance of cyberbullying detection. Burnap et al. [19] studied the advantages of ensemble learning for detecting cyber hate speech in Twitter comments. They combined results from three different classifiers, namely Bayesian Logistic Regression(BLR), Random Forest Decision Tree (RFDT), and Support Vector Machine (SVM). Later, Zimmerman et al. [28] studied the ensembling of different neural network models with different hyperparameters. To the best of our knowledge, the approach of ensemble learning with different models following the attention-based mechanism has not been investigated so far.

3. Datasets

We used two publicly available datasets for our experiments: Fine-Grained Cyberbullying Dataset [29] and Twitter Parsed Cyberbullying Dataset [30]. Both of these datasets contain multiple classes of labeled cyberbullying samples and are publicly available.

3.1. Fine-Grained Cyberbullying Dataset (FGCD)

FGCD is a balanced dataset of about 48,000 Twitter comments distributed into six cyberbullying classes, 'Age', 'Ethnicity', 'Gender', 'Religion', 'Other', and 'NotCb (not cyberbullying)'. Wang et. al [9] proposed this dataset by combining six different datasets shown in Table 1a and by further classifying the tweets of these six datasets by hand and grouping the same classes together. To remove the class imbalance, they used Dynamic Query Expansion (DQE) and increased the number of samples of each class in a semi-supervised manner. As the focus of the authors was to construct a fine-grained dataset containing cyberbullying tweets, they randomly sampled 8000 tweets of each class

mentioned above and formed a balanced dataset of size 48000.

3.2. Twitter Parsed Cyberbullying Dataset

This dataset was provided by Waseem et al. [30] and contains 16848 tweet comments distributed into three classes, 'Sexism', 'Racism' and 'Not Cyberbullying' as shown in Table 1b. As this is an imbalanced dataset where two-thirds of the sample belongs to the 'not cyberbullying' class, we chose this dataset to assess how our proposed architecture performs on an imbalanced dataset with different class labels.

The dataset summary and the class distribution of these two datasets are presented in Table 1 and in Figure 1 respectively.

4. Proposed Methodology

Our proposed architecture accepts a comment as an input and returns the category of cyberbullying for that comment as an output. Figure 2 depicts a schematic representation of our proposed architecture.

In this section, we describe our proposed methodology in detail, which includes preprocessing, tokenization, feature extraction, classification, ensemble methods, and evaluation metrics.

4.1. Preprocessing

We used transformer-based pretrained models to efficiently extract features from our input, which were the user comments posted on Twitter. We experimented with four different pretrained models- BERT [11], DistilBERT [12], RoBERTa [13], and XLNet [14]. Since these models are pretrained on large corpora of texts, they perform well on unprocessed or uncleaned text inputs [34]. That is why our preprocessing steps are very minimal and limited to removing the excessive use of spaces or tabs from individual comments.

4.2. Tokenization

Tokenizer prepares the inputs for our pretrained feature extractors. A tokenizer takes a string as an input. Then, the input string is converted into sub-word token strings. Finally, token strings are converted to IDs or integers. We have used pretrained tokenizers corresponding to the feature extractors. These tokenizers not only transform texts to token IDs, but they also add new tokens to the vocabulary in a fashion that is unaffected by the underlying architecture. The tokenization process of a single comment is depicted in Figure 3.

The sample comment used for demonstration of the tokenization process is categorized as a cyberbullying comment which attacks the 'Ethnicity' of the victim. After converting a comment to token strings, some special tokens are generated to indicate the beginning of a sentence, mask, etc. In Figure. 3, [CLS] and [SEP] are special tokens. [CLS] is a special token referring to a classification task and [SEP] token helps the model to understand the end of one input and the beginning of another input in the same sequence. Transformer-based pretrained models require the input samples to be of the same size. Despite the variable length of the comments, our generated vectors of token IDs from these comments must be equal in length, and we limit the maximum token number to 128. If the number of tokens in a comment is less than 128, additional padding will be added to make the token count exactly 128. Token string for padding is [PAD], and the token ID for padding is 0. If the number of tokens in a comment is more than 128, then additional tokens are removed. However, we have seen that the maximum number of tokens in a single comment from our datasets almost never exceeds our maximum token length. It should be further clarified that a single token usually corresponds to a single word, not a single character. Therefore, allowing a maximum of 128 tokens will almost always accommodate the maximum character limit in a comment. For instance, the maximum character limit for comments in Twitter is 280, and it is highly unlikely that more than 128 tokens can be formed from this 280 characters, unless many of these tokens are single-character tokens.

4.3. Transformer Based Pretrained Feature Extractor

In language-related tasks, a deep bidirectional model outperforms left-to-right or right-to-left models. We used bidirectional encoder-based feature extractors because standard language models only allow unidirectional training. Pretraining these models on large-scale unlabeled text corpora and fine-tuning the models on downstream tasks have been extremely successful, though the ultimate objective of pretraining differs between the models, with BERT, DistilBERT, and RoBERTa relying on Auto Encoding (AE) language modeling and XLNet on Auto-Regressive (AR) language modeling.

4.3.1. BERT. The capability of pretrained representations was heavily restricted by the previous language representation models such as ELMo [35] which employs task-specific architecture and the OpenAI GPT [36], which is built on a unidirectional left-to-right architecture where each token is capable of handling the previous tokens only in the transformer's self-attention layers. Through the proposal of BERT: Bidirectional Encoder Representations from Transformers, Devlin et. al [11] mitigated this issue by incorporating context from both directions while employing fine-tuning based techniques to token-level tasks. While pretraining, BERT alleviates the unidirectionality constraint by using Masked Language Model (MLM) which randomly masks some of the input tokens and the goal of the pretraining task is to predict the original token ID based on

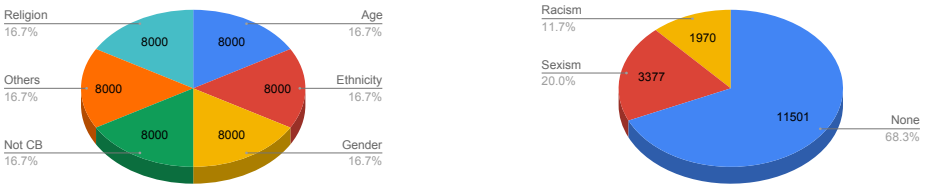
Table 1: Dataset Summary

(a) Fine Grained Cyberbullying Dataset (FGCD)

Author	Total	CB	Not CB	Age	Ethnicity	Gender	Religion	Other
Agrawal [7]	16050	5963	10087	0	13	2841	1922	187
Brettschneider [31]	4475	183	4292	49	29	34	0	71
Chatzakou [32]	1500	1278	222	17	100	115	10	1036
Davidson [15]	205	181	24	5	27	121	1	27
Waseem [33], [30]	12899	8900	3999	0	86	3339	0	5475
WISC [1]	4095	1078	3024	94	17	39	0	921
Collective Before DQE	39224	17583	21648	165	272	6489	1933	7717
Collective After DQE	69767	50468	19299	10010	12730	10277	9367	8084
Included in FGCD (Randomly Sampled)	48000	8000	8000	8000	8000	8000	8000	8000

(b) Twitter Parsed Dataset

Classes	Samples
None	11501
Sexism	3377
Racism	1970



(a) Fine Grained Cyberbullying Dataset (b) Twitter Parsed Dataset

Figure 1: Dataset Class Distribution

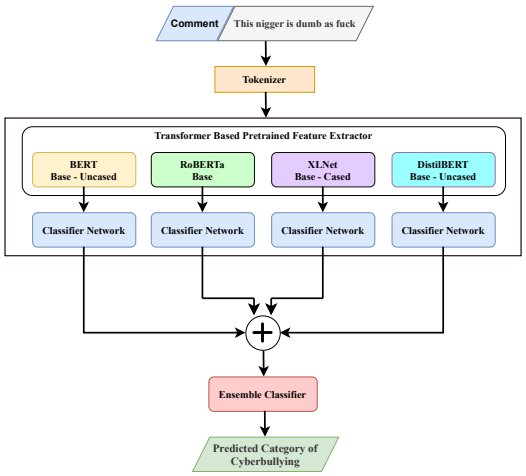


Figure 2: Overview of the Proposed Cyberbullying Classification Network from Comments

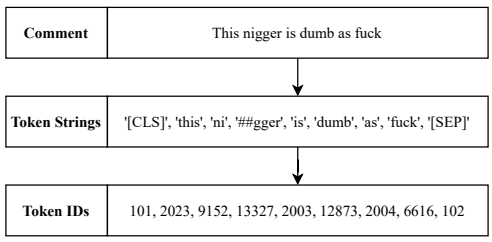


Figure 3: Tokenization process of a comment

the context of the input. Like any other transformer-based pretrained feature extractor, BERT framework consolidates two steps - pretraining and fine-tuning. The reason behind BERT-based architectures not being task-specific is that, during the pretraining stage, BERT is pretrained with different unlabeled data over various pretraining tasks. We fine-tuned the pretrained BERT model with domain-specific data to effectively predict cyberbullying. BERT uses multi-layer bidirectional Transformer encoder described by Vaswani et. al [10] and WordPiece embeddings [37] with 30K token vocabularies. In our proposed architecture, we have used $BERT_{BASE}(L = 12, H = 768, A = 12)$ where, L refers to the Transformer block count, H refers to the hidden size, and A refers to the number of self-attention heads.

4.3.2. RoBERTa. Liu et. al [13] proposed an improved configuration for BERT training referred as **R**obustly **o**ptimized **BERT** **a**pproach or RoBERTa which shows that BERT is capable of outperforming recent models like XLNet [14] without changing any internal model architecture. RoBERTa's pretraining techniques improvement over BERT are as follows-

- Unlike BERT's static masking approach, RoBERTa is pretrained with dynamic masking, where the masking pattern i.e., the random tokens that are to be masked, are generated every time they are fed to the model i.e., in every epoch.
- During pretraining, BERT model encountered input sequences from two separate documents (with probability, $p = 0.5$) and one of the subtasks of the model was to determine whether the input sequences are originated from the same sequence or not via Next

Sequence Prediction (NSP) loss. Without having any NSP loss, RoBERTa pretrained the model with complete sentences from one or more sources.

- A larger mini-batch size enhances downstream task performance, as well as optimization speed, according to recent research findings [38]. $BERT_{BASE}$ was trained using a batch size of only 256 for a total of 1 million steps, whereas RoBERTa was pretrained using a batch size of 2 thousand for a total of 125 thousand steps.
- After preprocessing using heuristic tokenization rules, BERT was pretrained with a character level Byte-Pair Encoding (BPE) vocabulary of size 30K. Unlike BERT, RoBERTa did not use preprocessing and used a much larger 50K byte-level BPE vocabulary. This resulted in a large increase in RoBERTa's parameter count while also enhancing downstream work performance.

Along with these pretraining optimizations, RoBERTa was trained with 160 GB of text data which is larger than the other BERT variants we experimented with an almost 10 times larger than the pretraining data of $BERT_{BASE}$.

4.3.3. XLNet. AE-based models like BERT mask portions of the input tokens with the help of synthetic symbols such as [MASK] during pretraining, but these symbols are not present in real data during the fine-tuning phase, resulting in a pretrain-finetune disparity. On the other hand, Traditional AR language modeling uses an autoregressive model to estimate the probability distribution of a text corpus by factorizing the probabilities into either a forward or a backward product, making it futile at modeling deep bidirectional contexts. Yang et. al [14] addresses these limitations in XLNet by using the best features offered by both language modeling. Unlike the traditional AR approaches, XLNet tries to obtain the best log-likelihood of an input sequence with respect to all potential permutations of the factorization order which captures the bidirectional context. XLNet, as a generalized AR pretraining method with a permutation language modeling objective, assumes that predicted input tokens are dependent on each other and does not mask them, thereby eliminating pretrain-finetune disparity. XLNet has the same architecture hyperparameters as BERT, but it incorporates the state-of-the-art AR language model Transformer-XL [39] instead of employing the conventional transformer [10] architecture. XLNet is pretrained using nearly 10 times more data and batch size is eight times larger than the original BERT implementation [13].

4.3.4. DistilBERT. Research work on pretrained models like MegatronLM [40] indicates that larger models such as models with multiple billions, even a trillion parameters, typically result in superior performance on downstream tasks. However, the overall improvement in performance comes at the expense of increased computational power and memory requirements for both training and inference, making them unsuitable for running on the edge, such as smartphones. Sanh et. al [12] addressed this issue by proposing DistilBERT which has an architecture identical to BERT and is pretrained on the same corpus: a combination of English Wikipedia and Toronto Book Corpus [41]. DistilBERT decreases the number of layers by a factor of two by eliminating token-type embeddings and the pooler from the BERT implementation, since hidden size dimensions have less of an impact on computing performance than the number of levels. DistilBERT is pretrained through knowledge distillation via the supervision of a larger model incorporating triple loss functions (Distillation Loss - L_{ce}, Masked Language Modelling Loss - L_{mlm}, and Cosine Embedding Loss - L_{cos}). DistilBERT retains 97% of BERT performance on downstream tasks with 40% fewer parameters, and the fundamental cause behind this is a compression technique known as knowledge distillation, which helps a compact model to reproduce the behavior of larger models along with the components of triple loss. In our cyberbullying detection task, DistilBERT required 50% less training time and 42% less inference time compared to BERT while achieving comparable performance as shown in Section 5.

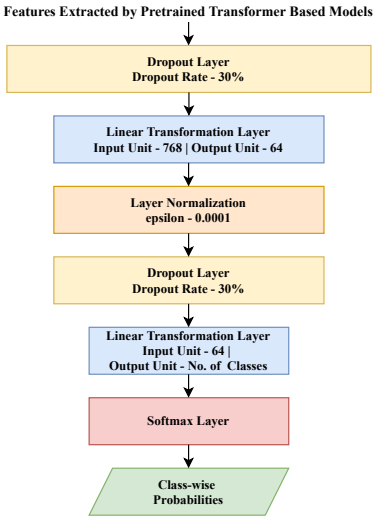


Figure 4: Classifier Network

4.4. Classifier Network

As mentioned in subsection 4.1, the transformer-based models we used in our proposed architecture, were pretrained on a very large corpus of text, which makes them very efficient feature extractors on any domain. To fine-tune the features extracted by the pretrained models, instead of feeding the features directly to a linear transformation layer having output units corresponding to the number of classes, we added an additional classifier network. In our classifier network, we used a combination of Dropout, Normalization, Linear Transformation, and Softmax layers as shown in Figure 4.

Unlike the layers present in the pretrained models, the layers of our classifier network were trained from scratch

to extract the features which were only relevant to our task, classifying the category of cyberbullying. In our classifier, we used two linear transformation layers. Let x be the input of a linear transformation layer and y is the output of the layer. Then, y will be calculated according to the Formula 1 where A represents the weight vector and b is the bias.

$$y = xA^T + b \tag{1}$$

Between the two linear transformation layers, we used a normalization [42] and a dropout layer. Like batch normalization [43] layer, the normalization layer reduces the training time in a feed-forward network significantly by normalizing the activities of the neurons. The batch normalization layer calculates the scalar scale and bias for each mini-batch. However, the normalization layer applies scalar scale and bias to single samples during both the training and testing phase. For an input x and output denoted as y, normalization is conducted based on the Formula 2 where γ and β are learnable affine transform parameters.

$$y = (\frac{x - E[x]}{\sqrt{Var[x] + \epsilon}})\gamma + \beta \tag{2}$$

The dropout layer randomly sets 30% of the output units from the extracted features by the pretrained models and the previous normalization layer to 0 which prevents overfitting. n - dimensional output tensor from the last linear transformation layer is given as an input to the Softmax layer where n is the number of classes. We consider $n = 6$ for the FGCD dataset described in subsection 3.1, and $n = 3$ for the Twitter Parsed Cyberbullying Dataset described in subsection 3.2. Then the final Softmax layer gives the class-wise probabilities.

4.5. Ensemble Methods

In our proposed architecture we experimented with two well-known ensemble techniques. They are Max-voting and Probability Averaging [44].

4.5.1. Max-voting. From our classifier network, for every comment/sample, we obtained class-wise probabilities. Our proposed architecture is an amalgamation of four transformer-based pretrained models and classifier networks. Outputs from each of these classifier networks are passed through an argmax layer. The output of this layer is the class label which has the maximum probability score. From the four classifier networks, we receive a total of four class labels. Now, among these four labels, the highest occurring label is selected as the final label for the input sample.

4.5.2. Probability Averaging. Unlike the Max-voting ensemble technique, in Probability Averaging we did not pass the outputs of four classifier networks through an argmax layer. After passing the same sample through four different networks we obtain four sets of class-wise probabilities. Then these four probability values were aggregated for each class and divided by 4. This gave us the average probability for each class combining the four classifiers. Then, these probability values were passed through an argmax layer to predict the final class label. For an input sample x, the class label can be mathematically written as:

$$class(x) = argmax \sum_{k=1}^{4} P_{M_k}(y = c_i|x) \tag{3}$$

where M_k denotes classifier k and $P_{M_k}(y = c|x)$ denotes the probability of y obtaining the value c for a sample x. In our proposed architecture, the value range for k is from 1 to 4 since we are using four separate classifiers.

Table 2: Hardware and Software Configuration

Name	Parameter
CPU	Intel Xeon Base Clock Speed- 2.3 GHz
GPU	NVIDIA Tesla T4
RAM	13 GB
VRAM	15 GB
CUDA Ver.	11.2

4.6. Experimental Setup

The experiments used PyTorch[3] as the deep learning framework and were run on Google Colab[4] with necessary libraries. The software and hardware configuration is shown in Table 2.

Instead of updating the model weights based on a single epoch, the input samples were divided into several mini-batches with a batch size of 16. The gradients for updating the weights were calculated based on each mini-batch of samples [45]. There are several benefits of this technique such as fitting data into memory while there is a resource constraint, reducing the generalization error, and creating a regularization effect.

As our optimizer, we have used AdamW [46] optimizer which is a modified implementation of Adam optimizer [47]. Unlike common implementations of Adam, AdamW uses a fixed weight decay.

The learning rate was set to $3e{-}5$ and 20% of the steps were set to warm up steps which denote that the training phase will require the first 20% of the steps to increase the learning rate from 0 to $3e - 5$. The number of steps was calculated according to Formula 4 where $NumSamples$ denotes the total number of training samples, $BatchSize = 16$, and $NumEpochs = 15$.

$$Steps = (NumSamples/BatchSize) * NumEpochs \tag{4}$$

4.7. Evaluation Metrics

To get an estimate about the performance of our proposed architecture, we picked accuracy, precision, recall, F1-score, Matthews Correlation Coefficient (MCC), model size,

3. https://pytorch.org/docs/
4. https://colab.research.google.com/

and parameter count as our evaluation metrics. Accuracy is a very well-known metric for measuring the performance of a classifier, but it is sensitive to class imbalance. For example, the accuracy of a model can be very high even while misclassifying a larger portion of samples from a class with less number of samples compared to other classes. Since we tested our proposed architecture on both balanced and imbalanced datasets, we also measured precision, recall, and F1-score. Due to the asymmetric behavior based on the class of interest of precision, recall, and F1-score, we also computed the MCC score of our proposed architecture.

Our proposed architecture was evaluated on test samples that were unseen to the network during the training phase.

4.7.1. Accuracy. The ratio between the number of successfully predicted samples from the test set and the total number of samples in the test set yields accuracy. It reflects the architecture's overall performance.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} * 100\% \quad (5)$$

where,
$TP =$ True positive sample count.
$TN =$ True negative sample count.
$FP =$ False positive sample count.
$FN =$ False negative sample count.

4.7.2. Precision. The ratio between the number of successfully predicted positive samples and the total number of samples that have been predicted as positive is known as precision.

$$Precision = \frac{TP}{TP + FP} \quad (6)$$

4.7.3. Recall. In a multiclass scenario, the ratio of successfully identified samples to the total number of samples that should have been predicted for that class is known as recall.

$$Recall = \frac{TP}{TP + FN} \quad (7)$$

4.7.4. F1-Score. The weighted average of both precision and recall is known as the F1-Score. It is critical to attain a higher F1-Score for an imbalanced dataset.

$$F1 = \frac{2 * Precision * Recall}{Precision + Recall} \quad (8)$$

4.7.5. Matthews Correlation Coefficient (MCC). MCC is determined in the same way that the correlation coefficient between any set of variables is computed. Higher correlation between the true and predictive values indicate better prediction. MCC is obtained using Formula 9.

$$MCC = \frac{a - b}{\sqrt{c * d}} \quad (9)$$

where,
$a = TP * TN$
$b = FP * FN$
$c = (TP + FP)(TP + FN)$
$d = (TN + FP)(TN + FN)$

4.7.6. Model Size. The amount of space required to save model information is characterized as model size and measured in KiloBytes (KB), MegaBytes (MB), and GigaBytes (GB), etc.

4.7.7. Parameter Count. Through backward propagation of losses, the values of learnable parameters are changed to fine-tune the network thus providing better accuracy. The total number of learnable parameters in a network is closely related to the network's inference time, model size, training time, and other critical metrics.

5. Results and Discussion

To assess the competence of our proposed architecture, we analyzed the experimental results based on two different datasets described in Section 3. In this section, we illustrate and validate our obtained results.

5.1. Classification Results

Following the baseline provided by Wang et al. [9], we provide results obtained by evaluating our proposed model for the Fine-grained Cyberbullying Dataset (FGCD), five classes (40,000 tweets) which only finds out the type of cyberbullying. The authors did not evaluate their model for the 'not cyberbullying (Not CB)' class. So it can lead to a wrong classification if a test sample from the excluded class is provided. Therefore, we also evaluate our proposed model on the same FGCD dataset considering all six classes so as to avoid the wrong classifications of comments that are 'not cyberbullying' in nature. In a practical scenario, it is obvious that not every comment on social media will be a case of cyberbullying.

5.1.1. FGCD - Five Classes (40,000 tweets). Our experimental results in Table 3 show that, among the variants of the transformer-based model, the RoBERTa-based model outperforms almost every other model on the five-class classification problem on the FGCD dataset. Between the two ensemble learning approaches, Probability Averaging outperforms every individual model in all evaluation metrics. The model-wise performance is analyzed in Subsection 5.4.

5.1.2. FGCD - Six Classes (48,000 tweets). As shown in Table 4, with the inclusion of 'not cyberbullying (Not CB)' class in the test sample, all the models' performance decreases significantly. Like the experiment with five classes, RoBERTa outperforms all the other BERT variants with six classes as well. The probability averaging ensemble on the four BERT variants outperforms all the individual classifiers, but the accuracy and F1-score drop around 5% as compared with the five-class experiment. The class-wise performance

Table 3: Model-wise Performance of Fine-grained Cyberbullying Detection **(Five Classes-40,000 tweets)**

Model	Model Size (KB)	Parameter Count	Accuracy	F1-Score	Precision	Recall	MCC
BERT	427,943	109531909	92.60%	92.61%	92.63%	92.60%	90.76%
DistilBERT	259,464	66412549	92.75%	92.75%	92.79%	92.75%	90.96%
RoBERTa	**487,176**	**124695301**	**94.97%**	**94.96%**	**94.96%**	**94.97%**	**93.71%**
XLNet	456,197	116768005	94.97%	94.94%	94.92%	94.97%	93.72%
MaxVoting (Ensemble)	N/A	N/A	93.90%	93.90%	93.93%	93.90%	92.38%
Probability Averaging (Ensemble)	**N/A**	**N/A**	**95.60%**	**95.59%**	**95.58%**	**95.60%**	**94.50%**

Table 4: Model-wise Performance of Fine-grained Cyberbullying Detection **(Six Classes-48,000 tweets)**

Model	Model Size (KB)	Parameter Count	Accuracy	F1-Score	Precision	Recall	MCC
BERT	427,943	109531974	80.38%	80.34%	80.35%	80.38%	76.46%
DistilBERT	259,464	66412614	81.33%	81.21%	81.13%	81.33%	77.61%
RoBERTa	**487,176**	**124695366**	**84.19%**	**84.06%**	**84.04%**	**84.19%**	**81.06%**
XLNet	456,197	116768070	83.75%	83.73%	83.71%	83.75%	80.51%
MaxVoting (Ensemble)	N/A	N/A	78.40%	77.61%	77.20%	78.40%	74.15%
Probability Averaging (Ensemble)	**N/A**	**N/A**	**90.76%**	**90.65%**	**90.57%**	**90.76%**	**88.91%**

Table 5: Class-wise Classification Report using Probability Averaging Ensemble

Experiment	Classes	Precision	Recall	F1-Score
Five Classes	Age	0.9843	0.9886	0.9864
	Ethnicity	0.9880	0.9905	0.9893
	Gender	0.9200	0.9159	0.9180
	Religion	0.9837	0.9874	0.9856
	Others	0.9039	0.8983	0.9011
Six Classes	Age	0.9060	0.9631	0.9337
	Ethnicity	0.9670	0.9433	0.9550
	Gender	0.8593	0.8630	0.8611
	Not CB	0.5761	0.5593	0.5676
	Others	0.6237	0.6139	0.6188
	Religion	0.9340	0.9363	0.9352

presented in Table 5 clearly indicates the reason behind this drop in performance which is analyzed in Subsection 5.3.

5.1.3. Twitter Parsed Dataset. FGCD is a completely balanced dataset and our proposed ensemble architecture performs well on that dataset. To evaluate how our proposed architecture performs on an imbalanced dataset, we used the Twitter parsed dataset provided by Waseem et al. [30]. The model-wise performance on this dataset is presented in Table 6. Unlike the FGCD dataset, XLNet performs slightly better than RoBERTa among the individual classifiers. The probability averaging ensemble method outperforms all the individual classifiers on this dataset as well.

5.2. Comparison with Baseline

5.2.1. FGCD - Five Classes (40,000 tweets). Efficient feature selection can have a significant impact on the performance of machine learning models and Wang et al.'s [9] provided baseline for five classes of Fine-grained Cyberbullying Dataset (FGCD) proved this fact. They achieved decent accuracy and F1-Score with simple word embedding methods like Bag of Words (which counts the word frequencies) and TF-IDF (which counts weighted word frequencies) with traditional classifiers, such as decision trees. The baseline evaluation metrics were test accuracy and F1-Score. Our proposed ensemble model of probability averaging outperforms their best model by 1.22% in test accuracy in 1.15% in F1-Score. Table 7 compares the results of our study with the baseline models.

5.2.2. Twitter Parsed Dataset. The baseline of this dataset was provided using different combinations of semantic embeddings, namely, char n-gram, TF-IDF, and bag of words, with simple classifiers, such as logistic regression, Support Vector Machine (SVM), and decision tree. As presented in Table 8, our proposed architecture outperforms all the baseline methods by a large margin.

Our proposed architecture can learn abstract features with the attention mechanism and performs better on these datasets compared to the feature-based approaches applied by the respective authors.

5.3. Error Analysis

The probability averaging of the four BERT variants on FGCD (six classes) reports lower performance on all evaluation metrics compared to FGCD (five classes) experiments. Although it still achieves very decent accuracy on FGCD (six classes) experiments, we performed an error analysis on the result of the probability averaging ensemble. We followed the research of Aken et al. [48] to find the common error issues in ensembling.

We took 7950 (around 20% of the dataset) test samples for the experiment with FGCD (five classes) and 9541 (around 20% of the dataset) test samples for FGCD (six classes) experiments. As from Table 5, 'Others' class had the F1-Score of 0.9011 in the FGCD (five classes) experiment. After the inclusion of the 'not CB' class, it decreases to 0.6188 and 'not CB' has a very low F1-Score of 0.5676. Evaluation metrics for other classes remained almost steady in both experiments. To better analyze this, we refer to the confusion matrices of the two experiments in Figure 5. In the FGCD (five classes) experiment excluding the 'not-CB' class, the 'Others' class had the correct

Table 6: Model-wise Performance on Twitter Parsed Dataset

Model	Model Size (KB)	Parameter Count	Accuracy	F1-Score	Precision	Recall	MCC
BERT	427,942	109531779	84.63%	84.71%	84.91%	84.63%	68.42%
DistilBERT	259,464	66412419	84.87%	84.72%	84.81%	84.87%	67.92%
RoBERTa	487,176	124695171	86.27%	86.25%	86.26%	86.27%	71.31%
XLNet	**456,197**	**116767875**	**86.47%**	**86.49%**	**86.57%**	**86.47%**	**71.93%**
MaxVoting (Ensemble)	N/A	N/A	86.50%	86.46%	86.54%	86.50%	71.78%
Probability Averaging (Ensemble)	**N/A**	**N/A**	**87.42%**	**87.28%**	**87.37%**	**87.42%**	**73.29%**
BOW + XGBoost	N/A	N/A	84.21%	83.51%	83.89%	84.21%	63.85%

Table 7: Performance Comparison with Baseline of FGCD Dataset Using Test Accuracy and F1-Score

Paper	Model Used	Test Accuracy	F1-Score
Wang et. al [9]	BOW + XGBoost	94.38%	94.44%
Our Model	**Probability Averaging (Ensemble)**	**95.60%**	**95.59%**

Table 8: Performance Comparison with Baseline of Twitter Parsed Dataset

Work	Method	Precision	Recall	F1-Score
Baselines	Char n-gram +Logistic Regression	72.9%	77.8%	75.3%
	TF-IDF+ Balanced SVM	81.6%	81.6%	81.6%
	TF-IDF+GBDT	81.9%	80.7%	81.3%
	BoW+Balanced SVM	79.1%	78.8%	78.9%
	BoWV+GBDT	80.0%	80.2%	80.1%
Our Model	**Probability Averaging (Ensemble)**	**87.37%**	**87.42%**	**87.28%**

classification rate of 89.83%. After the inclusion of the 'not-CB' class, it drops down to 76.66% and the 'not-CB' class has the lowest correct classification rate of 74.49%. 289 samples (around 18.24%) from 'not CB' class were misclassified as 'Others', and 287 samples (around 18.11%) from the 'Others' class were misclassified as 'not CB' class. This is mostly due to doubtful labels or bad labeling. To look into this, we investigated the labels of these two classes. We extracted the tokens of these two classes using the bag of words (BOW). 18460 notable tokens were found for the 'not CB' class and 15372 tokens were found for the 'Others' class after stop words and punctuation removal. 7030 tokens were found common in both of the classes. In Table 9, the most frequent 100 tokens from 'not CB' and 'Others' classes have been listed, where 66 tokens were common in these two classes. The doubtful labels of these two classes are apparent from the test samples presented in Table 10. The tweets mentioned for both classes are used for explanation or self-reproach, but because of carrying toxic words, some of them were classified as 'Others' types of cyberbullying. We observed a high number of similar cases and we conclude that our classifier misclassified these samples mostly because of the doubtful labels.

The confusion matrix for the experiment on the Twitter parsed dataset presented in Figure 6 shows that there was significant misclassification between the 'Racism' and 'None', and also 'Sexism' and 'None' classes. Around 26% of the 'Sexism' test samples were misclassified as 'None', whereas 21% 'Racism' samples were misclassified as 'None' class. One of the reasons behind this misclassification might be the heavy imbalance of the dataset. The 'Sexism' and 'Racism' classes had very fewer training samples compared to the 'None' class. We extracted the tokens of all the classes. Surprisingly, more tokens were common between the 'Racism' and 'None' classes despite the lower misclassification rate between these two classes. We observed from the dataset that the 'Racism' class had more decisive labeling than the 'Sexism' class. This is another evidence for our claim that doubtful labeling causes more misclassification in cyberbullying detection.

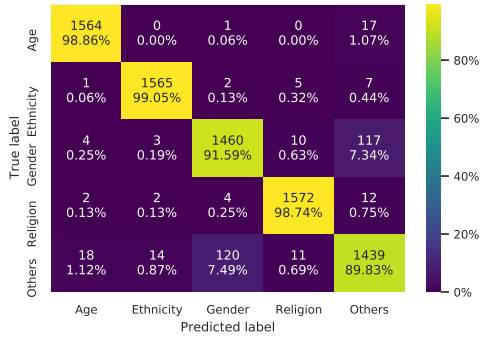
Table 9: Most Frequent 100 Tokens from 'not CB' and 'Others' class of FGCD Dataset

Category	Most Frequently Used Tokens
Not CB	mkr, http, rt, bullying, just, bully, like, don, school, kat, amp, people, know, time, andre, think, good, love, que, going, lol, ve, really, women, make, need, got, oh, want, https, class, shit, twitter, gt, isis, com, way, right, stop, day, did, fuck, say, se, eu, ll, bad, today, lt, islam, hate, sure, daesh, thing, mkr2015, bullied, work, yes, girls, pretty, let, fucking, freebsdgirl, didn, mykitchenrules, look, nao, home, colin, better, thanks, yeah, getting, round, hope, bulling, doing, doesn, come, new, things, great, na, feel, talk, life, haha, man, im, best, college, tell, men, isso, cook, read, looks, old, does, watching
Others	rt, http, just, bully, https, like, don, people, fucking, know, ve, mkr, think, time, bullying, amp, got, bullied, twitter, going, hate, really, idiot, need, want, oh, bad, lol, women, good, ass, freebsdgirl, feminazi, make, ll, things, fuck, stop, love, did, gamergate, way, shit, sure, right, pretty, lt, blameonenotall, thing, isn, yes, say, harassment, getting, stupid, didn, doesn, let, person, coon, doing, look, work, use, idiots, lot, does, today, new, talk, yeah, wadhwa, day, abuse, better, help, thequinnspiracy, tell, dude, tweets, trying, feel, bitch, hell, talking, stuff, gt, im, idea, spacekatgal, chriswarcraft, called, said, men, life, used, tweet, big, sorry, tech

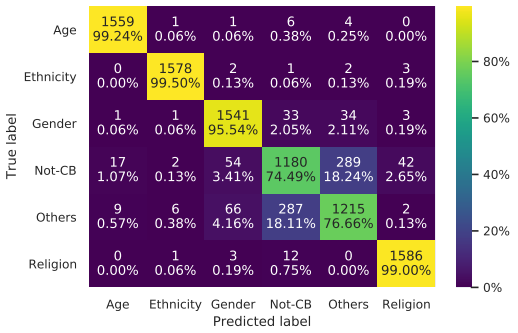
5.4. Performance Analysis

Among the four transformer-based models, RoBERTa performed better in almost all experiments. Although, throughout the experiments, the performances of RoBERTa and XLNet were almost similar. The possible reasons for better performance are as follows -

- Both RoBERTa and XLNet were pretrained with almost 10 times more data than BERT and DistilBERT.



(a) Experiment with Five Classes



(b) Experiment with Six Classes

Figure 5: Confusion Matrices of FGCD Dataset for Probability Averaging Ensemble

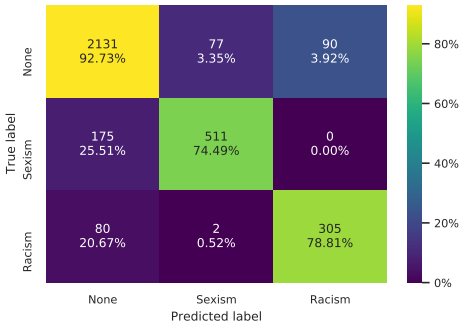


Figure 6: Confusion Matrix of Twitter Parsed Dataset for Probability Averaging Ensemble

Table 10: Test Samples of doubtful labeling from 'not CB' and 'Others' Class of FGCD Dataset

Not CB	**Others**
Bullying linked to decreases in GPA, longer lasting negative psychological effects, and obesity. More evidence... http://t.co/UCdxYcj	@momarazzi95 Respect to you and Zach. Bullying should never be tolerated in any form. Have a nice day, Kerry :)
Started watching Port v Magpies but turned it off. I hate to see bullying in the workplace	Why should managers be allowed to bully just because they are "higher" than you :(#stopbullying

- RoBERTa and XLNet have around 20 million more trainable parameters than $BERT_{BASE}$.
- RoBERTa used a more optimized pretraining configuration with a higher batch size than other transformer-based models.
- XLNet removed the pretrain-finetune discrepancy which is still present in the other BERT variants we used.
- XLNet used all possible permutations of the input factorization order to capture bidirectional context.

DistilBERT's triple loss function along with the supervision of a parent model during pretraining helps it to achieve a smaller model size and parameter count, which makes it an ideal choice for using in underpowered devices, i.e., in some cases when deploying a neural network model on servers is not possible for applications which use end-to-end device encryption. Despite having a low parameter count and faster training, the performance of DistilBERT is still on par with the much larger models we used. In the experiments with the Twitter parsed dataset, XLNet overshadowed the performance of RoBERTa. We assume that the removal of the pretrain-finetune discrepancy along with considering all possible permutations of the input factorization order helps XLNet to perform better with an imbalanced dataset with a smaller amount of samples.

Our best performing architecture, the probability averaging ensemble, outperforms all individual transformer-based models such as BERT, RoBERTa, XLNet, and DistilBERT since it incorporates all optimizations that individual models introduced to outperform the base variant, BERT. Incorporating four model probabilities makes sure that the predicted label has the highest decision probability among all models, not just the vote count. On the contrary, our max-voting ensemble could not perform as well as the other models, mostly because of the fact that in our voting mechanism we used an even number of models which in some cases resulted in a scenario where two different class labels obtained equal amount of votes. In this case, picking one of the labels randomly deteriorated the model performance.

6. Conclusion

We introduced an ensemble approach to classify cyberbullying from social media comments which achieves decent performance on both balanced and imbalanced datasets. We also illustrated the capability of our proposed architecture by comparing it with other state-of-the-art researches and analyzed the performance of transformer-based models and their ensembles. We further conducted an in-depth error analysis on the performance of our proposed architecture, which can be pioneering for future research work in this domain. In future, we want to extend this work by incorporating our experimental setup with datasets from other social networking platforms.

References

[1] J. Xu, K.-S. Jun, X. Zhu, and A. Bellmore, "Learning from bullying traces in social media," in *NAACL*, 2012.

[2] P. B. O'Sullivan and A. J. Flanagin, "Reconceptualizing 'flaming' and other problematic messages," *New Media & Society*, vol. 5, pp. 69 – 94, 2003.

[3] N. Willard, "Cyberbullying and cyberthreats: Responding to the challenge of online social aggression, threats, and distress," 2007.

[4] N. Djuric, J. Zhou, R. Morris, M. Grbovic, V. Radosavljevic, and N. L. Bhamidipati, "Hate speech detection with comment embeddings," *Proceedings of the 24th International Conference on World Wide Web*, 2015.

[5] C. Nobata, J. R. Tetreault, A. Thomas, Y. Mehdad, and Y. Chang, "Abusive language detection in online user content," *Proceedings of the 25th International Conference on World Wide Web*, 2016.

[6] K. Reynolds, A. Kontostathis, and L. Edwards, "Using machine learning to detect cyberbullying," *2011 10th International Conference on Machine Learning and Applications and Workshops*, vol. 2, pp. 241–244, 2011.

[7] S. Agrawal and A. Awekar, "Deep learning for detecting cyberbullying across multiple social media platforms," *ArXiv*, vol. abs/1801.06482, 2018.

[8] C. Van Hee, E. Lefever, B. Verhoeven, J. Mennes, B. Desmet, G. Pauw, W. Daelemans, and V. Hoste, "Automatic detection and prevention of cyberbullying," 10 2015.

[9] J. Wang, K. Fu, and C.-T. Lu, "Sosnet: A graph convolutional network approach to fine-grained cyberbullying detection," in *2020 IEEE International Conference on Big Data (Big Data)*, 2020, pp. 1699–1708.

[10] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, Ł. Kaiser, and I. Polosukhin, "Attention is all you need," in *Advances in neural information processing systems*, 2017, pp. 5998–6008.

[11] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[12] V. Sanh, L. Debut, J. Chaumond, and T. Wolf, "Distilbert, a distilled version of bert: smaller, faster, cheaper and lighter," *arXiv preprint arXiv:1910.01108*, 2019.

[13] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," *arXiv preprint arXiv:1907.11692*, 2019.

[14] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," *Advances in neural information processing systems*, vol. 32, 2019.

[15] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *ICWSM*, 2017.

[16] A. Mittos, S. Zannettou, J. Blackburn, and E. D. Cristofaro, ""and we will fight for our race!" a measurement study of genetic testing conversations on reddit and 4chan," in *ICWSM*, 2020.

[17] R. Ottoni, E. Cunha, G. Magno, P. Bernardina, W. Meira, and V. A. F. Almeida, "Analyzing right-wing youtube channels: Hate, violence and discrimination," *Proceedings of the 10th ACM Conference on Web Science*, 2018.

[18] S. Malmasi and M. Zampieri, "Challenges in discriminating profanity from hate speech," *Journal of Experimental & Theoretical Artificial Intelligence*, vol. 30, pp. 187 – 202, 2018.

[19] P. Burnap and M. Williams, "Cyber hate speech on twitter: An application of machine classification and statistical modeling for policy and decision making," *Policy & Internet*, vol. 7, pp. 223–242, 2015.

[20] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," *Proceedings of the 26th International Conference on World Wide Web Companion*, 2017.

[21] J. H. Park and P. Fung, "One-step and two-step classification for abusive language detection on twitter," in *ALW@ACL*, 2017.

[22] B. Gambäck and U. K. Sikdar, "Using convolutional neural networks to classify hate-speech," in *ALW@ACL*, 2017.

[23] Z. Zhang and L. Luo, "Hate speech detection: A solved problem? the challenging case of long tail on twitter," *CoRR*, vol. abs/1803.03662, 2018. [Online]. Available: http://arxiv.org/abs/1803.03662

[24] A. Jha and R. Mamidi, "When does a compliment become sexist? analysis and classification of ambivalent sexism using twitter data," in *NLP+CSS@ACL*, 2017.

[25] I. Kwok and Y. Wang, "Locate the hate: Detecting tweets against blacks," in *AAAI*, 2013.

[26] S. Sharma, S. Agrawal, and M. Shrivastava, "Degree based classification of harmful speech using twitter data," in *TRAC@COLING 2018*, 2018.

[27] G. L. De la Pena Sarracén, R. G. Pons, C. E. M. Cuza, and P. Rosso, "Hate speech detection using attention-based lstm," *EVALITA Evaluation of NLP and Speech Tools for Italian*, vol. 12, p. 235, 2018.

[28] S. Zimmerman, U. Kruschwitz, and C. Fox, "Improving hate speech detection with deep learning ensembles," in *LREC*, 2018.

[29] J. Wang, K. Fu, and C.-T. Lu, "Fine-grained balanced cyberbullying dataset," 2020.

[30] Z. Waseem and D. Hovy, "Hateful symbols or hateful people? predictive features for hate speech detection on twitter," in *NAACL*, 2016.

[31] U. Bretschneider, T. Wöhner, and R. Peters, "Detecting online harassment in social networks," in *ICIS*, 2014.

[32] D. Chatzakou, I. Leontiadis, J. Blackburn, E. D. Cristofaro, G. Stringhini, A. Vakali, and N. Kourtellis, "Detecting cyberbullying and cyberaggression in social media," *ACM Transactions on the Web (TWEB)*, vol. 13, pp. 1 – 51, 2019.

[33] Z. Waseem, "Are you a racist or am i seeing things? annotator influence on hate speech detection on twitter," in *NLP+CSS@EMNLP*, 2016.

[34] M. Karim, S. K. Dey, B. R. Chakravarthi *et al.*, "Deephateexplainer: Explainable hate speech detection in under-resourced bengali language," *arXiv preprint arXiv:2012.14353*, 2020.

[35] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[36] A. Radford, K. Narasimhan, T. Salimans, and I. Sutskever, "Improving language understanding with unsupervised learning," 2018.

[37] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[38] M. Ott, S. Edunov, D. Grangier, and M. Auli, "Scaling neural machine translation," *arXiv preprint arXiv:1806.00187*, 2018.

[39] Z. Dai, Z. Yang, Y. Yang, J. Carbonell, Q. V. Le, and R. Salakhutdinov, "Transformer-xl: Attentive language models beyond a fixed-length context," *arXiv preprint arXiv:1901.02860*, 2019.

[40] M. Shoeybi, M. Patwary, R. Puri, P. LeGresley, J. Casper, and B. Catanzaro, "Megatron-lm: Training multi-billion parameter language models using model parallelism," *arXiv preprint arXiv:1909.08053*, 2019.

[41] Y. Zhu, R. Kiros, R. Zemel, R. Salakhutdinov, R. Urtasun, A. Torralba, and S. Fidler, "Aligning books and movies: Towards story-like visual explanations by watching movies and reading books," in *Proceedings of the IEEE international conference on computer vision*, 2015, pp. 19–27.

[42] J. L. Ba, J. R. Kiros, and G. E. Hinton, "Layer normalization," *arXiv preprint arXiv:1607.06450*, 2016.

[43] S. Ioffe and C. Szegedy, "Batch normalization: Accelerating deep network training by reducing internal covariate shift," in *International conference on machine learning*. PMLR, 2015, pp. 448–456.

[44] L. Rokach, "Ensemble methods for classifiers," in *Data mining and knowledge discovery handbook*. Springer, 2005, pp. 957–980.

[45] S. Khirirat, H. R. Feyzmahdavian, and M. Johansson, "Mini-batch gradient descent: Faster convergence under data sparsity," in *2017 IEEE 56th Annual Conference on Decision and Control (CDC)*. IEEE, 2017, pp. 2880–2887.

[46] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," *arXiv preprint arXiv:1711.05101*, 2017.

[47] D. P. Kingma and J. Ba, "Adam: A method for stochastic optimization," *arXiv preprint arXiv:1412.6980*, 2014.

[48] B. van Aken, J. Risch, R. Krestel, and A. Löser, "Challenges for toxic comment classification: An in-depth error analysis," in *ALW*, 2018.