Cyberbullying Tweets Classification: Group D Final Project

Satyen Sabnis
sas60739@uga.edu
University of Georgia
Athens, Georgia, USA

Mamadou Ly II
mal33176@uga.edu
University of Georgia
Athens, Georgia, USA

Rachel Chesebro‘
rec05958@uga.edu
University of Georgia
Athens, Georgia, USA

Jesus Cruz
jfc17956@uga.edu
University of Georgia
Athens, Georgia, USA

ABSTRACT

Cyberbullying on social media has become more prevalent as more users gain access to social media platforms. As a result, its effect of mental health comes into question which calls for the need to detect instances of bullying on these platforms. The objective of this report is to demonstrate the steps taken to tackle this task through various models learned in the Data Science Capstone Course. The task required the use of Gated Recurrent Units (GRUs), Long Short-Term Memory networks (LSTMs), and a pre-trained Bidirectional Encoder Representation from Transfomers (BERT) model to aid in classifying cyberbullying from the cyberbullying tweet dataset. These models were evaluated using various metrics such as precision, accuracy, recall, and F-1 score to decide which model performed the best for the task.

ACM Reference Format:
Satyen Sabnis, Mamadou Ly II, Rachel Chesebro‘, and Jesus Cruz. 2023. Cyberbullying Tweets Classification: Group D Final Project. In *Proceedings of ACM Conference (Conference'17).* ACM, New York, NY, USA, 5 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 BACKGROUND

Cyberbullying involves repeated harmful behavior directed at individuals through digital platforms, ranging from spreading rumors and posting hurtful comments to making aggressive threats. The psychological impact on victims can be severe, including heightened risks of depression, anxiety, and increased suicide attempts. Consequently, the detection and prevention of cyberbullying have become increasingly important to maintain a safer online environment.

Manually detecting cyberbullying is challenging due to the vast amount of data and the subtlety of the language involved. Therefore, automated tools leveraging machine learning (ML) and natural language processing (NLP) are necessary. Models like GRUs, LSTMs, and transformers (e.g., BERT) can help understand patterns and detect harmful content effectively.

Conference'17, July 2017, Washington, DC, USA

ACM ISBN 978-x-xxxx-xxxx-x/YY/MM…$15.00
https://doi.org/10.1145/nnnnnnn.nnnnnnn

2 RELATED WORK

- Bag-of-Words and Feature Engineering: Earlier works by Dinakar et al. (2011) utilized traditional ML models, combining Bag-of-Words and handcrafted features to identify cyberbullying. Though these methods were groundbreaking at the time, their performance was limited due to the variability of language in social media.
- Deep Learning Approaches: More recent studies, such as Rosa et al. (2019), used convolutional neural networks (CNNs) to capture semantic and syntactic features of texts, outperforming earlier methods.
- Sentiment Analysis with BERT: Devlin et al. (2018) introduced BERT, which has been adapted for various NLP tasks beyond cyberbullying detection, such as sentiment analysis, where it significantly improves the accuracy over previous methods.
- GRU for Text Classification: Cho et al. (2014) originally developed GRUs for machine translation, but their ability to understand sequential data has led to their adoption in numerous NLP tasks, including text classification and sentiment analysis.
- LSTM for Hate Speech Detection: LSTM models have been used by researchers like Badjatiya et al. (2017) to identify hate speech, which shares characteristics with cyberbullying in terms of language patterns and contextual nuances.
- Transformer Models in Offensive Language Detection: Zhang et al. (2020) used transformers, similar to BERT, to detect offensive language, a task closely related to cyberbullying. Their model benefited from transformers' ability to understand context across entire sentences

3 METHODOLOGY

3.1 Data Processing

Given a dataset of collected tweets alongside their appropriate cyberbullying classifications, our task was to construct a model that would correctly classify these tweets under six different labels: religion, age, ethnicity, gender, not.cyberbullying, and other.cyberbullying.

In order to properly construct the model, the dataset needed to be processed before use. Many parts of the tweets provided insight into what classified as cyberbullying while others served moreso as noise that could disrupt the results of any model. Unnecessary URLs, usernames, and special characters needed to be removed from each tweet in the dataset. Regular expression syntax helped

to remove these pieces of the tweets while leaving the rest of the tweet.

Another element that was deemed unnecessary were stopwords; words that are deemed insignificant but are frequently used. Normally, stopwords include items such as articles and prepositions which do not hold much information in the context of the tweet. Lastly, we removed long and short tweets alike. After cleaning the tweets, the data was partitioned into a training, testing, and validation dataset, to help train, fine-tune, and evaluate our models.

3.2 GRU

As we are working with text data we want to use models that are designed to handle sequential data. We looked at various different models that we could apply such as RNN (Recurrent Nueral Network), LSTM (Long Short Term Memory), GRU(Gated recurrent unit) , Transformers, and also prompt engineering using LLM. Ultimately we decided to use GRU, LSTM, and the BERT transformer model published by Google. All these models have shown to have efficient performance for sequential data.

GRU or Gated Recurrent Unit is a type of recurrent neural network architecture that is designed to capture long-range dependencies on sequential data. GRU's help solve the vanishing gradient problems of traditional RNN's by being able to maintain long-term dependencies within the input data. In a RNN as you move farther and farther in a sequence the influence of the starting inputs diminishes as the input series increases. Especially in longer input sequences there might be a loss of information and slow convergence of learning. GRU's combat this by having an update gate and a reset gate. These gates help determine what information is brought forward through the network and what information should be forgotten. In particular, the update gate helps the model to determine how much of the past information needs to be passed along to the future.While the reset gate decides how much of the past information to forget.

We implemented the GRU architecture using the Tensorflow package in Python. Tensorflow is a machine learning library that can be used to build and train deep learning models. The first step in building this model was to actually determine how to feed the data into the model. First we split the data into training, validation, and testing data. Then, we tokenized and padded the training and validation data. Tokenization was done to look at the individual words to later apply word embeddings. Padding was done to make sure the data was the same size before feeding it into the model. The next step was to create an embedding matrix that used Google's Word2vec model to convert our text data into embeddings with 300 dimensions. The matrix would be used in the embedding layer of our model. In our training process we defined our Adam optimizer and our loss function, categorical cross entropy. Then we used the Sequential function in tensorflow to add our layers which consisted of the embedding, gru, and two dense layers. The last dense layer was of size 6 which was the number of classes we were trying to predict.

3.3 Pre-Trained BERT Model

Bert or Bidirectional Encoder Representation from Transformers is a transformer based model introduced by Google in 2018. It is designed to understand the context of words by considering the surrounding words (i.e able to read the entire sequence of words at once instead of looking at it in one direction). It improves on Large Language models like RNN or LSTM which typically process text in a unidirectional manner. Some limitations of these models are that they are slow and fail to capture the full context of a word. BERT improves this by applying bidirectional training and utilizes the transformer attention mechanism. BERT learns language by training on two unsupervised tasks simultaneously; Masked Language Model (MLM) and Next Sentence Prediction (NSP). MLM helps BERT understand bidirectional context within a sentence by training to predict the missing words in a sentence based on the context provided by the surrounding words. While NSP, helps BERT understand context across different sentences by receiving sentences in pairs and learns to predict if the second sentence in the pair is the subsequent sentence in the original document. When training bert these two are trained together to create the BERT mechanism.

Training a BERT model from scratch is a very intensive operation so instead you fine tune an already pretrained model. In our implementation we used the transformers package to import our bert tokenizer and bert model. The specific tokenizer and model we used was the 'bert-base-uncased' which is a pretrained model that was developed by Google. To simplify the training process, the transformers package provides a Trainer class that is optimized for training. This makes it easier to start training without manually writing your own training loop. Like with our GRU model, we tokenized and padded our data. To tune hyperparameters the transformers package has a class called TrainingArguments which contains all the hyperparameters you can tune as well flags for activating different training options. For our model we mainly focused on tuning the learning rate and the number of epochs to evaluate the model. Then, in the trainer class we inputted our pretrained model, the training arguments (contains our hyperparameters), the train and validation dataset, and a compute_metrics() function which will evaluate how well our model performed. Finally to train our model all we had to do was use the .train() function on our trainer class to train our model.

3.4 LSTM

To construct the LSTM model, Tensorflow's Keras library was utilized to establish a Sequential model that would house the necessary layers for the LSTM model. The first layer was an Embedding layer that would take in the input text and represent it as a vector. Here, the tweets would be tokenized based on its content words before they became a numerical representation for the model to process. Next, the LSTM layer was added. This layer had the hyperparameter to adjust the number of hidden nodes/units in this layer which was adjusted in the ablation studies. Lastly, a Dense layer was added to serve as the output layer for this LSTM model. The final activation function in this layer is the 'softmax' function which converted the output vectors of previous layers into six probabilities from 0 to 1. Each probability represented the chances of a certain tweet representing a one of the six labels in the classification model. The highest probability of the six labels determined how the cyberbullying tweet was classified.

Before compiling, the model was instructed to train properly by using Binary Cross Entropy (BCE) Loss as our loss function and the optimizer was set to be Adam. The goal of this model was the minimize this BCE loss as the model iterated the training dataset through various batches across several epochs. The number of epochs and the batch size during training was also adjusted for the ablation studies of the LSTM model.

After training, predicted values were generated from our model after it was given data it had not seen before: the testing data set. From these predicted values, a classification report that displayed the precision, accuracy, recall, and F-1 score was produced after comparing with the testing data set's ground truth labels.

After modifying the hyperparameters of this LSTM network, the classification report metrics served to compare each combination of hyperparameters. The final, best model from these modifications was taken to compare with the other two models.

4 EXPERIMENT

4.1 GRU

In our GRU model we tried to experiment with many hyperparameters to get the best accuracy. The hyperparameters we focused on were the learning rate, the batch size, the number of epochs, and the number of units in the GRU layer.

LR	Accuracy
0.01	0.840
0.001	0.846
0.0001	0.854
0.00005	0.850

Table 1: Metric Table for GRU Learning Rate Adjustments

Epochs	Accuracy
5	0.856
10	0.854
20	0.851

Table 2: Metric Table for the GRU Epochs

Batch Size	Accuracy
16	0.856
32	0.854
64	0.851
128	0.846

Table 3: Metric Table for the GRU Batch Size

Number of Units in GRU Layer	Accuracy
32	0.829
64	0.854
128	0.845
256	0.856

Table 4: Metric Table for the GRU Units

The accuracy stayed relatively around the same when testing different hyperparameters. But generally what led to the best accuracy was a learning rate of 0.0001, a batch size of 16. 5 epochs, and 256 units in the GRU layer.

These were the evaluation metrics for our best GRU model.

	precision	recall	f1-score	support
not_cyberbullying	0.68	0.58	0.63	1394
gender	0.90	0.86	0.88	1488
religion	0.94	0.96	0.95	1596
age	0.97	0.97	0.97	1536
ethnicity	0.98	0.97	0.98	1610
other_cyberbullying	0.55	0.70	0.61	988
accuracy			0.85	8612
macro avg	0.84	0.84	0.84	8612
weighted avg	0.86	0.85	0.86	8612

Figure 1: Classification Report for the GRU Model

The accuracy, precision, recall, and F-1 score were all around 84-85%. The model struggled to predict the not_cyberbullying and other_cyberbullying targets.

4.2 Pre-Trained BERT Model

In our Bert model, since training time was very long, we determined the epochs with the best accuracy was 4 and experimented mainly with the learning rates as that had the most effect on the accuracy.

Epochs = 4

LR	Accuracy
6e-6	0.832
2e-5	0.923
5e-5	0.931
1e-4	0.879

Figure 2: Learning Rate Adjustments For Epochs = 4 BERT Model

Changing the learning rates drastically changed the accuracy as a high learning rate,1e-4, most likely overshot the optimal solution and a low learning rate, 6e-6, probably overfitted the data which led to low accuracy. The optimal learning rate with the best accuracy was 5e-5.

These were the evaluation metrics for our best BERT model.

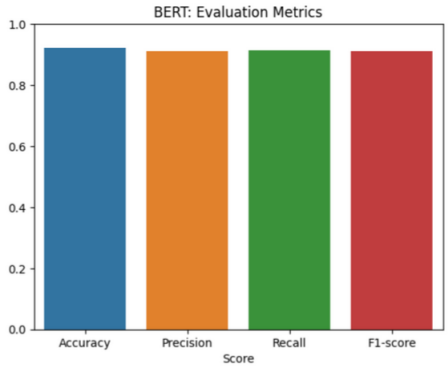

Figure 3: Metrics for the Best BERT Model

	precision	recall	f1-score	support
not_cyberbullying	0.84	0.74	0.79	1395
gender	0.95	0.93	0.94	1449
religion	0.97	0.98	0.97	1621
age	0.99	0.99	0.99	1600
ethnicity	1.00	0.99	0.99	1599
other_cyberbullying	0.70	0.87	0.78	948
accuracy			0.92	8612
macro avg	0.91	0.91	0.91	8612
weighted avg	0.93	0.92	0.92	8612

Figure 4: Classification Report for the BERT Model

Improved on the GRU model with receiving scores around 92% for most metrics. It did a better job of classifying the not_cyberbullying and other_cyberbullying targets probably due the BERT model being more complex i.e being able to capture more information due to its bidirectional nature.

4.3 LSTM

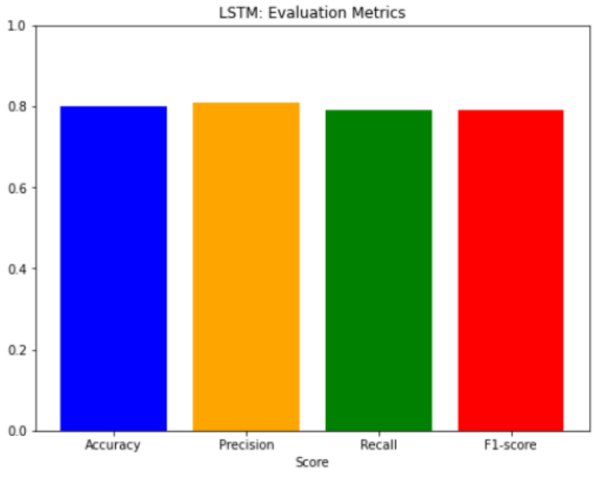

Figure 5: Metrics for the Best LSTM Network Model

	precision	recall	f1-score	support
not_cyberbullying	0.54	0.50	0.52	1448
gender	0.83	0.84	0.84	1530
religion	0.95	0.92	0.94	1594
age	0.97	0.93	0.95	1544
ethnicity	0.97	0.97	0.97	1615
other_cyberbullying	0.50	0.58	0.54	1319
accuracy			0.80	9050
macro avg	0.80	0.79	0.79	9050
weighted avg	0.81	0.80	0.80	9050

Figure 6: Classification Report for the LSTM Network Model

In the ablation study for the LSTM model, the default model used for comparison was a model trained in 10 epochs, with 128 units in the LSTM layer, and when batch size was 64. The best accuracy obtained was 0.817 when the epochs were reduced to 5 from 10. LSTM layer units and batch size stayed consistent to the default model. Still, examining the LSTM network's classification report from Figure 6 revealed that this model struggled more to properly classify when a tweet was not cyberbullying or some other type of cyberbullying not labeled in the dataset. Its precision, recall, and F-1 score circled around 0.5 for both of these labels. This underperformance indicates that a tweet that is an unlabeled type of cyberbullying or not at all might be too complex for this model to classify. The other two models struggled as well, but not to the same extent.

In comparison to the other two models, this LSTM model did not produce competitve results, so it is safe to deem the other two models as better options for cyberbullying classification.

4.4 SHAP

Understanding the decisions made by machine learning models, specifically deep learning models like BERT, is crucial for building trust and identifying potential biases or errors. Model interpretability and explainability techniques aim to explicate the inner workings of these complex models. SHAP (SHapley Additive exPlanations) has emerged as a powerful method for explaining the output of machine learning models. It provides a framework for decomposing the model's prediction into contributions from individual features, allowing for a deeper understanding of how each feature influences the model's decision-making process. The Shapley values output by the model "can be approximated in time linear to the number of features." (Kokalj).

In the context of cyberbullying classification, where the impact of misclassification can be significant, the ability to explain model predictions is critical. SHAP offers a systematic approach to interpret the behavior of models like BERT, which are known for their complexity and black-box nature. By using SHAP, we can address the opacity of deep learning models and gain insights into which words or tokens are most indicative of cyberbullying behavior. This not only enhances the transparency of the classification process but also enables us to identify potential biases and improve model performance. Integrating SHAP into our cyberbullying classification project allowed us to go beyond accuracy metrics and delve deeper into the underlying mechanisms driving our model predictions. Through visualizations and analysis of SHAP explanations, we

aimed to provide actionable insights for mitigating cyberbullying and fostering a safer online environment.

In our project to implement effective cyberbullying classification, we explored a diverse set of models, including traditional methods and sophisticated deep learning architectures. As a group, we ventured into the realm of deep learning, leveraging powerful models such as BERT (Bidirectional Encoder Representations from Transformers), GRU (Gated Recurrent Unit), and LSTM (Long Short-Term Memory). These deep learning architectures offer the capability to capture intricate patterns in textual data, which is crucial for nuanced tasks like cyberbullying detection. To enhance the interpretability of our complex deep learning models like BERT, GRU, and LSTM, we integrated SHAP into our project pipeline. This involved defining a prediction function tailored to each model architecture, enabling the generation of predictions for input text. Utilizing the SHAP library, we created explainer objects that encapsulated the underlying mechanisms of our models. These explainer objects facilitated the computation of SHAP values, providing insights into the contribution of individual features to model predictions. Through the integration of SHAP, we aimed to shed light on the decision-making processes of our models and enhance their transparency and interpretability.

SHAP explanations were used to interpret the predictions of our models. Visualizations such as bar plots and text plots were generated to illustrate the importance of different features (words or tokens) in predicting cyberbullying behavior. The SHAP bar plot provides insights into the importance of different features in predicting cyberbullying behavior.

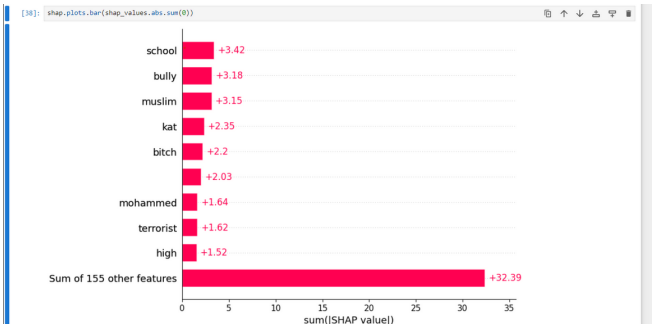

Figure 7: SHAP Bar Plot

From the SHAP values, we can identify the most influential words or tokens that contribute to the model's decision-making process. In our analysis, the following features emerged as particularly influential in predicting cyberbullying behavior: "school" with a SHAP score of +3.42; "bully" with a SHAP score of +3.18; "muslim" with a SHAP score of +3.15; "kat" with a SHAP score of +2.35; "b*tch" with a SHAP score of +2.2. These high SHAP scores indicate that these features strongly contribute to the model's predictions of cyberbullying tweets. Conversely, features with lower SHAP scores, such as "mohammed" and "terrorist," have less (but still notable) influence on the model's decision-making process. The SHAP text plot visualizes the impact of each word/token in a sample sentence on the model's prediction.

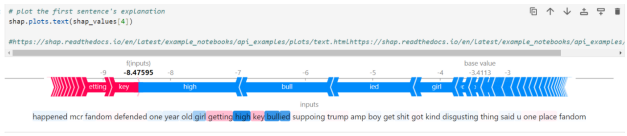

Figure 8: SHAP Text Plot

By examining the SHAP values associated with each word/token, we can understand how individual elements contribute to the overall prediction. For example, in the phrase "happened mcr fandom defended one year old girl getting high key bullies supporting trump amp boy get shit got kind disgusting thing said u one place fandom," the SHAP text plot highlights words like "bullies," "trump," and "disgusting" as having a strong negative impact on the prediction, indicating their association with cyberbullying behavior. Conversely, words like "defended" and "supporting" have a positive impact, suggesting a lower likelihood of cyberbullying.

Overall, the SHAP text plot provides valuable insights into how specific words and tokens influence the model's predictions, aiding in the interpretation of its decision-making process. In the future, to combat issues with "the perturbation-based explanation process when dealing with textual data," we could implement TransSHAP in hopes of gaining a better understanding of how SHAP can be applied to understand the BERT model (Kokalj).

REFERENCES

Fine-Tune a Pretrained Model, huggingface.co/docs/transformers/en/training. Accessed 1 May 2024.

Horev, Rani. "Bert Explained: State of the Art Language Model for NLP." Medium, Towards Data Science, 17 Nov. 2018, towardsdatascience. com/bert-explained-state-of-the-art-language-model-for-nlp-f8b21a9b6270.

Kokalj, Enja, et al. "Bert Meets Shapley: Extending Shap Explanations To ..." BERT Meets Shapley: Extending SHAP Explanations to Transformer-Based Classifiers, aclanthology.org/2021.hackashop-1.3.pdf. Accessed 30 Apr. 2024.

Kostadinov, Simeon. "Understanding GRU Networks." Medium, Towards Data Science, 10 Nov. 2019, towardsdatascience.com/understanding-gru-networks-2ef37df6c9be.

Or, Dr Barak. "The Exploding and Vanishing Gradients Problem in Time Series." Medium, MetaOr Artificial Intelligence, 14 Dec. 2023, medium.com/metaor-artificial-intelligence/the-exploding-and-vanishing-gradients-problem-in-time-series-6b87d558d22.

Received 3 May 2024