STOR565 Final Report:

Enhancing Online Safety with Abusive Speech Detection

Team MLFs: Sathvik Chatta, Lillian Hurban, Elizabeth Singletary, Grace Sosa, Rujula Yete

Abstract

In today's digital age, children are increasingly exposed to social media content at young ages, despite age restrictions intended to protect them from harmful material. This project aims to develop a Machine Learning model to classify textual data into abusive and non-abusive categories. Given that text is a primary form of interaction on most social media platforms, including those commonly used by children, such as YouTube, TikTok, and Instagram, there is a significant risk of exposure to damaging content.

We plan to create a model that can automate language classification as abusive or non-abusive, thereby creating safer online spaces for children. With the vast amount of textual data on social media platforms, human moderators can only review a limited amount of content. This model could assist in automating this process, potentially leading to the development of tools like a Chrome extension that filters out abusive speech, making online experiences safer and more enjoyable for young users.

1. Training Dataset

1.1 Original Dataset Source & Context

Our 24,783-entry dataset was originally compiled by Cornell University researchers for the purpose of training machine learning models to differentiate between hate speech and offensive language[1]. Using Twitter API to take a random sample of tweets containing terms from an already existing offensive language lexicon[2], the researchers hired human workers to manually label the tweets, with each tweet being sorted into one of three categories matching its respective “class” label in the dataset:

- **0: Hate speech** (1,430 instances)
- **1: Offensive but not hate speech** (19,190 instances)
- **2: Neither offensive nor hate speech** (4,163 instances)

Given that the purpose of this project is to create infrastructure that will assist in filtering out social media content that is unsuitable for children and young teenagers, the distinction between non-offensive language, offensive language, and hate speech is too specific to be relevant. Thus, for our project, we chose to focus on building models that will classify unsuitable content by generalizing instances of offensive language and hate speech from this dataset into a new category: Abusive language (as hate speech is offensive by definition). After applying this change to our data, the new class labels for each instance were as follows:

- **0: Non-Abusive** (4,163 instances)
- **1: Abusive** (20,620 instances)

1.2 Data Cleaning & Preprocessing

In a Natural Language Processing project, text cleaning is crucial for accurate analysis. Removing capitalization, punctuation, and numbers, are just a few things recommended in the data cleaning process. Considering the source and context of our data (scraped from Twitter), we took a few steps to remove excess information that is is either not relevant to this project or that the computer will not understand:

- Converted all text to lowercase
- Removed URLs/links
- Removed Twitter user handles (any contiguous string after an @ symbol)
- Removed stopwords ('the, 'is', 'and', etc..)
- Removed special characters
- Removed "rt" (indicating a retweet)
- Removed extra whitespace
- Removed non-English words/characters
- Converted contractions (isn't becomes is not, etc.)
- Removed numbers

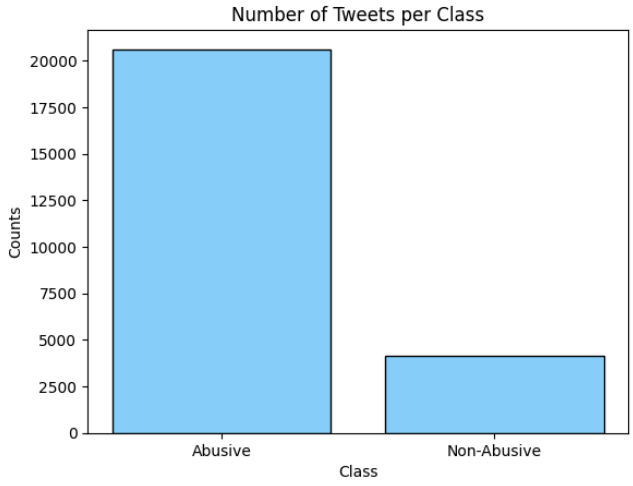
An example of a tweet from our dataset before and after processing is shown below.


```
worldseriesgame hunter pence annoying red sox
player shave fool take vyvanse yankees
```

2. Exploratory Data Analysis

Before building our classification models, we conducted some preliminary analysis to better understand the key characteristics of our dataset. Throughout our EDA process, we discovered a few major issues with our dataset that we will discuss in detail and provide solutions for in this section.

2.1 Class Frequency Analysis

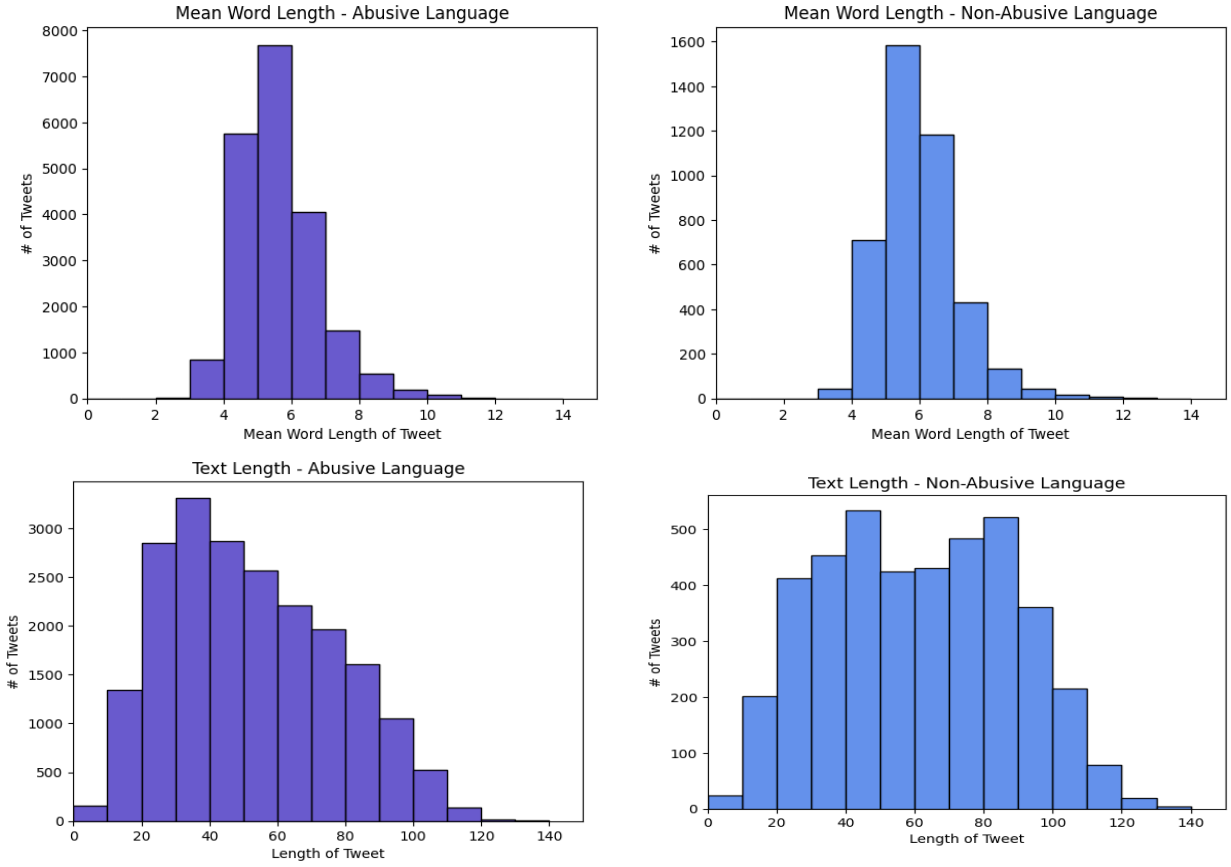


A major issue that immediately arose with our Twitter dataset was the presence of a high class imbalance. When left unaddressed, this class imbalance heavily skewed the results of our models, causing a noticeable pattern in which the models tended to classify a post as Abusive (the majority class) most of the time. Despite our intent to weigh False Negatives (mistakenly classifying Abusive content as Non-Abusive) as far more costly than a False Positive (this will be explained more in a later section), other evaluation metrics showed that

the class imbalance was detrimental to the performance of all of our models in the long run, and solutions to this problem needed to be explored.

2.2 Tweet & Word Length Analysis

Next, we examined the length of the tweets between the classes by character and by word (after cleaning). Histograms were compiled showing the mean number of characters and words in each tweet by category, showing that Abusive tweets may be shorter overall. We took into account that the character limit for a tweet during the time period in which the data was compiled was 140 characters.



Upon further reflection of this length analysis, another glaring issue became apparent: Twitter data is, by nature, extremely short. This problem became even more obvious later on, when our models that were trained on it struggled to accurately classify text longer than a few words, or text that was sourced from social media platforms other than Twitter.

2.3 Solutions to EDA Concerns

To combat the issues outlined in the previous parts, we decided to bring in more data. We considered a few other options before doing this — synthetically resampling instances from the minority class (A.K.A. SMOTE) did an okay job at resolving the poor model performance but increased the lack of diversity in various features of the corpus, while using different evaluation metrics specifically designed to account for class imbalance (Cohen's Kappa + Geometric Mean) again improved the performance but at the cost of result interpretability. In the end, we chose to source two additional datasets[3, 4] that use textual data from a combination of various social media platforms, including Twitter, Reddit, and YouTube comments. Like our original dataset, both were

manually labeled by humans as either Abusive or Non-Abusive, and were cleaned appropriately with consideration to their source (non-Twitter data did not need “rt” removed for example).

2.4 Final Ensemble Dataset

Our final combined dataset contains 48,762 posts from various social media platforms, with 18,927 of these being classified as Non-Abusive while the remaining 29,835 are classified as Abusive. The addition of extra data combatted the class imbalance problem while also enriching the diversity of the features in the corpus with text from a wider range of sources and contexts, which will enhance our models’ abilities to generalize Abusive language detection across different social media environments.

3. Feature Extraction with Data Vectorization

As we have learned throughout this course, our computers cannot really comprehend language in the same ways that we as humans experience it, so it is up to us to translate it for them in a way that they can understand. Hence why before building any models, it is critical for us to vectorize our textual data into numerical data, which allows the formulas used in classification problems to work properly. There are several techniques that are used for word embedding - the process by which we represent text mathematically - but for our project, we have chosen to compare two methods of word embedding for our models: TF-IDF and Word2Vec.

3.1 TF-IDF Vectorizer

TF-IDF (Term Frequency-Inverse Document Frequency) is a vectorization method that computes a word’s significance based on its frequency within a document (tweet, YouTube comment, etc.) versus its frequency amongst all documents in a corpus. A vector is created for each document, with lengths equal to the total number of unique words in the dataset. Each element is a TF-IDF score per word. With this calculation, words that are more common have lower scores, and are thus less significant to a model’s calculation. The final result is a set of n vectors, equal to the number of rows in the dataset.

$$\mathrm{TF}(t,d) = \frac{\text{number of times term } t \text{ appears in document } d}{\text{total number of terms in document } d}$$

$$\mathrm{IDF}(t) = \log\left(\frac{\text{total number of documents}}{1 + \text{total number of documents containing term } t}\right)$$

$$\text{TF-IDF} = \mathrm{TF}(t,d) \times \mathrm{IDF}(t)$$

3.2 Word2Vec Embeddings

Compared to TF-IDF, Word2Vec is an unsupervised technique that mathematically computes the semantic closeness of words in a dataset. A vector is created per word, each element being a measure of how often the word occurs with another word in the dataset, and the result is a vector space of word embeddings. The result is a set of n vectors of a predetermined length, where n is the number of unique words in the dataset. To generate these vectors, we integrated GloVe vectors (Global Vectors for Word Representation) and Word2Vec embeddings. We use a pre-trained LLM[5] specifically trained on social media data, giving us an initial set of weights. GloVe vectors are then transformed to Word2Vec format for compatibility, and text is converted into numerical data by averaging Word2Vec vectors of constituent words.

The key difference between TF-IDF and Word2Vec is what the vectors in each model actually mean— in TF-IDF, a vector is created per observation, and classification models compare the TF-IDF scores in the vector for categorizing. In Word2Vec, a vector is created per word, and

classification models can compare the occurrences of the word in each observation with similarity measures to other words provided in the vector to categorize observations.

4. Model Selection

4.1 Logistic Regression

Logistic regression is a supervised classification technique that acts as an extension of linear regression by using a sigmoid function to return a probability value for each observation. This is the probability of said observation belonging to a specific class. If the predicted probability of an observation being Abusive or Non-Abusive is greater than 0.5, then the observation is predicted to be in that class. Logistic regression assumes that the observations are independent of each other, and that there is a linear relationship between the predictor variables and the log odds of the result. Despite its simplicity, logistic regression is computationally efficient and works well with sparse data, making it particularly suitable for our large dataset after being vectorized using TF-IDF.

4.2 Naive Bayes

Naive Bayes is a supervised learning technique for binary classification. It is commonly used in text data classification, such as for sentiment analysis and spam detection. The model is derived from Bayes' theorem, which finds the probability of a data point being in a class based on conditional probabilities from the features. It is called 'naive' because it assumes conditional independence between features. In Naive Bayes, each feature is weighted equally, and assumes a normal distribution between continuous features and a multinomial distribution between discrete features. For our model, we vectorized the data using TF-IDF, and classified as Abusive and Non-Abusive depending on if the predicted probability was greater than 0.5 for a class. Naive Bayes is beneficial for its fast computation and is known to be effective for high-dimensional cases.

4.3 K-Nearest Neighbors

K-Nearest Neighbors (KNN) is a supervised learning algorithm used for classification tasks, where it assigns a class label to a data point based on the majority class of its k nearest neighbors. The choice of k is a crucial hyperparameter that significantly influences the model's performance and must be carefully fine-tuned based on the characteristics of the dataset.

4.4 SVM

Support Vector Machines (SVMs) are a powerful class of supervised learning algorithms used for classification and regression tasks. They work by finding the hyperplane that best separates different classes in the feature space. SVMs are memory efficient as they only need to store the support vectors, which are the data points closest to the hyperplane and crucial for defining it. One of their key advantages is their ability to generalize well to unseen data, thanks to their margin maximization objective, which also helps in reducing overfitting. SVMs are also interpretable, as the decision boundary is a linear combination of input features. However, SVMs may struggle with datasets that are not linearly separable, requiring the use of kernel tricks to map the data into a higher-dimensional space where linear separation is possible.

4.5 Multilayer Perceptron (FFNN)

A Multilayer Perceptron (MLP), also known as a Fully Connected Feed Forward Neural Network (FFNN), is a type of artificial neural network consisting of multiple layers of neurons, with each neuron in one layer is connected to every neuron in the next layer. Leveraging the use of Word2Vec and a pretrained LLM (GloVe model), the Word2Vec embeddings were fed into the network as an input layer. This combination works particularly well, as the multilayer perceptron captures complex semantic patterns between word vectors. Using Word2Vec word embeddings, each neuron in the input layer corresponds to a dimension in your generated word vectors. The core of the MLP are the hidden layers, where each hidden layer transforms the inputs from the previous layer using a weighted sum followed by an activation function. In this context, we utilized hidden ReLU layers, with a Sigmoid activation function as the output layer.

5. Validation and Test Data

We settled on a random 70/10/20 split for training, validation, and testing sets respectively. The 70% of the data allocated to training was used to fit the models, then tuned using the validation data, and subsequently evaluated for performance on the remaining 20% of the data saved for testing.

10% of the data randomly split into a validation set was used for fine-tuning the hyperparameters of our models. Using GridSearchCV and taking into account the limitations of our hardware, we found the optimal parameters for our models to be:

```
Best parameters for SVM: {'C': 0.1, 'kernel': linear, 'dual': True}
Best parameters for Naive Bayes: {'alpha': 0.1}
Best parameters for KNN: {'n_neighbors': 10, 'p': 2}
Best parameters for Logistic Regression: {'C': 10}
```

6. Model Performance Evaluation

6.1 Chosen Metrics

To evaluate the performance of our models, we chose to focus on four specific metrics: AUC, Recall, F1-score, and Accuracy. Given that the purpose of this project was to develop the infrastructure needed to filter out abusive content from children's social media feeds, we chose to specifically focus on a model's Recall (on 1), that is, the model's ability to correctly identify True Positives, which in this context are the instances of abusive text.

6.2 Recall

We decided it was better practice for our models to mistakenly flag non-abusive content as abusive (False Positive), as opposed to not flagging actual abusive content (False Negatives). This addresses the concern of minimizing any exposure to sensitive content, the main goal in the context of this project. Thus, maximizing the Recall minimizes the risk of abusive content being seen due to not being flagged.

6.3 F-1 Score

F-1 Score is the harmonic mean of Recall and Precision, the F1-Score can help to reflect a balance of completeness and accuracy of abusive speech detection. A model with a high F-1 score

minimizes false positive and false negative predictions. This metric is therefore the best indicator of overall model performance.

6.4 Accuracy

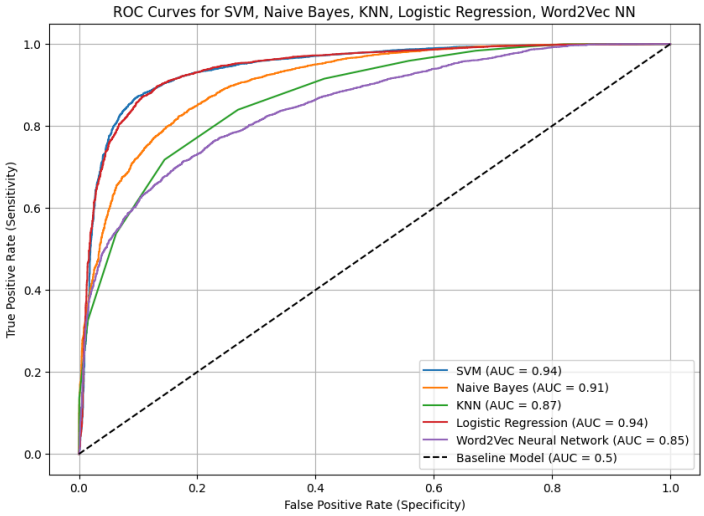
Although accuracy on its own is not the most reliable metric for gauging a model's utility, it is highly interpretable and familiar to most people, as well as being good for establishing a baseline metric for understanding general model performance.

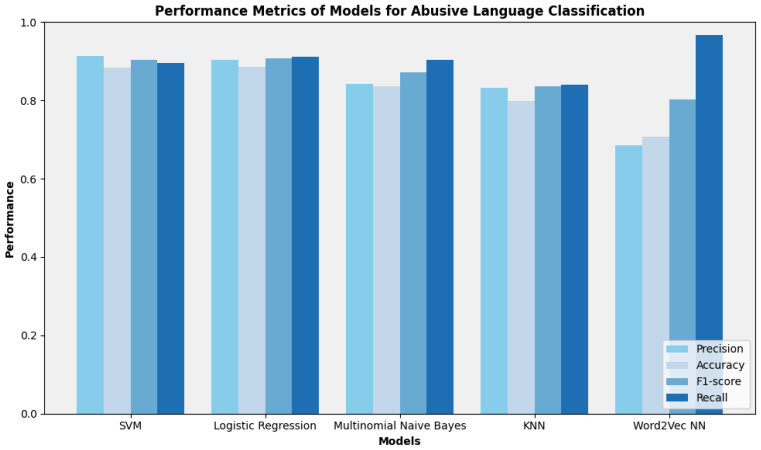
6.5 AUC

Lastly, the AUC addresses a probabilistic interpretation of model performance. It is highly interpretable, a good measure for overall general classifier performance, and is great for visualization.

Overall Model Performance

We explored utilizing Support Vector Machines (SVMs), Logistic Regression, K-Nearest Neighbors/Clustering, Naive Bayes, and a Multilayer Perceptron (FFNN). Below are the AUC scores, as well as classification report with Precision, Recall, Accuracy, and F1-Score. Overall, Linear SVM and Logistic Regression had the highest AUC of 0.94. Logistic Regression and SVM had the highest F1-Score of around 0.90. And the Multilayer Perceptron had the highest Recall of 0.97.





7.1 Best Overall Performance: Linear SVM

Overall Linear SVM was shown to have the best performance, in AUC, F1-Score, and Recall. It is important to note, Logistic Regression had similar performance metrics, but we found upon testing on several datasets, and upon the Twitter datasets, Youtube and Reddit data separately, Linear SVM consistently performed slightly better than Logistic Regression. Additionally, SVMs

have a high propensity to do well in the context of NLP and text classification tasks, and more opportunities for model and hyperparameter tuning, in comparison to Logistic Regression.

SVM

	precision	recall	f1-score	support
0	0.8393	0.8664	0.8526	3780.0000
1	0.9137	0.8950	0.9043	5973.0000
accuracy	0.8839	0.8839	0.8839	0.8839

To improve our SVM model, we explored SMOTE resampling to account for initial unbalanced Twitter data, where approximately 80% of classified tweets were flagged as abusive. In the original dataset, SMOTE resampling seemed to show some improvement in F1-Score and Precision. However, after adjusting and combining our Twitter dataset with Reddit and Youtube comment data, our dataset was more balanced and SMOTE resampling showed no significant improvement. To verify this, we performed a T-test on the F1-Scores of our Linear SVM with and without SMOTE resampling. Our p-value was 0.52, which is not statistically significant at the alpha = .05 level.

Additionally, we explored various C values (0.1, 1, 2, 5, 10, 100), Linear, Polynomial, and RBF, different polynomial degrees ('3', '4', '5'), and 'auto' and 'scale' gamma. After utilizing sklearn's GridSearchCV, and RandomizedSearchCV over k = 5 folds, it was still found that an SVM with C = 1, and Linear Kernel performed the best.

7.2 High Potential: Word2Vec & Multilayer Perceptron

Word2Vec Neural Network

	precision	recall	f1-score	support
0	0.8518	0.2966	0.4400	3780.0000
1	0.6848	0.9674	0.8019	5973.0000
accuracy	0.7074	0.7074	0.7074	0.7074

Although our Multilayer Perceptron (MLP), with Word2Vec embeddings obtained from pretrained GloVe vectors as the input, did not achieve top performance all around, with a much lower precision, and accuracy, our MLP significantly outperformed the other models in terms of recall, with a recall of 0.97. In this context, this could debatably be the most important metric, if our goal is to flag all abusive content on social media. However, since the accuracy is sitting at approximately 0.71, and precision slightly below that, we did not pick this as our best performing model.

Here we see our use of an Adam optimizer, binary cross entropy loss, 50 epochs, two hidden layers, and a Sigmoid activation function for the output:

```
# Define the model architecture
model = Sequential([
    Dense(128, activation='relu', input_dim=x_train_vectors.shape[1]),
    Dropout(0.5),
    Dense(64, activation='relu'),
    Dropout(0.5),
    Dense(1, activation='sigmoid')
])

model.compile(optimizer=Adam(), loss='binary_crossentropy', metrics=['accuracy'])
```

```
model.fit(
    x_train_vectors, y_train,
    epochs=50,
    batch_size=32,
    validation_data=(x_test_vectors, y_test),
    validation_split=0.2,
    class_weight=class_weights_dict,
    callbacks=[early_stopping, lr_scheduler]
)
```

As for improving our MLP, we utilized sklearn's compute_class_weight function to add class_weights, 0: 0.78 and 1: 1.32, emphasizing more weight on 1 since we are focusing on flagging as much abusive content as possible. However, exploring other class weights can help to optimize the balance between high recall on 1, and decent precision, accuracy, and F-1. Additionally, we utilized keras.callbacks' EarlyStopping, to prevent overfitting, as well as a LearningRate Scheduler to optimize learning rate, calculating larging learning rates in early epochs, and smaller learning rates in later stages.

We included the MLP as an honorable mention, as it offers extensive customization potential, where it can be fine tuned to excel in specific performance metrics depending on desired outcomes and the context.

Future Goals and Improvement

8.1 Model Improvement & Exploration

To enhance the robustness of our model in detecting abusive speech, we propose training the model using labeled data sourced from a wider array of social media platforms that we were unable to find for this project. By incorporating diverse data sources, the model can learn more effectively from a variety of linguistic styles, cultural contexts, and forms of abusive language present across different platforms. This approach will enable the model to generalize better and identify abusive content more accurately across multiple platforms, thereby enhancing its overall performance and utility.

In addition to expanding the training data, we suggest further exploration of pre-trained LLMs, instead of just the GloVe vector Twitter model for generalization for all social media. Some potential pretrained large language models we could use are BERT (Bidirectional Encoder Representations from Transformers) and RoBERTa (Robustly Optimized BERT Approach). An additional avenue for enhancement, after utilizing different LLMs, is fine-tuning the feedforward neural network used in the current model, more specifically, exploring more hidden layers (3, 4, 5, etc.). By adjusting the network's hyperparameters, hidden layers, and architecture, we can potentially improve its ability to capture complex patterns in the data and enhance its discriminatory power in identifying abusive speech.

8.2 Potential Applications

Implementing the abusive speech binary classification algorithm into applications and browser extensions involves several key considerations to ensure effective functionality and user experience. One approach is to develop a standalone application that users can install on their devices, such as a mobile app or desktop software. This application can integrate the algorithm to analyze text input in real-time, providing users with immediate feedback on whether the text contains abusive language. Another approach is to develop a browser extension that can be installed on popular web browsers. The extension can add functionality to text fields in web pages, allowing users to analyze text for abusive language before submitting comments or posts online. This integration can help users avoid inadvertently posting abusive content and promote more respectful online interactions.

An application or browser extension implementing abusive speech binary classification can play a vital role in safeguarding children's online experiences. By integrating the algorithm into parental control software or child-friendly browsers, parents can monitor and filter the content their children are exposed to. The algorithm can scan text on websites, social media platforms, or in chat applications, flagging potentially abusive language or content and alerting parents to intervene.

Furthermore, the application or extension can be customized to suit the age and maturity level of the child, ensuring that the feedback and guidance provided are age-appropriate and effective. This tailored approach can help children develop healthy digital habits from a young age and navigate the online world with confidence and awareness.

Overall, an application or browser extension implementing abusive speech binary classification that prioritizes user experience, privacy, and ongoing refinement of the algorithm can be a valuable tool in protecting children from harmful online content and promoting positive online interactions. By combining technology with education and parental guidance, these tools can help create a safer and more respectful online environment for children to explore and learn.

Conclusion

Overall, the Linear SVM model performed the best in terms of overall, followed closely by Logistic Regression, while our Multilayer Perceptron had the highest recall, which is useful for the context of flagging abusive content. For future improvements, we plan to source our training data from a variety of different social media platforms, as well as utilizing techniques like modifying the number of hidden layers in our neural network or using different pre-trained LLMs. Future expansions could include developing user-friendly applications such as a Chrome extension for real-world deployment.

Works Cited

[1] Davidson, T., Warmsley, D., Macy, M., & Weber, I. (2017). Automated Hate Speech Detection and the Problem of Offensive Language. *Proceedings of the International AAAI Conference on Web and Social Media*, *11*(1), 512–515. https://doi.org/10.1609/icwsm.v11i1.14955

[2] Hatebase LLC. *Hate Speech Lexicon*. Hatebase. https://hatebase.org/

[3] Ashraf, N., Zubiaga, A., & Gelbukh, A. (2021). Abusive language detection in YouTube comments leveraging replies as conversational context. *PeerJ Computer Science*, *7*. https://doi.org/10.7717/peerj-cs.742

[4] Kennedy, C., Bacon, G., & Sahn, A. (2020). *Constructing Interval Variables via Faceted Rasch Measurement and Multitask Deep Learning: A Hate Speech Application*, https://doi.org/10.48550/arXiv.2009.10277