Text Classification For Identifying Hate Speech

Final Capstone Report

By Kohisha Aruganti

Table of Contents:

Introduction

In recent years, online platforms have become an important means of communication for the people to express their opinions and share their views. This type of communication has brought a lot of benefits by creating diverse perspectives and enabling the exchange of knowledge. However, this also has also led to widespread propagation of hate speech, which has had various effects on individuals and communities. Hate Speech has the potential to inflict emotional harm and create boundaries and also generate violence among different communities. The rise of hate speech on various online platforms has created an urgency to address this issue effectively. Hate speech classification in text has gained popularity using Natural Language Techniques with the widespread use of social media. To solve these emerging problems in social media platforms this study provides different feature embedding techniques, along with machine learning, deep learning models and LIME algorithm to explain model predictions. So, the aim of this project is to develop an effective text classification system to identify hate speech in text data. The predictions generated from the models were tested manually and throughout the evaluation we observed models performed effectively in predicting and explaining the results.

Aim and objectives

The purpose of this study is to examine the effectiveness of identifying text classification on hate speech and look at potential areas of improvement. To work on it, the following objectives have been understood.

- Crucial step is to collect diverse data sets from various sources that contain instances of hate speech and not hate speech.
- To understand the insights often faced when working with hate speech that include issues related to context, sarcasm and evolving language trends.
- Exploring NLP (Natural Language processing) techniques for preprocessing the data and for feature extraction.
- Incorporate various machine learning models and advance deep learning models for classifying hate speech considering factors such as model interpretability.
- To produce results using streamlit to analyze the prediction of each model.

Data collection and preparation

This process involves the aggregation of text data from a wide array of platforms like twitter data, fox news comments, comments from various twitter ids and civil comments dataset. The first dataset is gathered from this paper (Gao & Huang, 2017). In this authors Lei and Huang, created the dataset containing around 1528 user comments which were taken from 10 discussion threads from 10 news articles. This dataset preserves the context information and all the comments in the news articles are covered without any bias. Out of 1528 comments, 438 are labeled as hateful and for the annotations of the comments the author has followed the guidelines similar to this paper (Chikashi, 2016). The annotations are moderated by the individuals who had at least an undergraduate degree and were familiar with the concept of judging text passages for different types of annotation tasks and requirements. Before taking on the actual moderation task, they were trained in order to familiarize themselves with text judgment guidelines.

The second dataset was created by the ConversationalAI team from Google and Jigsaw (Lacy, 2022). Their primary goal was to study negative online behaviors, like toxic comments and classify the type of toxic comments into various categories such as toxic, severe toxic, obscene, threat, insult. The data has been sourced from the comments of the Wikipedia talk page edits. It contains text samples with a sizeof more than 160K comments which are classified into various hate types and it also has the comments about 'No hate'. The labels for this dataset have been done by human raters.

The third dataset is sourced from this paper (Smaranda & Preslav, 2022), where they have gathered data from the publicly available Civil Comments dataset. The label to these comments has been done by the crowd annotators to highlight the spans that constitute "anything that is rude, disrespectful, or unreasonable that would make someone want to leave a conversation" (Smaranda & Preslav, 2022). Along with the type of hate, the comments are also labeled with the span from the comments where there is hate. Initially from more than million comments, the data has been filtered to 30k comments and used human raters to mark the toxic word sequences of each post by highlighting each toxic span on their screen. The raters were selected from the group of most experienced contributors. The raters were asked to mark the toxic word sequences of each post by highlighting each toxic span on their screen.

The fourth dataset has been sourced from hugging face website (Ona et al., 2018). This data has been sourced from Stormfront, a white supremacist forum where random posts are sampled and split into sentences. These sentences are labeled manually based on certain guidelines for the annotators. They categorized hate if the sentences contain deliberate attack or directed towards a specific group of people or motivated by aspects of the group’s identity. The content from Stormfront was extracted using web-scraping techniques and the data contained from diverse topics and the total number of posts were grouped and divided into multiple batches for manual annotations. In total there are close to 10k sentences which are labeled as ‘Hate’ or ‘Non Hate’.

Finally, we have collected data from multiple sources which creates diversity and creates balances which ensures that each class - hate speech and non-hate speech - is adequately represented. Each dataset is represented in various forms such as json and csv files. There are multiple categories in hate for some dataset such as toxic, insult, obscene and in the data preparation stage we need to consolidate all these datasets and create one final dataset. After consolidating the total samples will be around 180k.

Data Preprocessing

Several research studies have explained that using text preprocessing makes better classification results (Shaikh s. & Muhammad, n.d.). In this project, different preprocessing techniques are used to filter noisy and non informative features from the text. Removing these features will not affect the further analysis but instead it helps in increasing execution time. In the preprocessing, text has been converted into lower cases and URLs, punctuations, stopwords, numbers, emoji have been removed using pattern matching techniques from the collected data. Alternatively, from the preprocessed data; tokenization, stemming and lemmatization have been performed. Tokenization converts each piece of text into tokens or words. Stemming converts words to their base or root form. Lemmatization is also similar to stemming but it converts words to its root or dictionary form considering the context and part of speech of the word.

Word Embeddings

Machine learning or deep learning algorithms cannot understand the classification rules of text. Only numerical data can be fed as input to models. Word embeddings are a crucial step when dealing with text data because this extracts features from the raw text and converts them into numerical form. It converts in such a way that computers can understand the word and relationships between the words. For this project, three different embedding techniques have been performed namely, TFIDF, Word2Vec and GloVe.

Word2Vec(Word to Vector)

Word2Vec converts words to vectors capturing the semantic relationship between the words. Word2Vec predicts the target words in the given context; it uses CBOW(continuous bag of words) to learn the word embeddings. Predicted representations are used as features for training and evaluating machine learning models.

TFIDF(Term Frequency-Inverse Document Frequency)

These embeddings excel at finding distinctive features(words) that differentiate from the entire dataset. Term frequency measures how often a word occurs in the dataset. Inverse document frequency measures the importance of the word across the collection of dataset. This enables machine learning models to prioritize words that are uniquely important to each piece of text, contributing to more accurate analysis.

GloVe(Global Vectors for Word Representation)

Glove embeddings are pre-trained on a large corpus(Wikipedia,Books) and captures the semantic relationships. It uses matrix to represent the word-to-word co-occurrence of the corpus. Because of using pre-trained embeddings, it helps to enhance the features, patterns and semantic meaning across the corpus. The difference between GloVe and Word2Vec is that GloVe is a count-based model while Word2Vec is a prediction-based model.

Data Splitting

I have used the 75-25 ratio to split the preprocessed data i,e: 75% for Training data and 25% for Testing data. Training data is used to train the classification models. Test data is used to evaluate the classification models.

Model Selection

According to the “no free lunch theorem” (Zafar, 2021), a single model cannot perform better on all kinds of datasets. Hence, we need to apply different models to the dataset and observe which one provided better results. Therefore, four different machine learning models along with two deep learning models have been used for text classification.

Machine Learning Models

Support Vector Machine(SVM)

SVM is a powerful supervised learning algorithm which is commonly used for text classification. It’s an effective model for hate speech detection because SVM excel at finding hyperplanes that discriminate between different classes and divide hate speech from non-hate speech in a high-dimensional space. This makes SVM robust when handling complex patterns within textual data.

Naive Bayes

Naive Bayes is a probabilistic classification algorithm that analyzes and categorizes the text data. It has the ability to handle large volumes of text data which makes this algorithm suitable for this project. It can effectively calculate the probability of a text and categorizes it if the text belongs to either hate speech or non-hate speech.

Gradient Boosting Classifier

This model is effective for capturing complex relationships within the data and improves the model’s ability to discern hate speech from non-hate speech.

Decision Tree

Decision trees are well known for their interpretability which makes it easier to understand decision-making processes. For text classification interpretability is crucial for understanding why certain texts are classified as hate speech. Decision trees capture complex relationships which help the model to adapt to various nuances within the text.

Deep Learning Models

CNN-LSTM(Convolutional Neural Networks and Long Short-Term Memory)

Combination of Convolutional Neural Networks and Long Short-Term Memory can provide valuable insights and capture semantic relationships. This combination helps the model to capture both local features and long term dependencies or temporary dependencies within the dataset.

BERT(Bidirectional Encoder Representation from Transformers)

BERT is a state-of-art NLP model that applies bidirectional training to language modeling tasks. Bidirectional training provides deeper insights into the context of the language. Unlike directional models, which reads text either from left to right or right to left, BERT reads entire text of words simultaneously allowing the model to learn the context of the word based on its surroundings.

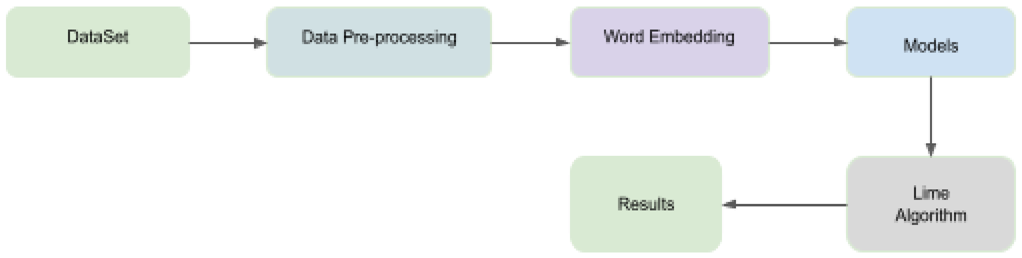

Fig1: A pipeline of proposed Methodology

LIME Algorithm(Local Interpretable Model-agnostic Explanations)

LIME is a technique used to enhance the interpretability of “Machine learning” or “Black Box models”. Black box models are complex models and understanding their inner workings can be challenging so, lime bridge this gap and provide individual predictions for the modes. The model shows the important features(word) that contribute to the prediction. It works by tweaking the input slightly and observing changes in the predictions. These tweaked data points are assigned weights based on their proximity to the original data points. Subsequently, any machine learning or deep learning models are trained on the dataset containing these variations along with their assigned weights. Each of the original data points can then be explained with the newly trained explanation model (Johannes et al., 2023). In this text classification we are generalizing between hate and non-hate. LIME takes a specific text instance for explanations. Any variations in the results can help to understand and making changes in the training data can lead to better results.

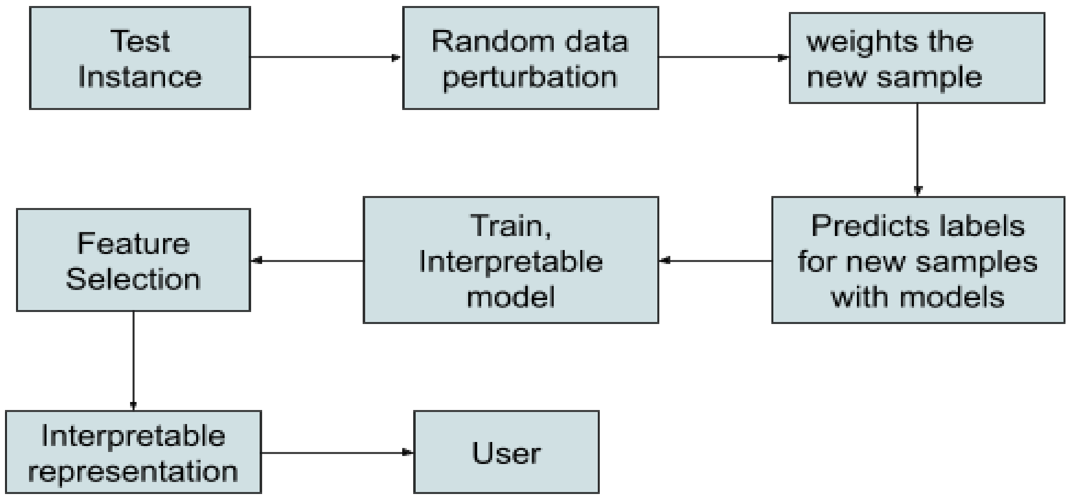
Test Instance
Random data perturbation
weights the new sample
Predicts labels for new samples with models
Train, Interpretable model
Feature Selection
Interpretable representation
User

Fig 2 : Flowchart of LIME methodology (Zafar, 2021)

The idea of training the LIME mode:

- Choose the instance for which the user seeks the explanations for the model prediction.
- Make random changes to the dataset and obtain predictions for these new points
- Assign weights to the new points based on proximity
- Train interpretable models by taking into account variations and adding the assigned weights.
- The predictions are explained with their interpretability. (Smaranda & Preslav, 2022)

Model Evaluation

The model categorizes the text into one of two classes: “hate” or “not hate” based on the patterns it learned during the training phase. The machine learning model performance is evaluated using a confusion matrix by calculating true negatives(TN), false positives(FP), false negatives(FN), true positives(TP). Alongside, performance metrics like precision, recall, F1 score and accuracy are used to assess the performance of the constructed model. Then integrate models to the lime algorithm. (Seliya et al., 2019)

1. Precision: Precision is the ratio of correctly predicted positives(hate) to the total predicted positives. It measures the accuracy of the model when it correctly predicts a text as hate.(Jayaswal, 2020)

$$precision \; = \; \frac{TP}{(TP + FP)}$$

2. Recall: Recall is the ratio of correctly predicted positives(hate) to the actual positives. It identifies all instances of hate speech.(Jayaswal, 2020)

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

3. F1 Score: It is the harmonic mean of precision and recall. It considers both false positives and false negatives.(Jayaswal, 2020)

$$F1\,Score \; = \; \frac{2 \times (precision \times recall)}{(precision + recall)}$$

4. Accuracy: Accuracy is the ratio of correctly predicted(hate and not hate).

$$Accuracy \; = \frac{(TP + TN)}{TP + FP + TN + FN}$$

	Predicted No	Predicted Yes
Actual No	TN	FP
Actual Yes	FN	TP

Table I: Confusion matrix

The deep learning model performance is evaluated by using epochs for training models. Each epoch represents one complete pass through of the entire training data. By training models over multiple epochs improves the model performance and enables it to iteratively learn from the data. Each epoch's accuracy helps to understand convergence and potential overfitting. For the BERT model GloVe embedding is used as feature vectors for classification. Initially it converts input sequences to tokens. The token embedding vectors are created by adding segments, tokens together. For sentence classification uses “CLS” for classification, CLS indicates unique tokens to indicate the starting position of the classification task. For training testing the BERT model the model parameters are Learning rate = 2e-5, Number training epoch= 2 and batch size = 32.

CNN-LSTM; Embedding layer starts with an input dimension of 50 words and output shape(None, 50, 50) ; this indicates that 50 words are represented by vectors of length 50 Dense layer. Three convolutional layers with 64 filters are used with 2 x 2 max pooling layers to reduce the dimensionality. Three fully connected layers with dropout to prevent overfitting. Four LSTM layers each with 256 filters used to capture patterns and temporal dependencies. Dense vector has 3 neurons followed by an activation layer.
In total the model has been trained on a total of 11,045,465 parameters indicating a complex architecture capable of capturing both spatial and temporal features in the input data.

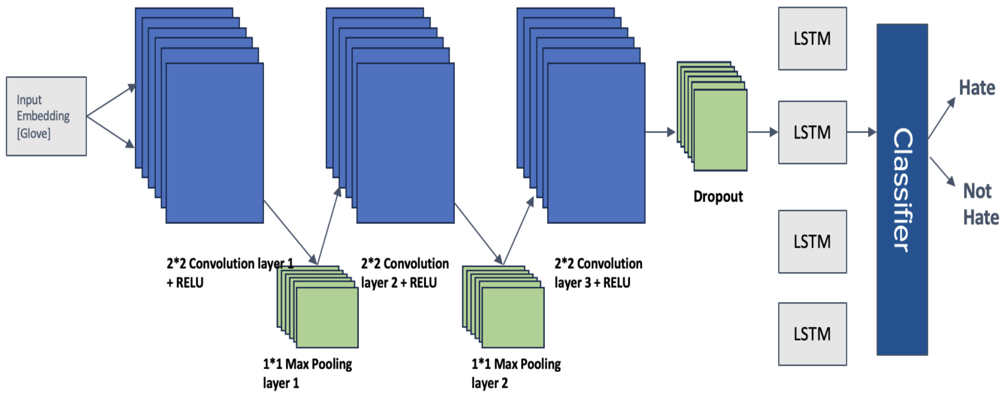

Fig 3: CNN-LSTM architecture

Results

This section explains the overall results of the models. Tables II to V show the precision, recall, F1 score and accuracy of all the 4 machine learning models respectively. Maximum and minimum values are represented in bold. The tables show the performance metrics for various features along with classification models. The lowest recall, precision, accuracy and F1-score value is 0.74 for gradient boosting classifiers using TFIDF features. The highest recall, precision, accuracy and F1-score value is 0.87 for SVM classifiers using TFIDF features. In feature representation, TFIDF performance was slightly better compared to Word2Vec. Logistic regression model was used instead of Naive Bayes classifier for Word2Vec features because Naive Bayes can not handle negative values and it is typically used for word counts. In text classification models, the SVM and Naive Bayes Classifiers performed better compared to the other two models.

Features	**Decision Tree**	**Naive Bayes/Logistic Regression**	**SVM**	**Gradient Boosting Classifier**
TFIDF	0.80	0.86/NB	**0.87**	0.74
Word2Vec	0.76	0.84/LR	0.83	0.84

Table II. Accuracy of 4 models

Features	**Decision Tree**	**Naive Bayes/Logistic Regression**	**SVM**	**Gradient Boosting Classifier**
TFIDF	0.80	0.86/NB	**0.87**	0.74
Word2Vec	0.75	0.83/LR	0.85	0.85

Table III. Recall of 4 models

Features	**Decision Tree**	**Naive Bayes/Logistic Regression**	**SVM**	**Gradient Boosting Classifier**
TFIDF	0.80	0.86/NB	**0.87**	0.74
Word2Vec	0.77	0.85/LR	0.83	0.83

Table IV. Precision of 4 models

Features	**Decision Tree**	**Naive Bayes/Logistic Regression**	**SVM**	**Gradient Boosting Classifier**
TFIDF	0.80	0.86/NB	**0.87**	0.74
Word2Vec	0.76	0.84/LR	0.84	0.84

Table V. F1 score of 4 models

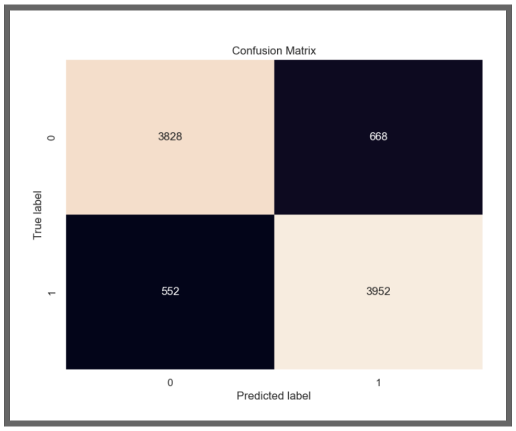

Fig 4: Confusion matrix(Feature: TFIDF, Classifier:Naive Bayes)

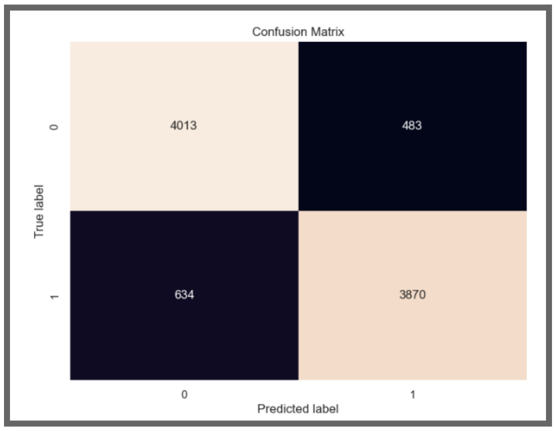

Fig 5: Confusion matrix(Feature: TFIDF, Classifier:SVM)

Furthermore, Fig 4 and Fig 5 have produced better analysis. Fig 1 shows the confusion matrix of Naive Bayes classifiers using TFIDF features. As shown in the plot, 3828 were correctly classified as hate speech, while 668 were incorrectly classified as not-hate. Moreover, there are 552 instances incorrectly classified as hate speech which were actually not-hate. However, Fig 5 shows the confusion matrix of SVM classifiers using TFIDF features. The overall performance of SVM is higher than Naive Bayes. The SVM algorithm performed better in terms of correctly classifying hate speech.

Now, talking about evaluating the results of the BERT classification model using GloVe features. We run the classification model for two epochs. The best accuracy obtained is 0.94%. We evaluated the results for CNN-LSTM model using GloVe features and ran a classification model for five epochs. The best accuracy obtained is 0.91%. We have important functions namely: training and validation. Training function trains the model on a batch of data and calculates the loss and accuracy. The validate function evaluates the model on a batch of data and calculates the loss and accuracy. We have used classifier models and then interpreted the results in LIME. The LIME produces a probability distribution for each class along with a probability score assigned to each feature for each class(hate or not hate).

Epoch	**Batch_Size**	**Learning rate**	**Training accuracy**	**Validation accuracy**
1	32	2e-5	0.92	0.92
2	32	2e-5	0.94	0.93

Table VI. Accuracy of BERT Model

Features	**CNN-LSTM**	**BERT**
GloVe	0.91	0.94

Table VII. Accuracy for two models

Results interpretation using LIME:

Lime highlights the words with different colors for each class hate and not hate. The below figure 6 shows the sentence is not hate using SVM model. In this case the actual and predicted class is also not hate. According to the word contribution, the color intensity increases with predicted probabilities. Thus, the classifier predicts this sentence as not hate which also agrees with human sense.

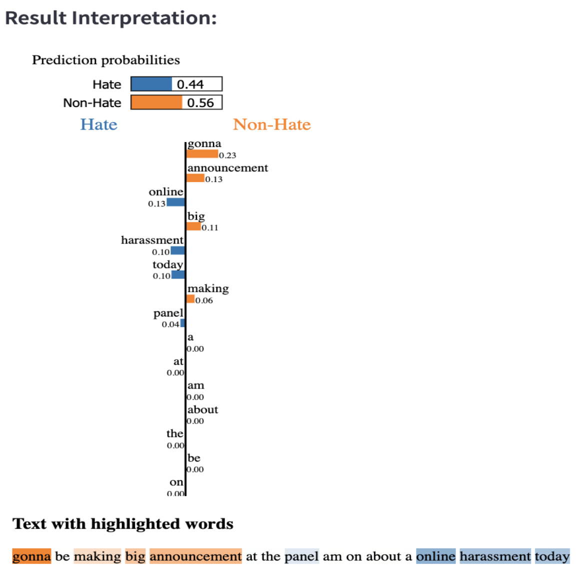

Fig 6: Results of applying LIME on a sentence

Another prediction using Naive Bayes shows that in this case the predicted class is hate. The surrounding words influence the classifier to predict it as hate. This sentence is more like disappointment by the inclusion of the term “feminazi” but when the sentence is taken as a whole it can be controversial and offensive. Thereby, we can understand that models are classifying a sentence considering the surrounding words.

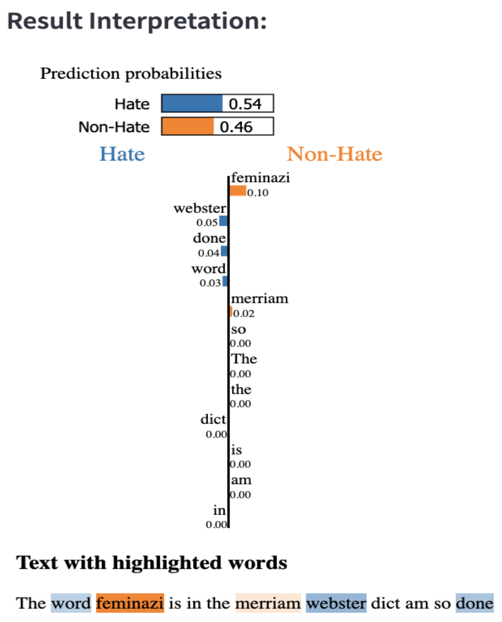

Fig 7: Results of applying LIME on a sentence

Conclusion and Future work

In this study, we focus on the impact of hate speech on social media and understand the danger they pose due to different factors. To solve this problem, we proposed methodologies to detect hate speech on the collected data and provided explanations for the modes using feature vectors. The experimental results produced using TFIDF showed better results in machine learning models and the results produced using GLoVe features showed better results compared to TFIDF. Moreover, SVM and Naive Bayes algorithm showed better results compared to decision trees and Gradient boosting classifiers in ML models. BERT and CNN-LSTM both performed better results with high accuracy compared to all machine learning models. Each of the models used have its own benefits and flaws which can provide valuable insights during predictions. This work opens numerous future works, such as improving proposed deep learning models and exploring additional models like R CNN architecture. This architecture can also be extended with different datasets that include textual or image data.

References

Chikashi, N. (2016). *Abusive Language Detection in Online User Content*. ACM Digitial Library. https://dl.acm.org/doi/abs/10.1145/2872427.2883062

Gao, L., & Huang, R. (2017, October 20). *[1710.07395] Detecting Online Hate Speech Using Context Aware Models*. arXiv. Retrieved December 19, 2023, from https://arxiv.org/abs/1710.07395

Jayaswal, V. (2020, September 13). *Performance Metrics: Confusion matrix, Precision, Recall, and F1 Score*. Towards Data Science. Retrieved December 19, 2023, from https://towardsdatascience.com/performance-metrics-confusion-matrix-precision-recall-and-f1-score-a8fe076a2262

Johannes, R., Siebers, M., & Ute, S. (2023, July 18). *,.* , - YouTube. Retrieved December 19, 2023, from https://link.springer.com/article/10.1007/s10994-021-06048-w

Lacy, S. (2022, October 2). *Toxic Comments*. . - YouTube. Retrieved December 19, 2023, from https://www.kaggle.com/competitions/jigsaw-toxic-comment-classification-challenge/overview/citation

Ona, G. d., Naiara, P., & Montse, C. (2018, September 12). *[1809.04444] Hate Speech Dataset from a White Supremacy Forum*. arXiv. Retrieved December 19, 2023, from https://arxiv.org/abs/1809.04444

Seliya, N., Khoshgoftaar, T. M., & Hulse, J. V. (2019, June 24). *A study on the relationships of classifier performance metrics*. . - YouTube. Retrieved December

19, 2023, from https://scholar.google.com/scholar?hl=en&as_sdt=0%2C9&q=Seliya%2C+N.%2C+T.M.+Khoshgoftaar%2C+and+J.+Van+Hulse.+A+study+on+the+relationships+of+classifier+performance+metrics.+in+2009+21st+IEEE+international+conference+on+tools+with+artificial+intelligence.

Shaikh s., & Muhammad, D. S. (n.d.). *Aspects Based Opinion Mining for Teacher and Course Evaluation | Sukkur IBA Journal of Computing and Mathematical Sciences*. Sukkur IBA University. Retrieved December 19, 2023, from http://sjcmss.iba-suk.edu.pk:8089/SIBAJournals/index.php/sjcms/article/view/375

Smaranda, M., & Preslav, N. (2022, may 15). *From the Detection of Toxic Spans in Online Discussions to the Analysis of Toxic-to-Civil Transfer*. ACL Anthology. Retrieved December 19, 2023, from https://aclanthology.org/2022.acl-long.259/

Zafar, M. R. (2021, June 30). *Deterministic Local Interpretable Model-Agnostic Explanations for Stable Explainability*. MDPI. Retrieved December 19, 2023, from https://www.mdpi.com/2504-4990/3/3/27