JOURNAL OF COMPUTER SCIENCES AND INFORMATICS
2024; 1(1): 17-32.
10.5455/JCSI.20240613103822

WisdomGale

RESEARCH ARTICLE

Open Access

Hate speech detector based on hybridized BERT-attention mechanism and context analyzer

Enesi Femi Aminu, Ayobami Ekundayo, Shedrack David Sarkibaka, Oluwaseun Adeniyi Ojerinde, Uchenna Cosmas Ugwuoke

Department of Computer Science, Federal University of Technology, Minna, Nigeria

ABSTRACT

Aim: The research aims to create a new hate-speech detection model by utilizing a hybridized method that captures complex contextual linkages within textual data. Hate speech remains a threat to the peaceful coexistence of humans in societies especially via open social networks in this current age, presenting grave obstacles to online safety, and promoting inclusive environments.

Methods: This is achieved by combining the advantages of bidirectional encoder representations from transformers (BERTs) attention processes with a context analyzer. Careful data augmentation was carried out utilizing back translation, which is made possible by the deep-translator library, enhancing the dataset's diversity and quantity to guarantee a comprehensive and reliable dataset.

Results: The training of the frozen BERT layer out of the two layers of the model produced a total accuracy of 0.99 on the 20th epoch by identifying the multi-labeled classes of hate speech using the Adam optimizer and softmax. Promising performance is shown by the trained model's assessment metrics, which include a macro precision of 0.79875, a macro recall of 0.71587, and a macro F1-score of 0.74825.

Conclusion: By utilizing the hybridized BERT model, damaging information can be understood holistically as it can identify not only explicit hate speech but also subtle sensitivities and underlying meanings.

ARTICLE HISTORY

Received June 13, 2024
Accepted July 15, 2024
Published July 25, 2024

KEYWORDS

Hate speech; context analyzer; BERT-attention mechanism; natural language processing (NLP); detection model

Introduction

In common parlance, "hate speech" is derogatory rhetoric directed towards a group or an individual because of basic traits such as gender, race, or religion, which has the potential to cause social instability. According to Jahan and Oussalah [1], this definition lacks legal conclusiveness and goes beyond the parameters of "incitement to discrimination, hostility, or violence," as defined by international human rights standards. Profane or abusive speech with the intention of humiliating and targeting individuals, a specific community, or groups of people is called Hate [2]. In this era of digital communications, hatred data is not only in social media comments and posts or text messages but also in voice messages and videos. Content like this causes cyberbullying, rioting, fraud, loss of respect, and even murder [3]. The lack of proper moderation provides freedom for users to perform cyberbullying or other forms of attacks. The dissemination of hate speech becomes very common in this scenario since anybody with an Internet connection can become a disseminating agent [4]. Hate speech can appear in a variety of offline and online formats, such as pictures, cartoons, memes, real or imaginary items, gestures, and symbols. It is considered to be "pejorative" or "discriminatory," exhibiting bias, prejudice, or intolerance, and is frequently aimed towards particular people or groups. The study of Mollas et al. [5], stated that hate speech centers on

Contact Enesi Femi Aminu enesifa@futminna.edu.ng Department of Computer Science, Federal University of Technology, Minna, Nigeria.

the real or perceived "identity factors" of people or groups, which include, but are not limited to, language, economic or social background, ethnicity, nationality, race, color, descent, gender, disability, health status, or sexual orientation. To address this pressing issue, some technological solutions such as artificial intelligence (AI) have been employed.

There have been growing interest in using AI and natural language processing (NLP) to address social and ethical issues. To mention the latest trends on *AI for social good*, where the emphasis is on developing applications to maximize "good" social impacts while minimizing the likelihood of harm and disparagement to those belonging to vulnerable categories [6]. AI, which includes subfields like machine learning and deep learning, is the application of computer science to problems by integrating large datasets. The creation of expert systems for predictions or classifications based on incoming data is a component of these subfields. Diverse levels of attention have been paid to AI development, with OpenAI's ChatGPT marking an important turning point. Previous breakthroughs focused on computer vision, but this one is in NLP. Beyond language, generative models have proven to be adept in comprehending photographs, software code, chemical structures, and a variety of other data kinds. This change represents AI's ongoing development.

On the other hand, machine learning, a branch of computer science and AI, uses data and algorithms to gradually improve accuracy in an effort to mimic human learning processes. The history of machine learning has been greatly influenced by an IBM researcher; Arthur Samuel, who is credited with coining the term during his work on checkers [7]. The emergence of machine learning-based models has been spurred by technological developments. The significant increases in processing power and storage capacity throughout the years have propelled this advancement.

Furthermore, in data science, machine learning is essential. Through data mining, algorithms trained with statistical techniques are used for classification, prediction, and insight extraction. These insights support business expansion and well-informed decision-making, especially in light of the growing amounts of big data. Consequently, some other robust (deep) learning models are promising to effectively learn from huge volumes of data such as the social data in all open social networks. Bidirectional encoder representations from transformers (BERT), attention mechanism, and long short-term memory networks (LSTM) are some of the models that are promising. Therefore, this research is motivated to employ the hybrid of BERT-Attention and Context Analyzer in order to detect hate speech in textual content. This is because BERT learns multiple attentions concurrently at the same time. The remaining sections of this paper are organized as follows: section 2 gives an account of the existing related literature to the subject matter, while the methodology and methods employed were accounted for by section 3, section 4 discusses the results obtained, and finally the conclusion section.

Related Works

Based on research on Twitter abuse of Black people conducted by Kwok and Wang [8], the need for better classification algorithms was brought to light by the inadequacy of the bag-of-words method used to correctly classify anti-black tweets. They were able to get an average accuracy of 76% and an average error rate of 24% by applying the 10-fold cross-validation process. Nevertheless, they ignore factors like text sentiment in favor of a categorization scheme that just considers unigrams. As an illustration, their research showed that derogatory terms like "black," "white," and "filthy" are frequently employed in hate speech directed. On the other hand, these expressions might not always imply racism when given out of context. Bigrams are not taken into account by their classifier when capturing word connections; therefore, tweets that contain these terms can be incorrectly categorized as racist, which could lower overall accuracy.

Davidson et al. [9] studied the issue of offensive language and automated hate speech detection. The top-performing model has a high F1 score (0.90), recall (0.90), and overall precision (0.91). Nonetheless, it was found that the precision and recall scores for the hate class were 0.44 and 0.61, respectively; indicating that around 40% of hate speech is incorrectly labeled. The machine is more likely than human coders to classify tweets as less nasty or insulting because the majority of misclassifications happen on the upper side of this matrix. A smaller proportion of tweets are falsely tagged as more offensive or hateful than they actually are; roughly 5% of hostile tweets and 2% of innocuous tweets are mistakenly classed as hate speech. Thus, a more hybridized approach would improve the results.

Poletto, et al. [10] carried out a methodical examination of benchmark corpora and resources for the purpose of identifying hate speech. An increasing amount of materials and benchmark corpora for many languages have been available since 2016, indicating a growing interest in the identification of hate speech and abusive language on social media. However, due to specific target characteristics, categorization needs, and potential biases in data collection, this profession faces challenges. Although earlier studies called for a common methodology and benchmarking materials, significant progress has been made in creating diverse benchmark datasets for the identification of abusive language. It is a promising research direction to address the stability and performance of detection models across languages and domains. Moreover, a common taxonomy is needed to link numerous similar ideas. Biases in the creation and annotation of corpora, like racial biases, are extremely concerning and require attention. As fine-grained annotations become more common in benchmark corpora, error analysis especially in shared tasks can provide insights about dataset quality and annotation systems. The literature equally echoed the need for data augmentation in order to avoid errors in results.

The study of Kovács, et al. [11] aims to tackle the challenges of hate speech identification in Social Media, the model was applied to the HASOC2019 corpus data, and they were successful in achieving a macro F1score of 0.63 in hate speech identification on the HASOC test set. The ability of DNNs to train efficiently is linked to an increased danger of overfitting, though. Particularly in situations with limited training data (e.g., HASOC). They examined the impact of adding more data, both labeled and unlabeled, on the task of automatically recognizing hateful and offensive speech, using the HASOC 2019 challenge as an example. The state-of-the-art results at the time of submission by utilizing a variety of traditional machine learning techniques in addition to many deep learning models that included RoBERTa and FastText were reported. Hence, this motivated the research to exploit the capability of the BERT model where the attention mechanism is incorporated.

Annotating a Danish dataset of user-generated comments from Facebook and Reddit to capture different kinds and targets of offensive language is done in light of Sigurbergsson and Derczynski [12] study on the detection of hate speech and offensive language in Danish. They created four automatic categorization algorithms, and they were made to function in both Danish and English. The macro-averaged F1 scores of the best-performing system are 0.70 for Danish and 0.74 for the offensive language in English. When determining whether an offensive post is targeted, the best-performing system for Danish achieves a macro-averaged F1-score of 0.73, whereas the best system for English earns a macro-averaged F1-score of 0.62. In summary, the best algorithms for Danish and English identify the target type in a targeted offensive post with macro-averaged F1 scores of 0.63 and 0.56, respectively. A better result is envisaged when the hybridized strategy is deployed.

Xia, et al. [13] aim to decrease racial bias in hate speech identification served as inspiration for their adversarial objective, which they employed to stop the model from encoding information on the protected attribute. Adversarial training is well known for its ability to change models to gain representations that are insensitive to undesirable features, such as subjects and demographics, even though it seldom totally disentangles traits. Their macro-average F1 score was 76.05. They also obtained a macro-average accuracy of 90.68. For hate speech, there was a false positive rate of 2.60, and for offensive language, 17.69. Similarly, the work of Mosca, et al. [14] carried out a study to understand and interpret the impact of user context in hate speech detection. To capture the variations in their behavior, they standardized and streamlined their text processing. As an alternative, the second model makes use of information from three sources: the content of the tweet, the network of followers, and the user's vocabulary. The initial input is the same as that which is fed into the text models. The second pulls every tweet from the dataset in an attempt to replicate the author's general writing style. They illustrated how user features affect the model's judgment, and how they affect the feature space the model has learned using explain-ability techniques.

This research explores the potential of applying domain-specific word embedding in a Bidirectional LSTM-based deep model for automated hate speech detection and classification, building on the work of Saleh, et al. [15] on hate speech detection using BERT and hate speech word embedding with deep model. The problem of binary classification in hate speech and how to approach it with the transfer learning language model (BERT) was also looked into. Based on a balanced combination of existing hate speech datasets, the research discovered that a 93% F1-score was obtained using domain-specific word embedding using the Bidirectional

LSTM-based deep model, and up to 96% with BERT. The attention mechanism being integrated with BERT would offer a better result.

Mathew, et al. [16] used state-of-the-art models to analyze HateXplain, they found that even models that perform exceptionally well in classification do not score highly on explain ability measures like faithfulness and model plausibility. Additionally, it was found that human reasoning-trained models perform better than other algorithms when it comes to minimizing inadvertent bias against target populations. HateXplain with attention has a macro F1 score of 0.687 and an accuracy of 0.698 in BERT. BiRNN-HateXplain with attention obtained a 0.629 macro accuracy and 0.629 F1 score. With a macro accuracy of 0.627 and an F1 score of 0.606, CNN-GRU lime-based performed well. Similarly, in line with the strength of the deep learning approach, Roy, et al. [17] designed an automated approach that uses a deep convolutional neural network (DCNN) to identify hate speech. The proposed DCNN model uses the GloVe embedding vector and a convolution operation to extract semantic content from tweets. The best-projected recall values for HS and NHS were 0.88 and 0.99, respectively, with F1 Score and Precision of 0.97 and 0.92, using the recommended DCNN model and 10-fold cross-validation.

Mullah and Zainon [18] aim to utilize machine learning models for the advancements of social media hate speech identification. The datasets used in this study were obtained from IEEE Explore, ACM, ScienceDirect, University Sains Malaysia, and Scopus. The search retrieval process using keywords or phrases identified all profanity, hate speech, violent and abusive remarks, cyberbullying, and poisonous remarks on social media. Accuracy is not always the best option in diverse and unbalanced datasets; additional evaluation factors may be utilized instead. The recall was 0.80, the accuracy was 0.67, and the F-measure was 0.72.

The study of MacAvaney, et al. [19] assessed the effectiveness of the proposed system and other hate speech detection methods using a variety of hate speech datasets, The researchers recommended using a multi-view SVM model for hate speech classification. A multiple-view stacked support vector machine (mSVM) was used to train the model using the Stormfront, TRAC (Facebook), HatebaseTwitter, and HatEval datasets. Only two of the models they recommended were actually employed. It obtained an accuracy of 0.8033 and a macro F1 score of 0.8031 on mSVM for stormfront. It obtained an accuracy of 0.6121 on TRAC and a macro F1 score of 0.5368 on mSVM. More so, based on the research conducted by Gomez, et al. [20] on exploring hate speech detection in multimodal publications, all photos were resized to 500 pixels at their smallest size. Using online data augmentation, 299 × 299 patches are randomly cropped and mirrored during training. Using a CNN as the image features extractor and an Imagenet pre-trained Google Inception v3 architecture, they trained a single layer LSTM with a 150-dimensional hidden state to categorize hate/not-hate messages or tweets. Next, the text image was subjected to the Google Vision API Text Detection module. Several models were trained in this investigation; however, the Feature Concatenation Model performed exceptionally well for image text, tweet text, and photos, achieving an accuracy of 68.4 and an F1 score of 0.704. Similar to this, the Spatial Concatenation Model performed admirably, obtaining an accuracy of 68.5 and an F1 score of 0.702. Therefore, this research aims to uniquely hybridize the strength of both BERT with attention mechanism and context analyzer to enable a robust mechanism of detecting hate speech in textual form.

The Methodology: Hybridization Approach of BERT-Attention and Context Analyzer

Hybridization methodology often comes with a unique strategy and offers a robust result in many cases. Drawing from this fact, the hybridization approach is considered in this research. This is by bringing the mechanism of BERT-Attention with the context analyzer in detecting hate speech. Figure 1 presents the conceptual framework of the entire process to realize the detective model.

On the aspect of the data augmentation process, generally speaking, data augmentation in NLP enhances model performance and generalization by creating diverse training examples. Back translation is a key technique of data augmentation that this research has employed. This is because in a situation where a sentence is translated to a target language and then back to the source language, generating paraphrases that retain the original meaning but vary in structure and word choice. Hence, the choice of a deep-translator library becomes inevitable and ideal for this task due to the following reasons. It supports multiple translation APIs, and the library offers an intuitive interface for performing translations. More importantly, practitioners can effortlessly incorporate back translation into their data pipelines, thus enhancing model

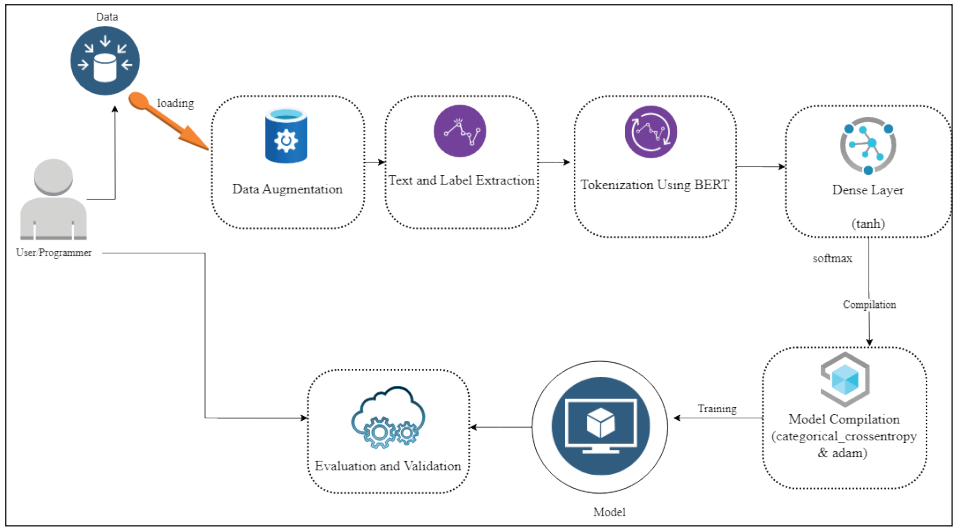

Figure 1. Conceptual framework of the proposed hate-speech detector model.

robustness and decreasing overfitting through augmented and diverse training data when the deep translator is used. The deep_translate module is used to enhance the dataset with translations using the Google translation API after it has been loaded. An enhanced dataset with more linguistic diversity is produced by carefully translating the dataset's remark column into French and then back to English. Then, before being utilized to create a BERT-based machine learning model architecture, the enriched data is tokenized and preprocessed. This architecture is then trained on the enhanced dataset to distinguish hate speech patterns and linguistic nuances. After training, the model is examined to measure its efficacy in hate speech identification, producing comprehensive metrics. The revised model aims to demonstrate a better ability to identify complex linguistic structures, making it more effective in detecting hate speech in a variety of linguistic contexts.

Figure 1 shows the Hate Speech Detector's conceptual layout. The design outlines a numbers of elements and how they work together; including the categorization layer, BERT-Attention Mechanism, Embedding Layer, moderator, and user interface.

The user evaluates and verifies the efficacy of the model by executing actions based on the specific kind of hate speech identified, either through tools or algorithms. Back translation was added to the dataset, which is the process of translating a sentence from one language to another and back again. Back translation may result in changes to the translated content because different languages have different linguistic meanings, grammar, and word choices. This procedure aids in producing new sentences that are both distinct and somewhat similar. The dataset contains the texts as well as the labels that correspond with them. The classifications encompass seven distinct categories of hate speech, with differing degrees of hate speech represented in the text. Preparing unprocessed text input for use in NLP tasks involves cleaning and modifying it.

The architecture leverages on the powerful contextual embeddings generated by BERT and fine-tunes them with additional layers for the specific classification task. By freezing the BERT layer, the model maintains the rich language understanding from pre-training while focusing on learning task-specific patterns from the training data. To be specific, the role of each component of the architecture are detailed as follows:

i. Text and label extraction: this can be described as an embedding extraction. The BERT encoder processes input sentences and generates embeddings for each token. Typically, the embedding of the [CLS] token (which represents the entire sentence) is used as the sentence representation. This fixed-size embedding captures the meaning and context of the input text, making it suitable for downstream tasks.
ii. BERT encoder layer (Frozen): At the core of the model is a pre-trained BERT layer, which serves as the feature extractor. This layer is kept frozen during training, meaning its weights are not updated. The BERT layer's primary function is to transform input text into high-dimensional

embeddings that capture contextual information and semantic nuances.

iii. Dense layer: Following the BERT encoder, a fully connected (dense) layer is added to further process the extracted embeddings. This layer applies a linear transformation to the embeddings, which can help in refining the features for the specific classification task. The dense layer often includes an activation function, such as ReLU (Rectified Linear Unit), to introduce non-linearity and enhance the model's expressive power.

iv. Dropout layer: To prevent overfitting and improve generalization, a dropout layer is incorporated after the dense layer. Dropout randomly sets a fraction of the input units to zero during each forward pass, which helps the model become more robust by preventing reliance on specific neurons.

v. Output layer: The final component is a softmax output layer tailored for multi-class classification. This layer outputs a probability distribution over the predefined classes, with each value representing the likelihood of the input belonging to a particular class. The number of neurons in this layer corresponds to the number of target classes in the classification task.

There are several steps involved: by considering words in different cases as the same thing (e.g., "Word" and "word"), lowercase text encourages uniformity. Deleting any extraneous symbols and punctuation.

Removing numbers: If a number is unnecessary, it may be removed in order to make text analysis simpler.

Managing contractions: Depending on the assignment, this phase may vary, however words like "can't" can be expanded for uniformity.

Removing stop expressions: Take off terms like "and," "the," and so on to highlight words that are more important. Lemmatization and stemming are two methods that reduce words to their most fundamental or root forms. For example, "running" becomes "run". Lemmatization ensures the validity of the words.

The tokenization step, which comes next, splits text into smaller pieces known as tokens. Depending on the assignment, tokens can be words, phrases, sub words, or characters. Since it enables representation as numerical vectors (word embedding), which may be fed into machine learning algorithms or neural networks, it is crucial for many NLP activities. In order for the neural network to learn complicated patterns, add non-linearity, decrease dimensionality, and map the learned features to the particular classes in your multi-class classification problem, dense layers are necessary. They are essential in turning the meaningless representations from the previous layers into accurate forecasts.

In addition, for multi-class classification applications, the categorical cross-entropy loss function is used in the model's construction. The model is trained using the Adam optimizer, and the performance of the model is evaluated using the accuracy metric. The sequential actions of the model are shown in Figure 2.

The sequence diagram shows how information moves between the various modules and parts of the system, from an unprocessed text input to the development of a machine-learning model that is trained and stored in a file for later use. Each module is crucial to the processing and manipulation of data, which eventually results in the creation of a neural network model that is successful in text classification.

a. User: Identifies the user engaging with the system.
b. Data processing (DP): Handles data-related tasks like loading, preprocessing, and augmentation. Receives user requests to load and preprocess data.
c. Model creation (MC): Creates and defines the machine learning model architecture.
d. Training (T): Utilize preprocessed data for model construction training. After the model creation stage, activation takes place. Table 1 shows the algorithm of the proposed model.

Lastly, specific metrics were chosen to comprehensively evaluate the model's performance due to the nature of the classification task and the need for a nuanced understanding of how well the model distinguishes between classes. Choosing the recall, precision, and f1-score metrics together provides a comprehensive evaluation of the model's performance. While accuracy gives a general overview, precision, recall, and F1-score offer insights into the model's behavior in more detail, particularly in handling false positives and false negatives. This holistic approach ensures a well-rounded understanding of the model's strengths and weaknesses.

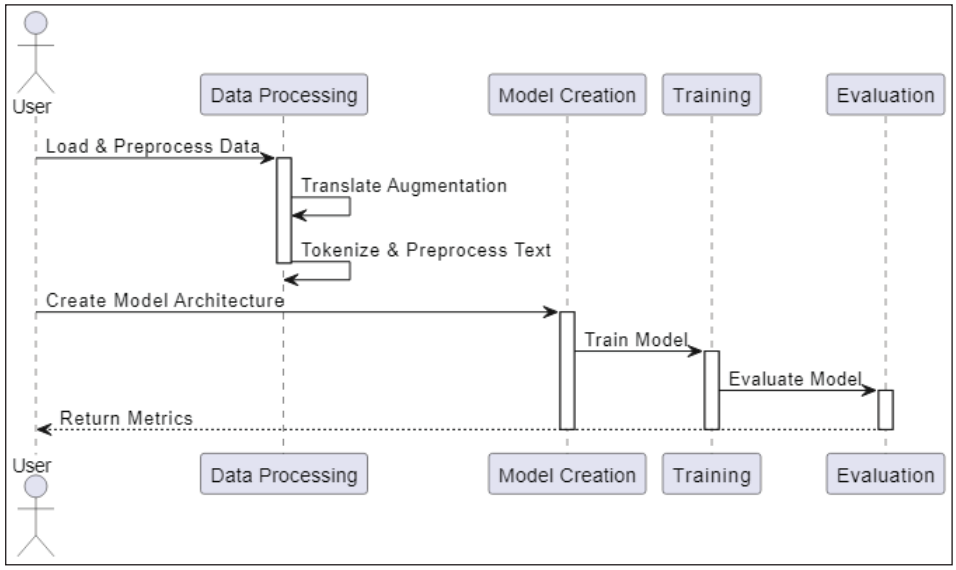

Figure 2. Sequence diagram of the proposed model.

Table 1. Algorithm of the proposed model.

Input: load Hate Speech Dataset
Output: macro precision, macro recall, macro f1-score
Parameters: Hate Speech Dataset (t), data augmentation (da), ResNet (r_A), tokenization using BERT(tBERT), EfficientNet (e_A), Custom Optimized CNN (c_A), macro precision (mp), macro recall (mr), macro f1- score (mf), deep-translator lib (l), augmented dataset (t_a), clean dataset (c_d)

Procedure:
1. *Input t*
2. *t_a = back translation (t) using l // data augmentation*
3. *c_d = Preprocess t_a*
4. *Tokenize (c_d) // tokenize the preprocessed text using tBERT*
5. *r_A (output of line 4); e_A (output of line 4); c_A (output of line 4) // Invoke model's Architectures*
6. *Trains model*
7. *Evaluates(results) // results of the architectures performances*
8. *Returns (mp, mr, mf))*

Results and Discussion

Python is the programming language used to put the recommended approach into practice. Because of its vast ecosystem and numerous advantages, Python is the ideal language for NLP and machine learning applications. Its broad library support which includes TensorFlow, PyTorch, NLTK, and spaCy offers vital resources for creating intricate models and handling text data quickly. Python's readability and ease of use make it simple to write and comprehend code, which is crucial for complicated jobs.

The dataset used for system development and evaluation of the proposed system was obtained from 'ETHOS' (multi-label hate speech detection dataset), a textual dataset with two variants: binary and multi-label, based on YouTube and Reddit comments validated using the Figure. Eight crowdsourcing platform. The sample of unaugumented dataset is shown in Figure 3.

All of the dataset's attributes were displayed in the first and last four rows of the full dataset. A certain amount of various hate speech is assigned to each comment. Figure 4 shows the data augmentation process.

Using the deep_translate module, which leverages the Google translation API, the comment column was translated methodically during the project's phase. Every remark was carefully examined and accurately translated into French. The delay time between translations was set at 1 second in order to guarantee effective API consumption and prevent overloading the service. After that, every comment was translated into English once more, producing an extensive augmented dataset.

The goal of this augmentation process is to expose the model to a wide range of linguistic subtleties and technical terminology in addition to growing the dataset size. This varied dataset was

In [ ]:
```
df
```

Out[123]:

	comment	violence	directed_vs_generalized	gender	race	national_origin	disability	religion	sexual_orientation
0	You should know women's sports are a joke	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
1	You look like Sloth with deeper Down's syndrome	0.000000	0.833333	0.000000	0.000000	0.000000	1.000000	0.000000	0.000000
2	You look like Russian and speak like Indian. B...	1.000000	0.500000	0.142857	0.142857	0.714286	0.000000	0.000000	0.000000
3	Women deserve to be abused, I guess.	0.666667	0.166667	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
4	Women are made for making babies and cooking d...	0.000000	0.000000	1.000000	0.000000	0.000000	0.000000	0.000000	0.000000
...	...	...	...	...	...	...	...	...	...
428	All people are fucked up because of these quee...	0.000000	0.000000	0.571429	0.000000	0.000000	0.000000	0.000000	0.571429
429	Actually THIS WEIRD PEOPLE are HARMFUL! They n...	0.000000	1.000000	0.000000	0.000000	0.333333	0.666667	0.000000	0.000000
430	ABU DA BI ABA DU BA BA HAMUD! See? I'm an arab...	0.000000	0.000000	0.000000	0.285714	0.000000	0.000000	0.857143	0.000000
431	a homo? so you fuck your brother instead of yo...	0.000000	0.666667	0.000000	0.000000	0.000000	0.000000	0.000000	1.000000
432	(((They))) will be chased out of every white c...	0.333333	0.000000	0.000000	0.666667	0.000000	0.000000	0.333333	0.000000

433 rows × 9 columns

Figure 3. Sample of unaugmented dataset.

```python
from deep_translator import GoogleTranslator
import time

# Function to perform back translation on the 'comment' column and maintain labels
def perform_back_translation_with_labels(dataframe, target_language='fr', delay=1):
    augmented_data = []

    for index, row in dataframe.iterrows():
        text = row['comment']
        original_labels = row.drop('comment').tolist()  # Extract original labels

        # Translate to target language
        translated = GoogleTranslator(source='auto', target=target_language).translate(text)

        # Introduce a delay between requests
        time.sleep(delay)

        # Translate back to original language
        back_translated = GoogleTranslator(source=target_language, target='en').translate(translated)

        # Append original and back-translated comments and their respective labels to augmented data
        augmented_data.append({'comment': text, **dict(zip(dataframe.columns[1:], original_labels))})
        augmented_data.append({'comment': back_translated, **dict(zip(dataframe.columns[1:], original_labels))})

    return pd.DataFrame(augmented_data)

# Perform back translation on the 'comment' column and maintain labels
# Using delay=1 second between requests
augmented_df = perform_back_translation_with_labels(df, delay=1)
```

Figure 4. Data augmentation process.

included with the intention of strengthening the model's comprehension of minute linguistic details and facilitating its acquisition of related linguistic constructs, thereby enhancing its capacity to identify hate speech. Figure 5 shows a snippet of the augmented dataset.

Owing from the augmented dataset obtained as shown in Figure 5, there is a need to develop an algorithm that would be able to perform label extraction, and at the same time, normalize the data. Thus, the following algorithm was implemented.

i. String labels = [property]
ii. Text = augmented_df['comment']
iii. Label_df = augmented_df[labels]
iv. Def transform_labels(records)
v. Max = record.argmax():
vi. For 1, v in records.item();
vii. | If 1 != record.index[_max];
viii. | | Record[1] = 0
ix. | Else:
x. | | Record[1] = 1
xi. Return record

```
In [8]: augmented_df.head(20)
```

Out[8]:

	comment	violence	directed_vs_generalized	gender	race	national_origin	disability	religion	sexual_orientation
0	You should know women's sports are a joke	0.000000	0.000000	1.000000	0.000000	0.000000	0.0	0.0	0.000000
1	You should know that women's sports are a joke	0.000000	0.000000	1.000000	0.000000	0.000000	0.0	0.0	0.000000
2	You look like Sloth with deeper Down's syndrome	0.000000	0.833333	0.000000	0.000000	0.000000	1.0	0.0	0.000000
3	You look like Sloth with deeper Down syndrome	0.000000	0.833333	0.000000	0.000000	0.000000	1.0	0.0	0.000000
4	You look like Russian and speak like Indian. B...	1.000000	0.500000	0.142857	0.142857	0.714286	0.0	0.0	0.000000
5	You look like a Russian and speak like an Indi...	1.000000	0.500000	0.142857	0.142857	0.714286	0.0	0.0	0.000000
6	Women deserve to be abused, I guess.	0.666667	0.166667	1.000000	0.000000	0.000000	0.0	0.0	0.000000
7	Women deserve to be mistreated, I guess.	0.666667	0.166667	1.000000	0.000000	0.000000	0.0	0.0	0.000000
8	Women are made for making babies and cooking d...	0.000000	0.000000	1.000000	0.000000	0.000000	0.0	0.0	0.000000
9	Women are made to make babies and cook dinner ...	0.000000	0.000000	1.000000	0.000000	0.000000	0.0	0.0	0.000000
10	why haven't you faggots taken the bi-pill yet	0.500000	0.500000	0.000000	0.000000	0.000000	0.0	0.0	1.000000
11	why haven't you fags taken the bi-pill yet?	0.500000	0.500000	0.000000	0.000000	0.000000	0.0	0.0	1.000000
12	Why don't you go beck your god to murder you? ...	0.833333	0.500000	0.000000	0.000000	0.000000	0.0	1.0	0.000000
13	Why don't you go ask your god to murder you? C...	0.833333	0.500000	0.000000	0.000000	0.000000	0.0	1.0	0.000000
14	why does that african american monkey hold gun...	0.000000	0.500000	0.000000	1.000000	0.000000	0.0	0.0	0.000000
15	Why is this African American monkey holding a ...	0.000000	0.500000	0.000000	1.000000	0.000000	0.0	0.0	0.000000
16	Why does Bulgaria exist as a country? It's one...	0.000000	0.500000	0.000000	0.000000	0.428571	0.0	0.0	0.571429
17	Why does Bulgaria exist as a country? He's one...	0.000000	0.500000	0.000000	0.000000	0.428571	0.0	0.0	0.571429
18	Why do negroes tear down their own neighborhoo...	0.250000	0.000000	0.000000	1.000000	0.000000	0.0	0.0	0.000000
19	Why do Negroes destroy their own neighborhoods...	0.250000	0.000000	0.000000	1.000000	0.000000	0.0	0.0	0.000000

Figure 5. Augmented dataset.

```
In [27]: l_tmp
```

Out[27]:

	violence	gender	race	national_origin	disability	religion	sexual_orientation
0	0.0	1.0	0.0	0.0	0.0	0.0	0.0
1	0.0	1.0	0.0	0.0	0.0	0.0	0.0
2	0.0	0.0	0.0	0.0	1.0	0.0	0.0
3	0.0	0.0	0.0	0.0	1.0	0.0	0.0
4	1.0	0.0	0.0	0.0	0.0	0.0	0.0
...	...	...	...	...	...	...	...
861	0.0	0.0	0.0	0.0	0.0	1.0	0.0
862	0.0	0.0	0.0	0.0	0.0	0.0	1.0
863	0.0	0.0	0.0	0.0	0.0	0.0	1.0
864	0.0	0.0	1.0	0.0	0.0	0.0	0.0
865	0.0	0.0	1.0	0.0	0.0	0.0	0.0

866 rows × 7 columns

Figure 6. Sample of normalized dataset.

From the algorithm, lines 1–3, depict the extraction of the label using a general variable property as argument. The values of it include gender, violence, disability, religion, national, and the likes. Afterward, the labels were taken out and placed in a variable named "label," a program was created to normalize the dataset and turn it into a multiclass binary classification dataset. The argmax function

was then used to turn the dataset into binary form. In order to do this, each index was cycled through and the property with the highest percentage value was given a value of 1, while the values of all other indexes were truncated to 0. After that, the function was used on the designated labels. Figure 6 shows the sample of the normalized dataset.

The sample of the normalized dataset in Figure 6 is the product of the implemented algorithm in order to have a normalized dataset that can represented and understand by machine. Figure 7 shows the loading of the pre-trained BERT model and BERT tokenizer.

The training process is initiated using fit(X_train, y_train, epochs = epochs, validation_split = 0.1, shuffle = True, batch_size = batch_size). The training labels are indicated by y_train, and the input training data (tokenized text sequences) is represented by x_train. 10% of the training data will be used for validation during training, according to the validation_split = 0.1 parameter. In order to avoid overfitting, the shuffle = True option ensures that the training data is jumbled prior to each epoch. This keeps the model from learning the sample order. Finally, the batch_size option shows the batch size that was used for training, which is the same as the amount that was previously established.

Using the given loss function and optimization technique, the model learns to improve its internal parameters (weights and biases) while mapping input sequences to their labels during training. Training progress and performance metrics such as accuracy, loss, validation accuracy, and validation loss are recorded and stored in the history variable.

The training process for the text classification model, implemented in the research, was conducted on Google Colab using an NV to comprehensively evaluate the model's IDIA T4 GPU to expedite the training phase. Leveraging the T4 GPU on Google Colab significantly accelerated the training process, enabling efficient handling of the computational demands required for fine-tuning the BERT model. This setup facilitated a robust training regimen, ensuring the model achieved high performance on the text classification task. On the aspect of hyperparameters, the learning rate was set to 2e–5, balancing between slow convergence and the risk of overshooting optimal solutions. A batch size of 32 was chosen to maximize the utilization of available GPU memory while maintaining efficient gradient updates. The model was trained for 20 epochs, ensuring sufficient training time to capture patterns in the data without overfitting. AdamW optimizer was used, which is tailored for transformer models by including weight decay to prevent overfitting. On the part of the loss function, the cross-entropy loss was utilized, suitable for the multi-class classification task at hand. A dropout rate of 0.1 was applied to the dense layer to introduce regularization and prevent overfitting.

For the training duration, the training process lasted approximately 2 hours. This time encompasses data loading, preprocessing, model training, and evaluation at the end of each epoch. The

```
In [5]: import tensorflow as tf
        from transformers import TFBertModel, BertTokenizer
        from tensorflow.keras.layers import Input, Dense, Activation, Flatten
        from tensorflow.keras.models import Model
        from sklearn.utils.class_weight import compute_class_weight
        from tensorflow.keras.utils import to_categorical

In [6]: # Load pre-trained BERT model
        bert_model = TFBertModel.from_pretrained("bert-base-uncased")
        tokenizer = BertTokenizer.from_pretrained("bert-base-uncased")

        Downloading (…)lve/main/config.json:   0%|          | 0.00/570 [00:00<?, ?B/s]

        Downloading model.safetensors:   0%|          | 0.00/440M [00:00<?, ?B/s]

        Some weights of the PyTorch model were not used when initializing the TF 2.0 model TFBertModel: ['cls.predictions.transform.den
        se.bias', 'cls.seq_relationship.weight', 'cls.predictions.bias', 'cls.seq_relationship.bias', 'cls.predictions.transform.LayerN
        orm.weight', 'cls.predictions.transform.dense.weight', 'cls.predictions.transform.LayerNorm.bias']
        - This IS expected if you are initializing TFBertModel from a PyTorch model trained on another task or with another architectur
        e (e.g. initializing a TFBertForSequenceClassification model from a BertForPreTraining model).
        - This IS NOT expected if you are initializing TFBertModel from a PyTorch model that you expect to be exactly identical (e.g. i
        nitializing a TFBertForSequenceClassification model from a BertForSequenceClassification model).
        All the weights of TFBertModel were initialized from the PyTorch model.
        If your task is similar to the task the model of the checkpoint was trained on, you can already use TFBertModel for predictions
        without further training.

        Downloading (…)okenizer_config.json:   0%|          | 0.00/28.0 [00:00<?, ?B/s]

        Downloading (…)solve/main/vocab.txt:   0%|          | 0.00/232k [00:00<?, ?B/s]

        Downloading (…)/main/tokenizer.json:   0%|          | 0.00/466k [00:00<?, ?B/s]
```

Figure 7. Loading of pre-trained BERT tokenizer and model.

computational resources includes GPU of NVIDIA T4 GPU, and memory of 16 GB RAM. The training process was carried as follows:

i. Data preparation: Text data was tokenized using BERT's tokenizer, converting text into token IDs and padding sequences to a uniform length.
ii. Model initialization: The pre-trained BERT model was loaded with the specified configurations, and additional layers were initialized.
iii. Training loop: For each epoch, the model processed batches of data; forward passes computed predictions; losses were calculated based on predictions and true labels. Backward passes updated weights via gradients computed by the optimizer.
iv. Validation: Post each epoch, the model was evaluated on a validation set to monitor performance and adjust if needed.
v. Checkpointing: Model checkpoints were saved periodically to secure training progress.

Through this training process, the neural network's ability to classify text input can be improved iteratively by adjusting its parameters in order to minimize the loss function. The model's performance on unidentified data may be evaluated after training, providing information on how well-suited the model is for the particular classification task. Figure 8 presents the accuracy of the model.

As seen, the model performs superbly on the training and validation sets, and a steady increase in accuracy shows that it learns and generalizes with each epoch. However, loss during training and validation of dataset is inevitable as rightly shown by Figure 9.

The model is learning to make fewer mistakes in both the training and validation datasets, as seen by the graph's steady decline in loss following each epoch in Figure 9. Several models were constructed and tested using the dataset, and various outcomes were yielded. Figure 10 shows the class confusion matrix for the model.

The snippet shows the class evaluation matrices for various classes from class 1 to class7.Figure 12 shows the macro evaluation scores.

The macro evaluation scores for precision, recall and F1-score are 79.88%, 71.59% and 74.83% respectively by approximation. Table 2 presents the key findings of the proposed hate speech detector in comparison with the findings of related studies.

Table 2 presents the findings of the proposed model against the related studies based on the metrics of macro precision, macro recall, and macro f1

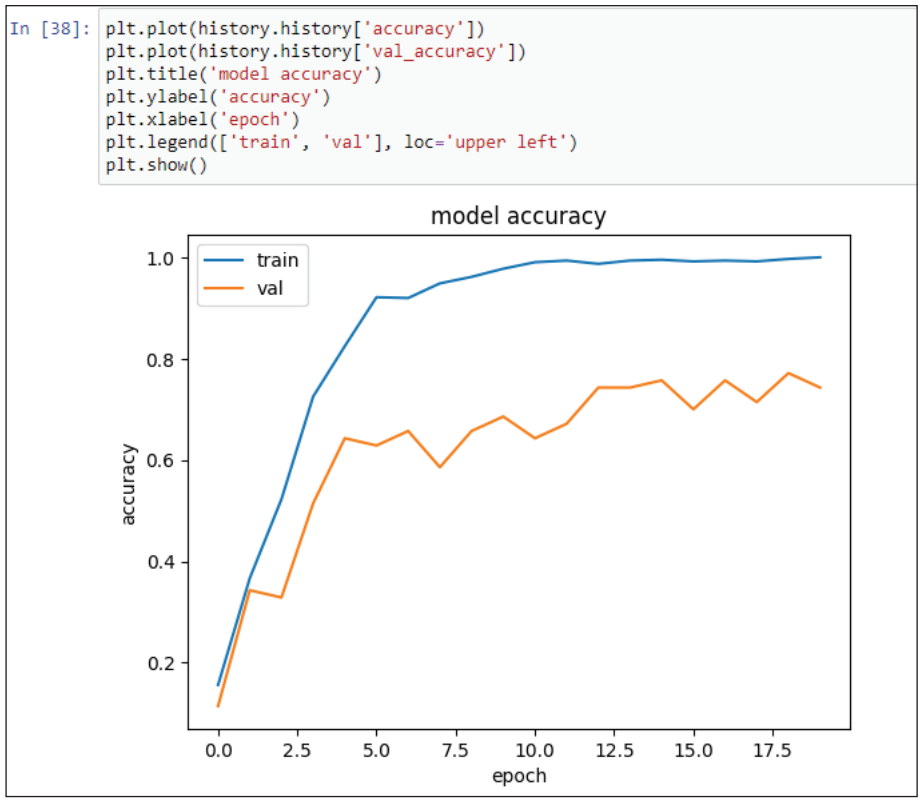

Figure 8. Accuracy of the model based on the training dataset.

```
In [39]: plt.plot(history.history['loss'])
         plt.plot(history.history['val_loss'])
         plt.title('model loss')
         plt.ylabel('loss')
         plt.xlabel('epoch')
         plt.legend(['train', 'val'], loc='upper left')
         plt.show()
```

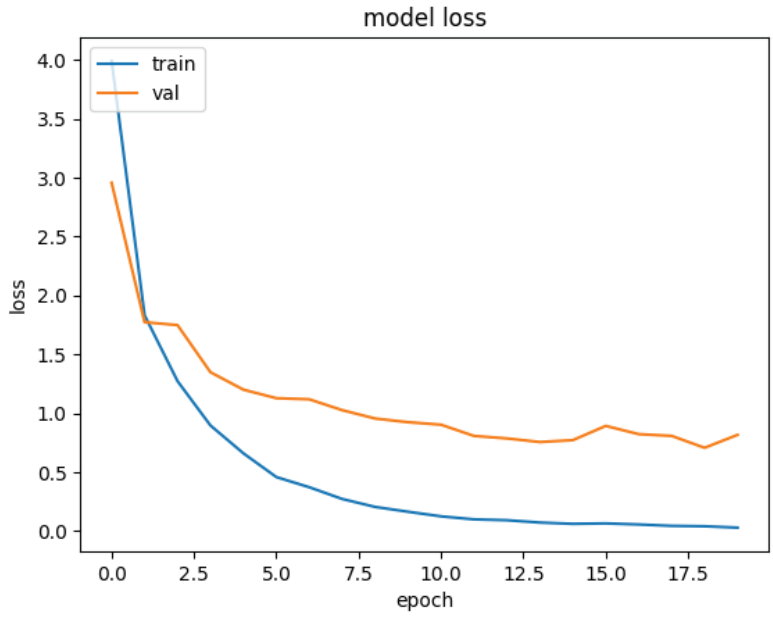

Figure 9. Model loss on training and validation dataset.

```
In [42]: from sklearn.metrics import multilabel_confusion_matrix

         # Compute the confusion matrix
         confusion_matrix = multilabel_confusion_matrix(y_test, y_pred_binary)
         print(confusion_matrix)

[[[141   4]
  [ 16  13]]

 [[150   4]
  [  2  18]]

 [[141   5]
  [  5  23]]

 [[155   2]
  [  6  11]]

 [[146   6]
  [  6  16]]

 [[131   8]
  [  8  27]]

 [[148   3]
  [  7  16]]]
```

Figure 10. Class confusion matrix for the model.

score. The work of Kovács, et al. [11] only presented the macro f1 score of 63% based on RoBERTa and FastText employed in their work with same dataset. Similarly, Mollas, et al. [5] equally used the same dataset and employed several models, which include SVM and BERT. While the former reported

```
    [[141, 4], [16, 13]],  # Class 1
    [[150, 4], [2, 18]],   # Class 2
    [[141, 5], [5, 23]],   # Class 3
    [[155, 2], [6, 11]],   # Class 4
    [[146, 6], [6, 16]],   # Class 5
    [[131, 8], [8, 27]],   # Class 6
    [[148, 3], [7, 16]]    # Class 7
]


# Calculate precision, recall, and F1-score for each class
for idx, confusion_matrix in enumerate(confusion_matrices, start=1):
    tn, fp = confusion_matrix[0]
    fn, tp = confusion_matrix[1]

    if tp + fp == 0 and tp + fn == 0:
        # Both precision and recall are 0, so assign F1-score as 0
        f1 = 0
    else:
        precision = tp / (tp + fp) if tp + fp != 0 else 0
        recall = tp / (tp + fn) if tp + fn != 0 else 0
        f1 = 2 * (precision * recall) / (precision + recall) if preci

    print(f"Class {idx}:")
    print("Precision:", precision)
    print("Recall:", recall)
    print("F1-score:", f1)
    print("-----------------------")
```

```
Class 1:
Precision: 0.7647058823529411
Recall: 0.4482758620689655
F1-score: 0.5652173913043479
-----------------------
Class 2:
Precision: 0.8181818181818182
Recall: 0.9
F1-score: 0.8571428571428572
-----------------------
Class 3:
Precision: 0.8214285714285714
Recall: 0.8214285714285714
F1-score: 0.8214285714285714
-----------------------
Class 4:
Precision: 0.8461538461538461
Recall: 0.6470588235294118
F1-score: 0.7333333333333334
-----------------------
Class 5:
Precision: 0.7272727272727273
Recall: 0.7272727272727273
F1-score: 0.7272727272727273
-----------------------
Class 6:
Precision: 0.7714285714285715
Recall: 0.7714285714285715
F1-score: 0.7714285714285715
-----------------------
Class 7:
Precision: 0.8421052631578947
```

Figure 11. Classes' evaluation matrix.

```
In [44]: # Lists to store precision, recall, and F1-score for each class
         precisions = []
         recalls = []
         f1_scores = []

         # Calculate precision, recall, and F1-score for each class
         for idx, confusion_matrix in enumerate(confusion_matrices, start=1):
             tn, fp = confusion_matrix[0]
             fn, tp = confusion_matrix[1]

             # Calculate precision, recall, and F1-score for the current class
             precision = tp / (tp + fp) if tp + fp != 0 else 0
             recall = tp / (tp + fn) if tp + fn != 0 else 0
             f1 = 2 * (precision * recall) / (precision + recall) if precision + recall != 0 else 0

             # Append the calculated metrics to the lists
             precisions.append(precision)
             recalls.append(recall)
             f1_scores.append(f1)

         # Calculate macro-averaged precision, recall, and F1-score
         macro_precision = sum(precisions) / len(precisions)
         macro_recall = sum(recalls) / len(recalls)
         macro_f1 = sum(f1_scores) / len(f1_scores)

         print("Macro Precision:", macro_precision)
         print("Macro Recall:", macro_recall)
         print("Macro F1-score:", macro_f1)

         Macro Precision: 0.7987538114251957
         Macro Recall: 0.7158738185201844
         Macro F1-score: 0.7482468876878815
```

Figure 12. Macro evaluation scores.

Table 2. Hate speech's results comparison.

Model	Precision (%)	Recall (%)	F1-score (%)
Kovács, et al. [11] (RoBERTa and FastText)	-	-	63.00
Mollas, et al. [5](SVM)	66.47	66.70	66.07
Mollas, et al. [5] (BERT)	79.89	79.73	79.60
Proposed model	79.88	71.59	74.83

66.47%, 66.70%, and 66.07%, the latter presented 79.89%, 79.73%, 79.60% for precision, recall, and f1 score respectively. However, the proposed hybridized model of BERT Attention mechanism with context analyzer in this research presented 79.88%, 71.59%, and 74.83% for macro precision, macro recall, and macro f1 score respectively. It is important to note that Mollas, et al. [5] used BERT for binary classification, while in the research BERT is used for multiclass or labels classification specifically, eight classes of hate speech. Therefore, this statement explained better the rationale behind the performance of the proposed model and the existing BERT model of Mollas, et al. [5]. Consequently, with the 74.83% f1 score of the proposed model considering the number of classes of hate speech, the hybridized approach proposed in this research is promising to effectively detect misinformation.

Finally, in order to ascertain the actualization of the set objectives, and effectiveness of the model, the following research questions were formulated.

i. How do to ascertain the level of performance of back translation's data augmentation technique?
ii. How to ascertain the level of robustness and generalization of the proposed model?

The fact that the model performed better than the base research suggests that the augmentation technique, specifically back translation, effectively enhanced the model's ability to generalize and learn from a larger and more diverse dataset. Back translation introduces synthetic data by translating sentences from the target language to a foreign language and back to the target language, which diversifies the linguistic patterns and contexts seen during training. Hence, an objective for question one, which is to enhanced model performance has been successfully achieved.

Similarly, the improved performance indicates that the model is more robust and capable of generalizing across different types of hate speech instances. It suggests that the model is not overly

fitting to the training data but rather learning meaningful representations that apply well to unseen data; a crucial aspect for practical applications where hate speech manifests in diverse forms across various online platforms. Thus, another objective derived from question two, which is the robustness and generalization of the model has actualized.

Conclusion

This study employed cutting-edge NLP techniques to combat hate speech online, which is a significant social concern. Hate speech is a sort of harmful communication that keeps spreading over digital platforms, causing serious problems for both individuals and online communities. The project intends to create a novel hate speech detection system based on a hybridized approach that combines the capabilities of a context analyzer and the BERT attention mechanism in response to this growing issue. Comprehensive knowledge of hate speech and its nuances in various contexts is the basis of this research. Hate speech encompasses a wide range of discriminatory statements made against particular social groups on the basis of traits like race, gender, sexual orientation, and religion. Careful attempts were taken to augment the existing data in order to offer a robust and comprehensive dataset. The deep-translator library's back translation capability was one of the data augmentation techniques that was employed. This method made it possible to create additional data points, which increased the amount and diversity of the dataset—a crucial component for developing a successful hate speech detection program.

The work aims to capture the complex contextual links within textual data using state-of-the-art BERT attention mechanisms, enabling a more in-depth analysis of hate speech patterns. Additionally, by including a context analyzer, the model is better able to discriminate between speech that is considered hate speech and speech that is lawful, which leads to increased detection accuracy. The system carefully pre-processes, tokenizes, and trains textual data to provide a robust and effective hate speech detection model. The hybridized BERT attention mechanism allows for a more thorough comprehension of damaging material by identifying not only overt hate speech but also subtle subtleties and hidden meanings. Online safety and the creation of inclusive digital environments are greatly enhanced by the addition of a context analyzer, which enhances the model's predictive abilities by enabling it to distinguish between legitimate phrases and potentially dangerous words.

In the continuous fight against hate speech on the internet, this initiative is a significant step forward. With the use of state-of-the-art NLP methods, the project aims to create a sophisticated, precise, and contextually aware hate speech detection system. This program contributes to a safer and more accepting digital environment by encouraging diversity, cultivating polite online interactions, and lessening the damaging effects of hate speech on marginalized communities. In the areas of internet safety and NLP, this work marks a substantial advancement. This innovative method demonstrates a deep understanding of the nuances of hate speech by fusing the power of BERT attention processes with a context analyzer, enabling accurate identification and categorization of harmful content. The project's all-encompassing methodology, which makes use of advanced modeling techniques, tokenization, and pre-processing, guarantees that the system can reliably and robustly discern between acceptable and inappropriate language.

This research not only significantly advances ongoing efforts to reduce online toxicity, but it also lays the foundation for the creation of more polite and inclusive online environments, since hate speech is still an issue in digital contexts. This research is notable for its cutting-edge technology and intelligent interpretation of hate speech. However, despite all the resources that have been deployed in this work, there are still inherent limitations, which have been outlined and discussed in this work along with suggestions for further improvements. The effectiveness of back translation and other augmentation techniques may vary significantly depending on the specific dataset used. Future research should explore the generalizability of these findings across different datasets that encompass a wider range of languages, dialects, and cultural contexts.

More so, while the study utilized standard metrics like accuracy, precision, recall, and F1-score, these metrics may not fully capture the nuances of hate speech detection, especially in terms of understanding context and subtleties in language. Future research could explore more nuanced evaluation metrics or combine multiple metrics to provide a more comprehensive assessment of model performance. Similarly, although efforts were made to mitigate biases in the training data, bias in AI models remains a significant concern in hate speech detection. Future studies should focus on developing

techniques to detect and mitigate bias more effectively, ensuring that AI systems do not perpetuate or amplify societal biases. Finally, the study primarily focused on model performance under controlled experimental conditions. Future research should investigate how these AI techniques perform in real-world settings, considering factors such as real-time processing, scalability, and integration into existing content moderation systems.

References

1. Jahan MS, Oussalah M. A systematic review of hate speech automatic detection using natural language processing. Neurocomputing [Internet] 2023; 546:1–30. Available via https://doi.org/10.1016/j.neucom.2023.126232
2. Balouchzahi F, Shashirekha HL, Sidorov G. HSSD : Hate speech spreader detection using N-grams and voting classifier. In: Conferemce and Labs of the Evaluation Forum. Bucharest, Romania, p. 1–8, 2021.
3. Boishakhi FT, Shill PC, Alam MGR. Multi-modal hate speech detection using machine learning. In: IEEE International Conference on Big Data (Big Data). pp. 4496–9, 2021.
4. Cruz RM, De Sousa WV., Cavalcanti GD. Selecting and combining complementary feature representations and classifiers for hate speech detection. Online Soc Netw Media 2022; 28(100194):1–37.
5. Mollas I, Chrysopoulou Z, Karlos S, Tsoumakas G. ETHOS: a multi-label hate speech detection dataset. Complex Intell Syst [Internet] 2022; 8(6):4663–78. Available via https://doi.org/10.1007/s40747-021-00608-2
6. Chiril P, Wahyu E, Farah P, Véronique B, Viviana M, Patti V. Emotionally informed hate speech detection : a multi—target perspective. Cognit Comput [Internet] 2022; 14:322–52. Available via https://doi.org/10.1007/s12559-021-09862-5
7. Aljabri M, Zagrouba R, Shaahid A, Alnasser F, Saleh A, Alomari DM. Machine learning-based social media bot detection: a comprehensive literature review [Internet]. Soc Netw Anal Min 2023; 13:1–40. Available via https://doi.org/10.1007/s13278-022-01020-5
8. Kwok I, Wang Y. Locate the hate: detecting tweets against blacks. In: Proceedings of the Twenty-Seventh AAAI Conference on Artificial Intelligence. Wellesley 2013; 27(1):1621–2.
9. Davidson T, Warmsley D, Macy M, Weber I. Automated hate speech detection and the problem of offensive language. In: Proceedings of the 11th International Conference on Web and Social Media, ICWSM 2017. Montreal, Canada, pp. 512–5, 2017.
10. Poletto F, Basile V, Sanguinetti M, Bosco C, Patti V. Resources and benchmark corpora for hate speech detection: a systematic review. Lang Resour Eval [Internet] 2021; 55(2):477–523. Available via https://doi.org/10.1007/s10579-020-09502-8
11. Kovács G, Alonso P, Saini R. Challenges of hate speech detection in social media: data scarcity, and leveraging external resources. SN Comput Sci [Internet] 2021; 2(95):1–15. Available via https://doi.org/10.1007/s42979-021-00457-3
12. Sigurbergsson GI, Derczynski L. Offensive language and hate speech detection for danish. In: LREC 2020—12th International Conference on Language Resources and Evaluation, Conference Proceedings. 2023. p. 1–13.
13. Xia M, Field A, Tsvetkov Y. Demoting racial bias in hate speech detection. In: Proceedings of the 55th Annual Meeting of the Association for Computational Linguistics. Canada, pp. 7–14, 2017.
14. Mosca E, Wich M, Groh G. Understanding and interpreting the impact of user context in hate speech detection. In: Ku L-W, Li C-T, editors. SocialNLP 2021-9th International Workshop on Natural Language Processing for Social Media, Proceedings of the Workshop [Internet]. Online, SocialNLP; p. 91–102, 2021. Available via: https://aclanthology.org/2021.socialnlp-1
15. Saleh H, Alhothali A, Moria K. Detection of hate speech using BERT and hate speech word embedding with deep model. Appl Artif Intell [Internet] 2023; 37(1):1–23. Available via https://doi.org/10.1080/08839514.2023.2166719
16. Mathew B, Saha P, Yimam SM, Biemann C, Goyal P, Mukherjee A. HateXplain: a benchmark dataset for explainable hate speech detection. In: 35th AAAI Conference on Artificial Intelligence, AAAI 2021, Washington, DC, 2021, p. 14867–75.
17. Roy PK, Tripathy AK, Das TK, Gao XZ. A framework for hate speech detection using deep convolutional neural network. IEEE Access 2020; 8:204951–62.
18. Mullah NS, Zainon WMNW. Advances in machine learning algorithms for hate speech detection in social media: a review. IEEE Access 2021; 9:88364–76.
19. MacAvaney S, Yao HR, Yang E, Russell K, Goharian N, Frieder O. Hate speech detection: challenges and solutions. PLoS One 2019; 14(8):1–16.
20. Gomez R, Gibert J, Gomez L, Karatzas D. Exploring hate speech detection in multimodal publications. In: Proceedings-2020 IEEE Winter Conference on Applications of Computer Vision, WACV 2020. Snowmass Village, CO, pp. 1470–8, 2020.