Received: July 16, 2024. Revised: September 30, 2024.

International Journal of
Intelligent Engineering & Systems

http://www.inass.org/

An Improved Toxic Speech Detection on Multimodal Scam Confrontation Data Using LSTM-Based Deep Learning

Agustinus Bimo Gumelar[1] **Eko Mulyanto Yuniarno[1,2]** **Arif Nugroho[3]**
Derry Pramono Adi[1] **Indar Sugiarto[4]** **Mauridhi Hery Purnomo[1,2,5,*]**

[1]*Department of Electrical Engineering, Faculty of Intelligent Electrical and Information Technology (ELECTICS), Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia*
[2]*Department of Computer Engineering, Faculty of Intelligent Electrical and Information Technology (ELECTICS), Institut Teknologi Sepuluh Nopember, Surabaya 60111, Indonesia*
[3]*Department of Electrical Engineering, Faculty of Engineering, Universitas Negeri Yogyakarta, Yogyakarta 55281, Indonesia*
[4]*Department of Electrical Engineering, Petra Christian University, Surabaya 60236, Indonesia*
[5]*University Center of Excellence on Artificial Intelligence for Healthcare and Society (UCE AIHeS), Surabaya, Indonesia*
* Corresponding author's Email: hery@ee.its.ac.id

Abstract: Toxic speech has gained substantial attention, focusing on its detrimental effects and prevalence across online platforms. This phenomenon often exhibits discernible patterns in pronunciation analogous to emotions such as happiness or anger. It has been relatively underexplored in prior studies, which predominantly addressed offensive language, hate speech, and sarcasm without considering their emotional properties. Social media platforms have emerged as spaces where individuals share personal encounters with toxic speech that impacts on their well-being. To address this challenge, our study introduces a novel approach that combines speech and text data within a Long Short-Term Memory (LSTM) framework. Unlike existing methods that primarily focus on text analysis, our approach uniquely integrates both speech and text, thereby enhancing the model's ability to accurately detect toxic content. This multimodal data strategy is such an innovative step forward that it provides a more comprehensive solution to the problem of toxic speech detection. Our collected dataset comprises two-way conversations from online fraud reports and confrontations related to loan scams uploaded on YouTube, conducted in the Indonesian language. The absence of subtitles can emerge any ambiguity of homonyms, so it is required to transcribe the audio content to text. To do this, we used native speakers to make sure the transcription was correct in the Indonesian language of the toxic context. In addition, speech features, such as pitch, intensity, and speaking rate, were utilized alongside text features, including Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF). As a result, validation through F1-score measurement yielded 92.73% for text data and 89.09% for speech data. Our proposed approach provided a substantial improvement of approximately 12%-30% compared to the previous LSTM models. The performance comparison results confirmed that our proposed approach can enhance the accuracy of toxic speech detection.

Keywords: Toxic speech detection, Speech pitch, Speech intensity, Bag-of-words, Term frequency-inverse document frequency, Long short-term memory.

1. Introduction

Toxic speech is acknowledged for its negative impact, causing emotional distress and disrupting social connections [1]. Understanding how toxic speech operates on social media is crucial to deal with its consequences. Generally described as offensive language, toxic speech can potentially lead to self-harm and trigger long-lasting depression [1, 2]. Spotting toxic speech in everyday conversations is not easy [3]. However, by focusing on specific contexts or cases that inherently involve toxic speech, toxic-related instances can be recognized and further automated using Deep Learning model classification.

Our study focuses on online interactions, making data collection from the social media platforms for capturing relevant and accurate insights. Unlike other social media platforms, YouTube is a major hub for video content, which often includes user comments and interactions that can provide valuable insights into online behavior. The platform's popularity and extensive use for both content creation and consumption make it an ideal source for our study on toxic speech detection. Additionally, the variety of content and the engagement it generates allow us to capture a wide range of speech patterns and interactions, making our analysis more comprehensive and representative of the general online discourse in Indonesia [4].

We collected data from YouTube specifically because as of 2024, Indonesia ranks third in the number of YouTube users globally, with approximately 139 million active users. This large user base provides a rich and diverse dataset for analyzing toxic speech. The importance of taking data from YouTube for toxic speech research lies in its vast user base, which includes significant contributions from Indonesia [5]. Analyzing YouTube content allows for a comprehensive understanding of toxic speech in an active online community.

Moreover, scam confrontation videos on YouTube would often follow this scenario: the reporters brought attention to a troubling scenario involving online loan scams. The uploaded YouTube reports on these scams would feature conversations between victims and scammers during confrontations via telephone calls. Our collection of online loan scam reports adheres to Dove's criteria of scam-related phenomena, which is close to investment scams, Ponzi schemes, and potential identity theft [6].

In this case, we specifically selected Indonesia for our study. The reason is that the Central Bureau of Statistics' report indicates a low percentage of loan disbursements to Gross Domestic Product (GDP), suggesting underutilization of its financing capacity [7]. Consequently, the citizens of Indonesia are susceptible to quick online loan schemes. However, the use of financial technology in these online loan schemes makes them vulnerable to scams, leading to confrontations that result in toxic speech from both the victims and the perpetrators.

According to Dove, scam schemes can start in a complex motion [6]. Scammers would initiate random calls to various numbers, sourced from different outlets like the dark web or public hotlines disclosed on social media, targeting potential loan-seekers [8].

Through phishing techniques, they manipulate victims by posing as fake agents representing corporations, convincing them of unnecessary quick loan schemes. Prior to the initiation of these supposed loan schemes, scammers request "administrative funds" to finalize the process. Once victims transfer the funds, the scammers sever all communication by blocking the number [6]. Conversely, scammers may also apply for loans using stolen identities, maintaining a semblance of a steady income in a bank account. After the loan approval, the leaves the actual victim, whose identity was stolen, to deal with debt collectors [9]. This often results in a perplexing conversation between the victim and the debt collector, as the victim is unaware of the loan. According to Hess, varying intonations can impart different meanings to a neutral word, transforming it into toxic language [10].

Nguyen et al. introduced the Vietnamese Constructive and Toxic Speech Detection dataset (UIT-ViCTSD), a crowdsourced and text-based dataset designed to facilitate research in toxic speech detection [11]. Moon et al. contributed to this area by presenting a Korean Corpus for toxic speech detection, consisting of 9,400 manually labeled entertainment news comments sourced from a widely used crowdsourcing media platform [12]. D'Sa et al. explored the use of Bidirectional Encoder Representations from Transformers (BERT) and fastText embeddings to detect toxic speech in online media, achieving an 84% F1-measure during validation [13]. Similarly, Malik et al. employed BERT and fastText embeddings in their toxic speech detection system [14]. Additionally, Lees et al. introduced a crowdsourced toxic speech dataset, "coverttoxicity," although the handling steps in managing crowdsourced datasets remain manual [17].

Even though toxic speech detection has been widely discussed in previous studies, there remains a notable gap in research regarding speech-based methods [13]. Lin et al. corroborated this observation, noting a prevailing focus on text-based solutions in published work on toxic speech detection [15]. The role of voice in conveying toxic speech is significant, encompassing intonation, tone, and nuances closely tied to emotions. Toxic speech often exhibits discernible patterns in pronunciation analogous to emotions such as happiness or anger [15]. This aspect has been relatively underexplored in prior studies, which predominantly addressed offensive language, hate speech, and sarcasm without considering their emotional properties [16].

This study investigates multimodal toxic speech detection using voiced speech from crowdsourced media. The toxic speech detection process involves multiple steps. Lees et al. highlighted nuanced toxicity indicators like microaggression and condescension, forming the criteria for labeling data into "Toxic" and "Non-Toxic" classes [17]. Voice, as a carrier of information, encompasses various features, including speech features such as Fundamental Frequency (F0), denoting mean, range, and standard deviation, speaking rate, Harmonic-to-Noise ratio (HNR), energy, and Mel Frequency Cepstral Coefficients (MFCC) to Linear Predictive Coding (LPC) [18-19].

There exist plenty of Deep Learning models. Despite not specifically for toxic speech detection, many speech and text classification tasks in Natural Language Processing have achieved excellent accuracy using Deep Learning models, including LSTM [20]. One commonly used Deep Learning model is the Recurrent Neural Network (RNN), which is effective in capturing sequential patterns in text data. Variants of RNNs, such as Long Short-Term Memory (LSTM) are often used due to their ability to remember long-range dependencies in text. Studies have demonstrated the effectiveness of LSTMs in this domain, with research showing their superiority in handling contextual information necessary for accurately detecting toxic speech [21-22].

Despite significant advances in toxic speech detection, existing methods face notable limitations. Traditional text-based approaches often overlook the emotional cues present in speech, such as tone, pitch, and intensity, which are critical for accurately identifying toxic speech in real-world conversations. When models rely solely on text data, they tend to miss these nuances, leading to inaccuracies in detection. Although Convolutional Neural Networks (CNNs) have been successful in text and image recognition tasks, they are less effective when handling sequential speech data, where the order of words and vocal patterns play a vital role. Similarly, while Long Short-Term Memory (LSTM) networks excel at capturing long-term dependencies, their performance heavily depends on careful feature selection and hyperparameter tuning.

Additionally, techniques like *MinMaxScaler*, commonly used for normalizing feature values and improving model convergence, can sometimes oversimplify complex speech characteristics, potentially losing important nuances in tone and intensity. To address these challenges, our method combines both speech and text features, applying *MinMaxScaler* for feature scaling alongside Random Forest (RF) for feature importance ranking. This approach enhances the accuracy of toxic speech detection by preserving the most relevant and informative aspects of the data while ensuring proper normalization.

Our approach is distinguished by its integration of multiple modalities (speech and text) collected from real-world online fraud confrontation data. By optimizing the LSTM model with advanced feature selection and hyperparameter tuning, we achieve superior performance compared to traditional methods. This novel combination of multimodal data, feature ranking, and model optimization positions our work as a significant improvement over conventional toxic speech detection technique.

In this study, we resolved the multimodal toxic speech detection research by three-fold contributions:

1. We created a new dataset, featuring voice recordings from online fraud confrontation incidents in Indonesia, all compiled from YouTube. These incidents may indicate toxic speech and deserve further investigation in toxic speech detection. This dataset encompasses multiple modalities, incorporating both speech and transcripted text by Indonesian native speakers. Our dataset can be found in [23].
2. We combined a specific set of respective speech features and text features. We found that basic speech and text features matter a lot, but their combination of use can influence the outcome. Basic speech features include pitch, intensity, and speaking rate. While basic text features include word and character n-grams such as unigram, bigram, and trigram, represented through Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) methods. Based on the many features set on respective data modalities (speech and text), we ranked the importance of each feature set using Random Forest. We hypothesized that ranked feature sets would have a positive correlation with higher accuracy in our toxic speech recognition study.
3. We optimized the learning model Long Short-Term Memory by selecting the best value of hyperparameter, be it numerical or categorical. We also used Random Forest as a meta learner in automatic hyperparameter optimization, which would yield higher accuracy than vanilla LSTM [21].

These contributions offer important improvements over existing methods for detecting toxic speech. First, our new dataset, which includes real-world voice recordings from online fraud confrontations in Indonesia, provides more realistic examples of toxic speech than other datasets. By using both speech and text features, like pitch and intensity in the voice alongside the words being spoken, our model can better recognize toxic speech, especially when the tone matters. Moreover, by systematically ranking the importance of speech and text features using Random Forest, we can ensure that our model addresses on the most relevant aspects. As a result, it can lead to improved classification performance.

Finally, using Random Forest to help fine-tune the LSTM model's settings gives us better results than standard methods. This combination of novel data, feature selection, and optimized LSTM model produces a more robust and effective toxic speech detection system.

The rest of this study is summarized as follows: Section 2 shows some previous studies regarding toxic speech detection, while Section 3 shows our proposed approach to toxic speech detection using speech and text data. From our proposed approach, Section 4 illustrates the results and our findings from it. Finally, the conclusions and future research are provided in Section 5.

2. Related work

Guo in 2022 claimed that from a data analysis perspective in language modeling for the task of recognizing human emotions, humor, stress, and toxic speech in the text-based source becomes highly beneficial [24]. Furthermore, textual-based recognition has received so many pre-trained models for language modeling in the past decade [25]. To identify toxic and hate speech, Rodriguez et al. use textual emotion analysis on social media cases. Their study's objective was to track down and examine the unstructured material from a sample of social media posts with the hostile [26]. However, Rodriguez et al. mentioned no concept of toxic speech. Cao et al. surveyed deep learning techniques that are commonly used in assessing emotion in text data. They also draw attention to the problems and difficulties associated with text emotion recognition [27]. Additionally, Cao et al.'s survey has shown that LSTM and CNN accurately assessed emotions [27].
Li et al. presented the foundations of techniques to automatically recognize spoken languages for computational and phonological purposes [28]. Subsequent years have seen significant improvements in the field of spoken language comprehension [29], many of which have been powered by advances in associated signal-processing fields such as pattern recognition and cognitive science [30]. Studies indicate that while the field has grown in recent years, it seems far from flawless, especially in language characterization [31].

2.1 Convolutional neural network in toxic speech recognition

Deep Convolutional Neural Networks (CNNs) have hierarchical patterns of multiple layers investigated using a completely convoluted 2D data recognition [32, 33]. CNN is structured to handle massively complex data in the form of multiple arrays or tensors. CNN usually manages input data that integrates three simple ideas: local networking, collective weights, and organized pooling in a sequence of interconnected layers. CNN extracted global portrayals of the entire input and collected local features to recognize each component of sequential objects (in this case, textual-based toxic speech).

While convolutional layers recognize a local subset of inputs from the preceding stage, the pooling layer aims to add local features to an even more global representation [34]. Pooling is accomplished by sliding a non-overlapping window over the convolution layer's output to gain a collected value for each window. In addition, all CNN learning weights (and those of a fully connected layer) are computed using the standard backpropagation method, Gradient-Descent Optimization [35, 36]. Sadly, CNN cannot yield the best result by itself in text recognition. Some past work used CNN in text recognition by converting text to image [37], using attention mechanism [38], using encoder mechanism [39], and even using LSTM [40]. CNN is used in many text recognitions including emotion recognition [41, 42], text-based hate speech recognition [43, 44], and so forth.

Recent studies from 2019 to 2024 have demonstrated CNN's effectiveness in text-based toxic speech recognition, achieving excellent accuracies by leveraging various hyperparameters and text features of TF-IDF and Bag of Words, yet these studies primarily focus on text data from social media platforms. D'Sa et al. experimented with CNN and BERT embedding for their toxic speech multiclass classification. They stated that TF-IDF and BoW are largely used as input patterns. In result, they achieved 84% in F1-measure validation [13]. Georgakopoulos et al. explored toxic comment detection using BoW as a text feature with the CNN model, demonstrating superior accuracy at 91.2% [45]. Saif et al. combined LSTM and CNN to detect toxic comments, yielding excellent classification

Received: July 16, 2024. Revised: September 30, 2024.

results, although the features and mechanisms of the layers were not extensively detailed [46].

Malik et al. employed BERT and fastText embeddings in their toxic speech detection system, although with lower result of 82%. Additionally, Malik et al. also worked with both TF-IDF and BoW. Even using the same Deep Learning model and input features, Malik's experiment still lower in accuracy than the work of D'Sa et al. The reason is that their work lost too many data because they completely remove slangs and no translation effort done from slangs to formal words, therefore resulting in dataset that is shallow in context and imbalanced in terms of data count. They have since stated a future work of using data augmentation (Synthetic Minority Oversampling Technique or SMOTE) to alleviate this research gap [14]. We can conclude that even input feature and Deep Learning model choice is important, balanced data in terms of count is also very important for toxic speech recognition research.

Convolutional Neural Networks (CNN) have been widely applied in image processing and text-based tasks. However, CNNs struggle with sequential and time-dependent data, such as speech, where the order of words and tone of voice are crucial. Unlike CNNs, our approach leverages LSTM models that are effective for sequential speech data, thus capturing long-term dependencies in speech patterns.

2.2 Long short-term memory in toxic speech recognition

Although CNN is superb in capturing local and global features, CNN is not suitable for sequential objects because the length of sequential objects varies, and the input data should be configured to a fixed size. A predecessor of LSTM is called Recurrent Neural Network (RNN). It can capture meaningful temporal patterns in sequential speech data. Ultimately, this behavior leads to improved results. The RNN architecture contains hidden layers that preserve the previous input sequence's elements [47].

Despite sequential data modeling performance, RNN is faced with difficulties using traditional backpropagation techniques for data sequence training with higher separation degrees [48]. The Long Short-Term Memory (LSTM) networks overcome this restriction by establishing "gates," which are hidden units that regulate how much information is retained or lost during backpropagation [48]. To increase output, bidirectional RNNs would treat sequential data processing [49]. However, because the complete sequence needs to be available for processing, this kind of technology can obstruct real-time operation.

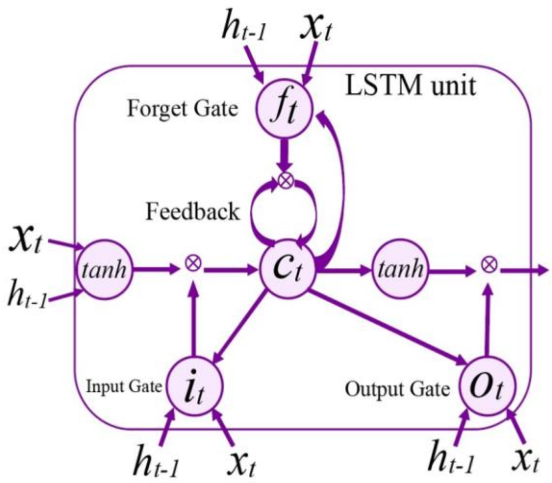

Figure. 1 One Memory Cell of LSTM Unit

Gated Recurrent Unit (GRU) is another LSTM derivate for the context of Neural Machine Translation (NMT). The GRU indicates that it will do plenty to deal with the short sentence of the NMT problems. According to Zuo et al., there are a few different versions in the LSTM, such as GRU, that are contrasted with one another [39]. Greff et al. have identified that the original LSTM structure is better compared to the different recognition tasks [50]. LSTM is used in emotion recognition [51, 52], text-based hate speech recognition [53, 54], generic text detection [55, 56], and so on.

As shown in Fig. 1, an LSTM diagram has an input gate denoted by i_t, and accepts input denoted by x_t which is all independent features by reference, which is an input vector at time t, while $h_{t-1}$is the previous hidden state and pass-through sigmoid function denoted by $tanh$. It then restricts the number between zero and one, if it must discard the previous memory, it will output a vector which will have all zero, this is the work of the forget gate [57].

Previous studies have employed deep learning methodologies, including Recurrent Neural Networks (RNN), Long Short-Term Memory (LSTM), and Convolutional Neural Network (CNN) models, to tackle the challenges of toxic speech classification. Koratana and Hu utilized a GRU-RNN model based on LSTM and RNN paradigms, alongside the Very Deep Convolutional Neural Network (VDCNN), achieving successful toxic speech detection using logistic regression with text word and character n-grams, employing Bag-of-Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) representations [58].

Sutejo and Lestari conducted a similar LSTM-based study, achieving an F1-score of 83.91% in toxic speech detection using TF-IDF and Bag-of-

Words (BoW) as text features [59]. Miok et al. reported robust toxic speech classification using various text features with LSTM, achieving 81% accuracy with TF-IDF [60].

Toxic speech detection studies also leverage speech features. Sutejo and Lestari collected multimodal data, incorporating speech and text from social media platforms, achieving an F1-score of 82.5% using a one-layered LSTM with Time Distributed layer and extracting speech features such as MFCC, INTERSPEECH, and Prosody_Acf [59]. Rakov and Rosenberg reported successful toxic speech detection utilizing speech features, including pitch, intensity, speaking rate, and prosodic contour patterns, achieving an accuracy of 81.57% with K-means clustering and Simple Logistic classification (LogitBoost) [61]. Previous studies have motivated us to leverage the integration of speech features (pitch, intensity, and speaking rate), text features (TF-IDF and BoW), and the application of the LSTM model as they are deemed beneficial for effective toxic speech detection.

LSTM models are a powerful method for handling sequential data, but they are highly sensitive to the selection of features and require extensive tuning to achieve optimal performance. Existing LSTM-based methods often overlook the importance of combining both speech and text features, thereby limiting their ability to fully capture toxic speech. By integrating speech and text features and employing Random Forest for feature importance ranking, this method can improve LSTM performance and guarantee that only the most relevant features are used for classification.

2.3 Feature scaling using MinMaxScaler

Feature scaling, particularly using the *MinMaxScaler*, can significantly impact the performance of LSTM models. LSTM models perform better when input features are scaled. Scaling ensures that all features contribute equally during training, preventing any single feature from dominating the model. When using *MinMaxScaler*, features are transformed to a range between 0 and 1. This normalization helps the LSTM model converge faster and improves its ability to learn from the data [62]. Proper scaling speeds up convergence, allowing the model to learn more efficiently. It also helps avoid issues related to feature magnitude discrepancies.

The *MinMaxScaler* transforms features by scaling each feature to a given range, usually between 0 and 1. The transformation is given by the following equations for standardization in Eq. (1) and scaling to the feature range in Eq. (2).

$$X_{std} = \frac{X - X_{min}}{X_{max} - X_{min}} \tag{1}$$

$$X_{scaled} = X_{std}\,(max - min) + min \tag{2}$$

where X is the input feature, X_{min} and X_{max} are the minimum and maximum values of X respectively. Additionally, *min* and *max* represent the intended range of the transformed data [63].

In the context of toxic speech detection, these features could be various characteristics extracted from the speech signal, such as Mel-Frequency Cepstral Coefficients (MFCCs) [64]. After scaling, these features can be fed into a LSTM model for toxic speech detection $MinMaxScaler$ does not reduce the effect of outliers. It linearly scales them down into a fixed range, where the largest occurring data point corresponds to the maximum value and the smallest one corresponds to the minimum value [63].

MinMaxScaler is an essential preprocessing step in many machine learning tasks. It can ensure that features are scaled to a consistent range, which is important for optimizing model performance. However, one limitation of *MinMaxScaler* is that it can sometimes reduce the variability of more complex features, such as the dynamic range of speech characteristics like pitch and intensity. To overcome this problem, we combine the use of *MinMaxScaler* with a feature-ranking technique using Random Forest. This allows us to retain the most important and informative features such that the scaling process does not negatively impact the detection of toxic speech. By carefully selecting and scaling features, we maintain both the benefits of feature normalization and the richness of the data.

3. Proposed methodology

This study developed several steps, including data collection, feature extraction, and classification, to detect toxic speech using LSTM as well as voice and text transcription. Overall, the study steps are presented in Fig. 2.

3.1 Data collection

The collected data is in the form of two-way real-life conversations [23]. Many have reported their experience of scam confrontation, which largely involves online loan scam cases. They recorded the whole conversation and then uploaded it to YouTube.

The voice was taken from recorded online fraud conversations on YouTube. Victims and fraudsters utter toxic words in online fraud recordings due to anger or annoyance.

Received: July 16, 2024. Revised: September 30, 2024.

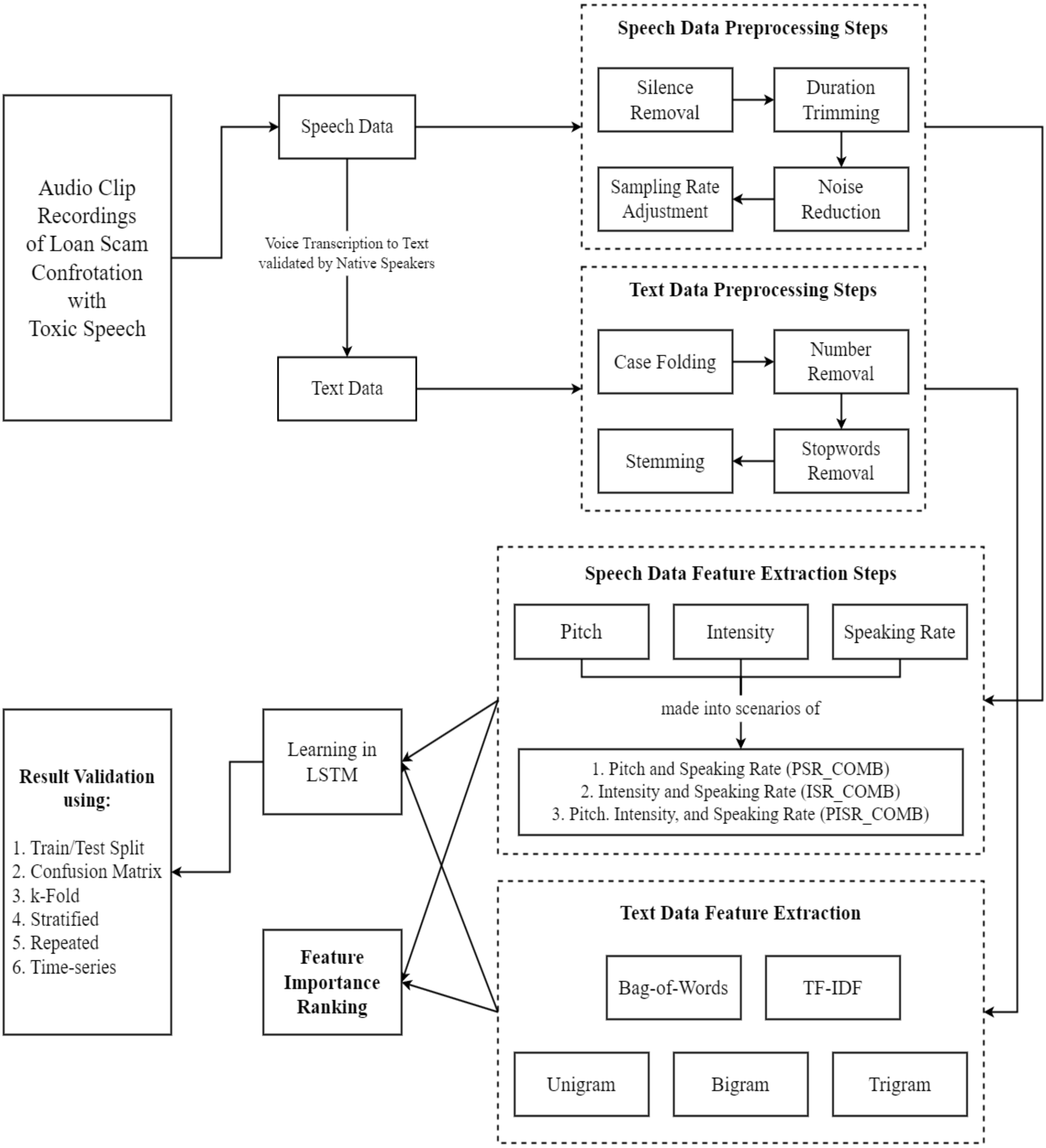

Figure. 2 Proposed Experiment Pipeline

Table 1. Total Instances in Toxic Speech Dataset

Data Type	Toxic	Non-Toxic	Total
Voice/audio	200	200	400
Text (voice/audio transcript)	200	200	400

Since the data had no labels, manual labelling was performed with a class of 0 and 1, where 0 represented Non-Toxic, while 1 meant Toxic. The voice was also transcribed for use as text data. However, the transcription was manual [65] in order to obtain more accurate results. The data instances are presented in Table 1.

3.2 Speech data preprocessing steps

This section covers data pre-processing, which typically follows data collection.

Received: July 16, 2024. Revised: September 30, 2024.

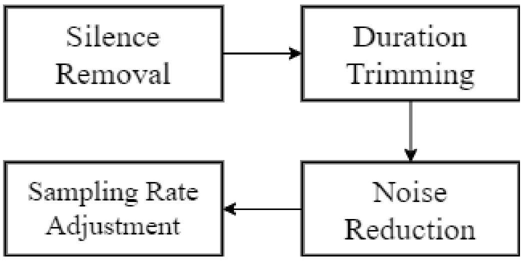

Figure. 3 Speech Data Preprocessing Steps

Raw data had many shortcomings, such as high noise, many silence segments, and too long duration. Subsequently, the voice [66-69] and text [59, 70] data were normalized.

Algorithm 1. Speech Data Preprocessing Pseudocode

input: Voice/audio files (.wav)
output: Preprocessed voice/audio files (.wav)

TrimSilenceSegment: module to trim audio silence segment
ReduceNoise: module to reduce audio noise
TrimDuration: module to trim the audio duration
MoveAudioToPreprocessedDir: function to move preprocessed audio file into preprocessed directory

```
1   while voice/audio file in the audio directory exists do
2       filepath = get original audio filepath
3       outputpath = set output filepath same as original audio filepath
4       duration = get audio duration
5       TrimSilenceSegment(filepath, outputpath)
6       ReduceNoise(filepath, outputpath)
7       if duration > 4 seconds
8           TrimDuration(filepath, outputpath)
9       endif
10      preprocessed_dir = set preprocessed directory location
11      MoveAudioToPreprocessedDir(filepath, preprocessed_dir)
12  endwhile
```

The combination of preprocessing functions makes the accuracy results better than the previous works' result [59]. The features of the preprocessed dataset were extracted for further processing. Fig. 3 shows data pre-processing steps for voice/audio data, while Algorithm 1 show our toxic speech data preprocessing steps in pseudocode.

Algorithm 1 mainly provide the pseudocodes of data pre-processing steps of speech data. Below is the explanation of data pre-processing steps:

- Silence Removal
  Trimming the voice silence segment aimed to avoid empty data segments during feature extraction [59, 71]. The trimming function is shown in Algorithm 1, line 6.
- Duration Trimming
  Trimming duration to a maximum of 4 seconds aimed to obtain the approximate toxic conversation pattern [59]. The trimming function is shown in Algorithm 1, lines 8-10.
- Noise Reduction
  Noise reduction aimed to obtain clearer conversations and improve the features' quality [65]. The noise reduction function is shown in Algorithm 1, line 7.
- Sampling Rate Adjustment
  Adjusting the sampling rate is crucial for ensuring compatibility with the LSTM's requirements. This adjustment involves resampling the audio signals to match the desired rate, such as 16 kHz or 44.1 kHz, which are common in speech processing tasks. This step helps in standardizing the input data and improving the LSTM's performance.

3.3 Text data preprocessing steps

Fig. 4 shows data pre-processing steps for text data, where text data is acquired after the text transcription process. Algorithm 2 primarily presents the pseudocode for the data pre-processing steps of text data. The following is a detailed explanation of these pre-processing steps:

- Case folding
  Converting the entire text to lowercase, known as Case Folding, aimed to avoid possible sensitive cases [59]. By standardizing the text in this way, we reduce complexity and improve the consistency of the input data. The case folding function is shown in Algorithm 2, line 5.
- Number removal
  The numbers in the text were removed and converted into letters [59]. This ensures the text analysis remains focused on words rather than digits, helping reduce dimensionality and making the training process more efficient. The removal function is shown in Algorithm 2, line 6.

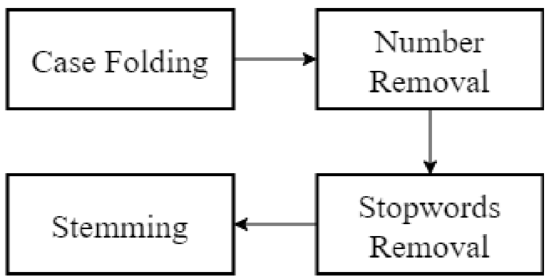

Figure. 4 Transcripted Text Data Preprocessing Steps

- Stopwords removal
  Stopwords with a high text occurrence frequency were removed [70], such as “dan (and)”, “atau (or)”, “tapi (but),” in Indonesia. This process increases the accuracy and speed of training and classification because the text becomes more efficient concerning the number of words. The stopwords removal function is shown in Algorithm 2, lines 7-8.
- Stemming
  Stemming involves changing text words into stem forms to avoid expanding word form patterns [70]. For instance, the words “kemungkinan (possibility)”, “dimungkinkan (will be possible)”, “mungkinkah (is it possible)”, are changed to “mungkin (possible)”. The stemming function is shown in Algorithm 2, lines 9-10.

Algorithm 2. Text Data Preprocessing Pseudocode

input: Voice/audio transcript file (.csv)
output: Preprocessed voice/audio transcript file (.csv)

NumberRemoval: module to convert number in text to word
StopwordsRemoval: module to remove stopwords in text Stemming: module for sentence stemming

```
1   preprocessed_sentences = initial empty array to store new preprocessed sentences
2   while row in the audio transcript file exists do
3       str = original sentence
4       str = str to lower case()
5       str = NumberRemoval(str)
6       sw_factory = load stopword remover factory module
7       str = StopwordsRemoval(str, sw_factory)
8       st_factory = load stemmer factory module
9       str = Stemming(str, st_factory)
10      preprocessed_sentences.push(str) push str into preprocessed sentences
11  endwhile
12  save new preprocessed sentences into preprocessed audio transcript file (.csv)
```

3.4 Speech feature extraction

After the preprocessing step, the speech features (Algorithm 3 lines 4-17) were extracted (as used in [61]), including pitch, intensity, and speaking rate. Pitch, energy, and other speech features were selected because they are closer to the human voice characteristics [72]. Intensity and pitch were normalized by constructing a contour function to reduce variation between voice or audio sessions [73]. The total speech features and the speaking rate were seven features. According to the recommendation of Rakov and Rosenberg [61], all speech features in Algorithm 3 lines 20-23 were divided into a combination of (1) Pitch and Speaking Rate (PSR_COMB); (2) Intensity and Speaking Rate (ISR_COMB); and (3) Pitch, Intensity and Speaking Rate (PISR_COMB). All speech features were extracted using Parselmouth, a Praat implementation in Python [74]. Furthermore, Fig. 5 illustrates extracted speech features in this study.

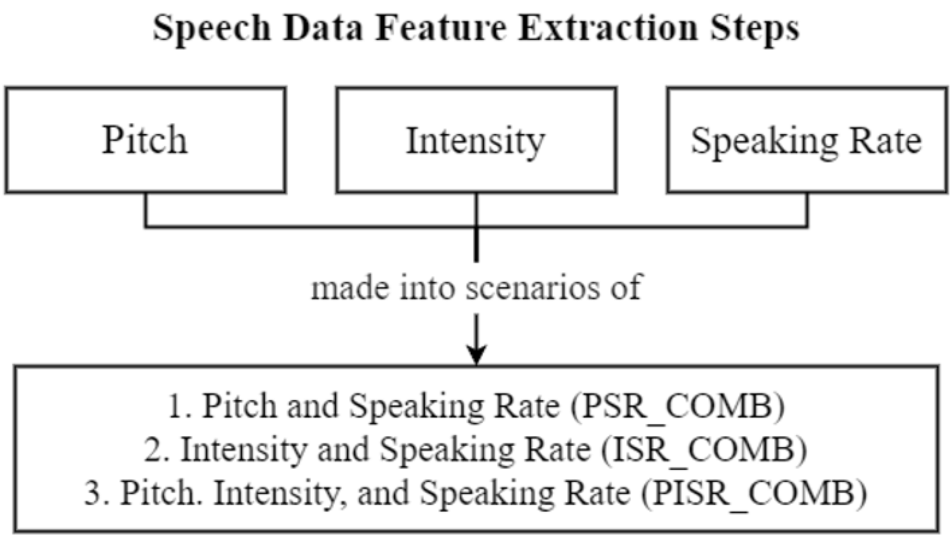

Figure. 5 Speech Feature Extraction Steps

Algorithm 3. Speech Data Preprocessing Pseudocode

```
input: Voice/audio files (.wav)
output: Extracted speech features file (.csv)
parselmouth: parselmouth module
speech_features = initial empty array to store extracted speech features
while voice/audio file in the audio directory exists do
    s = parselmouth.Sound() initialize parselmouth module and read voice/audio file
    ptcs = s.to_pitch() get list of pitchs from the audio
    mp = calculate mean pitch from ptcs
    rp = calculate range pitch from ptcs
    sp = calculate std pitch from ptcs
    ints = s.to_intensity() get list of intensities from the audio
    mi = calculate mean intensity from ints
    ri = calculate range intensity from ints
    si = calculate std intensity from ints
    sprs = load praat source run (parselmouth.praat)
    spr = calculate speaking rate from sprs
    speech _features.push((mp, rp, sp), (mi, ri, si), spr) push and store the extracted features
endwhile
Save speech features data into extracted file (.csv)
PSR_COMB = format (mp, rp, sp, spr) from extracted features
ISR_COMB =. format (mi, ri, si, spr) from extracted features
PISR_COMB = format (mp, rp, sp, mi, ri, si, spr) from extracted features
```

3.5 Text feature extraction

The text features were extracted from the transcription by first using word n-grams consisting of unigrams, bigrams, and trigrams with BoW and TF-IDF representations. BoW implementation is useful for analysis, classification [45], and TF-IDF [74]. However, the vocabulary representation formed by word n-grams only collects words often appearing in the text [76]. The transcription texts produced were almost all short sentences of between 1 to 5 words.

This study also used character n-grams such as unigrams, bigrams, and trigrams, besides n-grams words. In character n-grams, each feature attribute was a character string considered a bag of character n-grams [77] that capture shorter feature categories [78]. For instance, the 2-grams or bi-gram character of the sentence toxic speech would be extracted to |*to*|, |*ox*|, |*xi*|, |*ic*|, |*c_*|, |*_s*|, |*sp*|, |*pe*|, |*ee*|, |*ec*|, and |*ch*|. Subsequently, BoW & TF-IDF representations produced were more balanced. When the word was 2-grams, the sentence would be extracted into |toxic speech|. The best combination of text features was determined using word and character n-grams because both features are effectively useful for text processing [77-79]. All text features were extracted using *sklearn* in Python.

3.6 Feature importance ranking using random forest

It is important to introduce the decision tree before discussing RF. The decision tree is a straightforward supervised learning algorithm based on the if-then-else rule. It offers strong interpretability and aligns with human intuitive thinking. RF, or Random Forest, is composed of multiple decision trees that are uncorrelated. Each decision tree in the forest independently judges and classifies new input samples during classification tasks, but only once their classification result is obtained. Subsequently, RF determines the final result based on the majority decision among the decision trees.

In essence, RF has two key advantages:

- First, it can effectively balance errors in imbalanced data in multiple classes.
- Second, it provides a means to rank the importance of features or different sets of features.

These characteristics make RF an ideal choice for this paper's explanatory algorithm. Specifically, the Gini coefficient is used to assess each feature's contribution in each tree of the RF. These contributions are averaged and compared to determine the relative importance of features. Additionally, cross-validation of features has been employed to validate the RF results.

With ease, one can construct and use RF models by importing the *RandomForestClassifier* from the *sklearn*, a Python library offering a wide array of Machine Learning algorithms. In this approach, the RF model divides each feature, calculating the decrease in the Gini index for each feature split. The significance of a feature is determined by the magnitude of the reduction in the Gini index post-splitting: the larger the reduction, the more the feature contributes to enhancing the dataset's purity, highlighting its importance [78].

In addition, Gilles Louppe gave a different version in [80]. Instead of counting splits, the actual decrease in node impurity is summed and averaged across all trees. (weighted by the number of samples it splits). In *sklearn*, we implement the importance as described in [81]. It is defined as the total decrease in node impurity (weighted by the probability of reaching that node (which is approximated by the proportion of samples reaching that node) averaged over all trees of the ensemble. However, the implementation of our study remains to be consistent with what Gilles Louppe described.

Parameters are selected to control the behavior of a random forest classifier. The "Bootstrap" parameter dictates whether bootstrap samples are employed when constructing individual decision trees within the RF. When set to False, the entire dataset is utilized for each tree's construction. The *Max_depth* parameter governs the maximum depth of each decision tree within the random forest. Setting it to the same length as the model's sequence length can help mitigate overfitting by restricting the tree's level count. The *n_estimators* parameter determines the number of decision trees within the RF ensemble. In this study, 1,000 estimators are utilized due to the limited quantity of Toxic and Non-Toxic data across both speech and text modalities.

3.7 Toxic speech LSTM model

This study used Long Short-Term Memory (LSTM). LSTM uses memory cells as its hidden layer and classifies data with a long sequence range [82]. The LSTM had a memory cell, an input, an output, and a forget gate [83]. This method was selected because the data extracted was sequential and had a shape input that matched the architectural characteristics of the LSTM. Additionally, LSTM's ability to capture long-term dependencies made it well-suited for modeling the complex patterns present in both speech and text data.

LSTM generally overcomes the vanishing gradient problem in RNN by inserting a gating function into the state, facilitating better sequential data processing.

Table 2. Architecture of Proposed Toxic Speech in LSTM Model

Layer Type	Output Shape	Param #
Input Layer	(None, None)	0
Embedding Layer	(None, 100, 128)	640000
LSTM Layer	(None, None, 64)	49408
Dropout	(None, None, 64)	0
Flatten Layer	(None, None, 64)	0
Dense Layer	(None, 2)	130
Total params : 689,538		
Trainable params : 689,538		
Non-trainable params : 0		

The input gate determines the information stored in the memory cell, while the forget gate determines the previous information that needs removal from the memory cell. The output gate controls the information removed from the hidden state [59]. A total of two LSTM models were created for each feature input, including the Toxic Speech LSTM and Toxic Text LSTM. All of the LSTM models we used in this study were built using *Keras* in Python.

Algorithm 4. Proposed Toxic Speech LSTM Model Pseudocode

input: Sequential data (speech features combination)
output: Classification prediction & validation accuracy

```
1  X, Y = get speech features combination & classes
2  X_train, X_test, Y_train, Y_test = train/test split (X, Y)
3  X_train = scaling (MinMaxScaler)
4  X_test = scaling (MinMaxScaler)
5  X_train = reshape input dimension (samples, 1, 7)
6  X_test = reshape input dimension (samples, 1, 7)
7  model = initialize LSTM model Input → LSTM (50) → Dropout (0.2) → Dense (7) → Dense (1)
8  Fit the model (X_train, val_acc=X_test)
9  Get prediction & validation accuracy
```

The Toxic Speech LSTM model is getting input from speech features of voice pitch, intensity, and speaking rate. In Table 2, the LSTM structure for the Toxic Speech LSTM model used a layer with 50 memory units. Furthermore, we used a 0.2 value for Dropout and a hidden layer with seven neurons according to the number of speech features to reduce overfitting. Our model is running approximately 1,000 epochs (stopped with early stopping scheduler) with Adam optimizer and MSE loss function.

The input layer consisted of seven feature variables from the previous speech feature extraction. LSTM requires 3-dimensional input, comprising samples, time steps, and features. Optimization was made by reshaping input shown in Algorithm 4 lines 5-6, and it became a format |*sample=total data*|, |*time steps=1*| and |*features=7*|.

3.8 Toxic text LSTM model

A Toxic Text LSTM model was used to detect toxic speech based on transcripted text and character n-grams. Table 3 shows that we used one Embedding layer that processes data sequence BoW and TF-IDF.

Table 3. Architecture of Proposed Toxic Text in LSTM Model

Layer Type	Output Shape	Param #
Input Layer	(None, 100)	0
Embedding Layer	(None, 100, 256)	2560000
LSTM Layer	(None, 100, 128)	197120
Time-Distributed Layer	(None, 100, 128)	16512
Flatten Layer	(None, 12800)	0
Dense Layer	(None, 2)	25602
Total params : 2,800,234		
Trainable params : 2,800,234		
Non-trainable params : 0		

This study used optimization of 100 top words, 100 sequence word padding, and 128 batch sizes.

Algorithm 5. Proposed Toxic Text LSTM Model Pseudocode

input: Sequential data (text features combination)
output: Classification prediction & validation accuracy

```
1  X, Y = get text features combination & classes
2  X_train, X_test, Y_train, Y_test = train/test split (X, Y)
3  model = initialize LSTM model Embedding layer → LSTM (embedding output) →
       Dropout (0.2) → TimeDistributed (Dense (LSTM output)) → Flatten()→ Dense (1)
4  Fit the model (X_train, val_acc=X_test)
5  Get prediction & validation accuracy
6  X, Y = get text features combination & classes
```

The classification used an LSTM layer with the input shape and number of memory units from the Embedding layer output and a 0.2 Dropout layer. Furthermore, we added the Time-Distributed layer and Flattened layer before the output layer, as shown in Table 3 and Algorithm 5 lines 3-7. The first Dense used in this study was layered with a Time-Distributed layer to change or reduce the dimensions of the output shape from the LSTM layer and was processed optimally. The second Dense or output layer used sigmoid activation with 1 binary neuron. Since only 0 and 1 classification classes were used, a Flattened layer was added to handle the Time-Distributed layer output. The Flattened layer changed the output dimensions of the previous layer, becoming the optimal sequence dimension in the last Dense or output layer with one binary neuron.

3.9 Hyperparameter optimization using random forest

To ensure high recognition accuracy, hyperparameter optimization is oftentimes a mandatory step before the learning stage. There is a popular hyperparameter search technique, namely the grid search. Sadly, it almost always hardly struggles to adapt to high dimensions [84]. Therefore, a substantial amount of newer studies has concentrated on superior methods, namely Bayesian Optimization and its derivative, namely the Random Forest-based Optimization. The Random Forest-based Optimization follows the sequential version of Bayesian Optimization.

Random Forest-based Optimization is an efficient tool for global optimization of costly black-box functions f . A complete Random Forest-based Optimization stage is defined in Algorithm 6. Random Forest-based Optimization starts by function inquiry f to the h values in an initial space and record $\langle \Psi_i, f(\Psi_1)\rangle_{i=1}^{t}$ as the $\langle input, output\rangle$ result pair.

Algorithm 6. Random Forest-based Optimization Pseudocode

input: Target $\boldsymbol{f^X}$;
limit $\boldsymbol{H}$;
hyperparameter space $\boldsymbol{\Psi}$;
initial space $\langle \boldsymbol{\Psi_1}, \dots, \boldsymbol{\Psi_t}\rangle$
output: Best hyperparameter configuration as $\boldsymbol{\hat{\Psi}}$

for $i+1$ to h
  do y_i ← evaluate $f^X(\Psi_i)$
  for j ← to $h+1$ to H
    do $\mathcal{M}$ ← fit model on performance data $\langle \Psi_i, y_i\rangle_{i=1}^{j-1}$
    select $\Psi_j \in arg\ max_{\Psi \in \Omega}\ \alpha(\Psi, \mathcal{M})$
  end for
  y_j ← evaluate $f^X(\Psi_j)$
end for
return $arg\ min_{\Psi_j \in \Psi_1,\dots,\Psi_T} y_j \xrightarrow{yields} \hat{\Psi}$

Then it fits a probabilistic-based model $\mathcal{M}$ to the previous recorded. Later on, $\mathcal{M}$ is used to select input for Ψ which happened to be evaluating function value from input $\Psi \in \Omega$ through acquisition function of $\alpha(\Psi, \mathcal{M})$. Finally, it evaluates function in Ψ newest input.

3.10 Validation methods

The classification accuracy matrices were systematically calculated and reanalysed by utilising the standard F1-score derived from the LSTM model's accuracy, grounded in the actual and predicted class data [85]. F1-score is calculated according to Eq. (3), where Precision and Recall is formulated in Eq. (4) and Eq. (5).

$$F1 = \frac{2 \times Precision \times Recall}{Precision \times Recall} \tag{3}$$

$$Precision = \frac{TP}{TP+FP} \tag{4}$$

$$Recall = \frac{TP}{TP+FN} \tag{5}$$

The Confusion Matrix (CM) does not assume distributional parameters but only on rough data information from the model created. CM is often used to evaluate the prediction results of deep learning models [86]. Since this study only used Toxic and Non-Toxic classes, CM had two columns that informed the actual class. Moreover, it had two predicted columns that informed the predicted class using LSTM. The *sklearn* in Python was used to produce CM.

This study used the train or test split function for the validation method. The training data set took 70% of the total data, while the test sets took 20, and are used randomly. The experiment results took the F1-score from the train or test split, while the comparison used the Cross-Validation method (k-fold, Stratified, Repeated, and Time Series). This method ensures that the validation of the LSTM model performance is appropriate and does not deviate [87]. The observation value to 5-fold (k=5) was set to balance computational complexity and validation accuracy. Additionally, the mean F1-score in Cross-Validation was set based on the number of observations made, while a heatmap was used to visualize the comparison of validation methods.

4. Result and discussion

Before conducting classification, scaling was performed using the *MinMaxScaler* method to convert the data sequence into a range between 0 - 1. The process was conducted on the features in the Toxic Speech LSTM model shown in the previous section in Algorithm 4 lines 3-4, while the text features were directly inserted into the embedding layer on the Toxic Text LSTM model. In the Confusion Matrix results table, the five columns compiled were TP(0), TP(1), FP(0), FP(1), and score. TP indicated True Prediction, and FP indicated False Prediction. TP(0) was the correct prediction accuracy for the Non-Toxic class, while TP(1) was for the toxic. Furthermore, FP(0) is the false prediction accuracy for the Non-Toxic class, while FP(1) is the false prediction accuracy for the Toxic class. The "Score" column was a calculation of the overall Confusion Matrix accuracy. The following are the experiment and analysis results.

4.1 Result of toxic speech LSTM model

The results showed that PISR_COMB had the best F1-score of 0.8909 or 89.09%, with the lowest MSE score of 0.1429. Meanwhile, PSR_COMB had a performance of about 7% better than ISR_COMB, with an F1-score of 0.8174 or 81.74%. Furthermore, ISR_COMB produced a high MSE score of 0.3571, making the lowest accuracy and performance. However, the F1-score of ISR_COMB still reaches 0.7414 or 74.14%.

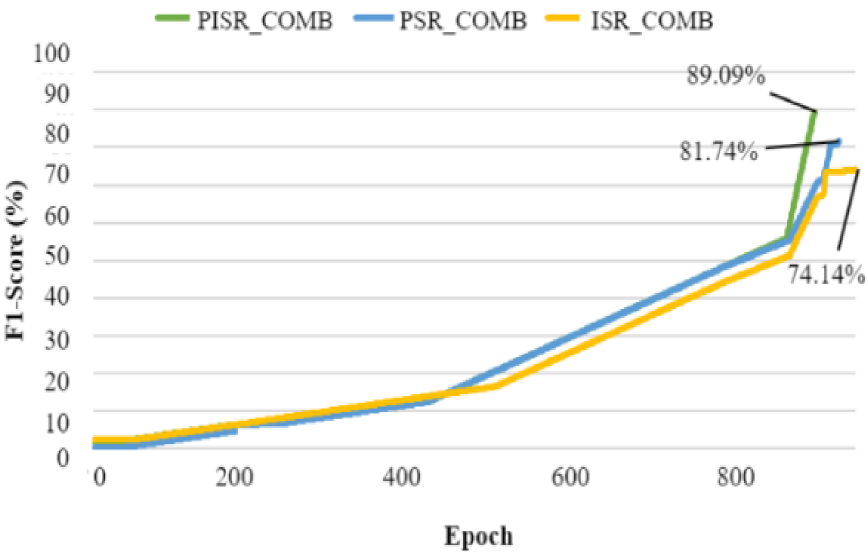

Figure. 6 Graphical Accuracy of F1-Score of Best Result from Toxic Speech LSTM Model per its epoch

Received: July 16, 2024. Revised: September 30, 2024.

Table 3. Result of Toxic Speech LSTM Model

Features	MSE	Recall	Precision	F1-score
PSR_COMB	0.2500	0.8545	0.7833	0.8174
ISR_COMB	0.3571	0.7818	0.7049	0.7414
PISR_COMB	**0.1429**	**0.8909**	**0.8909**	**0.8909**

Table 4. Confusion Matrix Result of Toxic Speech LSTM Model

Features	TP (0)	TP (1)	FP (0)	FP (1)	Score
PSR_ COMB	0.55	0.85	0.45	0.15	0.7500
ISR_ COMB	0.38	0.78	0.62	0.22	0.6429
PISR_ COMB	**0.79**	**0.89**	**0.21**	**0.11**	**0.8571**

In addition, graphical accuracy of F1-score achieved by using PISR_COMB, PSR_COMB, and ISR_COMB feature set is shown in Fig. 6. In Fig. 6, accuracy is depicted per its epoch, which in PISR_COMB case, stopped automatically at 939; in PSR_COMB case, stopped automatically at 971; in PSR_COMB case, stopped automatically by early stopping scheduler technique at 993.

In addition, Table 3 shows the result of Toxic Speech LSTM Model based on MSE, Recall, Precision, and F1-score; while Table 4 shows the Confusion Matrix result of Toxic Speech LSTM Model. The Confusion Matrix validation on the Toxic Speech LSTM model produced features with the best score following the LSTM model classification results. PISR_COMB obtained the best evaluation score of 0.8571 or 85.71%. The accuracy of PISR_COMB on the Confusion Matrix in predicting the Non-Toxic class was 0.79 or 79%, while the correct prediction for the Toxic class was 0.89 or 89%. These results were good because the wrong class predictions were only 0.21 or 21% for Non-Toxic and 0.11 or 11% for Toxic, respectively. PSR_COMB and ISR_COMB received evaluation scores of 0.7500 or 75.00% and 0.6429 or 64.29%, respectively. In this case, PSR_COMB correctly predicted that the toxic class was quite good, reaching 0.85 or 85%, but the correct prediction for the Non-Toxic class was only 0.55 or 55%.

4.2 Result of toxic text LSTM model

Using binary cross-entropy, the Toxic Text LSTM model conducted 50-100 epochs with Adam optimizer. The Toxic Speech LSTM Model was less in epochs because the range of sequences in the Toxic Text LSTM Model was longer.

Table 5. Result of Toxic Text LSTM Model

Features	MSE	Recall	Precision	F1
BOW_ UNIGRAMS	0.1429	0.8909	0.8909	0.8909
BOW_BIGRAMS	0.2857	0.9636	0.7067	0.8154
BOW_ TRIGRAMS	0.3452	0.9455	0.6667	0.7820
BOW_CHAR_ UNIGRAMS	0.1905	0.8364	0.8679	0.8519
BOW_CHAR_ BIGRAMS	**0.0952**	**0.9273**	**0.9273**	**0.9273**
BOW_CHAR_ TRIGRAMS	0.2262	0.7455	0.8913	0.8119
TFIDF_ UNIGRAMS	0.2976	0.8909	0.7206	0.7967
TFIDF_BIGRAMS	0.2738	0.9818	0.7105	0.8224
TFIDF_ TRIGRAMS	0.3333	0.9818	0.6667	0.7941
TFIDF_CHAR _UNIGRAMS	0.3452	1.0000	0.6548	0.7914
TFIDF_CHAR _BIGRAMS	0.3214	1.0000	0.6707	0.8029
TFIDF_CHAR _TRIGRAMS	0.2857	1.0000	0.6962	0.8209

Table 6. Confusion Matrix Result of Toxic Text LSTM Model

Features	TP(0)	TP(1)	FP(0)	FP(1)	Score
BOW_ UNIGRAMS	0.79	0.89	0.21	0.11	0.8571
BOW_ BIGRAMS	0.24	0.96	0.76	0.04	0.7143
BOW_ TRIGRAMS	0.10	0.95	0.90	0.05	0.6548
BOW_CHAR_ UNIGRAMS	0.76	0.84	0.24	0.16	0.8095
BOW_CHAR_ BIGRAMS	**0.86**	**0.93**	**0.14**	**0.07**	**0.9048**
BOW_CHAR_ TRIGRAMS	0.83	0.75	0.17	0.25	0.7738
TFIDF_ UNIGRAMS	0.34	0.89	0.66	0.11	0.7024
TFIDF_ BIGRAMS	0.24	0.98	0.76	0.02	0.7262
TFIDF_ TRIGRAMS	0.07	0.98	0.93	0.02	0.6667
TFIDF_CHAR _UNIGRAMS	0.00	1.00	1.00	0.00	0.6548
TFIDF_CHAR _BIGRAMS	0.07	1.00	0.93	0.00	0.6786
TFIDF_CHAR _TRIGRAMS	0.17	1.00	0.83	0.00	0.7143

Furthermore, more than 100 epochs do not show significant results on the Toxic Text LSTM model.

Received: July 16, 2024. Revised: September 30, 2024.

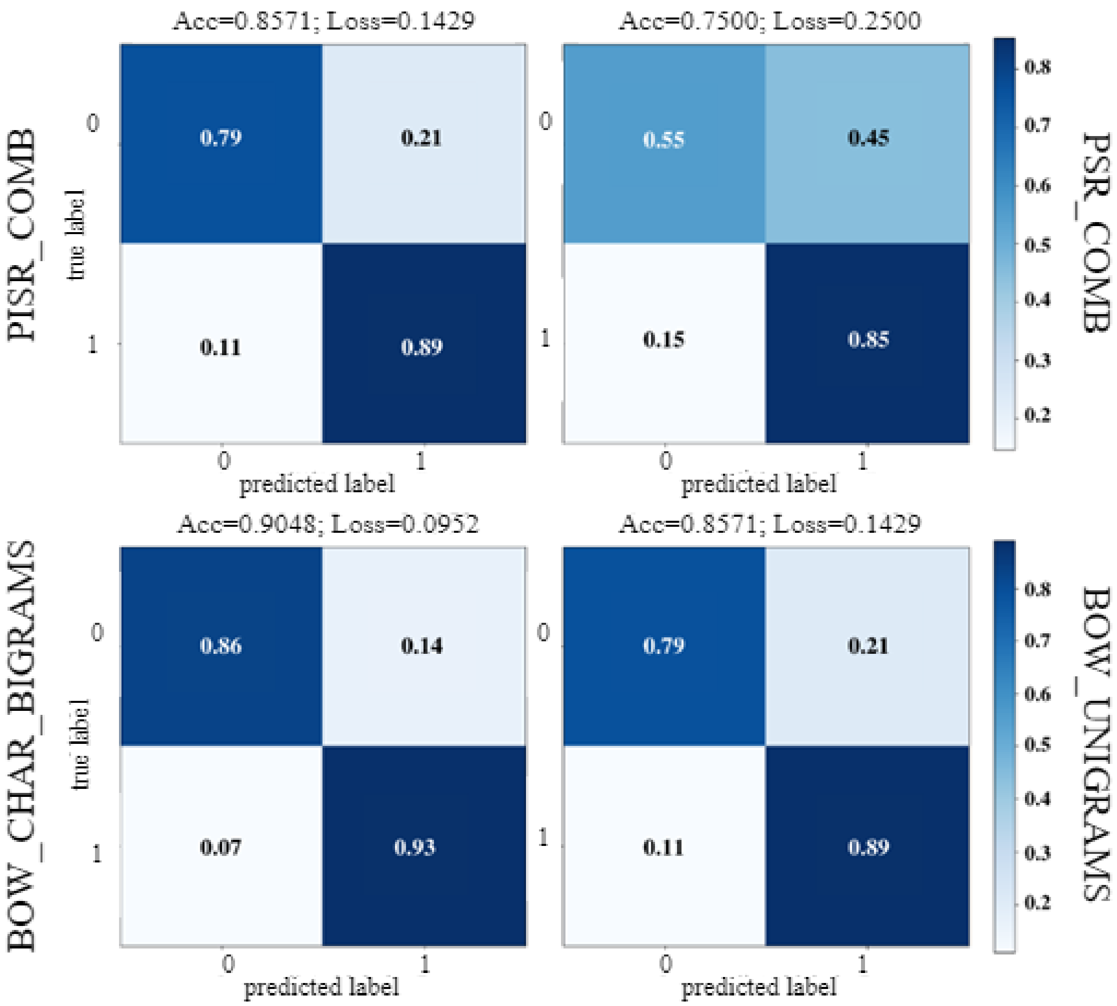

Figure. 7 Graphical Confusion Matrix of Best Result from Toxic Speech LSTM Model (PISR_COMB) and Toxic Text LSTM Model (BOW_CHAR_BIGRAMS)

Table 5 shows the result of Toxic Text LSTM Model based on MSE, Recall, Precision, and F1-score; while Table 6 shows the Confusion Matrix result of Toxic Text LSTM Model. The results show that BoW word unigram performance was about 10%, superior to TF-IDF with an F1-score of 0.8909 or 89.09%.

However, TF-IDF word bigram and trigram performance were better, though it was only a difference of about 1%. For character n-grams, BoW performance was much better by 20% than TF-IDF. BoW bigram's character resulted in the best F1-score of 0.9273 or 92.73%, with a very low MSE of 0.0952. BoW and TF-IDF had the lowest performance for word and char trigrams compared to unigram and bigram. This suggests that simpler n-gram models may capture the key features needed for toxic speech detection more effectively. Moreover, these results highlight the importance of choosing the right feature representation for different types of input data. Additionally, both graphical confusion matrix of the best result from Toxic Speech LSTM and Toxic Text LSTM is shown in Fig. 7.

The confusion matrix validation on the Toxic Text LSTM model produced features with the best score following the classification results. The character BoW bigram obtained the best evaluation score of 0.9048 or 90.58%, correctly predicting the Non-Toxic class by 0.86 or 86% and the Toxic class by 0.93 or 93% (Fig. 7). This result was good because the wrong class predictions were 0.14 or 14% for Non-Toxic and 0.07 or 7% for Toxic, respectively. The Toxic Text LSTM model experiment found that character n-grams with TF-IDF had overfitting problems. The Confusion Matrix evaluation showed that the correct prediction for the Toxic class was 1.00 or 100%, while for the Non-Toxic was only less than 0.20 or 20%. This means the wrong prediction for the Toxic class was high, ranging from 0.83 or 83% to 1.00 or 100%, which was not recommended.

Moreover, the evaluation of the Cross-Validation method on the Toxic Text LSTM model was quite different.

Received: July 16, 2024. Revised: September 30, 2024.

Table 7. F1-Score Result using Cross-Validation (k=5) in Toxic Speech LSTM Model and Toxic Text LSTM Model

Features	k-Fold	Stratified	Repeated	Time Series
PSR_ COMB	0.8070	0.7928	0.7890	0.8043
ISR_ COMB	0.7679	0.7241	0.7578	0.7416
PISR_ COMB	**0.8155**	**0.8224**	**0.8257**	**0.8276**
BOW_ UNIGRAMS	**0.8571**	**0.8829**	**0.8814**	**0.8200**
BOW_ BIGRAMS	0.7879	0.7874	0.7727	0.7664
BOW_ TRIGRAMS	0.7794	0.7669	0.7634	0.7850
BOW_CHAR_ UNIGRAMS	0.7636	0.7961	0.8000	0.7959
BOW_CHAR_ BIGRAMS	0.7563	0.7458	0.7478	0.7347
BOW_CHAR_ TRIGRAMS	0.8095	0.7840	0.7937	0.7885
TFIDF_ UNIGRAMS	0.8548	0.8095	0.8421	0.7736
TFIDF_ BIGRAMS	0.7969	0.7874	0.8000	0.8174
TFIDF_ TRIGRAMS	0.7794	0.7761	0.7789	0.8174
TFIDF_CHAR _UNIGRAMS	0.7794	0.7737	0.7794	0.7679
TFIDF_CHAR _BIGRAMS	0.7794	0.7737	0.7794	0.8103
TFIDF_CHAR _TRIGRAMS	0.7794	0.7737	0.7794	0.8103

BoW word unigram obtained the best score on the k-fold, Stratified, Repeated, and Time Series (Table 7) methods. Additionally, the Character BoW Bigram only obtained a Cross-Validation score of around 74%, while the TF-IDF character was overfitting in Cross-Validation.

Besides, the evaluation of the Cross-Validation method on the Toxic Speech LSTM model was appropriate. PISR_COMB obtained the best score on the k-fold, Stratified, Repeated, and Time Series methods. Based on the experimental results of the Toxic Speech model, all the combinations of speech features worked well without overfitting.

In addition, graphical accuracy of F1-score achieved by using BOW_UNIGRAMS, BOW_BIGRAMS, and BOW_TRIGRAMS feature set is shown in Fig. 8. In Fig. 8, accuracy is depicted per its epoch, which in BOW_UNIGRAMS case, stopped at 100; in BOW_BIGRAMS case, stopped at 100; and in BOW_TRIGRAMS case, stopped automatically by early stopping scheduler technique at 88.

Next, graphical accuracy of F1-score achieved by using BOW_CHAR_UNIGRAMS, BOW_CHAR_BIGRAMS, and BOW_CHAR_TRIGRAMS feature set is shown in Fig. 9. In Fig. 9, accuracy is depicted per its epoch, which in BOW_CHAR_UNIGRAMS case, stopped automatically by early stopping scheduler technique at 96; in BOW_CHAR_BIGRAMS case, stopped automatically by early stopping scheduler technique at 84; in BOW_CHAR_TRIGRAMS case, stopped at 100.

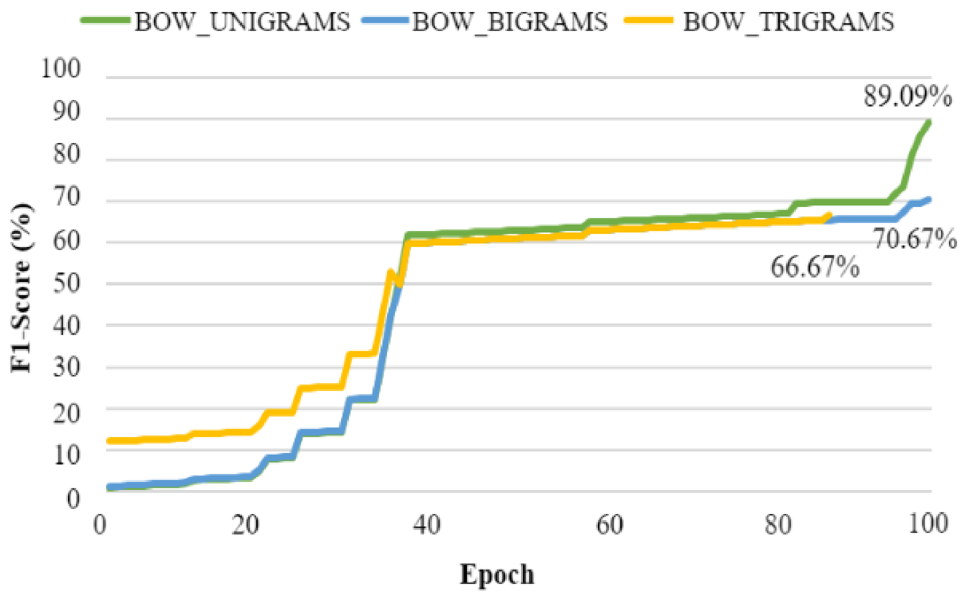

Figure. 8 Graphical Accuracy of F1-Score of Best Result from Toxic Text LSTM Model per its epoch for BOW_UNIGRAMS, BOW_BIGRAMS, and BOW_TRIGRAMS

Received: July 16, 2024. Revised: September 30, 2024.

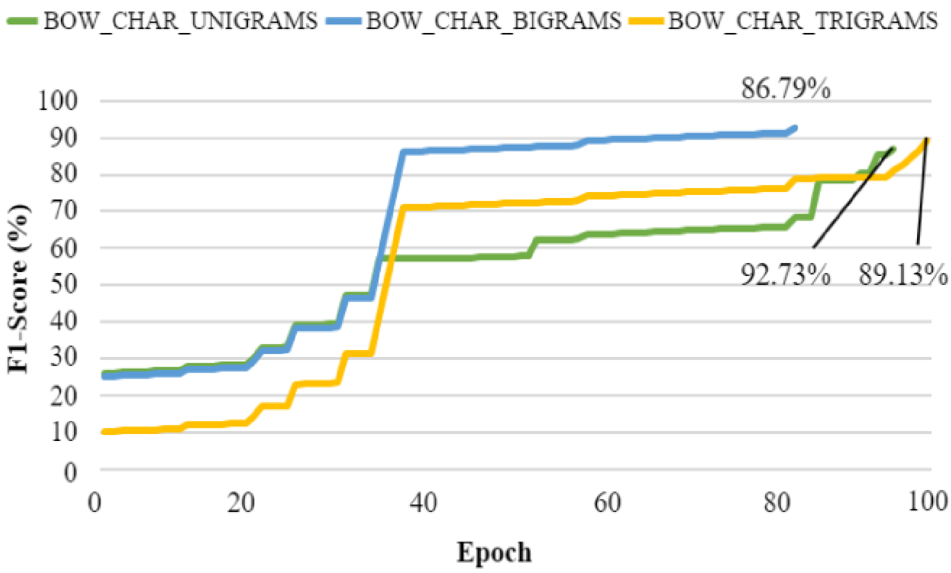

Figure. 9 Graphical Accuracy of F1-Score of Best Result from Toxic Text LSTM Model per its epoch for BOW_CHAR_UNIGRAMS, BOW_CHAR_BIGRAMS, and BOW_CHAR_TRIGRAMS

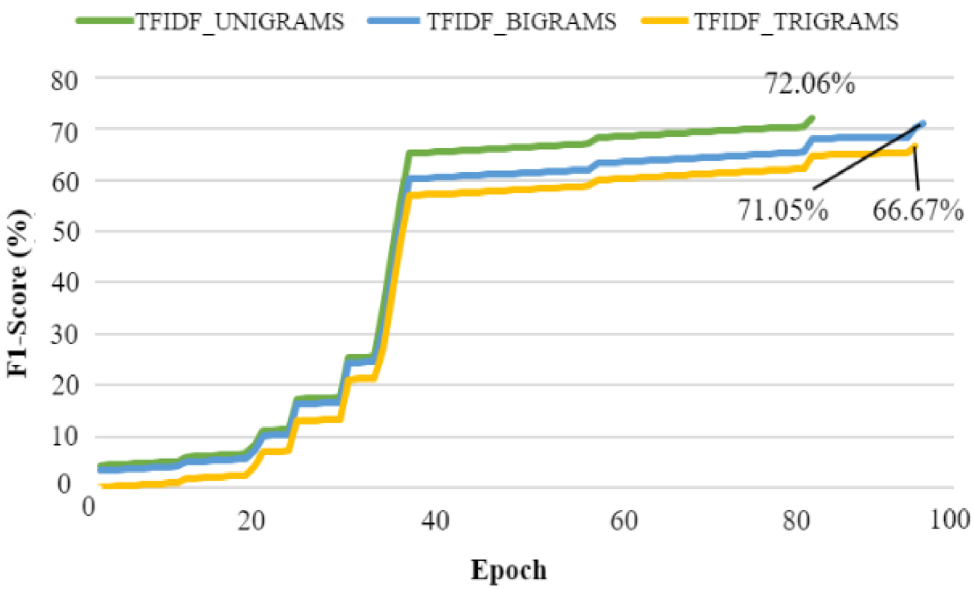

Figure. 10 Graphical Accuracy of F1-Score of Best Result from Toxic Text LSTM Model per its epoch for TFIDF_UNIGRAMS, TFIDF_BIGRAMS, and TFIDF_TRIGRAMS

Another graphical accuracy of F1-score achieved by using TFIDF_UNIGRAMS, TFIDF_BIGRAMS, and TFIDF_TRIGRAMS feature set is shown in Fig. 10. In Fig. 10, accuracy is depicted per its epoch, which in TFIDF_UNIGRAMS case, stopped automatically by early stopping scheduler technique at 84; in TFIDF_BIGRAMS case, stopped automatically by early stopping scheduler technique at 97; in TFIDF_TRIGRAMS case, stopped automatically by early stopping scheduler technique at 96.

The last graphical accuracy of F1-score achieved by using TFIDF_CHAR_UNIGRAMS, TFIDF_CHAR_BIGRAMS, and TFIDF_CHAR_TRIGRAMS feature set is shown in Fig. 11. In Fig. 11, accuracy is depicted per its epoch, which in TFIDF_CHAR_UNIGRAMS case, stopped automatically by early stopping scheduler technique at 94; in TFIDF_CHAR_BIGRAMS case, stopped automatically by early stopping scheduler technique at 99; in TFIDF_CHAR_TRIGRAMS case, stopped automatically by early stopping scheduler technique at 92.

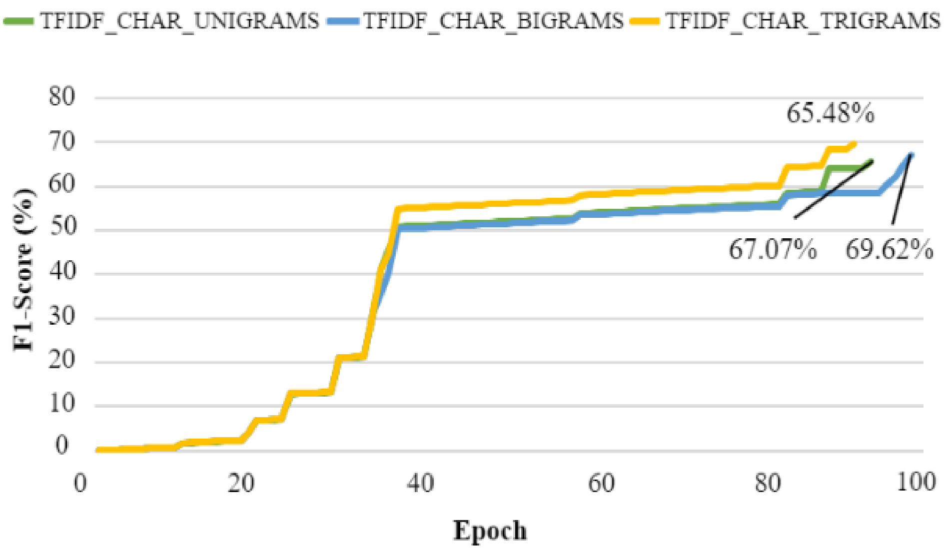

Figure. 11 Graphical Accuracy of F1-Score of Best Result from Toxic Text LSTM Model per its epoch for TFIDF_CHAR_UNIGRAMS, TFIDF_CHAR_BIGRAMS, and TFIDF_CHAR_TRIGRAMS

4.3 Result of feature importance ranking

The Random Forest (RF) model is a robust ensemble learning method comprising numerous decision trees, commonly employed for classification and regression tasks. Notably, RF can assess the significance of features. In this section, following the feature importance ranking by RF, we reorganized the features to train the model at the shortest sequence length. The classification performance is shown in Table 8 below.

Table 8. Importance Ranking based on Different Feature Set in Toxic Speech Recognition

Feature Set	Importance Rank	F1-Score	k-fold Accuracy
PSR_COMB	0.8664	0.8174	0.8070
ISR_COMB	0.8752	0.7414	0.7679
PISR_COMB	**0.8992**	0.8909	**0.8155**
BOW_ UNIGRAMS	**0.9111**	0.8909	**0.8571**
BOW_ BIGRAMS	0.7181	0.8154	0.7879
BOW_ TRIGRAMS	0.7712	0.7820	0.7794
BOW_CHAR_ UNIGRAMS	0.7453	0.8519	0.7636
BOW_CHAR_ BIGRAMS	0.8646	**0.9273**	0.7563
BOW_CHAR_ TRIGRAMS	0.7124	0.8119	0.8095
TFIDF_ UNIGRAMS	0.7767	0.7967	0.8548
TFIDF_ BIGRAMS	0.6544	0.8224	0.7969
TFIDF_ TRIGRAMS	0.8893	0.7941	0.7794
TFIDF_CHAR_ UNIGRAMS	0.8184	0.7914	0.7794
TFIDF_CHAR_ BIGRAMS	0.7866	0.8029	0.7794
TFIDF_CHAR_ TRIGRAMS	**0.9201**	0.8209	0.7794

4.4 Discussion

The experimental results showed that the speech features comprising pitch, intensity, speaking rate, and text features consisting of the word and character n-grams produced the best F1-score of more than 85%. Fig. 12 shows a comparison of the performance of the train or test split, confusion matrix, and Cross-Validation (k=5) using a heatmap.

In addition, Table 9 and Table 10 shows a comparison of the experimental results with several previous studies, for textual data and speech data, respectively. These previous studies are the state-of-the-art speech detection studies with almost the same speech and text features also LSTM model employed.

Mazari et al. employed Bidirectional Encoder Representations from Transformers (BERT) as a pre-trained model, stacking Bidirectional Long-Short Term Memory (BiLSTM) and/or Bidirectional Gated Recurrent Units (BiGRU) on GloVe and fastText word embeddings [88]. However, their approach achieved an F1-score of only 62%. In contrast, Marshan et al. (2023) used a BiLSTM model with various n-gram feature settings, incorporating a feature selection method based on Mutual Information, resulting in a significantly higher F1-score of 88% [89].

Received: July 16, 2024. Revised: September 30, 2024.

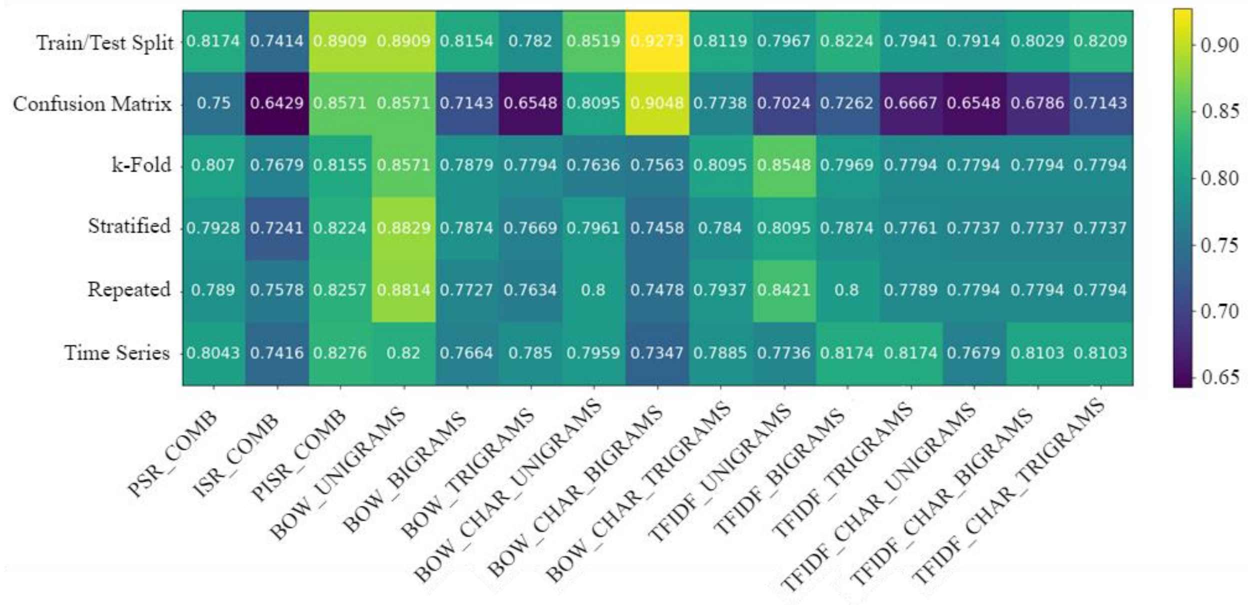

Figure. 12 Graphical Heatmap Result of Train/Test Split, Confusion Matrix, k-Fold, and Cross-Validation (k=5) methods

Table 9. Accuracy Result Comparison for Text Data with Previous Works

Authors	Model and Features	F1-Score
Mazari et al. [88] (2022)	BERT with GloVe and fastText word embeddings	62%
Marshan et al. (2023) [89]	BiLSTM with Text Feature n-gram and feature selection Mutual Information (MI)	88%
Our Proposed Work	**LSTM (Time Distributed and Flatten) + Text Features of BOW_CHAR_BIGRAMS**	**92%**

Table 10. Accuracy Result Comparison for Speech Data with Previous Works

Authors	Model and Features	F1-Score
Islam et al. (2022) [90]	3DCNN + Time Distributed and Flatten + Bi-LSTM with MFCC + Short Time Fourier Transform (STFT) + Chroma STFT	87%
Jacobs et al. (2023) [91]	CAE-RNN with Acoustic Word Embeddings	77%
Our Proposed Work	**LSTM (Speech Features of PISR_COMB)**	**89%**

Despite extensive research on hate speech detection, studies focusing on actual speech datasets (audio recordings) remain relatively rare, as most research targets textual data. Nevertheless, some key studies have explored multimodal approaches that include speech. Islam et al. (2022) proposed a model combining 3D Convolutional Neural Networks (3DCNN), Time Distributed layers, Flatten layers, and BiLSTM. Their method used comprehensive features, including Mel-Frequency Cepstral Coefficients (MFCC), Short-Time Fourier Transform (STFT), and Chroma STFT, achieving an accuracy of 87% [90].

Jacobs et al. (2023) focused on detecting toxicity from radio recordings and utilized a Contextual Autoencoder with RNN (CAE-RNN) for model learning. This represents one of the few state-of-the-art studies on toxic speech detection using LSTM models. Their research, conducted in Swahili, employed Acoustic Word Embeddings (AWE)—analogous to GloVe and fastText but specifically designed for voiced speech—and achieved an F1-score of 77% [91].

Meanwhile in our study, the Toxic Speech LSTM model structure worked well. The combination of speech feature functions PISR_COMB detected toxic speech with the best F1-score of up to 89.09% and the best evaluation of confusion matrix of 85.71%. Additionally, PSR_COMB obtained an F1-score of 81.74%, while ISR_COMB reached an F1-score of 74.14%.

The experimental results also showed that the Toxic Text LSTM model performed better than the Toxic Speech LSTM model in detecting toxicity in a conversation. The Toxic Text LSTM model obtained the best F1-score of 92.73% and the best confusion matrix evaluation of 90.48% using the bigram character (BoW). Furthermore, the F1-score of

89.09% and the best Cross-Validation score of around 88% were obtained using word unigram.

5. Conclusion

This study detected toxic speech using speech features and text features. We developed the Toxic Speech LSTM model and Toxic Text LSTM models for toxic classification. The best accuracy with an F1-score of 89.09% and a confusion matrix of 85.71% was obtained on the Toxic Speech LSTM model using PISR_COMB comprising pitch, intensity, and speaking rate. In the Toxic Text LSTM model, the LSTM model constituting the Time-Distributed and Flattened layers and adjusted batch size, and input shape was optimized and obtained the best accuracy. The results were an F1-score of 92.73% and a confusion matrix of 90.48%, using BoW or bigram characters. The Cross-Validation best score was around 88% using BoW or word unigram. Based on the Toxic Text LSTM model result, character bigram and word unigram performed better than other combinations of n-grams.

We suggest that further studies should examine toxic speech detection using PISR_COMB speech features comprising pitch, intensity, and speaking rate, and text features consisting of the word and character n-grams using the LSTM method. Studies on text features from transcription text with short sentence variations should use bigram characters and word unigrams. Random Forest as the feature importance ranking method can also be further utilized as the reasoning behind using text and/or speech feature set, especially in a recognition task.

Nomenclature

Term	Definition
AWE	Acoustic Word Embeddings
BERT	Bidirectional Encoder Representations from Transformers
BoW	Bag-of-Words
CAE	Contextual Auto Encoder
CNN	Convolutional Neural Network
GDP	Gross Domestic Product
GRU	Gated Recurrent Unit
HNR	Harmonic-to-Noise Ratio
ISR_COMB	Intensity and Speaking Rate Combination of Speech Features
LPC	Linear Predictive Coding
LSTM	Long Short-Term Memory
MFCC	Mel-Frequency Cepstral Coefficients
NMT	Neural Machine Translation
PISR_COMB	Pitch, Intensity, and Speaking Rate Combination of Speech Features
PSR_COMB	Pitch and Speaking Rate Combination of Speech Features
RF	Random Forest
RNN	Recurrent Neural Network
sklearn	Scikit-Learn
tanh	Hyperbolic Tangent Function
TF-IDF	Term Frequency-Inverse Document Frequency
UIT-ViCTSD	University of Information Technology - Vietnamese Constructive and Toxic Speech Detection
VDCNN	Very Deep Convolutional Neural Network

Conflicts of Interest

The authors declare no conflict of interest.

Author Contributions

Agustinus Bimo Gumelar: conceptualization of study and methodology. **Eko Mulyanto Yuniarno:** data curation and preparation. **Derry Pramono Adi:** writing—review and editing. **Arif Nugroho:** formal analysis and practical methodology. **Indar Sugiarto:** supervision and theoretical methodology. **Mauridhi Hery Purnomo:** original draft and supervision.

Acknowledgments

This work is supported by the Kementerian Riset dan Teknologi (Indonesian Ministry of Research and Technology) and the Badan Riset dan Inovasi Nasional (Indonesian National Research and Innovation Agency) by providing the research grant with contract number 848/PKS/ITS/2021.

References

[1] L. Tirrell, "Toxic speech: Toward an epidemiology of discursive harm", *Philosophical Topics*, Vol. 45, No. 2, pp. 139-161, 2017, doi: 10.5840/philtopics201745217.

[2] S. Arabi, *The Highly Sensitive Person's Guide to Dealing with Toxic People: How to Reclaim Your Power from Narcissists and Other Manipulators*, New Harbinger Publications, 2020.

[3] N. Babakov, V. Logacheva, and A. Panchenko, "Beyond Plain Toxic: Building Datasets for Detection of Flammable Topics and Inappropriate Statements", *Lang Resour Eval*, pp. 1-46, 2023.

[4] A. B. Gumelar, D. P. Adi, E. Setiawan, A. Widodo, M. Y. T. Sulistyono, and others, "Machine Learning Performance Comparison for Toxic Speech Classification: Online Payday

Loan Scams in Indonesia", In: *Proc. of 2020 International Seminar on Application for Technology of Information and Communication (iSemantic)*, pp. 603-608, 2020.

[5] R. Shewale, "YouTube Statistics For 2024 (Users, Facts & More)", *Demand Sage*, 2024, Available: https://www.demandsage.com/youtube-stats/

[6] M. Dove, *The Psychology of Fraud, Persuasion and Scam Techniques: Understanding what Makes Us Vulnerable. Milton: Taylor & Francis Group*, 2020.

[7] PricewaterhouseCoopers, PwC Indonesia - Fintech Series, *Indonesia's Fintech Lending: Driving Economic Growth Through Financial Inclusion*, 2019. Available: https://www.pwc.com/id/en/fintech/PwC_FintechLendingThoughtLeadership_ExecutiveSummary.pdf

[8] H. Tu, A. Doupé, Z. Zhao, and G.-J. Ahn, "Users Really Do Answer Telephone Scams", In: *Proc. of the 28th USENIX Conference on Security Symposium*, *Santa Clara*, CA: {USENIX} Association, pp. 1327-1340, 2019.

[9] S. Mbarek and D. Trabelsi, "Crowdfunding without Crowd-fooling: Prevention is Better than Cure", *Corporate Fraud Exposed*, Emerald Publishing Limited, 2020.

[10] L. Hess, "Slurs and Expressive Commitments", *Acta Analytica*, Vol. 36, No. 2, pp. 1-28, 2020, doi: 10.1007/s12136-020-00445-x.

[11] L. T. Nguyen, K. Van Nguyen, and N. L.-T. Nguyen, "Constructive and Toxic Speech Detection for Open-Domain Social Media Comments in Vietnamese", *Advances and Trends in Artificial Intelligence. Artificial Intelligence Practices 2021, Lecture Notes in Computer Science*, Vol 12798, pp 572–583, 2021, doi: 10.1007/978-3-030-79457-6_49.

[12] J. Moon, W. I. Cho, and J. Lee, "BEEP! Korean Corpus of Online News Comments for Toxic Speech Detection", In: *Proc. of the Eighth International Workshop on Natural Language Processing for Social Media*, 2020, doi: 10.18653/v1/2020.socialnlp-1.4.

[13] A. G. D'Sa, I. Illina, and D. Fohr, "BERT and fastText Embeddings for Automatic Detection of Toxic Speech", In: *Proc. of 2020 International Multi-Conference on: "Organization of Knowledge and Advanced Technologies" (OCTA)*, pp. 1-5, 2020. doi: 10.1109/OCTA49274.2020.9151853.

[14] P. Malik, A. Aggrawal, and D. K. Vishwakarma, "Toxic Speech Detection using Traditional Machine Learning Models and BERT and fastText Embedding with Deep Neural Networks", In: *Proc. of 2021 5th International Conference on Computing Methodologies and Communication (ICCMC)*, pp. 1254-1259, 2021. doi: 10.1109/ICCMC51019.2021.9418395.

[15] W.-C. Lin and D. Emmanouilidou, "Toxic Speech and Speech Emotions: Investigations of Audio-based Modeling and Intercorrelations", In: *Proc. of 2022 30th European Signal Processing Conference (EUSIPCO)*, pp. 115-119, 2022. doi: 10.23919/EUSIPCO55093.2022.9909856.

[16] H. Madhu, S. Satapara, S. Modha, T. Mandl, and P. Majumder, "Detecting offensive speech in conversational code-mixed dialogue on social media: A contextual dataset and benchmark experiments", *Expert Syst Appl*, Vol. 215, p. 119342, 2023, doi: 10.1016/j.eswa.2022.119342.

[17] A. Lees, D. Borkan, I. Kivlichan, J. Nario, and T. Goyal, "Capturing Covertly Toxic Speech via Crowdsourcing", In: *Proc. of the First Workshop on Bridging Human-Computer Interaction and Natural Language Processing, Online: Association for Computational Linguistics*, pp. 14-20, 2021.

[18] K. Han, D. Yu, and I. Tashev, "Speech Emotion Recognition using Deep Neural Network and Extreme Learning Machine", In: *Proc. of 15th Annual Conference of the International Speech Communication Association*, 2014.

[19] S. Debnath and P. Roy, "Audio-Visual Automatic Speech Recognition Using PZM, MFCC and Statistical Analysis", *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 7, No. 2, p. 121, 2021, doi: 10.9781/ijimai.2021.09.001.

[20] Z. B. Nezhad and M. A. Deihimi, "A Combined Deep Learning Model for Persian Sentiment Analysis", *IIUM Engineering Journal*, Vol. 20, No. 1, pp. 129-139, 2019.

[21] K. Dubey, R. Nair, Mohd. U. Khan, and Prof. S. Shaikh, "Toxic Comment Detection using LSTM", In: *Proc. of 2020 Third International Conference on Advances in Electronics, Computers and Communications (ICAECC)*, IEEE, pp. 1-8, 2020, doi: 10.1109/ICAECC50550.2020.9339521.

[22] Z. Zhang, D. Robinson, and J. Tepper, "Detecting Hate Speech on Twitter Using a Convolution-GRU Based Deep Neural Network", In: *Proc. of the Semantic Web ESWC 2018, Lecture Notes in Computer Science*, Vol 10843, pp. 745–760, 2018, doi: 10.1007/978-3-319-93417-4_48.

[23] A. B. Gumelar, E. M. Yuniarno, A. Nugroho, D. P. Adi, I. Sugiarto, and M. H. Purnomo, “Indonesian Toxic Speech Dataset (IndoToxSpeech)”, *IEEE Dataport*, 2024, doi: 10.21227/dbgb-j630.
[24] J. Guo, “Deep Learning Approach to Text Analysis for Human Emotion Detection from Big Data”, *Journal of Intelligent Systems*, Vol. 31, No. 1, pp. 113-126, 2022, doi: 10.1515/jisys-2022-0001.
[25] A. F. Adoma, N.-M. Henry, and W. Chen, “Comparative Analyses of BERT, RoBERTa, DistilBERT, and XLNet for Text-Based Emotion Recognition”, In: *Proc. of 2020 17th International Computer Conference on Wavelet Active Media Technology and Information Processing (ICCWAMTIP)*, IEEE, pp. 117-121, 2020. doi: 10.1109/ICCWAMTIP51612.2020.9317379.
[26] A. Rodriguez, Y.-L. Chen, and C. Argueta, “FADOHS: Framework for Detection and Integration of Unstructured Data of Hate Speech on Facebook Using Sentiment and Emotion Analysis”, *IEEE Access*, Vol. 10, pp. 22400-22419, 2022, doi: 10.1109/ACCESS.2022.3151098.
[27] L. Cao, S. Peng, P. Yin, Y. Zhou, A. Yang, and X. Li, “A Survey of Emotion Analysis in Text based on Deep Learning”, In: *Proc. of 2020 IEEE 8th International Conference on Smart City and Informatization (iSCI)*, pp. 81-88, 2020.
[28] H. Li, B. Ma, and K. A. Lee, “Spoken Language Recognition: from Fundamentals to Practice”, In: *Proc. of the IEEE*, Vol. 101, No. 5, pp. 1136-1159, 2013.
[29] A. A. Alashban, M. A. Qamhan, A. H. Meftah, and Y. A. Alotaibi, “Spoken Language Identification System Using Convolutional Recurrent Neural Network”, *Applied Sciences*, Vol. 12, No. 18, p. 9181, 2022, doi: 10.3390/app12189181.
[30] R. Haeb-Umbach, S. Watanabe, T. Nakatani, M. Bacchiani, B. Hoffmeister, M.L. Seltzer, H. Zen, M. Souden., “Speech Processing for Digital Home Assistants: Combining Signal Processing With Deep-Learning Techniques”, *IEEE Signal Processing Magazine*, Vol. 36, No. 6, pp. 111-124, 2019, doi: https://doi.org/10.1109/MSP.2019.2918706.
[31] E. Lleida and L. J. Rodriguez-Fuentes, “Speaker and Language Recognition and Characterization: Introduction to the CSL Special Issue”, *Elsevier*, 2018.
[32] Y. Gao, Y. Chen, J. Wang, M. Tang, and H. Lu, “Reading Scene Text with Fully Convolutional Sequence Modeling”, *Neurocomputing*, Vol. 339, pp. 161-170, 2019.
[33] N. Das, N. Padhy, N. Dey, S. Bhattacharya, and J. M. R.S. Tavares, “Deep Transfer Learning-Based Automated Identification of Bird Song”, *International Journal of Interactive Multimedia and Artificial Intelligence*, Vol. 8, No. 4, p. 33, 2023, doi: 10.9781/ijimai.2023.01.003.
[34] R. Imbriaco, C. Sebastian, E. Bondarev, and others, “Aggregated Deep Local Features for Remote Sensing Image Retrieval”, *Remote Sens (Basel)*, Vol. 11, No. 5, p. 493, 2019.
[35] J. Kim, H. Kim, and others, “An Effective Intrusion Detection Classifier using Long Short-Term Memory with Gradient Descent Optimization”, In: *Proc. of 2017 International Conference on Platform Technology and Service (PlatCon)*, pp. 1-6, 2017.
[36] K. Lv, S. Jiang, and J. Li, “Learning Gradient Descent: Better Generalization and Longer Horizons”, In: *Proc. of International Conference on Machine Learning*, pp. 2247-2255, 2017.
[37] D. Wang and K. Mao, “Task-generic Semantic Convolutional Neural Network for Web Text-aided Image Classification”, *Neurocomputing*, Vol. 329, pp. 103-115, 2019, doi: 10.1016/j.neucom.2018.09.042.
[38] H. Xie, S. Fang, Z.-J. Zha, Y. Yang, Y. Li, and Y. Zhang, “Convolutional Attention Networks for Scene Text Recognition”, *ACM Transactions on Multimedia Computing, Communications, and Applications*, Vol. 15, No. 1s, pp. 1-17, 2019, doi: 10.1145/3231737.
[39] L.-Q. Zuo, H.-M. Sun, Q.-C. Mao, R. Qi, and R.-S. Jia, “Natural Scene Text Recognition Based on Encoder-Decoder Framework”, *IEEE Access*, Vol. 7, pp. 62616-62623, 2019, doi: 10.1109/ACCESS.2019.2916616.
[40] J. Wang, W. Xu, X. Fu, G. Xu, and Y. Wu, “ASTRAL: Adversarial Trained LSTM-CNN for Named Entity Recognition”, *Knowl Based Syst*, Vol. 197, p. 105842, 2020.
[41] S. K. Bharti, S. Varadhaganapathy, R. K. Gupta, P. K. Shukla, M. Bouye, S. K. Hingaa, A. Mahmoud, “Text-Based Emotion Recognition Using Deep Learning Approach”, *Computational Intelligence and Neuroscience*, Vol. 2022, p. 2645381, 2022, doi: https://doi.org/10.1155/2022/2645381.
[42] E. Hamdi, S. Rady, and M. Aref, “A Convolutional Neural Network Model for Emotion Detection from Tweets”, pp. 337-346, 2019. doi: 10.1007/978-3-319-99010-1_31.

[43] O. E. Ojo, T. T. Hoang, A. Gelbukh, H. Calvo, G. Sidorov, and O. O. Adebanji, "Automatic Hate Speech Detection Using CNN Model and Word Embedding", *Computación y Sistemas*, Vol. 26, No. 2, 2022, doi: 10.13053/cys-26-2-4107.

[44] A. Safaya, M. Abdullatif, and D. Yuret, "KUISAIL at SemEval-2020 Task 12: BERT-CNN for Offensive Speech Identification in Social Media", In: *Proc. of the Fourteenth Workshop on Semantic Evaluation*, pp. 2054-2059, 2020.

[45] S. V Georgakopoulos, S. K. Tasoulis, A. G. Vrahatis, and V. P. Plagianakos, "Convolutional Neural Networks for Toxic Comment Classification", In: *Proc. of the 10th Hellenic Conference on Artificial Intelligence*, pp. 1-6, 2018.

[46] M. A. Saif, A. N. Medvedev, M. A. Medvedev, and T. Atanasova, "Classification of Online Toxic Comments using the Logistic Regression and Neural Networks Models", *AIP Conference Proceedings*, p. 60011, 2018.

[47] S. O. Manoj, A. Kumar, A. K. Dubey, and J. P. Ananth, "An Adaptive Salp-Stochastic-Gradient-Descent- Based Convolutional LSTM With MapReduce Framework for the Prediction of Rainfall", *International Journal of Interactive Multimedia and Artificial Intelligence*, pp. 1-13, 2024, doi: 10.9781/ijimai.2024.01.003.

[48] A. Sherstinsky, "Fundamentals of Recurrent Neural Network (RNN) and Long Short-Term Memory (LSTM) Network", *Physica D*, Vol. 404, p. 132306, 2020.

[49] H. Zhao, S. Zarar, I. Tashev, and C.-H. Lee, "Convolutional-Recurrent Neural Networks for Speech Enhancement", In: *Proc. of 2018 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 2401-2405, 2018.

[50] K. Greff, R. K. Srivastava, J. Koutník, B. R. Steunebrink, and J. Schmidhuber, "LSTM: A Search Space Odyssey", *IEEE Trans Neural Netw Learn Syst*, Vol. 28, No. 10, pp. 2222-2232, 2016.

[51] M.-H. Su, C.-H. Wu, K.-Y. Huang, and Q.-B. Hong, "LSTM-based Text Emotion Recognition Using Semantic and Emotional Word Vectors", *2018 First Asian Conference on Affective Computing and Intelligent Interaction (ACII Asia)*, IEEE, pp. 1-6, 2018, doi: 10.1109/ACIIAsia.2018.8470378.

[52] Y. Yu, X. Si, C. Hu, and J. Zhang, "A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures", *Neural Comput*, Vol. 31, No. 7, pp. 1235-1270, 2019, doi: 10.1162/neco_a_01199.

[53] G. L. De la Peña Sarracén, R. G. Pons, C. E. Muñiz Cuza, and P. Rosso, "Hate Speech Detection using Attention-based LSTM", *EVALITA Evaluation of NLP and Speech Tools for Italian*, Accademia University Press, pp. 235-238, 2018, doi: 10.4000/books.aaccademia.4784.

[54] A. R. Isnain, A. Sihabuddin, and Y. Suyanto, "Bidirectional Long Short Term Memory Method and Word2vec Extraction Approach for Hate Speech Detection", *IJCCS (Indonesian Journal of Computing and Cybernetics Systems)*, Vol. 14, No. 2, p. 169, 2020, doi: 10.22146/ijccs.51743.

[55] M. P. Kantipudi, S. Kumar, and A. Kumar Jha, "Scene Text Recognition Based on Bidirectional LSTM and Deep Neural Network", *Comput Intell Neurosci*, Vol. 2021, pp. 1-11, 2021, doi: 10.1155/2021/2676780.

[56] X. She and D. Zhang, "Text Classification based on Hybrid CNN-LSTM Hybrid Model", In: *Proc. of 2018 11th International Symposium on Computational Intelligence and Design (ISCID)*, pp. 185-189, 2018.

[57] D. Avola, M. Bernardi, L. Cinque, G. L. Foresti, and C. Massaroni, "Exploiting recurrent neural networks and leap motion controller for the recognition of sign language and semaphoric hand gestures", *IEEE Trans Multimedia*, Vol. 21, No. 1, pp. 234-245, 2018.

[58] A. Koratana and K. Hu, "Toxic Speech Detection", *Neural Information Processing System*, p. 9, 2018.

[59] T. L. Sutejo and D. P. Lestari, "Indonesia Hate Speech Detection Using Deep Learning", In: *Proc. of the 2018 International Conference on Asian Language Processing*, IALP 2018, pp. 39-43, 2019, doi: 10.1109/IALP.2018.8629154.

[60] K. Miok, D. Nguyen-Doan, B. Škrlj, D. Zaharie, and M. Robnik-Šikonja, "Prediction Uncertainty Estimation for Hate Speech Classification", In: *Proc. of International Conference on Statistical Language and Speech Processing*, pp. 286-298, 2019.

[61] R. Rakov and A. Rosenberg, " 'Sure, I Did The Right Thing ': A System for Sarcasm Detection in Speech", *Interspeech Proceedings*, pp. 842-846, 2013, doi: https://doi.org/10.21437/interspeech.2013-239

[62] M. Rahimzad, A. Moghaddam Nia, H. Zolfonoon, J. Soltani, A. Danandeh Mehr, and H.-H. Kwon, "Performance Comparison of an

Received: July 16, 2024. Revised: September 30, 2024.

LSTM-based Deep Learning Model versus Conventional Machine Learning Algorithms for Streamflow Forecasting", *Water Resources Management*, Vol. 35, No. 12, pp. 4167-4187, 2021, doi: 10.1007/s11269-021-02937-w.

[63] Z. Peng, Y. Lu, S. Pan, and Y. Liu, "Efficient Speech Emotion Recognition Using Multi-Scale CNN and Attention", In: *Proc. of ICASSP 2021 - 2021 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pp. 3020-3024, 2021, doi: 10.1109/ICASSP39728.2021.9414286.

[64] Y. Zhao and X. Shu, "Speech emotion analysis using convolutional neural network (CNN) and gamma classifier-based error correcting output codes (ECOC)", *Sci Rep*, Vol. 13, No. 1, p. 20398, 2023, doi: 10.1038/s41598-023-47118-4.

[65] K. Yu, M. J. F. Gales, and P. C. Woodland, "Unsupervised training with directed manual transcription for recognising Mandarin broadcast audio", In: *Proc. of International Speech Communication Association - 8th Annual Conference of the International Speech Communication Association*, Interspeech 2007, Vol. 4, pp. 2896-2899, 2007.

[66] A. Tursunov, S. Kwon, and H.-S. Pang, "Discriminating Emotions in the Valence Dimension from Speech using Timbre Features", *Applied Sciences*, Vol. 9, No. 12, p. 2470, 2019.

[67] A. Balakrishnan and A. Rege, "Reading Emotions from Speech using Deep Neural Networks", *CS224S Project Reports, Stanford University, Computer Science Department*, 2017. [Online]. Available: https://web.stanford.edu/class/cs224s/semesters/2024-spring/reports_2017

[68] K. Rajaratnam, K. Shah, and J. Kalita, "Isolated and Ensemble Audio Preprocessing Methods for Detecting Adversarial Examples against Automatic Speech Recognition", *ACL Anthology*, pp. 16-30, 2018, Accessed: Jan. 15, 2024. [Online]. Available: https://aclanthology.org/O18-1002/

[69] F. Dong, G. Zhang, Y. Huang, and H. Liu, "Speech Emotion Recognition Based on Multi-Output GMM and SVM", In: *Proc. of 2010 Chinese Conference on Pattern Recognition (CCPR)*, pp. 1-4, 2010.

[70] M. O. Ibrohim and I. Budi, "Multi-label Hate Speech and Abusive Language Detection in Indonesian Twitter", In: *Proc. of the Third Workshop on Abusive Language Online*, pp. 46-57, 2019.

[71] F. Dong, G. Zhang, Y. Huang, and H. Liu, "Speech Emotion Recognition Based on Multi-Output GMM and SVM", In: *Proc. of 2010 Chinese Conference on Pattern Recognition (CCPR)*, pp. 1-4, 2010.

[72] M. Selvaraj, R. Bhuvana, and S. Padmaja, "Human Speech Emotion Recognition", *International Journal of Engineering & Technology*, Vol. 8, pp. 311-323, 2016.

[73] D. Bone, M.P. Black, Chi-Chun Lee, M.E. Williams, P. Levitt, S. Lee, S. Narayanan., "Spontaneous-speech acoustic-prosodic features of children with autism and the interacting psychologist", In: *Proc. of Interspeech 2012*, 2012, doi: 10.21437/interspeech.2012-307.

[74] Y. Jadoul, B. De Boer, and A. Ravignani, "Parselmouth for Bioacoustics: Automated Acoustic Analysis in Python", *Bioacoustics*, pp. 1-19, 2024.

[75] A. Aziz, Mohd Aizaini Maarof, and Anazida Zainal, "Hate Speech and Offensive Language Detection: A New Feature Set with Filter-Embedded Combining Feature Selection", In: *Proc. of 2021 3rd International Cyber Resilience Conference (CRC)*, Langkawi Island, Malaysia, pp. 1-6, 2012, doi: 10.1109/CRC50527.2021.9392486.

[76] Q. Jin, C. Li, S. Chen, and H. Wu, "Speech Emotion Recognition with Acoustic and Lexical Features", In: *Proc. of 2015 IEEE international conference on acoustics, speech and signal processing (ICASSP)*, pp. 4749-4753, 2015.

[77] I. Alfina, R. Mulia, M. I. Fanany, and Y. Ekanata, "Hate speech detection in the Indonesian language: A dataset and preliminary study", In: *Proc. of 2017 International Conference on Advanced Computer Science and Information Systems*, ICACSIS 2017, Vol. 2018-Janua, No. October, pp. 233-237, 2018, doi: 10.1109/ICACSIS.2017.8355039.

[78] U. Sapkota, S. Bethard, M. Montes, and T. Solorio, "Not All Character N-grams are Created Equal: A Study in Authorship Attribution", In: *Proc. of the 2015 conference of the North American chapter of the association for computational linguistics: Human language technologies*, pp. 93-102, 2015.

[79] R. R. Oliveira and R. F. De Oliveira Neto, "Using character n-grams and style features for gender and language variety classification: Notebook for PAN at CLEF 2017", In: *CEUR Workshop Proc*, Vol. 1866, 2017.

[80] G. Louppe, "Understanding Random Forests", *University of Liège*, 2014, Accessed: Mar. 12, 2024. [Online]. Available:

https://orbi.uliege.be/bitstream/2268/170309/1/thesis.pdf.

[81] A. Perrie, “Feature Importance in Random Forests”, Data Science. [Online]. *Alexis Perrier-Data Science*, 2015. [Online]. Available: https://alexisperrier.com/datascience/2015/08/27/feature-importance-random-forests-gini-accuracy.html

[82] H. Daoud, M. Dhouib, J. Rancati, C. Faron, and A. Tettamanzi, “A Hybrid Bi-LSTM-CRF Model for Sequence Labeling Applied to the Sourcing Domain”, In: *Proc. of PFIA-APIA 2020 -5ème Conférence Nationale sur les Applications Pratiques*, Accessed: Mar. 11, 2024. [Online]. Available: https://inria.hal.science/hal-02932095/document.

[83] Y. Wang, M. Huang, X. Zhu, and L. Zhao, “Attention-based LSTM for Aspect-level Sentiment Classification”, In: *Proc. of the 2016 conference on empirical methods in natural language processing*, pp. 606-615, 2016.

[84] M. Raiaan, S. Sakib, N. Fahad, A. Al Mamun, M. Rahman, S. Shatabda, and M. Mukta, “A systematic review of hyperparameter optimization techniques in Convolutional Neural Networks”, *Decision analytics journal*, pp. 100470-100470, 2024, doi: https://doi.org/10.1016/j.dajour.2024.100470.

[85] S. Thirukumaran and A. F. C. Archana, “Speech Emotion Classification Analysis using Short-term Features”, *J Sci*, Vol. 8, No. 1, 2017.

[86] I. Düntsch and G. Gediga, “Confusion Matrices and Rough Set Data Analysis”, In: *Proc. of International Journal of Physics: Conference Series*, p. 12055, 2019.

[87] H. Moss, D. Leslie, and P. Rayson, “Using J-K-fold Cross Validation to Reduce Variance When Tuning NLP Models”, *ACL Anthology*, pp. 2978-2989, 2018, Accessed: Sep. 24, 2024. [Online]. Available: https://aclanthology.org/C18-1252.

[88] A. C. Mazari, N. Boudoukhani, and A. Djeffal, “BERT-based Ensemble Learning for Multi-Aspect Hate Speech Detection”, *Cluster Comput.*, 2023, doi: 10.1007/s10586-022-03956-x.

[89] A. Marshan, F. N. M. Nizar, A. Ioannou, and K. Spanaki, “Comparing Machine Learning and Deep Learning Techniques for Text Analytics: Detecting the Severity of Hate Comments Online”, *Inf. Syst. Front.*, 2023, doi: 10.1007/s10796-023-10446-x.

[90] M. R. Islam, M. A. H. Akhand, M. A. S. Kamal, and K. Yamada, “Recognition of Emotion with Intensity from Speech Signal Using 3D Transformed Feature and Deep Learning”, *Electronics*, Vol. 11, No. 15, p. 2362, 2022, doi: 10.3390/electronics11152362.

[91] C. Jacobs, N. C. Rakotonirina, E. A. Chimoto, B. A. Bassett, and H. Kamper, “Towards Hate Speech Detection in Low-Resource Languages: Comparing ASR to Acoustic Word Embeddings on Wolof and Swahili”, In: *Proc. of INTERSPEECH 2023*, pp. 436-440, 2023. doi: 10.21437/Interspeech.2023-421.