Applied Mathematics & Information Sciences
An International Journal

http://dx.doi.org/10.18576/amis/180413

Semantic Safeguards: Harnessing BERT and Advanced Deep Learning Models Outperforming in the Detection of Hate Speech on Social Networks

Deema Mohammed Alsekait[1], *Ahmed Younes Shdefat*[2,*], *Zakwan AlArnaout* [2,*], *Nermin Rafiq Mohamed*[3], *Hanaa Fathi*[4], *and Diaa Salama AbdElminaam*[4,5,6,7]

[1] Department of Computer Science and Information Technology, Applied College, Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia
[2] College of Engineering and Technology, American University of the Middle East, Egaila 54200, Kuwait
[3] Department of Educational Fundamentals, Faculty of Physical Education, Sadat City University, Sadat City, 32958, Egypt
[4] Applied Science Research Center, Applied Science Private University, Amman, Jordan
[5] Information Systems Department, Faculty of Computers and Artificial Intelligence, Benha University, Benha, Egypt
[6] Data Science Department, Faculty of Computer Science, Misr International University, Cairo, Egypt
[7] MEU Research Unit, Middle East University, Amman, Jordan

Received: 22 Feb. 2024, Revised: 2 May 2024, Accepted: 16 May 2024
Published online: 1 Jul. 2024

Abstract: This paper presents an innovative approach for hate speech detection on social media platforms utilizing optimized deep learning algorithms. Capitalizing on the strengths of four machine learning algorithms (Decision Trees, Support Vector Machines, Naive Bayes, and K-Nearest Neighbors), two deep learning algorithms (Bidirectional Long Short-Term Memory and Recurrent Neural Networks), and a transformer model (Bidirectional Encoder Representations from Transformers, BERT), this research aims to classify text as hate speech efficiently. By implementing feature extraction techniques—TF-IDF for machine learning models and embedding layers for deep learning and transformer models—we leverage two datasets comprising English tweets from Twitter and Facebook. The results indicate a superior performance of the BERT model, achieving an impressive 95% accuracy on the HSOL dataset and 67% on the HASOC dataset, thus significantly advancing the hate speech detection methodology. This paper's methods and findings enhance the existing body of knowledge and provide a reliable model for improving online social interaction safety. The novelty of our work lies in the comprehensive preprocessing and the application of BERT in this context, marking a significant scientific contribution with practical implications for creating a more inclusive online community.

Keywords: Hate Speech, Transformer Models, Social Media Text Analysis , Deep Learning, Transformer Neural Networks, Bidirectional Encoder Representations from Transformers (BERT), Emotion and Speech Analysis

1 Introduction

In an era dominated by technology, people constantly use digital platforms to share experiences and express their opinions. Although social media undoubtedly has many benefits, it unfortunately provides people with a means to hide behind a layer of anonymity. As individuals engage in discussions across different platforms, the risk of being targeted and exposed to harmful and abusive language significantly increases. This has a detrimental impact on users' emotional and mental well-being [1], ultimately leading to, in some extreme cases, acts of violence and hate crimes. To alleviate these repercussions, several companies, such as YouTube, Facebook, and Yahoo, ban hate speech by deploying algorithmic solutions to identify and distinguish hateful content [3]. These hate speech detection algorithms are crucial for creating a safer online environment and preserving the well-being of individuals interacting over the internet.

There are multiple obstacles associated with hate speech detection. The primary challenge is that hate speech carries context with it and can have different

* Corresponding author e-mail: ahmed.shdefat@aum.edu.kw ; zakwan.alarnaout@aum.edu.kw

interpretations depending on its use. Another challenge is that people cannot reach a consensus on what can be categorized as hate speech, making the creation of a universal machine-learning algorithm that detects it difficult. Furthermore, since the datasets are labeled manually, they generally reflect the opinions of the people who collected or labeled the data [4].

Artificial intelligence has brought about substantial advancements over the years. It has evolved from basic rule-based systems to concepts such as machine learning (ML), and deep learning (DL). Machine learning models rely on data patterns to make predictions, while deep learning, a subset of ML, use complex neural networks to extract features and patterns from complex data automatically. While ML models are effective for various problems, DL models excel at capturing sequential patterns and text classification tasks as they can understand and analyze the contextual dependencies within textual data.

Deep learning models include Recurrent Neural Networks (RNNs), and Long Short-Term Memory (LSTM) networks. RNNs are used in tasks that depend on the order and context of the input data to make predictions. However, when processing long data sequences, they face the vanishing gradient problem, which hinders their ability to effectively capture information from distant parts of the sequence during training. LSTM's architecture addresses this problem, making it better suited for tasks like natural language processing (NLP) and text generation. Transformer models, such as Bidirectional Encoder Representations from Transformers (BERT), are a type of deep learning architecture based on a multi-head attention mechanism that has also made significant achievements in the task of natural language processing.

Our paper addresses the hate speech detection problem by employing a comprehensive approach involving seven distinct models: four ML models, two DL models, and one transformer model. We aim to conduct a comparative analysis to determine the model that yields the best results. These models were applied to two distinct datasets to better assess each model's performance.

To summarise, the following represents our contribution:

- –Extensive data preprocessing of tweets entails removing Twitter labels, URLs, contractions, and emojis, replacing usernames, lemmatization, and converting to lowercase. Punctuation, special characters, and numbers, stop words are also removed.
- –Feature extraction using Term Frequency-Inverse Document Frequency (TF-IDF) for machine learning models and embedding layers for deep learning and transformer models.
- –A thorough comparison evaluating decision trees, Support Vector Machines (SVM), k-nearest Neighbors (KNN), Naive Bayes (NB), Bidirectional LSTM (BiLSTM), Recurrent Neural Networks (RNN), and BERT alongside each other.
- –A comprehensive analysis of results when the models are applied on two different datasets that contain a significantly large number of tweets.

The rest of the paper is structured as follows. Section 2 presents relevant literature and prior work in the field. Section 3 introduces our methodology, encompassing data collection, preprocessing, feature extraction techniques, data splitting, optimization parameters for both Machine Learning (ML) and Deep Learning (DL), as well as the classification based on ML and the DL methods. Additionally, this section covers the implementation of a Transformer-based model. Experimental results are detailed in section 4. Section 5 concludes the paper and provides future work.

2 Related Work

Many approaches of Hate Speech Detection have been explored over the years. In [5], Md Saroar et al. systematically reviewed all of the literature in the hate speech detection area. In addition, [6] surveys several text feature extractions, dimensionality reduction methods, algorithms and techniques, and evaluation methods. Finally, [7] also examines the five basic baseline components of hate speech classification using ML algorithms – data collection and exploration, feature extraction, dimensionality reduction, classifier selection and training, and model evaluation. The goals of this study were to present the critical steps involved in hate speech detection using ML algorithms and display the weaknesses and strengths of each method to guide researchers in the algorithm choice dilemma. For this project, we were only concerned with the results of several algorithms in classifying hate speech. Described below is a summary of the methods used in several papers to classify hate speech:

- –**Machine Learning Methods**
  In [8], compared the performance of Naïve Bayes, SVC, Logistic Regression, Decision Trees, Random Forest, SGD, Ridge, Perceptron, and Nearest Centroids in the Detection of Hate Speech. They evaluated the algorithms across data collected from popular OSNs with a web crawler. The labels of the dataset were hate/not hate. The highest accuracy (97.59%) was achieved with the Complement Naïve Bayes method. In [9] , compared Naïve Bayes, Decision Tree, Multi Level Perceptron (MLP), Support Vector Machine (SVM), and AdaBoost Classifier. These algorithms were run on 4,002 samples of data collected by crawling Twitter. The dataset's labels were hate / not hate In [10] , Compared Naive Bayes, SVM, KNN, Decision Tree, Random Forest, AdaBoost, MLP, and Logistic Regression on 14509 tweets collected by a web

crawler. Each tweet fell into one of the following categories: hate speech, offensive but not hate speech, and neither hate speech nor offensive speech. When used with the support vector machine algorithm, Bigram features best perform with 79% . In [11] , used SVM and a deep learning algorithm to classify comments from Italian pages on Facebook to strong hate, weak hate, no hate. The deep learning algorithm performed better than SVM. In [12] , used logistic regression, naive Bayes, decision trees, random forests, and linear SVMs to classify 24,783 tweets into hate, offensive language, and neither hate nor offensive language. logistic regression with L2 regularization performed best and yielded an accuracy of 91%.

–Deep Learning Methods

In [11], Fabio Del Vigna et al. have used SVM and LSTM to classify comments from Italian pages on Facebook as strong hate, weak hate, and no hate. The LSTM algorithm performed better with an accuracy of 79%. In [13], researchers classified 6,655 tweets into racism, sexism, both, non-hate. They used CNN with multiple embedding layers: Random Vectors, Word2vec, Character n-grams, and Word2vec + character n - n-grams. The Word2vec model without character n-grams achieved the best results of all the compared models, with precision, recall, and F-score values of 85.66%, 72.14%, and 78.29%, respectively. Also using a dataset from Twitter, [14] and [15] use the same dataset of 16k tweets to classify them into racism, sexism, or neutral. [**?**] shows applying, Baseline Methods, DNNs only, and DNNs + GBDT Classifier to classify 16k tweets into l. The best method was LSTM+Random Embedding+GBDT with precision of 93%. On the other hand, [15] uses Ensembling LSTM models for the task and achieved an F-score of 0.9517 for the Neutral class, 0.7084 for the Racism class, and 0.9986 for the Sexism class. In [16], explored a wider range of algorithms to classify Arabic tweets in 3 datasets into misogyny, racism, religious discrimination, abuse, and normal. They used SG-CNN, SG-CNN-LSTM, SG-BiLSTM-CNN, and MUSE for the task. The best accuracy was obtained with the SG-BiLSTM-CNN model (Accuracy = 80%). In [17] explores the LSTM-CNN variation more by applying TextCNN, Bi-GRU-CNN and Bi-GRU-LSTM-CNN to detect Hate Speech in Vietnamese social media text. The Bi-GRU-LSTM-CNN achieved the best performance among the three models with an F score of 70.576%. Furthermore,In [18] proposed 2 NN models, CNN-LSTM and LSTM-CNN, to classify text into positive and negative. The best model was LSTM-CNN at 75.2% accuracy. The most comprehensive literature on comparing deep learning algorithms was by Ryan Ong [19]. He compared 13 different LSTM and CNN variations and produced the highest accuracy with the GRU model (79%).

–Transformer Methods

- –[4] Compared BERT with SVM and ELMO+SVM to classify text into hate or not hate and classify the text as hate, offensive, or profane. BERT + SVM performed better for both tasks, yielding an accuracy of 88.33% for task 1 and 81.57% for task 2.
- –[20] used BERT, RoBERTa, DistilBERT, XLNET, and LSTM with attention to classify text into hate speech, offensive, and neither. The best performer was DistilBERT, with an accuracy of 92%.

3 Methodology

Figure 1 and proposed algorithm illustrate the proposed framework for Hate Speech Detection (HSD), which contains mainly seven significant steps:

- –Data collection
- –Data preprocessing
- –Features extraction techniques
- –Data splitting
- –Optimization parameters for Machine Learning (ML) and Deep Learning (DL)
- –Classification based on ML and the proposed DL
- –Transformer-based model
- –Prediction and evaluation metrics

3.1 Data collection

Some criteria were defined before the collection of our data. Our task focused on English, emoticons containing emojis featuring and hashtags including tweets from social media platforms. Given such criteria, the time constraint, and the limited budget, it was impossible to collect such data from Twitter (especially since the removal of the free plan of the Twitter API). As a result, we focused on 2 distinct public English datasets featuring hate speech data. The data sets are HASOC and Thomas Davidson's Hate Speech and Offensive Language Datasets. The data in those datasets was collected from Facebook and Twitter **Table 1**.

1. **Thomas Davidson's Hate Speech and Offensive Language Dataset (HSOL)** [21]: This dataset was collected in 2017. It includes 24,783 English tweets annotated with hate speech, offensive, and neither labels (1,430 hate speech, 19,190 offensive, and 4,163 neither). The authors used a hate speech lexicon containing words and phrases identified by internet users as hate speech, compiled by Hatebase.org. Using the Twitter API, they searched for tweets containing terms from the lexicon, resulting in a sample of tweets from 33,458 Twitter users. They extracted the timeline for each user, resulting in a set of 85.4 million tweets. From this corpus they took a

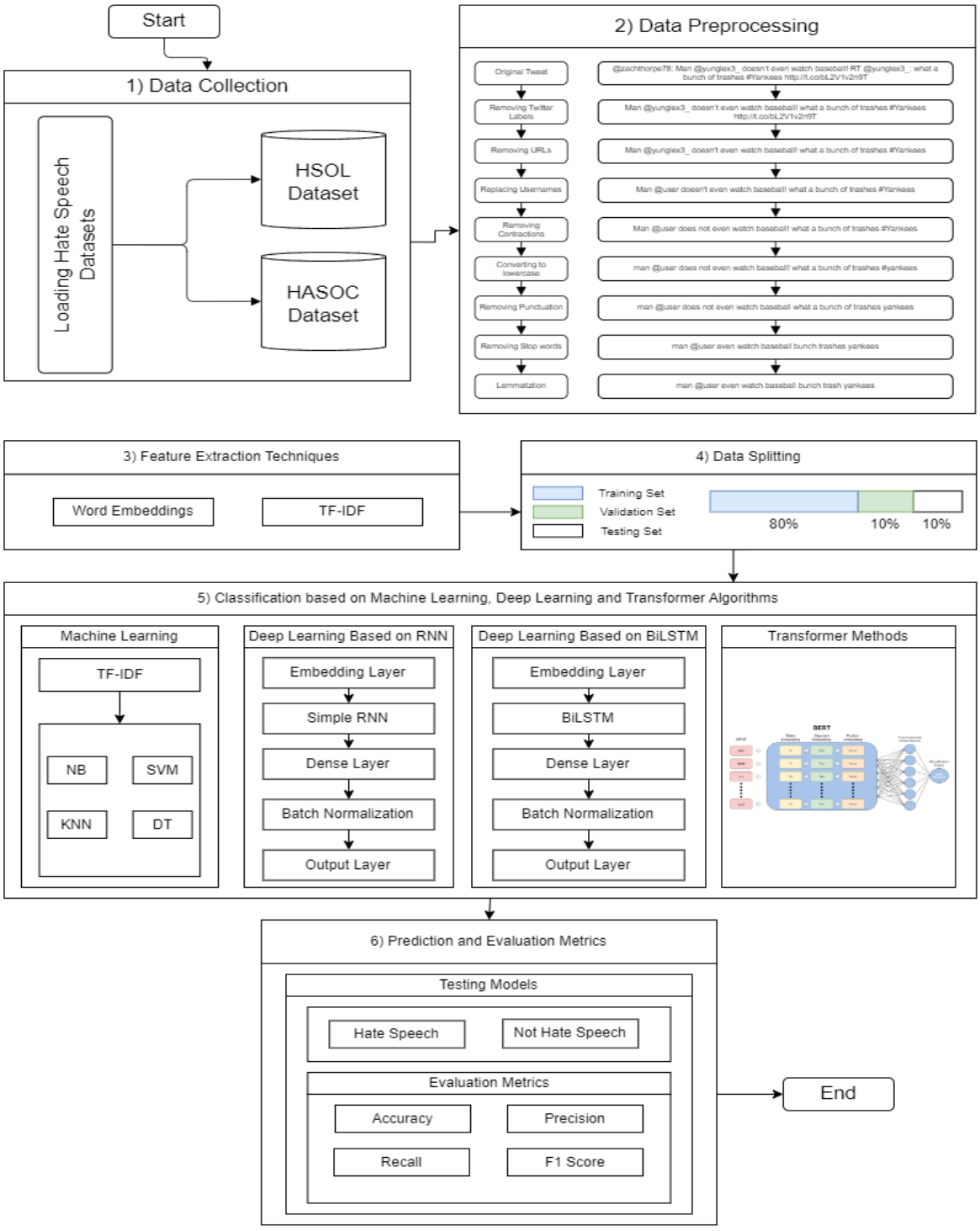


Fig. 1: The Proposed System of Hate Speech Detection.

random sample of about 25k tweets containing terms from the lexicon and had them manually coded by CrowdFlower (CF) workers.

Data: Public English datasets (HSOL, HASOC) featuring hate speech from Twitter and Facebook.
Result: Best performing model and its metrics.

```
// Data Collection
Collect data from public English datasets (HSOL, HASOC) featuring hate speech from Twitter and Facebook
// Data Preprocessing
Remove Twitter-specific labels and URLs
Replace mentions with "@user"
Remove emojis, punctuation, and numbers
Convert all text to lowercase
Remove stop words
Apply lemmatization
// Feature Extraction
Use TF-IDF for Machine Learning models
Use embedding layers for Deep Learning and Transformer models
Calculate TF-IDF for each term in the document
Term Frequency (TF): Calculate the number of times a term appears in the document divided by the total number of terms in the document
Inverse Document Frequency (IDF): Calculate the log of the ratio of the total number of documents to the number of documents containing the term
TF-IDF: Multiply TF by IDF for each term
// Data Splitting
Divide the data into 80% for training, 10% for validation, and 10% for testing
// Hyperparameter Optimization
For ML models: Use Random Search and Grid Search to optimize parameters
Optimize parameters like the number of neighbors (K) for KNN and depth for DT
For DL models: Use Keras-tuner to optimize embedding size, hidden layer units, and dropout rates
// Model Training and Classification
Train Decision Trees, KNN, SVM, Naive Bayes, BiLSTM, RNN, and BERT models on the training set
Validate models on the validation set
// Transformer-Based Model
Implement and fine-tune the BERT model for bidirectional contextual understanding
// Evaluation Metrics
Compute Accuracy (ACC), Precision (PREC), Recall (REC), and F1-Score (F1) for each model on the testing set
ACC: Calculate (True Positives + True Negatives) / (Total Population)
PREC: Calculate True Positives / (True Positives + False Positives)
REC: Calculate True Positives / (True Positives + False Negatives)
F1: Calculate 2 * (Precision * Recall) / (Precision + Recall)
// Select Best Model
Determine the model with the highest performance metrics
return Best performing model and its metrics
```

Algorithm 1: Detailed Hate Speech Detection Methodology

2. **Hate Speech and Offensive Content (HASOC)** [22]: The English Dataset for Hate Speech Detection. During 2019, the dataset was retrieved from Facebook and Twitter. It includes 5,853 posts labeled as Hate Speech and Offensive or Not (2,259 Hate Speech posts, and 3,594 Not Hate Speech or Offensive). This dataset includes emojis, emoticons, retweets, mentions, and hashtags.

3.2 Data Preprocessing

Data preprocessing is a crucial step for any system that deals with text data. The main goal of the preprocessing step is to transform the data into a format easily understood by machine learning models. Various preprocessing techniques have been conducted on the Social Media English datasets before applying the hate speech detection task (i.e., ML/DL techniques). The conducted preprocessing techniques are removing Twitter-specific labels (i.e, "RT" and "VIDEO"), removing URLs, replacing mentions with "@user", removing contractions, converting text to lowercase, removing emojis, removing punctuation (including the "#" sign) and numbers, removing stop words, and finally lemmatization. These preprocessing methods, including cleaning and transformation steps, prepare the datasets to be fitted in the ML/DL models. This removes unnecessary information in the data which helps achieve better results. The preprocessing step includes a series of sub-phases described in the following steps. **Table 2** show the Phases of preprocessing:

- **–Phase 1: Removing Twitter Labels:** In this phase, Twitter-specific labels such as "RT," which means retweet, and "VIDEO," which indicates a video, are removed. Also, the "@username:" that prefixes some tweets, which indicates the author of the tweet, is removed (see **Table 3**)
- **–Phase 2: Removing URLs** In this phase, URLs such as https://www.twitter.com were removed, including URLs using http, https or without an http prefix such as www.twitter.com.
- **–Phase 3: Replacing usernames** In this phase, we replaced all usernames such as "@SomePerson" with "@user", since the twitter handle itself of the user was irrelevant to weather a tweet is hate speech or not. It was decided not to remove hashtags since they sometimes provide important information in the text, but the hashtag symbols "#" were removed.
- **–Phase 4: Removing Contractions** In this phase, contractions like (I'll and don't) were replaced with their normal form (I will and do not).
- **–Phase 5: Converting to Lowercase** In this phase, all characters in the tweets were converted to lowercase to prevent the model from treating the same word with different casing as different words.

Table 1: The used Hate Speech Datasets.

Dataset Name	Year	Dataset Size	Hate Speech	Non-Hate Speech	Source
HSOL	2017	5,593	1,430	4,163	Twitter
HASOC	2019	5,853	2,259	3,594	Twitter, Facebook

Table 2: tweets pre-processing examples

Original Tweet	@zachthorpe78: Man @yunglex3_ doesn't even watch baseball! RT @yunglex3_: what a bunch of trashes #Yankees http://t.co/bL2V1v2n9T
After removing twitter labels	Man @yunglex3_ doesn't even watch baseball! what a bunch of trashes #Yankees http://t.co/bL2V1v2n9T
After removing URLs	Man @yunglex3_ doesn't even watch baseball! what a bunch of trashes #Yankees
After replacing usernames	Man @user doesn't even watch baseball! what a bunch of trashes #Yankees
After removing contractions	Man @user does not even watch baseball! what a bunch of trashes #Yankees
After converting to lowercase	man @user does not even watch baseball! what a bunch of trashes #yankees
After removing punctuation	man @user does not even watch baseball what a bunch of trashes yankees
After removing stop words	man @user even watch baseball bunch trashes yankees
After lemmatization	man @user even watch baseball bunch trash yankees

Table 3: Punctuation and special characters removal

Value To Replace	Replace By
.	Null
,	Null
“	Null
‘	Null
&	Null
%	Null
$	Null

–Phase 6: Removing Emojis In this phase, all emojis were removed; it was decided not to replace them with their respective meaning for two reasons. The first was that it was difficult to find a library that would be able to translate all kinds of emojis, and the second was that, in a lot of cases, the emojis were spammed and if we were to replace the emojis with words then the majority of the text in the tweet would be just the description of the emoji and the important context of the tweet would be just a tiny portion which may result in a model whose decision can be heavily biased when a lot of emojis are used.

–Phase 7: Removing punctuation, special characters, and numbers In this phase, all punctuation marks such as (. : “” ; ’), special characters such as ($ % & | _ – ...) and numbers were removed (see **Table 3**). This step also removes emoticons since they're primarily created using punctuation marks and special characters.

–Phase 8: Removing Stop Words In this phase, stop words that don't provide much context to the tweet were removed. Some of the English stop words are : a, the, is, are

–Phase 9: Lemmatization After cleaning the dataset, the lemmatization process was performed to reduce all words to their original form (see **Table 4**).

Table 4: Lemmatization

Token	Lemmatized Token
go	go
goes	go
went	go
gone	go

3.3 Applying Feature Extraction Techniques for ML and DL Models

In this step, we implemented Feature Extraction techniques for Standard ML models and DL/transformer algorithms as shown in the following subsections:

3.3.1 Machine Learning Feature Extraction Method

The Frequency-Inverted Document Frequency (TF-IDF) Feature Extraction Technique using unigram has been used in this step. The TF-IDF is a well-known statistical Feature Extraction Technique used to evaluate the importance of words in a certain document. **Table 5** shows an example of ML Features Extraction using TF-IDF.

In TF-IDF, the weight is determined by Eq.(1), Eq.(2) and Eq. (3):

$$TF(i,j) = \frac{\text{Frequency of term i in tweet } j}{\text{Total number of terms in tweet } j} \quad (1)$$

$$IDF(i,j) = \log\left(\frac{\text{Total number of tweets in the datasets}}{\text{Number of tweets which include } i \text{ term}}\right) \quad (2)$$

$$W(i,j) = TF(i,j) \times \text{IDF}(i,j) \quad (3)$$

Where $TF(i,j)$ is the frequency of term i in review j, $IDF(i,j)$ is the frequency of feature with respect to all reviews. Finally, the weight of feature i in review j , $W(i,j)$ is calculated by Eq.(3).

3.3.2 The DL Feature Extraction Method

In this step, word embeddings were used to convert words in the tweets into vectors that the model can work with. In the DL models, the features were extracted using a Keras word embedding layer. Word embeddings provide an efficient and dense representation for words with similar encoding (the model determines the similarities between words during training). An embedding is a dense vector of floating point values (the length of the vector is a hyperparameter). The embeddings themselves are trainable parameters in the model (just like how the weights in a dense layer are trainable). An embedding with a vector size of 32 was used in the DL models.

3.4 Data splitting

In this step, the data is split into the training set, validation set and testing set using the holdout method (80% training 10% validation, and 10% testing).

3.5 Hyperparameter Optimization Methods

3.5.1 Hyperparameter Optimization for ML algorithms

Optimizing each model's hyperparameters is essential to achieve the best accuracy; we analyzed several hyperparameters of all the ML algorithms. There are several options to perform this task. Random Search [23] and Grid Search [24] are popular methods for tuning hyperparameters. Unfortunately, they both have unavoidable downsides. Grid Search works by exhaustively applying all the possible combinations of all hyperparameters, which causes the search for optimal parameters to take too long. In Random Search, a fixed number of parameter settings is sampled from the specified values for the hyperparameters. The downside of this approach is that it might be difficult to find the global optimum. For this project, we chose to analyze one hyperparameter at a time while using the default values for the rest of the hyperparameters. This helps us get an insight into how every hyperparameter affects the accuracy. Below are the hyperparameters explored for each model:

- **–KNN:**
  - –the number of neighbors (K)
- **–Decision Tree:**
  - –The maximum depth of the tree
  - –The minimum samples needed to split a node into two branches
  - –The maximum number of leaf nodes
  - –The minimum samples in each leaf node
- **–SVM:** None
- **–Naive Bayes:** None

3.5.2 Hyperparameter Optimization for DL algorithms

In this phase, the Keras-tuner library is used to find the best hyperparameters of the embedding layer, the hidden layer, the dense layer, and the dropout rate for English hate speech detection in simple Recurrent Neural Network (RNN) and Bidirectional Long Short-Term Memory (BiLSTM). Additionally, a dropout layer was combined with a hidden layer, and the Softmax activation function and the Adam optimizer are included in the output layer. The embedding size, hidden layer units, dense layer units, and the dropout rate for each data set of the proposed models are optimized using the Keras Tuner library as shown in Table 6

3.6 Classification based on ML models

In this step, four regular ML algorithms, including (DT), (KNN), (SVM), and (NB), were used to classify English tweets as hateful or not.

Table 5: ML Feature Extraction Technique Example using TF-IDF

Words	TF (for A)	TF (for B)	IDF	TF-IDF(A)	TF-IDF(B)
Jupiter	1/5	0	In(2/1) = 0.69	0.138	0
is	1/5	1/8	In(2/2) = 0	0	0
the	1/5	2/8	In(2/2) = 0	0	0
largest	1/5	0	In(2/1) = 0.69	0.138	0
planet	1/5	1/8	In(2/2) = 0	0.138	0
Mars	0	1/8	In(2/1) = 0.69	0	0.086
fourth	0	1/8	In(2/1) = 0.69	0	0.086
from	0	1/8	In(2/1) = 0.69	0	0.086
sun	0	1/8	In(2/1) = 0.69	0	0.086

Table 6: Deep Learning optimal hyperparameters for the three Arabic Tweet Datasets

Datasets	**HASOL**				**HASOC**			
Models	Embedding Size	Hidden Layer Units	Dense Layer Units	Dropout	Embedding Size	Hidden Layer Units	Dense Layer Units	Dropout
RNN	32	16	512	0.5	32	16	512	0.5
BiLSTM	16	64	521	0.5	64	32	521	0.4

–**Naive Bayes (NB)** [25] is a simple yet effective supervised classification algorithm based on Bayes' Theorem of conditional probability with the assumption of independence between features. The model calculates the probability of each class given a set of features using the following equation (4), and assigns the class with the highest probability.

$$P(C|x) = \frac{P(x|C)P(C)}{P(x)} \tag{4}$$

Where
- $P(C|x)$ is the posterior probability of class C given input x,
- $P(C)$ is the prior probability of class C,
- $P(x|C)$ is the probability of the input x given class C,
- $P(x)$ is the prior probability of the input x.

–**K-Nearest Neighbor (KNN)** [**?**] is a simple supervised classification algorithm that depends on one parameter: K. It operates by finding the K nearest data points in the feature space to a given input and makes predictions based on the majority class or average among these neighbors.

–**Support vector machine (SVM)** [27] is a supervised algorithm that works by finding the hyperplane that best separates different classes in the feature space, maximizing the margin between the closest points of each class, known as support vectors.

–**Decision Tree (DT)** [28] is mostly used in supervised ML. It builds a tree-like model of decisions, where each node represents a feature, each branch a decision rule, and each leaf node a class label. The algorithm splits the dataset into subsets based on feature value comparisons, aiming to create as homogeneous subgroups as possible.

3.7 The proposed DL models:

The proposed deep neural network architecture is depicted in **Figure 2**. The developed models classify hate speech from English social media datasets. The input for the models is a vector containing the integer encoded-words from the text. this input is fed into an embedding layer, followed by hidden layers that include a Simple Recurrent Neural Network (RNN) and a Bidirectional Long Short-Term Memory (BiLSTM); the output of those is flattened and then fed into a dense layer, followed by batch normalization, and then finally, an output layer. The models also use dropout to prevent overfitting. A Keras Tokenizer encodes the words and converts tweets into integer vectors. The embedding layer, hidden layer, dense layer, batch normalization, and output layer for each DL model is defined as follows:

–**Embedding Layer**: We used the built-in Embedding layer in the Keras library [29] to implement the embedding layer. The arguments of the Keras embedding layer were configured as follows: 1) input_dim, which defines the size of the dataset's vocabulary, was set to 5000, 2) output_dim which defines the size of the embedding vector, was set to 32, and finally 3) input_length which defines the length of the size of the input vector, was set to 50.

–**Simple Recurrent Neural Network (RNN)**: The first proposed DL model is a simple recurrent neural network. The basic principle in recurrent networks is that the input vector and some information from the previous step (generally a vector) are used to calculate

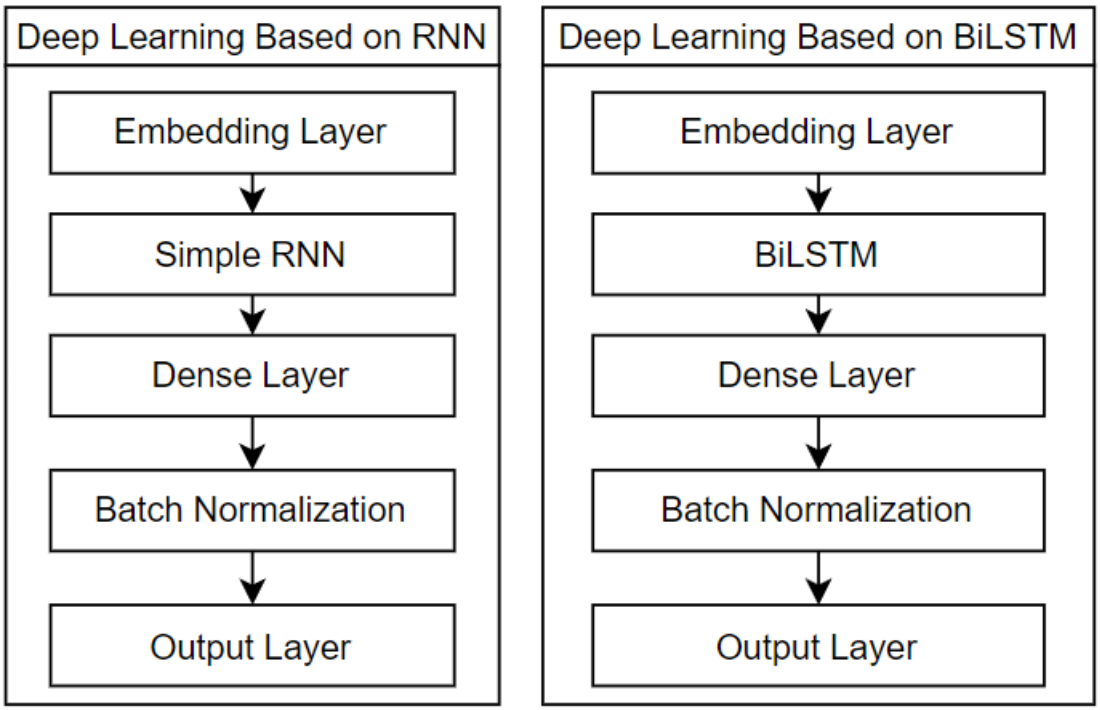


Fig. 2: The Proposed DL models Architecture .

the output and information passed to the next step. The formulas used to calculate the output values in each step are called units. An RNN with 16 units and a tanh activation function was used.

–**Bidirectional Long Short-Term Memory network (BiLSTM)**: The second proposed DL model is a bidirectional LSTM. First a normal LSTM is a special type of RNN, that solves the RNN's problem of vanishing and exploding gradients, also it is better than a normal RNN in recognizing relationships between values at the beginning and the end of the input sequence. An LSTM consists of three gates, which are the forget gate (decides how much information is received from the previous step), the input gate (decides what new information to be stored in the cell state), and the output gate (decides the output from the LSTM cell). The definitions of these gates are shown in the following Equations:

$$f_t = \sigma\left(W_f.[h_{t-1},x_t]+b_f\right) \tag{5}$$

$$i_t = \sigma\left(W_i.[h_{t-1},x_t]+b_i\right) \tag{6}$$

$$\tilde{C}_t = tanh\left(W_C.[h_{t-1},x_t]+b_C\right) \tag{7}$$

$$C_t = f_t * C_{t-1} + i_t * \tilde{C}_t \tag{8}$$

$$o_t = \sigma\left(W_o\left[h_{t-1},x_t\right]+b_o\right) \tag{9}$$

$$h_t = o_t * tanh(C_t) \tag{10}$$

In these equations, the W is the weight matrix, σ is a sigmoid function; f, i, C o h are the forget gate, input gate, cell state, output gate, and the cell output respectively.

A bidirectional LSTM is like a normal LSTM, but the input flows in both directions and can utilize information from both sides of the input vector. This is done by using two LSTM layers where the input sequence flows forward in one layer and backward in the other layer, and the output from both layers is combined.

–**Dense Layer:** The output from the RNN or LSTM is flattened and then fed into a dense layer with 512 neurons with a Relu activation function and an L1 kernel regularizer.

–**Batch Normalization:** Before the result from the dense layer is fed into the output layer, the values are normalized to have a mean close to 0 and a standard deviation close to 1. This makes the network more stable during training, it also allows higher learning rates, leading to an accelerated convergence.

–**Output Layer** The output layer generates the model's final decision. This layer consisted of 2 neurons (one for each verdict). This layer used the softmax

activation function. All DL models used the Adam optimizer and a dropout of 0.5 during training.

3.8 Transformer-Based Model

The pre-trained BERT transformer model is our third approach to the hate speech detection problem. BERT, which stands for Bidirectional Encoder Representations from Transformers, is a transformer encoder stack that is trained on large English corpora. BERT's architecture consists of 12 transformer layers, a feed-forward network with 768 hidden units, and 12 attention heads. Since the corpus BERT is trained on is generic, fine-tuning is necessary to be able to apply the model. During fine-tuning, BERT's parameters are updated when trained on the labeled hate speech datasets.

Each encoder layer in BERT processes the input bidirectionally, capturing contextual information. The classification token [CLS], the output from the last transformer layer, represents the entire input sequence. This output token is passed on to a classification layer which consists of a dense neural network and a softmax activation function. The classification layer is responsible for predicting which class the tweet belongs to. [4]

3.9 Performance metrics

The performance of the suggested models is assessed using four common performance metrics: Accuracy (ACC), Precision (PREC), Recall (REC), and F1-score (F1); they are computed in the manner described below:

$$\text{ACC} = \frac{TP+TN}{TP+TN+FP+FN} \tag{11}$$

$$\text{PREC} = \frac{TP}{TP+FP} \tag{12}$$

$$\text{REC} = \frac{TP}{TP+FN} \tag{13}$$

$$F1 = \frac{2.\text{PREC}}{\text{PREC} + \text{REC}} \tag{14}$$

4 Experimental Results and Discussion

To examine the efficiency of Machine Learning (ML methods), Deep Learning models (DL), and Transformer Methods (T) over Hate Speech Detection, we have assessed two datasets: the Davidson dataset and the HASOC dataset. It is important to highlight that the extraction of features in ML methods (DT, KNN, SVM, and NB) is realized by TF-IDF. However, Deep Learning and Transformer models utilized embedding layers as an input for the three models, i.e., RNN, BiLSTM, and BERT.

4.1 Case Study I (HSOL Dataset)

Table. 7 shows the values of four metrics including Accuracy (ACC), Recall (REC), Precision (PREC) and F1-score (F1). Out of the four machine learning classifiers, SVM represents the best classifier in terms of (ACC, 94%, REC 93%, PREC 90% and F1 91%). This performance is due to the effectiveness of SVM in high-dimensional spaces. The NB classifier obtains the lowest performance with (ACC of 74%, PREC of 69%, REC of 73%, and F1 of 70%). Due to NB's assumption of feature independence, this behavior is evident and logical as it struggles to capture the complex dependencies between features, especially in the context of hate speech detection. By inspecting the results of DL and BERT over the HSOL dataset, the transformer-based model BERT achieved an improvement in accuracy, precision, recall, and f1 score of 1%, 1%, 3%, and 2% respectively. This is because, in comparison to some deep learning models and traditional machine learning, BERT performs better at detecting hate speech because it can grasp word context in both directions, pre-train on large amounts of data to capture complex language relationships, and fine-tune on specific hate speech datasets to enable it to learn and adapt to the complexities of hate speech.

4.2 Case Study II (HASOC Dataset)

Based on Table. 8 results, it can be concluded that SVM is the best machine learning model for ACC (67%), REC (62%), and F1 (62%). The SVM's kernel technique, which converts the inputted data (tweets) into a higher-dimensional space and allows for finding a non-linear decision boundary, can be used to understand this performance. Once more, the Naive Bayes Classifier, which obtained (ACC (51%), PRES (57%), REC (56%), and F1 (50%), is the weakest machine learning classifier out of the four classifiers. Because NB is a simplistic model, it cannot capture intricate relationships between the input data, which accounts for the low outcomes in both scenarios. Once again, the BERT model is the best of all three (ML, DL, and Transformer). Its recall and F1-score have improved by 3% and 4%, respectively. BERT outperforms more conventional models like SVM, DT, KNN, RNNs, BiLSTMs, and NB in hate speech identification because of its transformer design, bidirectional contextual awareness, and significant pre-training on data. BERT is a better option for tasks requiring context-aware and complicated language understanding because of its capacity to capture nuanced language, handle terms that are not in the lexicon, and utilize transfer learning. Compared to more conventional and straightforward machine learning models, BERT's all-encompassing approach frequently yields state-of-the-art performance in hate speech detection. In contrast, model efficacy varies depending on particular use cases.

Table 7: The performance results of Machine Learning, Deep Learning, and Transformer methods for HSOL dataset.

Type	Classifiers	Extraction methods	Results			
			ACC	PREC	REC	F1
Machine Learning	DT	TF-IDF	0.85	0.91	0.70	0.74
	KNN	TF-IDF	0.88	0.91	0.77	0.81
	SVM	TF-IDF	**0.94**	**0.93**	**0.90**	**0.91**
	NB	TF-IDF	0.74	0.69	0.73	0.70
Deep Learning	RNN	Embedding layer	0.90	0.87	0.88	0.87
	BiLSTM	Embedding layer	0.91	0.89	0.88	0.88
Transformer Model	BERT	Embedding Layers	**0.95**	**0.94**	**0.93**	**0.93**

Table 8: The performance results of Machine Learning, Deep Learning, and Transformer methods for HASOC dataset.

Type	Classifiers	Extraction methods	Results			
			ACC	PREC	REC	F1
Machine Learning	DT	TF-IDF	0.64	**0.79**	0.54	0.46
	KNN	TF-IDF	0.65	0.68	0.56	0.52
	SVM	TF-IDF	**0.67**	0.65	**0.62**	**0.62**
	NB	TF-IDF	0.51	0.57	0.56	0.50
Deep Learning	RNN	Embedding layer	0.62	0.59	0.58	0.58
	BiLSTM	Embedding layer	0.65	0.62	0.61	0.61
Transformer Model	BERT	Embedding Layers	**0.67**	**0.66**	**0.65**	**0.66**

4.3 Graphical analysis

For the two Hate Speech Tweets datasets, HSOL and HASOC, Figure 3 summarized the best values of metrics in terms of ACC, PREC, REC, and F1, produced by the ML methods, DL models, and T approaches. BERT surpassed all ML and DL algorithms for the HSOL dataset regarding F1, accuracy, recall, and precision. BERT surpassed all competing DL and ML-based algorithms for the HASOC dataset regarding F1, accuracy, recall, and precision.

5 Conclusion and future work

Hate speech detection is essential in maintaining inclusivity in social networking sites. This paper used seven classification algorithms to specify whether input text is hate speech or not. We selected two datasets to test these classification algorithms on. For the four Machine Learning algorithms, we used the TF-IDF method to extract features from the provided text. We used the Keras word embedding layer for the remaining three Deep Learning and Transformer Algorithms to extract features.

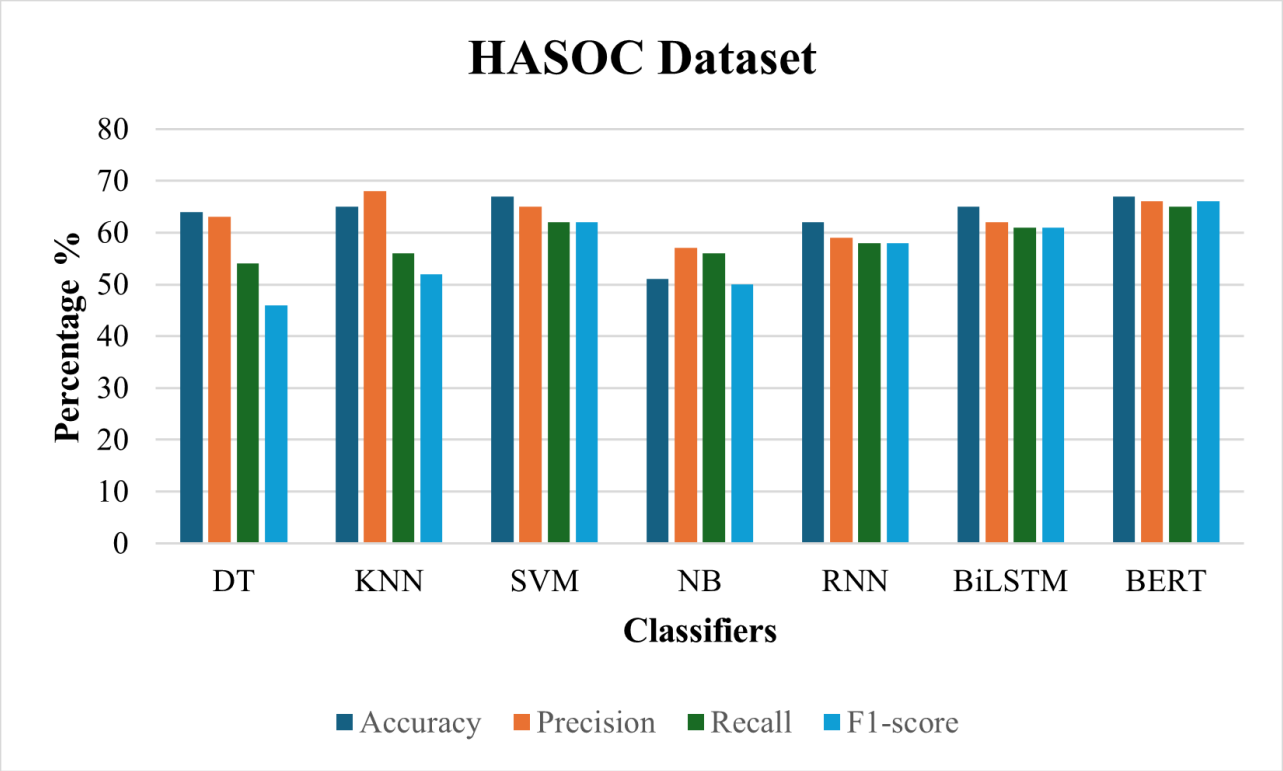
The best-performing algorithm for the HSOL Dataset was the BERT transformer model, which achieved accuracy of 95% . Of the Machine Learning Models used, SVM yielded the best accuracy (94%).

Regarding the HASOC Dataset, the BERT Transformer Model also yielded the best results, with an accuracy of 67% . In terms of the Machine Learning Models used, SVM yielded the best accuracy (67Also, We can see that the performance of the models in the second dataset is much less than that of the first dataset. This is because of the abundance of informal language in the second dataset.

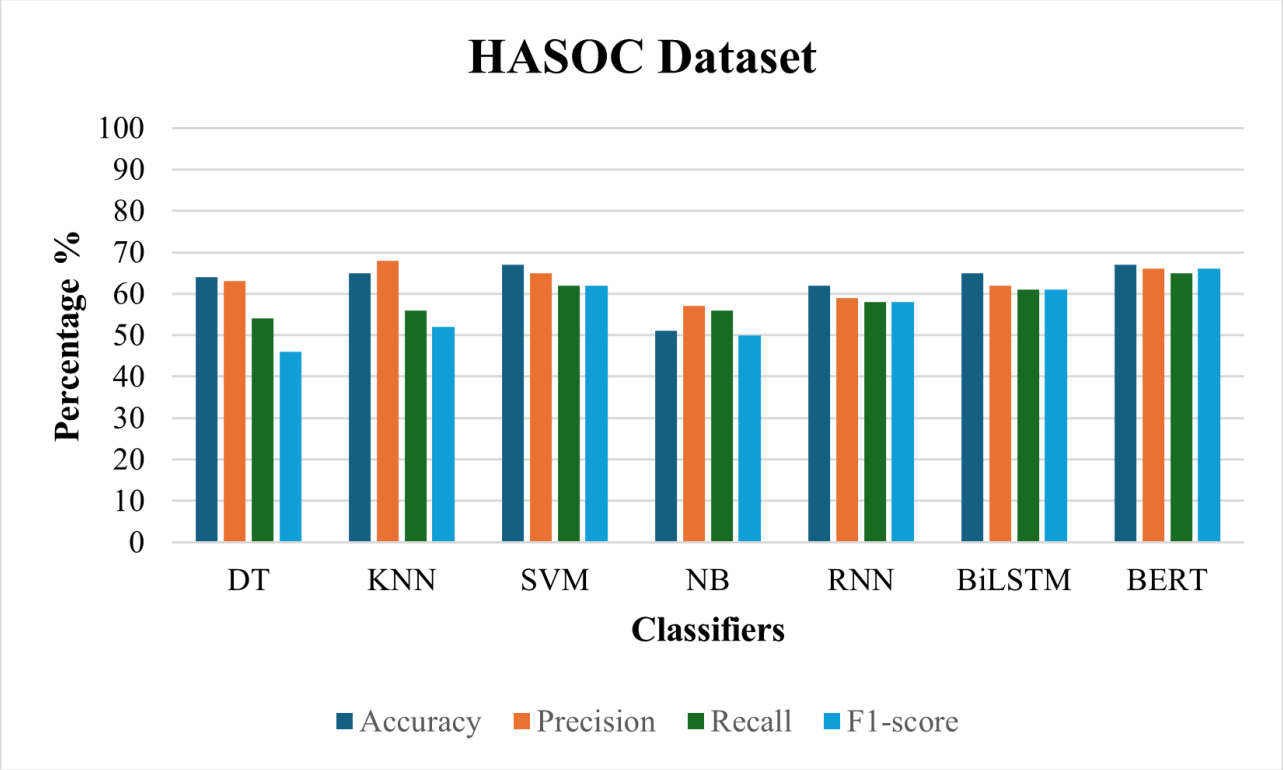
For both datasets, the worst-performing classifier is Naive Bayes. This is because Naive Bayes assumes that all features (words) are independent, but words often have contextual dependencies in language.

In the future, we would like to compare even more deep learning and transformer algorithms. In addition, use grid search in tuning hyperparameters.

The authors are grateful to the anonymous referee for a careful checking of the details and for helpful comments that improved this paper.



Performance results for the HSOL dataset



Performance results for the HASOC dataset

Fig. 3: The performance metrics for HSOL and HASOC Datasets using ML, DL and T methods

References

[1] J. B. Walther, "Social media and online hate," *Current Opinion in Psychology*, vol. 45, art. no. 101298, 2022, doi: https://doi.org/10.1016/j.copsyc.2021.12.010

[2] N. Djuric et al., "Hate speech detection with comment embeddings," in *Proceedings of the 24th International Conference on World Wide Web*, pp. 29-30, 2015.

[3]

[4] S. Dowlagar and R. Mamidi, "Using BERT and Multilingual BERT models for Hate Speech Detection," *CEUR Workshop Proceedings*, vol. 2826, pp. 180-187, CEUR-WS, 2021. https://arxiv.org/abs/2101.09007v1

[5] M. S. Jahan and M. Oussalah, "A systematic review of hate speech automatic detection using natural language processing," *Neurocomputing*, vol. 546, pp. 126232, Elsevier, August 2023. https://doi.org/10.1016/J.NEUCOM.2023.126232

[6] K. Kowsari et al., "Text Classification Algorithms: A Survey," *Information*, vol. 10, no. 4, art. no. 150, 2019. www.mdpi.com/journal/information

[7] N. S. Mullah et al., "Advances in Machine Learning Algorithms for Hate Speech Detection in Social Media: A Review," *IEEE Access*, vol. 9, pp. 88364-88376, Institute of Electrical and Electronics Engineers Inc., 2021. https://doi.org/10.1109/ACCESS.2021.3089515

[8] A. Omar, T. M. Mahmoud, and T. Abd-El-Hafeez, "Comparative Performance of Machine Learning and Deep Learning Algorithms for Arabic Hate Speech Detection in OSNs," *Advances in Intelligent Systems and Computing*, vol. 1153 AISC, pp. 247-257, Springer, 2020. https://doi.org/10.1007/978-3-030-44289-7_24

[9] T. T.A. Putri et al., "A comparison of classification algorithms for hate speech detection," *IOP Conference Series: Materials Science and Engineering*, vol. 830, no. 3, pp. 032006, IOP Publishing, May 2020. https://doi.org/10.1088/1757-899X/830/3/032006

[10] S. Abro et al., "Automatic Hate Speech Detection using Machine Learning: A Comparative Study," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 8, 2020. www.ijacsa.thesai.org

[11] F. Del Vigna et al., "Hate me, hate me not: Hate speech detection on Facebook," 2017. http://www.alexa.com/topsites

[12] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated Hate Speech Detection and the Problem of Offensive Language," 2017. www.facebook.com

[13] B. Gambäck and U. K. Sikdar, "Using Convolutional Neural Networks to Classify Hate-Speech," in *Proceedings of the Annual Meeting of the Association for Computational Linguistics*, pp. 85-90, ACL, 2017. https://aclanthology.org/W17-3013

[14] P. Badjatiya et al., "Deep Learning for Hate Speech Detection in Tweets," *26th International World Wide Web Conference 2017, WWW 2017 Companion*, pp. 759-760, International World Wide Web Conferences Steering Committee, 2017. http://arxiv.org/abs/1706.00188

[15] G. K. Pitsilis, H. Ramampiaro, and H. Langseth, "Effective hate-speech detection in Twitter data using recurrent neural networks," *Applied Intelligence*, vol. 48, no. 12, pp. 4730-4742, Springer New York LLC, 2018. https://link.springer.com/article/10.1007/s10489-018-1242-y

[16] R. Duwairi, A. Hayajneh, and M. Quwaider, "A Deep Learning Framework for Automatic Detection of Hate Speech Embedded in Arabic Tweets," *Arabian Journal for Science and Engineering*, vol. 46, no. 4, pp. 4001-4014, Springer Science and Business Media Deutschland GmbH, 2021. https://link.springer.com/article/10.1007/s13369-021-05383-3

[17] T. V. Huynh et al., "Hate Speech Detection on Vietnamese Social Media Text using the Bi-GRU-LSTM-CNN Model," 2020. http://arxiv.org/abs/1803.03662

[18] P. M. Sosa, "Twitter Sentiment Analysis using combined LSTM-CNN Models," 2017.

[19] R. Ong, "Offensive Language Analysis using Deep Learning Architecture," 2019. https://arxiv.org/abs/1903.05280v3

[20] R. T. Mutanga et al., "Hate Speech Detection in Twitter using Transformer Methods," *International Journal of Advanced Computer Science and Applications (IJACSA)*, vol. 11, no. 9, 2020. www.ijacsa.thesai.org

[21] T. Davidson et al., "Automated hate speech detection and the problem of offensive language," in *Proceedings of the International AAAI Conference on Web and Social Media*, vol. 11, no. 1, pp. 512-515, 2017.

[22] "Hate Speech and Offensive Content Dataset (HASOC)," 2019. https://hasocfire.github.io/hasoc/2019/dataset.html

[23] "Scikit Learn Randomized Search," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.RandomizedSearchCV.html

[24] "Scikit Learn Grid Search," https://scikit-learn.org/stable/modules/generated/sklearn.model_selection.GridSearchCV.html

[25] S. Ray, "Naive Bayes Classifier Explained: Applications and Practice Problems of Naive Bayes Classifier," 2023. https://www.analyticsvidhya.com/blog/2017/09/naive-bayes-explained/

[26] S. Uddin et al., "Comparative performance analysis of K-nearest neighbour (KNN) algorithm and its different variants for disease prediction," *Scientific Reports*, vol. 12, no. 1, art. no. 6256, Nature Publishing Group UK, London, 2022.

[27] A. Saini, "Guide on Support Vector Machine (SVM) Algorithm," 2024. https://www.analyticsvidhya.com/blog/2021/10/support-vector-machinessvm-a-complete-guide-for-beginners/

[28] L. Rokach and O. Maimon, "Decision Trees," in *The Data Mining and Knowledge Discovery Handbook*, vol. 6, pp. 165-192, 2005

[29] "Keras Embedding Layer," https://keras.io/api/layers/core_layers/embedding/

Deema AlSekait is a talented assistant professor with a strong information technology (IT) background. She received her PH.D. in information technology from Towson University, United States. Dr. AlSekait's extensive research portfolio covers a wide range of topics, from machine learning to health informatics to artificial intelligence and cloud computing. Additionally, Deema advocates for improving access to Science,

Technology, Engineering, and Math careers. She is a shining example of an outstanding IT professional in an ever-changing field, as she perfectly combines her research expertise, pedagogical knowledge, and uncompromising commitment to social advancement. Deema's tireless efforts continue to inspire innovation, ensuring that the future of information technology remains bright and inclusive.

Ahmed Shdefat received his Ph.D. degree from the Software Engineering Department, School of Engineering, at Inje University, South Korea, in 2018. Previously, he served as a Research Assistant in the same department at Inje University. He is now an Assistant Professor in Computer Science at the College of Engineering and Technology, American University of the Middle East. His research interests encompass IoT infrastructure and architecture, ECG signal processing, bioinformatics, real-time processing, recognition of human activities, and e-health systems. He is also a reviewer for the IEEE Transactions on AgriFood Electronics and pervasive and mobile computing journals.

Zakwan AlArnaout received a PhD degree from the School of Engineering and Computer Science at Victoria University of Wellington, Wellington, New Zealand. He is currently working as an assistant professor and department chair of the IST/TNT Department of the American University of the Middle East, Kuwait. His research interests include content caching and replication, multi-hop wireless networks, detection and prevention of distributed denial of service attacks, and key management schemes in WSNs. He is a member of the IEEE and the IEEE Computer Society.

Nermin Rafiq Mohamed is a distinguished Professor of Sports Psychology and the Head of the Department of Educational Fundamentals at the Faculty of Physical Education, Sadat City University. She embarked on her academic journey with a graduation in 1992, followed by a Master's degree in 2002, and earned her Doctorate of Philosophy in 2005. Prof. Mohamed progressed to Assistant Professor by 2011 and achieved full Professor status in 2019. She is also a Fellow of the National Defense College since 2018. Her research interests span psychological sciences, including psychological processes and the applications of psychological sciences across various fields. She is particularly focused on integrating psychological sciences with other disciplines and artificial intelligence.

Hana Fathi received her Ph.D. in computer science from the Faculty of Science, Menufia University, Egypt in 2022. She is an Assistant Professor at the Faculty of Information Technology, Applied Science private University. He has worked on several research topics. Hanaa has contributed more than 10+ technical papers in Feature Selection, classification, optimization, Machine learning, and web service in international journals. She attended the 2019 International Conference on Intelligent Systems and Advanced Computing Sciences (ISACS), in December 2019 in Taza, Morocco, and the fourth edition of the International Conference on Intelligent Systems and Computer Vision (ISCV 2020). She majors in machine learning, optimization, and medical application.

Diaa Salama is currently the Head of the Data Science Department at Misr International University, Egypt, on leave from Benha University since February 2020. He brings extensive experience in academia, having worked across several international and multicultural institutions. Dr. Salama has supervised numerous graduate projects, currently overseeing 13 master's and 3 Ph.D. students. His involvement in six research projects funded by King Faisal University, Saudi Arabia, in 2022, each supported by a grant of 40,000 Saudi Riyal, underscores his active research engagement. Dr. Salama has been recognized for his contributions to science and technology with several awards, including nominations for the Egyptian Encouragement Award for Advanced Science and Technology in 2021 and the Benha University Encouragement Award in 2019. He was also ranked among the top 2% of scientists worldwide by the Stanford University ranking in 2023. Additionally, he has consistently been ranked first at Benha University and highly among Egyptian and African universities by the AD Scientific Index from 2021 to 2024.His scholarly output includes over 150 research papers primarily in Q1 and Q2 journals, and he has presented at more than 30 international conferences. His publications have garnered over 2,670 citations. Dr. Salama is also an active member

of several prestigious organizations, including the National Committee of Communications and Information Technology in Egypt and the International Federation for Information Processing.He holds a B.Sc. in Information Systems from Zagazig University (2004), and an M.Sc. (2009) and Ph.D. (2015) in Cloud Computing from Menoufia University. His research interests include optimization and artificial intelligence applications in the medical sector and renewable energy, focusing on feature selection, text mining, and object recognition.