IEEE Access
Multidisciplinary : Rapid Review : Open Access Journal

Received 6 April 2023, accepted 20 April 2023, date of publication 5 June 2023, date of current version 9 June 2023.

Digital Object Identifier 10.1109/ACCESS.2023.3282834

RESEARCH ARTICLE

A Multi-Stage Machine Learning and Fuzzy Approach to Cyber-Hate Detection

LIDA KETSBAIA[1], BIJU ISSAC[1], (Senior Member, IEEE), XIAOMIN CHEN[1], AND SEIBU MARY JACOB[2], (Member, IEEE)
[1]Department of Computer and Information Sciences, Northumbria University, NE1 8ST Newcastle upon Tyne, U.K.
[2]School of Computing, Engineering & Digital Technologies, Teesside University, TS1 3BX Middlesbrough, U.K.

Corresponding author: Biju Issac (bissac@ieee.org)

ABSTRACT Social media has revolutionized the way individuals connect and share information globally. However, the rise of these platforms has led to the proliferation of cyber-hate, which is a significant concern that has garnered attention from researchers. To combat this issue, various solutions have been proposed, utilizing Machine learning and Deep learning techniques such as Naive Bayes, Logistic Regression, Convolutional Neural Networks, and Recurrent Neural Networks. These methods rely on a mathematical approach to distinguish one class from another. However, when dealing with sentiment-oriented data, a more ''critical thinking'' perspective is needed for accurate classification, as it provides a more realistic representation of how people interpret online messages. Based on a literature review conducted to explore efficient classification techniques, this study applied two machine learning classifiers, Multinomial Naive Bayes and Logistic Regression, to four online hate datasets. The results of the classifiers were optimized using bio-inspired optimization techniques such as Particle Swarm Optimization and Genetic Algorithms, in conjunction with Fuzzy Logic, to gain a deeper understanding of the text in the datasets.

INDEX TERMS Cyberbullying, fuzzy logic, logistic regression, multinomial Naive Bayes, PSO, VADER.

I. INTRODUCTION

It was the advancement of technology and the impulse of human communication that led to the evolution of social media, which altered how individuals interact online. Prior to the introduction of Information Communication Technology (ICT), human interactions were largely confined to geographical locations; however, Online Social Networks (OSNs) have eliminated geographical barriers [1].

It has become increasingly apparent that cyber-hate is a widespread issue due to the pervasiveness of easy-to-use technologies. Social media platforms have emerged as a medium for the perpetration of aggressiveness and bullying, making it a dangerous and elusive phenomenon. The ease with which perpetrators can commit harmful acts through the utilization of a laptop or mobile device connected to the internet renders young individuals highly vulnerable to online harassment. A conventional approach to detecting cybercrime involves manual flagging of data [2]; however, this method has been demonstrated to be neither ''effective nor scalable'' [2]. This has prompted researchers to investigate the potential of utilizing Machine Learning and Deep Learning techniques to design automated systems capable of detecting and preventing cyber-hate.

Considering the vast amount of content that can be found on OSNs related to aggressive and anti-social behaviour, the paper proposes an Optimized Machine Learning-Based framework to help identify online hate using fuzzy logic techniques. Several different machine learning models have been implemented, such as, Multinomial Naive Bayes and Logistic Regression, in conjunction with the Bio-Inspired Optimization methods, Genetic Algorithm and Particle Swarm Optimization. The implementation of Particle swarm Optimization selects the best feature selection subset that better represents the feature selection space. The aim is to decrease the quantity of redundant and unimportant features, thereby enhancing the accuracy of the classification process within a data set. Additionally, PSO improves the comprehensibility

The associate editor coordinating the review of this manuscript and approving it for publication was Kumaradevan Punithakumar.

of the learned model. Furthermore, Genetic Algorithm (GA) was implemented to optimize the performance of classifiers. The random mutation aspect of the GA provides a degree of assurance that a wide range of solutions are evaluated. Furthermore, the application of fuzzy rules is able to incorporate the fuzziness of positive and negative scores. The Fuzzy Logic-based systems were applied to deal with vagueness and ambiguity. The advantages of using the fuzzy approach are summarized as i) It provides a desirable way to deal with linguistic problems and ii) Deals with reasoning and gives closer views to the exact sentiment values. The paper is structured as follows: Section II presents related works, Section III outlines the datasets used, Section IV provides details relating to the framework, Sections V-VII identify the three stages of the framework, Section VIII describes the technical setup and environment used for conducting experiments in the report Section, IX presents results and discussion, and lastly Section X concludes the paper.

II. RELATED WORKS

This section presents a comprehensive examination of studies on online hate in the field of Machine Learning and Deep Learning-driven text categorization, along with an overview of optimization techniques and fuzzy classification in real-world applications.

In response to the rampant surge of cyber hate, several nations have enacted laws targeting cyberbullying. For example, the United Kingdom has enforced legal provisions as per the Malicious Communications Act 1988 [3]. This statute, upon conviction, stipulates punitive measures, which include a prison term of up to six months and the imposition of a financial penalty on the offender. Furthermore, if the online activities of the offender cause fear or distress to the victim, they could be liable for criminal charges under the Harassment Act 1997. Similarly, the Canadian legal system employs a range of preventative measures to counteract cyberbullying, including incarceration, confiscation of electronic devices, and compensation for the aggrieved parties. The severity of the cyberbullying incident dictates the potential charges faced by the perpetrators, which may include criminal harassment, uttering threats, intimidation, public incitement of hatred, and offence against the person and reputation [3].

In the United States, a variety of states have implemented legal statutes that prescribe a spectrum of punitive measures, including financial sanctions and custodial sentences, to address incidents of cyberbullying. Conversely, some states have yet to articulate a definitive and comprehensive elucidation of the legal frameworks pertaining to incidents of cyberbullying.

In 2016, the European Commission agreed with 17 of the world's leading companies, such as Facebook, Microsoft, Twitter, and YouTube, to a "Code of Conduct" to counter illegal hate speech online. By 2021, LinkedIn, TikTok, jeuxvideo.com, Dailymotion, Instagram, and Snapchat also joined. The implementation of the Code of Conduct is assessed through regular monitoring exercises, which are set up in collaboration with a network of organizations located in different EU countries. Using an agreed-upon methodology, these organizations test how the IT companies implement the Code [4].

In light of the limited amount of legislation throughout the world in addressing the problem of online hate, researchers have been motivated to develop automated systems that would detect and manage the problem. Abundant information on individuals and their societies was previously impossible to acquire and analyze; however, it can now be obtained due to the big data era we are currently living in. OSNs, such as Facebook, Twitter and Instagram, are able to generate information that can be used for analysis, such as, link prediction, community, content, and social influences.

Usually, the input information to natural language processing tasks undergoes several pre-processing phases before being processed. Pre-processing is commonly performed in order to reduce noise, thus improving the accuracy of the data. However, it can also result in the loss of useful context during the process. For example, in textual communication, capitalization is often used as a means of indicating shouting. Therefore, unintentionally converting uppercase words to lowercase may result in a loss of context. There is a wide range of pre-processing methods used in the literature reviewed. However, tokenization and stemming appear to be most prevalent, with n-grams and BoW (Bag-of-words) commonly used as features during stemming. This is a logical step, as by reducing words to their stems, their frequencies are consolidated into a single value, thereby emphasizing their importance in the dataset. Tokenization is often employed to break sentences and phrases into a sequence of characters and performed to enable a document to be represented as a function of its words.

Several studies reported that by removing repeated characters and correcting grammar and spelling, caused the quality of the results to improve [5]. However, [6] indicated that the use of stemming words might cause other legitimate words to be created, resulting in the meaning of the sentence being altered. A further argument was made that excessive repetition of characters in words is often intended to emphasize a point instead of being portrayed as a misspelling. The authors suggested that interpreting such repetition as additional emphasis is a more effective method than autocorrection.

According to the literature review conducted, supervised learning techniques were used in the majority of the papers examined to detect cyberbullying, with Yin et al. [7] being on of the earliest researcher found who employed this technique.

The authors Yin et al. [7] examined posts and comments from three social media websites. The researchers discovered that harassment posts have a relatively small percentage in a corpus, therefore the researchers hypothesized that harassment posts would differ significantly from their neighbouring posts in terms of appearance. This assumption led to k-fold validation being implemented with SVM (Support Vector

Machine), which identified an improvement in the classification performance when compared with experiments without k-fold.

An SVM classifier was trained on a gender-specific corpus of MySpace posts by using TFIDF on profane words and pronouns. According to their findings, gender-specific features significantly improved cyberbullying detection results in comparison to results obtained from a non-segregated dataset using the same classifier. The study's results encourage the incorporation of gender characteristics in the detection of online bullying. However, it is essential to note that gender information (as well as any other information provided by users) on social media can easily be manipulated. A reliable method of validating user-supplied information is therefore crucial to any method that uses such means. [8]

In the light of this work, [9] advocated an approach that incorporates the impact felt by the receiver in order to accurately determine the severity of bullying episodes by combining content-based approaches with an approach that incorporates the impact felt by the receiver. This can be accomplished by examining the replies or follow-up actions of a recipient within a particular or similar environment.

Moreover, Reynolds et al. [10] used labelled data in conjunction with machine learning to recognize bullying content. Both a C4.5 Decision Tree learner and an instance-based learner were able to identify the true positives with 78.5% accuracy.

Rafiq et al. [11] added four types of features to the classifiers, namely media session features, profile owner features, comment-based features, and N-grams. After considering the features, they employed four classifiers - Naive Bayes, AdaBoost, Decision Tree, and Random Forest - with 10-fold cross-validation. Their results indicated that AdaBoost produced the highest accuracy.

Current approaches to abusive content detection aim at binary text classification based on supervised learning, with a focus on finding the right set of features to perform the classification. Various features were investigated by supervised machine learning methods based on the content of comments, user profiles, user activities, and social graph structure of comments. Experts or crowdsourcing platforms were used to label the training data. However, these methods have the disadvantage that their performance is highly dependent on the quality of the training data, so they require a large amount of labeled data for training.

Furthermore, datasets concerning online hate exhibit imbalanced class distributions, where one class possesses a greater number of data instances than the other (i.e., the majority class). This is attributed to the significant class imbalance frequently observed in real-world domains, where the decision system is engineered to detect rare yet consequential events. Consequently, Chatzakou et al. [12] posited that Random Forest (RF) is not well-equipped to handle imbalanced training datasets, leading to a greater bias towards the majority classes. Nevertheless, Al-Garadi [13] proposed a solution to this problem by conducting comprehensive experiments to evaluate the performance of four selected classifiers and three feature selection algorithms, namely c2test, information gain, and Pearson Correlation, to identify the most significant feature. To address the imbalance inherent in the dataset, Al-Garadi employed SMOTE to augment the samples of the minority class. The classifiers' performance was measured using AUC, with Random Forest with SMOTE exhibiting the highest AUC (0.943) and f-measure (0.936). In references [15], [16], and [17], four effective approaches to mitigating issues associated with imbalanced datasets are discussed, namely sampling-based approaches, cost-based approaches, kernel-based approaches, and active learning approaches. In certain instances, these approaches have been effectively employed to balance online hate datasets.

One additional approach that may be employed for detecting abusive content is the lexicon-based filtering method. This technique involves the identification of abusive comments based on the presence of certain words or phrases within the text. According to Reynolds et al. [10], "bad" words are clear indicators of cyberbullying. A profane word list containing 296 terms was compiled by the authors, and they assigned a severity weight to each term. This approach was adopted to facilitate the authors' interest in utilizing both the number of "bad" words (NUM) and the frequency of "bad" words (NORM) as features for input into their learning tool. As part of their experiment, Sood et al. [18] tested the efficacy of the lexicon-based approach by using a profanity list from phorum.org. In order to determine whether a comment contains any of the words listed on phorum.org, they developed a system which flags comments as offensive. In addition to misspellings (which may be intentional), the inability to adapt to evolving offensive language and the context-specific nature of profanity are the primary factors responsible for the poor performance of this technique.

Furthermore, as a method of dynamically extending the vocabulary of insulting terms, Zhao et al. [19] assigned different weights to each term in order to obtain bullying characteristics using Word2Vec.

Nahar [20] utilized sentiment features, which were produced by employing Probabilistic Latent Semantic Analysis (PLSA), in combination with bag-of-words (BoW) features to train a Linear SVM classifier. Their findings indicated that the detection of cyberbullying was enhanced with the incorporation of sentiment features, in contrast to utilizing only BoW features. Despite an increase in bullying attributes, the authors did not consider word semantics, and the scalability performance was entirely arbitrary. In a subsequent study conducted by Nahar et al. [21], they achieved even more favorable outcomes by substituting the bag-of-words (BoW) feature with a weighted TFIDF scheme and employing Latent Dirichlet Allocation (LDA).

It has been suggested by some scholars that the sole reliance on lexical features may not effectively capture the intricacies of cyberbullying. As per Hosseinmardi et al. [22], among Instagram media sessions containing explicit or offensive material, only 30% were identified as acts of cyberbullying. Moreover, the authors reported that despite cyberbullying posts featuring a moderate proportion of negative terms, the most negative posts were not classified as cases of cyberbullying by the evaluators. Instead, these negative posts revolved around topics such as politics and sports. Thus, it is imperative to recognize these findings when devising approaches to identify and prevent cyberbullying.

Overall, conventional machine learning models are the most widely used in the classification of online hate. Most of the ML research applies supervised learning models. Support Vector Machines (SVM) and Naive Bayes (NB) have been found to work best within text classification, while other well-known models, such as Logistic Regression (LR), Decision Trees (DT), k-Nearest Neighbours (kNN), and Random Forests (RF), have also been applied. However, hybrid semi-supervised and unsupervised models have also been researched over the years. Researchers are turning to deep learning(DL) algorithms to address the problem of identifying abusive content and hate speech. Numerous types of DL neural networks, including Conventional Neural Networks (CNNs), Recurrent Neural Networks (RNNs), Gated Recurrent Units (GRUs), and Bidirectional Long Short-Term Memory (Bi-LSTM), have been developed. Recently, DL methods have gained significant popularity in the research community, and they have been found to perform better than other methods ([23], [24], [25], [26], [27]). It has been demonstrated in several studies that deep learning models, such as CNNs employing Word2Vec, GloVe, FastText, and other embeddings, outperform standard machine learning models such as SVMs, LRs, NBs, and RFs ([27], [28]).

In order to increase the performance of machine learning classifiers, optimization techniques have been applied to traditional ML methods. In their paper [29], the researchers combined a PSO-based feature selection with C4.5 for sentiment analysis on a dataset regarding online transportation in Indonesia. Various experiments were performed by considering different parameter settings of PSO. Their results indicated that in terms of accuracy metric, their proposed method significantly outperformed the performance of a simple C4.5 algorithm. The authors aimed to address the limitations of the C4.5 algorithm, such as its sensitivity to irrelevant features and its inability to handle large datasets. By using PSO as a feature selection method, the authors aimed to improve the accuracy of the C4.5 algorithm. However, it is important to note that the limitations of the C4.5 algorithm may still exist and that the use of PSO as a feature selection method may not completely resolve these limitations.

Moreover, PSO and Genetic Algorithm, in combination with DT classifiers, were utilized for feature selection in a dataset that consisted of 18 attributes collected from 1275 patients [30]. According to the study, PSO combined with DT generated a better classification result than GA combined with the DT classifier [30]. Another study comparing PSO and GA was conducted by Al-ab and Al-taani [31]. Their study evaluated the use of a meta-heuristic algorithm (PSO) which merged informative scoring with semantic scoring to create a shorter version of an original text in an Arabic document with a Genetic Algorithm (GA) and Harmony Search (HS). The results confirmed that the PSO algorithm achieved a higher precision and F-score measure than the GA and HS approach [31].

In their work ''Metaheuristic Ant Lion and Moth Flame Optimization-Based Novel Approach for Automatic Detection of Hate Speech in Online Social Networks'' [32], the researchers presented an optimized methodology using the Ant Lion Optimization (ALO) algorithm and the Moth Flame Optimization (MFO) algorithm to tackle online hate speech. The research represented a first-of-its-kind attempt to use optimization algorithms as solution-finding strategies for hate speech detection. Their initial step involved basic natural language processing procedures such as feature extraction using the Bag of Words (BoW), Term Frequency (TF), and Document Vector (Word2Vec). Subsequently, the performances of the newly proposed approaches were thoroughly evaluated on three data sets. The results were compared against eight well-established supervised machine learning algorithms; however, ALO and MFO algorithms were found to be superior to those of the machine learning methods.

Unconventional methods have also been applied to identify online hate. For example, Fuzzy Fingerprints were used by [33] to identify the unique fingerprint of positive cyberbullying examples in the training dataset. The F1 score achieved by the model was then compared to the baselines for unbalanced datasets.

Furthermore, [34] proposed a modified fuzzy approach with two-stage training for the classification of four types of hate speech, specifically those based on religion, race, disability, and sexual orientation. The proposed approach is designed to address the challenge of text ambiguity. The features are extracted using a combination of bag-of-words and word embedding methods, and the correlation-based feature subset selection method is used to select the relevant features. The performance of the proposed fuzzy approach is compared to popular methods and existing fuzzy approaches. The results of the experiments indicate that the proposed fuzzy approach outperforms the other methods (DT, NB, SVM, GBT, and DNN) in most cases.

The research conducted by Vashishtha and Susan [35] aims to evaluate the sentiment of social media posts through the utilization of a novel set of fuzzy rules that encompass multiple lexicons and datasets. The proposed fuzzy system integrates Natural Language Processing techniques and Word Sense Disambiguation using a unique unsupervised nine fuzzy rule-based system that enables the classification of posts into positive, negative, or neutral sentiment classes. To assess the effectiveness of this approach, a comparative analysis was conducted on nine publicly available Twitter datasets, three sentiment lexicons, four state-of-the-art unsupervised Sentiment Analysis approaches, and one state-of-the-art method for supervised machine learning. Although it is customary to use a single lexicon for Sentiment Analysis of Twitter data, the integration of fuzzy logic with lexicons for sentiment classification provides a new paradigm in Sentiment Analysis. The study's findings provide valuable insights to researchers concerning which lexicon may be most suitable for social media. The specific study of incorporating Fuzzy Logic and Sentiment Lexicon is used as the basis of our research.

Finally, [36] present MultiLexANFIS, an adaptive neuro-fuzzy inference system that integrates inputs from multiple lexicons to perform sentiment analysis on social media posts. The system classifies tweets into two classes: neutral and non-neutral, where the latter class encompasses both positive and negative polarities. This classification is particularly relevant for applications aimed at evaluating the neutrality of content posted on social media platforms. The proposed model leverages the integration of natural language processing with fuzzy logic to effectively handle the inherent fuzziness of natural language. The authors proposed a novel set of 64 rules for the neuro-fuzzy network that utilizes fuzzy features obtained from the VADER, AFINN, and SentiWordNet lexicons to accurately classify tweets. The proposed rules are domain independent and can be extended to other textual data that employs lexicons.

A table summarizing the literature can be viewed in Table 1.

TABLE 1. Review of literature.

Research	Classifier	Feature
[5]	NB , J48, SVM, JRip	keywords, TFIDF, Profanity, N-gram
[6]	EDLSI	Profanity
[7]	SVM	TFIDF, Pronouns, N-gram
[8]	SVM	TFIDF
[9]	SVM	TFIDF
[10]	J48	Lexical
[11]	NB, AdaBoost, DT, RF	
[12]	J48, LADTree, LMT, NBTree, RF, and Functional Tree	
[13]	KNN, SVM, RF KNN, SMOTE	Lexical + Semantic + User-Based + Twitter-Based
[14]	LR, RF, SVM, GBD, CNNs, LSTM,CNN + GBDT, LSTM + GBDT, FT + GBDT	N-gram, word embeddings.
[15]	NB, LDANB, CosSim, AB	BoW, Lexical, Contextual BoW, BoCH, GloVe
[16]	NB, KNN, BoostedLR, Stochastic GBDT, SVM	
[17]	LR, SVM, RF, RP, Xg-boost, MLP	characteristic n-grams
[18]	SVM	TFIDF, N-gram, Profanity, Levenshtein Distance
[19]	SVM	Continuous Bag of Words, Semantic-enhanced BoW Model, Embeddings-enhanced Bag-of-Words (EBoW), Latent Dirchilet Allocation (LDA),
[20]	SVM	Ngram
[21]	SVM, LDA, HITS	TFIDF, Pronouns, Profanity
[23]	CNN, CNN-LSTM, and BiLSTM-CNN	SG, CNN, CBOW
[24]	LSTM,GURU, CNN+GRU, CNN+LSTM	Word Embedding
[25]	CNN, LSTM, GRU	FastText, Word2Vec, GloVe
[26]	CNN and BiLSTM-CNN	
[27]	LR, SVM CNN, LSTM, GBDT	FastText, GloVe Random Embedding, Tf-IDF, BOW
[28]	BERT, Multilingual-BERT	ELMO
[29]	DT, Bagging, PSO	Ngram
[30]	C4.5, K-NN, SVM	PSO
[31]	PSO, Harmony Search	TFIDF

III. THE DATASETS USED

This study analyzed four publicly available datasets. All datasets were in CSV format with a text column and a corresponding label column that indicated whether the text was hateful or non-hateful.

Formspring was the first dataset applied, which was created by Kelly Reynolds and is currently available on Kaggle [10]. The text in the dataset was labeled as "cyberbullying" based on whether two annotators identified the text as "cyberbullying". The resultant dataset contains 12,772 samples, with 86 labeled as "non-cyberbullying". An example of this dataset is shown in Table 2.

TABLE 2. Formspring dataset.

Text	Label
your fucking ugly like your mom	1
your mad ugly true talk like your face	1
you should come next week ! k	0
you should die kay	1

The second dataset utilized was created by the University of Maryland for their paper "A Large Human-Labeled Corpus for Online Harassment Research" [37]. The creation of the dataset was motivated by three main objectives: first to train machine learning models, second to recognize linguistic features, and third to study the nature and culture of abusive online comments.

The dataset handles not only violent harassing comments but also sexually violent phrases, threats, racist, and derogatory comments. The reason why this was done is to provide a realistic view of how hate is portrayed online. In order for the data to be obtained, a list of search terms was used to download the required data using Twitter API. The terms, phrases and hashtags that were collected consisted of:

- Jews
- #fuckniggers,
- raghead,
- #WhiteLivesMatter,
- Feminist

TABLE 3. Maryland dataset.

Text	Label
A ghetto is where the Jews live.	1
the Jews killed Jesus, do u think their your friend?	1
I refuse to sit on the back of the bus, I as a white male have earned the right to sit at the front #FuckYouMatteo #WhitePower	1
Comedy. I had no idea Priam and Thor were actually black. The things you learn on twitter.	0

TABLE 4. Davidson dataset.

Text	Label
#Fact females that tweet ALL the dayum day r hoes that need attention #ijs #wontgetme	1
#Iowa is full of white trash	1
'single af" we don't care you're ugly anyways ! k	1
At some point, we need to discuss how delicious the vegan brownie from on u St. tastes!	0

- "fucking BLANK" (the "blank" is filled in with a religion or other derogatory term)
- religion of hate

As some tweets served as responses to other tweets, the authors included the original tweet as part of their analysis. The replies included a number of harassing comments directed at the original poster, as well as others agreeing with the harassing posts. Furthermore, the authors of the Davidson dataset ensured that the tweets collected were among the worst possible data, as they primarily contained racist, misogynistic, violent, and homophobic comments. The purpose of the dataset was to facilitate the understanding and evaluate the true extent of online hatred by users. The final labelled corpus took approximately three months to label manually, creating a final dataset of 35,000 tweets. Table 3 illustrates the difference between "Hate" Tweets and "Non-Hate" Tweets in the dataset.

The dataset referred to as Davidson throughout the paper was used in the article "Automated Hate Speech Detection and the Problem of Offensive Language" [38]. Using Twitter's API, the authors searched for words and phrases in a hate speech lexicon compiled by Hatebase.org that were identified as online hate by internet users. In their study, 25,000 tweets were randomly selected and categorized into three categories: offensive, hate speech, and neither. As a result, 24,802 tweets were labelled, of which approximately 5% were identified as hate speech, 76% as offensive language, and the remainder as non-offensive. Since the research focused on examining online hate and hate speech, the dataset was altered by combining the hate-speech and offensive language data into one category. Table 4 provides an example of the tweets within the dataset.

The final dataset used in this study is OLID [39]. It consists of a collection of English tweets that have been annotated in accordance with a hierarchical three-layer annotation model. The dataset contains 14,100 tweets. The tweets were labeled using the crowdsourcing platform Figure Eight Inc.

TABLE 5. OLID dataset.

Text	Label
@USER Certainly a weird view toward the Margaret Sanger geonicidal philosophy of aborting minorities. I'd like to know if the black population that suffers the most shares her opinion	0
@USER Fuck off	1
@USER You are lying. Stick with the facts. We pay more than that now.	0
@USER @USER @USER He is dumb and dumber all in one president. It's a two for one.	1

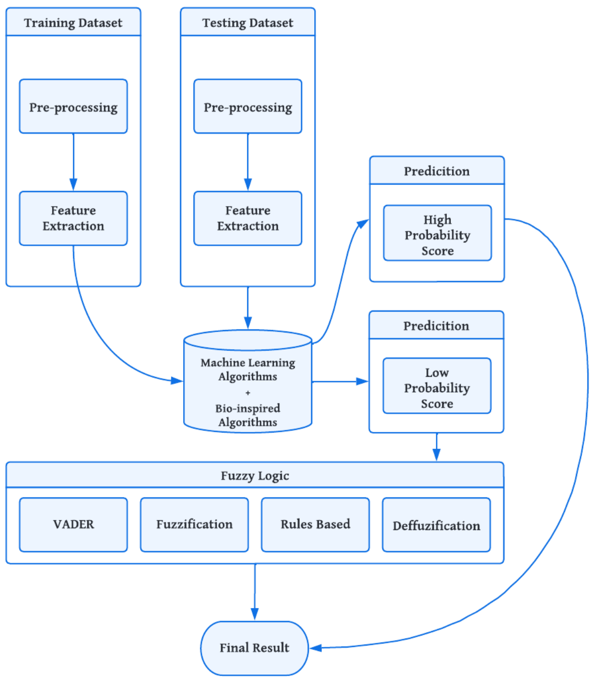

FIGURE 1. Multi-stage hate speech detection framework.

The authors of the OLID dataset ensured the quality of the annotation by hiring experienced annotators on the platform and using test questions to discard annotators who did not achieve a certain threshold. All tweets were annotated by two people. In case of disagreement, a third annotation was required, and ultimately, the majority vote was applied.

IV. MULTI-STAGE HATE SPEECH DETECTION FRAMEWORK

The proposed framework is characterized by three distinct stages. The first stage is the preprocessing, which removes noise in the datasets. The second stage implemented Machine learning classifiers using Bio-inspired optimization techniques such as Particle Swarm Optimization and Genetic Algorithms. The final stage applied Fuzzy Logic based on the Machine learning confidence scores that were derived in the second stage. The framework can be seen in Fig.1.

V. STAGE ONE OF FRAMEWORK

Stage 1 facilitates the pre-processing of all datasets, which involves cleaning the data, and their subsequent loading into

Negative Reviews Wordcloud

FIGURE 2. Negative word cloud - Formspring dataset.

Postive Reviews Wordcloud

FIGURE 3. Positive word cloud - Formspring dataset.

the program. The datasets are then divided and distributed into training and testing data.

A. PRE-PROCESSING

The framework starts of by processing multiple CSV files containing a text column and a label column. The text column will provide the online communication of individuals whereas the label column will identify whether the data is hate or non-hate using "1" for hate and "0" for non-hate related data. Since the research is primarily concerned with textual online messages, pre-processing is increasingly necessary because online communication contains a significant amount of unstructured information. With the help of pre-processing, a cleaner dataset can be generated, resulting in substantial improvements in the performance of the classifiers. The pre-processing techniques used include Lowercasing, Stemming, Normalization, Contractions, Removal of Special Characters, Tokenization, etc. Upon completion of the preprocessing process, a word cloud is generated in order to provide a visual representation of the text. This method provides insight into the most prominent words in the text, by visualizing the word frequency as a weighted list. Fig.2 and Fig.3 show an example of the negative and positive word clouds produced for Formspring dataset.

B. FEATURE EXTRACTION

The module 'Count Vectorizer' from Scikit-learn was applied as it assigns numbers to each word and provides its occurrence within a text. CountVectorizer uses a bag-of-words approach that ignores the text structure and only extracts information from the word counts. It will transform each document into a vector. A vector is constructed by taking into account the number of occurrences of each unique word in a document. Assuming there are m documents in the corpus, and n unique words are found, the CountVectorizer will transform the text data into a $m \cdot n$ sparse matrix [40].

This vectorizer is used to exclude English stopwords, namely, words such as: the, is, at, etc. It is imperative to recognize that stop words are words that are found within a text but have no impact on the overall meaning of the sentence. The reason for their removal is to reduce noise as well as the dimension of the feature set [42].

Additionally, the 'TfidfTransformer' module was included to compute the Inverse Document Frequency (IDF) to enable algorithms to read the data. TF-IDF(Term Frequency-Inverse Document Frequency) is a combination of two individual metrics, TF and IDF. Equation 1 identifies how the TF-IDF score for term t in document d is calculated [42],

$$\mathrm{TF}-\mathrm{IDF}(t,d) = \mathrm{TF}(t,d)\cdot \mathrm{IDF}(t) \tag{1}$$

(t, d) is the term frequency (t) within a document (d). TF is calculated using Equation 2,

$$\mathrm{TF}(t,d) = \frac{\text{term } t \text{ frequency in document } d}{\text{total words in document } d} \tag{2}$$

IDF however is calculated by Equation 3 [42],

$$\mathrm{IDF}(t) = \log_2 \frac{\text{total documents}}{\text{documents with term } t} \tag{3}$$

VI. STAGE TWO OF FRAMEWORK

This stage looks at the implementation of bio-inspired algorithms with machine learning classifiers.

A. MACHINE LEARNING CLASSIFIERS

1) NAIVE BAYES

The Naive Bayes model is a supervised learning technique and uses probability to perform classification tasks. The Naive Bayes model is denoted by Equation 4 [43]

$$\text{posterior} = \frac{(\text{prior probability})\cdot(\text{likelihood})}{\text{evidence}} \tag{4}$$

Consider a word vector W, whose class is given by C. In the case of current datasets, there are two classes C = N (Non-hate) and C = H (Hate). W is classified as the class which has the highest posterior probability $(P(C_k|W))$, which can be re-expressed as [44],

$$P(C_k|W) = \frac{p(C_k)\cdot p(W|C_k)}{P(W)} \tag{5}$$

If the 'Class' = Hate, the equation could be rewritten to find the hate text from the given words [41],

$$P(\text{Hate}|W) = \frac{P(W_i|\text{Hate})\cdot P(\text{Hate})}{P(W_1, W_2, \ldots W_n)} \tag{6}$$

NB has extensively been implemented to construct cyberbullying predictions and can be observed in models produced by several researchers. Naive Bayes algorithms have the assumption that all feature variables are independent. There are different kinds of Naive Bayes, such as Gaussian

Naive Bayes, Multinomial Naive Bayes, and Bernoulli Naive Bayes [45].

Multinomial Naive Bayes (MNB) would be one of the best-known Naive Bayes classification approaches using the term frequency to represent the document. MNB can be denoted using Equation 7 based on the probability of document d being in class c [46],

$$P(c|d) \propto P(c) \prod_{1<k<n_d} P(t_k|c) \tag{7}$$

$P(t_k|c)$ identifies the conditional probability that the term t_k occurs within a document of class c. $P(t_k|c)$ is interpreted as a measure of how much evidence is contributed by t_k that c is the correct class. $P(c)$ is the prior probability. In situations where a document's terms do not provide clear evidence for one class versus another, the one with the higher prior probability is chosen. $[t_1, t_2, \ldots t_n]$ are tokens within d that are a part of the vocabulary that was used for classification, whereas n_d is the number of such tokens in d.

2) LOGISTIC REGRESSION

Logistic Regression is a statistical method used for analyzing datasets with one or more independent variables that determine an outcome. This classifier belongs to the family of exponential or log-linear classifiers. The log-linear classifier works by extracting a set of weighted features from the input and combining them linearly. Logistic regression classifies an observation into one of two classes, where the output is a binary representation, i.e., 1 (Hate, Bullying, True, etc.) or 0 (Non-Hate, False, etc.). The objective is to find the best-fitting model to distinguish the relationship between the dependent variables and a set of independent variables. Logistic Regression is given in Equation 8, [47]

$$P(y = 1|x) = \frac{1}{1 + e^{\beta_0+\beta_1 x_1+\beta_2 x_2+\ldots\beta_n x_n}} \tag{8}$$

where $P(y = 1|x)$ is the probability of the outcome y being 1 when given a set of predictor variables $x_1, x_2, x_3 \ldots x_n$. Furthermore, $\beta_0, \beta_1, \beta_2, \ldots, \beta_n$ are the corresponding coefficients estimated by the LR model.

B. PARTICLE SWARM OPTIMIZATION

As a meta-heuristic optimization method, Particle Swarm Optimization (PSO) simulates the behaviour of flocks of birds. The PSO approach has been described as a simple optimization method; however, despite its simplicity, it has been found to converge to the optimum on a wide range of optimization problems. It is based on the assumption that information is optimized not only by individual experience but also by social interaction within the population [48]. Thus as individual solutions move in the search space, their velocity is dynamically adjusted by their own experiences and those of other particles [48].

In the original design, PSO was developed to solve problems of continuous-number search. However, feature selection, as with many other optimization problems, occurs in discrete search spaces. A binary version of the PSO was developed by Kennedy and Eberhart [49] to address the optimization problems associated with discrete domains. This binary version is known as BPSO. A particle's position can be expressed in two terms using BPSO: either "1" or "0". "If there is a particle x on d-dimensions, then its position can be defined as [49],

$$x = [x^1, x^2, x^3, \ldots, x^d], \text{ where } x^i \in 0, 1 \tag{9}$$

supposing that we are given a data set with d features. What we will do is that we're going to assign each feature as a dimension of a particle. Hence, once we have implemented Binary PSO and obtained the best position, we can then interpret the binary array simply as turning a feature on and off" [49].

The objective function is defined as,

$$f(X) = \alpha(1 - P) + (1 - \alpha)(1 - \frac{N^f}{N^t}) \tag{10}$$

It is the hyperparameter that decides the trade-off between the classifier performance P, and the size of the feature subset N^f with respect to the total number of features N^t. The classifier performance can be identified as the accuracy, F-score, precision etc.

By using a position-velocity update method, Binary Particle Swarm Optimization (binary PSO) finds the best solution from a set of candidate solutions.

The position update is defined using Equation 11 [50],

$$x_i(t + 1) = x_i(t) + v_i(t + 1) \tag{11}$$

Let $x_i(t)$ denote the position of particle i in the search space at time step t; unless otherwise stated, t denotes discrete time steps. The position of the particle is changed by adding a velocity, $v_i(t)$, to the current position. Equation 12 [50] illustrates the update rule for the velocity,

$$\begin{aligned} v_{ij}(t + 1) = w \cdot v_{ij}(t) + c_1 r_{1j}(t)[y_{ij}(t) - x_{ij}(t)] \\ + c_2 r_{2j}(t)[y_j(t) - x_{ij}(t)] \end{aligned} \tag{12}$$

i represents an individual variable within the swarm ranging from $1, \ldots, n$ (n represents the number of particles in the swarm). Similarly, j represents a specific position within a particle, ranging from $1, \ldots, m$ (m represents the dimensionality of each particle's position and velocity vectors). The variable t denotes the iteration number. $v_{ij}(t)$ refers to the velocity of the i^{th} particle at position j during iteration t. This velocity influences the particle's movement in the search space. The term $x_{ij}(t)$ represents the position of the i^{th} particle at the position j during iteration t. The position describes the particle's location within the search space. $c1$ and $c2$ represent the positive acceleration constants that are used to scale cognitive and social contributions, respectively, and $r_{1j}(t)$, $r_{2j}(t) \sim U\ (0, 1)$ represent random values within the range [0,1], sampled from a uniform distribution [51]. According to the parameters c_1 and c_2, the particle will follow one of two directions: (1) follow its own best position or (2) follow the swarm's best position. Thus, this determines whether the swarm is "explorative or exploitative" in nature [50]. Parameter w controls inertia for the movement of the swarm.

During the velocity update rule, particles compare their current positions to their nearest neighbours. Based on a distance metric, a k-dimensional tree determines the closest neighbours. As a result, the neighbours are calculated for each iteration. The nearest neighbours, however, can be changed to be equal to the number of particles in the swarm in order to produce a global-best PSO. By doing so, all particles will be able to see each other, which will result in a global best particle [52].

The position update rule is now decided by the following case expression [52],

$$X_{ij}(t+1) = \begin{cases} 0, & \text{if rand}() \geq S(v_{ij}(t+1)) \\ 1, & \text{if rand}() < S(v_{ij}(t+1)) \end{cases} \tag{13}$$

where rand() is the pseudo-random number selected from a uniform distribution over [0.0, 1.0] [53] Function $S(x)$ is the sigmoid function defined as [52],

$$S(x) = \frac{1}{1+e^x} \tag{14}$$

Algorithm 1 shows the PSO implementation used for the research. Additionally, the PSO parameters are viewed in Table 6 for each dataset.

C. GENETIC ALGORITHM

Genetic Algorithm is described as an Evolutionary Algorithm which finds the optimal solution in the process of natural selection and crossover. GA randomly generates individuals to produce an initial population. Each individual consists of a variable gene that signifies a solution to a specific problem encoded by a chromosome.

There are three main operators in the design of the GA: Selection, Crossover, and Mutation. The selection operator is the method of selecting next-generation individuals from the current generation. The selection operator is designed to select individuals with the highest Fitness Values and remove bad solutions. The crossover is the method of gene recombination which recombines two parents' chromosomes to generate new individuals to be used in the next generation. The mutation operator is the process of modifying one or more gene values that are randomly selected within the current chromosome. The generational process based on genetic operators is repeated to gradually evolve candidate solutions which converge on approximate solutions more and more. When the Genetic Algorithm process is terminated due to the given constraints, the optimum of the solutions is obtained to solve the problem [54].

As part of the GA implementation, the TPOT library was utilized. TPOT (Tree-based Pipeline Optimization Tool) is a tool specifically designed for creating optimal pipelines through the use of genetic programming. Preprocessor, Decomposition, Feature Selection, and Model are the four distinct "operators" used by TPOT. The operators are arranged "in a tree" with the leaves representing one or more copies of the input data.

Algorithm 1 PSO Implementation

```
$X_i$ = Stopword Removal;
$X_j$= Count Vectorizer;
$X_k$ = TFI-DF;
$Tr$ = ← Training Data;
$Te$ = ← Testing Data;
Define PSO():
    $s$ ← Index of Particles;
    $X_s$ ← Position of Particle $s$;
    $V_s$ ← Velocity of Particle $s$;
    $P_s$ ← Best position of Particle $s$;
    $P_g$ ← Global Best Position;
    Initialise PSO parameters:
    $C_1$ = Cognitive Parameter;
    $C_2$ = Social Parameter;
    $W$ = Weight;
    $i$ = NumberOfIteration;
    $p$ = NumberOfParticles;
    PSO Module ← Dimension($C_1, C_2, w, i, p$)
    PSO← Objective function
    Create instances of Machine learning classifier
    while $i < i_{max}$ do
        for p:=1 to number of particles do
            Update $V_s$ using Equation (10)
            Update $X_s$: using Equation (12)
            Evaluate Particle $s$ and update $P_s$
        end
        Update $P_g$
        $i = i + 1$
    end
    Calculate performance
    performance ← $P_g, Tr, Te$
    return Accuracy;
Define Objective Function():
    for t in test size do
        X_test and y_test = testing size;
        X_train and y_train = training size;
        Call PSO;
    end
```

TABLE 6. Particle swarm optimization parameters.

Dataset	c1	c2	w	k	p	alpha
Davidson	0.8	0.8	2	50	2	0. 9
Maryland LR	0.8	0.8	1	50	2	0.88
Maryland MNB	0.9	0.8	1.4	50	2	0.88
Formspring	0.8	0.8	1.2	100	2	0.88
OLID	0.9	0.9	1.2	100	2	0.9

The dataset is passed through the tree so that each feature evolves operator by operator, leading to the final node generating the best model (either a classification model or regression model). The following parameters were passed through TPOT:

TABLE 7. Genetic algorithm parameters.

Multinomial NB	alpha:[1e-3, 1e-2, 1e-1, 1., 10., 100.]	fit_prior: [True, False]	
Logistic Regression	penalty: ["l1", "l2"]	C: [1e-4, 1e-3, 1e-2, 1e-1, 0.5, 1., 5., 10., 15., 20., 25.]	dual: [True, False]

- Generation: Generation identifies the number of times the optimization process will be conducted. The default value is '100'. In regards to this study, the parameter was set to '10'.
- Population size: The initial population is generated randomly by default. The population size is the number of individuals participating in each generation. The parameter was set to '20'.
- Offspring size: '20'. The Offspring Size in TPOT represents the number of candidate pipelines produced in each iteration of the genetic programming algorithm. This parameter determines the size of the population of potential solutions that will be evaluated to identify the optimal pipeline for a specific problem. Although a larger offspring size may lead to improved results, it also increases the computation time.
- random_state: '42'. The "random state" parameter ensures reproducibility in the generation of randomized data splits.

Table 7 shows the parameters used for Multinomial Naive Bayes and Logistic Regression.

Additionally, Algorithm 2 shows the GA implementation used for the research.

D. PERFORMANCE METRICS

In order to evaluate the performance of the models, the following standard classification metrics were used: Accuracy, Precision, Recall, and F1-score. By identifying weaknesses within a single class of a problem, these metrics provide a deeper understanding of the classifier's behaviour. The metrics are defined in terms of true and false positives, as well as true and false negatives. When a true positive is identified, both the actual and projected classes are positive. A false positive, however, will identify an actual class of negative but an estimated class of positive. The performance evaluation techniques are described in equations 15-18 [55]

- Accuracy: The total number of correct predictions

$$\text{Accuracy} = \frac{\text{True Positive} + \text{True Negative}}{\text{Total Number of Observations}} \tag{15}$$

- Precision: The ability of a classifier to not label a positive when in fact it is negative

$$\text{Precision} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Positive}} \tag{16}$$

- Recall: The proportion of actual positive being identified correctly is given by recall

$$\text{Recall} = \frac{\text{True Positive}}{\text{True Positive} + \text{False Negative}} \tag{17}$$

Algorithm 2 Genetic Algorithm for Hyper-Parameter Tuning

```
$X_i$ = Stopword Removal;
$X_j$= Count Vectorizer;
$X_k$ = TFI-DF;
$Tr$ = ← Training Data;
$Te$ = ← Testing Data;
Define GeneticAlgorithm():
    population = initialise population(populationSize);
    chromosomeLength = len(population[0]);
    fittestOfGen = [];
    while generation < generationLimit do
        fitnessValues = [];
        //Evaluate the fitness F1 scores
        //Selection of the fittest chromosomes
        sorted = sort(fitnessValues);
        parent1 = sorted[0];
        parent2 = sorted[1];
        fittestOfGen.append(parent1);
        //Crossover of genes from the fittest parent
        offspring1 = [];
        offspring2 = [];
        for gene in chromosomeLength do
            if gene < (chromosomeLength/2) then
                offspring1.append(parent1[gene]);
                offspring2.append(parent2[gene]);
            else
                offspring1.append(parent2[gene]);
                offspring2.append(parent1[gene]);
                end
            end
            end
        end
        //Generate a new population from the offspring
        population = [];
        for index in populationSize do
            if index < populationSize/4 then
                chromosome = offspring1.copy();
            else if index < populationSize/2 then
                chromosome - offspring2.copy();
            else
                chromosome =
                  generateRandomChromosome();
                end
            end
            population.append(chromosome);
            end
        end
        //Mutate a random gene in each chromosome
        for each chromosome in population do
            mutate(chromosome);
            end
        end
        end
    end
bestChromosome = sort(fittestOfGen);
return bestChromosome;
for t in test size do
    X_test and y_test = testing size;
    X_train and y_train = training size;
    Call Genetic Algorithm;
end
```

- F1-score: The weighted mean of the Precision and Recall

$$\text{F1} = 2 \cdot \frac{\text{Precision}}{\text{Precision} + \text{Recall}} \tag{18}$$

VII. STAGE THREE OF FRAMEWORK

Upon applying the machine learning classifiers to the testing data, a probability score is produced that identifies how accurately the classifier predicted the correct sentiment. The data with a low probability score were then subjected to the Lexicon-Based Fuzzy Logic stage.

A. FUZZY LOGIC

The foundation of Fuzzy Logic can be traced back to the early stages of computer science as it was first proposed by Zadeh in 1965 [56]. In his paper, Zadeh described that the world cannot be viewed in a binary sense, and data cannot be described as black and white when the world is much more complex than that. Fuzzy Logic takes values between 0 and 1 and uses expressions written in IF-THEN rules [57]. Such rules allow machines to represent human thinking by looking at patterns. Fuzzy Logic is able to tackle uncertainty or vagueness in an extraordinarily productive way due to the presence of fuzziness. For example [58]

$$\text{IF } x \text{ is } A_i \text{ THEN } y \text{ is } B, i = 1, \dots, n \tag{19}$$

B. VADER

The research employed the sentiment lexicon VADER to analyze the testing dataset. With the rapid evolution of language, particularly among younger generations, several researchers have proposed methods to detect new slang and abbreviations used on social media platforms. Sentiment lexicons are lists of lexical features labelled according to their semantic orientation, which can be either positive or negative [59]. VADER, or Valence Aware Dictionary and Sentiment Reasoner, is a lexicon and rule-based sentiment analysis tool that is specifically designed to analyze sentiments expressed in social media. VADER is capable of handling words, slang, abbreviations, and emoticons that are commonly found in social media posts. It computes the overall score of a tweet by utilizing the lexicon's polarity_score, which determines the overall positive or negative sentiment of the text. Each piece of textual data generates a sentiment score vector that contains negative, neutral, positive, and compound polarities. The negative, neutral, and positive polarities are normalized to be between 0 and 1, with the compound polarity being the total sum of all sentiments found within the text, producing a normalized score between -1 (negative) and 1 (positive) [59]. For instance, the following text was processed using VADER: "make sure to bitch and whine like a girl about sexist emojis' instead of getting a degree that matters or a job". The VADER Lexicon will output: neg: 0.266, neu : 0.591, pos : 0.143, compound : -0.599 suggesting a negative sentiment in the text.

C. FUZZIFICATION

The positive and negative scores that were attained using the VADER Lexicon were transformed into a fuzzy representation using a triangular membership function. This function was utilized to depict the situation in question, and its

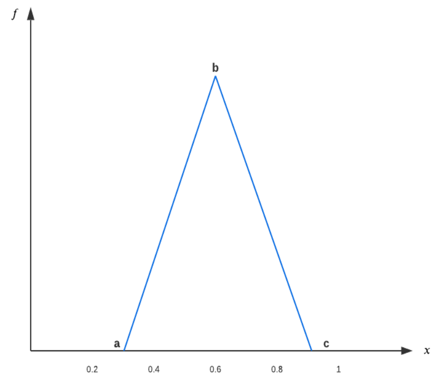

FIGURE 4. Fuzzy memberships.

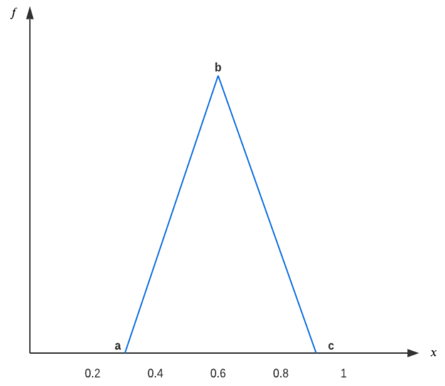

FIGURE 5. Fuzzy membership step 1.

selection is dependent on the specific parameters or dataset applied. Triangular membership functions are a popular choice in real-time applications due to their straightforward mathematical formula and efficient computational characteristics. The triangular membership function was employed because it encompasses three variables and establishes relationships between them. An illustration of a triangular membership function can be seen in Fig.4. To provide an example, the function considers three fuzzy variables a, b, and c where $(a \leq b \leq c)$ specifies the x-coordinates of the triangular membership function, within this case the co-ordinates are 0.3, 0.6 and 0.9 respectively.

A step-by-step example of the triangular fuzzy membership is shown in Fig. 5-7. In Fig.5, the X axis represents inputs from processes, while the Y axis represents the corresponding fuzzy values. When the input x equals b, it has full membership in the designated set, meaning that $f(x) = 1$ if $x = b$. On the other hand, if the input is less than a or greater than c, it does not belong to the fuzzy set and has a membership value of 0, as represented by $f(x) = 0$ for $x < a$ or $x > c$.

Furthermore, Fig.6 depicts the membership value of x, which falls within the range of a and b, and fluctuates between 0 and 1. As x approaches a, its membership value tends towards 0, and conversely, as x approaches b, its membership value approaches 1. The fuzzy value of x can be determined through the application of a similar triangle rule, which yields

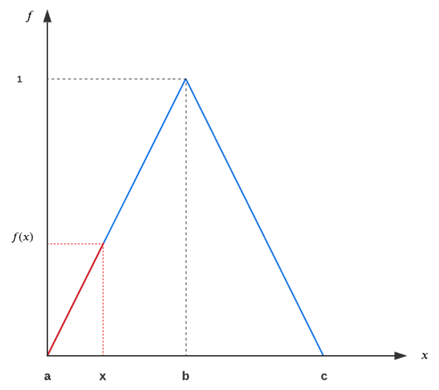

FIGURE 6. Fuzzy memberships step 2.

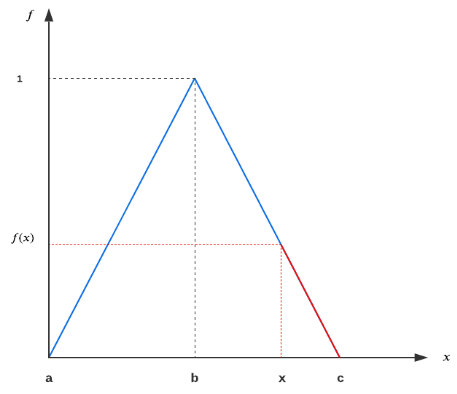

FIGURE 7. Fuzzy membership step 3.

the following expression:

$$f(x) = \frac{(x-a)}{(b-a)}, \text{ where } a \leq x \leq b. \tag{20}$$

Fig. 7 identifies that if x falls within the range of b and c, its membership value fluctuates between 0 and 1. If x is close to the value of b, its membership value approaches 1, and as x approaches the value of c, its membership value tends towards 0. The fuzzy value of x can be calculated using the similar triangle rule, which yields the following expression:

$$f(x) = \frac{(c-x)}{(c-b)}, \text{ where } b \leq x \leq c. \tag{21}$$

Once the linguistic terms are established, the next step is to perform fuzzification on the Positive, Negative and Neutral Scores. This is achieved by using the triangular membership functions described by equation (22) to calculate the membership degree f of the positive, negative and neutral scores in the lower, medium, and higher fuzzy sets. The triangular membership function [60] is characterized by a lower value (a), a medium value (b), and a higher value (c) with the constraint of $a \leq b \leq c$,

$$f(x; a, b, c) = \begin{cases} 0, & x \leq a \\ \frac{x-a}{b-a}, & a \leq x \leq b \\ \frac{c-x}{c-b}, & c \leq x \leq c \\ 0, & c < x \end{cases} \tag{22}$$

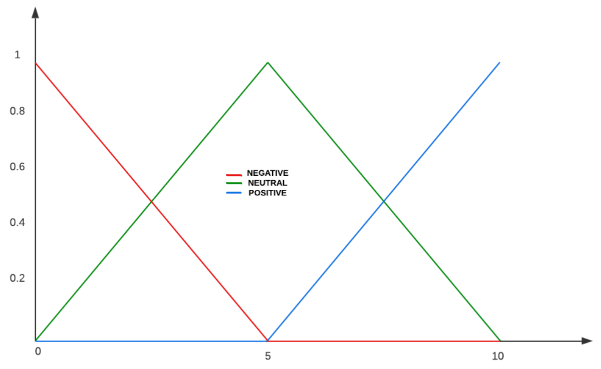

FIGURE 8. Fuzzy memberships for outputs.

$$= \max(\min(\frac{x-a}{b-a} - \frac{c-x}{c-b}), 0). \tag{23}$$

a, b, and c represent the x-coordinates of the triangle, x represents the crisp value from the isolated variable fuzzy universe of discourse.

Overall the output variable of the triangular membership, *output*, has a range of 0-10, with a minimum of 0 and a maximum of 10. The parameters for the three fuzzy sets (Negative, Neutral, and Positive), which represent the sentiment class, are:

- Negative (neg_sentiment): 0, 0, 5
- Neutral (neu_sentiment): 0, 5, 10
- Positive (pos_sentiment): 5, 10, 10

Each fuzzy set is described by a list of three parameters that establish the lower and upper bounds of the triangular membership function for that set. For instance, the Negative fuzzy set is defined by a triangular membership function that takes a value of 1 when output = 0, decreases linearly to 0 when output = 5, and remains at 0 beyond that point. Conversely, the Neutral fuzzy set has a value of 0 when output = 0, rises linearly to 1 when output = 5, and then decreases linearly to 0 when output = 10. Finally, the Positive fuzzy set begins with a value of 0 when output = 5 and rises linearly to 1 when output = 10, staying at 1 afterwards.

The sentiment of a piece of text can be inferred by fuzzy classification, which assesses the degree of membership of output in each fuzzy set. This computation involves evaluating the triangular membership function of each set according to the value of output and its parameters. The final sentiment class is determined based on the fuzzy set that the output variable has the highest degree of membership in.

Fig.8 depicts the fuzzy membership for outputs.

D. RULES BASED

The "spine" of any fuzzy logic system is the fuzzy rules applied. The Fuzzy Rule Base consists of several fuzzy rules which are the collection of conditional IF-THEN statements in linguistic form distinguishing how a Fuzzy Inference System should make decisions with respect to the inputs introduced to it. Fuzzy Rules are given in the form of [34],

$$R^1 = \text{IF } x_1 \text{ is } A_1 \text{ and } x_2 \text{ is } B_1 \text{ then } y \text{ is } C_1$$

TABLE 8. Fuzzy rules.

Rule	Positive	Negative	Output
1	low	low	neutral
2	low	medium	negative
3	low	high	negative
4	medium	low	positive
5	medium	medium	neutral
6	medium	high	negative
7	high	low	positive
8	high	medium	positive
9	high	high	neutral

$$R^2 = \text{IF } x_1 \text{ is } A_2 \text{ and } x_2 \text{ is } B_2 \text{ then } y \text{ is } C_2$$
$$\vdots$$
$$R^n = \text{IF } x_1 \text{ is } A_n \text{ and } x_2 \text{ is } B_n \text{ then } y \text{ is } C_n \tag{24}$$

where A_1 and A_2, are two different fuzzy membership functions for the input x_1 and B_1, B_2 for the input x_2, C_1, C_2 for the output y. The combination operator shown in the aforementioned Fuzzy Rule Base is "AND". However, there are various types of combination operators, such as "OR", "NOT", etc. Each combination operator gives a distinct type of Fuzzy Rule Base which, in turn, encodes distinct fuzzy representations in comprehension of the system model.

There are three common types of fuzzy rule-based systems, these rule-based systems are named after Mamdani, Sugeno, Tsukamoto, etc. [61]. The Mamdani fuzzy inference system is used and operates as follows: the first stage determines a set of fuzzy rules, the second stage fuzzifies inputs through the use of membership functions, and the fuzzy inputs are then combined based on the fuzzy rules to establish a rule strength. The consequences of the rules are found by merging the rule strength and output membership, the consequences are then combined to get an output distribution [61]. Once that occurs, defuzzification is applied.

Within research nine, different rules are constructed based on the assumption that the higher score (positive or negative) indicates the sentiment. In the case of situations where the scores are the same, the sentiment will remain neutral. The Rules can be viewed in Table 8.

Since decisions are based on testing all the rules, the rule outputs must be merged in some way. Aggregation is then applied which is the process by which the fuzzy sets that signify the outputs of each rule are combined into a single fuzzy set. Aggregation occurs once for each output variable, which is done prior to the final defuzzification step.

Equations 25-27 represent the firing strength or the level of satisfaction of the fuzzy rules associated with negative, positive and neutral sentences respectively.

$$\text{Neg} = \text{rule2} \vee \text{rule3} \vee \text{rule6} \tag{25}$$

$$\text{Pos} = \text{rule4} \vee \text{rule7} \vee \text{rule8} \tag{26}$$

$$\text{Neu} = \text{rule1} \vee \text{rule5} \vee \text{rule9} \tag{27}$$

For a new instance (W_i) to be classified using the fuzzy rules discussed above, it is necessary to identify the membership degree of W_i to the fuzzy set A_{nj} and B_{nj} in each dimension x_j. The firing strength of each rule is computed by combing the membership degrees of W_i to all fuzzy sets $(A_{1j}, A_{2j}, \ldots, A_{nj})$ and $(B_{1j}, B_{2j}, \ldots, B_{nj})$ which represent the j^{th} antecedent of Rule r_j. This is done using a T-norm $T \in [0, 1]$ such as min. Moreover, the membership degree for each class C_j is computed by combining the firing strengths of all rules of C_k, using a T-conorm $S \in [0, 1]$. The definition of T-norm and T-conorm can be viewed in equations 28 and 29 respectively [62]

$$T(\mu_A(x), \mu_B(x)) = \mu_{A\cap B} = \min(\mu_A(x), \mu_B(x)) \tag{28}$$

$$S(\mu_A(x), \mu_B(x)) = \mu_{A\cup B} = \max(\mu_A(x), \mu_B(x)) \tag{29}$$

T-norms and T-conorms are referred to as fuzzy norms and some popularly used ones include Min/Max norm, Product norm, Lukasiewicz norm and Yager norm [62].

Therefore, once the activation rules are established, they are implemented by using "*np.fmin*" to truncate the upper end of the output membership function that corresponds to each rule. Equations 30 - 32 show the membership functions of consequent parts of respective rules for negative, neutral and positive respectively

$$\text{activation\_low} = \text{neg} \wedge \text{neg\_sentiment} \tag{30}$$

$$\text{activation\_med} = \text{neu} \wedge \text{neu\_sentiment} \tag{31}$$

$$\text{activation\_high} = \text{pos} \wedge \text{pos\_sentiment} \tag{32}$$

The overall output is then obtained by aggregating activation_low + activation_med + activation_high.

E. DEFFUZIFICATION

The last step in the fuzzy rule system is defuzzification [61], [62]. The research implemented the Mean of Maximum (MOM). The purpose of MOM is to calculate the mean of all values that reach the maximum membership value in the corresponding fuzzy set. The Mean of Maximum applies the fuzzy output $A(x)$ by taking the mean of the x values at which $A(x)$ is maximized. Assuming that $x_1, x_2, \ldots x_n$ are the maximizing point of $A(x)$ then [62],

$$\text{MOM}[A(x)] = \frac{x_1, x_2, \ldots, x_n}{n} \tag{33}$$

In situations where the maximizing elements of $A(x)$ are between a and b then,

$$\text{MOM}[A(x)] = \frac{\int_a^b x dx}{\int_b^a dx} = \frac{(a+b)}{2} \tag{34}$$

The defuzzified output is checked using different ranges to classify the tweet according to its polarity: Negative, Neutral or Positive. Due to the research wanting to identify which text is deemed "Hateful", the positive and neutral

TABLE 9. Deffuzified output.

Tweet:	17 godfrey___ terrorists that are fucking muslim! like the last 100+ attacks across the planet! hence making it a muslim problem! retard!
VADER Output:	'neg': 0.445, 'neu': 0.468, 'pos': 0.087, 'compound': -0.9098
Negative Score:	0.8
Neutral Score:	0.2
Positive Score:	0.2
Activation low:	[0.8 0.8 0.6 0.4 0.2 0. 0. 0. 0. 0.]
Activation medium:	[0. 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2 0.2]
Activation high:	[0. 0. 0. 0. 0. 0. 0.2 0.2 0.2 0.2]
Aggregated Output:	[0.8 0.8 0.6 0.4 0.2 0.2 0.2 0.2 0.2 0.2]
Defuzzified Output:	3.09
Final Sentiment:	Negative

polarities are combined. The minimum and maximum of the output are 0 and 10 respectively. Therefore, negative data is retrieved when the score is between 0 - 3.33 whereas the scores between 3.34 - 10 will identify the Neutral and positive outcomes. This can be viewed in Equation 35

$$\text{Output} = \begin{cases} \text{Negative}, & 0 \leq x \leq 3.33 \\ \text{Positive + Neutral}, & 3.34 \leq x \leq 10 \end{cases} \tag{35}$$

An example of the deffuzified output can be seen in Table 9.

Whilst examining Table 9, it is evident that the VADER sentiment analysis is coupled with Fuzzy Logic to determine the sentiment of a tweet. The output of VADER is subsequently used as input for the Fuzzy Logic system, which performs further refinement of the sentiment classification.

For instance, the tweet "17 godfrey___ terrorists that are fucking muslim! like the last 100+ attacks across the planet! hence making it a muslim problem! retard!" undergoes VADER sentiment analysis, which produces scores for the negative, neutral, and positive sentiment components and a compound score representing the overall sentiment. In this particular case, the tweet elicits a high negative sentiment score (0.445) and a compound score of −0.9098, indicating a strong negative sentiment.

These firing strengths for Negative, Neutral and Positive are then computed using the fuzzy membership functions applied to the VADER scores. The firing strengths represent the extent to which the tweet portrays negative, neutral, and positive sentiments, respectively.

- Negative Score: 0.8
- Neutral Score: 0.2
- Positive Score: 0.2

Following the computation of the firing strengths, the aggregated output is generated by selecting the maximum activation of each membership function across the low, medium, and high categories. In this instance, the aggregated output is [0.80.80.60.40.20.20.20.20.20.2].

The defuzzified output is obtained through the "mom" (middle of maximum) method and amounts to a value of 3.09. Based on the defuzzified output, the sentiment classification for this tweet is Negative since the value falls within the range of 0 to 3.33.

In conclusion, the integration of VADER sentiment analysis and fuzzy logic yields a more refined analysis of the sentiment expressed in the tweet. VADER's output serves as the initial input for the fuzzy logic system, which, in turn, enhances the sentiment classification by accommodating the uncertainty and ambiguity inherent in natural language.

VIII. SETUP

The experiments described in this report were conducted utilizing the programming language Python 3.9.7, executed on the Anaconda platform, which provided a Jupyter Notebook environment for program implementation. To conduct experiments involving linear regression, Google Colab was employed due to its ability to accelerate the training and testing process across multiple datasets. Colab hosted Jupyter Notebook service that enables the execution of arbitrary Python code via a web browser and is particularly suitable for machine learning, data analysis, and educational applications. Moreover, it requires no setup and provides free access to computing resources, including GPUs.

The study was conducted on a standard laptop equipped with 16 GB RAM and an M1 Pro CPU. The Scikit-Learn module was employed for implementation, an environment integrated with the Python programming language that offers a comprehensive collection of supervised algorithms for research purposes. The library utilizes a high-level approach for training through "fit" methods and for making predictions via estimators (classifiers) through the "predict" method.

To optimize the accuracy of machine learning algorithms, libraries such as PySwarms for Particle Swarm Optimization and TPOT for Genetic Algorithm were utilized. The entire program required approximately a day to run all models across various platforms. Additionally, the fuzzy implementation was performed using Scikit-Fuzzy, which is a collection of fuzzy logic algorithms.

IX. DISCUSSION AND RESULTS

As previously indicated, the first step of the research was to pre-process the text found in the datasets. This process removed any noise that does not hold any value to the data. Data were pre-processed using various techniques such as tokenization, stemming, and stop-word removal. Table 10 shows an example of the data before and after pre-processing.

Once the pre-processing stage was complete, the text was converted into numerical representations using Scikit-Learn CountVectorizer and TfidfTransformer. By converting text into a numerical representation, it allows algorithms to have a better understanding of the data that it is fed. TF-IDF (term frequency-inverse document frequency) is a statistical measure that evaluates how relevant a word is to a document. This is done by multiplying two metrics, the first is how many times a word appears in a document, and the second

TABLE 10. Pre-processing.

Before Preprocessing	Label	After Preprocessing
1948Army of England helped the Jews to occupy...	1	army england helped jews occupy palestine afte...
@Grumpy @deanesmay feminists argue for...	1	feminists argue raising minimum wage cause wom...
@rlac72 #WHITELIVES-MATTER!	0	whitelivesmatter
In the Stone Age there was no word for cunt. B...	1	in stone age word cunt but muslims either

TABLE 11. Logistic regression results.

Dataset	Accuracy	Precision	Recall	F1-macro	F1-weighted
Davdison	0.99302	0.9077	0.8324	0.8639	0.9267
Maryland	0.7660	0.7313	0.5668	0.5590	0.7062
FormSpring	0.9483	0.9224	0.5904	0.6380	0.9318
OLID	0.7525	0.7752	0.6517	0.6604	0.7185

TABLE 12. Multinomial naive bayes results.

Dataset	Accuracy	Precision	Recall	F1-macro	F1-weighted
Davdison	0.8521	0.9073	0.5657	0.5755	0.8019
Maryland	0.7508	0.7822	0.5168	0.4629	0.6545
FormSpring	0.9389	0.9694	0.5042	0.4926	0.9080
OLID	0.7092	0.7737	0.5740	0.5457	0.6353

is the inverse document frequency of the word across a set of documents. Upon transforming the text Logistic Regression and Multinomial Naive Bayes were applied.

The results of Logistic Regression and Multinomial Naive Bayes can be viewed in Tables 11 and 12, respectively. The results show that throughout all four datasets, Logistic Regression performed significantly better than Multinomial Naive Bayes. It must be acknowledged that due to the highly imbalanced nature of the datasets, the results identified a high accuracy (between 70-90%). However, the macro f1-score was low, as it ranged between 46-60%. As previously mentioned, the F-1 score is simply the harmonic mean between precision and recall. Within this research, Macro-f1 and Weighted-f1 are used. The macro-averaged F1 score is computed using the arithmetic mean (unweighted mean) of all the per-class F1 scores. This method treats all classes equally regardless of their support values [63]. At the same time, weighted F1-scores are calculated for each class individually and then weighted by the class count [63]. Therefore, all samples are weighed equally.

After evaluating the performance of the Machine Learning classifiers, four different hybrid models were proposed. The first two models (LG-Fuzzy-PSO and LG-Fuzzy-GA) were designed to directly improve the results of Logistic Regression, while the other two models (NB-Fuzzy-PSO and NB-Fuzzy-GA) were aimed to enhance the results of Multinomial Naive Bayes.

- LG-Fuzzy-PSO uses Particle Swarm Optimization in combination with Logistic Regression and Fuzzy Logic
- LG-Fuzzy-GA uses Genetic Algorithm in combination with Fuzzy Logic and Logistic Regression
- NB-Fuzzy-PSO: Uses Particle Swarm Optimization in combination with Fuzzy Logic and Multinomial Naive Bayes
- NB-Fuzzy-GA: uses Genetic Algorithm in combination with Fuzzy Logic and Multinomial Naive Bayes

Prior to employing Particle Swarm Optimization (PSO) and Genetic Algorithms (GA), the complexity associated with machine learning classifiers, such as Logistic Regression and Multinomial Naive Bayes, is comparatively low. Both classifiers represent straightforward and effective techniques for addressing binary classification problems.

Logistic Regression (LR), a linear model, leverages the logistic function (or sigmoid function) to approximate the probability of a specific class or event. It identifies the optimal coefficients that minimize the discrepancy between predicted probabilities and actual classes. The LR algorithm's computation is primarily based on matrix multiplication and inversion operations used during training, which are influenced by the number of samples and features in the dataset.

Multinomial Naive Bayes (MNB), a probabilistic classifier, relies on the application of Bayes' theorem with stringent independence assumptions between features. This classifier is particularly well-suited for discrete data and performs effectively in text classification scenarios. The computation of MNB is dependent on the calculation of probabilities for each feature and class, which are determined by the number of samples, features, and classes.

The integration of PSO and GA into classifiers increases their complexity due to the additional components introduced by these optimization techniques. Pyswarms Binary PSO, an optimization method inspired by the social behaviour observed in bird flocking or fish schooling, searches for the optimal solution by updating candidate solutions (particles) based on their velocities and positions within the search space. The complexity of PSO is typically $O(n \cdot s \cdot i)$, where n is the number of particles, s is the dimension of the search space, and i represents the number of iterations.

In contrast, Hyperparameter Tuning with GA, is an optimization technique grounded in the principle of natural selection, generates a population of candidate solutions (chromosomes) using operations such as crossover, mutation, and selection, in order to identify the most advantageous set of hyperparameters for the classifier. The complexity of GA is $O(n \cdot g \cdot f)$, where n represents the population size, g denotes the number of generations, and f corresponds to the complexity of the fitness function evaluation.

In conclusion, incorporating bio-inspired optimization techniques such as PSO and GA into LR and MNB classifiers leads to increased computational demands. However, this increase in computational requirements may contribute to improved performance and enhanced classification outcomes for online hate text, justifying the inclusion of these optimization techniques in the study. Nonetheless, the optimization

TABLE 13. Testing dataset after incorporating PSO's predicted label and predict_proba.

Predicted Probability	Actual Label	Tweet	PSO
0.568206	1	adolf hilter kill all the jews shawn oakman i...	1
0.819998	0	th july white people everyone loves christmass...	0
0.758001	0	loving bible series jesus king king jews king me	0
0.661161	1	i blaming snobby little cunts went calling eve...	0

TABLE 14. Testing dataset after incorporating GA's predicted label and predict_proba.

Predicted Probability	Actual Label	Tweet	GA
0.631712	0	stop wasting time breath he complete waste...	1
0.766207	0	that cause got head wave least pounds couple...	0
0.704179	0	gun control anyone disarmhate...	1
0.661161	0	i while republicans difference agw belief phds c...	0

process requires greater computational resources and time compared to standard classifiers.

During the applications of Genetic Algorithm and Particle Swarm Optimization, Sklearn's "predict_proba" method was utilized with each optimization technique prior to Fuzzy Logic being employed. The "predict_proba" technique returns the class probabilities for each data point when the optimization technique is applied. It works by accepting a single argument that corresponds to the data over which the probabilities will be computed and returns an array of lists containing the class probabilities. The results generated by PSO and GA were integrated into the testing datasets, along with the results obtained from "predict_proba". Examples of the results can be viewed in Tables 13 and 14.

Finally, Fuzzy Logic is applied to the testing data's tweets which have a low "predicted probability" score. The first stage of the fuzzy logic implementation was to use the VADER Lexicon. VADER'S "polarity_score" identified how positive and negative strings of text are using a scale of 0 to 1. A triangular fuzzy membership for universe variables of positive, negative and neutral was applied using the fuzzy sets Low, Medium and High. Moreover, the fuzzy rules were incorporated and the firing strength of the tweets was evaluated. Since decisions are based on testing all the rules, the rule outputs must be merged in some way. Aggregation is applied as it is the process by which the fuzzy sets that signify the outputs of each rule are combined into a single fuzzy set. Lastly, defuzzification occurred using the Mean of Maximum method. Table 15 shows the resulting testing output after the implementation of Fuzzy Logic. The "predicted probability" represents the "predict_prob"a results when optimization was applied, "Label" identifies the actual

TABLE 15. Testing output after fuzzy logic is applied.

Predicted Probability	Label	Tweet	PSO	Fuzzy	Final Output
0.741060	0	long har look good men	0	0	0
0.538222	0	bother people ask questions answer bio nahhh	1	0	0
0.937763	0	u ever feared one ur parents wud take u middle...	0	1	0
0.547839	0	provided rednecks treated way hitler treated j...	1	0	0

TABLE 16. Davidson results.

Model	Accuracy	Precision	Recall	F1-Macro	F1-weighted
NB	0.8521	0.9073	0.5657	0.5755	0.8019
NB-Fuzzy-PSO	0.8799	0.9015	0.6560	**0.7024**	**0.8543**
NB-Fuzzy-GA	0.9030	0.8519	0.7794	**0.8975**	**0.9351**
LR	0.99302	0.9077	0.8324	0.8639	0.9267
LR-Fuzzy-PSO	0.9446	0.8894	0.9256	**0.9061**	**0.9458**
LR-Fuzzy-GA	0.9352	0.8894	0.8770	**0.8831**	**0.9346**
Fuzzy	0.5852	0.5788	0.6409	0.5303	0.6369

label of the corresponding Tweet, "PSO" acknowledges the results obtained using the Bio-inspired Optimization method and "Fuzzy" identifies the output implemented by the Fuzzy Implementation. The "Final Output" distinguishes the final label for the text. Furthermore, the determination of the final polarity relied on the accuracy score of the PSO/GA, irrespective of the correctness of the ultimate result. The sentence undergoes processing through the VADER Fuzzy Logic system, which yields the final output for the given input sentence. This approach facilitates the utilization of both a mathematical perspective, generated through machine learning optimization methods, and a more critical thinking-oriented perspective, conducted via VADER-Fuzzy when the optimized machine learning exhibits a low predicted probability.

Tables 16-19, present the results of the models incorporating Fuzzy genetic Algorithm against LR, MNB and Fuzzy Logic by itself. The models under consideration include Naive Bayes with Fuzzy Particle Swarm Optimization (NB-Fuzzy-PSO), Naive Bayes with Fuzzy Genetic Algorithm (NB-Fuzzy-GA), Logistic Regression with Fuzzy Particle Swarm Optimization (LR-Fuzzy-PSO), Logistic Regression with Fuzzy Genetic Algorithm (LR-Fuzzy-GA). The primary focus of this work is the F1 scores for each dataset due to their imbalanced nature. The model with the highest performance is highlighted in bold font.

The proposed models have consistently demonstrated superior F1 scores across all datasets. Specifically, the NB

TABLE 17. Maryland results.

Model	**Accuracy**	**Precision**	**Recall**	**F1-macro**	**F1-weighted**
NB	0.7508	0.7822	0.5168	0.4629	0.6545
NB-Fuzzy-PSO	0.7341	0.6039	0.5436	**0.5334**	**0.6825**
NB-Fuzzy-GA	0.7472	0.6518	0.5284	**0.4951**	**0.6690**
LR	0.7660	0.7313	0.5668	0.5590	**0.7062**
LR-Fuzzy-PSO	0.7083	0.6062	0.5969	**0.6005**	0.7016
LR-Fuzzy-GA	0.7405	0.6324	0.5768	**0.5808**	**0.7069**
Fuzzy	0.6332	0.5439	0.5480	0.5448	0.6412

TABLE 18. Formspring results.

Model	**Accuracy**	**Precision**	**Recall**	**F1-macro**	**F1-weighted**
NB	0.9389	0.9694	0.5042	0.4926	0.9080
NB-Fuzzy-PSO	0.9389	0.9694	0.5042	0.4926	0.9080
NB-Fuzzy-GA	0.9391	0.9695	0.5064	**0.4968**	**0.9104**
LR	0.9483	0.9224	0.5904	0.6380	0.9318
LR-Fuzzy-PSO	0.9498	0.9031	0.6110	**0.6648**	**0.9358**
LR-Fuzzy-GA	0.9559	0.8761	0.6895	**0.7477**	**0.9486**
Fuzzy	0.8665	0.5796	0.6487	0.5970	0.8864

TABLE 19. OLID results.

Model	**Accuracy**	**Precision**	**Recall**	**F1-macro**	**F1-weighted**
NB	0.7092	0.7737	0.5740	0.5457	0.6353
NB-Fuzzy-PSO	0.7218	0.7190	0.6136	0.6126	0.6802
NB-Fuzzy-GA	0.7329	0.7409	0.6269	**0.6294**	**0.6938**
LR	0.7525	0.7752	0.6517	0.6604	**0.7185**
LR-Fuzzy-PSO	0.7349	0.7124	0.6535	**0.6628**	0.7140
LR-Fuzzy-GA	0.7538	0.7348	0.6823	**0.6941**	**0.7385**
Fuzzy	0.6909	0.6433	0.6237	0.6286	0.6796

model was optimized using bio-inspired algorithms and subsequently subjected to Fuzzy Logic, resulting in a significant improvement in the performance of both NB-Fuzzy-PSO and NB-Fuzzy-GA. Remarkably, the integration of Fuzzy Logic into the NB-Fuzzy-GA model led to a remarkable upsurge in macro F1 scores, with an improvement of up to 32% observed across all four datasets. This is evidenced in Table 16, where the NB-Fuzzy-GA model demonstrated a substantial increase in macro F1 scores compared to the conventional ML techniques with optimization models, highlighting the efficacy of Fuzzy Logic in accurately classifying hate-related data.

In general, the models incorporating the Fuzzy Genetic Algorithm tend to exhibit better performance in terms of both F1-macro and F1-weighted, indicating their improved balance between precision and recall.

As illustrated in Tables 16-19, the performance of Fuzzy Logic as a standalone model is evident. It was observed that in the majority of datasets, Fuzzy Logic yielded superior results compared to the Naive Bayes model in terms of Macro-F1 score, excluding the Maryland dataset. Furthermore, the Macro-F1 score of Fuzzy Logic was higher than that of the Naive Bayes with Fuzzy Particle Swarm Optimization (NB-PSO-FUZZY) and Naive Bayes with Fuzzy Genetic Algorithm (NB-GA-FUZZY) for the Formspring and Maryland datasets. Additionally, in the OLID dataset, Fuzzy Logic outperformed both the Naive Bayes model and the NB-PSO-FUZZY model.

The PSO-based feature selection approach has shown promise in improving the accuracy of ML classifiers through the identification of pertinent features while concurrently removing irrelevant or redundant ones. Furthermore, such a technique can help reduce the complexity of the model, resulting in greater efficiency and interpretability, as well as reduced training times, particularly in time-critical scenarios. Additionally, the ability of PSO-based feature selection to handle incomplete or noisy data renders it a viable option for real-world ML applications. Similarly, the employment of GA-based hyperparameter tuning methodologies presents several benefits. Firstly, it can aid in determining the optimal set of hyperparameters to maximize model performance by selecting the most appropriate set of hyperparameters from an extensive pool of candidates, thereby reducing model complexity and increasing interpretability. Secondly, GA-based techniques can improve the efficiency of hyperparameter tuning processes by minimizing the number of evaluations needed to attain the optimal solution. Finally, the scalability of GA-based approaches enables their applicability to large datasets and complex models. In summary, the use of GA-based techniques in hyperparameter tuning offers notable benefits, including improvements in accuracy, efficiency, interpretability, and scalability, making it a valuable tool for a wide range of ML applications.

The utilization of the VADER lexicon, coupled with Fuzzy Logic, has shown promise in addressing issues such as emojis, slangs, and acronyms, while simultaneously evaluating the emotional content of textual data. This approach offers significant benefits, particularly in scenarios where the data is complex, such as micro-blogging websites. Additionally, Fuzzy Logic enables the handling of linguistic uncertainty, which is often present in datasets, by managing both ambiguity and uncertainty in textual data. As a result, the application of VADER and Fuzzy Logic can yield substantial improvements in the analysis of textual data, making it a valuable tool for a range of applications.

The present study investigates three methods for feature selection, and it is worth noting that each of them exhibits certain limitations. Firstly, the Particle Swarm Optimization (PSO) method is subject to computational complexity, representing a primary drawback. Specifically, a large number of fitness evaluations are required to identify an optimal feature subset, rendering PSO computationally expensive, especially when large datasets are involved. Additionally, PSO's scalability may be limited when applied to datasets with a significant number of features. As the number of features increases, the search space expands, making it more difficult for PSO to identify an optimal subset of features.

Secondly, PSO may converge on a suboptimal solution, implying that it may not always identify the globally optimal feature subset and may instead converge on a local optimum. This can lead to decreased classification accuracy, particularly when the dataset is complex and has a large number of features.

Lastly, PSO assumes that all features are of equal importance, which may not be the case in practice. Some features may have a greater impact on classification accuracy than others, and PSO may not capture this. Therefore, PSO may not always select the most informative features for classification, leading to reduced accuracy.

One of the primary limitations of the Genetic Algorithm (GA) is its computational expense, especially for large datasets and complex models. GA requires a significant number of fitness evaluations to identify an optimal set of hyperparameters, which can be time-consuming and resource-intensive. Consequently, GA may not be practical for applications with tight time or computational constraints.

Additionally, GA may suffer from premature convergence, which occurs when the GA population converges to a suboptimal solution before identifying the globally optimal set of hyperparameters. This can lead to decreased classification accuracy, as GA may not always identify the best set of hyperparameters.

Furthermore, GA requires careful selection of genetic operators, such as crossover and mutation, to strike a suitable balance between exploration and exploitation. Inadequately chosen operators can result in GA converging too quickly or not exploring the search space efficiently, leading to suboptimal outcomes.

Lastly, GA assumes that the fitness function is well-defined and can be evaluated efficiently. However, in some situations, evaluating the fitness function may be computationally expensive or time-consuming, making GA unfeasible or impractical.

Regarding Fuzzy logic, certain limitations must be overcome, such as sensitivity to parameter settings. The performance of a Fuzzy logic model can be sensitive to the choice of membership functions and fuzzy sets, which can be challenging to select and tune. Small parameter changes can result in significant changes in the model's output, making it challenging to achieve reliable and robust results. Additionally, Fuzzy logic models may overfit the training data if the model is too complex or if the data is noisy, leading to poor generalization performance and reduced accuracy when applied to new data. Finally, Fuzzy logic may not be suitable for all types of problems, as it is most effective in dealing with problems with imprecise or uncertain data. Problems that require precise numerical calculations or those with well-defined decision rules may be better suited to other models.

X. CONCLUSION

This study proposes an optimized machine learning - fuzzy logic approach for identifying hate speech in social media posts. The novelty of the approach lies in the incorporation of bio-inspired optimization techniques along with fuzzy logic to facilitate a deeper understanding of the linguistic aspects of the text. The proposed approach offers several advantages, such as the reduction of data dimensionality resulting from the implementation of optimization, which accelerates the classification process. Additionally, applying fuzzy logic resolves linguistic issues and provides a better understanding of text sentiment.

Genetic Algorithms (GA) have gained immense popularity in various fields due to their ability to solve complex engineering system problems. However, GA has a high implementation cost and usually requires a higher number of iterations than other algorithms. Particle swarm optimization (PSO) has emerged as an important heuristic algorithm based on the behaviour of swarming characteristics of living organisms. Both GA and PSO are evolutionary search methods that refine values over time using probabilistic and deterministic rules to improve them over time.

The research combines the two optimization models with fuzzy logic independently on four publicly available datasets: Maryland, Davidson, Formspring, and OLID. Compared to two state-of-the-art supervised machine learning classifiers, Logistic Regression and Multinomial Naive Bayes, the optimized fuzzy rule-based method consistently outperforms them with regard to accuracy and F1 scores. VADER performs exceptionally well with social media datasets due to its ability to handle vocabularies, abbreviations, capitalization, repeated punctuation, and emoticons commonly used to convey sentiment on social media platforms. However, the research found that using the LR-Fuzzy-GA system overall achieved the highest results.

The proposed methodology enables the Fuzzy logic-based system to handle the vagueness and ambiguity found within the text. In addition to providing a method for dealing with linguistic problems, fuzzy logic also deals with reasoning and provides a deeper understanding of the sentiment values in a sentence, thus offering a more reliable method for handling linguistic problems. It remains a challenge to accurately set fuzzy guidelines. Within the research fuzzy logic model activates for each sentence based on the results of the "predicted probability" of both GA and PSO when used with various ML classifiers. This aids in the classification of text when there is uncertainty in the optimization models. However, there were instances where GA and PSO correctly classified text, but as the predicted probability score was low, the text had to undergo classification through fuzzy logic.

Furthermore, the highly imbalanced datasets used in the research necessitated the use of F1 scores as a more reliable source of assessing the effectiveness of the methods. Although the research increased the results of ML algorithms, there is still room for improvement. Most ML algorithms perform poorly when the class distribution is imbalanced and require modification to avoid simply predicting the majority class in all situations. Future work will examine General Adversarial Networks (GANs), a deep generative reinforcement learning model that addresses the challenge of imbalance by augmenting the dataset with hateful tweets. This will be done by employing a two-component framework: a generator network and a discriminator network.

In addition, future work will examine sarcasm detection, a narrow area of NLP research that focuses on detecting sarcasm in a given text. Unlike sentiment analysis, where the sentiment categories are clearly defined, sarcasm has no clearly defined boundaries. Therefore, it is crucial to define what sarcasm is before attempting to detect it. However, current research shows that models often fail to detect sarcastic tweets due to their specificity to particular situations and cultures, requiring a high level of world knowledge that is not available to them. Sarcasm is often disguised as politeness, which makes it even more challenging to detect.

REFERENCES

[1] J. Hani, M. Nashaat, M. Ahmed, Z. Emad, E. Amer, and A. Mohammed, "Social media cyberbullying detection using machine learning," *Int. J. Adv. Comput. Sci. Appl.*, vol. 10, no. 5, pp. 703–707, 2019.

[2] B. Vidgen, E. Burden, and H. Margetts, "Social media cyberbullying detection using machine learning," Alan Turing Inst., London, U.K. Tech. Rep, Feb. 2022. [Online]. Available: https://www.ofcom.org.uk/__data/assets/pdf_file/0022/216490/alan-turing-institute-report-understanding-online-hate.pdf

[3] *4.4.1 A Sampling of Cyberbullying Laws Around the World*. Accessed: Nov. 1, 2023. [Online]. Available: https://socialna-akademija.si/joiningforces/4-4-1-a-sampling-of-cyber-bullying-laws-around-the-world/

[4] *The EU code of Conduct on Countering Illegal Hate Speech Online*. Accessed: Nov. 1, 2022. [Online]. Available: https://commission.europa.eu/strategy-and-policy/policies/justice-and-fundamental-rights/combatting-discrimination/racism-and-xenophobia/eu-code-conduct-countering-illegal-hate-speech-online_en

[5] K. Dinakar, R. Reichart, and H. Lieberman, "Modeling the detection of textual cyberbullying," in *Proc. Int. AAAI Conf. Web Social Media*, vol. 5, no. 3, Barcelona, Spain, 2011, pp. 11–17.

[6] A. Kontostathis, K. Reynolds, A. Garron, and L. Edwards, "Detecting cyberbullying: Query terms and techniques," in *Proc. 5th Annu. ACM Web Sci. Conf.*, May 2013, pp. 195–204.

[7] D. Yin, Z. Xue, L. Hong, B. D. Davison, A. Kontostathis, and L. Edwards, "Detection of harassment on web 2.0," in *Proc. Content Anal. Web*, Madrid, Spain, 2009, pp. 1–7.

[8] M. Dadvar, F. D. Jong, R. Ordelman, and D. Trieschnigg, "Improved cyberbullying detection using gender information," in *Proc. 25th Dutch-Belgian Inf. Retr. Workshop*, Ghent, Belgium, 2012, pp. 1–3.

[9] M. Dadvar, R. Ordelman, F. De Jong, and D. Trieschnigg, "Towards user modelling in the combat against cyberbullying," in *Proc. 17th Int. Conf. Appl. Natural Lang. Process. Inf. Syst.*, 2012, pp. 277–283.

[10] K. Reynolds, A. Kontostathis, and L. Edwards, "Using machine learning to detect cyberbullying," in *Proc. 10th Int. Conf. Mach. Learn. Appl. Workshops*, Honolulu, HI, USA, Dec. 2011, pp. 241–244.

[11] H. Hosseinmardi, S. A. Mattson, R. Rafiq, R. Han, Q. Lv, and S. Mishra, "Poster: Detection of cyberbullying in a mobile social network: Systems issues," in *Proc. 13th Annu. Int. Conf. Mobile Syst., Appl., Services*, May 2015, p. 481.

[12] D. Chatzakou, N. Kourtellis, J. Blackburn, E. De Cristofaro, G. Stringhini, and A. Vakali, "Mean birds: Detecting aggression and bullying on Twitter," in *Proc. ACM Web Sci. Conf.*, New York, NY, USA, Jun. 2017, pp. 13–22.

[13] M. A. Al-Garadi, K. D. Varathan, and S. D. Ravana, "Cybercrime detection in online communications: The experimental case of cyberbullying detection in the Twitter network," *Comput. Hum. Behav.*, vol. 63, pp. 433–443, Oct. 2016.

[14] V. S. Babar and R. Ade, "A review on imbalanced learning methods," *Int. J. Comput. Appl.*, vol. 975, no. 2, pp. 23–27, 2015.

[15] N. Aggrawal, "Detection of offensive tweets: A comparative study," *Comput. Rev. J.*, vol. 1, no. 1, pp. 75–89, 2018.

[16] I. Kayes, N. Kourtellis, D. Quercia, A. Iamnitchi, and F. Bonchi, "The social world of content abusers in community question answering," in *Proc. 24th Int. Conf. World Wide Web*, Florence, Italy, May 2015, pp. 570–580.

[17] P. Fortuna, "Automatic detection of hate speech in text: An overview of the topic and dataset annotation with hierarchical classes," M.S. thesis, Dept. Engenharia, Univ. Porto, Porto, Portugal, 2017.

[18] S. O. Sood, J. Antin, and E. Churchill, "Using crowdsourcing to improve profanity detection," in *Proc. AAAI Spring Symp.*, Stanford, CA, USA, 2012, pp. 69–74.

[19] R. Zhao, A. Zhou, and K. Mao, "Automatic detection of cyberbullying on social networks based on bullying features," in *Proc. 17th Int. Conf. Distrib. Comput. Netw.*, Jan. 2016, Art. no. 43.

[20] V. Nahar, S. Unankard, X. Li, and C. Pang, "Sentiment analysis for effective detection of cyber bullying," in *Proc. Asia–Pacific Web Conf.*, 2012, pp. 767–774.

[21] V. Nahar, X. Li, and C. Pang, "An effective approach for cyberbullying detection," *Commun. Inf. Sci. Manage. Eng.*, vol. 3, no. 5, p. 238, 2013.

[22] H. Hosseinmardi, R. I. Rafiq, R. Han, Q. Lv, and S. Mishra, "Prediction of cyberbullying incidents in a media-based social network," in *Proc. IEEE/ACM Int. Conf. Adv. Social Netw. Anal. Mining (ASONAM)*, Aug. 2016, pp. 186–192.

[23] R. Duwairi, A. Hayajneh, and M. Quwaider, "A deep learning framework for automatic detection of hate speech embedded in Arabic tweets," *Arabian J. Sci. Eng.*, vol. 46, no. 4, pp. 4001–4014, Apr. 2021.

[24] A. Al-Hassan and H. Al-Dossari, "Detection of hate speech in Arabic tweets using deep learning," *Multimedia Syst.*, vol. 28, no. 6, pp. 1963–1974, Dec. 2022.

[25] G. Rizos, K. Hemker, and B. Schuller, "Augment to prevent: Short-text data augmentation in deep learning for hate-speech classification," in *Proc. 28th ACM Int. Conf. Inf. Knowl. Manage.*, Beijing, China, Nov. 2019, pp. 991–1000.

[26] H. Mulki, H. Haddad, C. B. Ali, and H. Alshabani, "L-HSAB: A levantine Twitter dataset for hate speech and abusive language," in *Proc. 3rd Workshop Abusive Lang. Online*, Florence, Italy, 2019, pp. 111–118.

[27] P. Badjatiya, S. Gupta, M. Gupta, and V. Varma, "Deep learning for hate speech detection in tweets," in *Proc. 26th Int. Conf. World Wide Web Companion*, Kuala Lumpur, Malaysia, 2017, pp. 759–760.

[28] S. Dowlagar and R. Mamidi, "HASOCOne@FIRE-HASOC2020: Using BERT and multilingual BERT models for hate speech detection," in *Proc. Forum Inf. Retr. Eval. (FIRE)*, Hyderabad, India, 2021, pp. 1–8.

[29] R. Primartha, B. A. Tama, A. Arliansyah, and K. J. Miraswan, "Decision tree combined with PSO-based feature selection for sentiment analysis," *J. Phys., Conf.*, vol. 1196, Nov. 2018, pp. 1–6.

[30] R. H. Saputra and B. Prasetyo, "Improve the accuracy of C4.5 algorithm using particle swarm optimization (PSO) feature selection and bagging technique in breast cancer diagnosis," *J. Soft Comput. Explor.*, vol. 1, no. 1, pp. 47–55, 2020.

[31] R. Z. Al-Abdallah and A. T. Al-Taani, "Arabic single-document text summarization using particle swarm optimization algorithm," *Proc. Comput. Sci.*, vol. 117, pp. 30–37, Jan. 2017.

[32] C. Baydogan and B. Alatas, "Metaheuristic ant lion and moth flame optimization-based novel approach for automatic detection of hate speech in online social networks," *IEEE Access*, vol. 9, pp. 110047–110062, 2021.

[33] H. Rosa, J. P. Carvalho, P. Calado, B. Martins, R. Ribeiro, and L. Coheur, "Using fuzzy fingerprints for cyberbullying detection in social networks," in *Proc. IEEE Int. Conf. Fuzzy Syst. (FUZZ-IEEE)*, Rio de Janeiro, Brazil, Jul. 2018, pp. 1–7.

[34] H. Liu, P. Burnap, W. Alorainy, and M. L. Williams, "A fuzzy approach to text classification with two-stage training for ambiguous instances," *IEEE Trans. Computat. Social Syst.*, vol. 6, no. 2, pp. 227–240, Apr. 2019.

[35] S. Vashishtha and S. Susan, "Fuzzy rule based unsupervised sentiment analysis from social media posts," *Expert Syst. Appl.*, vol. 138, Dec. 2019, Art. no. 112834.

[36] S. Vashishtha and S. Susan, "Neuro-fuzzy network incorporating multiple lexicons for social sentiment analysis," *Soft Comput.*, vol. 26, no. 9, pp. 4487–4507, May 2022.

[37] J. Golbeck et al., "A large labeled corpus for online harassment research," in *Proc. ACM Web Sci. Conf.*, Troy, NY, USA, Jun. 2017, pp. 229–233.

[38] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. AAAI ICWSM*, 2017, pp. 512–515.

[39] M. Zampieri, S. Malmasi, P. Nakov, S. Rosenthal, N. Farra, and R. Kumar, "Predicting the type and target of offensive posts in social media," in *Proc. Conf. North Amer. Chapter Assoc. Comput. Linguistics, Hum. Lang. Technol.*, vol. 1, Minneapolis, MN, USA, 2019, pp. 1415–1420.

[40] Z. Zhu. *A Step-by-Step Tutorial for Conducting Sentiment Analysis*. Accessed: Nov. 3, 2022. [Online]. Available: https://towardsdatascience.com/a-step-by-step-tutorial-for-conducting-sentiment-analysis-9d1a054818b6

[41] S. Gibson, B. Issac, L. Zhang, and S. M. Jacob, "Detecting spam email with machine learning optimized with bio-inspired metaheuristic algorithms," *IEEE Access*, vol. 8, pp. 187914–187932, 2020.

[42] K. Chen. *Introduction to Natural Language Processing-TF-IDF*. Accessed: Nov. 12, 2022. [Online]. Available: https://kinder-chen.medium.com/introduction-to-natural-language-processing-tf-idf-1507e907c19

[43] S. Sawla. *Introduction to Naive Bayes for Classification*. Accessed: Nov. 12, 2022. [Online]. Available: https://medium.com/@srishtisawla/introduction-to-naive-bayes-for-classification-baefefb43a2d

[44] *Naive Bayes Classifiers*. Accessed: Nov. 5, 2022. [Online]. Available: https://www.geeksforgeeks.org/naive-bayes-classifiers

[45] G. Singh, B. Kumar, L. Gaur, and A. Tyagi, "Comparison between multinomial and Bernoulli Naïve Bayes for text classification," in *Proc. Int. Conf. Autom., Comput. Technol. Manage. (ICACTM)*, Apr. 2019, pp. 593–596.

[46] C. Manning, P. Raghavan, and H. Schütze, *Introduction to Information Retrieval*. New York, NY, USA: Cambridge Univ. Press, 2019.

[47] A. Saini. *Conceptual Understanding of Logistic Regression for Data Science Beginners*. Accessed: Nov. 12, 2022. [Online]. Available: https://www.analyticsvidhya.com/blog/2021/08/conceptual-understanding-of-logistic-regression-for-data-science-beginners/

[48] M. Mafarja, R. Jarrar, S. Ahmad, and A. A. Abusnaina, "Feature selection using binary particle swarm optimization with time varying inertia weight strategies," in *Proc. 2nd Int. Conf. Future Netw. Distrib. Syst.*, Jun. 2018, pp. 1–9.

[49] L. J. Miranda. *Feature Subset Selection*. Accessed: Nov. 4, 2022. [Online]. Available: https://pyswarms.readthedocs.io/en/development/examples/feature_subset_selection.html

[50] L. J. Miranda. *Pyswarms.Single Package*. Accessed: Nov. 4, 2022. [Online]. Available: https://pyswarms.readthedocs.io/en/latest/api/pyswarms.single.html

[51] S. Lee, S. Soak, S. Oh, W. Pedrycz, and M. Jeon, "Modified binary particle swarm optimization," *Prog. Natural Sci.*, vol. 18, no. 9, pp. 1161–1166, Sep. 2008.

[52] L. J. Miranda. *Pyswarms.Discrete Package*. Accessed: Nov. 5, 2022. [Online]. Available: https://pyswarms.readthedocs.io/en/latest/api/pyswarms.discrete.html

[53] H. Nezamabadi-Pour, M. Rostami-Shahrbabaki, and M. Maghfoori-Farsangi, "Binary particle swarm optimization: Challenges and new solutions," *J. Comput. Soc. Iran Comput. Sci. Eng.*, vol. 6, no. 1, pp. 21–32, 2008.

[54] S. S. Hong, W. Lee, and M. M. Han, "The feature selection method based on genetic algorithm for efficient of text clustering and text classification," *Int. J. Adv. Soft Comput. Appl.*, vol. 7, no. 1, pp. 1–19, 2015.

[55] G. Hollis, "Estimating the average need of semantic knowledge from distributional semantic models," *Memory Cogn.*, vol. 45, no. 8, pp. 1350–1370, Nov. 2017.

[56] L. A. Zadeh, "Fuzzy sets," *Inf. Control*, vol. 8, no. 3, pp. 338–353, Jun. 1965.

[57] L. Zadeh, "The concept of a linguistic variable and its application to approximate reasoning—II," *Inf. Sci.*, vol. 8, no. 4, pp. 301–357, 1975.

[58] I. Perfilieva, "Fuzzy IF-then rules from logical point of view," *Comput. Intell., Theory Appl.*, vol. 38, pp. 691–697. Sep. 2006.

[59] C. Hutto and E. Gilbert, "Vader: A parsimonious rule-based model for sentiment analysis of social media text," in *Proc. Int. AAAI Conf. Web Social Media*, 2014, vol. 8, no. 1, pp. 216–225.

[60] M. H. Azam, M. H. Hasan, S. Hassan, and S. J. Abdulkadir, "Fuzzy type-1 triangular membership function approximation using fuzzy C-means," in *Proc. Int. Conf. Comput. Intell. (ICCI)*, Bandar Seri Iskandar, Malaysia, Oct. 2020, pp. 115–120.

[61] *Fuzzy Logic—Inference System*. Accessed: Nov. 3, 2022. [Online]. Available: https://www.tutorialspoint.com/fuzzy_logic/fuzzy_logic_inference_system.htm

[62] J. Saade and H. Diab, "Defuzzification methods and new techniques for fuzzy controllers," *Iranian J. Electr. Comput. Eng.*, vol. 3, no. 3, pp. 161–174, 2004.

[63] D. Harbecke, Y. Chen, L. Hennig, and C. Alt, "Why only micro-F1? Class weighting of measures for relation classification," in *Proc. NLP Power 1st Workshop Efficient Benchmarking NLP*, Dublin, Ireland, 2022, pp. 32–41.

LIDA KETSBAIA received the B.Sc. degree in computer and digital forensics from Northumbria University. She is currently pursuing the Ph.D. degree in AI-based digital forensics. Her Ph.D. research work is centered around hate speech detection using machine learning, deep learning, and natural language processing (NLP).

BIJU ISSAC (Senior Member, IEEE) received the B.E. degree in electronics and communications engineering, the Master of Computer Applications (MCA) degree, and the Ph.D. degree in networking and mobile communications. He is currently an Associate Professor with Northumbria University, U.K. He has authored more than 100 refereed conference papers, journal articles, and book chapters. His research interests include networks, cybersecurity, applied machine learning, and deep learning.

XIAOMIN CHEN received the Ph.D. degree from the National University of Ireland. From December 2012 to August 2013, she was a Postdoctoral Research Fellow at the Hamilton Institute, National University of Ireland. She joined Northumbria University in 2016 and she is currently an Assistant Professor. Her research interests include wireless networking, machine learning, and security.

SEIBU MARY JACOB (Member, IEEE) received the B.Sc. and M.Sc. degrees in mathematics, the Post Graduate Diploma in Computer Applications (PGDCA) degree, and the bachelor's (B.Ed.) and Ph.D. degrees in mathematics education. She is currently a Senior Lecturer in engineering mathematics with Teesside University, U.K. She has authored more than 30 research publications as book chapters, journal articles, and conference papers.