IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 June 2022, accepted 4 July 2022, date of publication 14 July 2022, date of current version 21 July 2022.

Digital Object Identifier 10.1109/ACCESS.2022.3190960

RESEARCH ARTICLE

Offensive Language Detection in Arabic Social Networks Using Evolutionary-Based Classifiers Learned From Fine-Tuned Embeddings

FATIMA SHANNAQ[1], **BASSAM HAMMO**[1,2], **HOSSAM FARIS**[1], **AND PEDRO A. CASTILLO-VALDIVIESO**[3]

[1]King Abdullah II School of Information Technology, The University of Jordan, Amman 11942, Jordan
[2]King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman 11941, Jordan
[3]Department of Computer Architecture and Technology, ETSIIT-CITIC, University of Granada, 18011 Granada, Spain

Corresponding author: Hossam Faris (hossam.faris@ju.edu.jo)

This work was supported by the Ministerio Español de Ciencia e Innovación (DemocratAI::UGR) under Project PID2020-115570GB-C22.

ABSTRACT Social networks facilitate communication between people from all over the world. Unfortunately, the excessive use of social networks leads to the rise of antisocial behaviors such as the spread of online offensive language, cyberbullying (CB), and hate speech (HS). Therefore, abusive\offensive and hate detection become a crucial part of cyberharassment. Manual detection of cyberharassment is cumbersome, slow, and not even feasible in rapidly growing data. In this study, we addressed the challenges of automatic detection of the offensive tweets in the Arabic language. The main contribution of this study is to design and implement an intelligent prediction system encompassing a two-stage optimization approach to identify and classify the offensive from the non-offensive text. In the first stage, the proposed approach fine-tuned the pre-trained word embedding models by training them for several epochs on the training dataset. The embeddings of the vocabularies in the new dataset are trained and added to the old embeddings. While in the second stage, it employed a hybrid approach of two classifiers, namely XGBoost and SVM, and a genetic algorithm (GA) to mitigate the drawback of the classifiers in finding the optimal hyperparameter values to run the proposed approach. We tested the proposed approach on Arabic Cyberbullying Corpus (ArCybC), which contains tweets collected from four Twitter domains: gaming, sports, news, and celebrities. The ArCybC dataset has four categories: sexual, racial, intelligence, and appearance. The proposed approach produced superior results, in which the SVM algorithm with the Aravec SkipGram word embedding model achieved an accuracy rate of 88.2% and an F1-score rate of 87.8%.

INDEX TERMS Arabic harassment dataset, deep learning, evolutionary algorithm, fine-tuned word embedding, hate speech, offensive language, optimization.

I. INTRODUCTION

Information and communications technology (ICT) evolution increased the ease of accessibility and growth of global communication between online communities. However, false identities of online accounts and the presence of anonymity provide people the freedom to share their thoughts and post comments without any restrictions. Nowadays, aggressive behavior and hate speech spread widely and globally, raising many cybersecurity risks and vulnerabilities on Social Media Platforms (SMPs), such as Facebook, Twitter, and Internet forums. These incidents negatively impact the mental health and psychological status of victims and may lead to suicide in extreme cases. Moreover, online abusive behavior and hate may also drive to severe criminal activities [1], [2].

Profanity such as cursing or swearing has become very common in everyday informal and social media conversations. The use of abusive language can range from offensive, aggressive, hateful and violent. In [3], the authors differentiated between hate speech and offensive speech.

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

They define hate speech (HS) as a language to express hatred against a specific person or a group based on certain key characteristics such as religion, gender, race, sexual orientation, and various disability forms. Hate speech aims to humiliate or insult the target. In contrast, offensive speech has an eternal intent to hurt the recipient's feelings.

Moreover, [4] developed a typology of abusive language to present the similarities and differences between abusive language detection subtasks, such as hate speech, cyberbullying (CB), and trolling. Their topology relied on two main factors; (1) Directed\Generalized, and (2) Explicit\Implicit. The first factor indicates whether the language targets a specific individual or a generalized group, such as sexual orientation or ethnic groups. Whereas the second factor checks if the abusive context is explicit or implicit.

Government and social media networks should follow preventative methods to stop the misuse of social media platforms and prevent the prevalence of hate and Internet abuse. However, a vast amount of data is generated daily; hence manual human monitoring becomes impractical. Therefore, practical solutions and intelligent systems are needed to reduce these incidents. Machine Learning (ML) techniques and Natural Language Processing (NLP) applications applied on Twitter have efficient roles in building efficient systems to detect offensive and hate speeches.

Pre-trained word embedding gained popularity in the NLP community. It is a predictive approach for representing text in a numeric form understandable by ML and deep learning algorithms. It predicts a word from its neighbors by training a neural language model. In addition, it allows words with similar meanings to have an equal representation. By encoding word embedding in a densely populated space, we can represent words numerically in a way that captures them in vectors that have tens or hundreds of dimensions instead of millions as in one-hot encoding [5].

There are many static word embedding models available such as word2vec, from Google [5], Glove, from Stanford [6] and FastText, from Facebook [7]. However, there are many challenges when building these models. For instance, the short length of tweets and syntactic and grammar flaws make it difficult to extract representative features. Also, in some cases, static word embedding may lack some of the dataset vocabularies, and if the dataset is small, it is impractical to train the model from only the dataset.

The eXtreme Gradient Boosting (XGBoost) algorithm is a ML technique that received widespread interest in recent years because of its beneficial features such as saving time, optimizing memory resources, and executing in a parallel environment. However, XGBoost has a large number of hyperparameters which entails a massive amount of effort and resources to tune [8], [9].

Furthermore, the support vector machine (SVM) is a supervised ML model designed for analyzing the data and recognizing specific visible or hidden patterns. Its main idea is to map data points to a high dimension space with a kernel function, and then the data points can be separated by a hyperplane. However, the standard SVM method has some limitations, and the parameters setting of SVM is essential to enhance its performance [10].

Therefore, predicting offensive language in tweets is considered a single-objective ML-based classification problem, where the ultimate goal is to maximize performance through maximizing accuracy or minimizing the error rate.

However, searching for the optimal hyperparameter values for the ML models might require a vast amount of time, which entails making use of other methods, such as stochastic methods, including evolutionary algorithms, like the Genetic Algorithm [11].

Genetic Evolutionary Algorithm (GEA) is a nature-inspired algorithm and search heuristics based on the natural evolution theory. GEA uses a computer to mimic the process of natural selection, and reproduction [12]. It was first proposed by [13], and it has been successfully applied in the field of ML, especially for hyperparameter tuning and for the automatic design of deep neural network architectures. GEA helps to reduce the time and cost involved in the manual trial and error design methods and mitigates the drawback of XGBoost and SVM in automatically finding the optimal hyperparameter values to run the model with maximum prediction capacity.

The main contribution of this study is to design and implement an intelligent prediction system encompassing a two-stage optimization model to identify and classify the offensive from the non-offensive text in the Arabic language. In the first stage, the proposed approach fine-tuned the pre-trained word embedding models by training them for several epochs on the training dataset. The embeddings of the vocabularies in the new dataset are trained and added to the old embeddings. While in the second stage, it employed a hybrid approach of two classifiers, namely XGBoost and SVM, and a genetic algorithm (GA) to mitigate the drawback of the classifiers in finding the optimal hyperparameter values to run the proposed approach.

Many literature efforts have attempted to study and predict offensive content on various social media platforms. However, this is the first attempt to classify offensive text in Arabic tweets using a two-stage optimization approach to the best of our knowledge. Figure 1 depicts the pipeline of the proposed approach. It presents a flow-diagram for offensive language detection applied to the Arabic Cyberbullying Corpus (ArCybC), a dataset collected from Twitter using four domains: gaming, sports, news, and celebrities. The ArCybC dataset is organized into four categories based on the content type: sexual, racial, intelligence, and appearance. Several word embedding models have been used to extract the essential features to train the ML models to distinguish offensive from non-offensive tweets. The diversity of language and dialects and the short text length make detecting offensive context from tweets challenging.

The rest of the paper is organized as follows. Section 2 presents a background of the algorithms and techniques used in this work. Section 3 shows the related work. In Section 4, we discuss the proposed approach in detail. Section 5 presents the experimental setup and results. Section 6 discusses the results, and finally, Section 7 concludes the work.

II. BACKGROUND

A. WORD EMBEDDING

Word embedding is a predictive approach for representing text. It converts words into a numeric form understandable by ML or deep learning algorithms. Word embedding predicts a word from its neighbors by training a neural language model. It allows words with similar meanings to have an equal representation. Word embedding techniques are considered improvements to Bag-of-Word (BOW) and the statistical TF-IDF models that produce large and sparse vectors (most entries have 0 values), and they usually represent the documents but do not capture the semantics of the words [14].

Deep learning approaches have recently played a significant role in NLP applications. Most of the deep learning tasks in NLP applied methods using word vector representations [15]. Related words map to similar n-dimensional vectors in this representation, while dissimilar words have different vectors. The 'meaning' of a word can be reflected in its embedding space. Continuous vector representations of words, also referred to as word embedding, word2Vec, from Google [5], Glove, from Stanford [6] and FastText, from Facebook [7] are deep learning methods which can convert words into n-dimensional vectors.

1) Word2Vec

Word2Vec, developed by Google [14], is one of the popular techniques used for efficiently learning a standalone word embedding from a text corpus. Its main objective function is to direct words in a similar context to have similar embedding. Word2Vec has two neural network models for generating vectors from the words. The first approach uses the continuous bag of words (BoW) to predict the target word from the context. Whereas the second approach applies a skip-gram model to predict the context words (surrounding words) from the target word (center word) [14]. However, each model has its advantage [14]. Skip-Gram is more efficient with small training data. Moreover, in this model, infrequent words are well presented. On the other hand, CBOW is faster and works well with frequent words.

The Word2Vec methods have a distinct advantage in handling large datasets since they do not consume as much memory as some classic methods [16]. The accuracy of the Word2Vec models depends on the size of the text corpus. In other words, the accuracy increases with the growth of the text corpus [17].

2) AraVec

AraVec is a pre-trained distributed word embedding model, which has been adapted for the Arabic language and introduced by [18]. The model provides various word embedding models built on three different Arabic content domains, Twitter and Wikipedia Arabic articles. Recently, AraVec 3.0[1] consists of 16 different models divided into uni-gram and n-gram models.

3) GloVe

Global Vectors for Word Representation, or GloVe for short, is another word embedding algorithm produced by Stanford researchers [6]. It is a global log-linear regression model that combines the advantages of two models: global matrix factorization and local context window methods. It is similar to Word2Vec in incorporating context, but both techniques of Word2Vec (CBOW and skip-gram) build a term-document matrix that considers only local contexts. It does not take advantage of the global context. However, by contrast, GloVe embeddings leverage the same intuition behind the co-occurring matrix used distributional embeddings but use neural methods to decompose the co-occurrence matrix into more expressive and dense word vectors through building a term-term matrix. In this case, x(i,j) will be higher if the word i appears in the context of the word j often and vice versa.

We have adopted the GloVe version trained on Arabic corpus from various resources such as: Twitter, Wikipedia, and others.[2]

B. ML CLASSIFIERS

The following subsections briefly describe the ML algorithms used in this study.

- The DT algorithm breaks down a dataset into smaller subsets to build classification or regression models in the form of a tree structure. This structure consists of decision nodes and leaf nodes. A decision node has two or more branches, while a leaf node represents a classification or decision [19].
- The LR algorithm is a machine learning model that utilizes the logistic function to find the relationship between one dependent variable (the output variable) and one or more independent variables (input variables). LR assigns observations to a discrete set of classes based on probability. It specifies a threshold value to determine the discriminatory value at which the instance will belong to one class or another. Furthermore, the LR algorithm can be used for binary classification or for predicting the certainty of binary output [20], [21].
- KNN is a supervised instance-based classification algorithm used to solve classification and regression problems. Its main idea is that close instances assume to belong to the same class. Thus, to classify a new test point, the KNN algorithm identifies its K nearest neighbors and assigns a class to the test point based on the most frequent class of its specified neighbors [22].
- Naive Bayes (NB) is a statistical supervised machine learning classification method based on the Bayes' Theorem to determine the most likely class to which

[1]https://github.com/bakrianoo/aravec
[2]https://github.com/tarekeldeeb/GloVe

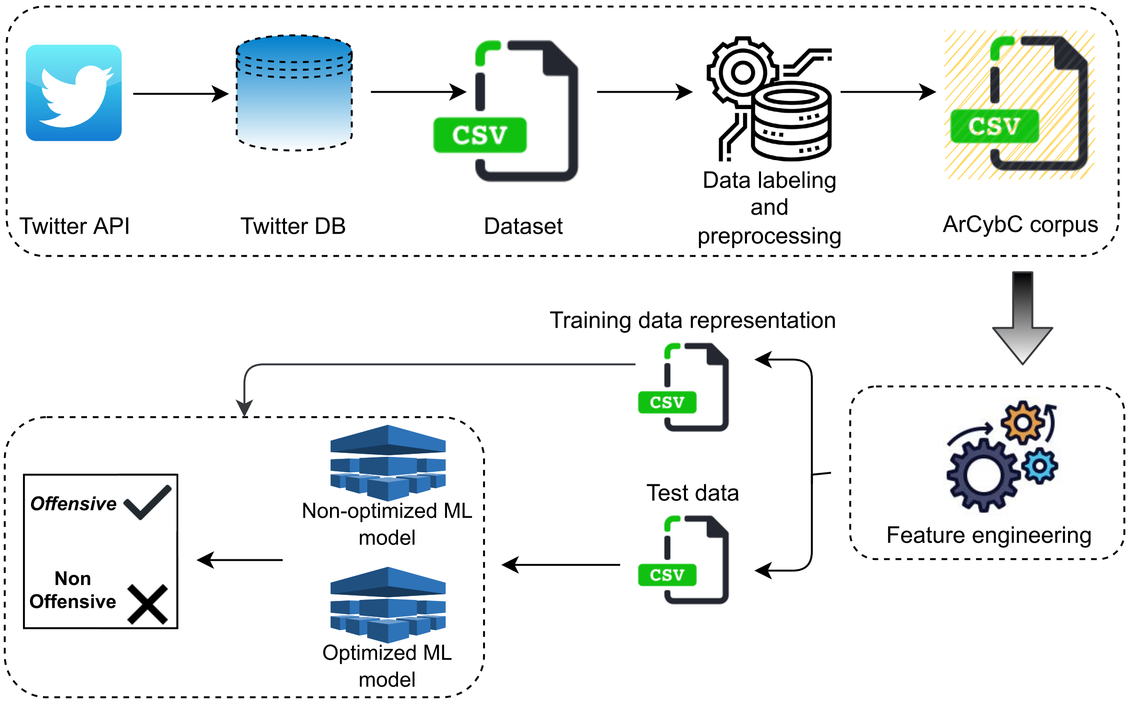


FIGURE 1. **The architecture of the offensive language detection approach.**

an instance belongs. It is a fast and straightforward algorithm because it does not depend on the use of rules or any other explicit representation of the classifier. NB assumes that a particular feature in a class is unrelated to the presence of any other feature. Therefore, it works well with independent features [23].

- The RF algorithm is a collection of decision trees, which rely on classification rules to classify the data. Every decision tree in the RF corresponds to a different subset of features selected from the training samples by replacement. The significant advantage of the novel structure of the RF algorithm is the avoidance of overfitting [24].
- The SVM algorithm is a supervised machine learning algorithm. SVM transforms the data into higher-dimensional spaces to better recognize the classes. It integrates a kernel function to map the dataset into the higher spaces [25].
- XGBoost is a scalable decision-tree-based ensemble ML algorithm that uses a gradient boosting framework. It provides satisfactory results in many ML competitions. In addition to being scalable, XGBoost has other advantages that distinguish it from the different classifiers. A few to mention are its ability to deal with missing values, handling sparse data, and utilizing the power of parallel and distributed processing, which makes it faster than many other algorithms as it uses multiple CPU cores to execute the model [9].

C. GENETIC EVOLUTIONARY ALGORITHM (GEA)

A genetic algorithm (GA) is a nature-inspired algorithm and search heuristics based on the natural evolution theory. GA uses a computer to mimic the process of natural selection and reproduction [12]. It was first proposed by [13] and has been successfully applied in the field of ML [8]. GA belongs to the family of evolutionary algorithms that can solve a specific problem repeatedly and produce a better solution.

GA starts with a randomly generated candidate set of solutions called population represented as a chromosome. Each individual in the chromosome is called a gene. Then the set is repeatedly refined using a fitness function that assesses the quality of each solution and keeps the best options for the next generation [26].

Three operators randomly affect the fitness value: selection, crossover, and mutation. In the selection phase, the fittest individuals or genes (two pairs) are selected to progress to the next generation level. The crossover operator, which identifies the most efficient operator in the GA, allows for determining a new region of the solution in the search space. It works by finding a crossover point or more within the randomly chosen genes for exchanging genes between chromosomes. Finally, in the mutation phase, a few genes may be subjected to an alteration to limit their premature convergence and maintain the diversity of the population. For example, in binary gene form, the genes may change from 1 to 0 and vice versa. Accordingly, replacing the old population through a diversity replacement or elitism strategy and generating a new population [8], [27]. Figure 2 summarizes these operators.

III. RELATED WORK

There are many attempts in the literature to detect abusive and offensive content in social media networks. Moreover, to boost the work in this area, various competitions were launched and equipped with large-scale datasets in different languages. For instance, the shared task of the fourth workshop on Open-Source Arabic Corpora and Corpora Processing Tools (OSACT) at the Language Resources and Evaluation Conference (LREC-2020) introduced two competitive tasks [28]. Subtask A aimed to detect the offensive language in Arabic speech. On the other hand, Subtask B aimed to detect hate speech. The organizers provided a Twitter dataset containing 10,000 manually labeled tweets about offensive and hate-speech language for testing. OffensEval 2020 is another example of shared tasks on offensive language identification that was organized at the International Workshop on Semantic Evaluation [29]. The competition provided a large-scale training dataset of around nine million English tweets and a multilingual dataset of four other languages: Arabic, Danish, Greek, and Turkish. The competition had three sub-tasks concerning the type and target of the offensive content; (1) Offensive language identification, (2) Automatic categorization of offense types, and (3) Offense target identification. The competition received more than 100 official submissions targeting the three tasks.

Researchers applied various methodologies to detect and classify offensive languages in these competitions. For instance, a few authors used ML-based methods, such as XGBoost [30], [31] and SVM [32]. Others applied a fine tuning of contextualized embedding, such as ULMFiT [33], AraBERT [34], [35], and multilingual BERT [36], [37]. However, [38], [39] employed distinct neural network models.

This section presents these attempts, whether in shared task competitions or other individual works. In OSACT4 Arabic offensive language detection, [37] achieved the highest score with an F1-score rate of 90.5%. The authors applied several preprocessing steps, such as removing non-Arabic words, diacritics, punctuation, and repeated characters. Then they used distinct ML and deep learning classifiers to detect offensive language. An ensemble classifier of SVM and character n-grams compound with pre-trained embedding (Mazajak[3]) and Deep Neural Network (DNN) achieved the best results.

The work of [40] focused on the detection and classification of the obscene and offensive Arabic tweets. For this purpose, they compiled two datasets: a Twitter dataset of 1,100 Arabic tweets and a dataset of 32K inappropriate comments collected from the Arabic news site Al Jazeera. They labeled the tweets as obscene, offensive, and clean. In addition, they extracted two types of features: (1) Log Odds Ratio (LOR) computed for each unigram and bigram word that occurs at least ten times, and (2) a list of obscene words called SeedWords (SW), used to tag each tweet as obscene if it included at least one of these words. Finally, they conducted several experiments using different combinations of these features. They achieved the best results when using SeedWords combined with LOR (unigram). The precision, recall, and F1-score rates were 97%, 44%, and 60%, respectively.

The work of [41] examined several ML classifiers using various feature extraction and selection techniques on a dataset of YouTube comments in Arabic to detect offensive language in online communications. They applied a variety of feature transformation techniques applied, including logistic regression with L1 regularisation (LR-L1), feature ranking with recursive feature elimination (RFE), ExtraTreesClassifier, tree-based ensemble methods, and singular-value decomposition (SVD). They trained five classical ML classifiers using the extracted features: SVM, Naive Base (NB), Decision Tree (DT), and Logistic Regression (LR). They achieved the best results from the SVM classifier with combined features selected by LR-L1 and RFE. The accuracy, precision, recall, and F1-score rates were 84%, 89%, 76%, and 81%, respectively.

Alsafari *et al.*, [42] developed a high-quality dataset for Arabic hate and offensive speech. The dataset has 5,340 tweets annotated with different types of hate. They evaluated various feature extraction schemes within a supervised classification framework, including ML and deep learning methods. In addition, they trained various classifiers on 2-class, 3-class, and 6-class datasets. The best model for the 2-class classification (abusive vs. normal) was the CNN using mBert features, where they obtained an F1-score rate of 87.03%.

Whereas the work of [43] designed a decision system to detect abusive text. The authors integrated unsupervised learning models based on word2Vec's skip-gram and the cosine similarity using special tools for filtering the offensive text. For instance, the authors used blacklists, n-grams, abbreviations, and punctuation to detect the deliberate obfuscation of abusive words. They used a corpus of news articles' comments from the economics and political sections and comments from the online community and Twitter to evaluate the system's performance. The integrated system achieved an F1-score rate of 86.93% in malicious word detection for news article comments, 85% for online community comments, and 92.09% for Twitter's tweets.

Deep Learning approaches have been successfully used for Arabic offensive language detection in online communication when [44] used the abusive Arabic tweet datasets described in [40] to train a character n-gram based deep learning classifier. In particular, they applied FastText, a deep learning-based text classifier, and compared their results with the baseline model used in [40] and the SVM classifier. They obtained superior results of recall and F1-score rates of 90%.

[3]http://mazajak.inf.ed.ac.uk:8000/

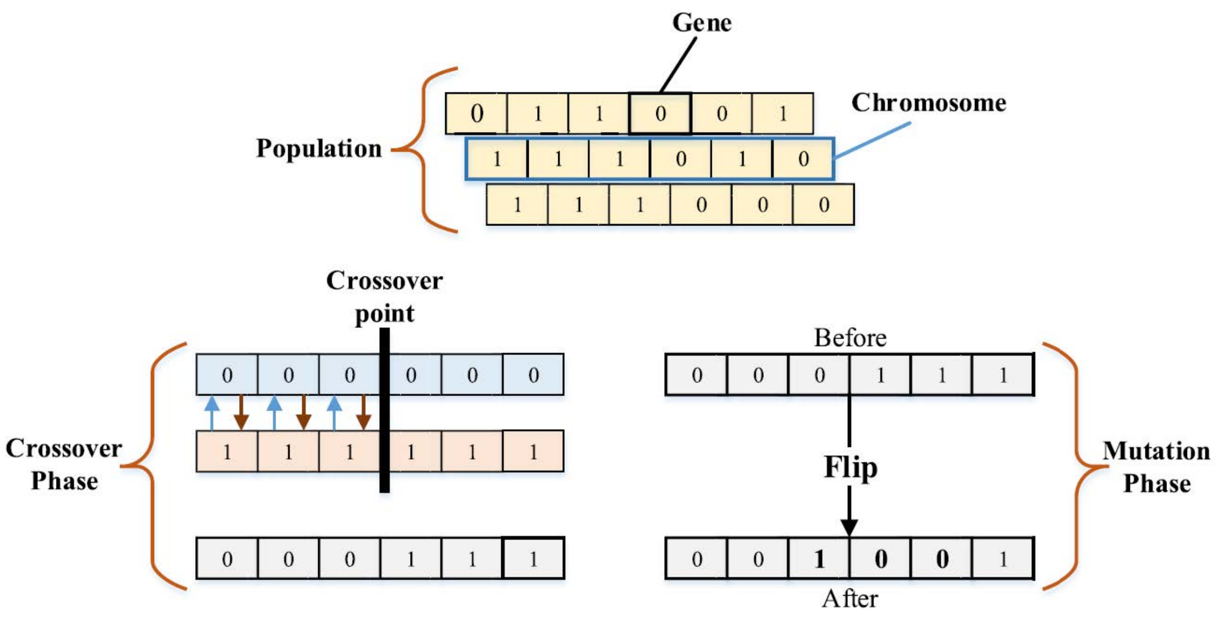


FIGURE 2. GA operators [8].

In [45] the authors applied FastText-based model to classify text from a Devanagari Hindi Offensive Tweets (DHOT) data corpus. The dataset was collected for three months, from September to November 2018, through Twitter API (Application Programming Interface). They used 20 abusive words in Hindi as seed terms to collect tweets. The dataset was composed of 24,596 filtered and cleaned tweets. They applied a grid-search method to tune hyperparameters during the model runs. Their model achieved an accuracy rate of 92.2%.

The authors in [2] used deep neural network models on a cyber-troll corpus. It is a dataset collected from Twitter by Data Turks for text classification intents and labeled as aggressive/ non-aggressive. For feature extraction, the authors used unigrams and bigrams encoded with TF–IDF. Then they selected the best representative features using SelectKBest function of Scikit-learn. Next, a Multi-Layer Perceptron (MLP) and a combination of CNN-LSTM and CNN-BiLSTM neural network models employed the extracted features to detect aggressive behaviors. The MLP model performed well with an accuracy rate of 92%.

The work of [46] produced the first abusive dataset in Turkish called Abusive Turkish Comments (ATC) to detect offensive comments from the Instagram platform. It is composed of 10,528 abusive and 19,826 not-abusive comments. They built ML classifiers to detect abusive messages, such as NB, SVM, and XGBoost. They also used Convolutional Neural Network (CNN) in their detection system. The CNN model achieved the best micro-averaged F1-score rate of 97.4%.

All the previous efforts applied a Single Task Learning (STL) to detect offensive language from social media platforms. SLT is generally carried out through learning task-specific features from one dataset at a time. However, [47] proposed a deep neural network model based on a Shared-Private Multi-Task Learning (SP-MTL), which aimed to take advantage of the correlation between related tasks, such as hate speech classification, offensive language detection, and racism and sexism identification to enhance classification by learning data in parallel. Their proposed model generates two groups of features for each task. The first group stores the shared features among related tasks by performing training jointly. While the second group stores only the task-dependent features. The SP-MTL model consists of various deep neural networks, like Convolution Neural Network (CNN), Long Short Term Memory (LSTM), and a combination of CNN and Gated Recurrent Unit (GRU). They tested their model on five benchmark datasets related to hate speech classification, offensive language detection, and racism identification. The framework achieved remarkable results compared with other state-of-the-art frameworks.

In recent years, the literature has brought several studies utilizing evolutionary algorithms to enhance the models proposed in different research areas, such as sentiment analysis [8], and feature selection [48], [49]. However, most of the studies applied GA to reduce the dimension of feature space rather than hyperparameter optimization.

Reducing the dimension of the features space becomes a vitally important step in the classification process since not all features are relevant for the classification task. In addition, the large number of features compared with the number of instances may lead to over-fitting [48]. Evolutionary algorithms, especially GA, are considered a perfect solution to explore the feature space. It can generate

TABLE 1. Summary of previous studies of offensive language detection methods.

Paper	Dataset	Dataset availability	Language	Methodology	Best classifier	Hyperparameter tuning	Text representation	Accuracy	F1-score
[2]	Twitter 20,001 tweets	Private	English	CNN-LSTM CNN-BiLSTM, MLP	MLP	_	n-gram encode with tf-idf	92.0	90.0
[35]	Twitter 10,000 tweets	Public	Arabic	LR, CNN, m-BERT, AraBERT	AraBERT	_	tf-idf, AraVec	93.0	88.0
[40]	Twitter 1,100 tweets	Public	Arabic	List-based classifier	LOR (unigrams)	_	unigrams and bigrams	_	60.0
[41]	YouTube 15,050 comments	Public	Arabic	SVM, NB, DT, LR, RF	SVM	_	n-gram	84.0	81.0
[43]	Twitter 1000 tweets	Private	English	Unsupervised learning	_	_	n-gram	_	92.1
[44]	Twitter 1, 100 tweets	Public	Arabic	SVM, FastText	FastText	_	Character n-grams	_	90.0
[45]	Twitter 24,596 tweets	Public	Hindi	FastText	FastText	Grid search	Word2Vec	92.2	_
[46]	Instagram 30,354 comments	Public	Turkish	CNN, NB, SVM, DT, RF, LR, AdaBoost, XGBoost	CNN	_	BoW, bi-gram, and tri-gram	_	97.4
[42]	Twitter 5,340 tweets	Public	Arabic	NB, SVM, LR, CNN, LSTM, GRU	mBert with CNN	Grid search	n-gram, AraVec, mBert, FastText	_	87.0
Our model	**Twitter 4,505 tweets**	**Public**	**Arabic**	**GA-SVM, GA-XGBoost**	**GA-SVM**	GA	**Fine-tuned AraVec, Fine-tuned GloVe**	**88.2**	**87.8**

numerous features subsets during reproduction operations to get the best subset that comprises the most relevant features [48]. For instance, [48] proposed an approach for Arabic opinions analysis, which combines SVM with a random subspace (RSS) algorithm, and applies GA to enhance the system. RSS is used to automatically generate different subsets of features vectors with limited size and replace the decision tree base classifier of RSS with SVM. The GA was applied to enhance the proposed methodology by avoiding the random choice adopted by RSS by generating features based on correlation criteria to avoid choosing incoherent features subsets. They trained the sentiment classifier on a corpus consisting of 1,000,000 Arabic reviews collected from online websites of Arabic Algerian newspapers. The enhancement made through using GA increased the accuracy rate of the proposed sentiment analysis system from 75.90% to 85.99%.

The binary firefly optimization (BFFO) is another nature-inspired optimization algorithm adopted by [49] for feature optimization in a multi-modal cyber-aggression detection system. The authors presented a combined model of deep learning and a BFFO algorithm to classify the social media posts into three classes; (1) high-aggressive, (2) medium-aggressive, and (3) non-aggressive. The corpus used to train the model consists of a combined feature of images and text collected from various social media sites like Twitter, Instagram, and Facebook. Besides searching Google engine using several query terms like bullying images and cyber-aggressive images. A pre-trained VGG-16 model extracted the image features, while a three-layered CNN extracted textual features. The obtained results showed an enhancement of 11% in terms of F1-score when using the optimized features of BFFO.

A high level of variance may occur when fine-tuning contextual word embedding models like BERT on downstream classification tasks due to various factors, such as the BERT architectural design (BERT layer and fine-tuning architectures) and the hyperparameter settings. Such variance produces different results for the same task [50]. Therefore, [50] used GA to automatically search for the near-optimal BERT architectural design and the best hyperparameter settings for the hate speech detection task. They tested their method on two publicly available hate speech datasets. The results showed the genetic algorithm's suitability to probe BERT layers for a downstream classification task. However, the main hindrance of their work was the selection of the hyperparameter choices to be part of the search process, in which introducing new parameters or removing existing ones may result in minor changes.

Husain and Uzuner [51] surveyed the various forms of harassment and offensive language detection in Arabic. They investigated the state-of-the-art methodologies adopted in literature to develop datasets and build detection systems in this field. They reported a lack of innovative approaches for feature extraction and model development that best work for the Arabic language, especially in adopting evolutionary algorithms to tune algorithms' hyperparameters. In addition, they concluded that previous approaches that used static word embeddings, such as Word2Vec and FastText, did not apply fine-tuning processes to feature extraction schemes. Therefore, it was mandatory to conduct a study that aims to improve the methodologies adopted for developing prediction systems at both levels; feature extraction and ML model development.

Table 1 summarizes these studies in terms of the dataset used, language, the social media platform, the adopted methodology and text representation approach, the classier that achieved the best results, the hyperparameter tuning approach (if applied), and the obtained results. The table shows that our approach is the first study that fine-tuned non-contextualized word embedding and applied hyperparameter tuning using GA.

IV. THE METHODOLOGY

This study focused on building an optimized offensive language detection model to classify Arabic tweets as offensive (Off) and non-offensive (Non-Off). Figure 3 provides an illustration of the methodology we followed in this work. It is made of two stages of optimization. In the first stage, the training dataset was used to fine-tune various word embedding models to extract the word features of the ArCybC corpus that will be fed to the second optimization stage. In the second stage, we used GA-based algorithms, which are a hybrid approach of GA and (SVM or XGBoost). The following subsections describe each module in more detail.

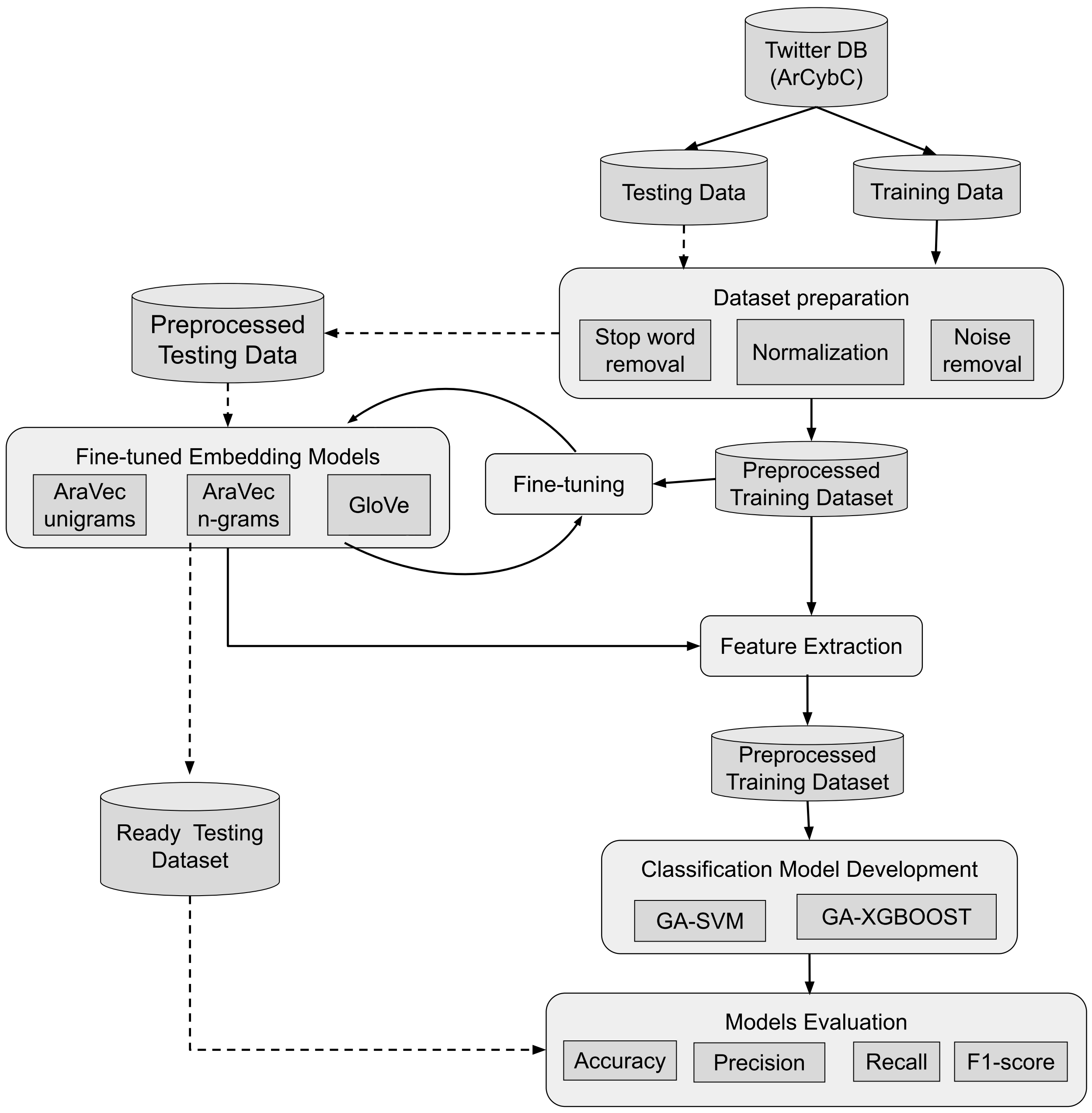


FIGURE 3. The research methodology.

A. DATASET PREPARATION

Arabic Cyberbullyig Corpus (ArCybC) [52] is the first publicly available cyberbullying dataset for the Arabic language.[4] Researchers can use it to classify tweets annotated as Cyberbullying (CB), Non-Cyberbullying (Non-CB), Offensive (Off), and Non-Offensive (Non-Off). The main objective of ArCybC was to build a benchmark repository for the Arabic language for researchers to study various aspects of online harassment, such as offensive language and CB. We used the Twitter streaming API to collect data from different Twitter accounts, Twitter is considered a popular micro-blogging platform that is a powerful communication tool as well as a broadcast tool, facilitating the expansion of social reach. Twitter REST and APIs enable data analysts, researchers, and developers to easily access public data streams and analyze tweets [53]. We focused on four distinct domains; gaming, sports, news, and celebrities. The rationale behind this decision was based on a thorough analysis of the most frequent accounts attracting people from different slices of the Arab community. Next, we extracted the top 50 hashtags of each domain, and we utilized the list of

[4]https://data.mendeley.com/datasets/z2dfgrzx47/draft?a=12a9ff5d-6c5c-4b2e-8990-7d044d7c12e2

TABLE 2. Final distribution of tweets per each domain.

Domain Name	Number of Tweets
Celebrities	615
Gaming	312
News	3128
Sports	450
Total	4,505

TABLE 3. Final distribution of tweets per each category.

Category Name	Number of Tweets
Appearance	905
Intellectual	769
Racial	2349
Sexual	482
Total	4,505

hashtags as seed terms for collecting tweets from Twitter between September 17, 2017, to September 17, 2020. Next, a lexicon of offensive words, which usually appear in humiliating comments, was used to filter the data collection. The lexicon contained 320 words classified into sexual, racial, physical appearance, and intellectual categories. For text normalization, we removed retweets to avoid repeated evaluation of posts. Then, we removed the uniform resource locator (URL) address. To preserve privacy and not disclose the names of users, we replaced the actual user name with @USERNAME. We ended up with 4,505 tweets that were potentially subject to hold harassment content. We hired a judgment group of five Arabic native speakers to label the tweets manually. The annotators were between 21-and 35 years old. Four annotators were undergraduate students majoring in information technology, and one was a graduate student doing a master's degree in business management. The annotators were instructed to do two labeling tasks. The first task was to label a tweet as CB or non-CB, while the second task was to label a tweet as Offensive or non-Offensive. The annotators followed restricted rules to accomplish their two tasks. The average agreement among the annotators was 64%.

Next, we divided the ArCybC corpus into 80% training and 20% testing datasets. Out of the 3604 tweets in the training dataset, 1519 tweets (42.1%) were labeled as offensive, whereas 1391 (38.6%) were labeled as cyberbullying. While for the testing dataset, out of the 901 tweets, 368 tweets (40.8%) were labeled as offensive, and 337 (37.4%) were labeled as cyberbullying. Table 2 shows the final distribution of the tweets per each domain, while Table 3 shows the distribution of the tweets per each category.

Social media posts usually contain a lot of noisy text that hamper the success of the classification task. We follow the following steps to remove noise and prepare the data to pass for machine learning experiments:

TABLE 4. Pre-trained word embedding model details.

Word embedding model	Dim size	Vocab size	Data Sources
AraVec Skip Unigrams	300	1259756	Twitter
AraVec Skip N-grams	300	1476715	Twitter
AraVec CBOW Unigrams	300	1259756	Twitter
AraVec CBOW N-grams	300	1476715	Twitter
GloVe	256	1538616	Multi-sources

- Remove digits, punctuation marks, stop words, non-Arabic words, repeated characters, and diacritics. Diacritics are short vowels and characters above and beneath letters, such as fatha, damma, kasra, etc.
- The normalization of Arabic characters such as changing the letter ي to ى, ة to ه, and آ ،إ ،ؤ ،ئ to ا

B. FEATURE EXTRACTION

We extracted textual features directly from tweets and presented text into proper numeric vectors. Figure 4 shows the most n-gram features for each domain.

In this study, we adopted various non-contextualized pre-trained word embedding models. Table 4 illustrates these models in terms of dimension size, vocab size, and data resources.

In the first optimization stage, we fine-tuned the word embedding models using the ArCybC training dataset. In particular, if the model could not recognize the words in the dataset, it would discard them. Therefore, to fine-tune the word embedding models, we called the build_vocab() method with the parameter (update = True) to extend the known vocabulary by creating embedding vectors for the new vocabs in the ArCybC dataset and added them to the pre-trained model. Next, we trained the model for several epochs on the ArCybC training dataset.

Initially, the AraVec model passes the context words, which are one-hot encoded vectors of size $(1\ x\ v)$, as input to an embedding layer of size $(v\ x\ N)$, initialized with random weights. Next, the input words are multiplied by the $(v$ x $N)$ weighted matrix to produce the embedding of each word, which is then averaged and passed to the hidden layer, as depicted in Figure 5. The v represents the number of all words, and N is the embedding size (in this study, N=300). In this case, the order of words is not considered; hence it is called bag-of-words. Following this step, the output of the hidden layer is multiplied by the weighted matrix $(N\ x\ v)$, which encloses the contextual semantics, to deliver the final output vector $(1\ x\ v)$. In contrast, the Skip-Gram predicts the context words from a given target (center) word in the same approach as CBOW, where it predicts one context word at a time. In both CBOW and Skip-Gram, the predicted output is compared with the actual target. The loss function (e.g., Softmax) is computed followed by backpropagation with each epoch to update the embedding layer in the process [54].

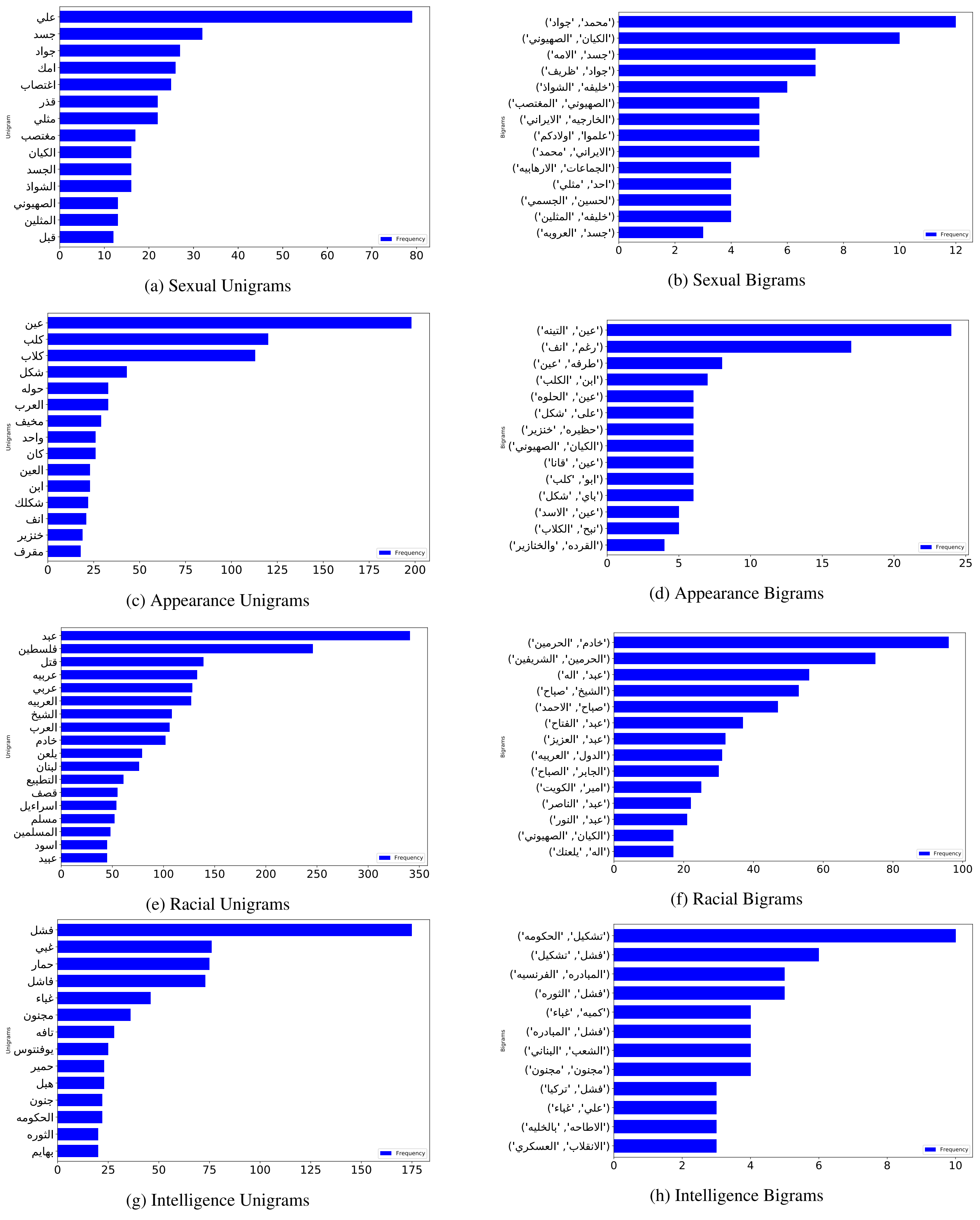


FIGURE 4. N-gram features.

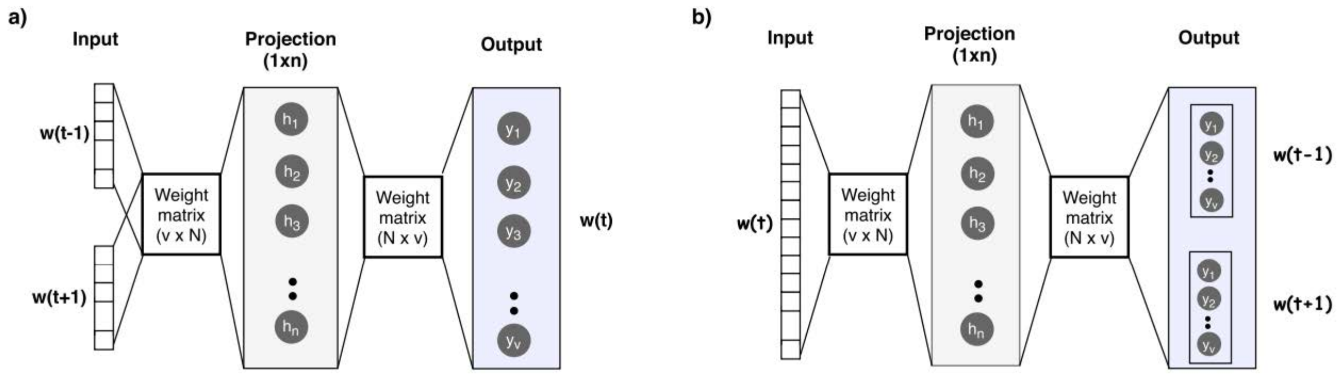


FIGURE 5. AraVec pretrained model, subfigure a) CBOW, and subfigure b) Skip-Gram.

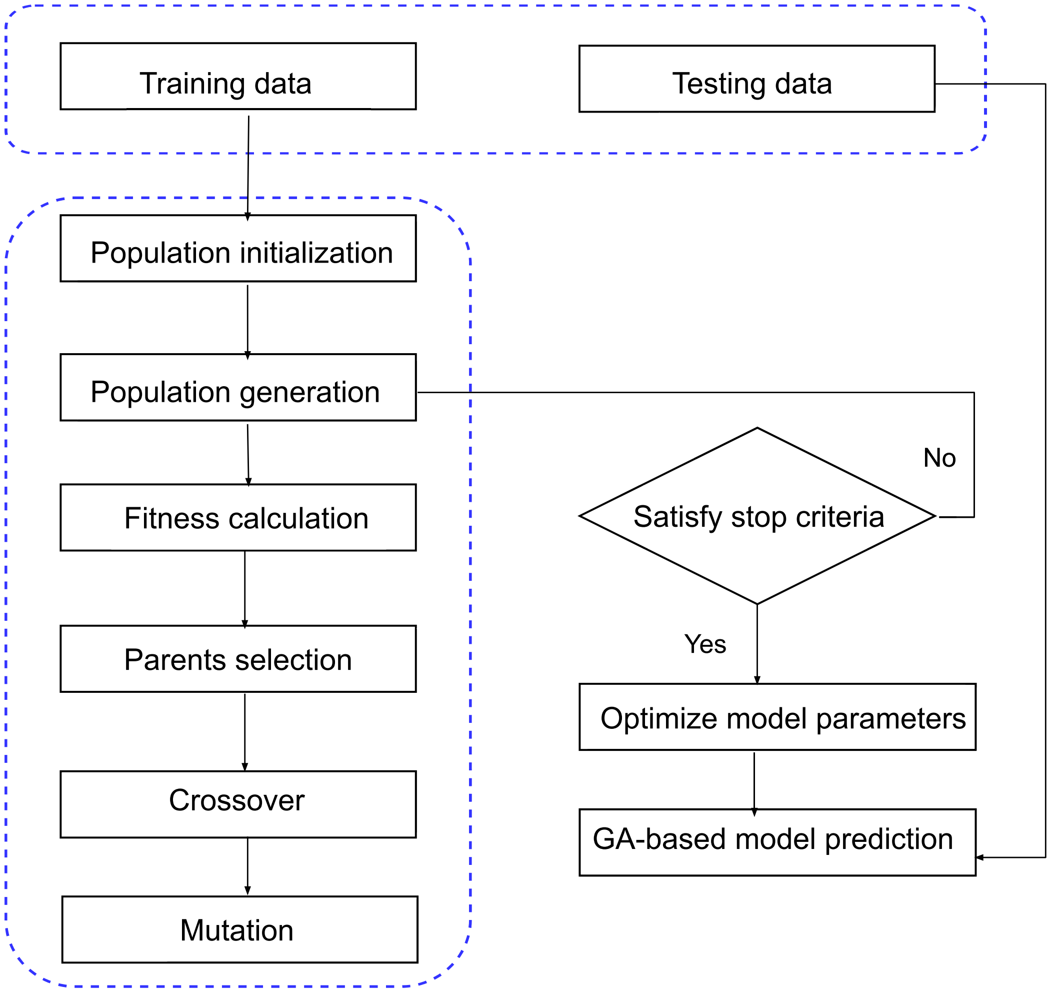


FIGURE 6. Flowchart of the GA-based parameter optimization model.

C. CLASSIFICATION MODEL DEVELOPMENT

The proposed approach uses the XGBoost and the SVM algorithms. In the first optimization phase, we tested the proposed approach by fine-tuning the pre-trained word embedding models using state-of-the-art ML classifiers, namely RF, LR, NB, and KNN, to demonstrate its effectiveness. While in the second optimization stage, we use the GA to optimize the XGBoost and the SVM hyperparameters. Figure 6 illustrates the process of the GA-based parameter optimization model training and prediction.

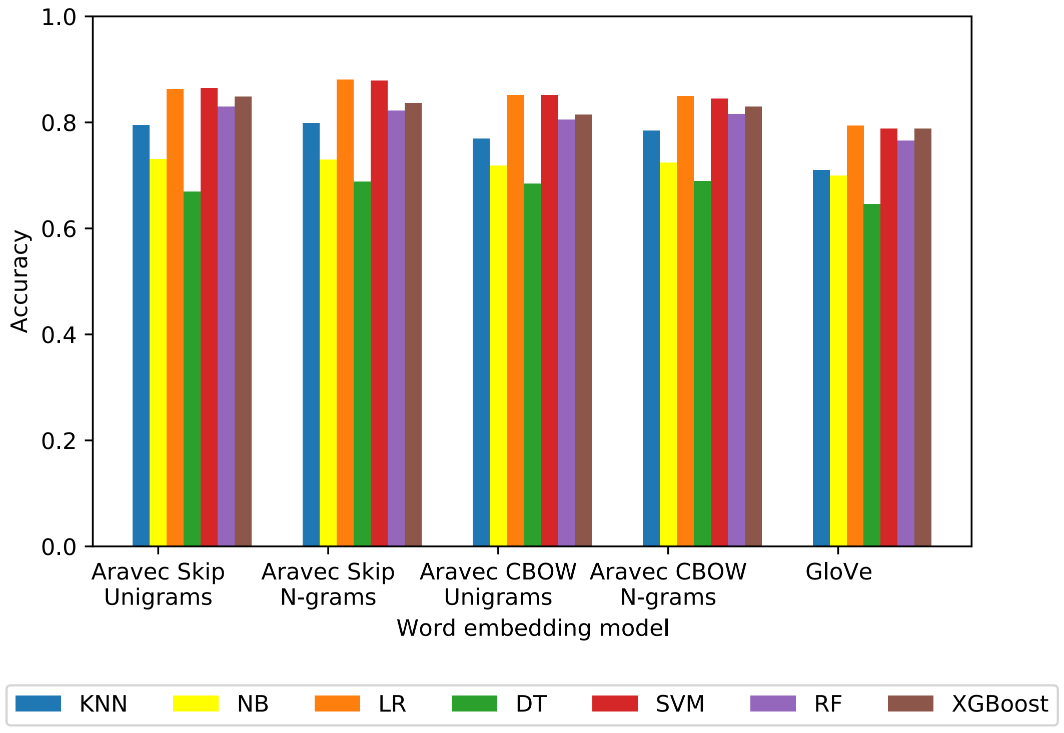


FIGURE 7. Comparison of classification models before optimization.

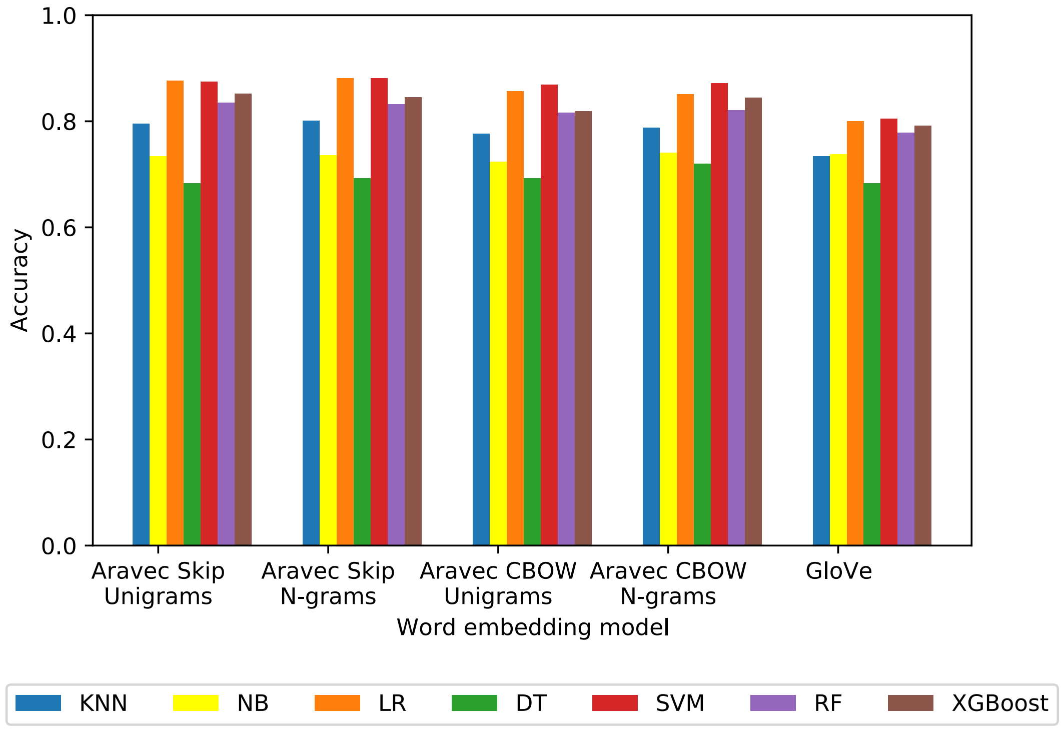


FIGURE 8. Comparison of models after fine-tuning pre-trained word embedding.

The following describes the two design aspects to consider when using the optimizer's algorithm. They include solution representation and fitness function.

1) ***Solution representation:*** GA as an optimization approach that is applied in this work to find the best parameters for the XGBoost and SVM models.

TABLE 5. List of XGBoost parameters, their descriptions, ranges, and default values.

Parameters	Description	Range	Default value
learning_rate	Step size	[0-1]	0.3
n_estimators	The number of trees used in the model	[10-1500]	100
max_depth	Maximum number of tree depth	[1-30]	6
min_child_weight	Minimum sum of instance weight needed in a child	[0.01-10.0]	1
gamma	Minimum loss reduction needed to make the partition.	[0.01,10.0]	0
subsample	Represent the ration of the training instances	[0.01, 1.0]	1
colsample_bytree	The subsample ratio of columns when constructing each tree	[0.01-1.0]	1
colsample_bylevel	The subsample ratio of columns for each level	[0.01-1.0]	1
colsample_bynode	The subsample ratio of columns for each node (split)	[0.01-1.0]	1
reg_lambda	L2 regularization term on weights	[0-10]	1
reg_alpha	L1 regularization term on weights	[10-100]	0

TABLE 6. List of SVM parameters, their descriptions, ranges, and default values.

Parameters	Description	Range	Default value
C	Penalty parameter of the error term	[0.001, 0.01, 0.1, 1, 10, 100]	1.0
gamma	Controls the distance of influence of a single training point	[0.001, 0.01, 0.1, 1, 10, 100]	scale

TABLE 7. Confusion matrix.

	Predicted (1)	**Predicted (0)**
Actual (1)	TP	FN
Actual (0)	FP	TN

In each iteration of the GA, the solution, represented as a chromosome, comprises a one-dimensional vector containing the parameters' values generated randomly. Each parameter's value has an upper and lower limits, and the randomly generated number should be rescaled to the desired limits by multiplying it by the size of the new range and adding the min value, as shown in the following equation:

$$A' = min + (A * (max - min)) \tag{1}$$

where A' is the new scaled value, and A represents the randomly generated value that needs to be scaled. *min* and *max* are the lower and upper bounds of the desired range, respectively.

One of the powerful feature of XGBoost is the degree of customization available due to its long list of hyperparameters that we can tune to prevent overfitting. Hence, to optimize XGBoost, we have selected 11 hyper-parameters. Accordingly, the chromosome consists of 11 genes. Table 5 shows a detailed description of the parameters, their ranges, and default values.

On the other side, to optimize SVM, we have selected two hyper-parameters, and therefore, the chromosome consists of 2 genes. Table 6 shows a detailed description of the parameters, their ranges, and default values.

2) ***Fitness function:*** In this step, we train the proposed approach and assess the solution generated from the GA by calculating the fitness value. We chose the Accuracy rate measure to find the fittest solution. Consequently, the individual chromosome) with a high Accuracy rate produces a high fitness value.

A summarization of the proposed algorithm is given by Algorithm 1.

D. MODELS EVALUATION

To evaluate the performance of the proposed approach, we utilized four evaluation metrics; accuracy, macro-precision, macro-recall, and macro-F1-score. Table 7 depicted the metrics derived from the "Confusion Matrix," which is a specific table layout that allows visualization of the performance of ML algorithms. The following describes the terminologies of the confusion matrix concerning the context of our study (i.e., predicting offensive language).

- True positive (TP), which represents the number of instances correctly labelled as offensive.
- True negative (TN), which represents the negative instances correctly classified as non-offensive.
- False positive (FP), which represents the number of instances incorrectly labelled as belonging to offensive class.
- False negative (FN), which represents the number of cases in offensive class that were wrongly predicted.

Algorithm 1 The Proposed GA-Based Model

```
Input:
    x_train, y_train, Train set
    x_val, y_val,// Validation set
    numberOfParents,// Number of solutions
    numberOfParentsMating,// Number of children
    numberOfParameters,// XGBoost_param=11,
    // SVM_param=2
    numberOfGenerations
Output:
    Best solution
Begin
    populationSize=(numberOfParents,numberOfParameters)
    population=initilialize_poplulation(numberOfParents)
    while $i < numberOfGenerations$ do
    // Calculate the fitness of each
    solution in the population
    fitness=train_population(population, x_train, y_train,
    x_val, y_val)
    // Select good parents for reproduction
    and perform the GA operators
    parents=new_parents_selection(population)
    // Apply uniform crossover
    children=crossover(parents)
    // Choose a random parameter and change
    a single gene in the offspring randomly
    children_mutated=mutation(children,
    numberOfParameters)
    End while
    Select and decode the individual with maximum fitness
    Return the best solution with optimized parameters
    Run the classifier with the optimized parameters on the
testing dataset
End
```

Based on those terminologies, the evaluation metrics can be derived as follows.

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{2}$$

$$Precision = \frac{TP}{FP + TP} \tag{3}$$

$$Recall = \frac{TP}{FN + TP} \tag{4}$$

$$F1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{5}$$

Then, we have calculated the macro-averages of F1-score, precision, and recall. A macro-average will compute the metric independently for each class and then take the average hence treating all classes equally. The macro-averages can be calculated as follows:

- Macro-F1 = average of within-category F1 values
- Macro-Recall = average of within-category Recall values
- Macro-Precision =average of within-category Precision values

TABLE 8. Phase I: Performance evaluation of base classifiers learned from static pre-trained embedding models.

	Accuracy	Precision	Recall	F1-score
Aravec Skip Unigrams				
KNN	0.795	0.795	0.775	0.781
NB	0.731	0.750	0.691	0.695
LR	0.863	0.858	0.859	0.859
DT	0.669	0.660	0.662	0.660
SVM	**0.865**	**0.859**	**0.864**	**0.861**
RF	0.830	0.831	0.815	0.820
XGBoost	0.849	0.845	0.842	0.843
Aravec Skip N-grams				
KNN	0.799	0.801	0.778	0.785
NB	0.730	0.763	0.685	0.686
LR	**0.881**	**0.877**	**0.876**	**0.877**
DT	0.688	0.678	0.679	0.678
SVM	0.879	0.874	0.877	0.875
RF	0.822	0.819	0.811	0.814
XGBoost	0.836	0.849	0.813	0.822
Aravec CBOW Unigrams				
KNN	0.769	0.767	0.748	0.753
NB	0.718	0.711	0.716	0.712
LR	**0.851**	**0.846**	**0.848**	**0.847**
DT	0.693	0.682	0.683	0.683
SVM	**0.851**	0.845	**0.848**	**0.847**
RF	0.805	0.798	0.796	0.797
XGBoost	0.815	0.808	0.808	0.808
Aravec CBOW N-grams				
KNN	0.784	0.777	0.772	0.774
NB	0.724	0.715	0.704	0.707
LR	**0.850**	**0.844**	**0.846**	**0.845**
DT	0.689	0.678	0.676	0.677
SVM	0.845	0.839	0.841	0.840
RF	0.816	0.810	0.807	0.809
XGBoost	0.830	0.824	0.825	0.825
GloVe				
KNN	0.710	0.703	0.684	0.688
NB	0.700	0.69	0.678	0.681
LR	**0.794**	**0.787**	**0.783**	**0.785**
DT	0.646	0.632	0.631	0.631
SVM	0.788	0.781	0.778	0.779
RF	0.766	0.762	0.747	0.752
XGBoost	0.788	0.782	0.777	0.779

V. EXPERIMENT AND RESULTS

To experiment with the proposed approach, we conducted three practical experiments using the ArCybC dataset to classify tweets into Offensive and Non-offensive. In the first experiment, we used seven ML algorithms, namely NB, KNN, SVM, RF, XGBoost, DT, and LR, to develop models that can predict offensive context using the text representation

TABLE 9. Changes in the word embedding models size.

Word embedding model	Vocab size (original)	Vocab size (Fine-tuned)	Difference
AraVec Skip Unigrams	1259756	1261247	1491
AraVec Skip N-grams	1476715	1478073	1358
AraVec CBOW Unigrams	1259756	1261190	1434
AraVec CBOW N-grams	1476715	1478158	1443
GloVe	1538616	1540525	1909

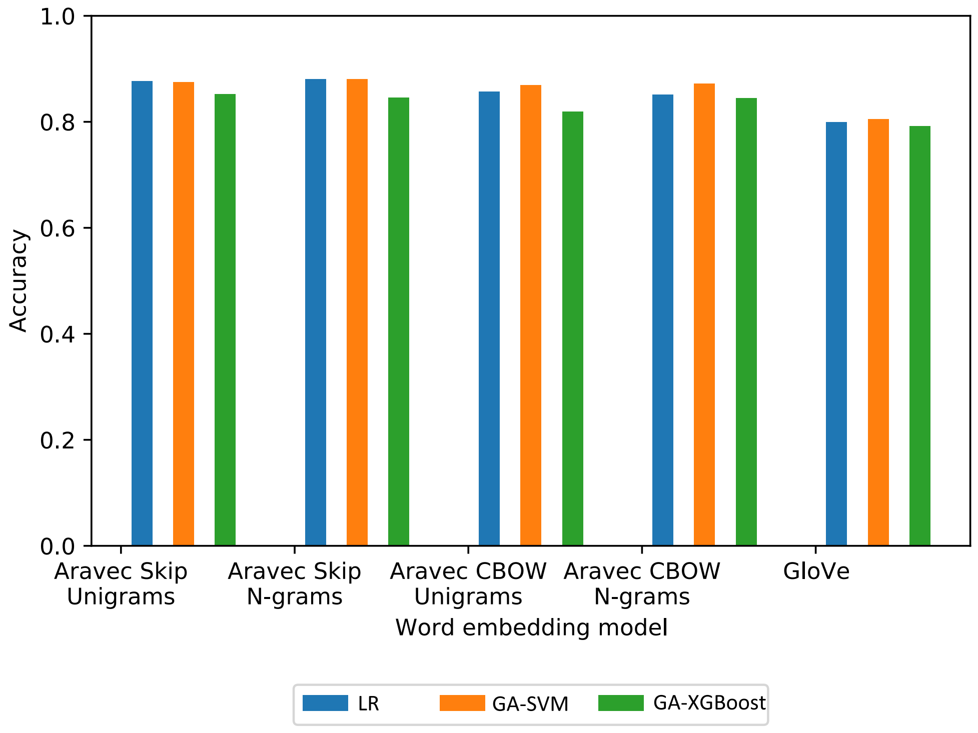


FIGURE 9. Comparison of the proposed approach with the baseline classifiers.

methods discussed in the previous subsection. In this phase, we did not apply any fine-tuning or optimization. In the second experiment, we repeated the first one after fine-tuning the pre-trained word embedding models by training them for several epochs on the training ArCybC dataset. While in the third experiment, we optimized the hyperparameters of the models that achieved the best results from the second experiment using GA. The following is a detailed description and discussion of the three experiments.

A. PHASE I: BASE CLASSIFIERS LEARNED FROM STATIC PRE-TRAINED WORD EMBEDDING MODELS

Table 8 shows the results of the base models without fine-tuning. As noticed from Figure 7, LR achieved the best results in almost all models; hence it will be considered as the baseline classifier. On the other hand, the SVM achieved the highest accuracy using the AraVec Skip-Unigrams model with an accuracy rate of 86.5%.

Typically, XGBoost and SVM algorithms give excellent results in many ML problems. However, we trained the models with default parameters; therefore, we did not use the best hyperparameters for each model, which is why they showed unexpected results.

B. PHASE II: BASE CLASSIFIERS LEARNED FROM FINE-TUNED PRE-TRAINED WORD EMBEDDING MODELS

Because we relatively have a small dataset and intended to apply static word embedding models, we encountered a few problems when the dataset's vocabularies did not show in the pre-trained model. To overcome this problem and incorporate the new vocabularies, we used the pre-trained models and fine-tuned them with the new data from the training dataset.

Consequently, we repeated the pipeline applied to the first stage in terms of text representation and classification methods. Next, we fine-tuned the word embedding models by training them with several epochs from the ArCybC training dataset.

For the AraVec skip-gram and CBOW training models, the dimension *size* was set to 300, and the size of the context window was set to ten. The *window* parameter determines the maximum distance between the current token and the predicted one. Also, the maximum number of *epochs* was five, and the *min_count* was one. While for Glove, the dimension *size* was set to 256, and the number of *epochs* was 30.

Table 9 shows the changes in the vocabulary size of the fine-tuned word embedding models. As one can notice, the vocabulary file size of the Glove model increased more than the other models (i.e., about 1909 words). However, in all AraVec models, the vocabulary files size increased by approximately the same amount.

Table 10 shows the performance evaluation of the base classifiers learned from fine-tuned pre-trained embedding models. Figure 10 compares the classifiers in terms of accuracy. We observed that there was an improvement in the performance of all models. For instance, the accuracy of SVM increased from 86.5% to 87.5%, and the XGBoost model increased from 84.9% to 85.2%. Moreover, the SVM classifier achieved superiority in four out of five models: AraVec Skip N-grams, AraVec CBOW Unigrams, N-grams, and Glove. In contrast, LR achieved its best results in the AraVec Skip Unigram model.

However, the Glove model achieved the lowest results for all its models because it was trained on Arabic corpus from various resources, while the other models were trained on a Twitter dataset compatible with the source of ArCybC dataset.

The results showed a slight improvement in the performance and this is because the size of the training dataset that is used to retrain the word embedding models is relatively small, where the training dataset contains 3604 instances and the added vocabularies to the models are less than 2000 vocabs.

C. PHASE III: GA-BASED MODELS LEARNED FROM FINE-TUNED PRE-TRAINED WORD EMBEDDING MODELS

XGBoost and SVM started their training using the initial parameters described in Tables 5 and 6 respectively. After they completed their training, they returned the fitness value to GA. All steps were repeated for the same training set until the maximum number of generations was reached. As long as the number of generations gets to the maximum value, the best chromosome generated by GA can be used for testing. We repeated the previous procedure ten times by taking the average of all values to get more realistic results.

Using the results obtained from running the GA-XGBoost and GA-SVM models on the training dataset, we examined their effects on the test dataset. We reported the accuracy, macro-average precision, recall, and F1-score. Table 11 compares the results of our proposed approach and the baseline models. The results showed that SVM achieved the best results across all models. While XGBoost exceeded the base classifiers in two models, AraVec CBOW N-grams and GloVe.

Figure 9 compares the accuracy of the proposed approach and the baseline classifiers. Figures 10 and 11 show the improved performance of XGBoost and SVM respectively in terms of accuracy after applying all steps in the proposed approach pipeline. The obtained results emphasized the effectiveness of the proposed approach for both models.

During the training of the proposed GA-based ML model, we evaluated the model's performance by computing the accuracy at each step. The learning curve is a widely used analytical tool in ML for monitoring the performance of algorithms that learn from a training dataset incrementally [55]. Figures 12 and 13 show how well the model

TABLE 10. Phase II: Performance evaluation of base classifiers learned from fine-tuned pre-trained embedding models.

	Accuracy	Precision	Recall	F1-score
	Aravec Skip Unigrams			
KNN	0.796	0.795	0.777	0.783
NB	0.734	0.773	0.687	0.688
LR	**0.877**	**0.872**	**0.875**	**0.873**
DT	0.683	0.675	0.679	0.676
SVM	0.875	0.869	0.873	0.871
RF	0.835	0.831	0.825	0.828
XGBoost	0.852	0.848	0.847	0.847
	Aravec Skip N-grams			
KNN	0.801	0.803	0.781	0.788
NB	0.736	0.752	0.697	0.702
LR	**0.881**	**0.878**	0.876	**0.877**
DT	0.693	0.683	0.683	0.684
SVM	**0.881**	0.877	**0.877**	**0.877**
RF	0.832	0.831	0.819	0.824
XGBoost	0.846	0.842	0.837	0.839
	Aravec CBOW Unigrams			
KNN	0.777	0.770	0.764	0.767
NB	0.724	0.716	0.721	0.717
LR	0.857	0.851	0.854	0.852
DT	0.684	0.674	0.675	0.674
SVM	**0.869**	**0.864**	**0.865**	**0.865**
RF	0.816	0.811	0.806	0.808
XGBoost	0.819	0.813	0.813	0.813
	Aravec CBOW N-grams			
KNN	0.788	0.782	0.777	0.779
NB	0.741	0.735	0.740	0.736
LR	0.851	0.846	0.848	0.847
DT	0.720	0.710	0.707	0.709
SVM	**0.872**	**0.868**	**0.869**	**0.868**
RF	0.821	0.816	0.813	0.814
XGBoost	0.845	0.840	0.837	0.839
	GloVe			
KNN	0.734	0.725	0.718	0.721
NB	0.738	0.73	0.733	0.731
LR	0.8	0.793	0.793	0.793
DT	0.683	0.677	0.682	0.677
SVM	**0.805**	**0.798**	**0.797**	**0.798**
RF	0.779	0.775	0.763	0.767
XGBoost	0.792	0.785	0.786	0.786

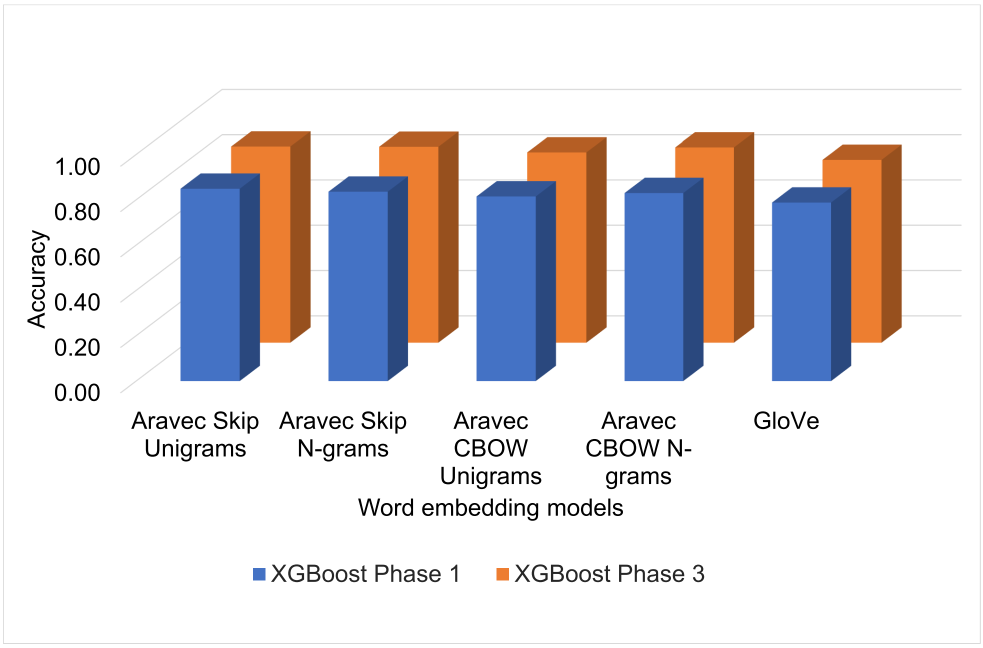


FIGURE 10. **Performance improvement of the XGBoost classifier.**

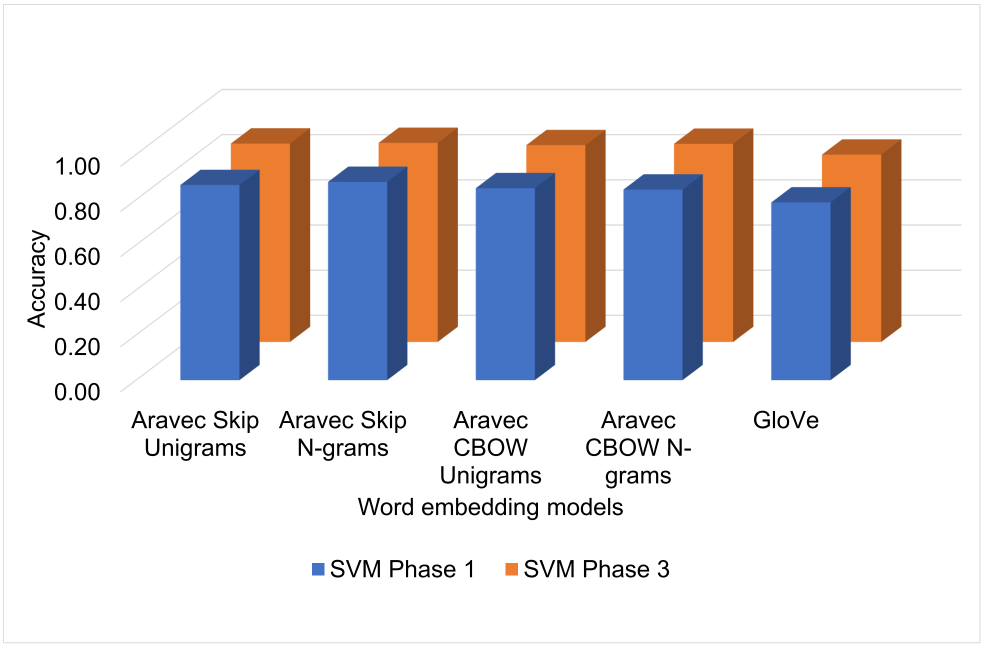


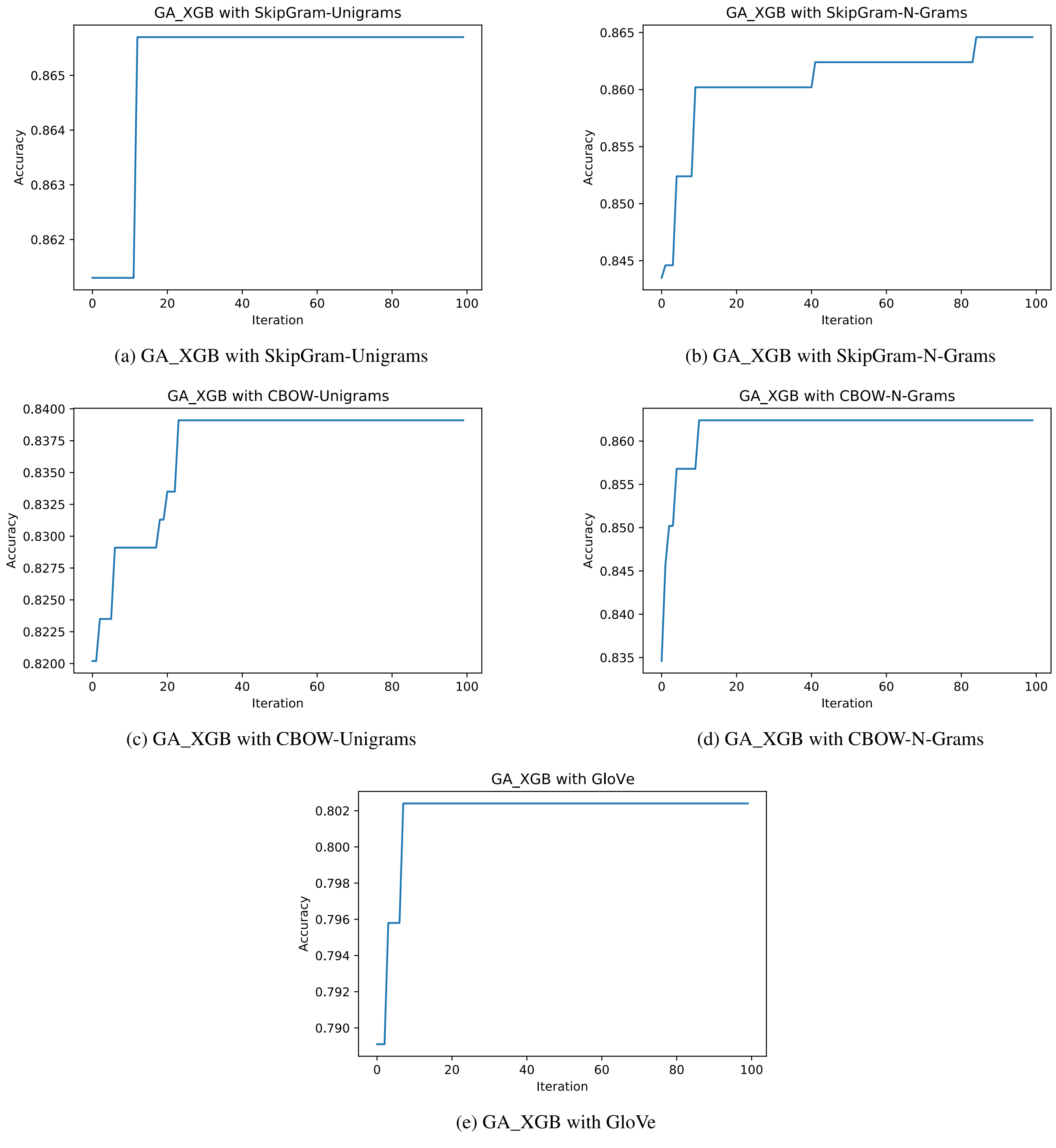
FIGURE 11. **Performance improvement of the SVM classifier.**

learned by illustrating the relationship between the number of iterations represented on the x-axis and the accuracy measure represented on the y-axis. The figures demonstrated that 100 iterations were sufficient to obtain the best results.

VI. DISCUSSION

Global openness and communication in social media resulted in the need to introduce intelligent systems to detect the misuse of social media platforms and prevent Internet abuse and the prevalence of offensive and hate. ML and NLP have efficient roles in building efficient systems to detect and solve offensive and hate problems.

Predicting offensive language in Twitter posts is considered a single-objective ML-based classification problem. Many researchers adopted various solutions to tackle this phenomenon. Some of these solutions were applied at

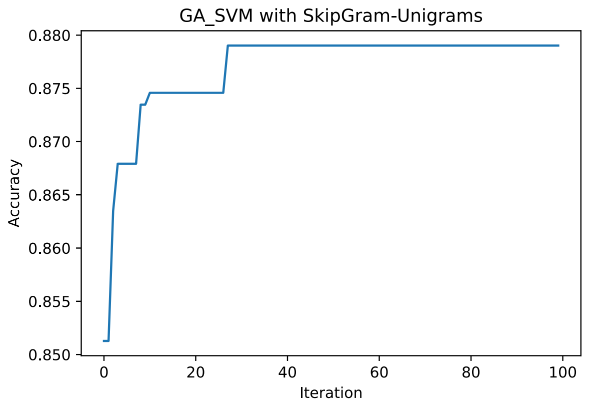


(a) GA_XGB with SkipGram-Unigrams (b) GA_XGB with SkipGram-N-Grams (c) GA_XGB with CBOW-Unigrams (d) GA_XGB with CBOW-N-Grams (e) GA_XGB with GloVe

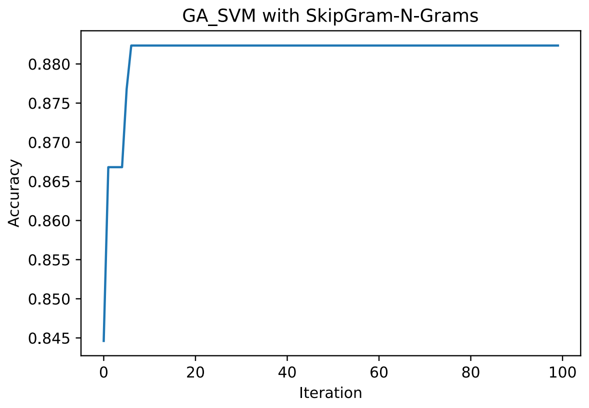
FIGURE 12. Learning curve for the XGBoost classifier.

the feature engineering levels, such as using pre-trained word embedding models rather than One Hot Encoding or the TF-IDF methods. Other ML models' solutions include optimizing the models' hyperparameters. However, searching for the optimal hyperparameter values for the ML models might encompass enormous time. For instance, XGBoost has many hyperparameters, which require a tremendous amount of effort and time to tune. In addition, the standard SVM classifier has a few limitations. The parameter setting of SVM is essential to enhance its performance. Therefore, discovering new techniques such as stochastic methods and evolutionary algorithms, like the Genetic Algorithm, was in great demand.

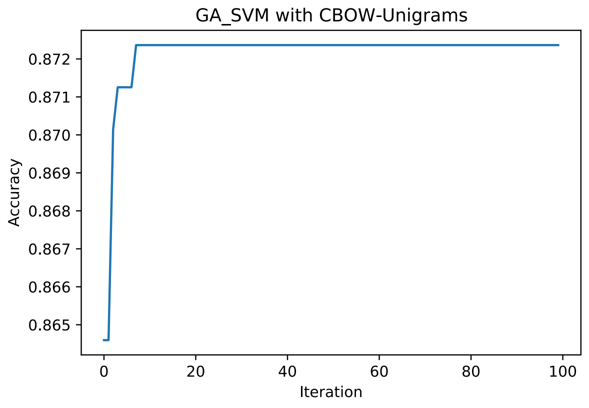
Therefore, this work attempted to maximize the models' performance by improving feature engineering and ML model building. The first solution tried to fine-tune the



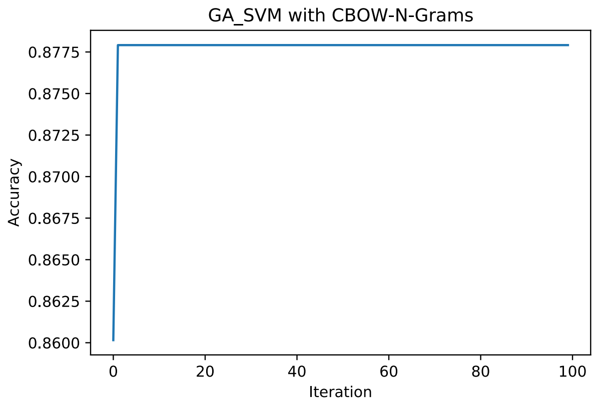
(a) GA_SVM with SkipGram-Unigrams



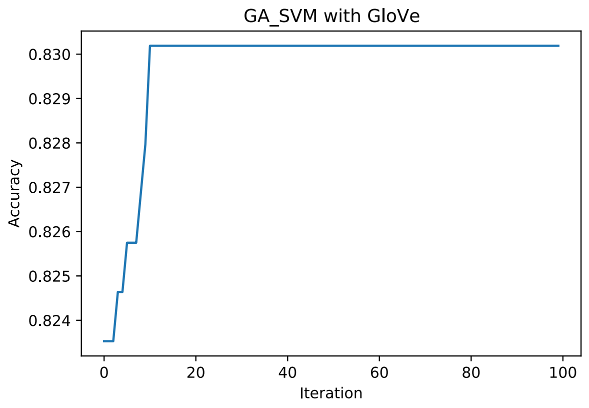
(b) GA_SVM with SkipGram-N-Grams



(c) GA_SVM with CBOW-Unigrams



(d) GA_SVM with CBOW-N-Grams



(e) GA_SVM with GloVe

FIGURE 13. **Learning curve for the SVM classifier.**

pre-trained models with new data from the training dataset; thus, the unseen vocabularies were added to the model. The results showed a slight improvement in the performance, and this was because the training dataset size that we used to retrain the word embedding models was relatively small, where the training dataset contains 3604 instances, and the added vocabularies to the models were less than 2000 vocabs. Although there was a slight improvement following this approach, the results encouraged us to apply this approach to a large scale of datasets.

Even though the first phase resulted in powerful algorithms such as XGBoost and SVM, the results were not convincing. This was due to the problem that we could not discover the best hyperparameters of the two models that generated the maximum accuracy. Therefore, we used a hybrid approach of classifiers (XGBoost and SVM) in the second phase with

TABLE 11. Performance evaluation of the proposed approach compared with the baseline models.

	Accuracy	Precision	Recall	F1-score
	Aravec SkipGram Unigrams			
Base classifier	0.877	0.872	0.875	0.873
GA-XGBoost	0.866	0.861	0.861	0.861
GA-SVM	**0.879**	**0.874**	**0.878**	**0.875**
	Aravec SkipGram N-grams			
Base classifier	0.881	**0.878**	0.876	0.877
GA-XGBoost	0.865	0.860	0.860	0.860
GA-SVM	**0.882**	**0.878**	**0.879**	**0.878**
	Aravec CBOW Unigrams			
Base classifier	0.857	0.851	0.854	0.852
GA-XGBoost	0.840	0.835	0.833	0.834
GA-SVM	**0.872**	**0.868**	**0.867**	**0.868**
	Aravec CBOW N-grams			
Base classifier	0.851	0.846	0.848	0.847
GA-XGBoost	0.862	0.858	0.858	0.858
GA-SVM	**0.878**	**0.873**	**0.876**	**0.874**
	GloVe			
Base classifier	0.800	0.793	0.793	0.793
GA-XGBoost	0.807	0.800	0.799	0.799
GA-SVM	**0.830**	**0.825**	**0.823**	**0.824**

a genetic algorithm (GA). The results showed a significant improvement in the models' performance.

The Glove model achieved the lowest results of all the models in all phases because it was trained on Arabic corpus from various resources. On the other hand, we trained the other word embedding models on a Twitter dataset compatible with the ArCybC dataset.

Moreover, we applied various transformer-based ML models such as AraBert. However, we did not obtain significant results where the accuracy rate did not exceed 87%. This was due to the small size of the ArCybC dataset. Therefore, we only presented the results from our proposed approach.

VII. CONCLUSION AND FUTURE WORK

In this study, an intelligent prediction system to detect the offensive language in Arabic tweets has been presented. We proposed a two-stage optimization prediction approach. In the first stage, we used fine-tuned pre-trained word embedding models. In the second stage, we used an evolutionary eXtreme Gradient Boosting (XGBoost) and the Support Vector Machine (SVM) algorithms based on a genetic algorithm (GA). We tested the proposed approach on an Arabic Cyberbullying Corpus (ArCybC), which contains data collected from Twitter. We experimented with three word embedding models: AraVec Unigram, AraVec N-gram, and GloVe.

The fine-tuning process involved training the models with more epochs from the ArCybC training dataset to increase the number of vocabs. The GA helped reduce the time and cost involved in the manual trial and error design methods and mitigate the challenges of the XGBoost and SVM to automatically reach the optimal hyperparameter values to run the models with maximum prediction capability. The proposed approach produced promising results even though the size of ArCybC dataset is relatively small. It indicated the effectiveness of the proposed system to be generalized to high-scale datasets in future research to increase the performance of the classifiers. Where then we can implement and evaluate Arabic language resources and ML models based on deep learning techniques, such as transfer-based models, like the Arabic version of BERT (AraBERT). We can use transfer-based models at both optimization levels; in which the BERT model could be used in the feature extraction phase to get representation from the frozen model and pass it to the task model, or system development phase to fine-tune weights for the target task.

ACKNOWLEDGMENT

The work of Bassam Hammo was on sabbatical leave from 2021 to 2022 from the King Abdullah II School of Information Technology, The University of Jordan to the Department of Software Engineering, King Hussein School of Computing Sciences, Princess Sumaya University for Technology, Amman, Jordan.

REFERENCES

[1] E. van der Walt, J. H. P. Eloff, and J. Grobler, "Cyber-security: Identity deception detection on social media platforms," *Comput. Secur.*, vol. 78, pp. 76–89, Sep. 2018.

[2] S. Sadiq, A. Mehmood, S. Ullah, M. Ahmad, G. S. Choi, and B.-W. On, "Aggression detection through deep neural model on Twitter," *Future Gener. Comput. Syst.*, vol. 114, pp. 120–129, Jan. 2021.

[3] T. Davidson, D. Warmsley, M. Macy, and I. Weber, "Automated hate speech detection and the problem of offensive language," in *Proc. Int. AAAI Conf. Web Social Media*, vol. 11, 2017, pp. 1–4.

[4] Z. Waseem, T. Davidson, D. Warmsley, and I. Weber, "Understanding abuse: A typology of abusive language detection subtasks," 2017, *arXiv:1705.09899*.

[5] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," in *Proc. Adv. Neural Inf. Process. Syst.*, 2013, pp. 3111–3119.

[6] J. Pennington, R. Socher, and C. Manning, "Glove: Global vectors for word representation," in *Proc. Conf. Empirical Methods Natural Lang. Process. (EMNLP)*, 2014, pp. 1532–1543.

[7] P. Bojanowski, E. Grave, A. Joulin, and T. Mikolov, "Enriching word vectors with subword information," *Trans. Assoc. Comput. Linguistics*, vol. 5, pp. 135–146, Dec. 2017.

[8] D. A. Al-Qudah, A. M. Al-Zoubi, P. A. Castillo-Valdivieso, and H. Faris, "Sentiment analysis for e-payment service providers using evolutionary extreme gradient boosting," *IEEE Access*, vol. 8, pp. 189930–189944, 2020.

[9] T. Chen and C. Guestrin, "XGBoost: A scalable tree boosting system," in *Proc. 22nd ACM SIGKDD Int. Conf. Knowl. Discovery Data Mining*, 2016, pp. 785–794.

[10] H. Faris, M. A. Hassonah, A. M. Al-Zoubi, S. Mirjalili, and I. Aljarah, "A multi-verse optimizer approach for feature selection and optimizing SVM parameters based on a robust system architecture," *Neural Comput. Appl.*, vol. 30, no. 8, pp. 2355–2369, Oct. 2017.

[11] H. Tao, M. Habib, I. Aljarah, H. Faris, H. A. Afan, and Z. M. Yaseen, "An intelligent evolutionary extreme gradient boosting algorithm development for modeling scour depths under submerged weir," *Inf. Sci.*, vol. 570, pp. 172–184, Sep. 2021.

[12] F. E. Ayo, O. Folorunso, F. T. Ibharalu, and I. A. Osinuga, "Machine learning techniques for hate speech classification of Twitter data: State-of-the-art, future challenges and research directions," *Comput. Sci. Rev.*, vol. 38, Nov. 2020, Art. no. 100311.

[13] D. E. Goldberg and J. H. Holland, "Genetic algorithms and machine learning," *Mach. Learn.*, vol. 3, pp. 95–99, 1988.

[14] T. Mikolov, K. Chen, G. Corrado, and J. Dean, "Efficient estimation of word representations in vector space," 2013, *arXiv:1301.3781*.

[15] O. Araque, I. Corcuera-Platas, J. F. Sánchez-Rada, and C. A. Iglesias, "Enhancing deep learning sentiment analysis with ensemble techniques in social applications," *Expert Syst. Appl.*, vol. 77, pp. 236–246, Jul. 2017.

[16] E. Altszyler, M. Sigman, S. Ribeiro, and D. Fernández Slezak, "Comparative study of LSA vs Word2vec embeddings in small corpora: A case study in dreams database," 2016, *arXiv:1610.01520*.

[17] M. Naili, A. H. Chaibi, and H. H. Ben Ghezala, "Comparative study of word embedding methods in topic segmentation," *Proc. Comput. Sci.*, vol. 112, pp. 340–349, Jan. 2017.

[18] A. B. Soliman, K. Eissa, and S. R. El-Beltagy, "AraVec: A set of Arabic word embedding models for use in Arabic NLP," *Proc. Comput. Sci.*, vol. 117, pp. 256–265, Jan. 2017.

[19] Y. Wang and I. Witten, "Inducing model trees for continuous classes," in *Proc. 9th Eur. Conf. Mach. Learn.*, Sep. 1997.

[20] S. Dreiseitl and L. Ohno-Machado, "Logistic regression and artificial neural network classification models: A methodology review," *J. Biomed. Inform.*, vol. 35, nos. 5–6, pp. 352–359, 2002.

[21] C.-Y. J. Peng, K. L. Lee, and G. M. Ingersoll, "An introduction to logistic regression analysis and reporting," *J. Educ. Res.*, vol. 96, no. 1, pp. 3–14, 2002.

[22] D. W. Aha, D. Kibler, and M. K. Albert, "Instance-based learning algorithms," *Mach. Learn.*, vol. 6, no. 1, pp. 37–66, 1991.

[23] P. Domingos and M. Pazzani, "On the optimality of the simple Bayesian classifier under zero-one loss," *Mach. Learn.*, vol. 29, nos. 2–3, pp. 103–130, 1997.

[24] S. R. Safavian and D. Landgrebe, "A survey of decision tree classifier methodology," *IEEE Trans. Syst., Man, Cybern.*, vol. 21, no. 3, pp. 660–674, May/Jun. 1991.

[25] B. E. Boser, I. M. Guyon, and V. N. Vapnik, "A training algorithm for optimal margin classifiers," in *Proc. 5th Annu. Workshop Comput. Learn. Theory (COLT)*, 1992, pp. 144–152.

[26] C.-L. Huang and C.-J. Wang, "A GA-based feature selection and parameters optimizationfor support vector machines," *Expert Syst. Appl.*, vol. 31, no. 2, pp. 231–240, Aug. 2006.

[27] A. Fawzy Gad, "PyGAD: An intuitive genetic algorithm Python library," 2021, *arXiv:2106.06158*.

[28] H. Mubarak, K. Darwish, W. Magdy, T. Elsayed, and H. Al-Khalifa, "Overview of OSACT4 Arabic offensive language detection shared task," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 48–52.

[29] M. Zampieri, P. Nakov, S. Rosenthal, P. Atanasova, G. Karadzhov, H. Mubarak, L. Derczynski, Z. Pitenis, and Ç. Çöltekin, "SemEval-2020 task 12: Multilingual offensive language identification in social media (OffensEval 2020)," in *Proc. SemEval*, 2020, pp. 1–23.

[30] A. Abuzayed and T. Elsayed, "Quick and simple approach for detecting hate speech in Arabic tweets," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 109–114.

[31] A. I. Alharbi and M. Lee, "Combining character and word embeddings for the detection of offensive language in Arabic," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 91–96.

[32] F. Husain, "OSACT4 shared task on offensive language detection: Intensive preprocessing-based approach," 2020, *arXiv:2005.07297*.

[33] M. Abdellatif and A. Elgammal, "Offensive language detection in Arabic using ULMFiT," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 82–85.

[34] M. Djandji, F. Baly, W. Antoun, and H. Hajj, "Multi-task learning using AraBert for offensive language detection," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 97–101.

[35] A. Keleg, S. R. El-Beltagy, and M. Khalil, "ASU_OPTO at OSACT4-offensive language detection for Arabic text," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 66–70.

[36] A. Elmadany, C. Zhang, M. Abdul-Mageed, and A. Hashemi, "Leveraging affective bidirectional transformers for offensive language detection," 2020, *arXiv:2006.01266*.

[37] S. Hassan, Y. Samih, H. Mubarak, and A. Abdelali, "ALT at SemEval-2020 task 12: Arabic and English offensive language identification in social media," in *Proc. 14th Workshop Semantic Eval.*, 2020, pp. 1891–1897.

[38] B. Haddad, Z. Orabe, A. Al-Abood, and N. Ghneim, "Arabic offensive language detection with attention-based deep neural networks," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 76–81.

[39] I. A. Farha and W. Magdy, "Multitask learning for Arabic offensive language and hate-speech detection," in *Proc. 4th Workshop Open-Source Arabic Corpora Process. Tools, Shared Task Offensive Lang. Detection*, 2020, pp. 86–90.

[40] H. Mubarak, K. Darwish, and W. Magdy, "Abusive language detection on Arabic social media," in *Proc. 1st Workshop Abusive Lang. Online*, 2017, pp. 52–56.

[41] A. Alakrot, M. Fraifer, and N. S. Nikolov, "Machine learning approach to detection of offensive language in online communication in Arabic," in *Proc. IEEE 1st Int. Maghreb Meeting Conf. Sci. Techn. Autom. Control Comput. Eng. (MI-STA)*, May 2021, pp. 244–249.

[42] S. Alsafari, S. Sadaoui, and M. Mouhoub, "Hate and offensive speech detection on Arabic social media," *Online Social Netw. Media*, vol. 19, Sep. 2020, Art. no. 100096.

[43] H.-S. Lee, H.-R. Lee, J.-U. Park, and Y.-S. Han, "An abusive text detection system based on enhanced abusive and non-abusive word lists," *Decis. Support Syst.*, vol. 113, pp. 22–31, Sep. 2018.

[44] H. Mubarak and K. Darwish, "Arabic offensive language classification on Twitter," in *Social Informatics*, I. Weber, K. M. Darwish, C. Wagner, E. Zagheni, L. Nelson, S. Aref, and F. Flöck, Eds. Cham, Switzerland: Springer, 2019, pp. 269–276.

[45] V. K. Jha, P. Hrudya, P. N. Vinu, V. Vijayan, and P. Prabaharan, "DHOT-repository and classification of offensive tweets in the Hindi language," *Proc. Comput. Sci.*, vol. 171, pp. 2324–2333, Jan. 2020.

[46] H. Karayiğit, Ç. İ. Aci, and A. Akdağli, "Detecting abusive Instagram comments in Turkish using convolutional neural network and machine learning methods," *Expert Syst. Appl.*, vol. 174, Jul. 2021, Art. no. 114802.

[47] P. Kapil and A. Ekbal, "A deep neural network based multi-task learning approach to hate speech detection," *Knowl.-Based Syst.*, vol. 210, Dec. 2020, Art. no. 106458.

[48] A. Ziani, N. Azizi, D. Zenakhra, S. Cheriguene, and M. Aldwairi, "Combining RSS-SVM with genetic algorithm for Arabic opinions analysis," *Int. J. Intell. Syst. Technol. Appl.*, vol. 18, nos. 1–2, pp. 152–178, 2019.

[49] K. Kumari and J. P. Singh, "Multi-modal cyber-aggression detection with feature optimization by firefly algorithm," *Multimedia Syst.*, to be published.

[50] K. J. Madukwe, X. Gao, and B. Xue, "A GA-based approach to fine-tuning BERT for hate speech detection," in *Proc. IEEE Symp. Comput. Intell. (SSCI)*, Dec. 2020, pp. 2821–2828.

[51] F. Husain and O. Uzuner, "A survey of offensive language detection for the Arabic language," *ACM Trans. Asian Low-Resource Lang. Inf. Process.*, vol. 20, no. 1, pp. 1–44, Apr. 2021.

[52] F. Shannag, B. H. Hammo, and H. Faris, "The design, construction and evaluation of annotated Arabic cyberbullying corpus," *Educ. Inf. Technol.*, to be published.

[53] F. B. Shannag and B. H. Hammo, "Lessons learned from event detection from Arabic tweets: The case of Jordan flash floods near dead sea," in *Proc. IEEE Jordan Int. Joint Conf. Electr. Eng. Inf. Technol. (JEEIT)*, Apr. 2019, pp. 806–811.

[54] M. Habib, M. Faris, A. Alomari, and H. Faris, "AltibbiVec: A word embedding model for medical and health applications in the Arabic language," *IEEE Access*, vol. 9, pp. 133875–133888, 2021.

[55] C. Perlich, F. Provost, and J. Simonoff, "Tree induction vs. logistic regression: A learning-curve analysis," *J. Mach. Learn. Res.*, pp. 211–255, 2003.

FATIMA SHANNAQ received the B.Sc. and M.Sc. degrees in computer information systems (CIS) from Yarmouk University, Jordan, in 2006 and 2009, respectively, and the Ph.D. degree in computer science from The University of Jordan, Jordan, in 2022. She was worked as an IT Instructor at Princess Nourah Bint Abdulrahman University, Riyadh, Saudi Arabia. She is currently an Instructional Designer at the Jordan University of Science and Technology, Jordan. Her research interests include evolutionary computation, machine learning, and natural language processing in social network analysis and other research fields.

BASSAM HAMMO received the B.S. degree in computer science from The University of Jordan (JU), Amman, in 1987, the Ph.D. degree in computer science from DePaul University, Chicago, IL, USA, in 2002, and the M.Sc. degree in computer science from Northeastern University, in 2003. He was worked as a QA Engineer with CCH-Walters Kluwer, CCC, and Classified Ventures, Chicago, from 1995 to 2003. Since September 2003, he has been with JU. He taught at Robert Morris University, Chicago, from 1996 to 2003, King Saud University, from 2010 to 2012, and Princess Sumaya University for Technology (PSUT), Amman, Jordan, in 2014. He has been promoted to an Associate Professor, in 2010 and to a Professor, in 2016. His administrative work at JU included the Chairperson of Department of Computer Information System, from 2008 to 2010, the Deputy Dean from 2012 to 2014, and the Dean of KASIT, from 2016 to 2018. He is currently on sabbatical leave from JU to PSUT. He published many papers in refereed journals and international conferences. He supervised and co-supervised over 50 theses/dissertations. His main research interests include NLP, HCI, data and text mining, and sentiment analysis.

HOSSAM FARIS received the B.A. degree in computer science from Yarmouk University, Jordan, in 2004, the M.Sc. degree in computer science from Al-Balqa' Applied University, Jordan, in 2008, and the Ph.D. degree in e-business from the University of Salento, Italy, in 2011. In 2016, he worked as a Postdoctoral Researcher with the GeNeura Team, Information and Communication Technologies Research Center (CITIC), University of Granada, Spain. He is currently a Professor with the Department of Information Technology, King Abdullah II School for Information Technology, The University of Jordan, Jordan. He co-founded The Evolutionary and Machine Learning (Evo-ML.com) Research Group. His research interests include computational intelligence, evolutionary computation, knowledge systems, data mining, and applied machine learning. He was awarded a Full-Time Competition-Based Scholarship from the Italian Ministry of Education and Research to pursue his Ph.D. degrees in e-Business at the University of Salento.

PEDRO A. CASTILLO-VALDIVIESO was born in Granada, in 1974. He received the B.Sc. degree in computer science and the Ph.D. degree from the University of Granada, Spain, in 1997 and 2000, respectively.

He was a Teaching Assistant at the Department of Computer Science, University of Jaén, Spain. He was also a Visiting Researcher at the Edinburgh Napier University, Edinburg, U.K., in July 1998, and the Santa Fe Institute, Santa Fe, NM, USA, in September 2000. He has been the Leader of several research projects, and directed five Ph.D. students. He is currently a Full Professor at the Department of Computer Architecture and Technology, University of Granada. His main research interests include bio-inspired systems, hybrid systems, and combination of evolutionary algorithms and neural networks.

• • •