Modified Metaheuristics Optimization for Cyberbullying Detection on Online Data Science Platform

NEBOJSA BACANIN[1a], LUKA JOVANOVIC[2b], ILJA UZELAC BUJISIC[2],
JELENA KALJEVIC[3c], JELENA CADJENOVIC[4d], MILOS ANTONIJEVIC[1e],
MIODRAG ZIVKOVIC[1f]

[1]Faculty of Informatics and Computing
Singidunum University
Belgrade, SERBIA

[2]Faculty of Technical Sciences
Singidunum University
Belgrade, SERBIA

[3]Faculty for Health and Business Studies
Singidunum University
Belgrade, SERBIA

[4]Deprtmant of Business Economics
Singidunum University
Belgrade, SERBIA

[a]0000-0002-2062-924X, [b]0000-0001-9402-7391, [c]0000-0002-5135-8083,
[d]0000-0002-3406-3347, [e]0000-0002-5511-2531, [f]0000-0002-4351-068X

Abstract: —Online harassment detection faces significant challenges due to its expansive reach and anonymity. Addressing this issue demands effective detection mechanisms capable of processing vast data streams and adapting to evolving online language. Leveraging advancements in artificial intelligence, we propose a novel approach grounded in natural language processing (NLP) and metaheuristic algorithms. Our methodology integrates term frequency-inverse document frequency (TF-IDF) encoding and the AdaBoost algorithm for classification. To tackle the NP-hard problem of hyperparameter selection, we introduce a modified crayfish optimization algorithm (COA), termed GI-COA. This paper represents a pioneering effort in utilizing metaheuristic algorithms for hyperparameter selection in harassment detection models. Through experimentation, we demonstrate the efficacy of our approach in fostering a safer online environment. The best performing optimize models demonstrate and accuracy exceeding 77%.

Key-words:—Online harassment, Hyperparameter optimization, Metaheuristic algorithms, Natural language processing, Hybridization,

Received: March 2, 2024. Revised: September 6, 2024. Accepted: October 4, 2024. Published: November 5, 2024.

1. Introduction

In recent decades, electronic communication became one of the most important communication channels. Informal in-person chats were replaced by online forums, posting on social media and direct messaging. This opened the doors of a new kind of harassment, online harassment [1]. Unlike few decades ago, when harassment was contained in either direct face-to-face insults or written communication (letters, articles, street graffiti,etc.), now it is mainly conducted on online platforms. This is why now it can have much broader reach, and combined with the anonymity of online landscape, but also it can keep individuals anonymous and therefore not accountable. This can have a disastrous effect on more vulnerable people and groups, especially younger ones who did not develop appropriate coping mechanisms [2].

It is important to conduct research and development in the field of detection and prevention mechanisms that will keep sure our online world more friendly, decent and less toxic. In general, detection is difficult due to two main reasons. First, online platforms generate vast data streams each second, so these mechanisms need to process large amounts of data. Second difficulty is the constant change of online language. All of these online trends (e.g. TikTok or Instagram trends and slangs) change rapidly [3], and each one of them opens new types of potential harassment. Since the culture changes quite fast, harassment takes new form all the time - which in return makes designing detection systems.

This is why the recent development of artificial intelligence (AI) comes very handy. Instead of relying on explicit instructions, AI systems can adopt themselves with new data. That is what makes them suitable for dealing with constantly changing ways of online harassment. In general AI models consist of architecture and hyperparameters that are chosen while they are designed and parameters that are tuned during training and are constantly updated with the flow of new data. In this paper the problem is tackled with natural language processing (NLP) [4], where the selection of the hyperparameters is an NP-hard problem. This is why metaheuristic is used concretely

crayfish optimisation algorithm (COA) [5] in the hyperparameter selection process.

This paper tackles this problem using few novel approaches. The first question is how to handle data, i.e. how to code words and their appearances in documents. To grasp this, term frequency - inverse document frequency (TF-IDF) [6] is implemented as the main metric. The classifier that classifies documents as harassment or non-harassment is based on AdaBoost [7] algorithm. The NP-hard problem of hyperparameter tuning is tackled by a modification of COA. This paper is a first bridge of using metaheuristical algorithms in hyperparameter selection in models for harassment detection.

The rest of the paper starts with the overview of the general works in NLP models and existing spam detection models. This is followed by a short discussion about metaheuristic algorithms in general. Brief descriptions of AdaBoost, TF-IDF, COA and the modified version of COA, GI-COA are given. After the experimental setup is covered, simulation outcomes are obtained and discussed before the conclusion of the paper.

2. Related Works

In general, work in harassment detection lies closely to spam detection problems as a part of wider NLP research. In this broad and fast-developing area some of the most important research papers are [8] introducing attention-based transformer models, [9] which introduced the original GPT model and demonstrated the effectiveness of generative pre-training on a diverse set of NLP tasks and [10] which introduced the more powerful GPT-2 which demonstrated effectiveness on various NLP tasks to the point that it sparked ethical considerations of such models.Some relevant papers in spam detection include BERT for Spam detection [11] which use BERT, bidirectional encoder representation from transformers introduced in [12]. which is based on support vector machines. The SVM based approach can be useful due to its effectiveness with dealing with high-dimensional data but it can be computationally expensive and requires careful selection of hyperparameters and kernel functions which might be impractical. On the other hand BERT based approaches show strong results on vast NLP tasks including spam, and large text corpus it was trained on makes it very generalizable. On the other hand, its large number of hyperparameters makes the training process expensive.

When using machine learning algorithms, some architecture is chosen and then parameters are trained using learning data. The difficulty of the process lies in selecting an appropriate model, finding and processing training data and then tuning the parameters using it. However, before the training process, it is often necessary to define some parameters that will not change during the training but will significantly affect the performance of the model. This values are referred to as hyperparameters. The process of finding appropriate hyperparameters consists of determining which parameters are hyperparameters, defining the search range for each of them, choosing a metric to grade each choice, defining the search strategy, tuning the hyperparameters, selecting the best choice and then testing the model with these hyperparameters on some independent test set. Since grid search if done directly can be an NP-hard problem, a metaheuristical approach is used to find a good choice of hyperparameters.

Literature presents many potential candidate algorithms of optimizations. Some popular choices used by researchers include established optimizers like the variable neighborhood search (VNS) [13], GA [14], and sine cosine algorithms (SCA) [15] are also examined. Metaphor-based optimizers, including bat algorithm (BA) [16], whale optimization algorithm (WOA) [17], and Harris hawk optimizer (HHO) [18], are also popular choices. Some newer examples include the Botox optimization algorithm BOA [19] inspired by medical treatment processes.

Optimizes have been successfully applied in several field such as cybersecurity where these algorithms tackle intrusion detection [20], spam identification [21] as well as fraud identification [22]. However, hybrid algorithms have also proven to be highly effective at tackling optimizations with notable implementations showing interesting results in finance [23] and medical fields [24]. With more recent examples tackling problems in energy [25] and production forecasting [26].

2.1 Adaboost

The AdaBoost [7] optimizer widely considered a good choice for optimization problems. This approach adopts ensemble techniques and uses several simpler models to formulate a decision. By applying weights to classifier votes better decisions can be made though a sort of consensus method. Should a correct classification be made weights of classifiers are increased, otherwise weights are decreased. Classifier errors can be determined as per Eq.(1):

$$\epsilon_t = \frac{\sum_{i=1}^{N} w_{i,t} \cdot \mathrm{I}(h_t(x_i) \neq y_i)}{\sum_{i=1}^{N} w_{i,t}} \tag{1}$$

the term ϵ_t represents a weighted adjustment for a weak model, where $w_{i,t}$ corresponds to the weight. The expression $h_t(x_i)$ refers to the prediction made by the weak classifier, and y_i indicates the actual value or ground truth. The function I converts the output to 1 if the classification is positive, and 0 if it is negative.

Additional classifiers are built using the determined weights, and this weight-adjustment process is repeated. Generally, a large number of classifiers are accumulated to create accurate models. Each sub-model is given a score, and these are combined to form a linear model. The classifier's weight within the ensemble can be computed using Eq. 2.

$$\alpha_t = \frac{1}{2} \ln \left(\frac{1 - \epsilon_t}{\epsilon_t} \right) \tag{2}$$

the term α_t signifies the assigned weight for each weak classifier. This value is determined based on the error value, ϵ_t, and defines the classifier's contribution to the final result. These weights are updated according to the following rule:

$$w_{i,t+1} = w_{i,t} \cdot \exp\left(-\alpha_t \cdot y_i \cdot h_t(x_i)\right) \tag{3}$$

where $w_{i,t}$ represents the weight a given instance, α_t denotes the weight of the weak classifier y_i is the ground truth value, and $h_t(x_i)$ signifies the prediction.

2.2 TF-IDF

It is difficult to directly convert words into numbers in a meaningful way. What matters is to somehow quantify sentences such that from that quantification there is some insight about it. To do that, a measure called TF-IDF is applied. That measure captures how relevant is each word in a document where a document belongs to some collection of documents - i.e. a corpus.

Intuitively speaking useful measure would be a one that increases proportionally with each new appearance of a word in a document, and also decreases with the frequency of word appearance in corpus. This makes sense, since to extract relevant words for a specific documents, the words that appear in many documents should be penalised. These words will not give any insight of this specific document that is being analysed. Formally for a word w , document d in a corpus C of documents *TF* and *IDF* are:

$$TF = \frac{\text{num of appearances of } w \text{ in } d}{\text{num of words in } d} \tag{4}$$

$$IDF = \log_2\left(\frac{\text{num of documents in } C}{\text{num of documents in } C \text{ containing } w}\right) \tag{5}$$

Then TF-IDF is defined by TF-IDF := *TF*IDF*. This is the quantity that will be used for relevant words. Note that the base of logarithm is not important since it just scales TF-IDF by a constant. More on TF-IDF can be found in [27]

3. Methods

3.1 The Original Crayfish Optimisation Algorithm

Crayfish, also known as crawfish or freshwater lobsters, are crustecans resembling small lobsters that typically live in river waters. The COA is a metaheuristic algorithm inspired by behaviour of crayfish. Behaviour consists of foraging, competition and summer vacation stage. First two stages are the exploitation stage while summer vacation is exploration stage of COA. First all the variables used in COA are explained. Let n be the number of crayfishes, T the number of iterations ,d the dimension of space in which the process takes place. The position of crayfishes at time t, for $0 \leq t \leq T$is an $n \times d$ matrix $X^{(t)}$ where $X_i^{(t)}$ is the d-dimensional vector representing the location of i-th crayfish at time t for $1 \leq i \leq n$.

The $n \times d$ matrices L_b and U_b are lower and upper bounds on position of crayfishes. At each time, the temperature $temp$ of the system is updated. Temperature defines in which of the three possible states (foraging, competition and summer vacation) the system is which is essential for knowing the next position. The location of food (in cases when used) will be X_{food}, X_{shade} the location of the cave, f the fitness function, that grades each position and p is a random variable depending on $temp$. More details on these variables can be found in [28]. The initial positions of crayfishes is chosen by $X^{(0)} = L_b + (U_b - L_b) * rand$ where $rand$ is a random variable uniformly distributed in $[0, 1]$. (Essentially a random point is picked in the d-dimensional space between the given lower and upper bounds for each crayfish.) In each new time-step, first the temperature $temp$ is updated. Take:

$$temp = 20 + 15 * rand \tag{6}$$

where $rand$ is a uniform random variable in $[0, 1]$ that is updated in each step.

Depending on whether $temp > 30$ or not there are two subcases. Algorithm enters in first case if $temp > 30$ (summer resort or competition stage). Then the two cases are separated : if $rand < 0.5$ it is in summer resort stage, otherwise it is in competition stage. In this case, crayfish will approach to cave for summer resort, instead of fighting. Then the values of X will be updated according to the following equation:

$$X_{i,j}^{(t+1)} = X_{i,j}^{(t)} + (2 - \frac{t}{T}) * rand * (X_{shade} - X_{i,j}^{(t)}) \tag{7}$$

To see how we update X_{shade} in each iteration, check [28]. This means that crayfishes start fighting for a cave. To capture the fight, another variable z is introduced:

$$z := round(rand * (n - 1)) + 1 \tag{8}$$

Then:

$$X_{i,j}^{(t+1)} = X_{i,j}^{(t)} - X_{z,j}^{(t)} + X_{shade}. \tag{9}$$

Algorithm enters second case if $temp \leq 30$. Here, the crayfish go to food which is located on $X_G = X_{food}$. The food size Q is defined as:

$$Q = 3 * rand * \frac{f(X_i^{(t)})}{f(X_{food})}. \tag{10}$$

Depending on the size of the food, crayfish either splits it into two parts, or it moves to it and eats it directly. If $Q > 2$, then X_{food} becomes $e^{\frac{-1}{Q}} * X_{food}$ and

$$X_{i,j}^{(t+1)} = X_{i,j}^{(t)} + X_{food} * p * (cos(2\pi * rand) - sin(2\pi * rand)) \tag{11}$$

Otherwise, if $Q \leq 2$, then:

$$X_{i,j}^{(t+1)} = ((X_{i,j}^{(t)} - X_{food}) + rand * X_{i,j}^{(t)}) * p \tag{12}$$

This gives the idea on how X evolves in time. The final X at time T is the solution that is taken as the result of COA for hyperparameter selection.
This was a brief overview of COA. To see more details, check [28].

3.2 Generically Inspired COA

Although the initial COA performs well, it has some shortcomings when evaluated using standard CEC [29] functions. To address these limitations and enhance COA's effectiveness, this study introduces hybridization techniques. Inspired by the Genetic Algorithm (GA) [14], we propose a new algorithm called the genetically inspired COA (GICOA).

In the GICOA algorithm, a unique mechanism is activated after each iteration, selecting a random agent and merging it with the best solution found so far. Their parameters

are uniformly blended using a control parameter denoted as pc, which is empirically set to 0.1. Additionally, parameter mutation is introduced: this process selects a random value within a specified range, then either adds or subtracts half of this value from the parameter based on the mutation direction parameter md, also set to 0.1.

After generating a new solution, the worst-performing solution in the swarm is replaced by the new agent. The evaluation of the new solution is deferred until the next iteration, keeping the computational complexity the same as the original algorithm. To provide a comprehensive understanding, the pseudocode is presented in Algorithm 1.

Algorithm 1 Pseudo code of the introduced GICOA.

```
Generate population P
while current; itteration is less than maxiumum; itterations do
    Assess agents in P
    for Each agent X in P do
        Update locations by applying the COA search
        Generate a new solution NS, with based on introduced mechanism
        Mutate genome of NS
        Replace the worst solution in P with NS
    end for
end while
return The best solution attained within P
```

4. Experimental Setup

To evaluate the performance of the introduced optimizer and the overall viability of the introduced approach a modified real world curated dataset is used acquired form Kaggle [1]. The parse Kaggle parsed portion of the dataset is used for simulations conducted in this work. Several contemporary optimizers are subjected to a comparative analysis. The optimizers included in the comparisons include the modified algorithm alongside the original COA [5]. Well established optimizers are also evaluated such as the VNS [13], GA [14] and SCA [15]. Further metaphor based optimizers such as the BA [16], WOA [17] and HHO [18] are evaluated. Finally the recently introduced BOA [19] are assessed. Algorithms are independently implemented according to the parameter settings provided in the original works that introduced the optimizers.

AdaBoost parameters are subjected to optimization with constraints provided in Table I. The respective constrains have been empirically determined. Each optimizer was allocated ten iterations and a population size of eight agents each. Optimizations are carried out in 30 independent runs. Standard classifications metrics are used to evaluate the tested optimizes including accuracy, precision, f1-score and recall. Finally the Cohen's kappa [30] metric is used as the indicator function due to the imbalance observed in the dataset.

TABLE I
ADABOOST HYPERPARAMETERS SUBJECTED TO OPTIMIZATION.

Parameter	constraints
count of estimators	$[10, 20]$
depth	$[1, 5]$
learning rate	$[0.01, 2]$

[1]https://www.kaggle.com/datasets/saurabhshahane/cyberbullying-dataset

TABLE II
OBJECTIVE FUNCTION OUTCOME FOR THE CONDUCTED SIMULATIONS.

Method	Best	Worst	Mean	Median	Std	Var
AB-GICOA	**0.479523**	**0.464299**	**0.471482**	**0.471334**	0.005674	3.22E-05
AB-COA	0.463551	0.440244	0.454147	0.455728	0.006455	4.17E-05
AB-VNS	0.464634	0.448140	0.458246	0.458168	0.004878	2.38E-05
AB-GA	0.473523	0.452341	0.460195	0.458979	0.006538	4.27E-05
AB-BA	0.475901	0.444988	0.457535	0.454765	0.009970	9.94E-05
AB-SCA	0.473954	0.451173	0.461481	0.461740	0.007691	5.92E-05
AB-WOA	0.471007	0.450067	0.459053	0.458005	0.005476	3.00E-05
AB-HHO	0.464634	0.453648	0.458976	0.458444	**0.003423**	**1.17E-05**
AB-BOA	0.473954	0.453393	0.463898	0.464007	0.006615	4.38E-05

TABLE III
CAPTION

Method	Best	Worst	Mean	Median	Std	Var
AB-GICOA	**0.221340**	**0.226190**	**0.223821**	**0.223986**	**0.001777**	**3.16E-06**
AB-COA	0.227072	0.232363	0.229773	0.229497	0.002061	4.25E-06
AB-VNS	0.226190	0.231922	0.229112	0.229056	0.002434	5.93E-06
AB-GA	0.223545	0.234127	0.229112	0.228395	0.003575	1.28E-05
AB-BA	0.223986	0.233686	0.229497	0.229277	0.003837	1.47E-05
AB-SCA	0.223986	0.233686	0.228560	0.228836	0.003436	1.18E-05
AB-WOA	0.223986	0.236332	0.228671	0.227734	0.003386	1.15E-05
AB-HHO	0.226190	0.235009	0.229497	0.229056	0.002736	7.48E-06
AB-BOA	0.223986	0.233245	0.226852	0.225970	0.002866	8.21E-06

5. Experimental Outcomes

Objective function outcomes for classifiers optimized by each model are provided in Table II. Outcomes are showcased in terms of best, worst, mean and median outcomes. Furthermore details on algorithms stability are provided in the form of the standard deviation and variance in outcomes over 30 independent runs. As shown the introduced optimizer outperformed competing models across al metrics including stability.

Indicator function outcomes for classifiers optimized by each model are provided in Table III. Outcomes are showcased in terms of best, worst, mean and median outcomes. Furthermore details on algorithms stability are provided in the form of the standard deviation and variance in outcomes over 30 independent runs. As shown the introduced optimizer outperformed competing models across all metrics including stability.

Comparisons in terms of stability are provided in Figure 1. The introduced modified algorithm outperforms competing optimizers including the original version, providing a high rate of stability as well as the highest quality of results.

Further detailed comparisons in terms of accuracy, per-class precision as well as macro and weight averages are provided in Table IV. The introduced optimize show the highest rate of accuracy, as well as macro and weighted averages. However, certain algorithms outperform the introduced optimizer in terms of precision and recall. This is to be somewhat expected, as according to the NFL no single optimizer performs equally well for all tasks and across all metrics. Hence experimentation is mandated in order to determine the optimal combination op problem and optimizer.

Further details on an optimizes ability to avoid local optima, and sufficiently explore the search space for good solutions can be discerned form the respective optimizers convergence graphs. Graphs for tested algorithms are provide in Figure 2. The introduced optimizer manages to avoid local optima that constrains other optimizers with insufficient exploration ability attaining the best comparative outcomes in the ninth iteration.

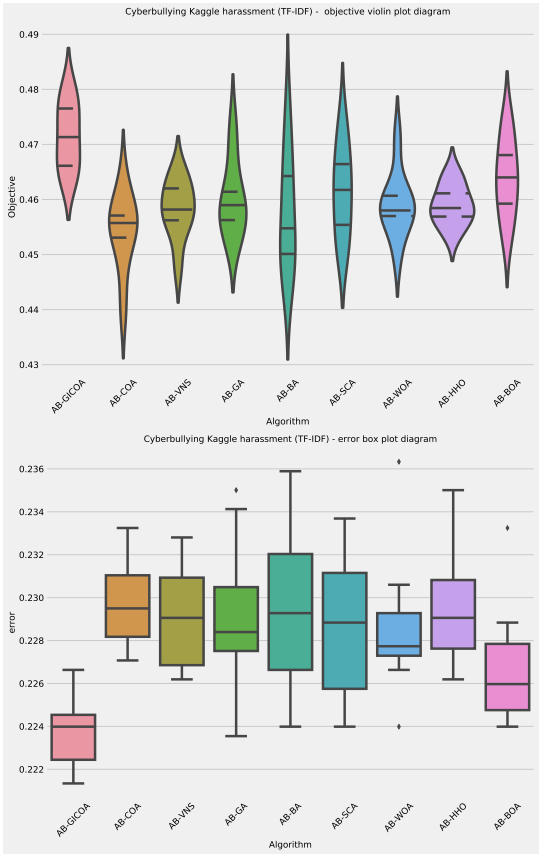

Fig. 1. Objective and indicator function distributions.

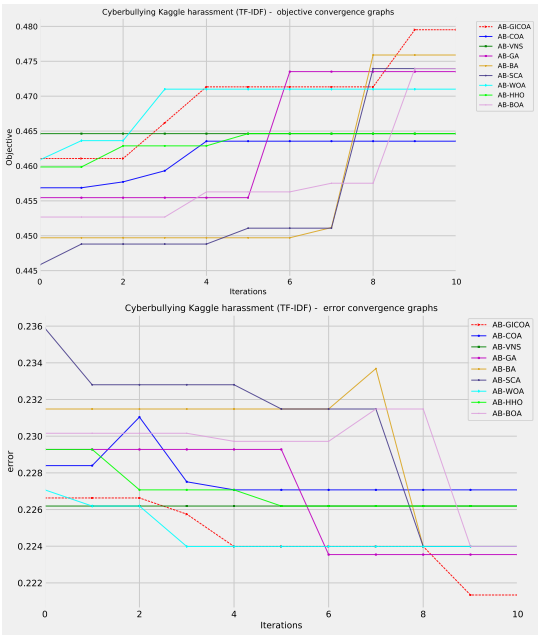

Fig. 2. Objective and indicator function convergence plots.

TABLE V
OPTIMIZER MODEL PARAMETER SELECTIONS MADE BY OPTIMIZERS.

Methods	Number of estimators	Depth	Learning rate
AB-GICOA	19	2	1.327088
AB-COA	16	2	1.267089
AB-VNS	20	2	1.185109
AB-GA	20	2	1.377726
AB-BA	20	2	1.371097
AB-SCA	20	2	1.320228
AB-WOA	17	2	1.346582
AB-HHO	17	2	1.332248
AB-BOA	20	2	1.324441

TABLE IV
CAPTION

Method	metric	non-harassment	harassment	accuracy	macro avg	weighted avg
AB-GICOA	precision	0.785420	0.758377	**0.778660**	**0.771899**	**0.775941**
	recall	0.906993	0.540881	**0.778660**	**0.723937**	**0.778660**
	f1-score	**0.841840**	**0.631424**	**0.778660**	**0.736632**	**0.768083**
AB-COA	precision	0.779138	0.753623	0.772928	0.766380	0.770194
	recall	0.907671	0.523270	0.772928	0.715471	0.772928
	f1-score	0.838507	0.617669	0.772928	0.728088	0.761097
AB-VNS	precision	0.778746	0.758242	0.773810	0.768494	0.771558
	recall	0.910387	0.520755	0.773810	0.715571	0.773810
	f1-score	0.839437	0.617450	0.773810	0.728443	0.761624
AB-GA	precision	0.783118	0.756228	0.776455	0.769673	0.773692
	recall	0.906993	0.534591	0.776455	0.720792	0.776455
	f1-score	0.840516	0.626382	0.776455	0.733449	0.765456
AB-BA	precision	**0.786350**	0.746141	0.776014	0.766245	0.772256
	recall	0.899525	**0.547170**	0.776014	0.723347	0.776014
	f1-score	0.839139	0.631350	0.776014	0.735244	0.766303
AB-SCA	precision	0.784325	0.751313	0.776014	0.767819	0.772754
	recall	0.903598	0.539623	0.776014	0.721610	0.776014
	f1-score	0.839748	0.628111	0.776014	0.733930	0.765563
AB-WOA	precision	0.781341	**0.759494**	0.776014	0.770417	0.773683
	recall	0.909708	0.528302	0.776014	0.719005	0.776014
	f1-score	0.840652	0.623145	0.776014	0.731899	0.764410
AB-HHO	precision	0.778746	0.758242	0.773810	0.768494	0.771558
	recall	0.910387	0.520755	0.773810	0.715571	0.773810
	f1-score	0.839437	0.617450	0.773810	0.728443	0.761624
AB-BOA	precision	0.784325	0.751313	0.776014	0.767819	0.772754
	recall	0.903598	0.539623	0.776014	0.721610	0.776014
	f1-score	0.839748	0.628111	0.776014	0.733929	0.765563
	support	1473	795			

Additional details on the performance of the best performing model optimized by the introduced modified algorithm mare produced in Figure 3 where the confusion matrix and PR curved are shown. Parameter selections made by each optimizer for the respective best performing models are also provided to encourage experimental repeatability in Table V.

6. Conclusion

Detecting online harassment poses substantial hurdles due to the wide-ranging scope and the anonymity it often involves. Overcoming these challenges requires robust detection methods capable of handling large volumes of data and adapting to the ever-evolving nature of online communication. Through the fusion of NLP and metaheuristic algorithms, particularly leveraging TF-IDF encoding and AdaBoost algorithm, this work proposes an data driven approach. To address the complexity of hyperparameter selection, a further contribution is given in the form of a introduced modification to the COA, dubbed GICOA. Experimental outcomes conducted with real world data suggest that the best-performing models achieving an accuracy surpassing 77%, thus contributing significantly to fostering a safer online ecosystem.

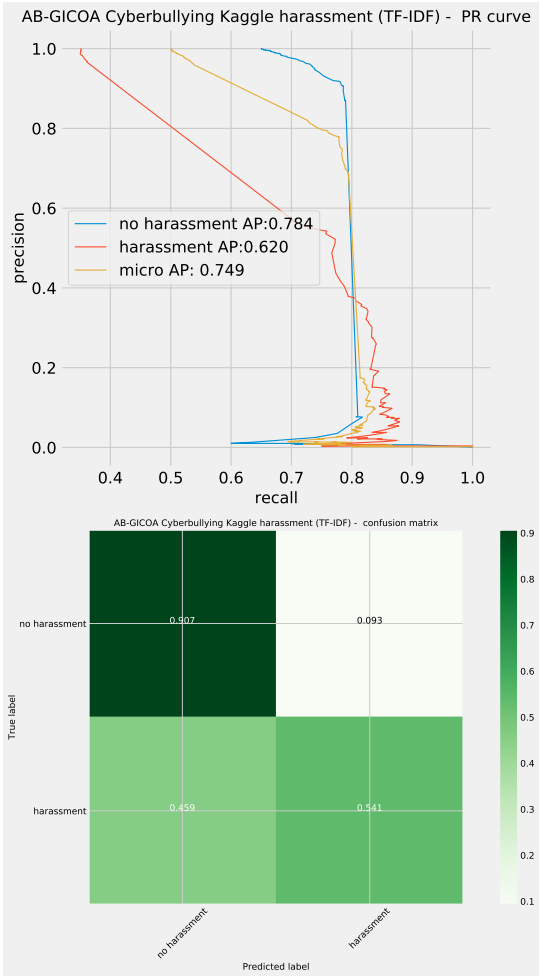

Fig. 3. Best performing model PR plot and confusion matrix.

Future works will focus on further refining the proposed approach. Additionally, further applications for the proposed optimizer will be explored.

References

[1] R. Kowalski, "Cyberbullying," in *The Routledge international handbook of human aggression*, pp. 131–142, Routledge, 2018.

[2] M. O. Lwin, B. Li, and R. P. Ang, "Stop bugging me: An examination of adolescents' protection behavior against online harassment," *Journal of adolescence*, vol. 35, no. 1, pp. 31–41, 2012.

[3] A. E. Mackey, *Evaluating the evolution of the English language as seen in TikTok slang*. PhD thesis, Wichita State University, 2023.

[4] K. Chowdhary and K. Chowdhary, "Natural language processing," *Fundamentals of artificial intelligence*, pp. 603–649, 2020.

[5] H. Jia, H. Rao, C. Wen, and S. Mirjalili, "Crayfish optimization algorithm," *Artificial Intelligence Review*, vol. 56, no. Suppl 2, pp. 1919–1979, 2023.

[6] H. Christian, M. P. Agus, and D. Suhartono, "Single document automatic text summarization using term frequency-inverse document frequency (tf-idf)," *ComTech: Computer, Mathematics and Engineering Applications*, vol. 7, no. 4, pp. 285–294, 2016.

[7] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," *Statistics and its Interface*, vol. 2, no. 3, pp. 349–360, 2009.

[8] A. Waswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," in *NIPS*, 2017.

[9] A. Radford, K. Narasimhan, T. Salimans, I. Sutskever, *et al.*, "Improving language understanding by generative pre-training," 2018.

[10] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, I. Sutskever, *et al.*, "Language models are unsupervised multitask learners," *OpenAI blog*, vol. 1, no. 8, p. 9, 2019.

[11] V. S. Tida and S. Hsu, "Universal spam detection using transfer learning of bert model," *arXiv preprint arXiv:2202.03480*, 2022.

[12] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," *arXiv preprint arXiv:1810.04805*, 2018.

[13] N. Mladenović and P. Hansen, "Variable neighborhood search," *Computers & operations research*, vol. 24, no. 11, pp. 1097–1100, 1997.

[14] S. Mirjalili and S. Mirjalili, "Genetic algorithm," *Evolutionary algorithms and neural networks: Theory and applications*, pp. 43–55, 2019.

[15] S. Mirjalili, "Sca: a sine cosine algorithm for solving optimization problems," *Knowledge-based systems*, vol. 96, pp. 120–133, 2016.

[16] X.-S. Yang and A. Hossein Gandomi, "Bat algorithm: a novel approach for global engineering optimization," *Engineering computations*, vol. 29, no. 5, pp. 464–483, 2012.

[17] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in engineering software*, vol. 95, pp. 51–67, 2016.

[18] A. A. Heidari, S. Mirjalili, H. Faris, I. Aljarah, M. Mafarja, and H. Chen, "Harris hawks optimization: Algorithm and applications," *Future generation computer systems*, vol. 97, pp. 849–872, 2019.

[19] M. Hubálovská, Š. Hubálovskỳ, and P. Trojovskỳ, "Botox optimization algorithm: A new human-based metaheuristic algorithm for solving optimization problems," *Biomimetics*, vol. 9, no. 3, p. 137, 2024.

[20] M. Zivkovic, L. Jovanovic, M. Ivanovic, N. Bacanin, I. Strumberger, and P. M. Joseph, "Xgboost hyperparameters tuning by fitness-dependent optimizer for network intrusion detection," in *Communication and intelligent systems: Proceedings of ICCIS 2021*, pp. 947–962, Springer, 2022.

[21] M. Salb, L. Jovanovic, M. Zivkovic, E. Tuba, A. Elsadai, and N. Bacanin, "Training logistic regression model by enhanced moth flame optimizer for spam email classification," in *Computer networks and inventive communication technologies: Proceedings of fifth ICCNCT 2022*, pp. 753–768, Springer, 2022.

[22] A. Petrovic, M. Antonijevic, I. Strumberger, L. Jovanovic, N. Savanovic, and S. Janicijevic, "The xgboost approach tuned by tlb metaheuristics for fraud detection," in *Proceedings of the 1st international conference on innovation in information technology and business (ICIITB 2022)*, vol. 104, p. 219, Springer Nature, 2023.

[23] N. Bacanin, M. Zivkovic, L. Jovanovic, M. Ivanovic, and T. A. Rashid, "Training a multilayer perception for modeling stock price index predictions using modified whale optimization algorithm," in *Computational Vision and Bio-Inspired Computing: Proceedings of ICCVBIC 2021*, pp. 415–430, Springer, 2022.

[24] L. Jovanovic, M. Zivkovic, M. Antonijevic, D. Jovanovic, M. Ivanovic, and H. S. Jassim, "An emperor penguin optimizer application for medical diagnostics," in *2022 IEEE Zooming Innovation in Consumer Technologies Conference (ZINC)*, pp. 191–196, IEEE, 2022.

[25] N. Bacanin, L. Jovanovic, M. Zivkovic, V. Kandasamy, M. Antonijevic, M. Deveci, and I. Strumberger, "Multivariate energy forecasting via metaheuristic tuned long-short term memory and gated recurrent unit neural networks," *Information Sciences*, vol. 642, p. 119122, 2023.

[26] L. Jovanovic, D. Jovanovic, N. Bacanin, A. Jovancai Stakic, M. Antonijevic, H. Magd, R. Thirumalaisamy, and M. Zivkovic, "Multi-step crude oil price prediction based on lstm approach tuned by salp swarm algorithm with disputation operator," *Sustainability*, vol. 14, no. 21, p. 14616, 2022.

[27] H. Christian, M. Agus, and D. Suhartono, "Single document automatic text summarization using term frequency-inverse document frequency (tf-idf)," *ComTech: Computer, Mathematics and Engineering Applications*, vol. 7, p. 285, 12 2016.

[28] H. Jia, H. Rao, C. Wen, and S. Mirjalili, "Crayfish optimization algorithm," *Artificial Intelligence Review*, vol. 1, p. 1, 09 2023.

[29] J. J. Liang, B. Y. Qu, and P. N. Suganthan, "Problem definitions and evaluation criteria for the cec 2014 special session and competition on single objective real-parameter numerical optimization," *Computational Intelligence Laboratory, Zhengzhou University, Zhengzhou China and Technical Report, Nanyang Technological University, Singapore*, vol. 635, no. 2, p. 2014, 2013.

[30] M. J. Warrens, "Five ways to look at cohen's kappa," *Journal of Psychology & Psychotherapy*, vol. 5, 2015.

Contribution of Individual Authors to the Creation of a Scientific Article (Ghostwriting Policy)

The authors equally contributed in the present research, at all stages from the formulation of the problem to the final findings and solution.

Sources of Funding for Research Presented in a Scientific Article or Scientific Article Itself

No funding was received for conducting this study.

Conflict of Interest

The authors have no conflicts of interest to declare that are relevant to the content of this article.