An Improved DE Algorithm to Optimise the Learning Process of a BERT-based Plagiarism Detection Model

Seyed Vahid Moravvej*, Seyed Jalaleddin Mousavirad†, Diego Oliva‡, Gerald Schaefer§ and Zahra Sobhaninia*
*Department of Computer Engineering, Isfahan University of Technology, Isfahan, Iran
†Computer Engineering Department, Hakim Sabzevari University, Sabzevar, Iran
‡Depto. de Innovacion Basada en la Informacion y el Conocimiento, Universidad de Guadalajara, Guadalajara, Mexico
§Department of Computer Science, Loughborough University, Loughborough, U.K.

2022 IEEE Congress on Evolutionary Computation (CEC) | 978-1-6654-6708-7/22/$31.00 ©2022 IEEE | DOI: 10.1109/CEC55065.2022.9870280

***Abstract*—Plagiarism detection is a challenging task, aiming to identify similar items in two documents. In this paper, we present a novel approach to automatic plagiarism detection that combines BERT (bidirectional encoder representations from transformers) word embedding, attention mechanism-based long short-term memory (LSTM) networks, and an improved differential evolution (DE) algorithm for weight initialisation. BERT is used to pre-train deep bidirectional representations in all layers, while the pre-trained BERT model can be fine-tuned with only one extra output layer without significant changes in architecture. Deep learning algorithms often use the random weighting method for initialisation, followed by gradient-based optimisation algorithms such as back-propagation for training, making them susceptible to getting trapped in local optima. To address this, population-based metaheuristic algorithms such as DE can be used. We propose an improved DE algorithm with a clustering-based mutation operator, where first a winning cluster of candidate solutions is identified and a new updating strategy is then applied to include new candidate solutions in the current population. The proposed DE algorithm is used in LSTM, attention mechanism, and feed-forward neural networks to yield the initial seeds for subsequent gradient-based optimisation. We compare our proposed model with conventional and population-based approaches on three datasets (SNLI, MSRP and SemEval2014) and demonstrate it to give superior plagiarism detection performance.**

***Index Terms*—Plagiarism detection, BERT, LSTM, attention mechanism, differential evolution.**

I. Introduction

With the rapid development of networked computers, access to content from anywhere, anytime, has become a sensitive topic, especially in educational and research environments. Plagiarism is considered dishonest behaviour that can damage academic integrity [1], and while sometimes it occurs subconsciously, most of the time, it is intentional. Plagiarism detection systems [2] aim to find similar items at the word, sentence, and document level.

Deep learning has attracted significant research in natural language processing [3]–[6]. [7] proposes a sentence-input method using recursive neural networks (RNNs), where the representation of each word is obtained using GloVe [8]. Then, words are entered sequentially into an RNN to generate a sentence representation, and similarity between sentences is estimated using cosine similarity. [9] employs a Siamese CNN to examine the local content of words in a sentence and to generate a representation of the relevance of a word and its vicinities. [10] presents a context-aligned RNN (CA-RNN) which combines textual information of aligned words in a sentence coupled for the inner hidden state generation. [11] develops two similarity methods for answer selection. The first combines language modelling (ELMo [12]) and BERT (bidirectional encoder representations from transformers) [13] with a transformer encoder, while the second uses two pre-trained transformer encoder models. [14] employs two attention mechanism-based LSTMs to extract sentence representations, and uses common techniques such as changing the loss function and data augmentation to address imbalanced classification.

Many deep learning models use random weight initialisation. Typically, gradient-based algorithms such as back-propagation (BP) are then employed to train the model starting from the initial values. However, gradient-based algorithms are sensitive to the initialisation conditions and can get trapped in local optima [15]. Population-based metaheuristic algorithms [16], [17] such as differential evolution (DE) [18], particle swarm optimisation [19], and human mental search [20] can be employed to address this problem. DE is a relatively simple yet effective algorithm that has been shown to perform well for various optimisation problems [21]–[24]. It comprises three steps in each iteration, mutation to generate a new solution based on scaling differences among candidate solutions, crossover to combine the generated mutation vector with the original vector, and selection to choose the best solutions for the next generation. In particular, the mutation operator plays an essential role in generating promising solutions [25], [26].

In this paper, we propose a novel plagiarism detection approach based on BERT word embedding [27], LSTMs, attention mechanisms, and a new clustering-based DE algorithm. Our approach includes two LSTM networks, one for each sentence (source and the one being checked) and a feed-forward network to predict the similarity of the two sentences.

This work is completed in 2021. On the date of publication of the final version, Seyed Jalaleddin Mousavirad is affiliated with Universidade da Beira Interior, Covilhã, Portugal, and is working on GreenStamp project.

To train the model, we generate positive and negative pairs, where a positive pair contains two identical sentences while a negative pair denotes two dissimilar sentences. Importantly, we introduce an improved DE algorithm for network weight initialisation to find a promising area to start the BP algorithm in the LSTM, attention mechanism and feed-forward networks. In our approach, the current DE population is clustered into several groups, and the best cluster selected. Then, the best candidate solution in this winner cluster is chosen as the initial vector in the mutation operator. Finally, a novel updating strategy is applied to generate candidate solutions in the current population. We evaluate our model on three benchmark datasets, SNLI, MSRP and SemEval2014, and demonstrate our algorithm to be superior to other methods that use random weighting as well as other metaheuristics.

The remainder of the paper is organised as follows. Section II introduces some related background, while Section III details our proposed approach. Section IV reports the experimental results, and Section V concludes the paper.

II. Background

A. Long Short-Term Memory Networks

LSTM networks [28] allow to learn long dependencies to solve the vanishing gradient problem. Each LSTM network includes an input gate i_t, a forget gate f_t, an output gate o_t, and a cell input c_t at each time step t, which are updated as

$$i_t = \sigma(W_i x_t + U_i h_{t-1} + b_i), \tag{1}$$

$$f_t = \sigma(W_f x_t + U_f h_{t-1} + b_f), \tag{2}$$

$$o_t = \sigma(W_o x_t + U_o h_{t-1} + b_o), \tag{3}$$

and

$$c_t = f_t c_{t-1} + i_t \tanh(W_j x_t + U_j h_{t-1} + b_j), \tag{4}$$

respectively, with

$$h_t = o_t \tanh(c_t), \tag{5}$$

where x_t and h_t are input and hidden vectors of size d_x and d_h, respectively, and $W \in \mathbb{R}^{d_h \times d_x}$, $U \in \mathbb{R}^{d_h \times d_h}$, and $b \in \mathbb{R}^{d_h}$ are the parameters that need to be learned.

While an LSTM network only processes inputs from one side, in some tasks including ours here, it is necessary to process the input text from both sides. A bi-directional LSTM (BLSTM) allows such processing, and produces two latent vectors, $\overrightarrow{h_t}$ and $\overleftarrow{h_t}$. The combination of the two, i.e. $h_t = [\overrightarrow{h_t}, \overleftarrow{h_t}]$, then forms a single hidden vector.

Although an LSTM processes long sequences, it assigns equal importance to all units, leading to network confusion. An attention mechanism can be introduced to avoid this problem. For each hidden vector h_t calculated at time step t, a weight α_t is assigned, and the final hidden vector is computed as

$$h = \sum_{t=1}^{T} \alpha_t h_t, \tag{6}$$

where T is the number of inputs.

B. Differential Evolution

DE [29] is a population-based algorithm based on three operators, mutation, crossover, and selection. The mutation operator creates a mutant vector for a D-dimensional candidate solution as

$$\overrightarrow{v}_{i,g} = \overrightarrow{x}_{r_1,g} + F(\overrightarrow{x}_{r_2,g} - \overrightarrow{x}_{r_3,g}), \tag{7}$$

where $\overrightarrow{x}_{r_1,g}$, $\overrightarrow{x}_{r_2,g}$ and $\overrightarrow{x}_{r_3,g}$ are three random candidate solutions selected from the current population, and F is a control parameter called the factor scaling which regulates the amplification of the difference vector.

Binomial crossover, a popular form of crossover, shuffles the mutant and parent vectors as

$$u_{i,j,g} = \begin{cases} v_{i,j,g} & \text{if } rand(0,1) \leq CR \text{ or } j = j_{rand} \\ x_{i,j,g} & \text{otherwise} \end{cases}, \tag{8}$$

where *CR* is the crossover rate and j_{rand} is an integer selected from $\{1, 2, ..., D\}$ specifying the fraction of $v_{i,g}$ inherited from $u_{i,g}$.

To determine the new individuals for the next population, a selection strategy is used to choose candidate solutions based on an objective function. For a minimisation problem, this operates as

$$\overrightarrow{x}_{i,g+1} = \begin{cases} \overrightarrow{u}_{i,g} & \text{if } f(\overrightarrow{u}_{i,g}) < f(\overrightarrow{x}_{i,g}) \\ \overrightarrow{x}_{i,g} & \text{otherwise} \end{cases}, \tag{9}$$

where f signifies the objective function.

III. Proposed Model

An overview of our proposed plagiarism detection model is illustrated in Fig. 1. In the following, we describe its components in detail.

A. Pre-processing

The purpose of pre-processing is to remove trivial words from a text in order to reduce the computational load and enhance the efficacy of the model. In our approach, we perform stop word removal to discard words without semantic information (such as 'and','or', etc.) which appear frequently and may cause the importance of other words to be overlooked, and stemming, which turns words into their root form (for example, the root of 'flowers' is 'flower').

B. BERT-based Word Embedding

Word embedding is the distributed supposition of word representation and extracts vectors for natural language words so that computers can understand them. The semantic relevance among words can be calculated based on the similarity between the vectors. Word embedding is generally used in NLP applications such as Word2Vec [30] and Skip-Gram. Bidirectional encoder representations from transformers (BERT) [13] is a pre-trained language model that has shown excellent performance for NLP tasks. BERT uses a masked language model and next sentence prediction to represent words and sentences, respectively, and is able to deal with

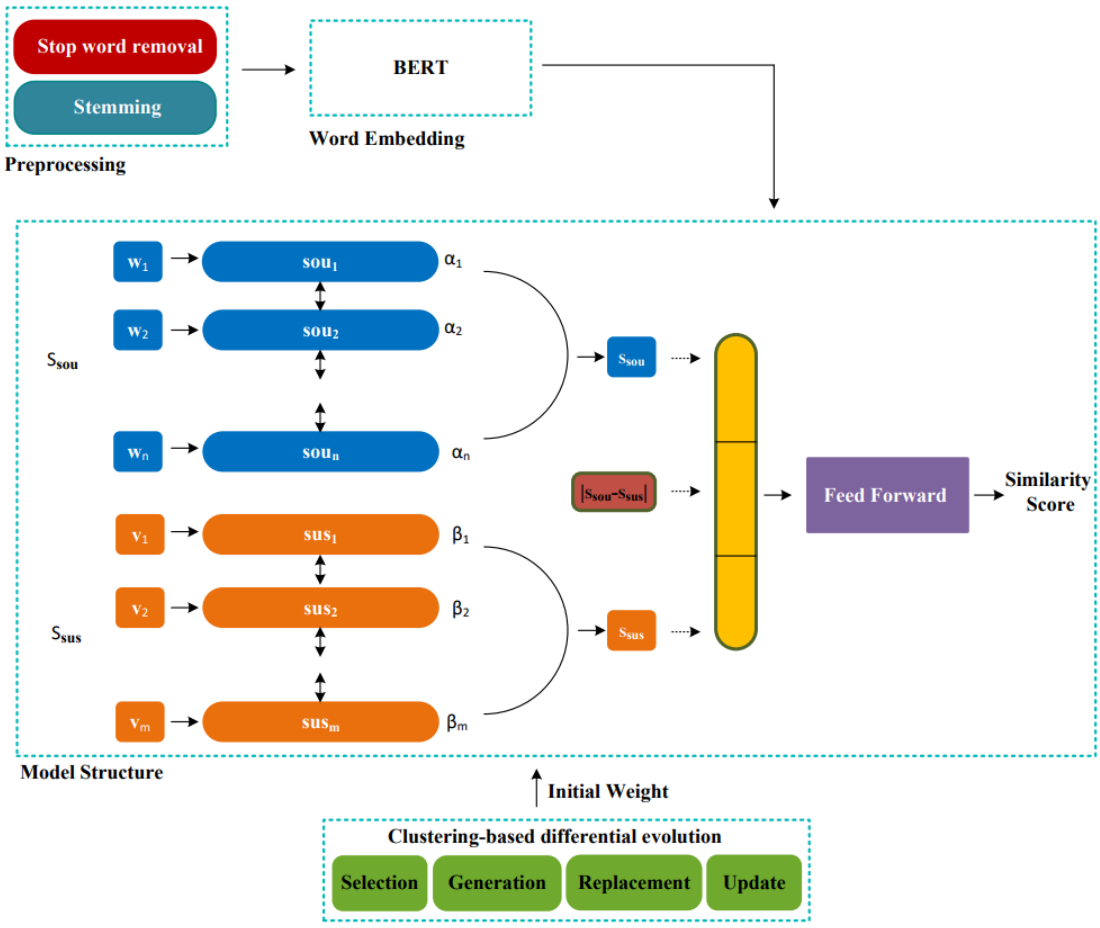

Fig. 1: Overall architecture of the proposed model.

deep semantic understanding and sentence relationships [31]. In our approach, we thus use BERT word embedding as model input.

C. Model Structure

In our plagiarism detection model, we employ LSTM and feed-forward as the main networks, attention mechanisms, and an improved DE algorithm for the initialisation of the learnable weights.

Let $S_{sou} = \{w_1, w_1, ..., w_n\}$ and $S_{sus} = \{v_1, v_1, ..., v_m\}$ be the source and suspicious sentence, respectively, where w_i and v_i are the i-th words in the source and suspicious sentence. S_{sou} and S_{sus} are limited to n and m words, respectively. S_{sou} and S_{sus} are fed into an LSTM network separately. The embedding of these sentences is calculated through the attention mechanism as

$$S_{sou} = \sum_{i=1}^{n} \alpha_i h_{sou_i}, \tag{10}$$

and

$$S_{sus} = \sum_{i=1}^{m} \beta_i h_{sus_i}, \tag{11}$$

where $h_{sou_i} = [\overrightarrow{h}_{sou_i}, \overleftarrow{h}_{sou_i}]$ and $h_{sus_i} = [\overrightarrow{h}_{sus_i}, \overleftarrow{h}_{sus_i}]$ show the i-th hidden vectors in the BLSTM, and $\alpha_i, \beta_i \in [0, 1]$ are the i-th attention weights for each unit calculated as

$$\alpha_i = \frac{e^{u_i}}{\sum_{j=1}^{n} e^{u_j}}, \tag{12}$$

and

$$\beta_i = \frac{e^{v_i}}{\sum_{j=1}^{m} e^{v_j}}, \tag{13}$$

with

$$u_i = \tanh(W_u h_{sou_i} + b_u), \tag{14}$$

and

$$v_i = \tanh(W_v h_{sus_i} + b_v), \tag{15}$$

where W_u, W_v, b_u and b_v are learning parameters for the attention mechanisms.

The input of the feed-forward network is s_{sou}, s_{sus}, and $|s_{sou} - s_{sus}|$ as shown in Fig. 1. Positive and negative pairs are employed to train the model, where for positive pairs the two sentences S_{sou} and S_{sus} are copies with an expected model output of 1, while for negative pairs S_{sou} and S'_{sus} are different sentences with an expected model output of 0.

D. Weight Optimisation

We introduce a novel DE algorithm for initialisation of all model weights, including LSTM, attention mechanism, feed-forward networks. We arrange all the weights into a vector as illustrated in Fig. 2 which we optimise based on the objective function defined as

$$f = \sum_{i=1}^{N} (y_i - \tilde{y}_i)^2, \tag{16}$$

where N indicates the total number of training samples including positive and negative pairs, and y_i and $\tilde{y}_i$ are the i-th target and model output, respectively.

In our enhanced DE algorithm, we modify the mutation operator to achieve further improvement inspired by [32]. In particular, we identify a promising area in search space based on clustering the current population (k-means clustering). The clustering divides the population into k areas in search space, with k chosen randomly chosen from [2,$\sqrt{N}$] [33], while the cluster with the lowest mean objective function (for a minimisation problem) is selected as the promising area.

A new clustering-based mutation operator is then applied as

$$\overrightarrow{v^{clu}}_{i,g} = \overrightarrow{win}_g + F(\vec{x}_{r_1,g} - \vec{x}_{r_2,g}), \tag{17}$$

where $\vec{x}_{r_1,g}$ and $\vec{x}_{r_2,g}$ are two randomly selected candidate solutions and $\overrightarrow{win}_g$ is the best candidate solution in the promising area. Note that $\overrightarrow{win}_g$ may not be the best solution in the current population. The clustering-based mutation operation is repeated M times. Then, the population is updated according to the generic population-based algorithm (GPBA) [34] using the following rules:

- *Selection:* k candidate solutions are generated randomly and considered initial seeds of the k-means algorithm;
- *Generation:* M candidate solutions are generated using clustering-based mutation as set v^{clu};
- *Replacement:* M candidate solutions are randomly selected from the current population as set B;
- *Update:* the best M candidate solutions are selected from $v^{clu} \cup B$ as set B'. Finally, the new population is obtained as $(P - B) \cup B'$.

Our clustering-based DE algorithm is given, in pseudo-code form, in Algorithm 1.

Algorithm 1: Pseudo-code of proposed DE algorithm.

Input: D: dimensionality of candidate solution; $MaxFES$: maximum number of function evaluations, F: scalling factor, CR: crossover probability

Initialise population $P = (\vec{x}_1, \vec{x}_2, \vec{x}_3, ..., \vec{x}_{NP})$ randomly
Calculate objective function of every solution in P
while $FES < MaxFES$ **do**
  for $i = 1$ *to* NP **do**
    Choose individuals $\vec{x}_{r_1,g}$, $\vec{x}_{r_2,g}$, $\vec{x}_{r_3,g}$ from P randomly with $\vec{x}_{r_1,g} \neq \vec{x}_{r_2,g} \neq \vec{x}_{r_3,g}$
    $\vec{v}_{i,g} = \vec{x}_{r_1,g} + F(\vec{x}_{r_2,g} - \vec{x}_{r_3,g})$
    Select j_{rand} as random number in interval [0,1]
    for $j = 0$ *to* D **do**
      if $rand(0,1) \leq CR$ *or* $j = j_{rand}$ **then**
        $u_{i,j,g} = v_{i,j,g}$
      else
        $u_{i,j,g} = x_{i,j,g}$
    if $f(\vec{u}_{i,g}) < f(\vec{x}_{i,g})$ **then**
      $\vec{x}_{i,g+1} = \vec{u}_{i,g}$
    else
      $\vec{x}_{i,g+1} = \vec{x}_{i,g}$
  Select k as random number in interval [2,$\sqrt{N}$]
  Cluster P into k clusters
  Compute mean objective function of every cluster
  $\overrightarrow{win}_g$ = the best solution in the winner cluster
  for $j = 1$ *to* M **do**
    Choose $\vec{x}_{r_1,g}$, $\vec{x}_{r_2,g}$ from P randomly with $\vec{x}_{r_1,g} \neq \vec{x}_{r_2,g}$
    $\overrightarrow{v^{clu}}_{i,g} = \overrightarrow{win}_g + F(\vec{x}_{r_1,g} - \vec{x}_{r_2,g})$
  Choose M candidate solutions randomly from P as set B
  Select the best M solutions from $v^{clu} \cup B$ as set B'
  Create new population as $(P - B) \cup B'$

IV. Experimental Results

A. Datasets

We use three imbalanced benchmark datasets, including, namely:

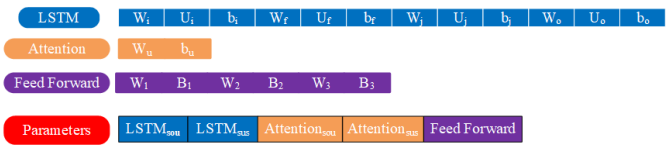

Fig. 2: All model weights are encoded into a weight vector.

- *SNLI:* the Stanford Natural Language Inference (SNLI) corpus [35] is a dataset with three labels (entailment, contradiction, and semantic independence) with a total of 570,152 pairs of sentences. Of these, 550,152, 1,000 and 1,000 sentence pairs are used for training, validation and testing, respectively.
- *MSRP:* the Microsoft Research Paraphrase Corpus (MSRP) [36] is a paraphrasing dataset of 5,801 pairs of sentences labelled by experts. The training and test datasets comprise 4,076 and 1,725 samples, respectively. About 67% of the pairs are paraphrased.
- *SemEval2014:* Semantic Evaluation (SemEval) [37] is a vital evaluator for STS work. The sentences from the Com-positional Knowledge (SICK) dataset [38] are used to evaluate the semantic relevance of sentences. It consists

of 1,000 pairs of English sentences, divided into 4,500, 500 and 5,000 sentence pairs for training, validation and testing, respectively. Each pair is labelled with a number in the interval $[1, 5]$ with sentences without similarity labelled 1 and the most similar sentences labelled 5. In our work, label 1 represents one class, and the remaining labels the other class.

B. Results

As evaluation criteria, we use accuracy, recall, precision, F-measure, and G-means [39]. Since the datasets are imbalanced, F-measure and G-means are generally more useful for comparison [40].

To put the obtained results into context, we also use seven deep learning-based approaches, namely RNN [7], Siamese CNN+LSTM [9], CA-RNN [10], AttSiaBiLSTM [37], LSTM+FNN+attention [41], CETE [11], and STS-AM [14], while for our proposed method we also provide results based on random weight initialisation. The obtained results are given in Table I, II, and III for SNLI, MSRP, and SemEval2014, respectively. As we can see from there, our proposed approach gives the best results on all three datasets and for all performance measures, demonstrating its usefulness and superiority over the other methods.

TABLE I: Plagiarism detection results on SNLI dataset for various deep learning models.

	accuracy	recall	precision	F-measure	G-means
RNN [7]	0.687	0.594	0.540	0.566	0.661
Siamese CNN+LSTM [9]	0.850	0.763	0.792	0.777	0.826
CA-RNN [10]	0.790	0.667	0.704	0.685	0.754
AttSiaBiLSTM [37]	0.695	0.569	0.554	0.561	0.658
LSTM+FNN+attention [41]	0.818	0.781	0.715	0.747	0.809
CETE [11]	0.874	0.855	0.795	0.824	0.870
STS-AM [14]	0.756	0.625	0.650	0.637	0.718
Proposed (random weights)	0.808	0.777	0.698	0.735	0.801
Proposed	**0.919**	**0.892**	**0.874**	**0.883**	**0.912**

TABLE II: Plagiarism detection results on MSRP dataset for various deep learning models.

	accuracy	recall	precision	F-measure	G-means
RNN [7]	0.853	0.922	0.866	0.893	0.812
Siamese CNN+LSTM [9]	0.863	0.916	0.882	0.899	0.833
CA-RNN [10]	0.880	0.928	0.896	0.912	0.854
AttSiaBiLSTM [37]	0.874	0.927	0.889	0.908	0.845
LSTM+FNN+attention [41]	0.889	0.917	0.916	0.916	0.873
CETE [11]	0.916	0.949	0.926	0.937	0.898
STS-AM [14]	0.899	0.940	0.910	0.925	0.876
Proposed (random weights)	0.875	0.908	0.905	0.906	0.858
Proposed	**0.924**	**0.952**	**0.935**	**0.943**	**0.910**

TABLE III: Plagiarism detection results on SemEval2014 dataset for various deep learning models.

	accuracy	recall	precision	F-measure	G-means
RNN [7]	0.809	0.822	0.963	0.887	0.750
Siamese CNN+LSTM [9]	0.775	0.787	0.958	0.864	0.720
CA-RNN [10]	0.811	0.826	0.961	0.888	0.742
AttSiaBiLSTM [37]	0.799	0.816	0.957	0.881	0.720
LSTM+FNN+attention [41]	0.733	0.746	0.949	0.835	0.670
CETE [11]	0.854	0.868	0.969	0.916	0.791
STS-AM [14]	0.823	0.834	0.966	0.895	0.768
Proposed (random weights)	0.839	0.855	0.964	0.906	0.766
Proposed	**0.865**	**0.876**	**0.973**	**0.922**	**0.814**

To further evaluate the performance of our proposed DE algorithm, we compare our algorithm with standard DE as well as other metaheuristic algorithms, namely firefly algorithm (FA) [42], bat algorithm (BA) [43], cuckoo optimisation algorithm (COA) [44], artificial bee colony (ABC) [45], grey wolf optimisation (GWO) [46], whale optimisation algorithm (WOA) [47], and salp swarm algorithm (SSA) [48]. For this, all parts of the plagiarism detection model remain the same, and only the weight initialisation is replaced by one of the algorithms. For all algorithms, the population size and maximum number of function evaluations are set to 200 and 3,000, respectively. The results are reported in Tables IV, V and VI for SNLI, MSRO and SemEvel2014, respectively. As is evident from the obtained results, our improved DE clearly outperforms the other methods on all three datasets and for all evaluation measures.

TABLE IV: Plagiarism detection results on SNLI dataset for various metaheuristics.

	accuracy	recall	precision	F-measure	G-means
DE	0.893	0.885	0.817	0.850	0.891
FA	0.860	0.799	0.794	0.796	0.844
BA	0.872	0.846	0.794	0.819	0.866
COA	0.856	0.807	0.780	0.793	0.843
ABC	0.881	0.865	0.803	0.833	0.877
GWO	0.837	0.775	0.756	0.765	0.821
WOA	0.879	0.828	0.821	0.824	0.866
SSA	0.859	0.816	0.782	0.799	0.848
Proposed	**0.919**	**0.892**	**0.874**	**0.883**	**0.912**

TABLE V: Plagiarism detection results on MSRP dataset for various metaheuristics.

	accuracy	recall	precision	F-measure	G-means
DE	0.909	0.947	0.919	0.933	0.889
FA	0.873	0.922	0.891	0.906	0.846
BA	0.894	0.932	0.911	0.921	0.874
A	0.886	0.924	0.907	0.915	0.866
ABC	0.883	0.934	0.895	0.914	0.856
GWO	0.868	0.914	0.891	0.902	0.844
WOA	0.885	0.926	0.904	0.915	0.863
SSA	0.868	0.911	0.893	0.902	0.845
Proposed	**0.924**	**0.952**	**0.935**	**0.943**	**0.910**

TABLE VI: Plagiarism detection results on SemEval2014 dataset for various metaheuristics.

	accuracy	recall	precision	F-measure	G-means
DE	0.858	0.869	0.971	0.917	0.804
FA	0.848	0.861	0.968	0.911	0.788
BA	0.854	0.866	0.970	0.915	0.796
COA	0.850	0.863	0.968	0.912	0.787
ABC	0.845	0.859	0.966	0.909	0.778
GWO	0.824	0.839	0.962	0.896	0.749
WOA	0.817	0.831	0.961	0.891	0.746
SSA	0.795	0.808	0.959	0.877	0.731
Proposed	**0.865**	**0.876**	**0.973**	**0.922**	**0.814**

V. CONCLUSIONS

In this paper, we have presented an plagiarism detection approach using BERT word embedding and attention-based

LSTM feed-forward networks, enhanced by an improved differential evolution algorithm to initialise the model weights. In our DE algorithm, the population is divided into clusters to identify a promising area in search space, and a new updating strategy is then employed to generate a new population. The obtained experimental results show that our proposed model can improve upon deep models that use random weights, while the proposed DE algorithm shows excellent performance compared to other other population-based algorithms. In future work, we intend to extend our approach to other types of deep learning approaches, while a multi-objective variant of the algorithm is also under investigation.

REFERENCES

[1] D. Gupta *et al.*, "Study on extrinsic text plagiarism detection techniques and tools." *Journal of Engineering Science & Technology Review*, vol. 9, no. 5, 2016.

[2] M. Sabeeh and F. Khaled, "Plagiarism detection methods and tools: An overview," *Iraqi Journal of Science*, pp. 2771–2783, 2021.

[3] S. V. Moravvej, A. Mirzaei, and M. Safayani, "Biomedical text summarization using conditional generative adversarial network," *arXiv preprint arXiv:2110.11870*, 2021.

[4] S. V. Moravvej, M. J. M. Kahaki, M. S. Sartakhti, and A. Mirzaei, "A method based on attention mechanism using bidirectional long-short term memory (blstm) for question answering," in *29th Iranian Conference on Electrical Engineering*, 2021, pp. 460–464.

[5] S. Moravvej, M. Maleki Kahaki, M. Salimi Sartakhti, and M. Joodaki, "Efficient GAN-based method for extractive summarization," *Journal of Electrical and Computer Engineering Innovations*, 2021.

[6] M. S. Sartakhti, M. J. M. Kahaki, S. V. Moravvej, M. javadi Joortani, and A. Bagheri, "Persian language model based on BiLSTM model on COVID-19 corpus," in *5th International Conference on Pattern Recognition and Image Analysis*, 2021, pp. 1–5.

[7] A. Sanborn and J. Skryzalin, "Deep learning for semantic similarity," in *CS224d: Deep Learning for Natural Language Processing*. Stanford University Stanford, CA, USA, 2015.

[8] J. Pennington, R. Socher, and C. D. Manning, "GloVe: Global vectors for word representation," in *Conference on Empirical Methods in Natural Language Processing*, 2014, pp. 1532–1543.

[9] E. L. Pontes, S. Huet, A. C. Linhares, and J.-M. Torres-Moreno, "Predicting the semantic textual similarity with Siamese CNN and LSTM," *arXiv preprint arXiv:1810.10641*, 2018.

[10] Q. Chen, Q. Hu, J. X. Huang, and L. He, "CA-RNN: using context-aligned recurrent neural networks for modeling sentence similarity," in *AAAI Conference on Artificial Intelligence*, vol. 32, no. 1, 2018.

[11] M. T. R. Laskar, X. Huang, and E. Hoque, "Contextualized embeddings based transformer encoder for sentence similarity modeling in answer selection task," in *12th Language Resources and Evaluation Conference*, 2020, pp. 5505–5514.

[12] M. E. Peters, M. Neumann, M. Iyyer, M. Gardner, C. Clark, K. Lee, and L. Zettlemoyer, "Deep contextualized word representations," *arXiv preprint arXiv:1802.05365*, 2018.

[13] 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.

[14] S. V. Moravvej, M. Joodaki, M. J. M. Kahaki, and M. S. Sartakhti, "A method based on an attention mechanism to measure the similarity of two sentences," in *7th International Conference on Web Research*, 2021, pp. 238–242.

[15] S. V. Moravvej, S. J. Mousavirad, M. H. Moghadam, and M. Saadatmand, "An lstm-based plagiarism detection via attention mechanism and a population-based approach for pre-training parameters with imbalanced classes," in *International Conference on Neural Information Processing*. Springer, 2021, pp. 690–701.

[16] S. Vakilian, S. V. Moravvej, and A. Fanian, "Using the artificial bee colony (ABC) algorithm in collaboration with the FOG nodes in the internet of things three-layer architecture," in *29th Iranian Conference on Electrical Engineering*, 2021, pp. 509–513.

[17] ——, "Using the cuckoo algorithm to optimizing the response time and energy consumption cost of fog nodes by considering collaboration in the FOG layer," in *5th International Conference on Internet of Things and Applications*, 2021, pp. 1–5.

[18] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[19] J. Kennedy and R. Eberhart, "Particle swarm optimization," in *International Conference on Neural Networks*, vol. 4, 1995, pp. 1942–1948.

[20] S. J. Mousavirad and H. Ebrahimpour-Komleh, "Human mental search: a new population-based metaheuristic optimization algorithm," *Applied Intelligence*, vol. 47, no. 3, pp. 850–887, 2017.

[21] S. J. Mousavirad and S. Rahnamayan, "A novel center-based differential evolution algorithm," in *IEEE Congress on Evolutionary Computation*, 2020, pp. 1–8.

[22] ——, "Evolving feedforward neural networks using a quasi-opposition-based differential evolution for data classification," in *IEEE Symposium Series on Computational Intelligence*, 2020, pp. 2320–2326.

[23] S. J. Mousavirad, D. Zabihzadeh, D. Oliva, M. Perez-Cisneros, and G. Schaefer, "A grouping differential evolution algorithm boosted by attraction and repulsion strategies for Masi entropy-based multi-level image segmentation," *Entropy*, vol. 24, no. 1, p. 8, 2021.

[24] M. H. Moghadam, M. Borg, and S. J. Mousavirad, "Deeper at the SBST 2021 tool competition: ADAS testing using multi-objective search," in *14th IEEE/ACM International Workshop on Search-Based Software Testing*, 2021, pp. 40–41.

[25] S. J. Mousavirad and S. Rahnamayan, "Differential evolution algorithm based on a competition scheme," in *14th International Conference on Computer Science & Education*, 2019, pp. 929–934.

[26] D. Bajer, "Adaptive k-tournament mutation scheme for differential evolution," *Applied Soft Computing*, vol. 85, p. 105776, 2019.

[27] Q. Wang, P. Liu, Z. Zhu, H. Yin, Q. Zhang, and L. Zhang, "A text abstraction summary model based on bert word embedding and reinforcement learning," *Applied Sciences*, vol. 9, no. 21, p. 4701, 2019.

[28] S. Hochreiter and J. Schmidhuber, "Long short-term memory," *Neural Computation*, vol. 9, no. 8, pp. 1735–1780, 1997.

[29] R. Storn and K. Price, "Differential evolution–a simple and efficient heuristic for global optimization over continuous spaces," *Journal of Global Optimization*, vol. 11, no. 4, pp. 341–359, 1997.

[30] K. W. Church, "Word2vec," *Natural Language Engineering*, vol. 23, no. 1, pp. 155–162, 2017.

[31] D. Jiang and J. He, "Tree framework with bert word embedding for the recognition of chinese implicit discourse relations," *IEEE Access*, vol. 8, pp. 162 004–162 011, 2020.

[32] S. J. Mousavirad, G. Schaefer, I. Korovin, M. H. Moghadam, M. Saadatmand, and M. Pedram, "An enhanced differential evolution algorithm using a novel clustering-based mutation operator," in *IEEE International Conference on Systems, Man, and Cybernetics*, 2021, pp. 176–181.

[33] S. J. Mousavirad, G. Schaefer, I. Korovin, and D. Oliva, "RDE-OP: A region-based differential evolution algorithm incorporation opposition-based learning for optimising the learning process of multi-layer neural networks," in *International Conference on the Applications of Evolutionary Computation*, 2021, pp. 407–420.

[34] K. Deb, "A population-based algorithm-generator for real-parameter optimization," *Soft Computing*, vol. 9, no. 4, pp. 236–253, 2005.

[35] S. R. Bowman, G. Angeli, C. Potts, and C. D. Manning, "A large annotated corpus for learning natural language inference," *arXiv preprint arXiv:1508.05326*, 2015.

[36] J. Chen, Q. Chen, X. Liu, H. Yang, D. Lu, and B. Tang, "The BQ corpus: A large-scale domain-specific Chinese corpus for sentence semantic equivalence identification," in *Conference on Empirical Methods in Natural Language Processing*, 2018, pp. 4946–4951.

[37] W. Bao, W. Bao, J. Du, Y. Yang, and X. Zhao, "Attentive Siamese LSTM network for semantic textual similarity measure," in *2018 International Conference on Asian Language Processing*, 2018, pp. 312–317.

[38] M. Marelli, S. Menini, M. Baroni, L. Bentivogli, R. Bernardi, R. Zamparelli *et al.*, "A SICK cure for the evaluation of compositional distributional semantic models." in *LREC*, 2014, pp. 216–223.

[39] B.-G. Hu and W.-M. Dong, "A study on cost behaviors of binary classification measures in class-imbalanced problems," *arXiv preprint arXiv:1403.7100*, 2014.

[40] M. Bekkar, H. K. Djemaa, and T. A. Alitouche, "Evaluation measures for models assessment over imbalanced data sets," *J Inf Eng Appl*, vol. 3, no. 10, 2013.

[41] Z. Chi and B. Zhang, "A sentence similarity estimation method based on improved Siamese network," *Journal of Intelligent Learning Systems and Applications*, vol. 10, no. 4, pp. 121–134, 2018.
[42] X.-S. Yang, "Firefly algorithm, stochastic test functions and design optimisation," *International Journal of Bio-Inspired Computation*, vol. 2, no. 2, pp. 78–84, 2010.
[43] ——, "A new metaheuristic bat-inspired algorithm," in *Nature Inspired Cooperative Strategies for Optimization*. Springer, 2010, pp. 65–74.
[44] X.-S. Yang and S. Deb, "Cuckoo search via Lévy flights," in *World Congress on Nature & Biologically Inspired Computing*, 2009, pp. 210–214.
[45] D. Karaboga and B. Basturk, "A powerful and efficient algorithm for numerical function optimization: artificial bee colony (ABC) algorithm," *Journal of Global Optimization*, vol. 39, no. 3, pp. 459–471, 2007.
[46] S. Mirjalili, S. M. Mirjalili, and A. Lewis, "Grey wolf optimizer," *Advances in Engineering Software*, vol. 69, pp. 46–61, 2014.
[47] S. Mirjalili and A. Lewis, "The whale optimization algorithm," *Advances in Engineering Software*, vol. 95, pp. 51–67, 2016.
[48] D. Bairathi and D. Gopalani, "Salp swarm algorithm (SSA) for training feed-forward neural networks," in *Soft Computing for Problem Solving*. Springer, 2019, pp. 521–534.