

Enhancing plagiarism detection methodology with the DQN algorithm on an improved differential evolution foundation

Yunbo Feng^{1,3}, and Pengbo Guo^{2,4,5}

¹Hebei University of Engineering, Handan, China

²University of Leeds, Leeds, The United Kingdom

³15028053132@163.com

⁴chenxuan20220222@163.com

⁵corresponding author

Abstract. Plagiarism detection has become increasingly crucial in real-world applications, demanding precise identification of content similarity. This paper introduces a novel plagiarism detection approach. Building upon LSTM as the foundation, it employs an enhanced DE (Differential Evolution) algorithm and reinforces learning with the DQN algorithm for sample classification and training. Throughout the training process, gradual parameter adjustments are made with the aim of improving the model's efficiency and accuracy.

Keywords: LSTM, differential evolution, plagiarism detection, DQN.

1. Introduction

With the widespread availability of extensive online information and the proliferation of powerful search engines, plagiarism has emerged as a substantial concern across diverse domains, particularly in the realm of education. Plagiarism can manifest both as a deliberate act and as an inadvertent oversight. The techniques employed for plagiarism detection also find applications beyond the educational sphere, extending into the field of information retrieval.

Numerous strategies have been formulated to tackle the issue of plagiarism detection, with text distance-based methods being among them. These methods are designed to gauge the semantic similarity between textual components, thereby discerning potential instances of plagiarism.

In recent years, deep learning approaches have gained popularity owing to their prowess in automated feature extraction. Nonetheless, methods grounded in machine learning continue to exhibit certain limitations. These drawbacks encompass a tendency to overlook semantic nuances and a limited flexibility in feature extraction, often relying on manually crafted rules. To surmount these challenges, meta-heuristic techniques, exemplified by Differential Evolution (DE), can be leveraged to enhance the learning process through optimization.

In existing research, the solution proposed by Moravvej et al. stands out as a pioneering approach. They have introduced an innovative attention mechanism-based LSTM model for plagiarism detection, incorporating BERT word embeddings and a clustering-based DE (Differential Evolution) algorithm.

Their utilization of BERT word embeddings, a recently developed model applicable to multiple languages, represents a novel contribution to the field of plagiarism detection. Furthermore, they have devised a new DE algorithm to initialize weight values and have employed focal loss-based training to address issues related to imbalanced classification by learning from minority class instances. Their approach has exhibited superior performance compared to alternative methods and has been rigorously evaluated using three reference datasets: MSRP, SNLI, and SemEval2014.

Moravvej et al. employed a pretraining method based on clustering and an improved differential evolution algorithm for their network. Additionally, they incorporated the use of Focal Loss to mitigate the class imbalance issue. However, the authors themselves acknowledged that Focal Loss has its limitations and may not represent the optimal choice. Class imbalance presents a significant hurdle to the accurate identification of plagiarism instances, as the number of negative samples significantly outweighs the positive ones [1].

Given a dataset consisting of positive and negative pairs, where positive pairs represent instances of plagiarism and negative pairs represent non-plagiarism instances, our goal is to train a model capable of accurately classifying new instances of plagiarism. However, due to class imbalance, the model often leans towards the majority class, leading to a decrease in plagiarism detection performance.

In this study, we introduce a novel approach centered around the Deep Q-Network (DQN) algorithm to address the challenge of class imbalance in plagiarism detection. Our approach harnesses the power of the DQN algorithm, which combines reinforcement learning with deep neural networks, to optimize the classification process. We provide a comprehensive methodology and workflow that leverage the DQN framework to enhance the performance of plagiarism detection models.

The primary contributions of this study are as follows:

1. Intervention with the use of the DQN algorithm in classifying samples helps to mitigate the class imbalance issue, reducing the disparity between positive and negative samples and resulting in a more balanced sample classification.
2. Enhanced model efficiency in the plagiarism detection process leads to improved precision and accuracy.

2. Related work

2.1. Long Short-Term Memory

In 2015, Google introduced a groundbreaking variant of recurrent neural network called the Long Short-Term Memory (LSTM) network, initially proposed by Hochreiter and Schmidhuber in 1997 [2]. This innovation revolutionized the capability to process longer sequences of speech and significantly improved the accuracy of speech recognition.

Recurrent neural networks (RNNs) represent a category of neural networks employed for tasks involving sequential data, such as speech recognition and video processing [3]. They incorporate a feedback loop within the network, where output is fed back along with the next input. The hidden layers in these networks are dynamically determined to capture sequential dependencies in the data. The hidden layers are determined as

$$h_t = H(W_{hh}h_{t-1} + W_{xh}x_t + b_h),$$

and the output layer as

$$y_t = H(W_{hy}h_{t-1} + b_y),$$

When W and b are the weight matrix and bias term, and H is the recurrent hidden layer function.

Long Short-Term Memory (LSTM) is a specialized type of recurrent neural network (RNN) primarily designed to address the challenges of gradient vanishing and exploding during training on long sequences. In essence, compared to regular RNNs, LSTM demonstrates superior performance in longer sequence contexts[4]. The structure of an LSTM cell is illustrated in Figure 2.1.

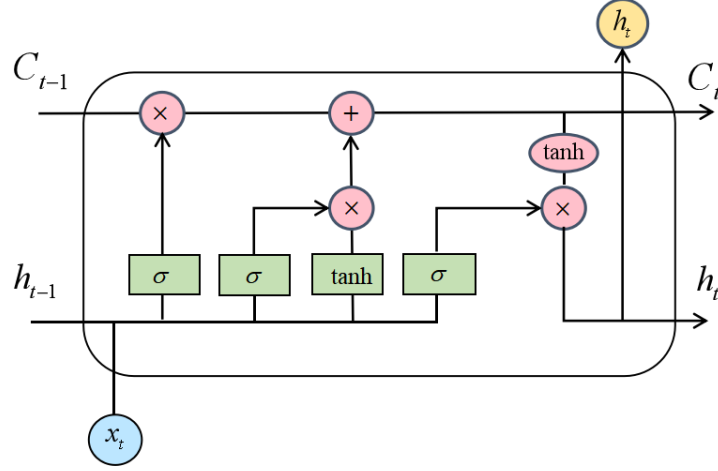


Figure 2.1. LSTM Cell.

By using the hidden layer as a memory unit, a long short-term memory network [5] is a specific kind of RNN that can handle correlations in both short- and long-term sequences. The input gate i_t , the forget gate f_t , and the output gate o_t make up an LSTM memory cell. The memory cell's current state is h_t , the data x_t is each gate's input, and its prior state is h_{t-1} .

The input gate is updated as

$$i_t = \sigma(W_{ix}x_t + U_{ih}h_{t-1} + b_i),$$

the forget gate as

$$f_t = \sigma(W_{fx}x_t + U_{fh}h_{t-1} + b_f),$$

and the output gate as

$$o_t = \sigma(W_{ox}x_t + U_{oh}h_{t-1} + b_o),$$

where σ is an activation function. The state update is then calculated as

$$h_t = o_t \tanh(c_t),$$

with

$$c_t = f_t C_{t-1} + i_t \tanh(W_{cx}x_t + U_{ch}h_{t-1} + b_c).$$

An LSTM network is extended by a bi-directional LSTM (BLSTM) to handle input from both directions. Since the user may create a new sentence by rearranging the words in the source sentence, this can be helpful in the detection of plagiarism.

2.2. Differential Evolution

A population-based optimization approach known as differential evolution was introduced by Price et al.[6], has demonstrated its effectiveness in solving a diverse range of optimization problems[7,8]. Differential evolution commences with an initial population, comprised of three fundamental operations: mutation, crossover, and selection, often sampled from a uniform distribution.

In this process, the crossover operation amalgamates the vectors of the mutant and the target, with the binomial crossover being a commonly utilized operator. Subsequently, the selection operator is employed to choose the superior solution from the trial and target vectors.

Using this algorithm, new solutions can be generated, and the performance of the algorithm can be gradually improved by evaluating and comparing these solutions. It has been proven to achieve excellent optimization results in multiple domains. As a result, it has become a popular algorithm choice and is widely used in engineering, science, and other fields.

2.3. Deep Q-Network

Deep Q-Network (DQN) is a reinforcement learning algorithm originally introduced by Google DeepMind in 2015. It represents a variation of Q-learning, a well-known reinforcement learning

algorithm that relies on a table to store values for state-action pairs. In contrast, DQN employs a deep neural network to approximate these Q-values.

The fundamental concept behind DQN is to leverage a deep neural network for acquiring a low-dimensional representation of the game state. This representation, in turn, is used to estimate the Q-values. The neural network undergoes training through a modified form of stochastic gradient descent known as "experience replay." This technique involves randomly sampling transitions from a replay buffer to mitigate correlations within the training data.

In this method, the original text and the candidate answer are expressed in vector form, and the relationship between them is learned by using DQN model, that is, the probability of determining whether an answer is plagiarism. Then, the improved differential evolution algorithm is used to screen and evaluate the candidate answers, and the answer with the highest probability is selected as the final result Li et al[9]. proposed a deep reinforcement learning framework based on DQN to generate dialogue responses. The framework takes advantage of the high efficiency and stability of DQN to learn to generate appropriate dialogue responses through reinforcement learning methods. Experimental results show that this framework can produce conversation responses with higher quality and is more effective than other deep learning methods.

3. DE Algorithm design

3.1. Principles

The DE (Differential Evolution) algorithm follows a fundamental principle encompassing four essential steps: initialization, mutation, crossover, and selection. At its core, the concept revolves around generating new solutions through mutation and crossover operations in each iteration, while retaining the more adaptive solutions through selection operations. Nonetheless, a critical aspect to consider lies in the delicate balance between exploration and exploitation within the search process, as this balance significantly influences the algorithm's performance.

3.1.1. Initialisation

The DE algorithm begins by generating an initial population. Each individual in the population is an n-dimensional vector of real numbers, where n is the dimension of the optimisation problem. Each component of the individuals is generated randomly within the definition domain of the problem. One potential issue with random initialisation is that it may lead to a poor distribution of individuals in the search space, which could affect the algorithm's efficiency.

3.1.2. Variants

For each target individual in the population, the DE algorithm randomly selects three mutually exclusive individuals from the population. The differences of these three individuals are weighted and summed with the target individuals to obtain a variance vector. The variation operation introduces diversity in the population and facilitates global search. However, the choice of mutation strategy and the scaling factor F significantly influence the balance between exploration and exploitation.

The mutation operator builds a mutant vector as

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

Where $\vec{x}_{r_1,g}$, $\vec{x}_{r_2,g}$, $\vec{x}_{r_3,g}$ are three (different) candidate solutions randomly chosen from the current population, and F is a factor scaling.

3.1.3. Crossover

The variance vector is crossed with the target individual to produce a test individual. The crossover operation is performed by randomly selecting some components of the parent individual and replacing them with the corresponding components of the variance vector. The crossover probability determines the degree of similarity between the test individual and the target individual. Although crossover helps

to maintain diversity, an inappropriate crossover probability may lead to either premature convergence or slow convergence.

3.1.4. Selection

Compare the fitness of the test individual with that of the target individual. If the test individual has a higher fitness, it is replaced by the target individual. Otherwise, the target individual remains unchanged. The selection operation ensures that the best individuals of the population are retained. However, a too-greedy selection process can lead to premature convergence, reducing the algorithm's ability to escape local optima. The selection operator then chooses the better solution from the trial and target vectors.

3.2. Algorithm Analysis

3.2.1. Advantages

(1) Strong global search capability for complex non-linear, non-convex and multi-modal problems, which makes DE a versatile optimization technique [10].

(2) Faster convergence and better search efficiency compared to other evolutionary algorithms due to its inherent simplicity and effective search mechanism [11].

(3) Fewer parameters, relatively easy to tune the parameters, and easy to implement, which contributes to its popularity in various applications.

3.2.2. Disadvantages

(1) For some problems, the DE algorithm may fall into a local optimum solution, and it is difficult to reach a global optimum solution. This limitation is a common issue in many optimization algorithms, and finding a balance between exploration and exploitation remains a challenge.

(2) In high-dimensional problems, the convergence speed of the algorithm may be slow, and the search efficiency may be reduced. Scaling the algorithm to high-dimensional search spaces requires more sophisticated strategies to maintain good performance.

(3) For discrete optimisation problems and combinatorial optimisation problems, the DE algorithm needs to be coded and decoded accordingly to suit the characteristics of the problem, which can increase the complexity of the algorithm [12].

3.2.3. Improvements and Applications

In order to overcome the limitations of the DE algorithm, researchers have proposed many improvement strategies, such as adaptive parameter control [13] hybrid strategies [14], parallel computing and so on. These improvement methods have improved the performance of the DE algorithm to a large extent. Nevertheless, there is still room for innovation in designing new strategies and techniques to address specific challenges in various application domains.

DE algorithms are widely used in function optimisation, constraint optimisation, neural network training, image processing, data mining and other fields [15]. The versatility of the DE algorithm and the continuous development of improvement strategies make it a promising tool for solving complex optimization problems in diverse areas.

3.2.4. Summary

The Differential Evolutionary Algorithm, as an excellent global optimisation algorithm, has achieved good performance in many practical problems. However, there are still some limitations of the DE algorithm for some specific problems. In order to further improve the performance of the algorithm, researchers should continue to focus on the improvement and application of the DE algorithm, considering the balance between exploration and exploitation, scalability to high-dimensional problems, and adaptation to various types of optimization problems.

4. Approach

4.1. Pre-Processing

The significance of pre-processing in Natural Language Processing (NLP) systems, highlighting its critical role in identifying key characters, words, and sentences that ultimately influence the model's final outcomes. However, it's worth noting that employing incorrect pre-processing techniques can actually lead to a degradation in model performance.

In our experiment, several common pre-processing techniques were applied, including stop word elimination and stemming. Stop words are essentially words that do not contribute significant meaning to the text, such as articles, prepositions, pronouns, and so on. These words tend to occupy space without enhancing the effectiveness of text mining applications. Consequently, the experimenters removed these stop words to enhance text efficiency and reduce the overall number of terms in line with Porter[16].

4.2. Bert

There are different techniques for word embedding, like Skip-gram [17] and GloVe [18]. Skip-gram is one technique and GloVe is another technique that uses matrix factorization. Both of these have been suggested to produce meaningful word representations for neural network models.

In this experiment, they used something called a pre-trained language model (PLM) called BERT. BERT is a bi-directional language model that's trained on large datasets like Wikipedia. It's designed to generate contextual representations of words, and it's typically fine-tuned for specific classification tasks.

4.3. DQN-based Approach

The DQN algorithm synergizes reinforcement learning with deep neural networks to enhance the decision-making process. We have chosen to embrace the DQN framework as a means to tackle the class imbalance issue within the context of plagiarism detection. Here's how we employ it:

4.3.1. Definition of the State space

Firstly, we define an appropriate state space to represent the input information in the plagiarism detection task. The state space may include features extracted from the source and suspicious sentences, such as BERT embeddings or sentence representation vectors. Additionally, other relevant information such as sentence length and syntax structure can also be considered.

4.3.2. Design of the Action Space

To address the class imbalance problem, it is necessary to design an action space that allows the model to dynamically adjust the classification decision threshold. This threshold determines the boundary for classifying plagiarism and non-plagiarism, thereby influencing the accuracy and recall of the classification results. By adjusting the decision threshold in the action space, a trade-off between precision and recall can be achieved to tackle the class imbalance problem.

4.3.3. Definition of the Reward Signal

In DQN, a reward signal needs to be defined to guide the learning process. In the case of class imbalance, the reward signal should take into account both the accuracy of classification and the degree of class imbalance. One possible design for the reward signal is to encourage the model to successfully detect minority class instances (i.e., plagiarism instances) by assigning a higher reward value, while assigning a lower reward value or penalty when the model misclassifies minority class instances as majority class instances (i.e., non-plagiarism instances).

4.3.4. Design of the Q-Network Architecture

Within the DQN framework, the Q-network accepts the current state as input and produces Q-values corresponding to each potential action. In our context, the action pertains to the adjustment of the classification decision threshold. Through the training of the Q-network, the model acquires the ability to learn the optimal policy for selecting the most appropriate action based on the current state. In our implementation, we employ a Convolutional Neural Network (CNN) as the architectural foundation for the Q-network [19].

4.3.5. Training and Optimization

During the training process, the model engages with the environment to accumulate experiences. In this interaction, the environment provides both the current state and a reward signal. These gathered experiences are subsequently employed to compute target values for the Q-values. The loss is determined by evaluating the disparity between these target values and the Q-values generated by the network. The weights of the neural network are then adjusted using the back propagation algorithm to minimize the loss function.

To initiate the process, the parameters of the Q-network require an initial setup. In our approach, we initialize these parameters with randomly assigned weights and biases. The current state is then fed into the Q-network to obtain the corresponding Q-values. The classification decision threshold is determined via ϵ -greedy selection. The chosen action is executed, and we observe both the classification result and the reward signal. Subsequently, the parameters of the Q-network are updated based on this feedback, aiming to optimize the policy. The current state and action are transitioned to the next state and action for the subsequent iteration.

To further enhance results, it is essential to continuously update the parameters of the Q-network. In each training step, we store the current state, action, reward, and next state in an experience replay buffer. Random batches of experience samples are then selected from this buffer to update the parameters of the Q-network. This approach fosters sample independence and diminishes training correlation. Adjustments to the algorithm's parameters are made based on the outcomes of training and feedback information.

5. Discussion

5.1. Evaluation and Performance Analysis

The trained DQN model is evaluated on a benchmark dataset, and its performance in addressing the class imbalance problem in plagiarism detection is assessed using the F1 score evaluation metric. The results are compared with existing methods, including the model proposed in the original paper, to demonstrate the effectiveness of the DQN-based approach.

5.2. Policy Improvement

After a certain number of training iterations, the model gradually learns the optimal policy. In the case of class imbalance, the model balances precision and recall by adjusting the classification decision threshold in the action space. As the training progresses, the model adjusts the decision threshold to adapt to the characteristics of imbalanced data[20]. In each training step, the model inputs the current state into the Q-network and selects the action (i.e., classification decision threshold) with the highest Q-value. The quality of the decision is evaluated based on the reward obtained from the selected action. By continuously interacting with the environment, collecting experiences, and adjusting the parameters of the Q-network, the model gradually improves its policy and becomes better equipped to address plagiarism detection issues on imbalanced datasets.

6. Conclusion

In this study, we propose a DQN-based approach that presents a promising solution to the class imbalance issue in plagiarism detection. Building upon a novel plagiarism detection model that

incorporates BERT word embeddings, an attention mechanism-based LSTM approach, and an enhanced DE (Differential Evolution) algorithm for network pre-training, our method offers a fresh and effective strategy for addressing class imbalance, particularly in binary imbalanced problems, within the context of plagiarism detection.

By harnessing the DQN algorithm, which amalgamates reinforcement learning and deep neural networks, our model can dynamically adapt its classification decision threshold. This adaptive mechanism significantly enhances the detection performance of plagiarism instances, particularly on datasets characterized by class imbalance.

Our research has yielded positive results in improving the accuracy of plagiarism detection. However, on the other hand, superior algorithms may exhibit better performance in sample classification problems. Certainly, this warrants further exploration in our future research endeavors.

References

- [1] Hu, Runqiao, Shaofan Wang, and Ke Li. "Visual Active Tracking Algorithm for UAV Cluster Based on Deep Reinforcement Learning." *Proceedings of 2022 International Conference on Autonomous Unmanned Systems (ICAUS 2022)*. Singapore: Springer Nature Singapore, 2023.
- [2] Hochreiter, S. (1997). Long Short-Term Memory. Technical Report, Institute for Computer Science, Austrian Federal Institute of Technology (ETH), Zurich, Switzerland
- [3] Larry R Medsker and LC Jain. Recurrent neural networks. *Design and Applications*, 5:64–67, 2001.
- [4] Navin Kumar Manaswi. RNN and LSTM. In *Deep Learning with Applications Using Python*, pages 115– 126. Springer, 2018.
- [5] Sepp Hochreiter and Jurgen Schmidhuber. Long short-term memory. *Neural computation*, 9(8):1735– 1780, 1997.
- [6] Storn, R., & Price, K. M. (1997). Differential evolution—a simple and efficient heuristic for global optimization over continuous spaces. *Journal of Global Optimization*, 11(4), 341– 359.<https://doi.org/10.1023/a:1008202821328>
- [7] Swagatam Das and Amit Konar. Automatic image pixel clustering with an improved differential evolution. *Applied Soft Computing*, 9(1):226–236, 2009.
- [8] Iztok Fister, Dusan Fister, Suash Deb, Uros Mlakar, and Janez Brest. Posthoc analysis of sport performance with difffferential evolution. *Neural Computing and Applications*, 32(15):10799– 10808, 2020.
- [9] Li, J., Monroe, W. S., Ritter, A., Jurafsky, D., Galley, M., & Gao, J. (2016). Deep Reinforcement Learning for Dialogue Generation. <https://doi.org/10.18653/v1/d16-1127>
- [10] Das, S., & Suganthan, P. N. (2011). Differential evolution: A survey of the state- of-the-art. *IEEE Transactions on Evolutionary Computation*, 15(1), 4-31.
- [11] Vesterstrom, J., & Thomsen, R. (2004). A comparative study of differential evolution, particle swarm optimization, and evolutionary algorithms on numerical benchmark problems. *Proceedings of the 2004 Congress on Evolutionary Computation*, 1980- 1987.
- [12] Tasgetiren, M. F., Sevkli, M., Liang, Y. C., & Gencyilmaz, G. (2004). Particle swarm optimization algorithm for single machine total weighted tardiness problem. *Proceedings of the 2004 Congress on Evolutionary Computation*, 1412- 1419.
- [13] Zhang, J., & Sanderson, A. C. (2009). JADE: Adaptive differential evolution with optional external archive. *IEEE Transactions on Evolutionary Computation*, 13(5), 945-958.
- [14] Qin, A. K., Huang, V. L., & Suganthan, P. N. (2008). Differential evolution algorithm with strategy adaptation for global numerical optimization. *IEEE Transactions on Evolutionary Computation*, 13(2), 398-417.
- [15] Liu, J., & Lampinen, J. (2005). A fuzzy adaptive differential evolution algorithm. *Soft Computing*, 9(6), 448-462.

- [16] Porter, M. F. (1997). An algorithm for suffix stripping. *Program: Electronic Library and Information Systems*, 14(3), 130– 137.<https://doi.org/10.1108/eb046814>
- [17] Word2Vec Tutorial - The Skip-Gram Model · Chris McCormick. (2016, April 19).
<http://mccormickml.com/2016/04/19/word2vec-tutorial-the-skip-gram-model>
- [18] Pennington, J., Socher, R., & Manning, C. D. (2014). Glove: Global Vectors for Word Representation.<https://doi.org/10.3115/v1/d14-1162>
- [19] Hessel, Matteo, et al. "Rainbow: Combining improvements in deep reinforcement learning." *Proceedings of the AAAI conference on artificial intelligence*. Vol. 32. No. 1. 2018.
- [20] Gokcesu, Kaan, and Hakan Gokcesu. "Generalized huber loss for robust learning and its efficient minimization for a robust statistics." *arXiv preprint arXiv:2108.12627* (2021).