HyCode: A Code Similarity Assessment Tool Utilizing Recurrent Neural Networks

James Marcel A. ABAWAG[a], **Aleczia S. TORDILLA**[b] **& Joshua C. MARTINEZ**[c]
Department of Computer Science, Ateneo de Naga University, Philippines
[a]ajaabawag@gbox.adnu.edu.ph
[b]atordilla@gbox.adnu.edu.ph
[c]joshuamartinez@gbox.adnu.edu.ph

Abstract: Academic dishonesty, particularly source-code plagiarism, poses significant ethical challenges in educational institutions and online coding platforms. It undermines the integrity of the learning and teaching process as well as the credibility of students and institutions. To address these challenges, this study developed a code similarity assessment tool utilizing deep neural networks, specifically character-level recurrent neural networks (char-RNN) and long short-term memory networks (LSTM), to detect source-code plagiarism. It leverages the strengths of both models while minimizing their weaknesses, allowing it to learn and capture both low-level and short-term patterns as well as complex and long-term dependencies in the source code. The dataset used was mainly from the “IR Plag Dataset", and data augmentation and various preprocessing techniques were performed. The final model configuration of the hybrid neural network architecture resulted in training and validation accuracy of 99% and 90% , respectively. Its evaluation was conducted using various metrics such as precision, recall, and F1-score. The hybrid neural network architecture achieved a precision of 0.94, a recall of 0.935, an F1-score of 0.94, and a final accuracy of 93.75% . In addition, the tool was also evaluated on real-world data and discovered to be capable of identifying a range of code similarities, providing assurance that the tool can effectively differentiate authentic or original work from work that may have been plagiarized. However, the evaluation also revealed the presence of false positives and negatives, which leaves room for improvement.

Keywords: Source Code, Plagiarism Detection, Neural Network

1. Introduction

Academic dishonesty has become a rampant and troubling ethical issue among students and educational institutions. Academic dishonesty includes acts such as cheating and plagiarism, wherein one’s ideas are copied or stolen by others without giving proper credit (Mahabeer & Pirtheepal, 2019) . These actions diminish the quality of learning and teaching processes, the credibility of students, as well as the reputation of the educational institution. In the context of programming, source-code plagiarism refers to the phenomenon where different pieces of code share similar structures, patterns, and functionalities (Holden, et. al, 2021) . Thus, this study presents the development and evaluation of a code similarity assessment tool that utilizes deep neural networks to analyze source codes and provides a probability of plagiarism. It sought to answer the following research questions: (a) How can the hybrid neural network architecture of Char-RNN and LSTM models enhance the detection of source code plagiarism? (b) What are the key factors that contribute to the performance of the hybrid neural network architecture of Char-RNN and LSTM models in detecting source code similarity? (c) How does the incorporation of bidirectional layers in both Char-RNN and LSTM models impact the detection of similar patterns in source code?

2. Methodology

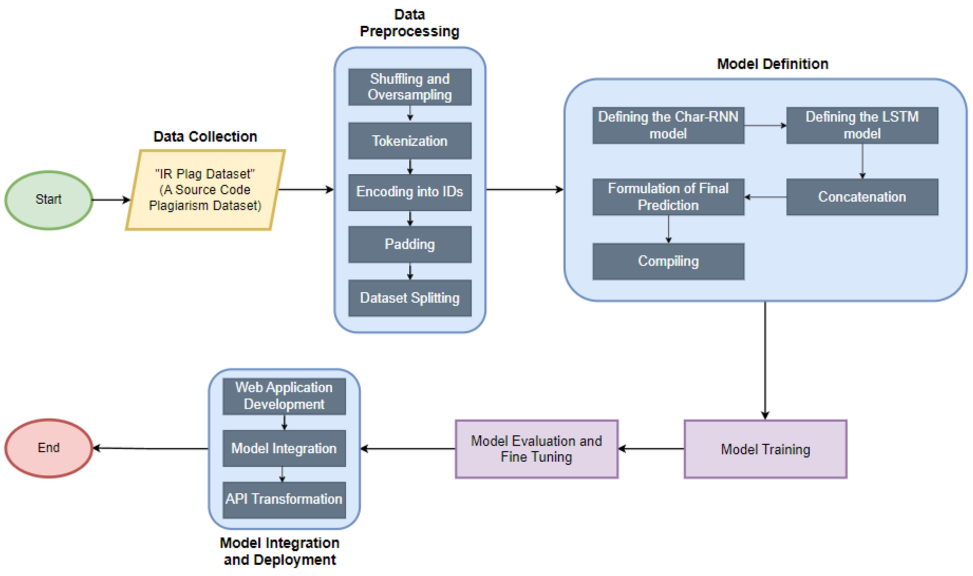
Data Preprocessing
Shuffling and Oversampling
Tokenization
Encoding into IDs
Padding
Dataset Splitting
Model Definition
Defining the Char-RNN model
Defining the LSTM model
Formulation of Final Prediction
Concatenation
Compiling
Data Collection
"IR Plag Dataset" (A Source Code Plagiarism Dataset)
Start
End
Web Application Development
Model Integration
API Transformation
Model Integration and Deployment
Model Evaluation and Fine Tuning
Model Training

Figure 1. Pipeline.

3. Results and Discussion

The Char-RNN model achieved a training accuracy of 90% and a validation accuracy of 70%. The LSTM model performed slightly better, with a training accuracy of 94% and a validation accuracy of 71%. Both models showed potential for improvement, particularly in terms of validation accuracy, which is an indication of how well the model generalizes to unseen data as well as preventing overfitting. The hybrid neural network, combining Char-RNN and LSTM, initially resulted in a training accuracy of 95%, resulting in an improvement of 5% over the Char-RNN model and 1% over the LSTM model. However, the validation accuracy remained at 70%, suggesting that overfitting was still an issue. In response to this, the hybrid neural network model was configured by adjusting parameters such as the number of units in bidirectional layers and hyperparameters such as the number of epochs, batch size, patience, and validation split. These modifications led to a significant increase in both training and validation accuracy, reaching 99% and 90%, respectively. This resulted in an improvement of 9% in training accuracy and 20% in validation accuracy over the Char-RNN model. Compared to the LSTM model, the improvements were 5% in training accuracy and 19% in validation accuracy.

Table 3. Comparative Results of the Models from Both Classes

Model	Precision	Recall	F1-Score
Char-RNN	0.85	0.85	0.85
LSTM	0.89	0.885	0.885
Hybrid Neural Network	0.94	0.935	0.94

In Table 3, a comparative result of the models' performance across various evaluation metrics such as precision, recall, and f1-score in both classes is presented. The Char-RNN model had scores of 0.85 across metrics. The LSTM model resulted in improved precision, recall, and F1-score metrics, all scoring around 0.885 to 0.89, and a final accuracy of 88.75%. The hybrid neural network model proves to be the most effective, with a precision of 0.94, a recall of 0.935, an F1-score of 0.94, and a final accuracy of 93.75%.

4. Conclusion

The study demonstrated that the hybrid neural network architecture, combining Char-RNN and LSTM models, significantly enhances the detection of source code plagiarism. The Char-RNN model excels at capturing low-level sequential patterns and short-term dependencies in the source code data. On the other hand, the LSTM model is adept at learning, capturing, and retaining complex sequential patterns and long-term dependencies. By integrating these two models into a hybrid neural network architecture, the tool leverages the strengths of both models. The hybrid hybrid neural network model achieved a final average training accuracy of 99% and a validation accuracy of 90%, compared to 90% and 70% for Char-RNN and 94% and 71% for LSTM.

This study identified several key factors that contribute to the performance of the hybrid neural network architecture in detecting source code similarity. The quality and diversity of the “IR Plag Dataset” provided a wide range of programming concepts and tasks. Secondly, the preprocessing steps applied to the dataset, including data augmentation, tokenization, encoding into IDs, pad sequencing, and splitting the dataset ensured that the data is in a suitable format for learning. Thirdly, the model’s performance was significantly influenced by careful tuning parameters like the bidirectional layer units, epochs, batch size, patience, and validation split. Lastly, the implementation of early stopping and other fine-tuning processes helped to lessen overfitting and improve the model’s performance.

Lastly, this study found that the incorporation of bidirectional layers in both Char-RNN and LSTM models significantly enhances the detection of similar patterns in source code. Bidirectional layers allow the model to process the input sequence in both forward and backward directions, enabling the model to capture dependencies in both temporal directions. In the Char-RNN model, the bidirectional layer enhances the model’s ability to capture low-level sequential patterns and short-term dependencies. On the other hand, in the LSTM model, the bidirectional layer improves the model’s capacity to understand intricate patterns within the character sequences and manage long-term dependencies. The incorporation of bidirectional layers in the hybrid neural network model resulted in a final average training accuracy of 90% and an average validation accuracy of 90%, indicating that it was highly effective and efficient in detecting similar patterns in source code. Therefore, the incorporation of bidirectional layers in both Char-RNN and LSTM models plays a crucial role in enhancing the detection of similar patterns in source code.

5. Recommendation

Future research should focus on several key areas based on this study's findings. Firstly, using larger and more diverse datasets can enhance model generalization. Secondly, exploring advanced preprocessing techniques, like byte pair encoding or context-aware encodings such as ELMo or BERT, could improve token representation. Additionally, investigating state-of-the-art models, such as transformers or capacitor networks, may offer better performance. Thirdly, upgrading computational resources could enhance training efficiency. Lastly, integrating the tool with APIs to include external sources, like online code repositories, could increase its robustness and versatility.

References

L. Holden Olivia, E. Norris Meghan, and A. Kuhlmeier Valerie. 2021. *Academic integrity in online assessment: a research review*. Frontiers in Education, 6. doi:10.3389/feduc.2021.639814.

P. Mahabeer and T. Pirtheepal. 2019. *Assessment, plagiarism and its effect on academic integrity: experiences of academics at a university in south africa*. South African Journal of Science, 115, 11/12. doi: https://doi.org/10.17159/sajs.2019/6323.