Journal of Computer Science

Research Article

Assessing Bi-LSTM Model's Performance in Identifying AI-Generated Text in Digital Media

[1]Tinashe Crispen Garidzira, [2,3]William Tichaona Vambe and [4]Courage Matobobo

[1]Department of Computer Science, University of Fort Hare, Alice, South Africa
[2]Department of Mathematical Science and Computing, Walter Sisulu University, East London, South Africa
[3]National Institute for Theoretical and Computational Sciences, Mthatha, South Africa
[4]Department of Business and Application Development, Walter Sisulu University, East London, South Africa

Article history
Received: 25-11-2024
Revised: 09-05-2025
Accepted: 20-05-2025

Corresponding Author:
William Tichaona Vambe
Department of Mathematical Science and Computing, Walter Sisulu University, East London, South Africa;
National Institute for Theoretical and Computational Sciences, Mthatha, South Africa
Email: vambewilliam@gmail.com

Abstract: The rapid growth of AI-generated content from ChatGPT, Gemini, and many others poses significant challenges in digital media. There are increasing instances of text generated by machine learning models being indistinguishable from human-written text. This calls for an effective way of detecting and distinguishing text generated by AI from human text. Using a Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology, a Bidirectional Long Short-Term Memory (Bi-LSTM) model for classifying AI-generated text in a digital media context was developed. A carefully curated dataset of human and AI-generated text samples was used. The Bi-LSTM model without an embedding layer was implemented, optimizing the model to capture complex linguistic patterns apparent in each text type. An experimental setup was used to evaluate the effectiveness of the model. It was noted that the model achieved a remarkable test accuracy of 99.79%, with a loss of 0.009. To facilitate practical implementation, we developed a web application using Next.js with our model served from a Flask server that enables real-time AI text detection. Our results highlight the model's ability to accurately identify AI-generated text, providing valuable insights into deploying such models to verify content on media platforms. This study highlights the potential of neural network-based classifiers to address the pressing need for automated AI text detection in an increasingly AI-influenced digital ecosystem.

Keywords: AI-Generated Text Detection (ATD), Bi-LSTM Model, Digital Media Classification (DM), Natural Language Processing (NLP)

Introduction

A rapid advancement in Machine Learning (ML), particularly Natural Language Processing (NLP), has led to a proliferation of AI-generated text content across various digital media platforms (Cheng *et al.*, 2024). The content generated by models such as OpenAI's GPT family and Google's BERT has become increasingly complex, making it nearly indistinguishable from human-written text (Kalyan, 2024). As a result, digital media platforms face significant challenges in distinguishing between AI-generated and human-written content, which has implications for content authenticity, reader trust, and the spread of misinformation (Jawahar *et al.*, 2020; Ott *et al.*, 2011). This predicament has affected many application areas, for example, higher education, as most students are now believed to use AI to write assignments and research. It has become difficult for Turnitin and other AI-detecting software to detect such.

Detecting AI-generated text poses challenges due to the complex linguistic patterns and syntax that advanced language models can mimic (Zellers *et al.*, 2019). Conventional text classification methods struggle to handle this level of nuance because they may not capture the subtle differences in linguistic features between AI and human-generated text (Zheng, 2023).

Several approaches have been used to address the above-mentioned problem.

Zellers *et al.* (2019) relied on data quality to train Grover to detect AI-generated information. However, their approach struggled to generalize across different contexts.

Research by Brown *et al.* (2020) explored the capabilities of the GPT-3 model and addressed issues related to text generation and detection. Although GPT-3 enables high-quality text generation, they point out that AI text detection remains challenging due to subtle

overlaps in syntactic patterns between human and AI text.

Solaiman *et al.* (2019) studied AI-generated texts' ethical implications and detection strategies. Their study involved testing various classifiers and found that although the detection models were promising, their performance dropped significantly in various real-world scenarios.

Recent studies have introduced the application of Deep Learning (DL) techniques, such as Bidirectional Long Short-Term Memory (Bi-LSTM) networks for emotion detection, and software requirements, respectively (Abbas *et al.*, 2024; Amadi *et al.*, 2023; Vambe & Garidzira, 2024). Their works proved that Bi-LSTM can effectively capture sequential dependencies in textual data, which is essential for identifying the nuanced linguistic structures characteristic of AI-generated text. However, these studies also inadequately address the risks of overfitting and do not provide a general framework applicable to different domains, thus limiting the broader applicability of their results in real-world situations.

Evidently, a lot has been done in trying to address the challenge. However, open research gaps remain, as articulated above, based on the weaknesses of the proposed solutions.

Therefore, the aim of this study is to:

a) Explore the use of the Bi-LSTM approach in correctly classifying (detecting) and distinguishing AI-generated text from human-generated text.

b) Assess the model's accuracy, effectiveness, and efficiency in identifying AI-generated text in digital media when compared with other existing models.

Furthermore, a web application was developed using Next.js, with the model deployed on the Flask server, facilitating real-time AI text detection to enable real-world applicability. The application provides media platforms with a content verification tool, ensuring greater transparency and control over published material.

The paper starts with a discussion of related work, followed by an explanation of the methodology and its stages as adopted in this work. Next, the design and implementation are detailed, and the results are presented. A discussion of the results follows. Lastly, a conclusion with future work is provided.

Related Works

The rapid growth of AI-generated content has prompted researchers to explore reliable methods for distinguishing it from human-authored text (Chakraborty *et al.*, 2023). This area has gained particular importance in recent years, and in application areas like education, it is paramount to check for plagiarism and the use of AI in writing, be it assignments, projects, or research. Software like Turnitin cannot detect most AI-written text, which calls for a reliable solution in this application area and other application areas.

Early work in detecting AI-written text relied on rule-based approaches and keyword analysis (Cheng *et al.*, 2024). Still, these methods have become inadequate as Large Language Models (LLMs) such as GPT-3, BERT, and others generate text with increasingly complex linguistic structures that are now existing (Cheng *et al.*, 2024).These models can mimic human-like writing patterns, which pose significant challenges to traditional text verification techniques (Zellers *et al.*, 2019).

Recent advances in ML, especially DL, have significantly improved the ability to classify text. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs) were initially popular choices for text classification, but they suffer from limitations in capturing long-term dependencies (Devlin *et al.*, 2019).

Bi-LSTM models have emerged as an effective solution, especially in natural language processing tasks that require preserving contextual information in both directions of a text sequence (Amadi *et al.*, 2023). Studies by Graves and Graves (2012) and Devlin *et al.* (2019) have demonstrated that Bi-LSTM models improve classification accuracy by effectively capturing context in text data, which is essential for distinguishing subtle linguistic cues in AI-generated content.

In the study by Petropoulos and Petropoulos (2024), the authors sought to address the challenge of detecting AI-generated text to prevent the misuse of advanced Text Generation Models (TGMs), such as those used in automated content generation and disinformation campaigns. They developed a ML classifier to distinguish between AI-generated and human-written texts. Although the specific tools and algorithms used were not detailed, it can be inferred that they likely used supervised learning techniques, possibly using DL models such as Bi-LSTM or Transformer-based architecture. The classifier demonstrated its effectiveness in distinguishing between the two types of text, achieving notable results, although exact performance metrics were not provided. The strength of their work lies in its practical applicability, offering a solution to a pressing problem directly related to combating the growing influence of AI in digital media and online content creation. However, one of the weaknesses of their approach was its lack of adaptability to rapidly evolving AI text generation models.

As AI-generated content advances and evolves, especially with new models that can better mimic human language features, classifiers trained on previous models may struggle to maintain high accuracy. This poses a challenge to ensure that the classifier remains robust over time as the TGM improves. Additionally, if the classifier is trained using a limited dataset, it may not generalize well to different linguistic or linguistic contexts, reducing its applicability across global digital platforms.

Addressing these limitations would improve the scalability and robustness of search, ensuring that detection methods remain relevant as AI technology advances.

Prova (2024) conducted a study to address the increasing challenge of distinguishing AI-generated text from human-written content. His study used multiple ML models, including XGB Classifier, SVM, and BERT. In their findings, BERT achieved the highest accuracy of 93%. The strength of this work lies in the extensive comparison of different algorithms, which shows that BERT is particularly effective at handling the complex linguistic structures inherent in AI-generated content. However, a notable weakness is the reliance on traditional models such as XGB and SVM, which may not fully capture the complex contextual dependencies in AI-generated text. Additionally, the study did not consider real-time detection, an important aspect of practical implementation in dynamic media environments.

In another study by Fraser *et al.* (2025), the researchers focused on the factors influencing the ability to detect AI-generated text, specifically with models such as GPT-3.5 and BERT. Their analysis highlighted the role of complexity and entropy as features that distinguish machine-generated text from human-generated text. The strength of their study lies in the innovative approach to using these features, which provides a unique perspective on AI detection beyond standard classification techniques. However, their work was limited by its focus on static text features, ignoring dynamic models that are capable of real-time detection. Their research did not explore real-time AI text detection, which may be necessary for practical implementation on media platforms.

Alkhairy (2024) proposed using Convolutional Neural Networks (CNNs) to detect AI-generated text. The model was able to capture subtle hierarchical patterns in text. Their results demonstrated that CNNs outperform traditional ML classifiers in some cases. The strength of their approach lies in the ability of CNNs to identify deep features in text data, which can lead to better performance in classifying complex AI-generated content. However, their study was limited by the inability of CNNs to capture long-term dependencies in text, a task better suited to sequential models such as Bi-LSTM or transformer-based architectures. Additionally, the study did not consider real-time detection capabilities, which are critical for effectively deploying these models in an ever-changing digital ecosystem.

Petropoulos and Petropoulos (2024) highlighted the pressing challenge posed by the misuse of Text Generation Models (TGMs) in the rapidly evolving era of artificial intelligence. They highlight how these models, while advancing industries and education, are also being exploited to generate fake news or evade intellectual effort.

Mitrović *et al.* (2023) focused on training a ML model to differentiate human-generated text and ChatGPT-generated text, using explainable AI to discover linguistic style patterns of ChatGPT, especially in short online reviews. They achieved 79% accuracy.

Ghosal *et al.* (2023) examined the advances and challenges in detecting AI-generated text, addressing the need for robust detection frameworks to combat the misuse of LLMs in fake news and other harmful applications.

Qasim *et al.* (2022) used Transfer Learning (TL) approaches to perform text classification, where pre-trained LLMs models, such as BERT or GPT, were fine-tuned on specific text datasets for classification. While these models perform well, they often demand high computational resources and large datasets, which may not be feasible for real-time applications. On the other hand, Bi-LSTM models offer a balance between efficiency and performance, making them suitable for applications that require both practical implementation and accuracy (Abbas *et al.*, 2024).

In addition, Wani *et al.* (2024) presented a robust AI-based framework that used the integration of Bi-LSTM and Word2Vec to identify and reduce AI-generated spam effectively. Their work highlighted advanced NLP techniques and the performance of DL models to improve the authenticity of online content.

The above literature shows that studies have been done to address the integration of AI text detection models into web applications for immediate deployment. However, to the best of our knowledge, all the reviewed works in this study do not provide a practical tool or solution for real-time AI text verification. This can also be confirmed as a research gap in the current existing studies (Ghosal *et al.*, 2023; Mitrović *et al.*, 2023; Petropoulos & Petropoulos, 2024). Therefore, this calls for a practical tool or solution for real-time AI text verification, which this work seeks to provide by implementing a Bi-LSTM model on the Flask server in a Next.js web application. If successful, the innovation should highlight the potential of neural network-based classifiers in real-world applications, laying a foundation for future advances in automated AI content detection systems. Thus, the work will give a theoretical and practical foundation for further research.

Methodology

The research used the Cross-Industry Standard Process for Data Mining (CRISP-DM) methodology. Adopting the six stages, namely: business (problem) understanding, data understanding, data preparation, modeling, evaluation, and deployment (Ayele, 2020) for building an ATD system that used AI vs a human public dataset from Kaggle by Gerami (2023) to build a Bi-LSTM model.

Problem Identification

Literature review (Chigbu *et al.*, 2023; Kunisch *et al.*, 2023) and document inspection (van Beusekom *et al.*, 2010) methods were used. These two methods helped the researchers identify the existing research gaps of not having a practical tool or solution for real-time AI text verification. It was from the problem that the research objectives were formed, which indicated that there was a need to design the model and have data to train it.

Data Collection and Processing

To design the model, the researchers used a public dataset from Kaggle (Gerami, 2023), which contains unbalanced classes between AI-generated and human-generated text. Since our study was focused on a single language, English, the dataset contains only English text. The dataset was unbalanced in favour of human-generated text.

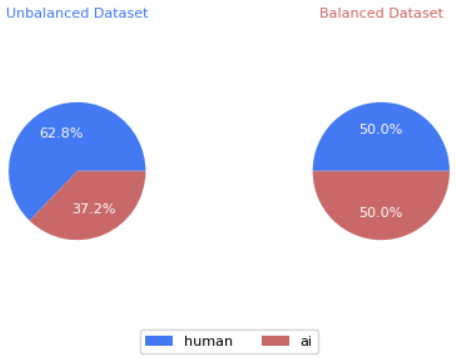

Fig. 1: Dataset distribution

The labels had to be balanced so that human and AI-generated labels would have a fraction of 0.5 each, because imbalanced classes in the dataset can ruin classification performance, particularly accuracy for minority classes, as articulated by Li *et al.* (2022). To achieve better precision, the dataset was balanced. Fig. 1 shows the distribution of labels in the original dataset and after balancing. After balancing, the dataset was split into three subsets: training, testing, and validation sets, as shown in Table 1.

Table 1: Dataset Distribution

Set	Examples	Fraction (%)
Training	232 240	64
Testing	72 576	20
Validation	58 060	16
Total	362 876	100%

Of the 362 876 sizes, 64% were allocated for training, 20% were reserved for testing, and 20% of the training data (58,060) were used for validation, representing 16% of the entire dataset. This dataset shows that a robust model can be built to ensure a reliable evaluation, as supported by Joseph and Vakayil (2022).

In the preprocessing stage, several steps were implemented to ensure data consistency and improve model performance. Tokenization was applied to convert text samples into individual tokens, preparing them for processing by the Bi-LSTM model. Text normalization, as defined by Joseph and Vakayil (2022) was followed. It includes lowercasing, punctuation removal, and space standardization to reduce input variability. Fig. 2 shows the visualization of word cloud plots for the text that was found in each set.

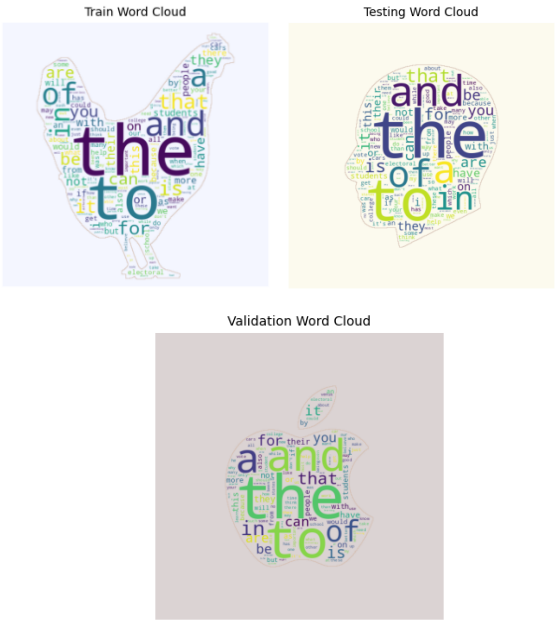

Fig. 2: Word clouds for text in sets

It can be noted from Fig. 2 that the most common words in this dataset were stop words. This brought our attention to removing stop words while preprocessing to improve model performance. This normalization step had an impact of reducing the training vocabulary to our model, as a result, reducing model parameters, which reduced the model training time.

Since ML models do not understand words but numbers, a vocabulary was created based on the training text after the text had been normalized. A vocabulary is essentially a word-to-index mapping, like a Python dictionary, used as a reference during text processing and model inference (Eichstaedt *et al.*, 2021). We used the training text to create this vocabulary to avoid information leakage. During vocabulary creation, we set the minimum frequency to 5, meaning that words that do not appear more than 5 times in our text were automatically converted to unknown words. This had a significant impact on reducing the vocabulary size to 47 619.

Larger vocabulary sizes increase the model's trainable parameters, which impact the training time, especially on CPU computers. During text tokenization, which is a process of converting a list of words into a list

of numbers (Vaswani *et al.*, 2017), we set the maximum length to 300 so that tokens of sentences that are less than 300 will be padded with a padding index. The ones that were larger than 300 words were truncated. This was done because LSTM or Recurrent Neural Networks generally require all sentences in a batch to have the same length, as supported by Oruh *et al.* (2022).

Word embeddings, specifically GloVe, were downloaded to be applied in the embedding layer of a Bi-LSTM model. A study by Krouska *et al.* (2020) shows that pre-trained word embeddings like Twitter GloVe and FastText improve tweet classification. FastText demonstrates consistency, and Twitter GloVe achieves high accuracy (Krouska *et al.*, 2020). We used the Glove vectors 300 dim, which were trained with about 840 billion words on our Bi-LSTM model, to get good accuracy.

Model Creation

The model for this study was created using three layers: Embedding, LSTM (bidirectional), and Linear. The embedding layer was initialized with pre-trained word embedding vectors that suit our data. A dropout of 50% was applied to the LSTM layer to prevent the model from over-fitting and to improve model accuracy, as supported by Lee and Lee (2020). A final dense layer with an output of one was then added as the last layer of the model. The model parameters were counted. Table 2 shows the model parameters.

Table 2: Model Parameters

Model	Total	Trainable	Non-Trainable
Bi-LSTM	17 005 957	17 005 957	0

From Table 2, it was observed that our model has more trainable parameters of about 17 million. Additionally, our dataset was also huge. Moreover, we utilized a free GPU from Google colab to reduce the model training time. Furthermore, we set our batch size to 128. Setting a huge batch size significantly reduces total training time in a single epoch, even though it reduces model generalization, as articulated by Smith *et al.* (2018). The model was trained using a supervised approach using Binary Cross-Entropy with Logits Loss (BCEWithLogitsLoss) as a function that computes the model's loss between predicted and real labels. The Adam optimizer was used to adjust the learning rate for each parameter because of its stability during the learning process, as supported by Kingma and Ba (2014).

Model Evaluation

The trained model was experimentally tested and evaluated. The model was trained for 6 epochs, and the training history was saved after each epoch for the model evaluation phase. The model was saved when the validation loss from the previous epoch was greater than the current loss. For model inference, the model was saved as a static file, and the vocabulary was saved together with the static Bi-LSTM model so that it could be used to create a GraphQL API using Graphene and Flask. Several metrics were used to evaluate the best-saved model before deploying it.

Model Deployment

As a final stage, the model and a Flask Server with Graphene were created to serve a serialized and saved .pt model as a GraphQL API, where the model server can receive text or text files and do the classification in real time of AGT model into a web application for real-time text classification. The same GraphQL API was configured to send requests to the Flask Server using a frontend web framework that was built using Next.js, a React-based framework (Svedas, 2024).

Design and Implementation

A robust framework for AI-generated text detection that integrates the Bi-LSTM model into a web application built with Next.js was implemented. Fig. 3 shows the architecture, which consists of several key components, namely: Next.js web application (client), a Flask server for model inference (backend), a static PyTorch-based Bi-LSTM model, which is served by the server, and a GraphQL-based data flow using Graphene and urql for the backend and client, respectively.

The goal of the system was to perform real-time discrimination between human and AI-generated text inputs, addressing the growing need for practical AI text detection solutions in digital media. As shown in Fig. 3, the framework has a front-end and back-end.

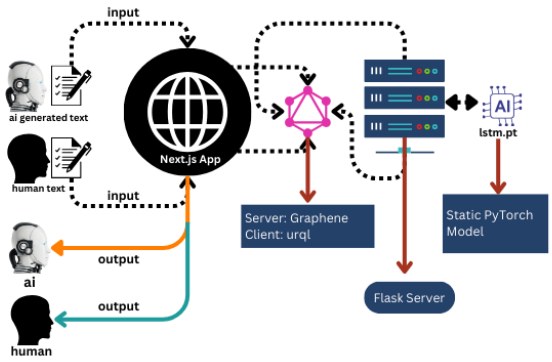

Fig. 3: Framework Architecture

Front-End

The application user interface is built using Next.js, a React-based web development framework that provides efficient server-side rendering and simplified deployment for scalable web applications (Svedas, 2024). Human or AI-generated text or text files are entered or uploaded through the Next.js interface. This text data is then passed to the backend for processing and inference.

Back-End

The back-end consists of a Flask server that serves the trained Bi-LSTM model for real-time inference through a GraphQL API. Flask was chosen for its lightweight structure and ease of integration with machine-learning models developed in Python (Fergus & Chalmers, 2022). The PyTorch-trained model is deployed as a static .pt file, which allows the model to be served based on the weights, and the model was served during training. The Bi-LSTM has been optimized to capture complex linguistic patterns in the input text, distinguishing between human-generated and AI-generated content with high accuracy.

Dataflow

To facilitate efficient data management and flow between the front-end and back-end, the application used a GraphQL API. This approach uses Graphene as the GraphQL server framework, with urql as the client, ensuring optimized data retrieval with minimal server load. GraphQL enables flexible queries, allowing the front-end to request only the specific data it needs, improving performance and scalability (Brito & Valente, 2020).

Model Architecture

The model architecture combines a Bi-LSTM network with pre-trained GloVe (glove.840B.300d) integration to classify text as AI-generated or human-written, which suits our Human vs. AI data (Gerami, 2023). The embedding layer captures semantic and syntactic nuances, improving the model's ability to understand the context of words, as supported by Krouska *et al.* (2020). Using bidirectional processing, the Bi-LSTM layer captures the context of past and future tokens, which is important for distinguishing text sources. We adopted the notion from previous studies by Chatelain *et al.* (2022) and Patil *et al.* (2023) which have shown that embedding methods such as Word2Vec and bidirectional models are effective in improving the task of text classification by increasing accuracy and confidence. Fig. 3 shows the model architecture of this study.

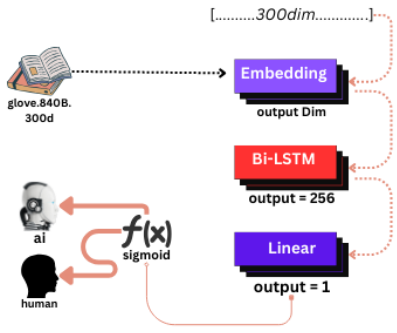

Fig. 4: Model Architecture

Fig. 4 shows the model architecture used for training and evaluation, it shows that after passing a 300-dim vector input to the embedding layer, the output is then passed to the Bi-LSTM, and the Bi-LSTM output is fed into a linear layer with a sigmoid activation function that provides a probability score, classifying the text based on a set threshold (Pratiwi *et al.*, 2020).

Framework Output

The model sends real-time classification results to a Next.js app, offering users an immediate, user-friendly experience for verifying text authenticity. This setup is efficient for web-based applications that need rapid and accurate AI-content detection.

Results

During Bi-LSTM model training, the metrics were observed per each iteration, including epoch training losses and accuracies for both the training and validation sets. Total training time, last saved epoch, and epoch training time were other metrics that were monitored during model training.

Training Time

Table 3 shows the total training time the model took to train after training for 6 iterations/epochs. The last epoch number of the best model was saved.

Table 3: Model training time and last saved epoch

Training Summary	Value
Total Epochs	6
Last Saved Epoch	6
Total Training Time	1hrs 56min 47.23sec

From Table 3 above, the model took 1 hour, 56 minutes, and 47 seconds to train, and the best model was saved on epoch 6 out of the total of 6. Fig. 6 shows how long the model took to train each single epoch in seconds (s).

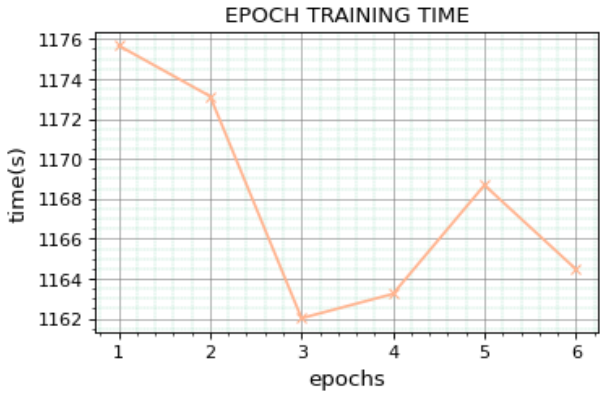

Fig. 5: Epoch training time

From Fig. 5, it can be noted that the model took more time to train in the first 2 epochs. The model quickly

completed its training on iteration 3 in 1 162 seconds.

Training and Validation Metrics

Fig. 6 shows the training and validation loss observed during the model training for 6 epochs.

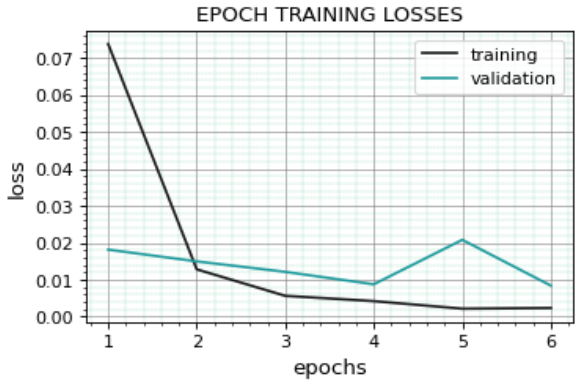

Fig. 6: Training and validation losses

From Fig. 6, the model started with a low validation loss below 0.02, while the training loss started above 0.07. It gradually decreased as the model trained for the second epoch. Although the training loss was lower than the validation loss after epoch 2, the validation loss remained relatively low, indicating that the model was neither overfitting nor underfitting.

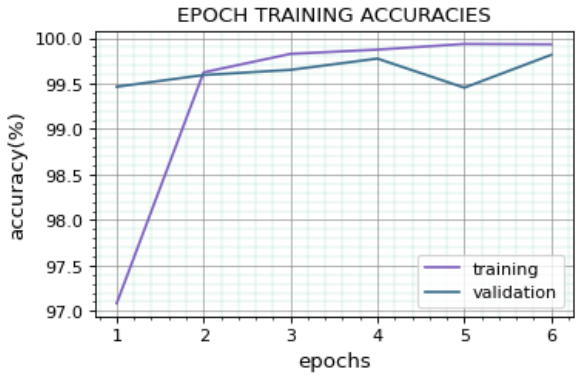

Fig. 7: Training and validation accuracies

Fig. 7 shows the training and validation accuracy of the Bi-LSTM model observed during model training for 6 epochs. The training accuracy was slightly below 97.5% on epoch 1, while the validation accuracy remained slightly above 99% from epoch 1 to epoch 6.

Best Model Accuracy and Loss

During model training, the best model was saved. Table 4 shows the loss and accuracy of the best-saved model after it has been evaluated on the test dataset.

Table 4: Best saved model test loss and accuracy

Metric	Value
Loss	0.009
Accuracy	99.79%

Table 4 above shows the best model's results regarding classification loss and accuracy on the test dataset. The model performed perfectly, with an accuracy of 99.79% and a loss of 0.009.

Best Model Confusion Matrix

Fig. 8 shows the classification confusion matrix of the best-saved model based on the test dataset.

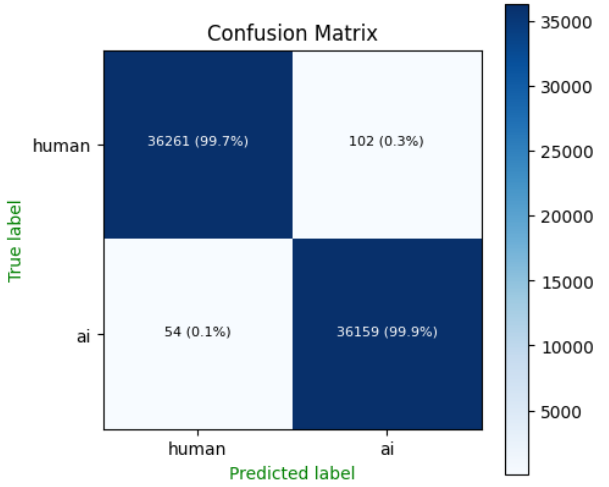

Fig. 8: Model classification report

Fig. 8 shows that the model achieved high accuracy, correctly classifying 99.7% of human-generated text and 99.9% of AI-generated text, with minimal classification error. A fraction of 0.1% (54 examples) of AI-generated text was misclassified as human text, and 0.3% (102 examples) of human text was misclassified as AI-generated.

Best Model Classification Report

Fig. 9 shows the classification report for the best model based on the testing dataset.

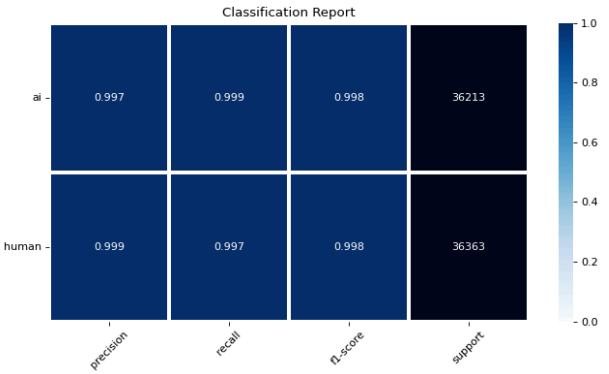

Fig. 9: Model classification report

Fig. 9 shows the Bi-LSTM model classification performance, with precision, recall, and F1 scores ranging from 99.7% to 99.9% for AI-generated and human-generated text, indicating highly accurate predictions.

Discussion

The findings of this study demonstrate that the Bi-LSTM model effectively distinguishes between AI-generated text and human-generated text in digital media. The model achieves high accuracy, precision, recall, and F1 scores, highlighting its robustness and reliability. As shown in the confusion matrix, Fig 7, the model's classification performance has a minimal classification error rate, supporting its suitability for this task.

The model took 1 hour 56 mininutes 47.23 seconds to train for only 6 epochs due to the total number of parameters the model has. The model had total trainable parameters of about 17 million. The dataset was large, with 232 240 pairs of 300-dimensional vectors of text with their paired labels in the training set and about 58 thousand 300-dimensional vectors of text with their paired labels. The model took about 1 176 seconds to complete the first epoch, while subsequent epochs took less time. This occurred because the data was not cached during the first iteration, a phenomenon also noted by Svogor *et al.* (2022).

The Bi-LSTM model quickly learns to reduce the loss while maximizing the accuracy during model training. The train and validation datasets had an accuracy of above 99% on the second epoch for both sets and a loss below 0.02, indicating that the model could learn the mathematical relationship between text features and their label. This is due to the effectiveness of RNNs, particularly the LSTM layer, in processing sequential data, as supported by Amadi *et al.* (2023) and Yu *et al.* (2019). Additionally, the data normalization that was done to the generated text had an impact in reducing the noise from text, resulting in the model having fewer things to focus on during learning.

The success of the Bi-LSTM model can be attributed to effective pre-processing techniques and careful feature selection, consistent with previous research, for example, Jang *et al.* (2020), highlighted the importance of feature engineering in improving model performance for text classification tasks. The Bi-LSTM architecture also allows the model to capture past and future contextual information in the text sequence, making it particularly well-suited to the nuanced differences between Human and AI text generation. These findings are consistent with the benefits of bidirectional networks highlighted in a study by Huang *et al.* (2015), which demonstrates how Bi-LSTM models excel at capturing long-term dependencies in sequential data.

However, computational requirements and model training time are limitations, suggesting that real-time applications may be difficult without optimization. Future work could explore techniques such as model pruning or quantization to reduce these requirements while maintaining accuracy. In a nutshell, this study provides evidence that the Bi-LSTM model is a promising approach for identifying AI-generated text in digital media because of the accuracy of the obtained data, which was above 95%.

Conclusion

This study shows the effectiveness of the Bi-LSTM model in classifying AI and human-generated text in digital contexts. The model's robust performance metrics reflect its reliability and accuracy, making it a valuable tool for content verification in an era where AI-generated content is increasingly important. This study contributes to ongoing efforts to develop robust detection algorithms capable of matching the sophistication of modern AI text generators.

This study has significance in many application areas, such as industries, including communications, education, and social media, where information authenticity is paramount. This study also demonstrates static model deployment and serving from web applications using GraphQL and Next.js integration using a Flask server. The finally deployed frontend web application allows users to verify the authenticity of textual content using text files or plain text.

Future Work

Several future research directions emerge from this study based on advances in AI text generation detection. Firstly, optimizing the Bi-LSTM model for real-time deployment could increase its applicability in rapidly changing digital environments. Model compression techniques such as pruning, quantization, or distillation could help reduce computational costs without significantly reducing performance, as supported by Dantas *et al.* (2024). Additionally, expanding the dataset to include more AI-generated content would likely improve the model's ability to generalize different text types.

Future research could also investigate the inclusion of additional linguistic features, such as sentiment or stylistic cues, to further improve classification accuracy. Hybrid models that combine CNNs with Bi-LSTMs can provide richer feature representations and produce even better results, as Sahoo and Chakraborty (2024) suggested. Another potential research direction is to examine the model's adaptability to different languages and dialects, reflecting the global reach of AI-generated content. This could facilitate cross-cultural applications of AI text detection and provide organizations worldwide with effective tools to verify the authenticity of content.

Another potential direction is creating mobile and desktop applications that verify text authenticity. We have already developed a web server that is serving a GraphQL endpoint, which can be used to make requests to the same server using mobile and desktop applications. Lastly, the model should be tested in application areas like education, where the need to detect plagiarism and AI-related text is paramount.

A potential improvement may be the development of a model that can distinguish between AI and a human in a variety of languages. Expanding the ability of the model to detect the difference in the style of writing in some languages will improve its flexibility and accuracy in determining the origin of the text. This approach can lead to better content classification, providing valuable information for tasks such as content censorship, religious detection, and automatic material analysis compared to human product materials.

Lastly, future work should incorporate adversarial testing to ensure the model's resilience, aligning with the goal of robust real-world applicability.

Acknowledgment

Special thanks go to Gerami for providing the dataset ("AI vs. Human Text" dataset), used in this research, found on Kaggle. The dataset played a crucial role in training and evaluating the model. Secondly, we thank Walter Sisulu University and the National Institute for Theoretical and Computational Sciences (NITheCS) for supporting the research.

Funding Information

The authors have not received any financial support or funding to report.

Author's Contributions

Tinashe Crispen Garidzira: Developed the model and web application, and authored the methodology, discussion, and results sections.

William Tichaona Vambe: Conceptualized the research topic, authored the introduction and literature review, and refined the methodology, results, and discussion sections.

Courage Matobobo: Crafted the abstract and wrote the remaining sections, including editing the discussion and results sections.

All authors read and approved the final manuscript.

Ethics

This work is original and has not been published anywhere. The authors declare that no ethical concerns are associated with this submission.

Conflict of Interest

The authors do not have any conflicts of interest to declare.

References

Abbas, J., Hu, Z., Kanwal, S., Ahmad, A., Almogren, A., & Altameem, A. (2024). Bi-LSTM-Based Model for Classifying Software Requirements. *Artificial Intelligence and Machine Learning*. https://doi.org/10.20944/preprints202410.2129.v1

Alkhairy, S. A. (2024). Rational-Exponent Filters with Applications to Generalized Exponent Filters. *IEEE Transactions on Circuits and Systems*, *72*(5), 2139–2152. https://doi.org/10.1109/TCSI.2025.3545459

Amadi, C., Odii, J., Okpalla, C., & La, O. C. I. (2023). Emotion Detection Using a Bidirectional Long-Short Term Memory (BiLSTM) Neural Network. *International Journal of Research Publication and Reviews*, *4*(11), 1718–1732.

Ayele, W. Y. (2020). Adapting CRISP-DM for Idea Mining. *International Journal of Advanced Computer Science and Applications*, *11*(6), 20–32. https://doi.org/10.14569/ijacsa.2020.0110603

Brito, G., & Valente, M. T. (2020). REST vs GraphQL: A Controlled Experiment. *2020 IEEE International Conference on Software Architecture (ICSA)*, 81–91. https://doi.org/10.1109/icsa47634.2020.00016

Brown, T. B., Mann, B., Ryder, N., Subbiah, M., Kaplan, J. D., Dhariwal, P., Neelakantan, A., Shyam, P., Sastry, G., Askell, A., Agarwal, S., Herbert-Voss, A., Krueger, G., Henighan, T., Child, R., Ramesh, A., Ziegler, D., Wu, J., Winter, C., & Amodei, D. (2020). Language Models are Few-Shot Learners. *Advances in Neural Information Processing Systems*, 1877–1901.

Chakraborty, S., Bedi, A. S., Zhu, S., An, B., Manocha, D., & Huang, F. (2023). On the Possibilities of AI-Generated Text Detection. *Computation and Language*. https://doi.org/10.48550/arXiv.2304.04736

Chatelain, Amélie., Djeghri, A., Hesslow, D., & Launay, J. (2022). Is the Number of Trainable Parameters All That Actually Matters? *Can't Believe It's Not Better! Workshop at NeurIPS 2021*, 27–32.

Cheng, H., Sheng, B., Lee, A., Chaudary, V., Atanasov, A. G., Liu, N., Qiu, Y., Wong, T. Y., Tham, Y.-C., & Zheng, Y. (2024). Have AI-Generated Texts from LLM Infiltrated the Realm of Scientific Writing? A Large-Scale Analysis of Preprint Platforms. *BioRxiv*. https://doi.org/10.1101/2024.03.25.586710

Chigbu, U. E., Atiku, S. O., & Du Plessis, C. C. (2023). The Science of Literature Reviews: Searching, Identifying, Selecting, and Synthesising. *Publications*, *11*(1), 2. https://doi.org/10.3390/publications11010002

Dantas, P. V., Sabino da Silva, W., Cordeiro, L. C., & Carvalho, C. B. (2024). A comprehensive review of model compression techniques in machine learning. *Applied Intelligence*, *54*(22), 11804–11844. https://doi.org/10.1007/s10489-024-05747-w

Devlin, J., Chang, M.-W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of Deep Bidirectional Transformers for Language Understanding. *Proceedings of NAACL-HLT*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

Eichstaedt, J. C., Kern, M. L., Yaden, D. B., Schwartz, H. A., Giorgi, S., Park, G., Hagan, C. A., Tobolsky, V. A., Smith, L. K., Buffone, A., Iwry, J., Seligman, M. E. P., & Ungar, L. H. (2021). Closed- and open-vocabulary approaches to text analysis: A review, quantitative comparison, and recommendations. *Psychological Methods*, *26*(4), 398–427. https://doi.org/10.1037/met0000349

Fergus, P., & Chalmers, C. (2022). Deploying and Hosting Machine Learning Models. *Computational Intelligence Methods and Applications*, 299–317. https://doi.org/10.1007/978-3-031-04420-5_13

Fraser, K. C., Dawkins, H., & Kiritchenko, S. (2025). Detecting AI-Generated Text: Factors Influencing Detectability with Current Methods. *Journal of Artificial Intelligence Research*, *82*, 2233–2278. https://doi.org/10.1613/jair.1.16665

Gerami, S. (2023). AI Vs Human Text. Kaggle. *Kaggle*.

Ghosal, S. S., Chakraborty, S., Geiping, J., Huang, F., Manocha, D., & Bedi, A. S. (2023). Towards Possibilities & Impossibilities of AI-generated Text Detection: A Survey. *Computation and Language*. https://doi.org/10.48550/arXiv.2310.15264

Graves, A. (2012). Long Short-Term Memory. *Supervised Sequence Labelling with Recurrent Neural Networks*, *385*, 37–45. https://doi.org/10.1007/978-3-642-24797-2_4

Huang, Z., Xu, W., & Yu, K. (2015). Bidirectional LSTM-CRF Models for Sequence Tagging. *Computation and Language*. https://doi.org/10.48550/arXiv.1508.01991

Jang, B., Kim, M., Harerimana, G., Kang, S., & Kim, J. W. (2020). Bi-LSTM Model to Increase Accuracy in Text Classification: Combining Word2vec CNN and Attention Mechanism. *Applied Sciences*, *10*(17), 5841. https://doi.org/10.3390/app10175841

Jawahar, G., Abdul-Mageed, M., & Lakshmanan, V.S., L. (2020). Automatic Detection of Machine Generated Text: A Critical Survey. *Proceedings of the 28th International Conference on Computational Linguistics*, 2296–2309. https://doi.org/10.18653/v1/2020.coling-main.208

Joseph, V. R., & Vakayil, A. (2022). SPlit: An Optimal Method for Data Splitting. *Technometrics*, *64*(2), 166–176. https://doi.org/10.1080/00401706.2021.1921037

Kalyan, K. S. (2024). A survey of GPT-3 family large language models including ChatGPT and GPT-4. *Natural Language Processing Journal*, *6*, 100048. https://doi.org/10.1016/j.nlp.2023.100048

Kingma, D. P., & Ba, J. (2014). Adam: A Method for Stochastic Optimization. *Machine Learning*. https://doi.org/10.48550/arXiv.1412.6980

Krouska, A., Troussas, C., & Virvou, M. (2020). Deep Learning for Twitter Sentiment Analysis: The Effect of Pre-trained Word Embedding. *Machine Learning Paradigms*, *18*, 111–124. https://doi.org/10.1007/978-3-030-49724-8_5

Kunisch, S., Denyer, D., Bartunek, J. M., Menz, M., & Cardinal, L. B. (2023). Review Research as Scientific Inquiry. *Organizational Research Methods*, *26*(1), 3–45. https://doi.org/10.1177/10944281221127292

Lee, S., & Lee, C. (2020). Revisiting spatial dropout for regularizing convolutional neural networks. *Multimedia Tools and Applications*, *79*(45–46), 34195–34207. https://doi.org/10.1007/s11042-020-09054-7

Li, Q., Zhao, C., He, X., Chen, K., & Wang, R. (2022). The Impact of Partial Balance of Imbalanced Dataset on Classification Performance. *Electronics*, *11*(9), 1322. https://doi.org/10.3390/electronics11091322

Mitrović, S., Andreoletti, D., & Ayoub, O. (2023). ChatGPT or Human? Detect and Explain. *Computation and Language*. https://doi.org/10.48550/arXiv.2301.13852

Oruh, J., Viriri, S., & Adegun, A. (2022). Long Short-Term Memory Recurrent Neural Network for Automatic Speech Recognition. *IEEE Access*, *10*, 30069–30079. https://doi.org/10.1109/access.2022.3159339

Ott, M., Choi, Y., Cardie, C., & Hancock, J. T. (2011). Finding Deceptive Opinion Spam by Any Stretch of the Imagination. *Computation and Language*. https://doi.org/10.48550/arXiv.1107.4557

Patil, R., Boit, S., Gudivada, V., & Nandigam, J. (2023). A Survey of Text Representation and Embedding Techniques in NLP. *IEEE Access*, *11*, 36120–36146. https://doi.org/10.1109/access.2023.3266377

Petropoulos, P., & Petropoulos, V. (2024). RoBERTa and Bi-LSTM for Human vs AI Generated Text Detection. *CLEF 2024: Conference and Labs of the Evaluation Forum*, 9–12.

Pratiwi, H., Windarto, A. P., Susliansyah, S., Aria, R. R., Susilowati, S., Rahayu, L. K., Fitriani, Y., Merdekawati, A., & Rahadjeng, I. R. (2020). Sigmoid Activation Function in Selecting the Best Model of Artificial Neural Networks. *Journal of Physics: Conference Series*, *1471*(1), 012010. https://doi.org/10.1088/1742-6596/1471/1/012010

Prova, N. (2024). Detecting AI Generated Text Based on NLP and Machine Learning Approaches. *Machine Learning*. https://doi.org/10.48550/arXiv.2404.10032

Qasim, R., Bangyal, W. H., Alqarni, M. A., & Ali Almazroi, A. (2022). A Fine-Tuned BERT-Based Transfer Learning Approach for Text Classification. *Journal of Healthcare Engineering*, *2022*, 1–17. https://doi.org/10.1155/2022/3498123

Sahoo, A. R., & Chakraborty, Pavan. (2024). Hybrid CNN Bi-LSTM neural network for Hyperspectral image classification. In *Electrical Engineering and Systems Science*. https://doi.org/10.48550/arXiv.2402.10026

Smith, S. L., Kindermans, P.-J., Ying, C., & Le, Q. V. (2018). Don't Decay the Learning Rate, Increase the Batch Size. In *Machine Learning*. https://doi.org/10.48550/arXiv.1711.00489

Solaiman, I., Brundage, M., Jack, C., Openai, A. A., Herbert-Voss, A. A., Openai, A. R., Openai, G. K., Wook, J., Openai, K., Kreps, S., Politiwatch, M. M., Newhouse, Alex., Blazakis, J., Mcguffie, K., & Wang, J. (2019). *OpenAI Report Release Strategies and the Social Impacts of Language Models*. Amanda Askell. https://doi.org/10.48550/arXiv.1908.09203

Svedas, E. (2024). *Building an application using Next. js*. Metropolia University of Applied Sciences.

Svogor, I., Eichenberger, C., Spanring, M., Neun, M., & Kopp, M. (2022). Profiling and Improving the PyTorch Dataloader for high-latency Storage: A Technical Report. In *Machine Learning*. https://doi.org/10.48550/arXiv.2211.04908

Vambe, W. T., & Garidzira, T. C. (2024). A Comparative Analysis of Convolutional Neural Networks, Bi-Directional Gated Recurrent Units, and Bi-Directional Long Short-Term Memory for Sequential Data Processing Using Human Emotions Dataset. *2024 4th International Multidisciplinary Information Technology and Engineering Conference (IMITEC)*, 1–8. https://doi.org/10.1109/imitec60221.2024.10851188

van Beusekom, J., Shafait, F., & Breuel, T. M. (2010). Document inspection using text-line alignment. *Proceedings of the 9th IAPR International Workshop on Document Analysis Systems*, 263–270. https://doi.org/10.1145/1815330.1815364

Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention Is All You Need. *31st Conference on Neural Information Processing Systems (NIPS 2017*, 1–11.

Wani, M. A., ElAffendi, M., & Shakil, K. A. (2024). AI-Generated Spam Review Detection Framework with Deep Learning Algorithms and Natural Language Processing. *Computers*, *13*(10), 264. https://doi.org/10.3390/computers13100264

Yu, Y., Si, X., Hu, C., & Zhang, J. (2019). A Review of Recurrent Neural Networks: LSTM Cells and Network Architectures. *Neural Computation*, *31*(7), 1235–1270. https://doi.org/10.1162/neco_a_01199

Zellers, R., Holtzman, A., Rashkin, H., Bisk, Y., Farhadi, A., Roesner, F., & Choi, Y. (2019). Defending Against Neural Fake News. *Advances in Neural Information Processing Systems*, 1–12.

Zheng, W. (2023). *AI vs. Human: A Comparative Study of Cohesion and Coherence in Academic Texts between Human-Written and ChatGPT-Generated Texts*.