Author Identification Project Report using Long Short Term Memory (LSTM)

# Introduction

This report will documents how this project was to conducted using machine learning (ML) and deep learning (DL) for creating a model for identifying the author of a given text based on writing style. Using a dataset with a title “Spooky Author Identification” [1], the classification model was created with a high accuracy and a minimum loss of 82.78% and 0.548 respectively. The dataset contains text samples written by 3 different authors. The purpose of this project was to use Long Short Term Memory (LSTM) to accurately classify text from multiple classes. Recent studies by [2] and [3] indicated that LSTM model performs well for sequential data processing. The model trained predicts which author is most likely to have written a given sentence. This project was implemented in Python using PyTorch for DL, Pandas and Spark for data analysis, and Matplotlib and Seaborn for data visualization. Word embeddings were constructed using pretrained GloVe vectors [4] to capture semantic and syntactic relationships between words.

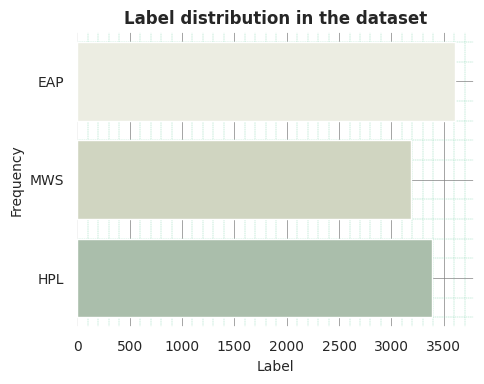
This report is organised into different sections; Section II documents methods used, Section III display and outline results and then finally, Section IV will conclude the report.

# Methods

This section of the report will document the methods that were used for data analysis, data cleaning, LSTM model training, model evaluation and model inference.

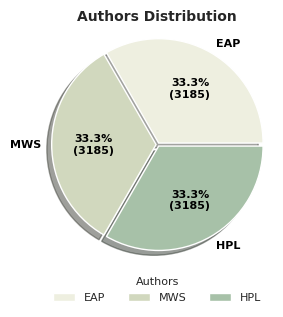
## Dataset, Data Cleaning and Preprocessing

The dataset that was used for this task was obtained from Kaggle [1]. The raw dataset contained two main files: train.csv and test.csv. In this task the train.csv file was the one that was used for model training, testing and evaluation. There were 19,579 total number of paired text with the respective to their corresponding authors. This dataset contained three columns id, text, and author [1]. Some of the text in the dataset were very short. As a data cleaning processing to improve quality all rows that has text less than 23 words were dropped together with rows that contain null values. After this process the dataset was reduces to 10,183 rows. Fig. 1 shows the distribution of labels after applying this preprocessing step.



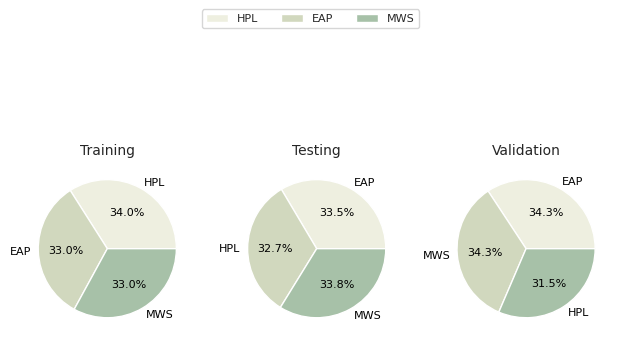
1. Label distribution in the dataset.

As shown in Fig.1, there was class misbalance between EAP, MWS and HPL authors. To resolve this down sampling was used to balance classes based on the class with least rows of data following recent study by [2], [5], [6]. According to [2], this balancing labels approach helps prevent the model from being biased toward the majority classes. Fig. 2 displays how the balanced dataset were distributed across samples.



1. Sample distribution after balancing the dataset based on author.

As shown in Fig. 2, after down sampling the dataset was balanced leading to each class having 3, 185 samples. Three (3) subsets were created from the whole dataset and the per class distribution of authors in each set are visualized in Fig. 3.



1. Per class author distribution after data set splitting.

Fig. 3 shows how the dataset was portioned, with 20% of the entire dataset reserved for the testing set, 80% of the data was used to train the model. This remaining 80% from the was rationed again into fraction of 20% and 80% for the validation and training data respectively.

Other data cleaning steps were performed to ensure data quality and consistency such as text normalization. During the text normalization process, the text was converted to lowercase, punctuation removed and special characters stripped using regular expressions following recent work by [3]. This ensures that there is uniform tokenization across authors’ text. Each text sample was tokenized, and a Counter object was used to record word frequencies. A vocabulary was created with a minimum frequency threshold of 2, ignoring rare tokens to reduce noise [3]. According to [3], a vocabulary is essential a word to index mapping that is used as a lookup table during model training and inference. Table I shows the most commonly used words by the 3 authors.

1. Most commonly used words in each set.

|  |  |  |
| --- | --- | --- |
| Training | Testing | Validation |
|  |  |  |

The most commonly used words across all the 3 sets were stopwords such as “the”, “of”, “in” and “and” as shown in Table I. Stopwords are words that carries less semantic meaning on their own [7]. During the preprocessing step stopwords were not removed because pretrained word embeddings were used following recent study by [2], [3].

## Exploratory Data Analysis (EDA)

EDA was performed using both Pandas and Spark to uncover writing patterns across authors.

1. Number of Records per Author

|  |  |
| --- | --- |
| Author | Count |
| EAP | 7, 044 |
| MWS | 5, 552 |
| HPL | 5, 451 |

1. Average Text Length per Author (10 rows)

|  |  |
| --- | --- |
| ****Author (Excerpt)**** | ****Average Length**** |
| yet satisfied me... | 3669.0 |
| nor device | 830.0 |
| “”c.; and Mr. He... | 651.0 |
| in a supplicatin... | 644.0 |
| in the minds of ... | 624.0 |
| “”she said: “”I ... | 591.0 |
| clew up all sail | 513.0 |
| and strengthen w... | 483.0 |
| “” by Mackey | 446.0 |

1. Most Common Words Overall (10 rows)

|  |  |
| --- | --- |
| ****Word**** | ****Count**** |
| the | 33,642 |
| of | 19,828 |
| and | 16,913 |
| to | 12,021 |
| a | 10,059 |
| i | 9,927 |
| in | 8,888 |
| was | 6,442 |
| that | 6,005 |
| my | 5,012 |

1. Vocabulary Diversity per Sentence

|  |  |  |  |
| --- | --- | --- | --- |
| ****Author Text (Excerpt)**** | ****Unique Words**** | ****Total Words**** | ****Vocab Ratio**** |
| and we continued... | 5 | 5 | 1.0 |
| who gave me this... | 5 | 5 | 1.0 |
| thet Afriky book?" | 1 | 1 | 1.0 |
| who art called o... | 8 | 8 | 1.0 |
| and the supposit... | 8 | 8 | 1.0 |
| turning abruptly... | 4 | 4 | 1.0 |
| and very happy | 2 | 2 | 1.0 |
| "It gave me the... | 22 | 22 | 1.0 |
| "continued Bedloe | 2 | 2 | 1.0 |
| astonished not m... | 4 | 4 | 1.0 |

Table V presents the vocabulary diversity across different author text excerpts. A ratio of 1.0 across all examples indicates that each word in the sentence is unique, meaning there is no word repetition within the same text excerpt. Such high vocabulary ratios are often found in short text segments, where repetition is rare due to brevity [8]. In the context of author identification, these features can contribute to model performance by capturing stylistic nuances such as word choice, phrasing and structural patterns which are key indicators for distinguishing between authors [8].

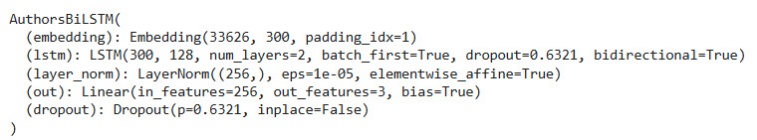
## Model Construction and Training

### Word Embeddings

To enhance the model’s ability to understand semantic relationships between words, pretrained GloVe embeddings were used [4]. The GloVe.6B.300d.txt file was loaded from google drive. The vectors were trained on approximately 6 billion words with each word represented as a 300-dimensional vector [4]. The use of these pretrained embeddings allows us to leverage prior knowledge of word semantics, which can improve model performance as supported by [2], [3]. The embeddings were aligned with the vocabulary derived from the dataset with tokens replaced with zero vectors. Studies such as [2], [3] demonstrate that pretrained embeddings significantly improve text classification accuracy, especially in low-resource domains.

### Model Architecture

A Bidirectional LSTM (BiLSTM) model was created to train on the dataset. This architecture was used because it is efficient in processing sequential data, as supported by previous studies [2], [3], [9]. Fig. 4 shows the architecture of the model that was constructed.



1. BiLSTM model architecture

Fig. 4 shows the BiLSTM model which consists of an Embedding at the top of the network. This layer was initialized with pretrained GloVe vectors [4]. Followed by bidirectional LSTM layers with 128 hidden units bidirectional and then Layer Normalization and Dropout was used to reduce overfitting during model training [10]. Finally the Fully connected linear layer mapping to 3 output classes was added to mitigate multiclass classification of 3 authors.This configuration yield the parameters that are shown in Table VI.

1. BiLSTM model parameters

|  |  |
| --- | --- |
| Parameter | Value |
| Total | 10,924,667 |
| Trainable | 10,924,667 |
| Non-Trainable | 0 |

The model had close to 11 million total parameters with all of them trainable as shown in Table VI.

### Training Process

The BiLSTM was trained for 10 iterations with data batched with a batch size of 64. Categorical accuracy was computed at each epoch. Training and validation curves showed smooth convergence with early stopping applied after performance plateaued. This BiLSTM model was trained on a google Colab instance utilizing the free cloud GPU following strategies from recent study by [11]. The GPU was done to accelerate model training and computations as highlighted by [12]. The evaluation of the model was done using metrics such as loss, accuracy, classification reports and confusion matrix. During training the model with the least validation loss was saved a common technique to retain the best model. This technique is called model checkpointing and have been a success for studies such as of [2], [3].

## Model inference

The best checkpointed model was used to create a Bot that process and analyse the text and predict the author based on the writing style in the text. The bot was fully implemented using Python programming. Table VII summarizes the technologies that was used for this task.

1. Summary of technologies that was used

|  |  |  |
| --- | --- | --- |
| ****Technology**** | ****Purpose**** | ****Reason for Use**** |
| Python 3.10 | Core language for implementing all data processing and model training tasks. | |  | | --- | | Broad ecosystem for NLP and deep learning. | |
| PySpark | Distributed EDA and large-scale text processing. | Scalability for text analytics. |
| Word Embeddings (GloVe) | 300-dimensional pretrained vectors. | Captures semantic and syntactic relationships. |
| PyTorch | Framework for building and training deep neural networks. | Flexibility and GPU acceleration. |
| Matplotlib & Seaborn | Created EDA plots and confusion matrix. | Visual interpretation of trends and performance. |
| WordCloud | Visualized most frequent author-specific words. | Intuitive pattern discovery. |
| Pandas | Reading comma seperated files. | Effective and easy to use when working with dataframe. |

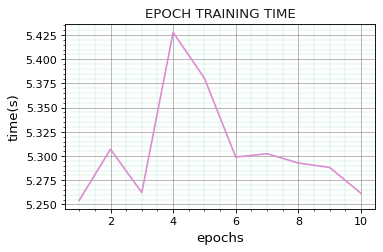
# Results

This section of the report will show the results that was obtained inform of figures and tables.

1. Training summary model training

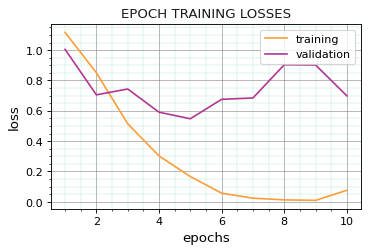
|  |  |
| --- | --- |
| Metric | Value |
| Epochs | 10 |
| Checkpointed epoch | 5 |
| Total training time | 53.08 seconds |

After training the BiLSTM model for 10 iterations, it took 53.08 seconds to complete training with the best model checkpointed on epoch 5 as shown in Table VIII. Fig. 5 shows the training time the BiLSTM model took to complete training for each epoch.



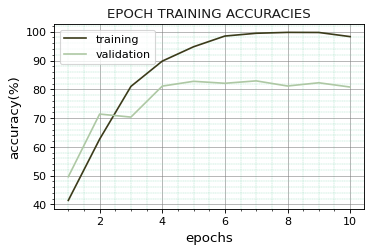
1. Training time per epoch.

Fig. 5 shows that the model took more training time on the 4th epoch as. Fig. 6 illustrate losses observed during training.



1. Training and validation losses per epoch

The model’s training loss was decreasing from the 1st to the 10th epoch while the validation loss was decreasing on a smaller rate as shown in Fig. 6. Fig. 7 shows the accuracies of the model recoded during training.



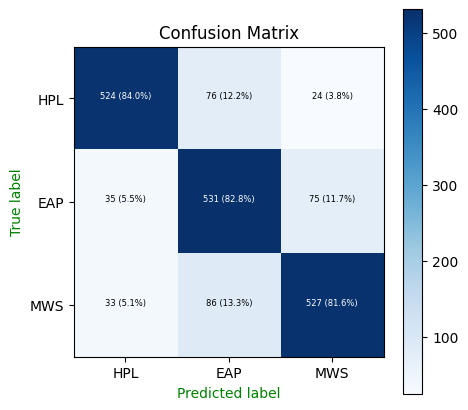
1. Training and validation accuracies.

The accuracy for both the validation and training was continuously increasing from the first epoch to the last epoch with the training accuracy leading to as shown in Fig. 7. Table shows the evaluation summary of the best saved model in terms of the loss and accuracy on the testing dataset.

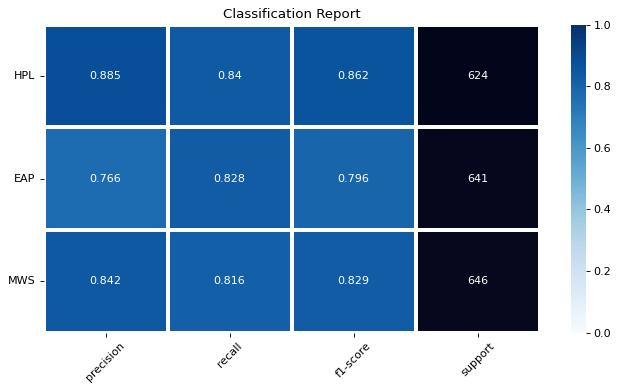
1. BiLSTM evaluation summary on the testing dataset

|  |  |
| --- | --- |
| Metric | Value |
| Loss | 0.548 |
| Accuracy | 82.78% |

The checkpointed BiLSTM model got a final accuracy of 82.78% with a minimum loss of 0.548 on the testing dataset as shown in Table IX. Fig. 7 and Fig. 8 shows the confusion matrix and classification report respectively of the checkpointed model evaluated after being evaluated on the “testing dataset”.



1. Best BiLSTM model confusion matrix.



1. Best BiLSTM model classification report.

Fig. 10 displays a screenshot for the Bot response to the query about the text of an author.

A screenshot of a computer

AI-generated content may be incorrect.

1. Bot inference query and response

# Conclusion

This study successfully developed an author identification system using a BiLSTM model to effectively distinguishes between the writing styles using the dataset by [1] from Kaggle. The model used pretrained GloVe embeddings to capture the semantic and syntactic relationships within text [4] while data preprocessing techniques such as normalization, tokenization, and down-sampling ensured data quality and balance [5], [6]. Exploratory data analysis revealed distinctive linguistic and stylistic differences among authors, highlighting the model’s potential in capturing subtle variations in vocabulary and structure [8]. The implementation in PyTorch, supported by GPU acceleration, facilitated efficient model training, while the use of dropout layers effectively minimized overfitting [10].

The BiLSTM model achieved a classification accuracy of 82.78% on the test dataset, confirming its capability to process sequential text data efficiently and extract author-specific features [9]. This performance aligns with recent findings that LSTM-based architectures outperform traditional models in text classification tasks [2], [3]. Furthermore, the integration of model checkpointing ensured optimal parameter selection and reliable predictions during inference. Overall, the study demonstrates that DL techniques, particularly BiLSTM with pretrained embeddings, are highly effective for author identification.

# References

1. M. Risdal and R. Tatman, “Spooky Author Identification,” 2017.
2. S. Ngwenya and T. C. Garidzira, “Enhanced Fake News Detection with Domain-Specific Word Embeddings: A TorchText-Based Method for News Semantics Representation.,” *The Indonesian Journal of Computer Science*, vol. 14, no. 4, Jul. 2025, doi: 10.33022/ijcs.v14i4.4831.
3. W. T. Vambe and T. C. Garidzira, “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,” *Proceedings of 2024 4th International Multidisciplinary Information Technology and Engineering Conference, IMITEC 2024*, pp. 531–538, 2024, doi: 10.1109/IMITEC60221.2024.10851188.
4. J. Pennington, R. Socher, and C. Manning, “Glove: Global Vectors for Word Representation,” in *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, Stroudsburg, PA, USA: Association for Computational Linguistics, 2014, pp. 1532–1543. doi: 10.3115/v1/D14-1162.
5. C. Padurariu and M. E. Breaban, “Dealing with Data Imbalance in Text Classification,” *Procedia Comput Sci*, vol. 159, pp. 736–745, 2019, doi: 10.1016/j.procs.2019.09.229.
6. J. Jang, Y. Kim, K. Choi, and S. Suh, “Sequential targeting: A continual learning approach for data imbalance in text classification,” *Expert Syst Appl*, vol. 179, p. 115067, Oct. 2021, doi: 10.1016/j.eswa.2021.115067.
7. S. Sarica and J. Luo, “Correction: Stopwords in technical language processing,” *PLoS One*, vol. 19, no. 12, p. e0315195, Dec. 2024, doi: 10.1371/journal.pone.0315195.
8. F. Zenker and K. Kyle, “Investigating minimum text lengths for lexical diversity indices,” *Assessing Writing*, vol. 47, p. 100505, Jan. 2021, doi: 10.1016/j.asw.2020.100505.
9. S. Siami-Namini, N. Tavakoli, and A. S. Namin, “The Performance of LSTM and BiLSTM in Forecasting Time Series,” in *2019 IEEE International Conference on Big Data (Big Data)*, IEEE, Dec. 2019, pp. 3285–3292. doi: 10.1109/BigData47090.2019.9005997.
10. N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, “Dropout: a simple way to prevent neural networks from overfitting,” *J. Mach. Learn. Res.*, vol. 15, no. 1, pp. 1929–1958, Jan. 2014.
11. V. Sharma, G. K. Gupta, and M. Gupta, “Performance Benchmarking of GPU and TPU on Google Colaboratory for Convolutional Neural Network,” 2021, pp. 639–646. doi: 10.1007/978-981-33-4604-8\_49.
12. M. Khairy, Z. Shen, T. M. Aamodt, and T. G. Rogers, “Accel-Sim: An Extensible Simulation Framework for Validated GPU Modeling,” in *2020 ACM/IEEE 47th Annual International Symposium on Computer Architecture (ISCA)*, IEEE, May 2020, pp. 473–486. doi: 10.1109/ISCA45697.2020.00047.