Authorship Attribution for English Short Texts

Tawfeeq Alsanoosy
Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah 42353, Saudi Arabia
tsanoosy@taibahu.edu.sa (corresponding author)

Bodor Shalbi
Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah 42353, Saudi Arabia
TU4359248@taibahu.edu.sa

Ayman Noor
Department of Computer Science, College of Computer Science and Engineering, Taibah University, Madinah 42353, Saudi Arabia
anoor@taibahu.edu.sa

Received: 4 July 2024 | Revised: 23 July 2024 | Accepted: 26 July 2024

Licensed under a CC-BY 4.0 license | Copyright (c) by the authors | DOI: https://doi.org/10.48084/etasr.8302

ABSTRACT

Internet and social media explosive growth has led to the rapid and widespread dissemination of information, which often takes place anonymously. This anonymity has fostered the rise of uncredited copying, posing a significant threat of copyright infringement and raising serious concerns in fields where verifying information's authenticity is paramount. Authorship Attribution (AA), a critical classification task within Natural Language Processing (NLP), aims to mitigate these concerns by identifying the original source of content. Although extensive research exists for longer texts, AA for short texts, namely informal texts like tweets, remains challenging due to the latter's brevity and stylistic variation. Thus, this study aims to investigate and measure the performance of various Machine Learning (ML) and Deep Learning (DL) methods deployed for feature extraction from short text data, using tweets. The employed feature extraction methods were: Bag-of-Words (BoW), TF-IDF, n-grams, word-level, and character-level features. These methods were evaluated in conjunction with six ML classifiers, i.e. Naive Bayes (NB), Support Vector Machine (SVM), Decision Tree (DT), Logistic Regression (LR), K-Nearest Neighbors (KNN), and Random Forest (RF) along with two DL architectures, i.e. Convolutional Neural Networks (CNNs) and Recurrent Neural Networks (RNNs). The highest accuracy achieved with an ML model was 92.34%, using an SVM with TF-IDF features. Even though the basic CNN DL model reached 88% accuracy, this outcome still surpassed the previously established baseline for this task. The findings of this research not only advance the technical capabilities of AA, but also extend its practical applications, providing tools that can be adapted across various domains to ensure proper attribution and expose copyright infringement.

Keywords-natural language processing; authorship attribution; machine learning; deep learning; authorship identification

I. INTRODUCTION

The digital age has witnessed an explosion of textual data in various formats, ranging from news articles and blogs to social media posts and tweets. This rapid expansion has brought about the issue of unattributed content, with individuals neglecting to acknowledge the original authors. As a result, cybercriminals are increasingly exploiting the aforementioned issue to create anonymous, fake, and illegal content or to perform illegal activities, including cyberattacks and information theft [1]. Social media platforms like Twitter and Facebook have further exacerbated the issue of misinformation and propaganda. The popularity of these platforms makes it difficult to ensure the accuracy and reliability of information sources [2].

Recent studies [3], have delved into the behaviors of scammers on social media platforms, revealing that a significant portion of spam accounts, i.e. 84%, are compromised accounts manipulated by scammers rather than automated bots, which constitute the rest 16%. In an attempt to provide credibility to malicious activity, these compromised accounts are used to establish connections with legitimate profiles, including those of celebrities and public figures. Consequently, incidences of cybercrime and copyright violations continue to rise steadily, underscoring the urgent need for effective solutions. AA approaches have emerged as promising tools in addressing these challenges. AA aims to determine the authorship of texts by analyzing distinctive writing styles and linguistic patterns [4-6]. It represents a critical classification challenge in natural language processing, leveraging ML models to recognize and attribute texts to their rightful authors [7]. This technique has found applications in diverse fields, such as literary studies [8-15], forensics, copyright protection, and cybercrime investigation [16-18] to effectively analyze the content of a given text. Moreover, the effectiveness of AA methods hinges on several critical factors, including the quality and diversity of features extracted from texts, the size of training datasets, as well as the length and complexity of the texts themselves [19-20].

Twitter is a prominent social media platform of active research and a rich source of public posts [41]. It has been ranked as the fourth most popular social media platform in the USA [21]. It has around 330 million active users monthly, while nearly 500 million tweets are posted on it daily. However, despite its potential, AA on Twitter remains a challenging task. Twitter users often employ informal language and slang, making it difficult for the author to be identified solely based on linguistic features. This paper aims to develop and evaluate the performance of ML and DL models specifically designed for AA on short text formats, focusing particularly on Twitter data. Several classifiers are commonly utilized for AA, including SVM [22-23], RF [24], KNN [25], and Long Short-Term Memory (LSTM) [26]. However, the results can be affected by various parameters, such as the limited length of the text (tweets), or the features and methods used. Most studies are conducted with only one or two ML or DL classifiers and confined feature sets. Previous studies have shown that AA methods perform best when applied to either smaller text collections or texts within specific genres [4]. Building upon this knowledge, the current research aims to develop an AA model specifically tailored to identify tweet authors. This study distinguishes itself by conducting multiple experiments with different features and ML and DL models. It will explore various feature sets, entailing character-level, n-grams, BoW, TF-IDF, and word-level embedding features, analyzing their effectiveness through rigorous computer experiments. Furthermore, a group of ML and DL classifiers will be deployed. By evaluating the performance of these techniques, this study aims to shed light on the quality of features extracted from Twitter datasets and their impact on AA tasks.

This study's experiments have led to new and improved results using specific ML and DL methods, which have surpassed the established baselines for the datasets utilized. The findings provide valuable insights for both researchers and practitioners involved in AA, particularly in the analysis of short texts and social media. The current work offers empirical evidence integrating N-gram, BoW, TF-IDF, word-level and character-level, which help to develop accurate methodologies and tools for identifying authors, which is crucial for combating misinformation and malicious activities on digital platforms. This contribution is particularly beneficial for enhancing the accuracy of AA methods, gaining insights into the factors that influence model accuracy, and offering detailed knowledge about the effectiveness of using different extracted features on model accuracy and reliability enhancement.

II. RELATED WORKS

Authors in [21] developed a model for classifying tweet authors using LR and NB classifiers. They collected 46,895 tweets, limited to three thousand tweets per author, comprising two classes: known and unknown authors. Known authors included 12 celebrities and prominent Twitter users, whereas unknown authors comprised 12 regular Twitter users with less popularity. The study involved fetching tweets, preprocessing, feature extraction, and applying classification algorithms, resulting in an LR based classifier, which achieved an accuracy of 91.1% compared to the NB classifier, which obtained 89.8% accuracy. However, the authors of that study used a small dataset, unlike the one adopted in the current work, which contains 7,000 authors. Authors in [27] proposed a model utilizing a Multi-Layer Perceptron (MLP), which is a feedforward Artificial Neural Network (ANN), to analyze features like word choice, punctuation use, and function words to identify the author of a given text. The authors trained the model on a collected tweet dataset. They gathered 400 tweets from 20 selected authors. The results demonstrated that the model can achieve a high accuracy of 96% in attributing authorship to the correct author based on stylistic patterns in short texts. Moreover, the results displayed that as the number of authors increases, there was a drop in the accuracy, with the latter being 67% for 20 authors. In the current work, a larger dataset was adopted, while six ML classifiers and two DL models were deployed. The results outperformed those of the model implemented in [27] even though the current study's model was trained with 50 authors. Authors in [31] investigated the use of feature extraction techniques for AA tasks. They employed SVM, NB, RF, and MLP, to classify micro-texts. The results revealed that the MLP classifier achieved the highest accuracy. Authors in [32] evaluated LR, RF, DT, NB, SVM, and KNN, for text binary and multi-classifications for the AA task. The findings disclosed that SVM outperformed the other models. A limited dataset was employed in that work, consisting of only four Chinese authors. This may lead to decreased accuracy when applied to a larger number of authors. Authors in [33] proposed a model using CNNs with a mixture of word and character n-grams to represent the text. The model exhibited improvements in identifying authors for short texts. Including latent posting styles further enhances CNN and LSTM models, even in complex scenarios with varying authors or fewer samples per author. The proposed approach combined character n-grams with ANNs, leveraging character-level features to improve author identification based on writing styles. The best method developed in that study achieved an accuracy of 83.6%. Nevertheless, the authors only used the character n-grams feature. In this paper, four feature extraction methods were followed: BoW, TF-IDF, n-grams, word-level and character-level features, and their performances were compared. Authors in [28] proposed a framework called DeepStyle to capture the unique writing style of each user through employing various embedding structures, including character, word n-gram, and POS tags, within a CNN model. They utilized multi-view representations of a user's post and triplet loss to learn how to differentiate between the writing styles of different users. The experiment results manifested that DeepStyle outperformed some models for the AA task on short texts.

Authors in [34] proposed a multi-channel CNN approach for micro-message author identification. The focus was on comparing CNNs architecture with character n-gram embeddings to multi-channel CNNs with pre-trained word embeddings. They combined word embeddings and character embeddings in a single CNN architecture, treating them as separate information channels on the Twitter dataset used in [29]. The results showed 74.50% accuracy for a set of 50 authors and 500 tweets per author. It was exhibited that CNNs that used pre-trained word embeddings performed better than the CNNs with character-level embeddings on a Twitter dataset. However, the current study's CNNs model displayed higher accuracy, reaching 88%. Authors in [28] deployed a capsule with a CNN model over character n-grams and a KNNs model for the AA task for short texts using the dataset applied in [29]. They conducted experiments comparing different text representations, such as BERT embedding, character bigram and character unigram. The results revealed that character unigrams achieved an accuracy of 86.62%, while character bigrams reached 83.82% accuracy by adopting the capsule-based CNN architecture. However, the performance of KNNs underperformed that of the CNN over character unigrams and outperformed CNN performance over bigrams. Once, again the accuracy achieved by the present study's CNNs model was higher. Authors in [30] presented a model to identify the author of a short online text through using Regularized Deep Neural Network (RDNN). The proposed method consisted of three components: CNN character-level layer, Distributed Highway Network (DHN), and Bidirectional Long Short-Term Memory (BLSTM). Four datasets were implemented for the model to be evaluated. The experimental results demonstrated that the RDNN achieved superior performance. The highest accuracy and F1-score values attained on the CCAT50 dataset were 93.20% and 92.20%, respectively. In this paper, the highest accuracy achieved with the ML model was 92.34%, using an SVM with TF-IDF features.

III. RESEARCH METHODOLOGY AND MODEL PROPOSAL

The proposed methodology to build the AA model is divided into five steps, as can be seen in Figure 1:

Step 1: Dataset selection.

Step 2: Dataset pre-processing.

Step 3: Feature extraction.

Step 4: Model training.

Step 5: Model evaluation.

A. Data Selection

The dataset provided in [29] was utilized. The particular dataset has been previously employed in several studies, for instance, in [18, 31, 34, 35]. The dataset comprises approximately 15% of all public tweets created from May 2009 to March 2010. It includes a total of 7,000 authors, 50 of whom were randomly selected. Each author contributed 1,000 tweets to the dataset. The dataset contains a total of 711360 words, with an average of 14.22 words per tweet. The tweets in the dataset have varying lengths, ranging from a minimum of 2 words to a maximum of 35 words. The shortest tweets have 25 characters, whereas the longest tweets contain 149 characters.

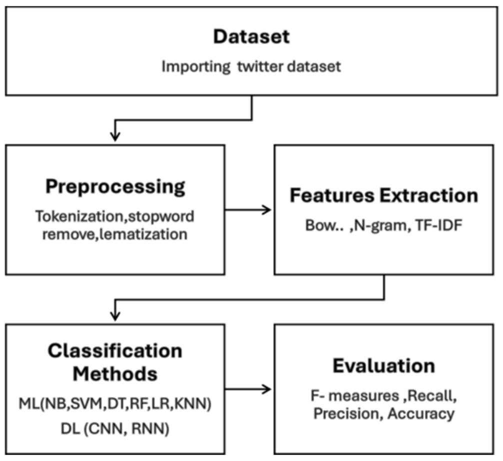

Fig. 1. The main steps for the proposed AA evaluation performance models.

B. Data Preprocessing

Data preprocessing is essential for extracting features from text data and utilizing them in calculations. The specific approach to preprocessing may vary across different research studies. Typically, it involves removing undesirable elements like hyperlinks, stop words, and outliers that lack relevance in text classification. Previous research has highlighted the significance of handling stop words, URLs, and other symbols. In this study's text classification task, the preprocessing stage was initiated by importing Python libraries. The primary libraries used for preprocessing short text datasets in ML and DL models entail NLTK, Scikit-learn, SpaCy, TextBlob. These libraries offer functionalities for tokenization, stop word removal, stemming, lemmatization, and vectorization, which can be customized based on specific task requirements. To assess the impact of preprocessing, the current study experimented with various techniques across ML and DL methods.

In the ML methods, the following text preprocessing techniques were deployed:

- Tokenization: This step breaks down the text into individual tokens, which are typically words.
- Stop word removal: Stop words such as "is," "the," and "and," were removed.
- Lemmatization: This process aims to reduce inflectional variations of words to their base form (lemma), which helps improve model performance by focusing on the core meaning of the words.
- Joining tokens: After preprocessing, the individual tokens are often joined back together into a single string.

Interestingly, for the conducted DL experiments, it was found that removing stop words and URLs during preprocessing actually had a negative impact on the model's performance, as reported in [18]. Consequently, a different preprocessing approach for this study's DL model was adopted:

- Lowercasing: All text is converted to lowercase characters.
- Label Encoding: Unique labels in the dataset are assigned numerical values using a dictionary mapping. This allows the model to handle categorical labels more effectively.
- Tokenization: Similar to ML preprocessing, the text data are broken down into tokens using a tokenizer function, such as Keras Tokenizer.
- Padding Sequences: To ensure that all input sequences have the same length for the neural network, shorter sequences are padded with additional elements, namely zeros.
- One-Hot Encoding: Labels are converted to a one-hot encoded representation using a function, like to_categorical, from Keras.

C. *Feature Extraction*

After the preprocessing steps, different feature extraction methods were implemented in the proposed models, tailored to the specific requirements of both ML and DL models. In ML, this study experimented with three techniques: BoW, TF-IDF, and N-grams features. Furthermore, it investigated various combinations of TF-IDF with N-grams, including unigram + bigram, unigram + trigram, bigram + trigram, and unigram + 4-gram. A pipeline for each classifier, where each pipeline consists of different combinations of feature extraction methods along with their corresponding vectorizers, was identified. In DL, word-level embeddings were utilized, which involves adding an embedding layer to the proposed DL models. The parameters were:

- input_dim: It specifies the size of the vocabulary, which is determined by the length of the ((tokenizer.word_index) + 1). Ensuring that the input dimension of the embedding layer matches the size of the vocabulary.
- output_dim: It defines the dimensionality of the dense embedding vectors. In this instance, each word is represented by a number of dimensional vectors.
- input_length: It determines the length of input sequences, set to (max_len). It specifies the length of the input sequences that will be fed into the embedding layer.

Therefore, a word embedding layer was created. This layer converts input word indices into dense word vectors of dimension suitable for word-level representations in the introduced CNN and RNN models.

D. *Model Training*

The considered dataset was divided into training and testing data, where 80% of the data were randomly assigned for training, leaving the remaining 20% for testing. A total of 900 tweets from each set of 1000 tweets for the 50 authors were randomly sampled to ensure unbiased evaluation of the model's performance. The source codes are available in [42].

For the ML models, a classifier Pipeline was employed to define multiple combinations of classifiers and vectorizers for text classification tasks. The classifier pipeline comprises the following ML classifiers: NB, RF, SVM, DT, LR, and KNN. Each classifier was paired with a set of vectorizers to extract features from the text data. The vectorizers used include BoW, TF-IDF, Ngram (1, 2), and a combination of TF-IDF and N-gram. This combination includes (1, 3), (2, 3), and (1, 4), providing insights into their effectiveness in the AA task.

For the DL models, a basic CNN model was initially implemented, consisting of four layers:

1) Input Layer (Embedding Layer): It transforms the input sequences into dense vectors with a fixed dimensionality.
2) Convolutional Layer: It employs 500 filters with a kernel size of 5. It performs convolution operations along the temporal dimension of the input sequence to extract relevant features. The ReLU activation function is subsequently applied to introduce non-linearity to the extracted features.
3) Pooling Layer: It serves to reduce the dimensionality of the feature maps generated by the convolutional layer. It takes the maximum value from each feature map, resulting in a global representation of the input sequence.
4) Dense Layer (Output Layer): It contains 50 units and utilizes the softmax activation function. It outputs the class probabilities for the classification task, with the number of units corresponding to the number of classes within the dataset.

Table I summarizes the architecture of this CNN model (CNN-B), including the number of layers and their hyperparameters. The model was compiled by specifying the following: optimizer: Adam, loss function: categorical cross-entropy, evaluation metric: accuracy.

TABLE I. CNN-B HYPERPARAMETERS

Layer type	No. of layers	Hyperparameters
Input (embedding)	1	Input_dim = vocabulary size, output_dim=300
Bidirectional (SimpleRNN)	2	Hidden units = 300, learning rate = 0.01
Dense	1	Units = 50, activation = softmax

Then the RNN model was utilized. It consists of three layers:

1) Embedding Layer: it determines the input dimension size based on the vocabulary size (length of tokenizer.word_index plus 1). It converts word indices into dense word vectors with a dimensionality of 300. This dimensionality specifies the richness of the learned word representations.
2) SimpleRNN Layers: A bidirectional wrapper allows the SimpleRNN layer to process the input sequence in both forward and backward directions. This effectively doubles

the number of units available for capturing information within the sequence. The underlying SimpleRNN layer has a specified number of hidden units (hidden_units), a dropout rate for regularization, and an option to return sequences or just the final output.

3) Dense Layer: The final dense layer has 50 units, representing the number of output classes. The softmax activation function is put into service to output a probability distribution over these classes.

Table II summarizes the SimpleRNN model architecture, involving the number of layers and their hyperparameters. The proposed RNN model's architecture consists of an embedding layer, followed by two bidirectional SimpleRNN layers. The hyperparameters used in the model were: embedding dimension: 256, number of hidden units in the RNN layer: 300, dropout rate: 0.5, and learning rate: 0.01. The model was compiled by specifying the following: optimizer: Adam, loss function: categorical cross-entropy, evaluation metric: accuracy (similar to the CNN-B model), batch size: 64, number of epochs: 20. This was finalized with a dense layer for classification.

TABLE II. CNN-B HYPERPARAMETERS

Layer Type	No. of layers	Hyperparameters
Input (embedding)	1	input_dim = vocabulary size
Conv1D	1	filters = 500, kernel_size = 5
GlobalMaxPooling1D	1	default
Dense	1	units = 50, activation = softmax

E. Model Evaluation

To evaluate the model's performance, the metrics defined in (1)-(4) were employed. These equations utilize the following notations: TP (True Positive), TN (True Negative), FP (False Positive), and FN (False Negative) [36-37]:

$$\text{Precision} = \frac{\text{TP}}{(\text{TP+FP})} \quad (1)$$

$$\text{Recall} = \frac{\text{FP}}{(\text{FP+TN})} \quad (2)$$

$$\text{Accuracy} = \frac{(\text{TP+TN})}{(\text{TP+TN+FP+FN})} \quad (3)$$

$$\text{F1 score} = 2 * \frac{\text{Precision} * \text{Recall}}{\text{Precision} + \text{Recall}} \quad (4)$$

IV. RESULTS AND DISCUSSION

This work presented a novel approach to AA for short texts by comprehensively evaluating multiple feature extraction methods alongside various ML and DL models, achieving improved accuracy. Table III illustrates the results of the models across all feature extraction methods used. In the initial experiment series, the BoW technique was utilized for each classifier. The findings indicated that the SVM classifier achieved the highest accuracy of 90.95%, followed by LR attaining an accuracy of 89.01%, RF obtaining 86.84% accuracy, while the NB and DT classifiers achieved 84% accuracy, with NB showing an increase of 0.52% over DT. Conversely, the KNN classifier exhibited the lowest accuracy, recording 49.82%. The impact of TF-IDF was further investigated by conducting an experiment within the same pipeline, revealing an improvement in accuracy for some ML models. The SVM classifier again attained the highest accuracy of 92.34%, LR achieved 87.22%, surpassing NB and RF by 0.44% and 0.8%, respectively. Additionally, the DT classifier achieved an accuracy of 81.48%, which was an improvement over KNN, resulting in 70.81% accuracy. Utilizing the combination of N-grams (unigram + bigram) vectorizers, the SVM classifier demonstrated the highest accuracy of 91.11%, followed by LR with 89% accuracy. The KNN classifier again yielded the lowest accuracy in this combination.

TABLE III. ACCURACY RESULTS OF THE ML CLASSIFIERS

Models	Feature extraction methods					
	BoW	TF-IDF	Unigram and bigram	TF-IDF + unigram to trigram	TF-IDF + bigram and trigram	TF-IDF + unigram to four grams
NB	84.55	86.78	86.67	88.22	74.84	88.28
SVM	90.95	92.34	91.11	90.96	75.50	90.67
RF	86.84	86.42	86.92	84.38	70.92	84.53
DT	84.03	81.48	83.94	78.01	67.04	77.83
LR	89.01	87.22	89.00	86.96	73.52	86.70
KNN	49.28	70.81	39.18	68.15	27.81	67.42

In the final experiment, TF-IDF was combined with various N-grams, including unigram + bigram, unigram + trigram, bigram + trigram, and unigram to four-gram. The results indicated improved accuracy for some models. Once again, the SVM classifier outperformed the others, achieving an accuracy of 90.96% with the combination of TF-IDF and unigram trigram. It also attained accuracies of 90.67% with the combinations of TF-IDF and unigram to four-gram. Figure 2 visually compares the performance of SVM, NB, and RF models using BoW, TF-IDF, and various N-gram combinations. The results reveal that SVM achieves the highest accuracy when using TF-IDF with unigrams and bigrams. Additionally, both RF and NB exhibit consistently noticeable accuracy, particularly when employing the combination of TF-IDF and N-grams.

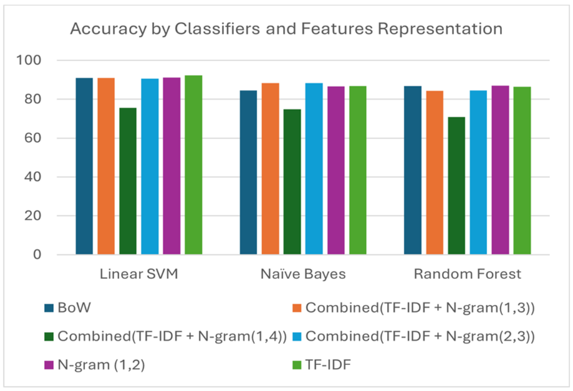

Fig. 2. SVM, NB, and RF model accuracy using BoW, TF-IDF feature extraction with different N-grams.

The proposed SVM model achieves superior performance compared to related studies. For example, authors in [38] reported an accuracy of 68.67%, which is approximately 24% lower than that of the presented SVM model, using BoW and stylometric features. Similarly, authors in [39] achieved 70% accuracy, 22% lower than the best result obtained by the SVM introduced in the present study. Although the proposed model performs well, it falls slightly short of the NB model presented in [2], which achieved an accuracy of 84% (a difference of 5%). It is worth noting that this study's RF classifier with (1,2)-grams achieved an accuracy of 86.92%. The experiments carried out consistently demonstrated the superiority of the SVM across all methods, particularly when combined with TF-IDF and n-grams. This suggests that SVMs have a robust capability for handling high-dimensional, sparse textual data. Conversely, KNN performance suffered significantly due to the high dimensionality, which often poses a challenge for distance-based models. Furthermore, the use of enhanced feature extraction methods, such as a combination of TF-IDF with different n-grams, proved beneficial. This highlights the importance of sophisticated textual representations in distinguishing individual writing styles. The derived results confirm that n-gram features can effectively identify unique writing styles, which is crucial for author identification. N-grams achieved relatively high accuracy in author detection tasks. Additionally, TF-IDF enhances the distinction between common language usage and specific stylistic or thematic choices that might characterize an author's writing. In the conducted ML experiments, TF-IDF yielded the best results when implemented with Naive Bayes and SVM classifiers. However, these results were less effective when combined with higher-order n-grams.

In DL models, a CNN-B model and a simple RNN model were employed to analyze a dataset comprising 1,000 tweets from each of 50 authors. The CNN-B model achieved an accuracy of 88%, significantly outperforming the simple RNN's 59% accuracy, as evidenced in Table IV. This underscores the superior capability of CNN-B in capturing local contextual features from textual data, which is crucial for distinguishing between authors.

TABLE IV. CNN-B AND SIMPLE RNN ACCURACY

Number of tweets per user	CNN-B	RNN
1000	88	59

This study's CNN-B model demonstrates superior performance in AA when compared to several previous studies. Notably, authors in [13] achieved a 74.5% accuracy using both word and character embeddings in their CNN. The proposed CNN-B model exceeds these results by 12.9% and 13.5%, respectively. Additionally, it surpasses the main baseline set by authors in [18], who documented an 86% accuracy with character n-grams, marking a 2% improvement in the proposed model's performance. Figures 3 and 4 portray the number of epochs versus loss, and the number of epochs versus accuracy, respectively. The optimal performance is observed at around 4 epochs, indicating both minimal loss and maximal accuracy at this point in training.

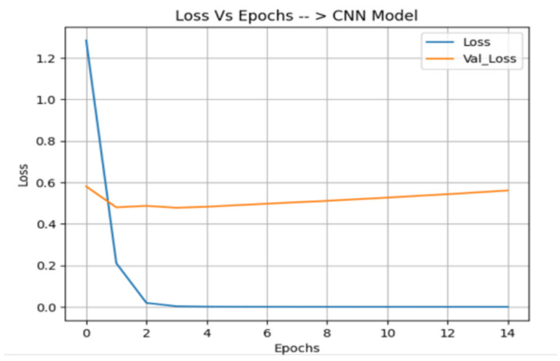

Fig. 3. Loss vs epochs for the CNN-B model.

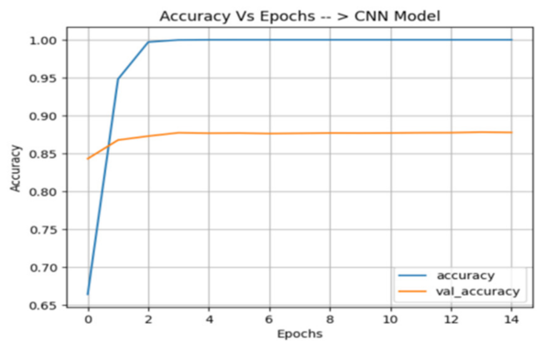

Fig. 4. Accuracy vs epochs for CNN-B model.

The Simple RNN model introduced in this study, exhibited lower performance, achieving only 59% accuracy. This contrasts sharply with the findings of the authors in [40], who reported 89.6% accuracy, using an RNN with LSTM enhanced by skip-gram word embedding-level features. This notable disparity underscores the limitations of the proposed RNN's simpler structure and points to a need for further experimentation with features to improve accuracy. Performance metrics for the presented RNN model, as depicted in Figures 5 and 6, reveal the number of epochs versus loss, and the number of epochs versus accuracy, respectively, for training and validation phases.

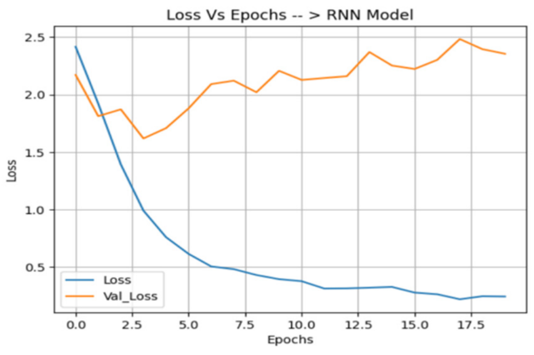

Fig. 5. Loss vs epochs for the Simple RNN model.

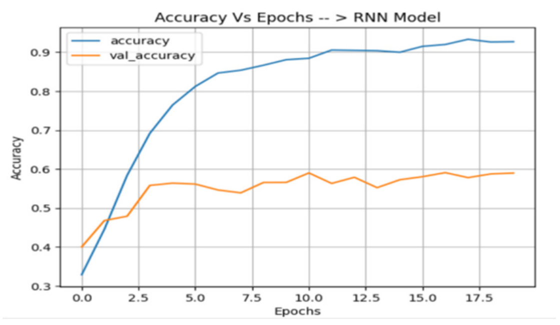

Fig. 6. Accuracy vs epochs for Simple RNN model.

The optimal results, both the lowest loss and the highest accuracy, are achieved at around 17.5 epochs, suggesting that this is the most effective point for training the model under current configurations. The modest success rate of the proposed RNN model sheds light on the potential challenges RNNs face when dealing with tweet datasets for AA. Even though RNNs are generally proficient in processing sequential data, they might falter with the short, noisy, and highly stylized text, typical of tweets. This could stem from difficulties in managing long-term dependencies or from an inability to effectively capture broad contextual cues, essential for this task.

V. CONCLUSION

This paper evaluated the performance of both ML and DL models on a standard Twitter dataset, which included 1000 tweets from each of 50 users and was derived from [29]. It experimented with six ML methods: NB, SVM, DT, LR, RF, and KNN. These methods were highlighted for their effectiveness with linguistic features BoW, TF-IDF, N-grams, and combinations thereof, particularly TF-IDF with N-gram (1,2) features. Then, this study experimented with a DL approach, including the use of a CNN and a Simple RNN with word-level embedding features. The highest accuracy, 92.34%, was achieved by SVM using TF-IDF. There is also a significant improvement in classification results reaching 98.01% using BoW for LR, and 88.28% for NB using TF-IDF + n-gram (1+4). The basic CNN model achieved 88% accuracy, outperforming the previous baseline on the same dataset. The simpler architecture of the Simple RNN achieved only 59% accuracy, indicating a need for enhancements through more layers and optimized hyperparameters. The findings demonstrate the significant benefits of implementing ML and DL methods for effectively detecting complex patterns within textual data. Furthermore, the study emphasizes the potential impact of minimizing data preprocessing steps on model performance. These findings hold significant value in developing sophisticated tools for analyzing social media content and ultimately improving the overall integrity of online information.

REFERENCES

[1] S. Shao, C. Tunc, A. Al-Shawi, and S. Hariri, "An Ensemble of Ensembles Approach to Author Attribution for Internet Relay Chat Forensics," *ACM Transactions on Management Information Systems*, vol. 11, no. 4, Jul. 2020, Art. no. 24, https://doi.org/10.1145/3409455.

[2] L. Chen, E. Gonzalez, and C. Nantermoz, "Authorship Attribution with Limited Text on Twitter," 2017, [Online]. Available: https://cs229.stanford.edu/proj2017/final-reports/5241953.pdf.

[3] C. Grier, K. Thomas, V. Paxson, and M. Zhang, "@spam: the underground on 140 characters or less," in *17th ACM Conference on Computer and Communications Security*, Chicago, IL, USA, Oct. 2010, pp. 27–37, https://doi.org/10.1145/1866307.1866311.

[4] K. Lagutina and N. Lagutina, "A Survey of Models for Constructing Text Features to Classify Texts in Natural Language," in *29th Conference of Open Innovations Association*, Tampere, Finland, Dec. 2021, pp. 222–233, https://doi.org/10.23919/FRUCT52173.2021.9435512.

[5] E. Aydogan and S. Sen, "Android Authorship Attribution Using Source Code-Based Features," *IEEE Access*, vol. 12, pp. 6569–6589, Jan. 2024, https://doi.org/10.1109/ACCESS.2024.3351945.

[6] A. Fedotova, A. Kurtukova, A. Romanov, and A. Shelupanov, "Semantic Clustering and Transfer Learning in Social Media Texts Authorship Attribution," *IEEE Access*, vol. 12, pp. 39783–39803, Jan. 2024, https://doi.org/10.1109/ACCESS.2024.3377231.

[7] W. Zheng and M. Jin, "A review on authorship attribution in text mining," *WIREs Computational Statistics*, vol. 15, no. 2, 2023, Art. no. e1584, https://doi.org/10.1002/wics.1584.

[8] R. Azimov, "Analysis of the Use of Methods and Feature Groups for Author Recognition on the Example of Texts in the Azerbaijani Language," in *5th International Conference on Problems of Cybernetics and Informatics*, Baku, Azerbaijan, Aug. 2023, pp. 1–4, https://doi.org/10.1109/PCI60110.2023.10325956.

[9] E. Ferracane, S. Wang, and R. Mooney, "Leveraging Discourse Information Effectively for Authorship Attribution," in *Eighth International Joint Conference on Natural Language Processing*, Taipei, Taiwan, Dec. 2017, pp. 584–593.

[10] R. Hou and C.-R. Huang, "Robust stylometric analysis and author attribution based on tones and rimes," *Natural Language Engineering*, vol. 26, no. 1, pp. 49–71, Jan. 2020, https://doi.org/10.1017/S135132491900010X.

[11] M. Kestemont *et al.*, "Overview of the author identification task at PAN-2018: Cross-domain authorship attribution and style change detection," in *Conference and Labs of the Evaluation Forum*, Avignon, France, Sep. 2018, pp. 1–25.

[12] M. Llorens and S. J. Delany, "Deep Level Lexical Features for Cross-lingual Authorship Attribution," in *First Workshop on Modeling, Learning and Mining for Cross/Multilinguality*, Padova, Italy, Mar. 2016, https://doi.org/10.21427/r8v7-fv65.

[13] S. Ruder, P. Ghaffari, and J. G. Breslin, "Character-level and Multi-channel Convolutional Neural Networks for Large-scale Authorship Attribution." arXiv, Sep. 21, 2016, https://doi.org/10.48550/arXiv.1609.06686.

[14] Y. Sari, A. Vlachos, and M. Stevenson, "Continuous N-gram Representations for Authorship Attribution," in *15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain, Apr. 2017, pp. 267–273.

[15] A. Sharma, A. Nandan, and R. Ralhan, "An Investigation of Supervised Learning Methods for Authorship Attribution in Short Hinglish Texts using Char & Word N-grams." arXiv, Dec. 26, 2018, https://doi.org/10.48550/arXiv.1812.10281.

[16] K. Y. Tai, J. Dhaliwal, and S. M. Shariff, "Online Social Networks and Writing Styles-A Review of the Multidisciplinary Literature," *IEEE Access*, vol. 8, pp. 67024–67046, Jan. 2020, https://doi.org/10.1109/ACCESS.2020.2985916.

[17] A. Pandey and A. Jain, "Detection of Compromised Accounts using Machine Learning Based Boosting Algorithms- AdaBoost, XGBoost, and CatBoost," in *14th International Conference on Computing Communication and Networking Technologies*, Delhi, India, Jul. 2023, pp. 1–6, https://doi.org/10.1109/ICCCNT56998.2023.10307557.

[18] C. Suman, A. Raj, S. Saha, and P. Bhattacharyya, "Authorship Attribution of Microtext Using Capsule Networks," *IEEE Transactions on Computational Social Systems*, vol. 9, no. 4, pp. 1038–1047, Dec. 2022, https://doi.org/10.1109/TCSS.2021.3067736.

[19] M. Hajja, A. Yahya, and A. Yahya, "Authorship Attribution of Arabic Articles," in *International Conference on Arabic Language Processing*, Nancy, France, Oct. 2019, pp. 194–208, https://doi.org/10.1007/978-3-030-32959-4_14.

[20] S. H. H. Ding, B. C. M. Fung, F. Iqbal, and W. K. Cheung, "Learning Stylometric Representations for Authorship Analysis," *IEEE Transactions on Cybernetics*, vol. 49, no. 1, pp. 107–121, Jan. 2019, https://doi.org/10.1109/TCYB.2017.2766189.

[21] O. Aborisade and M. Anwar, "Classification for Authorship of Tweets by Comparing Logistic Regression and Naive Bayes Classifiers," in *International Conference on Information Reuse and Integration*, Salt Lake City, UT, USA, Jul. 2018, pp. 269–276, https://doi.org/10.1109/IRI.2018.00049.

[22] E. Dauber, R. Overdorf, and R. Greenstadt, "Stylometric Authorship Attribution of Collaborative Documents," in *International Symposium on Cyber Security, Cryptology, and Machine Learning*, Beer-Sheva, Israel, Jun. 2017, pp. 115–135, https://doi.org/10.1007/978-3-319-60080-2_9.

[23] M. Eder, "Short Samples in Authorship Attribution: A New Approach," in *ADHO 2017*, Montreal, Canada, 2017.

[24] S. Minaee, N. Kalchbrenner, E. Cambria, N. Nikzad, M. Chenaghlu, and J. Gao, "Deep Learning--based Text Classification: A Comprehensive Review," *ACM Computing Surveys*, vol. 54, no. 3, Dec. 2021, Art. no. 62, https://doi.org/10.1145/3439726.

[25] M. Sage, P. Cruciata, R. Abdo, J. C. K. Cheung, and Y. F. Zhao, "Investigating the influence of selected linguistic features on authorship attribution using German news articles," in *5th Swiss Text Analytics Conference (SwissText) & 16th Conference on Natural Language Processing (KONVENS)*, Zurich, Switzerland, Jun. 2020, pp. 1–6.

[26] B. Alsulami, E. Dauber, R. Harang, S. Mancoridis, and R. Greenstadt, "Source Code Authorship Attribution Using Long Short-Term Memory Based Networks," in *European Symposium on Research in Computer Security*, Oslo, Norway, Sep. 2017, pp. 65–82, https://doi.org/10.1007/978-3-319-66402-6_6.

[27] N. Saha, P. Das, and H. N. Saha, "Authorship attribution of short texts using multi-layer perceptron," *International Journal of Applied Pattern Recognition*, vol. 5, no. 3, pp. 251–259, Jan. 2018, https://doi.org/10.1504/IJAPR.2018.094819.

[28] Z. Hu, R. K.-W. Lee, L. Wang, E. Lim, and B. Dai, "DeepStyle: User Style Embedding for Authorship Attribution of Short Texts," in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, Tianjin, China, Aug. 2020, pp. 221–229, https://doi.org/10.1007/978-3-030-60290-1_17.

[29] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel, "Authorship attribution of micro-messages: 2013 Conference on Empirical Methods in Natural Language Processing, EMNLP 2013," in *Conference on Empirical Methods in Natural Language Processing*, Seattle, WA, USA, Oct. 2013, pp. 1880–1891.

[30] A. Modupe, T. Celik, V. Marivate, and O. O. Olugbara, "Post-Authorship Attribution Using Regularized Deep Neural Network," *Applied Sciences*, vol. 12, no. 15, Jan. 2022, Art. no. 7518, https://doi.org/10.3390/app12157518.

[31] M. Joshi and N. Zincir-Heywood, "Classification of Micro-Texts Using Sub-Word Embeddings," in *International Conference on Recent Advances in Natural Language Processing*, Varna, Bulgaria, Sep. 2019, pp. 526–533, https://doi.org/10.26615/978-954-452-056-4_062.

[32] X. Tang, S. Liang, and Z. Liu, "Authorship Attribution of The Golden Lotus Based on Text Classification Methods," in *3rd International Conference on Innovation in Artificial Intelligence*, Suzhou, China, Mar. 2019, pp. 69–72, https://doi.org/10.1145/3319921.3319958.

[33] W. Huang, R. Su, and M. Iwaihara, "Contribution of Improved Character Embedding and Latent Posting Styles to Authorship Attribution of Short Texts," in *Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint International Conference on Web and Big Data*, Tianjin, China, Aug. 2020, pp. 261–269, https://doi.org/10.1007/978-3-030-60290-1_20.

[34] S. Aykent and G. Dozier, "Author Identification of Micro-Messages via Multi-Channel Convolutional Neural Networks," in *IEEE International Conference on Systems, Man, and Cybernetics*, Toronto, ON, Canada, Oct. 2020, pp. 675–681, https://doi.org/10.1109/SMC42975.2020.9283214.

[35] P. Shrestha, S. Sierra, F. González, M. Montes, P. Rosso, and T. Solorio, "Convolutional Neural Networks for Authorship Attribution of Short Texts," in *15th Conference of the European Chapter of the Association for Computational Linguistics*, Valencia, Spain, Apr. 2017, pp. 669–674.

[36] F. Ullah, J. Wang, S. Jabbar, F. Al-Turjman, and M. Alazab, "Source Code Authorship Attribution Using Hybrid Approach of Program Dependence Graph and Deep Learning Model," *IEEE Access*, vol. 7, pp. 141987–141999, 2019, https://doi.org/10.1109/ACCESS.2019.2943639.

[37] A. Alqurafi and T. Alsanoosy, "Measuring Customers&rsquo; Satisfaction Using Sentiment Analysis: Model and Tool," *Journal of Computer Science*, vol. 20, no. 4, pp. 419–430, Feb. 2024, https://doi.org/10.3844/jcssp.2024.419.430.

[38] A. Rabab'ah, M. Al-Ayyoub, Y. Jararweh, and M. Aldwairi, "Authorship attribution of Arabic tweets," in *13th International Conference of Computer Systems and Applications*, Agadir, Morocco, Dec. 2016, pp. 1–6, https://doi.org/10.1109/AICCSA.2016.7945818.

[39] A. S. Hossain, N. Akter, and Md. S. Islam, "A Stylometric Approach for Author Attribution System Using Neural Network and Machine Learning Classifiers," in *International Conference on Computing Advancements*, Dhaka, Bangladesh, Jan. 2020, pp. 1–7, https://doi.org/10.1145/3377049.3377079.

[40] H. A. Chowdhury, M. A. H. Imon, S. M. Hasnayeen, and M. S. Islam, "Authorship Attribution in Bengali Literature using Convolutional Neural Networks with fastText's word embedding model," in *1st International Conference on Advances in Science, Engineering and Robotics Technology*, Dhaka, Bangladesh, Dec. 2019, pp. 1–5, https://doi.org/10.1109/ICASERT.2019.8934492.

[41] M. Madhukar and S. Verma, "Hybrid Semantic Analysis of Tweets: A Case Study of Tweets on Girl-Child in India," *Engineering, Technology & Applied Science Research*, vol. 7, no. 5, pp. 2014–2016, Oct. 2017, https://doi.org/10.48084/etasr.1246.

[42] SaraML00, "Authorship-Attribution." [Online]. Available: https://github.com/SaraML00/authorship-attribution.git.