Deep Metric Learning with Different Distance Metrics for Enhanced Classification Model in Typing Style

Hendri Darmawan [1,*], Zulfa Muflihah [2], and Tita Karlita [3]

[1] Department of Electronic Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia

[2] Department of Engineering Physics, Institut Teknologi Sepuluh Nopember, Surabaya, Indonesia

[3] Department of Informatics and Computer Engineering, Politeknik Elektronika Negeri Surabaya, Surabaya, Indonesia

E-mail: hendridarmawan@pens.ac.id[1], 6009242007@student.its.ac.id[2], and tita@pens.ac.id[3]

ABSTRACT

Typing style classification, the task of identifying authors based on their unique textual expression, is an increasingly important research topic with applications in digital forensics, author profiling, and security. As writing offers a powerful and unique medium of self-expression, subtle stylistic features often differentiate individuals. To effectively capture these nuances, which can be challenging for traditional classification models, we propound a deep metric learning (DML) technique. This technique trains a model to learn discriminative embeddings by optimizing the distances between samples; specifically, it aims to ensure that texts from the same user are mapped closer to each other in the vector space than texts from different users, with various distance metrics like Cosine and Euclidean explored to quantify these relationships. The study also compared the effect of text pre-processing and distance metrics on model performance using tweet data from six different Twitter users. The outcomes of the study showed that the model without text pre-processing and with deep metric learning using the Cosine distance metric achieved the optimal result with an accuracy of 0.79, compared to the deep learning model with a categorical cross-entropy loss function which only had an accuracy of 0.76. Additionally, the model with text pre-processing also produced a good performance, with an accuracy of 0.63 using the deep metric learning approach and Cosine distance metric, and an accuracy of 0.64 using deep learning classification with a categorical cross-entropy loss function.

Keywords: Deep metric learning, Long Short-Term Memory (LSTM), natural language processing, triplet network, typing style classification

1. Introduction

Twitter is one of the most widely used social media sites worldwide, with millions of active daily users. This platform allows users to post brief messages known as tweets, limited to 280 characters. However, these tweets contain information that can reflect users' unique and distinctive typing styles. User typing styles are influenced by various factors such as background, profession, interests, and personality [1]. For example, specific individuals consistently employ formal and standardized typing styles, while others use informal and sometimes non-conventional ones. This distinct writing style can serve as a distinguishing characteristic of users, setting them apart from others. The identification of typing styles is an intriguing subject within the realm of Natural Language Processing (NLP), as it can provide insights into the characteristics and preferences of authors. Therefore, recognizing and classifying typing styles can be beneficial in various applications such as sentiment analysis, plagiarism detection, author profiling, and content recommendation. By employing machine learning techniques, we can classify user typing styles by leveraging features such as word choice, punctuation, text length, and other stylistic aspects. Several previous studies have been conducted to identify and classify typing styles using different features and techniques. Research [2] is an example of a study that utilizes data from social media to classify typing styles based on professions. This research employs Support Vector Machines (SVM), Naive Bayes algorithms, and NLP techniques

* Corresponding author.

Received: April 21st, 2025. Revised: June 3rd, 2025. Accepted: June 16th, 2025.

Available online: July 8th, 2025.

DOI: https://doi.org/10.12962/j24068535.v23i1.a1292

such as POS tagging and word stemming. The conducted testing concluded that the Naive Bayes classifier yields the best results with an accuracy of 84%, while SVM is less effective due to variance issues. The researchers suggest using methods for reducing dimensionality like Principal Component Analysis (PCA) and autoencoders to extract the most meaningful essence automatically. They also recommend incorporating unsupervised learning approaches like automatic clustering to enhance efficiency by avoiding manual labeling. Another study [3] examines the effectiveness of stylometric features in identifying alterations in typing styles of seven English novel writers. These features can measure lexical, syntactic, structural, and content-specific aspects of typing styles. Classification is performed using Logistic Regression (LR), and the obtained results demonstrate that stages of literary writing can be accurately identified (with over 70% success) for four out of the seven analyzed authors by utilizing various stylometric features. Moreover, the proposed model only fails for one author, where it achieves a maximum accuracy of 56.8%. The researchers also found that specific authors' typing styles can be identified with extremely high accuracy (88.9%). Additionally, researchers [4] have developed a method for estimating the reputation of social media users based on writing style features. This method employs word choice, grammatical, architectural, and content-related features and the RS-SVM classification technique. The method is applied to web forums in South Korea and successfully classifies users into good and bad reputation categories with high accuracy (94.50%) using a portfolio-based approach.

Within study, we suggest a deep metric learning approach for classifying typing styles of Twitter users on social media. Deep metric learning is a technique that has demonstrated exemplary performance in various computer vision and image problems but has yet to be extensively explored for NLP tasks such as text classification [5]. DML was popularized in computer vision for tasks like face recognition and image retrieval [6]. This approach trains a neural network to output embeddings such that examples from the same class (i.e., same user's typing style) are closer to each other in the embedding space, while examples from different classes are farther apart; a key advantage is that these learned embeddings can then be used to query a database (e.g., of Twitter users) to retrieve specific examples exhibiting similar typing styles to an input, offering more interpretable results beyond simple classification. The 'metric' aspect refers to the distance function (e.g., Cosine, Euclidean) used during training to compare these embeddings [7], with different metrics potentially leading to varied learned representations and model performance.

Therefore, this study explicitly aims to: 1) investigate the effectiveness of a deep metric learning (DML) approach for classifying the nuanced typing styles of Twitter users, and 2) compare its performance against a standard deep learning classification model. We hypothesize that DML is particularly well-suited for this task. Theoretically, typing style classification grapples with identifying authors based on subtle, often non-semantic and high-dimensional cues (e.g., idiosyncratic punctuation, characteristic use of emojis, unconventional capitalization, or preferred sentence structures) which can be diluted or overlooked by standard classification approaches primarily optimized for direct label prediction [8]. Deep metric learning provides a justifiable alternative because its core theoretical underpinning is to learn a specialized embedding space where a chosen distance metric directly reflects the desired notion of similarity—in this context, stylistic similarity. By employing mechanisms like triplet loss, DML explicitly trains the model to structure this embedding space such that representations of texts from the same author are pulled together, while representations from different authors are pushed apart, often by a specified margin. This direct optimization of the embedding space based on relative distances is crucial for capturing the fine-grained distinctions inherent in typing styles, offering a more robust approach to learning these nuances compared to models that only learn a mapping to predefined classes. Furthermore, this learned representation not only aids classification but also offers the potential to retrieve similar typing style examples from a larger corpus (e.g., Twitter data), providing more interpretable insights. To achieve these aims, we will employ a triplet network architecture to train the DML model, generating text embeddings which are then classified using the k-Nearest Neighbors (k-NN) algorithm. The study will systematically evaluate the impact of text pre-processing and different distance metrics (e.g., Cosine, Euclidean) on model accuracy and F1-scores. While specific quantitative results are detailed later, this introductory section establishes our goal to demonstrate how DML can enhance typing style classification by learning more effective feature representations.

2. Materials and Methods

Fig. 1 illustrates the overall system methodology. This pipeline begins with 1) data collection, the scraping of tweets from selected Twitter users. Stage 2), data pre-processing, details the text cleaning and normalization processes. It's important to note, as depicted conceptually in Fig. 1 and elaborated in our experimental design, that this pre-processing stage is comprehensive (including steps like case folding, removal of various elements, stop word filtering, normalization, and stemming) for one set of experiments, while a contrasting set of experiments involves minimal pre-processing (only tokenization) to preserve raw typing style characteristics. Following this is 3) word encoding, where words are converted into a numerical representation suitable for model input. The core of our comparative analysis, also represented in Fig. 1, is stage 4), model training. This stage visually distinguishes between two main modeling approaches: a standard deep learning classification model (using categorical cross-entropy loss) and the DML approach. For the DML pathway, include training a neural network (specifically, a triplet network architecture as shown in Fig. 3) to learn discriminative embeddings. A crucial aspect of this DML implementation, and a focus of our investigation, is the use of different distance metrics (e.g., Cosine, Euclidean) to optimize the embedding space, aiming to pull similar (same user) samples together and push dissimilar (different user) samples apart. The selection and impact of these distance metrics are evaluated in our experiments. Finally, stage 5) covers model testing, where the performance of both the DML (with k-NN classification on learned embeddings) and standard deep learning models is evaluated using metrics such as accuracy, F1 score, and confusion matrices [9]. We conducted two experiments with different conditions in the data pre-processing stage. In the first experiment, we applied comprehensive text pre-processing, which included case folding, removal of punctuation, emojis, URLs, mentions, hashtags, numbers, stop words, word normalization to standardize the vocabulary, and stemming from converting words to their base forms. In the second experiment, we only performed text tokenization, leaving the text as it is without eliminating the characteristics of the users' typing styles. These two experiments aim to assess the effect of text pre-processing on the model's accuracy in classifying user typing styles.

2.1. Data Collection

This study collected tweets from Twitter or X users using the snscrape library, a library for data scraping from social media [10]. We obtained 10,000 tweets from each research subject, ranging from the most recent to the oldest. The research subjects selected for this study were six active book authors on the Twitter social media platform. The writers were chosen as the research subjects because they have different and unique ways of expressing themselves on Twitter. Table 1 displays the example tweets from the Twitter users employed in this study. Based on the scraped data, we obtained a total of 60,000 tweets. These tweets were labeled with specific usernames, resulting in six distinct classes. The next step involved randomly splitting divide the data into training and testing sets, with 80% designated for training, equivalent to 48,000 tweets, and 20% for testing, amounting to 12,000 tweets. We employed a simple and uniform sampling technique to partition the data. Fig. 2 illustrates the word cloud that visualizes the most significant and frequently occurring words in the raw data, represented with a larger font size and distinct color for easy identification [11].

2.2. Data Pre-Processing

In this study, we conducted experiments under two conditions: with pre-processing and without data pre-processing. For the experiment with data pre-processing, we performed several processes as outlined below:

A. Case Folding

In this step, we cleaned the sentences by removing certain characters such as [!"#$%&'()*+,-./:;<=>?@^_`{|}~] and filtered out emojis, new lines, white spaces, links, username tags (@username), hashtags (#hashtag), URL links, and numbers. Furthermore, we converted all characters to lowercase, reducing the number of unique characters to a single type, irrespective of capitalization [12].

B. Stopword Filtering

The subsequent step involved removing stopwords, which are common and frequently occurring words, to extract only the relevant words after tokenization (the process of breaking text into tokens) [13]. In this step, we

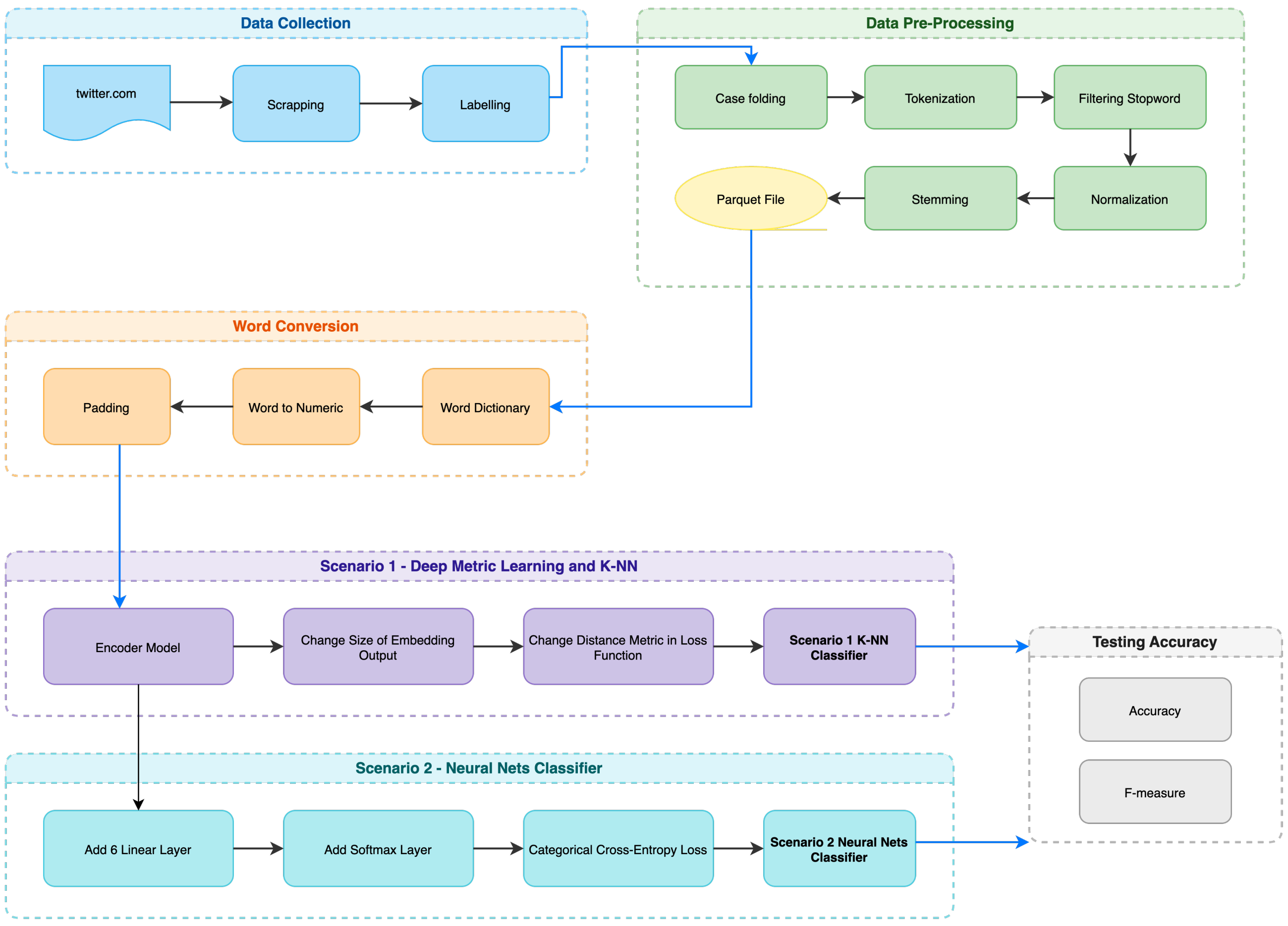


Fig. 1: System methodology.

Table 1: Sample Tweets from the Dataset.

Class	Username	Sample data
0	afrkml	[‘Kalau’, ‘gitu’, ‘dari’, ‘nutrisinya’, ‘?‘, ‘Makan’, ‘buah’, ‘dan’, ‘sayurnya’, ‘beda’, ‘?‘]
1	cursedkidd	[‘gk’, ‘brani’, ‘i’, ‘gw’]
2	ernestprakasa	[‘Ogitu’, ‘u’, ‘u’, ‘u’, ‘.‘, ‘.’, ‘.‘]
3	newsplatter	[‘Nanti’, ‘diajak’, ‘adu’, ‘beyblade’, ‘loh’, ‘.‘]
4	pandji	[‘Bener’, ‘banget’, ‘kak’]
5	pidibaiq	[‘Iya’, ‘,‘, ‘dong’]

utilized the StopWordRemoverFactory function from the Sastrawi library, along with additional custom stopwords such as "jd," "tdk," "utk," and others.

C. Normalization

Normalization standardizes words with similar meanings, usually addressing abbreviations or slang. This step transforms slang or abbreviated words into their standard forms [14]. The process relies on a mapping of non-standard words to their standard counterparts.

D. Stemming

Stemming is converting words to their root forms [13]. In this study, we used the StemmerFactory function from the Sastrawi library to perform stemming [15].

Fig. 2: Word cloud representation.

2.3. Word Encoding

The encoding process aims to transform the text into numerical representations that can be processed by machine learning models [16]. The first step involves creating a unique dictionary per word in the form of numbers, where each word is assigned a unique and sequential numeric value based on its frequency in the corpus, with more frequently occurring words assigned smaller values. Next, the words in the text are replaced with their respective numeric values, transforming the text into a sequence of integers [17]. The final step involves adding padding or zero values to ensure that each sequence of integers has the same length, determined by the maximum length in the corpus [18]. This padding ensures uniformity and facilitates text data processing within machine learning algorithms.

2.4. Deep Metric Learning

We employed deep metric learning as a feature learning method to compare vector representations (embeddings) from different data samples. We utilized two distance metrics, namely Euclidean and Cosine, to measure the similarity between embeddings. Our model employed a triplet network architecture that takes three input data samples: anchor input, positive input, and negative input, as seen in Fig. 3. The anchor and positive inputs were of the same class, whereas the negative input belonged to a different class [19]. Deep metric learning using triplet scheme seeks to reduce the distance between the embeddings of the anchor and positive samples while increasing the distance between the embeddings of the anchor and negative samples [20]. We also utilized a margin loss with a value of 1.0 to ensure adequate differentiation between the embeddings of the anchor and negative inputs, thereby enhancing the quality of feature learning. The margin determines the minimum distance that must be maintained between the embeddings of the anchor and negative inputs [21]. The model incurs a penalty if the distance falls below the specified margin. This mechanism assists the model in generating improved embeddings that are better able to discriminate between different classes.

2.5. Model Architecture

The deep learning model architecture utilized in this study is shown in Fig. 4. The model comprises of five layers: 1) the input layer, which receives encoded textual data as input; 2) the embedding layer, which transforms the numerical textual data into a 512-dimensional numerical vector representation; 3) the feature extraction layer, which employs bidirectional Long Short-Term Memory (LSTM) to extract essential elements from the textual data; 4) the fully connected layer, which links the extracted features to the target classes; and 5) the output layer, which generates the latent embedding for deep metric learning output (scenario 1) and prediction probability for classification output (scenario 2).

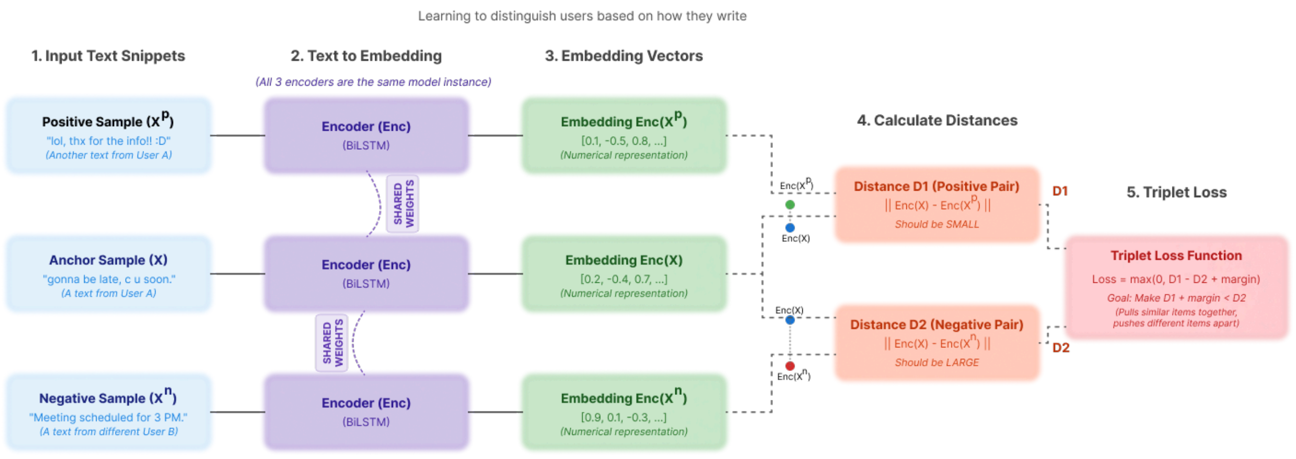

Learning to distinguish users based on how they write
1. Input Text Snippets
2. Text to Embedding
3. Embedding Vectors
(All 3 encoders are the same model instance)
Positive Sample (X^p)
"lol, thx for the info!! :D"
(Another text from User A)
Encoder (Enc)
(BiLSTM)
Embedding Enc(X^p)
[0.1, -0.5, 0.8, ...]
(Numerical representation)
SHARED WEIGHTS
Anchor Sample (X)
"gonna be late, c u soon."
(A text from User A)
Encoder (Enc)
(BiLSTM)
Embedding Enc(X)
[0.2, -0.4, 0.7, ...]
(Numerical representation)
SHARED WEIGHTS
Negative Sample (X^n)
"Meeting scheduled for 3 PM."
(A text from different User B)
Encoder (Enc)
(BiLSTM)
Embedding Enc(X^n)
[0.9, 0.1, -0.3, ...]
(Numerical representation)
4. Calculate Distances
Enc(X^p)
Enc(X)
Distance D1 (Positive Pair)
|| Enc(X) - Enc(X^p) ||
Should be SMALL
D1
Enc(X)
Enc(X^n)
Distance D2 (Negative Pair)
|| Enc(X) - Enc(X^n) ||
Should be LARGE
D2
5. Triplet Loss
Triplet Loss Function
Loss = max(0, D1 - D2 + margin)
Goal: Make D1 + margin < D2
(Pulls similar items together, pushes different items apart)

Fig. 3: Deep metric learning with triplet network.

A. Input Layer

The initial layer is the input layer, which takes the encoded textual data with a specified maximum length. In the pre-processing experiment, the maximum length is set to 59. In contrast, in the non-pre-processing experiment, it is set to 280, corresponding to Twitter's maximum character limit. This maximum length is determined based on the most extended tweet in the dataset. The input layer's function is to input the textual data into the deep learning model as integer sequences.

B. Initial Word Embedding Layer

This process transforms the sequential tokens from the input layer into numerical vectors of a specific dimension. It is accomplished by adding an embedding layer at the beginning of the deep learning model, which trains the numerical vectors representing the words in the dataset [22]. Some hyperparameters need to be set, namely the embedding dictionary size and the output size of the embedding layer. The initial word embedding layer has a fixed size 256, while the embedding dictionary size or maximum features is determined based on the maximum number of unique tokens from the tokenized text. In the pre-processed text experiment, the maximum features are set to 50,000, even though the maximum number of unique tokens is 36,765. In contrast, for the non-preprocessed text, the maximum features are set to 70,000, with a maximum number of unique tokens of 62,117. The maximum features can be adjusted according to different text processing conditions and usage, which can enhance the performance of the input of the embedding layer. In this study, we did not utilize pre-trained word embeddings, so the weights in the embedding layer are trained solely on the dataset used in the research.

C. Feature Extraction Layer

Bidirectional LSTM is a sequence processing model consisting of two LSTMs: one that processes the input sequentially from start to end and the other from end to start [22]. Bidirectional LSTM effectively enhances the network's access to information by capturing context from both the forward and backward directions of the sequence. It can capture temporal patterns in textual data better than a unidirectional LSTM. The LSTM cell structure comprises a cell state (C_t), an input gate (in_t), a forget gate (f_t), and an output gate (out_t) [23]. These structures determine which information should be stored, discarded, or transferred to the subsequent cell. The equations for these gates are stated in (1)–(6), where X_t is the input vector at time step t; h_{t-1} is the hidden state from the previous time step; W and b are the weight matrices and bias vectors for each respective gate (input, forget, output, cell); σ is the sigmoid activation function; and $tanh$ is the hyperbolic tangent activation function.

$$f_t = \sigma\left(W_f \cdot [h_{t-1}, x_t] + b_f\right) \tag{1}$$

$$i_t = \sigma(W_i \cdot [h_{t-1}, x_t] + b_i) \tag{2}$$

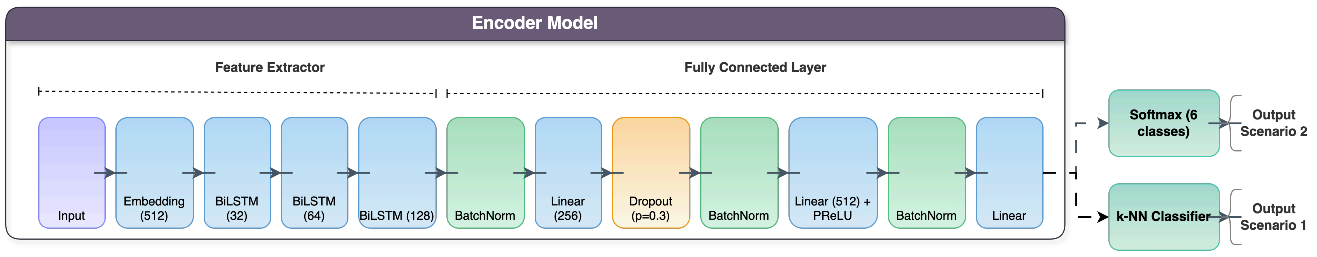


Fig. 4: Deep learning model architecture.

$$\tilde{C}_t = tanh(W_C \cdot [h_{t-1}, x_t] + b_C) \tag{3}$$

$$C_t = f_t \times C_{t-1} + i_t \times \tilde{C}_t \tag{4}$$

$$o_t = \sigma(W_o \cdot [h_{t-1}, x_t] + b_o) \tag{5}$$

$$h_t = o_t \times tanh(C_t) \tag{6}$$

D. Fully Connected Layer

Following the feature extraction layer, the subsequent layer is the fully connected layer, where each neuron receives input from every neuron in the preceding feature extraction layer and delivers output to every neuron in the following layer. Dropout, with a probability of 0.3, is employed to mitigate overfitting by randomly deactivating a selection of neurons within a particular layer during each iteration [24]. Additionally, batch normalization is employed to expedite the convergence of the model during training by normalizing the input distribution to each model layer. We also incorporate the PReLU activation function to boost the model's capability of capturing nonlinear characteristics from the textual data. The PReLU activation function is a modification of the ReLU activation function, incorporating the adjustable parameter alpha [25]. The PReLU activation function addresses the dying ReLU, which occurs when neurons remain inactive for all inputs.

E. Output Layer

In this study, we conducted testing using two modeling scenarios. The first scenario involved a classification model with optimization of triplet loss, while the second used a classification model with a categorical cross-entropy loss. In the first scenario, the output consists of embedding vectors derived from the final layer of the fully connected layer prior to being processed by the softmax layer for determining class probabilities. These embedding vectors are then classified using k-NN with the voting method of the nearest k neighbors. We chose k-NN because the embedding vectors, generated through deep metric learning approach training, exhibit a more distinct distribution across different classes as the input textual data is multiplied by model weights trained using distance-based loss functions. Meanwhile, in the second scenario, the output comprises class probabilities obtained from the softmax activation function.

2.6. Loss Function

In this study, we employed an artificial neural network approach, necessitating a loss function to measure the model's performance in prediction. We utilized two different types of loss functions for two distinct scenarios. For the first scenario, we employed triplet loss, which assesses the disparity between the output embeddings generated by the encoder for the anchor input (X), positive input (X^p), and negative input (X^n). The goal of this loss function is to reduce the gap between X and X^p, while simultaneously increasing the separation between X and X^n by applying a margin. The superscript 'p' and 'n' in X^p and X^n denotes "positive" and "negative" and is not an index. In the deep metric learning scenario, the loss function for the triplet network to be minimized is defined as (7).

$$L_{Triplet} = max(0, \| Enc(X) - Enc(X^p) \|_2 - \| Enc(X) - Enc(X^n) \|_2 + \alpha) \tag{7}$$

The loss formula shows that the distance between point X and point X^n is expected to be greater than between point X and point X^p, plus the margin for the loss to be minimized. This loss function maximizes the difference between the positive and negative distances by adding a margin. If this difference is more significant than the margin, the loss function evaluates to zero, indicating that the data representations satisfy the criterion. The model adjusts the neural network weights during training to push the anchor and negative samples apart. We employ two different distance metrics, namely Euclidean distance and Cosine distance, for this loss function. The Euclidean distance measures the straight-line distance between two points in a Euclidean space and is defined in (8) [26].

$$\text{Euclidean distance} = \sqrt{\sum_{i=1}^{d}(A_i - B_i)^2} \tag{8}$$

$\mathbf{A} = [A_1, A_2, ..., A_d] \in \mathbb{R}^d$ and $\mathbf{B} = [B_1, B_2, ..., B_d] \in \mathbb{R}^d$ are two real-valued vectors in a d-dimensional space, A_i and B_i denote the i-th components of vectors $\mathbf{A}$ and $\mathbf{B}$, respectively. The second distance metric we used was cosine distance, which quantifies the angular dissimilarity between two vectors. It is derived from cosine similarity defined as (9) [27].

$$\text{Cosine distance} = 1 - \frac{\sum_{i=1}^{d} A_i B_i}{\sqrt{\sum_{i=1}^{d} A_i^2}\sqrt{\sum_{i=1}^{d} B_i^2}} \tag{9}$$

The numerator $\sum_{i=1}^{d} A_i B_i$ is the dot product of vectors $\mathbf{A}$ and $\mathbf{B}$, the denominator represents the product of the Euclidean (L2) norms of $\mathbf{A}$ and $\mathbf{B}$, the resulting cosine distance ranges from 0 (identical orientation) to 2 (opposite orientation). These distance metrics are used to quantify the similarity or dissimilarity between vector representations, which plays a crucial role in optimizing the triplet loss function in our model.

For the second scenario, we utilize categorical cross-entropy, a typical loss function for multiclass classification tasks. This loss function computes the difference between the probability distribution generated by the model and the true probability distribution. The goal is to minimize the score, and the excellent value for cross-entropy is 0. This loss function mandates that the output layer comprises n nodes (one corresponding to each class), which, in this instance, amounts to six nodes, and it necessitates the employment of the 'softmax' activation function to estimate the likelihoods for each class. The categorical cross-entropy loss function can be articulated as (10).

$$CCEloss = -\frac{1}{N}\sum_{i=1}^{N}\sum_{c=1}^{C}\mathbf{1}_{y_i \in C_c}\, log\, p_{model}[y_i \in C_c] \tag{10}$$

N is the total number of samples in the batch, C is the total number of classes (six in this study), $\mathbf{1}_{y_i \in C_c}$ is an indicator function that is equal to 1 if the true label y_i for sample i belongs to class c and 0 otherwise, and $p_{\text{model}}[y_i \in C_c]$ is the model's predicted probability that sample i belongs to class c.

2.7. Model Optimization

We used a Tesla P100-PCIE-16GB GPU to train this model with a batch size 512 for 200 epochs. We employed the AdamW algorithm as the optimization method for backpropagation. This algorithm is a variation of the Adam algorithm that integrates weight decay as a means of regularization [28]. The AdamW algorithm adjusts the learning rate according to the first and second moments of the gradients and separately reduces the weights independent of the learning rate [29]. We used an initial learning rate of 0.0015.

2.8. Metrics Performance

The proposed research focuses on classification, and performance metrics for classification are needed to evaluate the model. We use accuracy as a metric to assess the overall model performance and F1-score to evaluate the model's performance in each class [30] can be expressed as (11).

$$Accuracy = \frac{TN + TP}{TN + FP + TP + FN} \tag{11}$$

F1-score provides more informative results as it combines precision and recall values for every class can be stated as (12).

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{12}$$

Precision evaluates how accurately the model identifies true positive cases, whereas recall assesses the model's ability to detect all the positive instances that it should have captured can be formulated as (13)-(14).

$$Precision = \frac{TP}{TP + FP} \tag{13}$$

$$Recall = \frac{TP}{TP + FN} \tag{14}$$

These metrics are used together to calculate the F1-score, which provides information about the balance between precision and recall. Additionally, we employ the confusion matrix to display the classification evaluation of the model visually. The confusion matrix is a table that shows the number of data instances classified correctly and incorrectly by the model for each class [30]. The confusion matrix helps analyze the strengths and weaknesses of the model in recognizing user typing styles.

3. Experimental Setup

To systematically evaluate the effectiveness of deep metric learning for typing style classification, we designed a series of experiments comparing two primary modeling scenarios under different data conditions. The core variables investigated were the data pre-processing strategy, the choice of distance metric, the dimensionality of the embedding space, and the number of nearest neighbors (k) for classification in deep metric learning scenario. The experiments were structured around two main scenarios:

3.1. Scenario 1: Deep Metric Learning (DML) with k-NN Classification

This scenario tests the proposed DML approach. The model is trained using a triplet loss function to produce discriminative embeddings, which are then classified using a k-NN algorithm. The following parameters were systematically varied:

- Data Condition: experiments were run on two separate datasets: (a) with comprehensive pre-processing and (b) without pre-processing (raw text).
- Output Embedding Dimension: the dimensionality of the final embedding vector was tested at three levels: 64, 128, and 256.
- Distance Metric: for the triplet loss calculation and k-NN classification, two metrics were compared: Cosine distance and Euclidean distance.
- Number of Neighbors (k): for each combination of the above parameters, the value of k for the k-NN classifier was varied to identify the optimal performance.

3.2. Scenario 2: Standard Deep Learning Classification

This scenario serves as the baseline for comparison. The model uses the same underlying architecture as in Scenario 1, but the final layers are adapted for standard multi-class classification.

- Loss Function: the model was trained using a standard categorical cross-entropy loss function with a softmax activation layer for output probabilities.
- Data Condition: as in Scenario 1, experiments were run with and without comprehensive text pre-processing.

- Final Dense Layer Neurons: to ensure a fair comparison with the DML approach, the number of neurons in the final fully connected layer (before the softmax output) was set to 64, 128, and 256, corresponding to the embedding dimensions tested in Scenario 1.

All models were trained and evaluated using the hardware and optimization parameters detailed in the "Model Optimization" section. The performance of each configuration was measured using accuracy and F1-score, as described in "Metrics Performance."

4. Result and Discussion

In this study, we performed multiple experiments to attain optimal results. The first experiment seeks to ascertain the ideal dimension of the output embedding vector for the triplet loss framework. The chosen dimensions of the output embedding vector from this experiment were also utilized as the count of neurons in the fully connected layer for the second scenario, which entailed classification using categorical cross-entropy loss.

The second experiment focused on identifying the most suitable distance metric for the triplet loss scenario. Both experiments used two approaches to handle the textual data: pre-processing and without pre-processing. These experiments aimed to explore the effect of data pre-processing on the performance of the models and to identify the best configurations for each situation. The experimental results were analyzed by examining three factors that impact the model’s accuracy: the dimensionality of the output embedding vector, the choice of distance metric, and the value of k (the number of nearest neighbors taken into account). In the experiments conducted with pre-processed textual data in Table 2, it can be observed from the results that the Euclidean distance metric in average accuracy outperformed the Cosine distance metric, despite Cosine distance achieved the highest accuracy.

Fig. 5a illustrates that the accuracy tends to increase with an increasing value of k, but the rate of improvement decreases after a particular value of k. For a 64-dimensional embedding size, the accuracy remains relatively constant across various k and distance metrics values, ranging from 0.612 to 0.626. For larger embedding sizes (128 and 256), the model’s performance is more influenced by the choice of distance metric and the value of k. The optimal value of k varies depending on the distance metric and embedding size. The best-performing model is achieved with a 64-dimensional embedding size and the Cosine distance metric, attaining an accuracy 0.626 at k = 61. Conversely, Fig. 5b depicts the results for the model trained on data without pre-processing, revealing a markedly different and superior performance. While a similar trend of accuracy increasing with k is observed, the overall accuracies are significantly higher across all configurations. This reinforces the finding that omitting pre-processing is beneficial for this task. In this scenario, models using the Cosine distance metric consistently outperform those using the Euclidean distance metric. The top-performing configuration is the model with a Cosine distance metric and a 256-dimensional embedding size, which reaches a peak accuracy of 0.786 at k = 41.

Table 3 presents the model evaluation results using triplet loss optimization without pre-processing data. The models were tested with different distance metrics (Cosine and Euclidean) and varying values of k for different output embedding sizes (64, 128, and 256). The findings suggest that the model’s precision fluctuates based on these parameters. The model with a Cosine distance metric and a 256-dimensional embedding size achieves the highest performance, with an accuracy of 0.786 at k = 41. However, the difference in accuracy between different values of k is not substantial, with accuracies around 0.78 for most k values. The model with a Euclidean distance metric performs slightly lower than the Cosine distance metric, with accuracies around 0.77 for most k values and embedding sizes. However, the accuracy difference between different k values is also not significant. Overall, it can be inferred that the model with a Cosine distance metric and a 256-dimensional embedding size exhibits the best performance. However, Tables 2 and 3 show that the model with a Cosine distance metric requires a higher value of k than that with a Euclidean distance metric to achieve high accuracy. We show the confusion matrix for the best model of both scenarios in Fig. 6. Overall, based on the conducted experiments, it can be identified that the model without employing pre-processing techniques on textual data achieves the highest accuracy. This model utilizes the Cosine distance metric and a 256-dimensional output embedding size.

Table 2: Accuracy with the Best k with Pre-Processing Text.

No	Embedding size	Euclidean distance		Cosine distance	
		k	Accuracy	k	Accuracy
1	**64**	21	0.619	**61**	**0.626**
2	128	11	0.517	71	0.524
3	256	41	0.618	81	0.530

Table 3: Evaluation Result for LSTM Modeling Without Pre-processing Text.

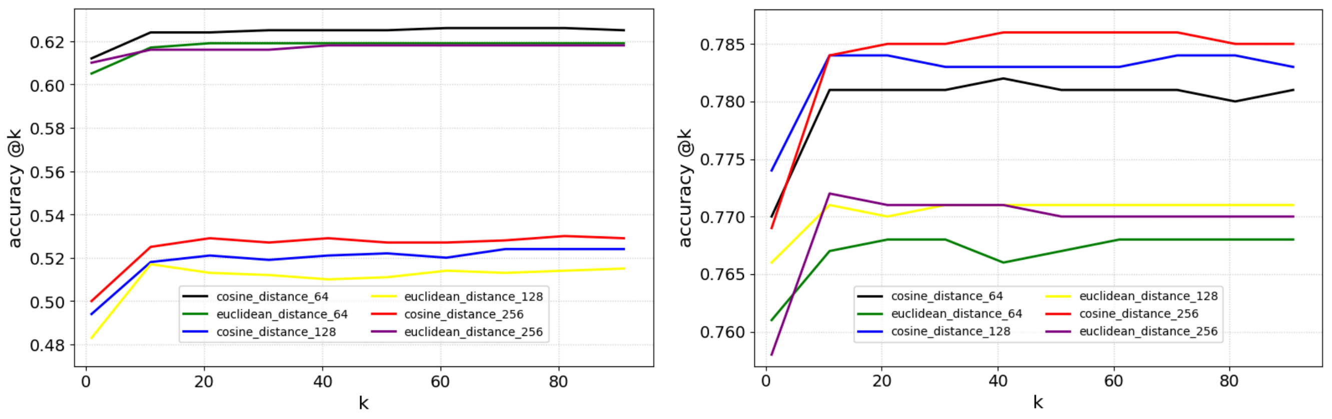
No	Embedding size	Euclidean distance		Cosine distance	
		k	Accuracy	k	Accuracy
1	64	31	0.768	41	0.782
2	128	11	0.771	21	0.784
3	**256**	11	0.772	**41**	**0.786**

Table 4: Comparison of Overall Scenario Accuracy.

No.	Text pre-processing	Deep metric learning	F1-score						Accuracy
			0	1	2	3	4	5	
1	Yes	No	0.71	0.7	0.55	0.53	0.6	0.74	0.64
2	No	No	0.79	0.83	0.71	0.72	0.7	0.81	0.76
3	Yes	Yes	0.72	0.71	0.53	0.49	0.56	0.75	0.63
4	**No**	**Yes**	**0.83**	**0.85**	**0.74**	**0.74**	**0.72**	**0.83**	**0.79**

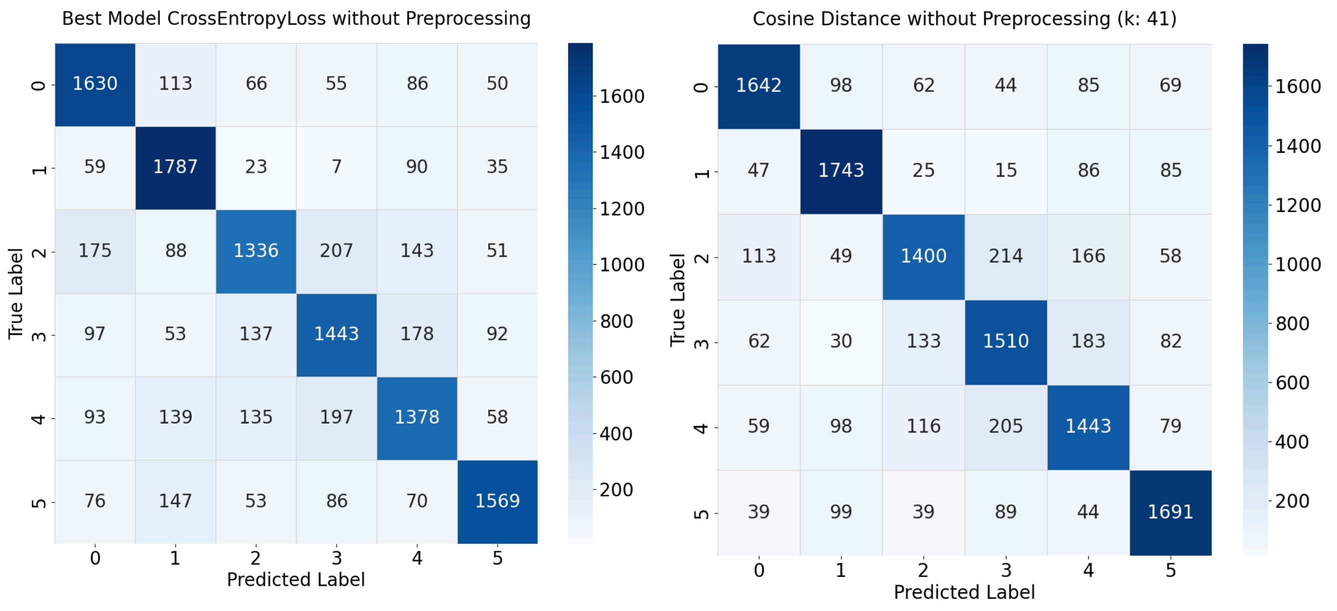
Table 4 compares the K-NN classification model with triplet loss and the standard deep learning classification model with the same hyperparameters and architecture, except for the softmax layer at the end of the encoder model. Table 4 shows that the model without employing text pre-processing techniques achieves higher accuracy than those using pre-processing techniques in both scenarios. It could be attributed to the fact that pre-processing techniques remove the distinctive typing styles of individual users, making it challenging for the model to recognize user-specific writing characteristics such as the use of periods at the end of sentences, spaces before emojis, repeated letters, emojis, and etc. The data indicates that the model optimized with deep metric learning and k-NN classification achieves higher accuracy than that optimized with categorical cross entropy in standard deep learning classification. We argue that this is because deep metric learning assists the model in learning more robust and discriminative feature representations for classification tasks. Deep metric learning forces the model to separate similar features from dissimilar ones. In this case, the model must distinguish between text with similar meanings but different typing styles and messages with different meanings but similar typing styles.

To evaluate the quality of the learned representations, we employed the t-Distributed Stochastic Neighbor Embedding (t-SNE) algorithm to reduce the high-dimensional output embeddings into a two-dimensional space for visual inspection. The visualizations reveal a stark contrast between the two training methodologies. As seen in Fig. 7, the DML approach, trained using Cosine distance, successfully learns an embedding space where the six classes form distinct, well-separated clusters. This indicates the model effectively groups texts by the same author while enforcing distance from other authors. In stark contrast, Fig. 8 shows that the standard classification model, trained with categorical cross-entropy, fails to produce meaningful separation. The resulting embeddings are highly "disorganized," with data points from all classes heavily intermingled and lacking discernible boundaries. This is an expected outcome, as a standard classifier is optimized for a final prediction task, not for explicitly organizing the feature space to reflect semantic similarity, which is the primary goal of DML. Drilling down into the successful DML results, Fig. 7 also highlights the impact of pre-processing on the embedding structure. The model trained on data without pre-processing (Fig. 7a) yields clusters that, while distinct, exhibit higher intra-class variance and appear more dispersed. Conversely, the model trained with pre-processed data (Fig. 7b) produces



(a) (b)

Fig. 5: Model Accuracy at k: (a) with pre-processing; (b) without pre-processing (the accuracy is higher).

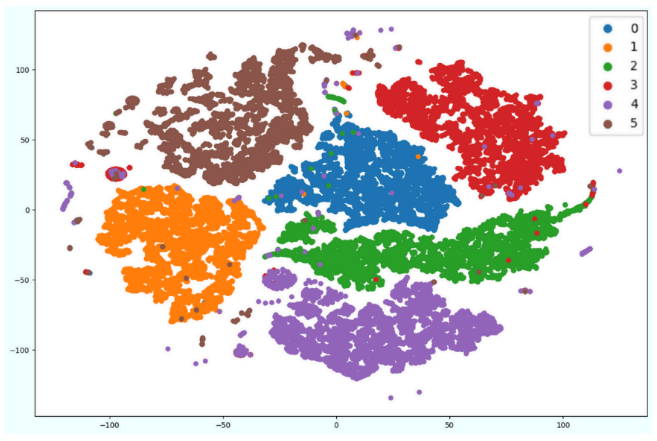


(a) (b)

Fig. 6: Confusion matrix: (a) CCE loss without pre-processing; (b) Cosine distance without pre-processing (k = 41) (the highest accuracy).

significantly more compact and globular clusters, indicating lower variance. This suggests that reducing feature variation through pre-processing enables the model to learn a more centralized and dense representation for each author's style. Notably, this distinction is absent in Fig. 8, where both pre-processed and non-pre-processed data result in similarly disorganized plots, further emphasizing that the training objective (not just the input data) is the dominant factor in creating separable embeddings. The findings of this study have significant implications.

The collective findings of this study carry significant implications. The demonstrated success of DML, especially with minimal pre-processing and Cosine distance, suggests its strong potential for various real-world applications sensitive to textual style. These include, but are not limited to, author profiling, forensic linguistics (e.g., verifying authorship of disputed texts), plagiarism detection that goes beyond simple content matching, and even the identification of coordinated inauthentic behavior online (e.g., bot networks or sock-puppet accounts that might betray themselves through stylistic inconsistencies or unnatural homogeneity). Moreover, the ability to retrieve stylistically similar texts could enhance personalized content recommendation systems or aid in a deeper understanding of online subcultures defined by unique linguistic styles. However, this study is not without limitations. The dataset, while providing distinct styles, is limited to six authors. Future research should validate these


(a)

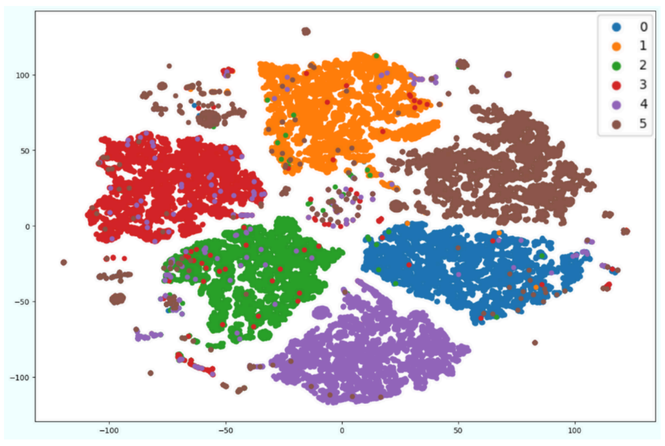
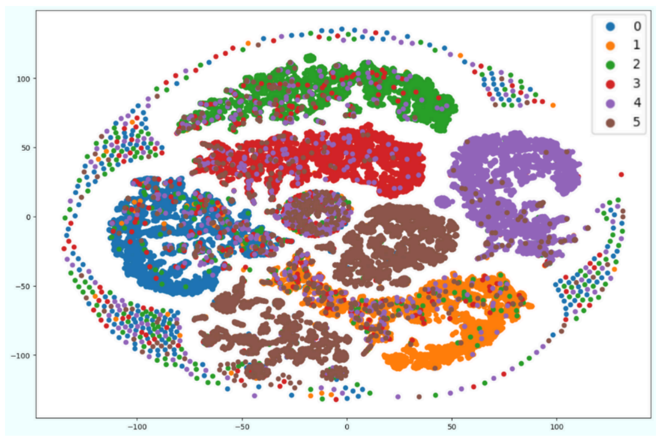

(b)

Fig. 7: Embedding space Cosine distance: (a) without pre-processing 256 (higher variance); (b) with pre-processing 64 (lower variance).


(a)

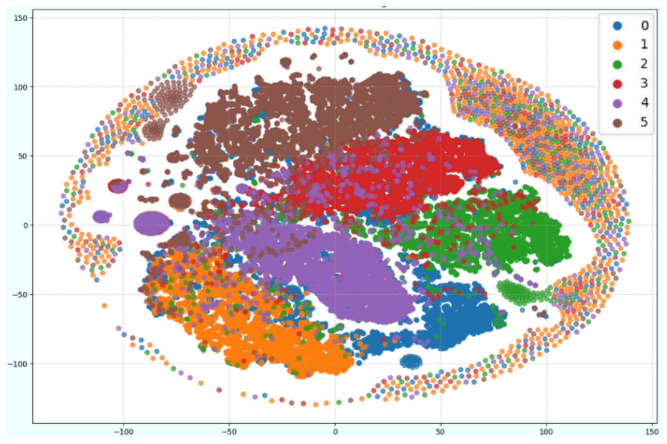

(b)

Fig. 8: Categorical cross entropy before softmax layer (the cluster is disorganized): (a) without pre-processing; (b) with pre-processing.

findings on larger, more diverse datasets encompassing a wider array of typing styles and demographic groups. Exploring alternative DML architectures, more sophisticated triplet (or quadruplet) selection strategies, and the explicit modeling of specific stylistic features (e.g., n-grams of characters, punctuation patterns) within the DML framework could yield further improvements. Additionally, investigating the temporal evolution of typing styles and the robustness of these models to adversarial stylistic manipulation are promising avenues for future work.

5. Conclusion

This research aimed to develop and evaluate a machine learning model for categorizing Twitter users' typing styles. Utilizing tweet data from six distinct users, we compared a deep metric learning (DML) approach for learning text vector representations against a standard deep learning classification approach with a categorical cross-entropy loss function. Experiments were conducted under two text pre-processing conditions: comprehensive pre-processing and no pre-processing. The findings quantitatively underscore the benefits of DML and minimal pre-processing. Specifically, the model without text pre-processing combined with deep metric learning (using the Cosine distance metric) achieved the highest accuracy of 0.79. This surpassed the standard deep learning model under similar no pre-processing conditions, which yielded an accuracy of 0.76. When text pre-processing was applied, the DML model (Cosine distance) achieved an accuracy of 0.63, while the standard deep learning model reached 0.64. These results highlight that retaining the distinctive characteristics of user typing styles by avoiding extensive pre-processing is crucial. Moreover, the superior accuracy of the DML approach, particularly the 0.79 achieved with Cosine distance and no pre-processing, demonstrates its enhanced capability to learn more discriminative feature representations essential for nuanced classification tasks like typing style identification.

CRediT Authorship Contribution Statement

Hendri Darmawan: Writing – review & editing, Writing – original draft, Validation, Software, Methodology, Conceptualization. **Zulfa Muflihah:** Writing – original draft, Formal analysis, Data curation, Conceptualization. **Tita Karlita:** Writing – original draft, Software, Investigation.

Declaration of Competing Interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data Availability

The data used to support the findings of this study are available from the corresponding author upon request, subject to Twitter’s terms of service and user privacy considerations.

Declaration of Generative AI and AI-assisted Technologies in The Writing Process

The authors used generative AI to improve the writing clarity of this paper. They reviewed and edited the AI-assisted content and take full responsibility for the final publication.

References

[1] S. Luo, M. Preiss, and E. Sanders, “Does personality functioning influence perception of social distance amongst Czech students studying in an international university, a state university and the general population?,” *Acta Psychologica*, vol. 258, p. 105230, 2025, doi: 10.1016/j.actpsy.2025.105230.
[2] X. Tang, “Author identification of literary works based on text analysis and deep learning,” *Heliyon*, vol. 10, no. 3, p. e25464, 2024, doi: 10.1016/j.heliyon.2024.e25464.
[3] V. Cammarota, S. Bozza, C.-A. Roten, and F. Taroni, “Stylometry and forensic science: A literature review,” *Forensic Science International: Synergy*, vol. 9, p. 100481, 2024, doi: 10.1016/j.fsisyn.2024.100481.
[4] J. H. Suh, “Comparing writing style feature-based classification methods for estimating user reputations in social media,” *SpringerPlus*, vol. 5, no. 1. SpringerOpen, Dec. 2016, doi: 10.1186/s40064-016-1841-1.
[5] O. Jia, H. Huang, J. Ren, L. Xie, and Y. Xiao, “Contrastive learning with text augmentation for text classification,” *Applied Intelligence*, vol. 53, no. 16. pp. 19522–19531, Aug. 2023, doi: 10.1007/s10489-023-04453-3.
[6] H. Darmawan, M. Yuliana, and Moch. Z. S. Hadi, “Cloud-based Paddy Plant Pest and Disease Identification using Enhanced Deep Metric Learning and k-NN Classification with Augmented Latent Fusion,” *International Journal of Intelligent Engineering and Systems*, vol. 16, no. 6. pp. 158–170, 2023, doi: 10.62527/joiv.8.3-2.3104.
[7] X. Zhe, S. Chen, and H. Yan, “Directional statistics-based deep metric learning for image classification and retrieval,” *Pattern Recognition*, vol. 93, pp. 113–123, 2019, doi: 10.1016/j.patcog.2019.04.005.
[8] C. Wang, W. Zheng, Z. Zhu, J. Zhou, and J. Lu, “Introspective Deep Metric Learning,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 4, pp. 1964–1980, 2024, doi: 10.1109/TPAMI.2023.3312311.
[9] F. Chen, Z. Zhu, C. Sun, and L. Xia, “Evaluating metric and contrastive learning in pretrained models for environmental sound classification,” *Applied Acoustics*, vol. 232, p. 110593, 2025, doi: 10.1016/j.apacoust.2025.110593.
[10] W. A. Akbari, T. Tukino, B. Huda, and M. Muslih, “Sentiment Analysis of Twitter User Opinions Related to Metaverse Technology Using Lexicon Based Method,” *Sinkron : jurnal dan penelitian teknik informatika*, vol. 7, no. 1, pp. 195–201, Jan. 2023, doi: 10.33395/sinkron.v8i1.11992.
[11] B. Wang, Q. Shi, X. Wang, Y. Zhou, W. Zeng, and Z. Wang, “EmotionLens: Interactive visual exploration of the circumplex emotion space in literary works via affective word clouds,” *Visual Informatics*, vol. 9, no. 1, pp. 84–98, 2025, doi: 10.1016/j.visinf.2025.02.003.
[12] M. A. Fauzi, “Word2Vec model for sentiment analysis of product reviews in Indonesian language,” *International Journal of Electrical and Computer Engineering (IJECE)*, vol. 9, no. 1, pp. 525–530, 2019, doi: 10.11591/ijece.v9i1.pp525-530.
[13] H. Dwiharyono and S. Suyanto, “Stemming for Better Indonesian Text-to-Phoneme,” *Ampersand*, vol. 9, p. 100083, 2022, doi: 10.1016/j.amper.2022.100083.
[14] A. A. Aliero, B. S. Adebayo, H. O. Aliyu, A. G. Tafida, B. U. Kangiwa, and N. M. Dankolo, “Systematic Review on Text Normalization Techniques and its Approach to Non-Standard Words,” *International Journal of Computer Applications*, vol. 185, no. 33. pp. 44–55, Sep. 2023, doi: 10.5120/ijca2023923106.
[15] M. A. Rosid, A. S. Fitrani, I. R. I. Astutik, N. I. Mulloh, and H. A. Gozali, “Improving Text Preprocessing For Student Complaint Document Classification Using Sastrawi,” *IOP Conference Series: Materials Science and Engineering*, vol. 874, no. 1, p. 12017, Jun. 2020, doi: 10.1088/1757-899X/874/1/012017.
[16] X. Yang, K. Yang, T. Cui, M. Chen, and L. He, “A Study of Text Vectorization Method Combining Topic Model and Transfer Learning,” *Processes*, vol. 10, no. 2, 2022, doi: 10.3390/pr10020350.
[17] R. Sakthi Murugan and V. Ananthanarayana, “WordCode using WordTrie,” *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 3, pp. 923–933, 2022, doi: 10.1016/j.jksuci.2019.05.011.
[18] W. Li, L. Zhu, Y. Shi, K. Guo, and E. Cambria, “User reviews: Sentiment analysis using lexicon integrated two-channel CNN–LSTM family models,” *Applied Soft Computing*, vol. 94, p. 106435, 2020, doi: 10.1016/j.asoc.2020.106435.

[19] E. Hoffer and N. Ailon, “Deep Metric Learning Using Triplet Network,” in *Similarity-Based Pattern Recognition*, 2015, pp. 84–92.
[20] M. Kaya and H. S. Bilge, “Deep Metric Learning: A Survey,” *Symmetry*, no. 11. 2019, doi: 10.3390/sym11091066.
[21] G. Wang, Y. Guo, Z. Xu, Y. Wong, and M. S. Kankanhalli, “Semantic-Aware Triplet Loss for Image Classification,” *IEEE Transactions on Multimedia*, vol. 25, no. , pp. 4563–4572, 2023, doi: 10.1109/TMM.2022.3177929.
[22] S. F. Sabbeh and H. A. Fasihuddin, “A Comparative Analysis of Word Embedding and Deep Learning for Arabic Sentiment Classification,” *Electronics*, vol. 12, no. 6, 2023, doi: 10.3390/electronics12061425.
[23] H. Yadav and A. Thakkar, “NOA-LSTM: An efficient LSTM cell architecture for time series forecasting,” *Expert Systems with Applications*, vol. 238, p. 122333, 2024, doi: 10.1016/j.eswa.2023.122333.
[24] S. Y. Ali and H. Maseeh, “Dropout: An Effective Approach to Prevent Neural Networks from Overfitting,” *Asian Journal of Research in Computer Science*, vol. 18, no. 2, pp. 163–185, Feb. 2025, doi: 10.9734/ajrcos/2025/v18i2569.
[25] A. Maniatopoulos and N. Mitianoudis, “Learnable Leaky ReLU (LeLeLU): An Alternative Accuracy-Optimized Activation Function,” *Information*, vol. 12, no. 12, 2021, doi: 10.3390/info12120513.
[26] Y. Kang, C. Zhang, Z. Sun, and Y. Li, “Investigating the effect of publication text similarity between reviewers and authors on the rigor of peer review: An intellectual proximity perspective,” *Journal of Informetrics*, vol. 19, no. 3, p. 101709, 2025, doi: 10.1016/j.joi.2025.101709.
[27] K. Park, J. S. Hong, and W. Kim, “A Methodology Combining Cosine Similarity with Classifier for Text Classification,” *Applied Artificial Intelligence*, vol. 34, no. 5, pp. 396–411, 2020, doi: 10.1080/08839514.2020.1723868.
[28] I. Loshchilov and F. Hutter, “Decoupled Weight Decay Regularization,” *7th International Conference on Learning Representations, ICLR 2019, New Orleans, LA, USA, May 6-9, 2019*, 2019,
[29] P. Zhou, X. Xie, Z. Lin, and S. Yan, “Towards Understanding Convergence and Generalization of AdamW,” *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 46, no. 9. pp. 6486–6493, Sep. 2024, doi: 10.1109/TPAMI.2024.3382294.
[30] J. Tu and Z. Wu, “Inherently interpretable machine learning for credit scoring: Optimal classification tree with hyperplane splits,” *European Journal of Operational Research*, vol. 322, no. 2, pp. 647–664, 2025, doi: 10.1016/j.ejor.2024.10.046.