Exploring Machine Learning Methods for Author Identification on Micro-Messages

by

Sarp Aykent

A dissertation submitted to the Graduate Faculty of
Auburn University
in partial fulfillment of the
requirements for the Degree of
Doctor of Philosophy

Auburn, Alabama
May 06, 2023

Keywords: Author Identification, Convolutional Neural Network, Micro-messages

Approved by

Cheryl D. Seals, Chair, Charles W. Barkley Professor of Computer Science & Software Engineering
Jakita O. Thomas, Philpott-WestPoint Stevens Associate Professor of Computer Science & Software Engineering
Gerry V. Dozier, Charles D. McCrary Eminent Chair Professor of Computer Science & Software Engineering
Yang Zhou, Assistant Professor of Computer Science & Software Engineering

Abstract

Author Identification, also known as Authorship Attribution, is the task of identifying an author of unknown text based on the writing style captured within a dataset of writing samples. Author Identification is used in a wide variety of fields including marketing, forensic linguistics, and influence tracing. The writing samples can be found in different forms based on their audience, length, and platform. The common forms of writing samples include books, articles, emails, and messages. With the increasing use of social media, millions of micro-messages, which are short messages with a length constraint, are exchanged daily. Although micro-messages are a powerful and efficient way to communicate among individuals, their anonymity and short-length characteristics give rise to a real challenge for Author Identification. As the majority of Author Identification research is focused on finding authors of long texts, the development of social media platforms, and the emergence of social media as a primary mode of communication has increased the interest in Author Identification of micro-messages. The increase in micro-message has attracted increasing attention in many fields such as social media forensics. The task of identifying authors of micro-messages has been shown to be more difficult than Author Identification using long texts. In this work, we systematically design a set of neural network approaches to tackle this problem. Novel evolutionary algorithms and neural network architectures are developed and thoroughly tested to validate their effectiveness in unique environments. Empirically, our proposed method successfully outperforms other state-of-the-art methods in identifying the authors of micro-messages.

Table of Contents

List of Figures

List of Tables

Chapter 1

Introduction

Exchanging information is crucial for society. Over the years, different mediums have been used to exchange information. With technological advancements, one of the popular ways to connect with people is with Micro-Messages. We can share our thoughts with one or many people at the same time.

With the increased use of micro-messages, the growing number of fake accounts, and their impact on shaping public opinion, tracing the source of messages has become an important concern for social-media companies as well as governments. Identifying the true author of a message is necessary in tasks such as unmasking an anonymous author, detecting fake accounts, identifying plagiarism, or finding ghostwriters.

1.1 Research Goals

In this research, we are going to explore the machine learning methods to identify the authors of Micro-Messages from a given a set of authors.

Our goal is to capture reusable stylistic writing styles. We wish to construct models that can use a combination of this stylistic information to capture the identity behind the message. This identity can be tracked over time and evolved just like environments that identify the authors of the messages.

1.2 Research Overview

The preliminary research which has been completed focuses on introducing novel machine learning methods to tackle the Author Identification problem. In the first phase, we evaluated

these methods on a dataset of more than 1,000 authors, and each author having at least 200 writing samples. We successfully identified the authors of the writing samples with 42.48% to 57.18% accuracy.

The second phase will be consisting of extending the preliminary investigations in two directions. First, we will evaluate the machine learning methods discussed in the preliminary research on extended datasets. The extended datasets will be collected from different sources and will have different size limitations compared to the dataset used in preliminary research. Second, we will study the performance of the machine learning methods by considering different combinations of the number of authors and the number of writing samples. This computational study will show the robustness of the proposed machine learning methods with respect to the different levels of writing sample availability per author.

The final phase of research will be concerned with the robustness of the proposed methodologies. Adversarial approaches will be employed to investigate the weaknesses of the proposed machine learning methods. We will use the distribution of adversarial samples to improve the robustness of the Author Identification systems.

1.3 Challenges

Author Identification plays a critical role in Natural Language Processing (NLP) research. Unlike the other NLP tasks, the Author Identification task relies heavily on exploring authors' linguistic styles. A number of works [12, 25, 27] have developed techniques to explore the stylistic features across documents. Although these works produce encouraging results on style-related tasks, several challenges still remain underexplored in the Author Identification of micro-messages task.

One of the greatest challenges in the author identification task is that the content of the documents varies heavily among the documents written by the same author. Thus the same author could write on various topics about different opinions. This is a clear distinction between other language understanding tasks and author identification tasks. The existing methods for natural language understanding are mainly concerned with extracting the knowledge from the document such as machine translation and question answering. Unlike other natural language

understanding tasks, the linguistic styles of an author play a crucial role in Author Identification.

Also, multiple factors can affect the writing style of the author including psychological and environmental factors. These factors cannot be directly observed since we are only interested in the documents. Some of the psychological factors include the state of mind, happiness, and tiredness of the author at the time of writing. These signals can introduce a noise that can mislead the methods to differentiate stylistic behaviors.

Furthermore, Micro-Messages introduce novel challenges by limiting the length of the document. The authors can only write documents shorter than the size limit of the platform. Thus, they are using different techniques to shorten their documents. This process also makes the problem harder by abstracting the author's writing style in a compressed document.

1.4 Contributions

In this section, we will outline our contributions for Authorship Attribution of Micro-Messages.

- We proposed a distributed evolutionary architecture to overcome the scaling issues. In real-life environments, there can be many candidate authors. When the number of authors increases, computational complexity increases with it. Our proposed method, Distributed Neural-Evolutionary Hybrid (DiNEH), allows horizontal scaling with respect to the number of authors. DiNEH introduces an expert for each author responsible for identifying the author's writing samples. The experts do not require to communicate with each other while they are being trained. Therefore, the authors can be placed in separate machines and utilize more resources.

- We introduced a Multi-Channel CNN model that takes advantage of various features. We analyzed the effectiveness of the features in varying training lengths. In addition to features, we conducted an ablation study to show what improves the performance on Multi-Channel CNN architectures for Author Identification.

- We proposed the use of refinement blocks in Multi-Channel CNN for Author Identification. This work shares the Multi-Channel property of the previous work discussed

but also extends it by using an extra-linguistic feature. In the refinement block, we use identity mapping in parallel to stacked convolutional layers.

- In addition, one of the expected contributions would be to create more comprehensive test datasets to study the robustness of various Author Identification algorithms.

Chapter 2

Background

In the literature, a variety of methods have been developed and used for Author Identification [43]. Stamatatos [60] classified Author Identification methods into two groups as profile-based or instance-based approaches according to how authors' training texts are treated by these methods. In the profile-based approach, all training texts of an author are combined into a single profile document, from which the properties of the author's style are extracted. In the instance-based approaches, each training text of an author is treated as a separate instance of the author's style. While traditional Author Identification methods, such as probabilistic methods, compression models, and common *n*-grams and variants are profile-based approaches, modern machine learning methods utilize the instance-based approach. The instance-based approach can capture the variability in the writing styles of an author, which is an important advantage for Author Identification of micro-messages since authors tend to use different writing styles in different micro-messages. Therefore, the methods proposed in this dissertation use the instance-based approach.

2.1 Stylometric Features

The first step of Author Identification is to extract a set of features from sample texts. A feature represents a distinctive attribute of a given text. By calculating a set of features, we can represent text samples with different sizes and contents as a standard feature vector that can be used to evaluate and compare those texts. A wide variety of features have been used for increasing the effectiveness of Author Identification [12,25,27]. These features can be grouped as lexical, character, and syntactic features as given in Table 2.1.

Table 2.1: Features used for Author Identification [60].

Feature Type	Feature	Used in the Previous Work
Lexical	Token-based	
	Vocabulary richness	[42]
	Word *n*-grams	[25, 55]
	Word frequencies	[42]
	Bag-of-Words	[12]
Character	Character types	[42]
	Character *n*-grams	[25, 42, 52, 55]
Syntactic	Parts of speech	[52]
	Errors	

In addition to the features given in Table 2.1, deep neural networks for Author Identification can also utilize word embeddings, which are pre-trained representations for text samples where individual words are mapped to real-valued vectors. One of the advantages of using world embeddings is that a group of words with similar meanings can be represented by the same vector. In this dissertation, we have developed a novel Multi-Channel CNN architecture that processes different features of text via word, character, and parts of speech (POS) embeddings. We hypothesize that this combination of various word embeddings can capture different stylometric features, hence, can improve the accuracy of CNN for Author Identification.

2.2 Techniques

Luyckx et al. [37] used memory-based learning and discovered that increasing the number of authors within a dataset has an adverse effect on Author Identification accuracy while increasing the number of writing instances per author as well as the size of each writing instance increases the identification rate. Baron [4] compared the following methods: Decision Trees

(PART and C4.5), Random Forest, *k*-Nearest Neighbors, Multilayer Perceptrons, Radial Basis Function Networks, and Naive Bayesian Classifiers for identifying authors. Their research suggests that cross-validation can result in over-estimation for Authorship Attribution.

In [6, 25, 32, 59, 62], the authors used similarity-based methods that utilized the concept of relative and cross-entropy. In [12, 14, 21], Support Vector Machines were used. Jockers and Witten [21] introduced the concept of introduced Nearest Shrunken Centroids (NSC) and Regularized Discriminant Analysis (RDA).

Bagnall [3] employed a multi-headed recurrent neural network that predicts the next character in the sequence using the previously seen characters. For identification, each author generated different sets of probabilities for the next character. Each author's model was trained with a chance of leakage to learn from other authors' texts in order to learn the language as a whole. This was the best performing approach for the PAN 2015 multi-language author identification tasks without using excessive techniques specific for each language.

Traditional Convolutional Neural Networks (CNNs) use either word sequences [28,51,53] or character sequences [53, 57] as an input. First, Kim [28] proposed a CNN architecture for sentence classification tasks. The architecture utilized multiple convolutional filter sizes (3, 4, and 5), dropout rate, and max-over-time pooling [9]. Similar architectures with modifications were later used by other researchers [51, 53, 57]. Kim's [28] proposed approach has different variations with a randomly initialized model, static model, non-static model, and Multi-Channel model (that combines static and non-static models). Kim [28] reported that these models, excluding the random model, performed similarly. The performance of the three above mentioned models were competitive against the other methods used. Similarly, Rhodes [51] proposed an approach for sentence classification using CNNs to attribute texts to authors. The hyperparameters were selected based on the performance on a dataset generated by the author, i.e., the Canada dataset. Then using the selected hyperparameters, the model was tested with the PAN 2012 author identification task [23]. The proposed approach correctly classified 12 writing samples out of 14 writing samples. Rhodes' [51] approach was better than many approaches except one which correctly classified thirteen writing samples. Later, Ruder et al. [53]

used a combination of previously proposed approaches by [9, 28, 67]. These combinations include single-channel and Multi-Channel CNNs with word and character embeddings. Ruder et al. [53] tested their approaches with state-of-art authorship attribution methods. In their experiments, they selected 10 and 50 authors with the most writing samples from five large datasets. The proposed approaches outperformed other methods in all five datasets except the IMDb dataset. The single-channel character embeddings outperformed all methods on three datasets. The Multi-Channel character and word embedding hybrid outperformed all other methods on IMDb dataset. Recently, Shrestha et al. [57] used a different approach for identifying the writers of short texts. The authors used a sequence of character n-grams for the input. The proposed approach performed better than the state-of-the-art approach by Schwartz et al. [56]. Furthermore, the authors analyzed the high performing n-grams in the best model. The CNNs were able to capture the stylistic behavior of both humans and chatbots. This also contributed to the ongoing analysis of the high performing n-grams on Authorship Attribution Tasks [26, 55].

This dissertation addresses several important gaps in the deep learning methods cited above for Author Identification:

- We design a novel Multi-Channel CNN architecture and a multi-branch refinement convolution block that can fully exploit micro-message data for authorship attribution. Applying multi-branch refinement convolution blocks and Multi-Channels simultaneously in CNNs has not been studied in the literature. Combining these two techniques can increase the accuracy of Author Identification.

- We integrate the concept of sharing convolutional filters on two embedding layers to capture the different stylometric features in our architecture, which renders the word/character tokens' conversion to vectors possible.

- The importance of features wasn't extensively studied in the literature. It is important to know the most salient features in neural networks. We investigate the relevance of each feature in neural networks and show the effectiveness of different feature types. We show that different feature sets can capture various types of stylometric features.

- We study the generalization and robustness of the proposed neural network architectures by conducting extensive experimental evaluations on the real-world Twitter dataset. This study demonstrates the superiority of our proposed Multi-Channel CNN architecture and refinement block in comparison to baseline algorithms.

Chapter 3

Datasets

This section introduces the micro-message datasets that will be used in the following sections to gauge the performance of the algorithms and models proposed in this dissertation. In addition, the dataset properties are investigated to understand the environmental factors, including message size, topics, and interactions among users. The datasets were collected from social media platforms namely, Twitter and Reddit.

3.1 CASIS

The CASIS dataset is subsets of CASIS-1000 [13,40]. The CASIS-1000 dataset is composed of blog entries from 1,000 authors. There are four writing samples for each of the 1,000 authors. Hence, there are a total of 4,000 writing samples in this dataset. Each subset, namely CASIS-N, is the first n authors of the CASIS-1000 dataset.

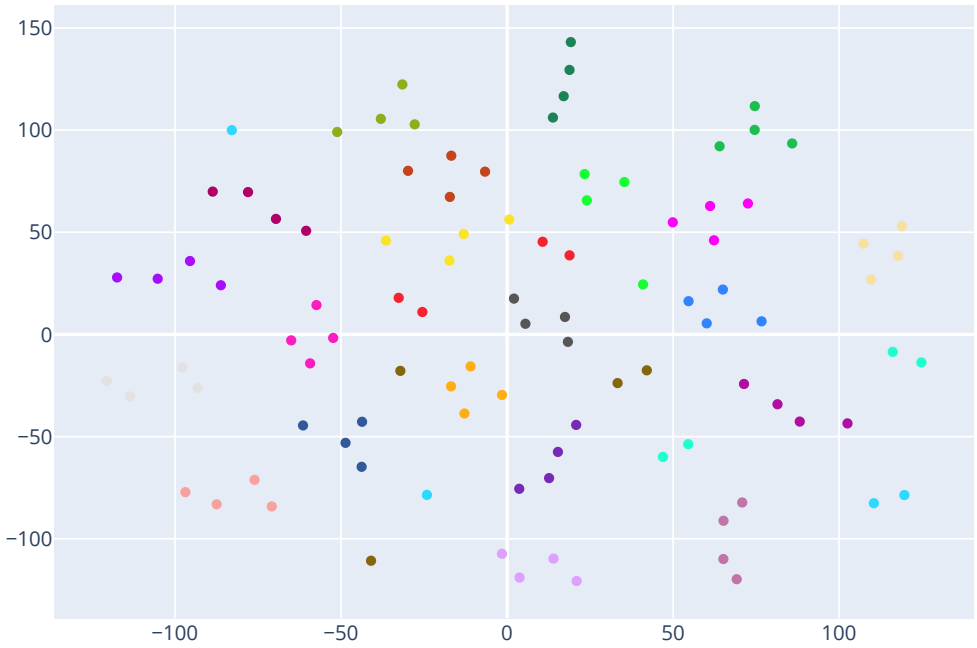

Figure 3.1: *t*-SNE plots of CASIS-25 subset.

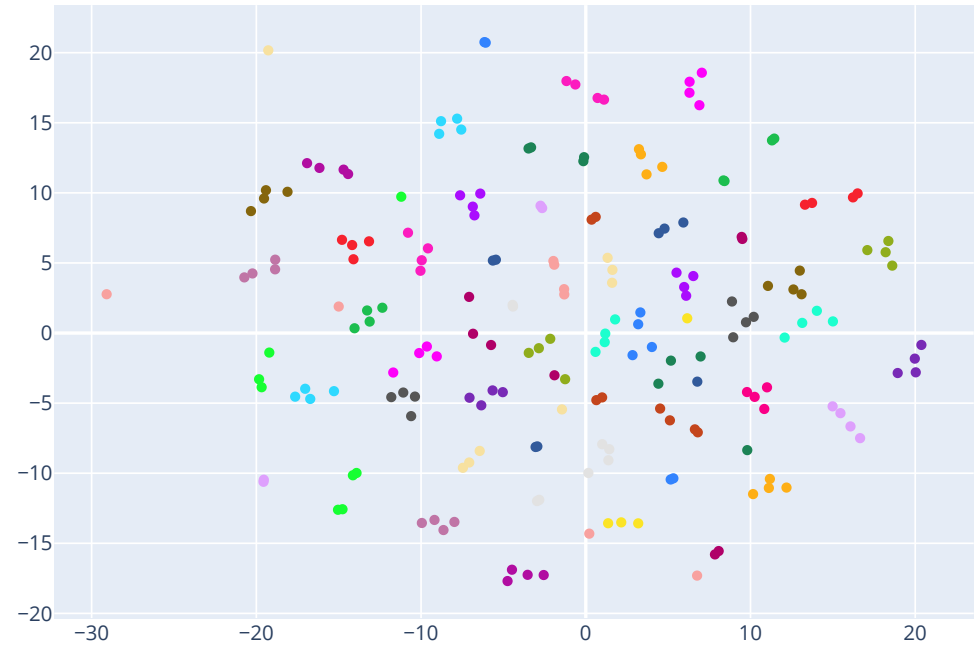

Figure 3.2: *t*-SNE plots of CASIS-50 subset.

Figure 3.1, Figure 3.2, Figure 3.3, and Figure 3.4 shows the visualization of CASIS-N subsets after Term Frequency-Inverse Document Frequency (td-idf) [54] transformation via *t*-Distributed Stochastic Neighbor Embedding (*t*-SNE) plots [39]. The tf-idf transformation scales down the impact of tokens that occur very frequently in a given corpus. Each point in the plot represents a writing sample, and the color of the points indicates the authors. We observe that the points with same color in Figure 3.1, Figure 3.2, and Figure 3.3 are grouped together. This shows that even unsupervised methods with td-idf transformations can capture authorship signals. However, this behavior is harder to observe in Figure 3.4 where the 1000 authors are visualized.

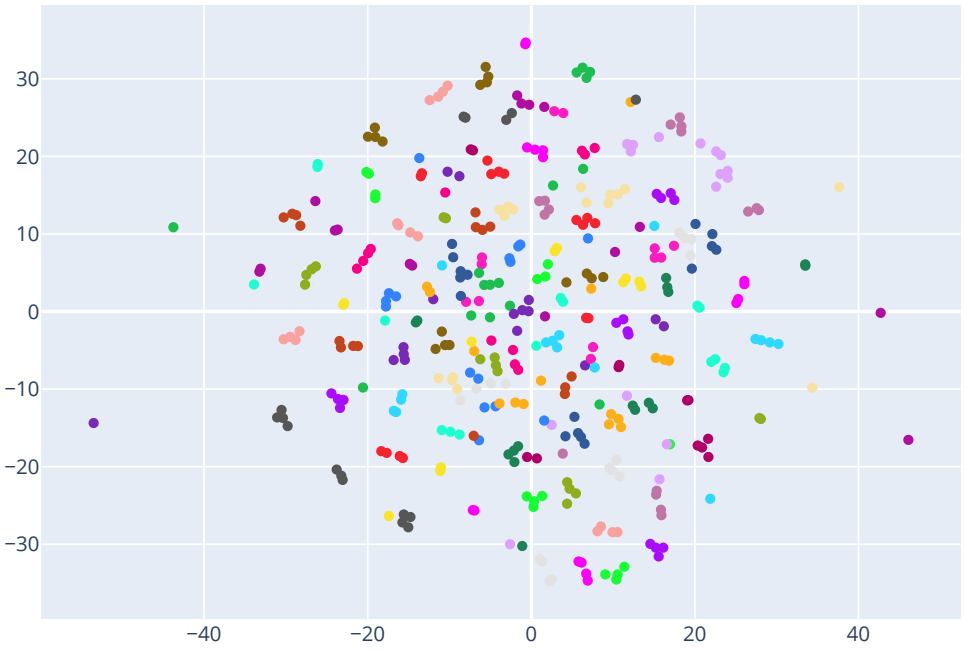

Figure 3.3: *t*-SNE plots of CASIS-100 subset.

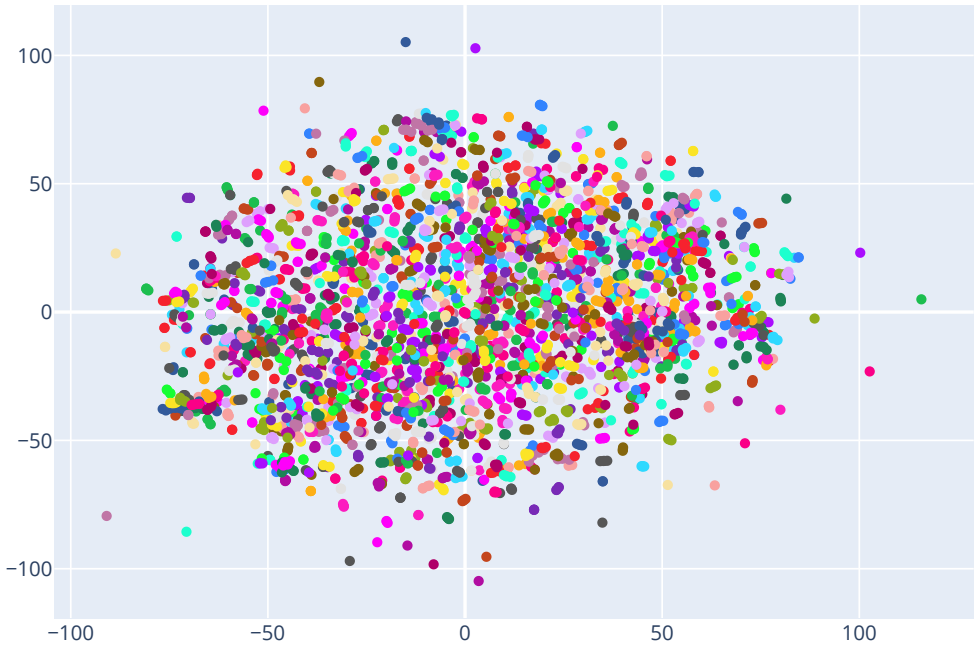

Figure 3.4: *t*-SNE plots of CASIS-1000 subset.

3.1.1 Preprocessing

The raw features collected from the writing samples are first preprocessed before the classification. Preprocessing of the feature vectors is done with the scikit-learn python library [47]. Each feature vector is passed through a td-idf transformer, then through StandardScaler, and finally normalized. The StandardScaler standardizes each feature by subtracting the mean and dividing it by the variance per feature. The normalization scales each feature vector to a unit vector.

3.1.2 Statistics

The feature sizes for subsets of CASIS-1000 are shown in Table 3.1 for both author-based and global Bag-of-Words. It is clear that the difference between a single feature set and an average author feature set increases when the dataset gets larger.

Table 3.1: Bag-of-Words feature sizes.

Dataset	Single Feature Set	Author Feature Sets	
		Average	Maximum
CASIS-25	6,082.00	488.20	1,147.00
CASIS-50	9,984.00	523.16	1,403.00
CASIS-100	14,351.00	491.00	1,849.00
CASIS-1000	52,919.00	488.81	2,084.00
C50	35,202.00	2,177.09	4,539.00

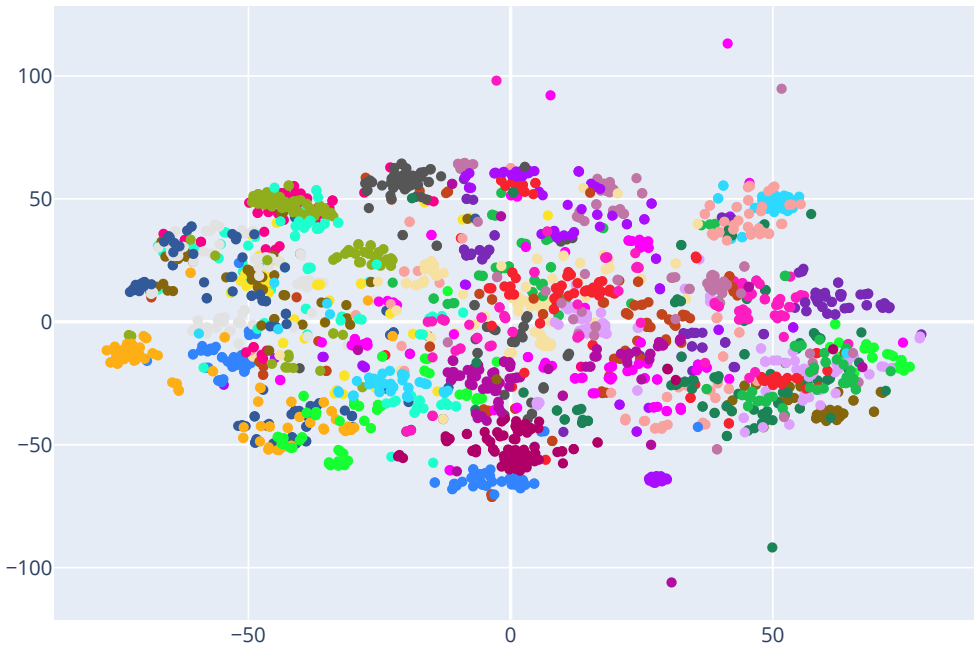

Figure 3.5: *t*-SNE plots of the C50 train set.

3.2 C50

The C50 dataset is widely used for Authorship Attribution [20]. The C50 dataset is composed of news articles of 50 authors. The dataset has 50 writing samples of each author for each

training and test set. Hence, there are a total of 2,500 writing samples for the training set and 2,500 writing samples for the test set. Figure 3.5 and Figure 3.6 shows the visualization of the train and test sets, respectively.

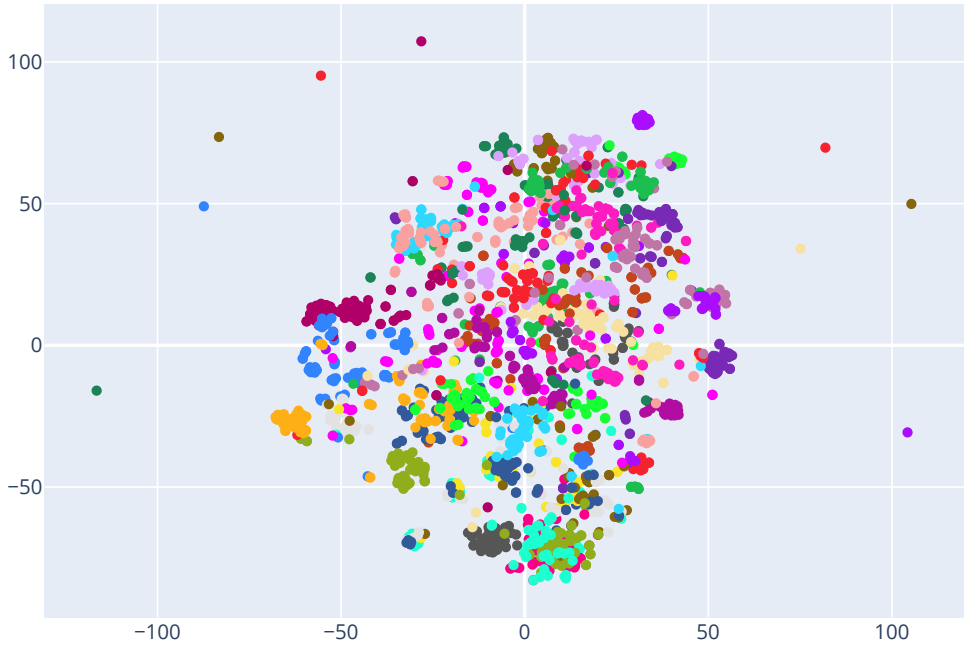

Figure 3.6: *t*-SNE plots of the C50 test set.

3.2.1 Preprocessing

The preprocessing steps in Section 3.1.1 were followed. The features are collected from the writing samples. Then, the features are passed through a tf-idf transformer, StandardScaler and normalized to a unit vector.

3.2.2 Statistics

Feature sizes for subsets of C50 are shown in Table 3.1. As in Section 3.1, one can see the difference between a single feature set and an average author feature set increases when the dataset gets larger.

3.3 Twitter

The Twitter dataset [56] is a collection of public tweets from May 2009 to March 2010. This dataset is used in a number of micro-message research concerning author identification [56,57]. The dataset has two groups of subsets. The subsets includes authors with at least 200 and 1,000 writing samples, respectively. The first subset contains a larger number of authors with fewer writing samples. It allows us to experiment on a dataset with a larger number of authors. Figure 3.7 shows the visualization of the dataset. As seen in the figure, some writing samples form clusters. For example, writing samples from authors on the right side of the Figure 3.7 with yellow, purple, dark green, and light blue are clustered together. This clustering is an indication of highly similar writing samples. The bots that post automated messages have similar properties since they tend to follow the same structure. We sampled writing samples from those clustered authors on the right side of Figure 3.7 and listed on Table 3.2. Table 3.2 lists the content of the writing sample in the first column and the author in the second column. The authors are listed in the same order and the same color as in Figure 3.7.

3.3.1 Preprocessing

We used several preprocessing methods to eliminate self-identifying writing samples and removed the noise. Self-identifying information including username references and website URLs replaced with pre-defined meta tags. In addition, we replaced the noisy information such as numbers, dates, and time with meta tags to avoid over-fitting. Since we need as much information as we can get from the tweets, we did not convert the text into lowercase to keep the information. The authors' choices of different types of word capitalization may reflect their writing style. These types include camelCase, PascalCase, snake_case, and kebab-case.

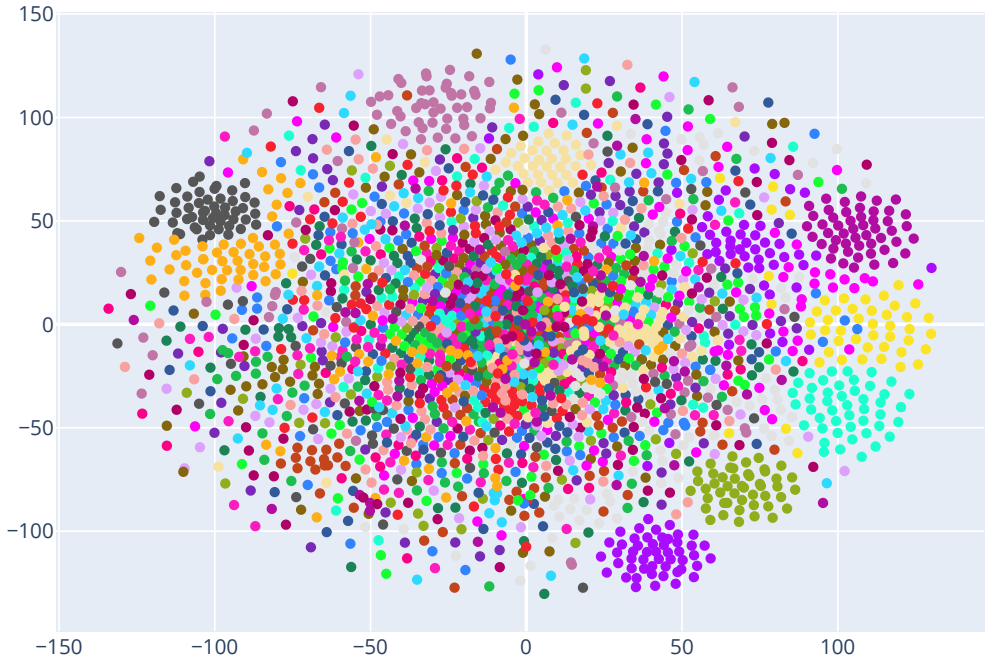

Figure 3.7: *t*-SNE plot of 50 authors randomly sampled from the Twitter dataset.

Table 3.2: Writing samples collected from the Twitter dataset.

Text	author
News Update: Windows SteadyState 1	1
News Update: Koobface : Check and Remove The Facebook Virus -...	1
VIDEO David Choi ? Valentines (Original): Like the video? Twi...	2
UCLA vs. Arizona State Recap: Eric Boateng had a huge game fo...	2
#mma UFC 4 Preview: Tyson Griffin Braces for Potential War Wi...	3
- GAMBURYAN WINS, LIGHTWEIGHTS SHOW OFF AT WEC: In a night of...	3
Hint #4 - If you are wrapping crystal chandeliers or lig...	4
Hint #4 - Your #4 source of research is eBay's past sale...	4
Anykind boat repairs (all) $4 1 #boats #sale	5
Pro Trail Bass Boat Trailer Low Profile (Avon) $4 1 #boats #...	5

3.3.2 Statistics

Table 3.3 shows the Twitter dataset [2] statistics for subsets of the dataset used in the experiments with 100, 200, 500, and 1,000 authors (a). Similarly, Table 3.4 shows the dataset statistics for subsets of the dataset used in the experiments with 50, 100, 200, and 500 writing samples (w). Tables 3.3 and 3.4 have the same structure. The first column shows the number of authors (a) in Table 3.3 and the number of writing samples (w) in Table 3.4. Then, the mean (μ) and standard deviation (σ) of the number of characters, words, and sentences for writing samples are given. The last column of Tables 3.3 and 3.4 shows the dictionary size ($|D|$), which is the number of unique words in the dataset.

In Table 3.3, the average number of characters are within the range of 72 ± 2 for all authors. The character limit for Twitter is 140, the average number of characters are close to half the size of the limit. This trend is also observable in the average number of words and sentences. Please note that each author group is sampled without replacement; hence, they are disjoint sets.

On the other hand, the dictionary size grows when the number of authors increases, which is expected since there are a larger number of writing samples for groups with a larger number of authors. The increase in the dictionary size was less than the increase in the number of authors since it gets harder to find a unique word when the dictionary is larger.

Table 3.3: Dataset statistics: varying number of authors.

a	Characters		Words		Sentences		$\|D\|$
	μ	σ	μ	σ	μ	σ	
100	71.57	33.81	14.10	6.50	1.69	0.90	47493
200	73.15	34.04	14.29	6.49	1.71	0.96	81735
500	73.79	34.11	14.42	6.55	1.69	0.95	161742
1,000	73.28	33.84	14.30	6.49	1.71	0.96	269774

In Table 3.4, the average number of characters are within range of 73 ± 0.5 for all values of w. In addition, the dictionary size gets larger with increasing number of writing samples.

Similar to the varying number of authors, an increase in writing samples results in a larger dictionary size.

Table 3.4: Dataset statistics: varying number of writing samples.

w	Characters		Words		Sentences		$\|D\|$
	μ	σ	μ	σ	μ	σ	
50	73.26	34.22	14.17	6.45	1.66	0.94	10,857.50
100	73.04	34.20	14.15	6.45	1.65	0.92	18,093.50
200	73.04	34.16	14.15	6.44	1.65	0.93	29,941.50
500	72.87	34.11	14.12	6.43	1.65	0.92	56,974.60

Chapter 4

Author Identification via a Distributed Neural-Evolutionary Hybrid (DiNEH)

In this chapter, we propose a non-traditional Genetic & Evolutionary Feature Selection (GEFeS) method for Author Identification. GEFeS is distributed algorithm because it evolves a feature vector for each author in a distributed manner. We refer to this new approach as a distributed neural evolutionary hybrid (DiNEH). We compare the performance of DiNEH with a number of well-known Authorship Attribution Techniques (AATs) from the literature. DiNEH was able to outperform all of the AATs on one dataset and was the second best performing on the other dataset by a narrow margin.

4.1 Introduction

Over the years, the field of Evolutionary Computation (EC) has seen a wide variety of approaches to distributed evolutionary computations (DECs). To date, DEC approaches can be classified as: domain-based [5, 49, 61], function-based [8], and/or variable-based [49]. In domain-based DECs, the population of candidate solutions, P, is distributed across k processors where each processor receives P/k candidate solutions (individuals). Function-based DECs distribute the functions of evolutionary operators and processes (e.g., Crossover, Function Evaluation, Selection, etc.) across k processors while variable-based DECs distribute the chromosome across k processors. In this chapter, we present a distributed neuro-evolutionary hybrid (DiNEH) approach that combines domain-based, function-based, and variable-based DEC. Our results show that DiNEH is an effective method for Author Identification [43].

Scalability is a big concern for Authorship Identification tasks as examined in [36]. We propose a distributed method for feature selection. DiNEH consist of a set of author cells (one

author cell for each author). Figure 4.1 illustrates the diagram of the feature selection process in author cells. As shown in Figure 4.1, an author cell is composed of six components. The first two components, self and non-self, represent the dataset as a whole. In an author cell, however, the self set contains only writing samples for the associated author while the non-self set represents the writing samples of the other authors. Therefore, an author cell represents a two-class identification problem, reducing the complexity of the classification problem from many to two. Each author cell has a Linear Support Vector Machine (LSVM) that is trained using leave-one-out. Finally, each author cell uses a GEFeS, which evolves a population of a candidate feature mask. The objective of GEFeS is to identify the best subset of features for each author.

4.2 Single GEFeS Method

The GEFeS is based on the steady-state genetic algorithm [10] that evolves a population of 20 candidate feature masks. The initial population of candidate feature masks is generated using a binary standard uniform distribution function. Hence, each feature mask generated in the initial population is anticipated to utilize half of the total number of features. In each generation, two parents are selected via binary tournament selection, and one offspring feature mask is created using a uniform crossover with a mutation rate of 5%. The worst feature mask in the population was replaced with the new offspring feature mask.

We use LSVMs because of their ability to cope with a large number of inputs and their applications for Authorship Identification in the literature [12]. The Scikit-learn library was used for the LSVM with the LinearSVC module. GEFeS calculates the accuracy of the classification using leave-one-out cross-validation.

In order to calculate the accuracy of a candidate feature mask, each writing sample is classified one by one. Each writing sample of the dataset classified by the LSVM results in a decision vector that has a number associated with each author in the dataset. The author with the highest score in the decision vector is selected as the author of that writing sample.

Three different feature sets used were Character Unigrams, Stylometry, and Bag-of-Words. The features that were used for Stylometry were similar to [42] and shown in Table 4.1.

Table 4.1: Stylometry feature set.

Category	Description	Count
Length	number of words/characters in post	2
Vocabulary richness	Yule's K and frequency of *hapax legomena*, *dis legomena*, etc.	12
Word shape	frequency of words with different combination of upper- and lower-case letters.	5
Word length	frequency of words that have 1-20 characters.	20
Letters	frequency of *a* to *z*, ignoring case	26
Digits	frequency of *0* to *9*	10
Punctuation	frequency of *.?!,;:()"-'*	11
Special Characters	frequency of other special characters "~@#$^&*_+=[]{}|/<>	22
Function words	frequency of words like 'a', 'about', 'after' etc.	320
Total		428

Note: The syntactic category pairs were omitted and different function words were used from [42].

4.3 DiNEH

DiNEH uses the author cell GEFeSs for evaluating the feature masks within the author cells. The fitness function for the GEFeS used in each author cell is as follows:

$$f(x) = \begin{cases} y & x > 0 \\ x + 1 & x \leq 0 \end{cases} \tag{4.1}$$

where x is the decision function of the LSVM such that $x > 0$ indicates that the writing sample belongs to the author, $x \leq 0$ otherwise, and y represents the number of writing samples belongs to the author.

$$fitness = \sum_{i=1}^{y} f(\text{LSVM}(self_i)) \tag{4.2}$$

where $self_i$ denotes the *i*th writing sample of the author.

The classification of an author cell LSVM is performed in the same way as with the single GEFeS approach. Each author cell classifies a writing sample using its feature mask. The

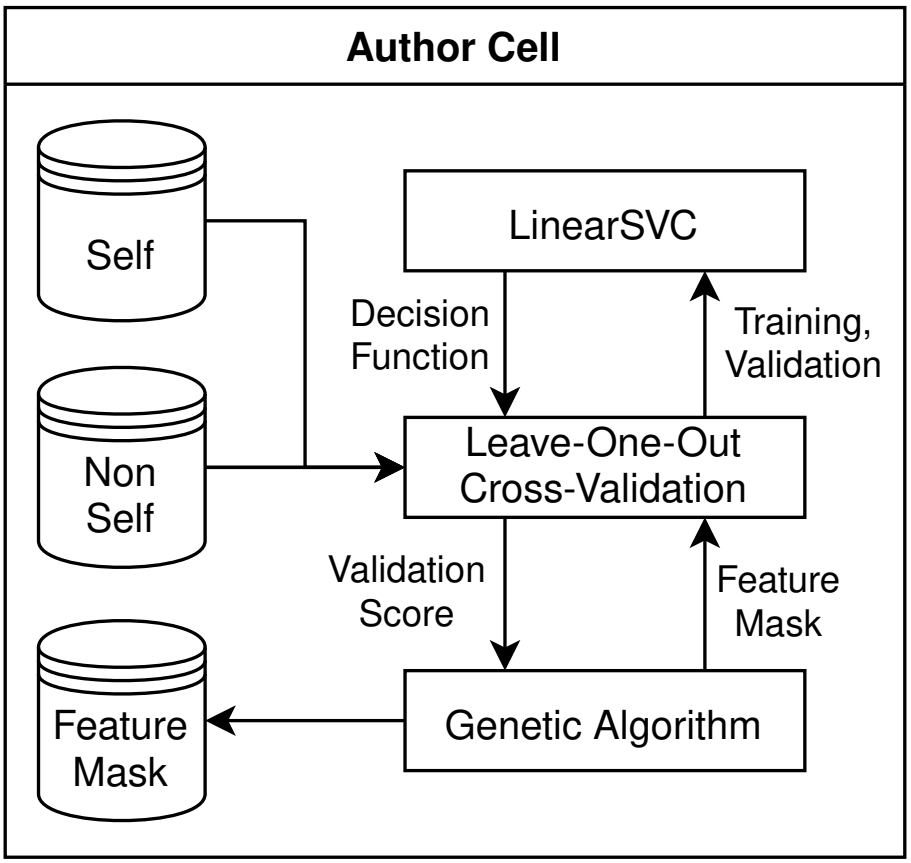

Figure 4.1: Diagram of the feature selection process in author cells.

resulting decision vector of the classification is a score between -1 and 1 such that -1 indicates that the sample belongs to another author and 1 indicates that the sample strongly belongs to the author.

DiNEH uses the same feature sets introduced in as the single GEFeS mentioned earlier, i.e., Character Unigrams, Stylometry, and Bag-of-Words. For Bag-of-Words, each author cell is trained with the Bag-of-Words associated with only their writing samples.

4.4 Experiments

For Authorship Attribution, we first extracted the features described in Section 4.2 from a set of writing samples. The writing samples were collected from the CASIS and C50 datasets described in Sections 3.1 and 3.2. Then, we used preprocessing techniques that are explained in the preprocessing section. We conducted feature selection experiments with the Single GEFeS

and DiNEH approaches on subsets of the CASIS-1000 dataset. Then, we compared the DiNEH approach with the authorship attribution systems proposed in [6, 25, 32, 59, 62].

In Experiment I, the baselines for CASIS-25, 50, 100, and 1000 were computed using a LSVM without GEFeS for the Character Unigram, Stylometry, and Bag-of-Words features. In Experiments II, III, and IV, the performance of Single GEFeS and DiNEH was compared for the CASIS-25, 50, and 100 datasets using Character Unigram and Stylometry[1].

In Experiment V, the performance of the five well-known author identification systems [6, 25, 59, 62] were compared with DiNEH methods on the CASIS-25, 50, 100, 1000, and C50 datasets.

4.5 Results

In the computational experiments, a maximum of 5,000 function evaluations were used to evaluate Single GEFeS and DiNEH. Both algorithms had population size of 20. Let n be the number of writing samples. Each function evaluation calculated the fitness using leave-one-out cross-validation, and each sample needed to be classified. Therefore, each classifier was trained $(n-1)$ times with all writing samples excluding the test writing sample. The computational effort for the Single GEFeS function evaluation ($\omega_{SingleGEFeS}$) is as follows, where n represents the number of writing samples:

$$\omega_{SingleGEFeS} = n^2 - n \tag{4.3}$$

The computational effort for DiNEH function evaluation (ω_{DiNEH}) was as follows:

$$\omega_{DiNEH} = \frac{n}{m} \times m \times (n-1) = n^2 - n \tag{4.4}$$

where m represents the number of writing instances per author. For the CASIS-1000 dataset, $n = 4,000$ and $m = 4$. For the C50 dataset, $n = 2,500$ and $m = 50$.

[1]The reason why Bag-of-Words was not used in Experiments II, III, and IV is due to inability of the Single GEFeS to scale as shown in Table 3.1

The computational effort required for a single function evaluation in DiNEH and the Single GEFeS approach (Equations (4.3) and (4.4)) were equal. For a fair comparison, the same number of function evaluations were used for DiNEH and the Single GEFeS approach.

4.5.1 Experiment I

Table 4.2 shows the baseline results of the LSVMs without feature selection. The first column lists the dataset used for Experiment I. The second, third, and fourth columns list the baseline accuracies of the LSVMs using Character Unigrams, Stylometry and Bag-of-Words, respectively. In Table 4.2, one can see that as the number of authors increases the accuracy decreases for all LSVMs. An unexpected result is that the Character Unigram LSVM outperformed the Stylometry LSVM on all four datasets. The Bag-of-Words LSVMs had the best overall performance on all four datasets.

Table 4.2: The performance comparison of baseline SVM classifiers. The accuracies are shown for CASIS-N subsets with with the baseline feature sets.

Dataset	Unigram	Stylometry	Bag-of-Words
CASIS-25	65.00%	58.00%	96.00%
CASIS-50	51.00%	44.00%	90.00%
CASIS-100	46.00%	34.25%	84.50%
CASIS-1000	24.53%	19.65%	47.57%

4.5.2 Experiment II

Table 4.3 shows the results of the LSVMs with Single GEFeS and DiNEH on the CASIS-25 dataset 30 times with 5,000 function evaluations for each run. In Table 4.3, the first column list the algorithm and feature set pair used in the experiment. The second column list the equivalent class of the methods that was determined by using ANOVA and the Student's t-test. The third and fourth columns list the best and average accuracies, respectively. The fifth and sixth columns list the lowest and average number of features used in the feature mask, respectively. One can see that DiNEH outperformed Single GEFeS in terms of equivalence class,

best accuracy, and average accuracy. DiNEH$_{uni}$ outperforms Single GEFeS$_{uni}$ in terms of the fewest number of features and the average number of features; however, the same outcome was not observed for DiNEH$_{sty}$ and Single GEFeS$_{sty}$. In this case, Single GEFeS$_{sty}$ outperformed DiNEH$_{sty}$ in terms of the fewest number of features used while DiNEH$_{sty}$ outperformed Single GEFeS$_{sty}$ in terms of the average number of features used. In Table 4.3, DiNEH$_{sty}$ has the best overall performance in terms of accuracy. Figure 4.2 visualizes the number of features used in Single GEFeS and DiNEH with Character Unigram and Stylometry feature sets on the CASIS-25 dataset.

DiNEH$_{uni}$ outperformed Single GEFeS$_{uni}$ in terms of the fewest number of features and the average number of features; however, the same outcome was not observed for DiNEH$_{sty}$ and Single GEFeS$_{sty}$. In this case, Single GEFeS$_{sty}$ outperformed DiNEH$_{sty}$ in terms of the fewest number of features used while DiNEH$_{sty}$ outperformed Single GEFeS$_{sty}$ in terms of the average number of features used. In Table 4.3, DiNEH$_{sty}$ has the best overall performance in terms of accuracy.

Table 4.3: Comparison of Single GEFeS and DiNEH on the CASIS-25 dataset. The results were gathered from accuracy of 30 individual runs. The equivalence classes are calculated by using ANOVA and the Student's t-test.

Algorithm	EQ Class	Accuracy		Number of Features	
		Best	Average	Lowest	Average
Single GEFeS$_{uni}$	III	83.00%	80.70%	49.00	55.17
Single GEFeS$_{sty}$	III	83.00%	81.20%	205.00	219.37
DiNEH$_{uni}$	II	93.00%	92.10%	47.40	48.88
DiNEH$_{sty}$	I	99.00%	98.10%	211.64	213.49

4.5.3 Experiment III

Table 4.4 summarizes the results of the LSVMs with Single GEFeS and DiNEH on the CASIS-50 dataset. In this experiments, the algorithms were run for 30 replications with 5,000 function evaluations for each run. As similar to Experiment II, in Table 4.4, the first column lists the

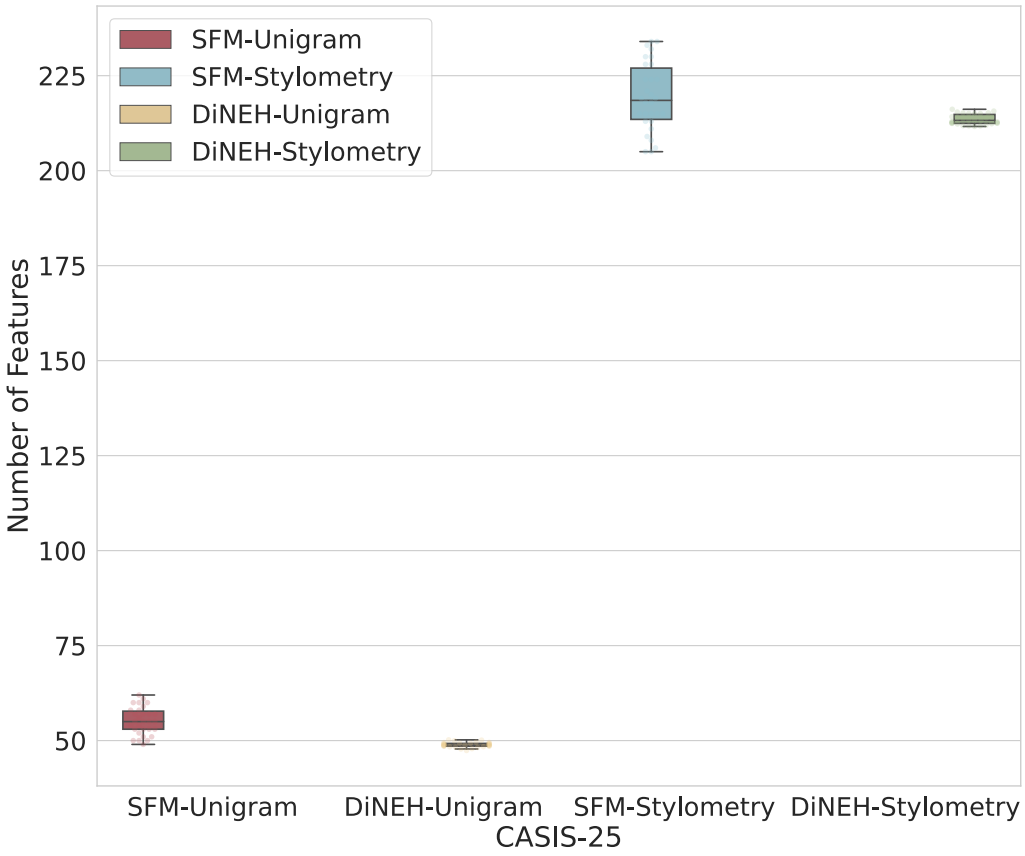

Figure 4.2: Average number of features used in the Single GEFeS approach and DiNEH using the Character Unigram and Stylometry feature sets.

algorithm and feature set pair used in the experiment. The second column lists the equivalent class of the methods that was determined by using ANOVA and the Student's t-test. The third and fourth column list the best and average accuracy values. The fifth and sixth columns list the lowest and average number of features used in the feature mask. As seen Table 4.4, DiNEH outperformed Single GEFeS in terms of all performance measures (equivalence class, accuracy, number of features). The Single GEFeS$_{uni}$ outperformed Single GEFeS$_{sty}$ in terms of equivalence class and accuracy while DiNEH$_{sty}$ outperformed DiNEH$_{uni}$ in terms of equivalence class and accuracy. As in Experiment II, DiNEH$_{sty}$ outperformed the others in terms of equivalence class and accuracy. The average numbers of features used in Single GEfeS and DiNEH with Character Unigram and Stylometry feature sets are shown in Figure 4.3 on the CASIS-50 dataset.

Table 4.4: Comparison of Single GEFeS and DiNEH on the CASIS-50 dataset. The results were based on 30 individual runs. The equivalence classes were calculated by using ANOVA and the Student's t-test.

Algorithm	EQ Class	Accuracy		Number of Features	
		Best	Average	Lowest	Average
Single GEFeS$_{uni}$	III	66.00%	64.80%	53.00	59.93
Single GEFeS$_{sty}$	IV	62.00%	60.55%	214.00	230.93
DiNEH$_{uni}$	II	82.00%	80.90%	47.14	48.47
DiNEH$_{sty}$	I	95.50%	94.60%	210.38	213.53

4.5.4 Experiment IV

Table 4.5 shows the results of the LSVMs with Single GEFeS and DiNEH on the CASIS-100 dataset using 30 random runs with 5,000 function evaluations for each run. In Table 4.5, the first column lists the algorithm and feature set pair used in the experiment. The second column lists the equivalent class of the methods that was determined by using ANOVA and the Student's t-test. The third and fourth columns list the best and average accuracies, respectively. The fifth and sixth columns list the lowest and average number of features used in the feature

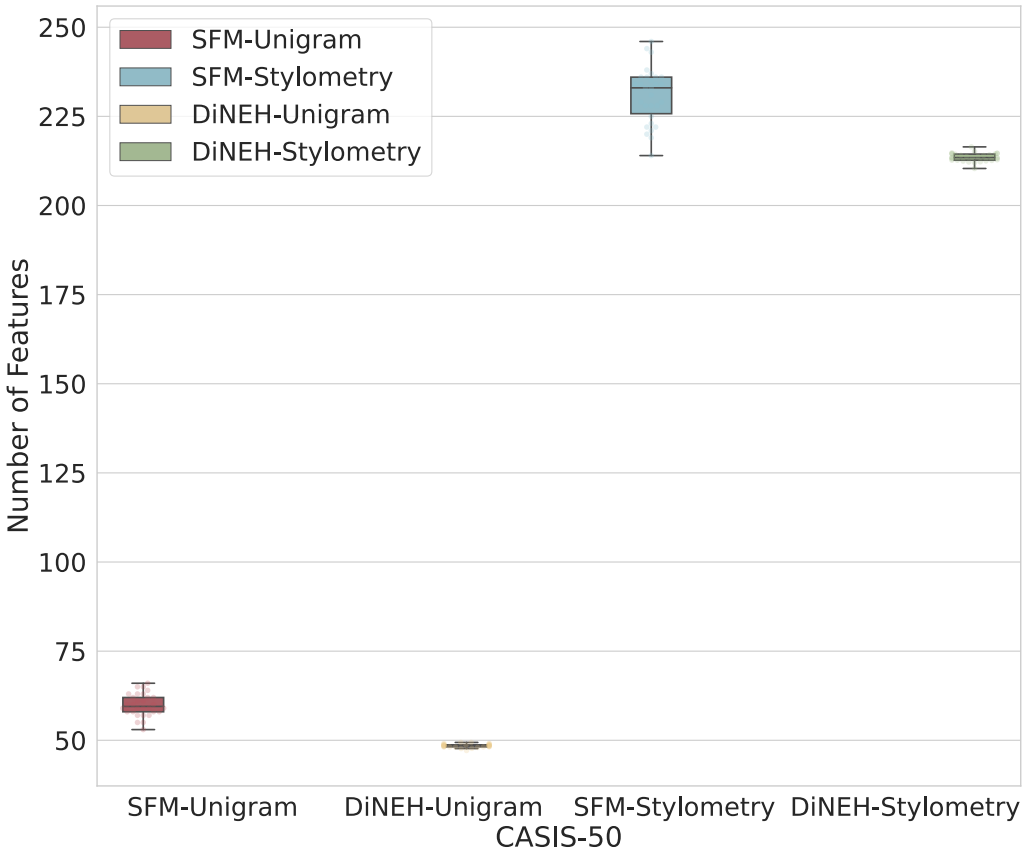

Figure 4.3: Average number of features used in the Single GEFeS approach and DiNEH using the Character Unigram and Stylometry feature sets.

mask, respectively. As seen in Table 4.5, DiNEH outperformed Single GEFeS in terms of equivalence class, accuracy, and the number of features used. As we have noticed earlier in Experiment III, Single GEFeS$_{uni}$ outperformed Single GEFeS$_{sty}$ and DiNEH$_{sty}$ outperformed DiNEH$_{uni}$ in terms of equivalence class and accuracy. In addition, DiNEH$_{sty}$ had the best overall performance. Figure 4.5 provides the plot of the average accuracy for all four methods on the CASIS-25, 50, and 100 datasets. Figure 4.4 visualizes the number of features used in Single GEfeS and DiNEH with Character Unigram and Stylometry feature sets on the CASIS-100 dataset.

Table 4.5: Comparison of Single GEFeS and DiNEH on the CASIS-100 dataset. The results were gathered from 30 individual runs. The equivalence classes were calculated by using ANOVA and the Student's t-test.

Algorithm	EQ Class	Accuracy		Number of Features	
		Best	Average	Lowest	Average
Single GEFeS$_{uni}$	III	54.00%	53.34%	63.00	69.41
Single GEFeS$_{sty}$	IV	47.50%	46.04%	214.00	229.12
DiNEH$_{uni}$	II	78.00%	77.45%	48.34	48.81
DiNEH$_{sty}$	I	90.75%	87.85%	212.44	213.56

4.5.5 Experiment V

In Table 4.6, we compared the DiNEH methods with five well-known author identification systems (AISs). The first column of Table 4.6 lists the methods that are compared. The second, third, fourth, and fifth columns are the accuracies of the five AISs and the average accuracies of DiNEHs on the CASIS-25, 50, 100, 1000, and C50 datasets, respectively. As in Experiments II-IV, the DiNEHs was run for 30 random replications on the CASIS datasets. On the C50 dataset, the DiNEHs was run for 10 replications because of the computational constraints. In terms of the AISs, Keselj had the best performance on all of the datasets. In terms of the DiNEHs, DiNEH$_{BoW}$ had the best performance on all of the datasets except for CASIS-25 where it was tie with Keselj, and on C50 dataset, it was the second best performing one after Teahan.

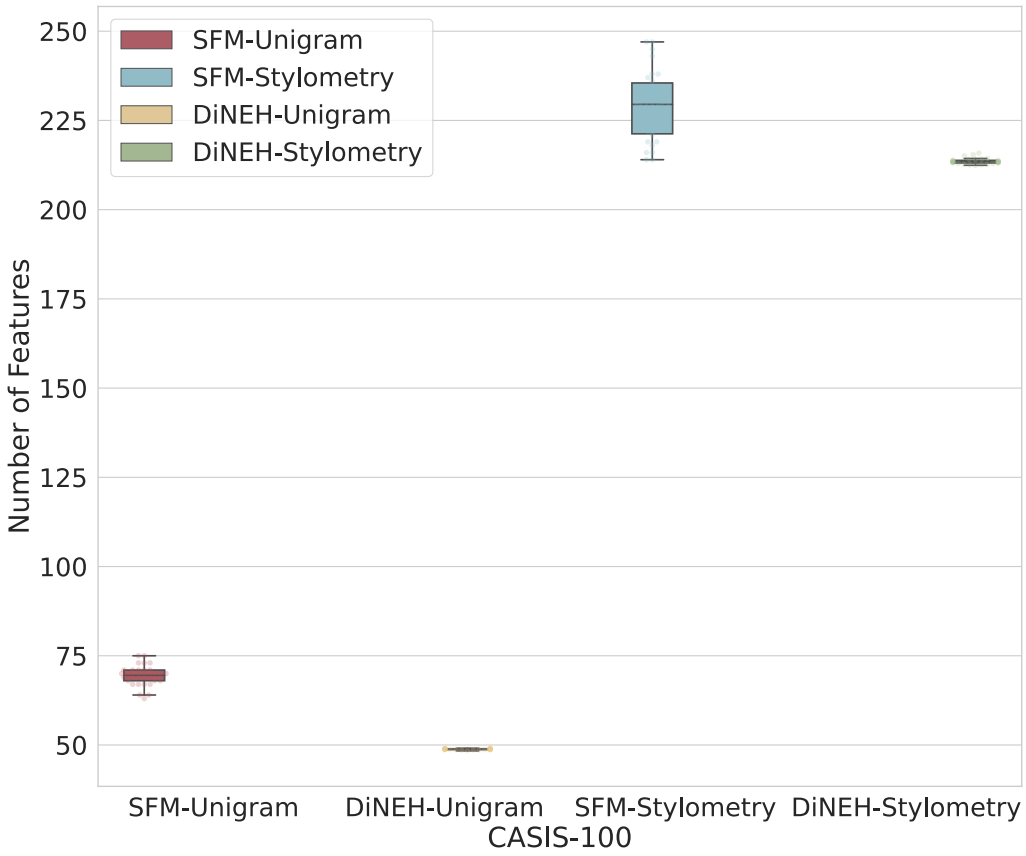

Figure 4.4: Average number of features used in the Single GEFeS approach and DiNEH using the Character Unigram and Stylometry feature sets.

Table 4.6: Performance Comparison baselines with DiNEH. The table lists average classification accuracy of 30 runs for each algorithm.

Algorithm	CASIS-25	CASIS-50	CASIS-100	CASIS-1000	C50
Koppel [32]	87.00%	74.00%	69.25%	51.55%	59.72%
Teahan [62]	89.00%	78.00%	65.75%	55.15%	**69.16%**
Keselj [25]	**99.00%**	93.00%	87.00%	66.05%	54.94%
Stamatatos [59]	10.00%	3.01%	3.00%	1.78%	18.18%
Benedetto [6]	75.00%	65.00%	40.75%	28.00%	60.84%
DiNEH$_{uni}$	92.10%	80.90%	77.45%	62.43%	44.28%
DiNEH$_{sty}$	98.10%	94.80%	87.85%	79.40%	48.88%
DiNEH$_{BoW}$	**99.00%**	**98.50%**	**95.75%**	**96.22%**	64.84%

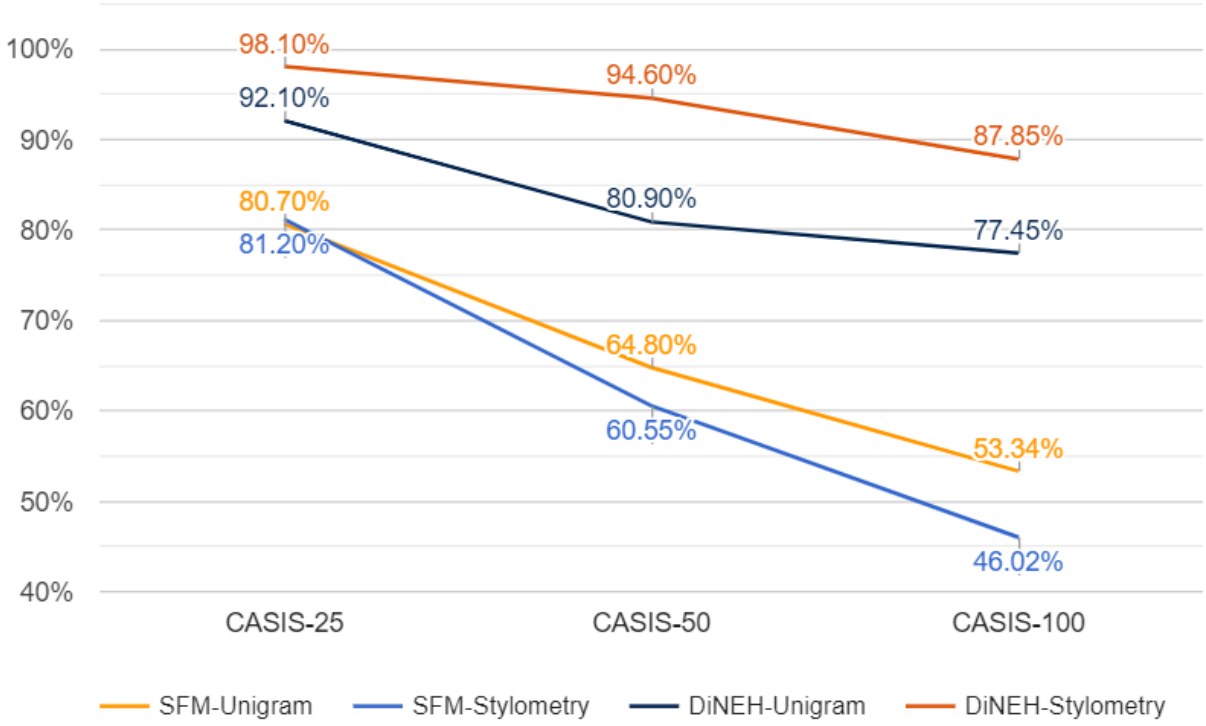

Figure 4.5: Average accuracy of the Single GEFeS approach and DiNEH using the Character Unigram and Stylometry feature sets.

4.6 Summary

In this chapter, we introduced a distributed neuro-evolutionary hybrid for Author Identification which was referred to as DiNEH. Our results show that DiNEH scales well to datasets with larger numbers of authors by evolving a distributed feature mask. DiNEH achieves that by distributing authors to individual machines allowing it to scale horizontally. We compared the DiNEH with five well-known AISs, and DiNEH had the best performance on all but two datasets (CASIS-25 and C50) both of which had 50 or fewer authors.

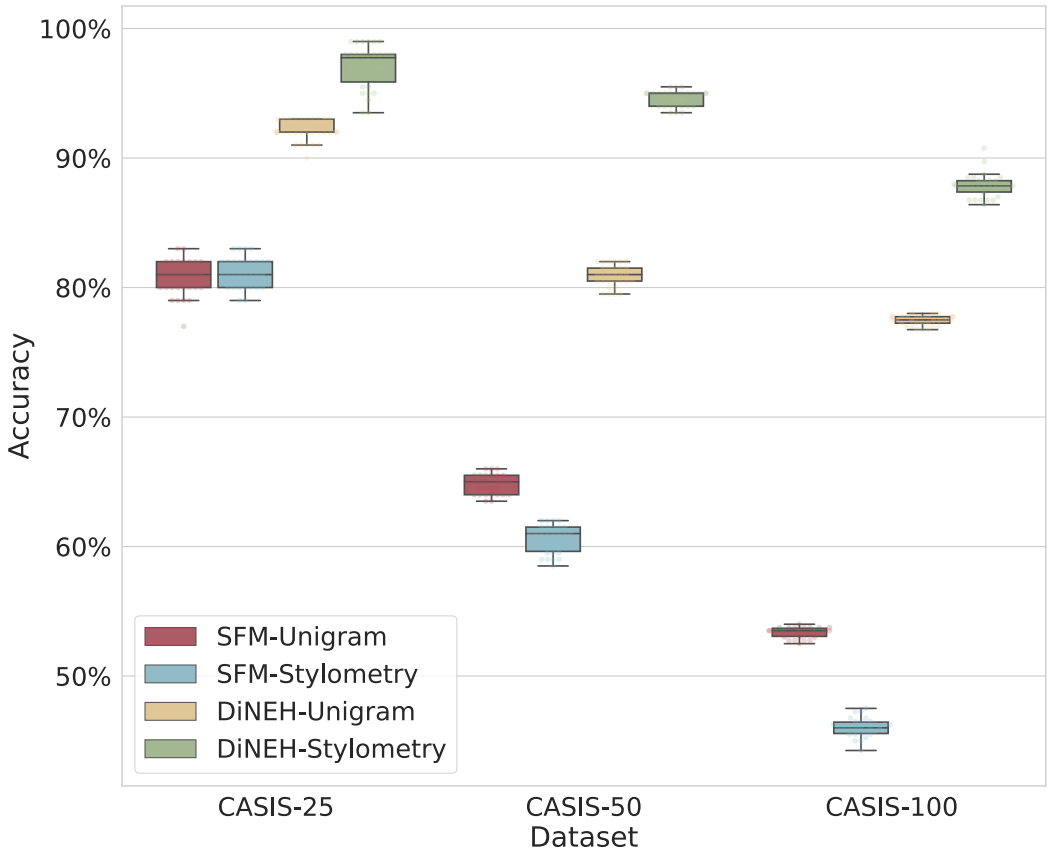

Figure 4.6: Average accuracy of the Single GEFeS approach and DiNEH using the Character Unigram and Stylometry feature sets.

Chapter 5

Author Identification of Micro-Messages via Multi-Channel Convolutional Neural Networks

In this chapter, the Author Identification of micro-messages problem via Convolutional Neural Networks (CNNs) is studied. Specifically, we introduce a novel Multi-Channel CNN architecture that processes different features of text via word and character embedding layers and utilizes both pre-trained word embedding and character bigram embeddings. We examine the usefulness of different feature types and show that the combination of embedding layers can capture different stylometric features. We conduct extensive experiments with a varying number of authors and writing samples per author. Our results show that our proposed method outperforms a state-of-the-art system on a Twitter dataset that contains 1,000 authors.

5.1 Introduction

Currently, the performance of existing Author Identification systems of micro-messages [56] is not any better than a chance of flipping a coin even with a small number of authors [57]. The recent development of CNNs has shown promising results on image classification tasks [16,33]. One of the difficulties of using these types of CNNs for Author Identification is basically a representational difference [9]. The tokens of the text (character, word, etc.) are discrete and not convertible to values that machines can understand [9]. Studies [22, 41] have shown that using a number of word embedding techniques can convert words into individual vectors. Some of these techniques include Word2Vec [41], FastText [22], etc. In the literature [4,12,14,21,56], tokens other than words could also be used as features in machine learning algorithms for Author Identification. These tokens include character *n*-grams [25,52,55], parts of speech [52], and stylometry [42].

Social media platforms such as Twitter [56] limit the number of characters for each post. Then, we can state the following research question: How can we identify an author of a single post message with 140 characters or less (as limited in [56])? Identifying authors of micro-messages has shown to be more difficult than Author Identification using long text [36] because of the inherit characteristics of micro-messages:

- Each micro-message itself is very short to properly extract user trails and traditional stylometric patterns.
- Micro-messages may lack the syntax of natural languages and contexts themselves.
- Micro-messages may contain more typos and syntax errors than book chapters or documents.

Our goal is to design a CNN architecture for identifying the authors of micro-messages to overcome the problems mentioned before. This architecture can assist social networks for auto-moderation to prevent undesired behavior like spreading fake news. In the literature [52], it is shown that different feature sets can capture a variety of stylometric features. Considering this information, we combine word embeddings with character embeddings under a single CNN architecture to form a Multi-Channel CNN architecture. These embedding layers can be considered as individual information channels for a CNN.

We compare our proposed architecture with the best performing state-of-art methods within the literature [57] as well as several popular authorship attribution methods [56]. We show that using Multi-Channel CNNs with pre-trained word embeddings perform the best.

5.2 Related Work

In this section, we present some related work to our research.

5.2.1 Author Identification of Micro-Messages

In the literature [43], a variety of methods have been developed and used for micro-message Author Identification problems. Schwartz et al. [56] introduces a method known as *k*-Signatures

to identify authors of micro-messages. The k-signatures method captures the style of an author by collecting the features only found in the writing samples of the author. Also, another condition for collecting the feature is that a given feature needs to be seen in $k\%$ of the author's writing samples. Finally, captured features are used by the classification algorithms for Author Identification.

Rocha et al. [52] presents an overview of Authorship Identification methods used in social media. In addition, they provide performance analysis of the feature sets of micro-messages using two classifiers, namely, a Power Mean SVM [65] and Random Forests. They suggest that using character 4-grams as feature set in their classifiers shows the best performance in Author Identification.

Shrestha et al. [57] uses a different approach for identifying the authors of micro-messages. They use a sequence of character n-grams as input. To date, this is the best performing algorithm for Author Identification of micro-messages.

5.2.2 Neural Networks for Author Identification

CNNs for Author Identification have shown promising results [28, 57]. Kim [28] proposed a CNN for sentence classification. The CNN utilized convolutional filter sizes of 3, 4, and 5, a dropout rate, and max-over-time pooling (Collobert et al. [9]). Kim's approach [28] uses a different combination of models such as: a) random model, b) static model, c) dynamic model, and d) multi-channel model that combines the static and dynamic models. The models, excluding the random model, used pre-trained word2vec word embeddings [41].

Recurrent Neural Networks (RNNs) [3], have also been used for Author Identification. Bagnall [3] uses an RNN that predicts the next character in the sequence based on the previously seen characters. Different sets of probabilities of the next character for each author are generated, then authors are identified based on these probabilities. This was the best performing method for the PAN 2015 multi-language author identification competition.

5.2.3 Multi-Channel Convolutional Neural Networks

Multi-Channel CNNs [15] have been extensively studied with applications in image classification, object detection, and speech recognization [19]. The color channels (such as RGB) in computer vision [15] or the wavelength channels in speech recognization [19] have been proven successful as Multi-Channel input for classification problems. Although natural language inputs are normally in the form of single-channel tokens or characters, the different sets of extracted features are shown to capture different stylometric features [25] [55] [12] [52]. To our knowledge, no previous work has been done on the micro-message Author Identification task using feature learning and model training from both word and character *n*-grams embeddings channels.

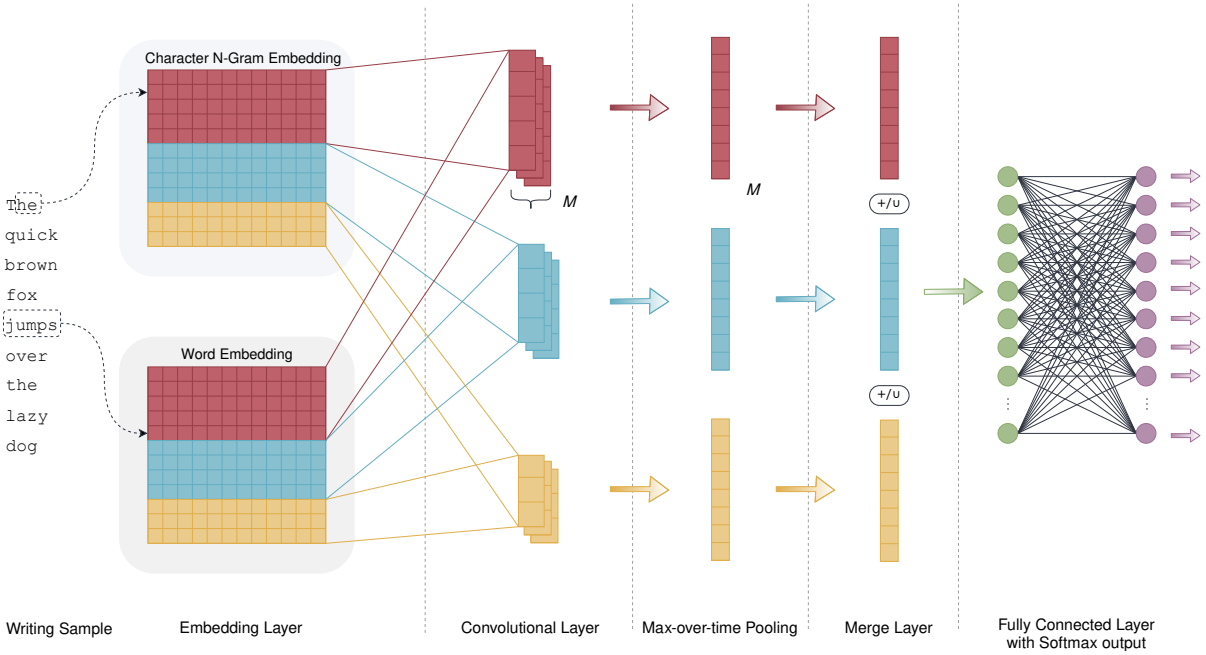

Figure 5.1: Multi-Channel CNN architecture diagram. The writing sample is "The quick brown fox jumps over the lazy dog". The writing samples are tokenized and fed to embedding layers. Word embedding and character bigram embedding layers are illustrated in the figure. The red, blue, and yellow colors indicate the different window sizes for each channel. The embeddings are then forwarded to convolutional layers with different window sizes. There are M numbers of filters for each window size. Max-over-time pooling is used in the the pooling layer, followed by the merge layer, which combines the features by either Concatenation$(\cup)$ or Add$(+)$, then fed into the fully connected layer with asoftmax output.

5.3 Multi-Channel Convolutional Neural Networks

5.3.1 Model Construction

In light of the above discussion, we devised a Multi-Channel CNN architecture utilizing both the word embeddings and character *n*-gram embeddings. The Multi-Channel convolutional network architecture is shown in Figure 5.1, and the model details are defined in the following subsections.

5.3.1.1 Embedding Layer

Figure 5.1 shows two feature sets used as individual input channels with their own embedding layers. These feature embeddings are padded to have the same size. The dropout [58] is applied to the embedding layer to avoid over-fitting. The feature embeddings (inputs) are then given to the convolutional layers with different window sizes in parallel.

5.3.1.2 Convolutional Layer

The convolutional layers apply filters to the input, feature embeddings in our case, and shifts the filter until the end of the sequence. The window size of the convolution operation corresponds to the size of the filter that is applied to the input each time. Following the work of [9], we used convolutional filters with window sizes of 3, 4, and 5. We used the Scaled Exponential Linear Unit (SELU) activation function [30] after each convolutional layer for self-normalizing properties. Also, the SELU activation function introduces non-linearity to the model. Hence, the model can construct non-linear mappings between the input and outputs of each layer.

The convolutional layer can be represented as follows:

$$selu(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases} \tag{5.1}$$

where x is the output of the convolutional filters, and λ and α are pre-defined constants.

The outputs of the convolutional filters are known as feature maps. Since we use three different filter size with $M = 500$ filters each total number of filters is $3 \times M = 1,500$. The same filters are applied for character embeddings and word embeddings input channels.

5.3.1.3 Pooling Layer

Convolutional layers are followed by max-over-time pooling [9], which takes the feature with maximum value in a given filter.

For the feature matrix output x by the convolutional layer, the max-over-time pooling operation can output $\hat{x}$ as follows:

$$\left[\hat{x}\right]_i = \max_t \left[x\right]_{i,t} \quad i = 1, \ldots, M \tag{5.2}$$

where t is the position in the sentence, i is the convolutional filter applied and M is the numbers of filters for each window sizes (Figure 5.1).

In this way, we make the network size the most helpful features produced by the convolutional layers. Thus, the number of selected values with max-over-time pooling is equal to the number of filters in the architecture per feature embedding.

5.3.1.4 Merge Layer

The resulting features are merged into a single feature map. To merge the feature maps of the character and word embeddings, we experimented with both concatenation ($\cup$) and add (+) operations in the merge layer. The differences between the two operations are that the concatenation operation concatenates the features into a single feature vector, while the add operation uses pairwise add operation between feature vectors. Here are the detailed explanation of the both operations:

- Concatenation operation: The features are concatenated into a single feature map. For given vector $a_{1..n}$ and $b_{1..m}$ contaction operation can be represented as follows:

$$concat(a, b) = [a_1, a_2, ..., a_n, b_1, b_2, ..., b_m] \tag{5.3}$$

- Add operation: Element wise add operation between feature vectors. For given vector $a_{1..n}$ and $b_{1..n}$ add operation can be represented as follows:

$$add(a, b) = [a_1 + b_1, a_2 + b_2, ..., a_n + b_n] \tag{5.4}$$

5.3.1.5 Fully Connected Layer with Softmax Function

After the merge layer, the last step is to feed the feature maps into the fully connected layer, followed by the softmax function for the classification.

The softmax function, which is shown in equation 5.5, take exponents of each element x_i from the input feature map x and normalize them by dividing by the sum of all the exponentials. The equation is as follows:

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{5.5}$$

where k is the total number of inputs. Softmax function scores the authors based on the closeness of the feature map to the class of an author. The network guesses the author of the writing samples with the highest score.

5.3.2 Hyper-parameter Tuning

The hyper-parameters were selected based on grid search [7]. The embedding dimension size of character and word used was 400. We used a dropout layer with a drop rate of 25%. For the convolutional layer, there were 500 filters per window size, $M = 500$, which makes a total of 1,500 filters. Hence, only one feature was selected per filter. We used a batch size of 64 for the experiment. Since wasn't any improvement observed after epoch 100, the epoch limit was set to 100. Also, to speedup the training process an early stopping condition was implemented to stop the training after 20 epochs without improvement. The learning rate updated during the training by the Adam optimizer [29]. The initial learning rate was $1e - 4$.

5.4 Experiments

In this section, we present our experiments. First, we investigate the impacts of different features for embedding layers. Then, we study the performance of our proposed architecture with increased difficulties, such as increasing the number of authors or decreasing the number of writing samples. The experiments are conducted in two different settings. We analyze the relation of the number of authors and the number of writing samples. The parameters are the number of authors and the number of writing samples, one of the parameters is kept static while the other one is tested on a range of values. Lastly, we conduct an experiment to compare different operations of merging layers. The preprocessing method for the dataset is used, which is similar to [56], in all four experiments. The steps can be summarized as follows. We replace the numbers, username references, date, time, and website URLs with pre-defined meta tags.

Our four experimental settings are explained in the following subsections.

5.4.1 Experiment I: Varying Character *n*-gram Embeddings

To assess the effectiveness of different features for the embedding layer, we compare the *n*-gram embedding performance and the number of epochs on the same dataset set. In this experiment, the individual performance of *n*-gram embeddings with different n values is evaluated. We test 1, 2, 3, 4, and 5 for *n* values based on their usage in the literature [52, 56]. For this purpose, we create a development dataset, also known as a validation set, with 10 authors and 200 writing samples per author. The authors in the development set are not used in any other experiment. We collect the performance of the CNNs using 100 and 1,000 epochs. Since the number of unique *n*-grams increases with higher values of n, we increase the number of epochs 1,000 to avoid under-fitting.

5.4.2 Experiment II: Varying Number of Authors

We explore how our proposed architecture performs with an increasing number of authors. We perform a set of performance evaluations with a different number of authors while keeping the number of writing samples static. The number of writing samples per author (w) is 200. The

number of authors (a) used in the experiments is 100, 200, 500, and 1,000. In this experiment, we use the same sampling used in [56]. As expected, the results are similar to the reported results in [57] and [56]. We also use 10-fold cross-validation on the experiments for each author group. Hence, we evaluate 40 train-test splits in total.

5.4.3 Experiment III: Varying Number of Writing Samples

In this experiment, we investigated the impact of the number of writing samples on author identification. We perform a set of performance evaluations with a varying number of writing samples per author, where the number of authors is 50. The number of writing samples per author used in the experiments was 50, 100, 200, and 500. We sample 10 different disjoint sets of groups for each writing sample size[1]. We use 10-fold cross-validation on the experiments for each author group. Hence, we evaluated 400 experiments in total.

5.4.4 Experiment IV: Impacts of Feature Map Merging Methods

In our final experiment, we study the impacts of different methods used in the merging layers. There are two popular ways to merge the feature maps, namely add and concatenate operations. We explore the impacts of the two methods under the same experimental setup for both varying number of authors and a varying number of writing samples where ($\cup$) stand for the concatenation operation and (+) stands for the add operations. The rest of the experimental setup is the same as described in section 5.4.2 and section 5.4.3.

5.5 Results

5.5.1 Baselines

We compared our proposed methods against the following baselines:

- *k*-signatures [56]: Uses features that include character and word *n*-grams. The features used by a single author at least *k%* of the documents are selected as k-signatures. Also, a

[1]We contacted the authors of the previous work but unfortunately, they told us that they do not have the sampling anymore.

method called Flexible Patterns was utilized. Patterns in this method can match partially and get a score based on closeness. Since a combination of *k*-signatures and Flexible Patterns techniques yielded better results, the reported results are a combination of the techniques.

- LSTM$_{Char2}$: Long Short- Term Memory (LSTM) [18] networks are widely used and known for sequence-to-sequence tasks. There are also applications for Author Identification. We used character bigrams as an input to the LSTM network based on our preliminary findings on our development set.

- CNN$_{CharN}$ [57]: All character *n*-grams are randomly initialized and then updated during training. The other parameters of the networks such as filter weights are also updated with Backpropagation. Here, we implement both the CNN$_{Char1}$ and CNN$_{Char2}$ methods to compare with our proposed method.

- CNN$_{W2VStatic}$ [28]: All words are initialized with a pre-trained word vector from Word2Vec. During the training, word embeddings are kept static. The rest of the network parameters, convolutional filters and fully connected layer weights, are updated.

- CNN$_{W2V}$ [28]: Pre-trained vectors in CNN$_{W2VStatic}$ are updated during the training. The pre-trained embeddings are trained on the Twitter dataset.

- CNN$_{FastText}$: All words are initialized with a pre-trained word vector trained with FastText. The network configuration is identical with CNN$_{W2V}$. The pre-trained embeddings are trained on the Twitter dataset.

We implemented an ablation study of our proposed architecture where different word embedding and character *n*-gram embedding methods are treated as individual input channels. The embeddings in each channel are updated during the training phase. The individual channels are merged with Concatenate ($\cup$) or Add (+) layers. The implementation information and detailed explanation of these settings are listed as follows:

- CNN$_{W2V\ (+/\cup)\ Char1}$: The word embeddings and character unigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from Word2Vec.

- CNN$_{W2V\ (+/\cup)\ Char2}$: The word embeddings and character bigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from Word2Vec.

- CNN$_{FastText\ (+/\cup)\ Char1}$: The word embeddings and character unigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from FastText.

- CNN$_{FastText\ (+/\cup)\ Char2}$: The word embeddings and character bigram embeddings are treated as an individual input channel. All words are initialized with a pre-trained word vector from FastText.

5.5.2 Results of Experiment I: Varying Character *n*-gram Embeddings

Figure 5.2 shows the performances of character *n*-grams on the development set. Figures 5.2 and 5.3 were trained for 100 and 1,000 epochs, respectively. Since no improvement observed after 1,000 epochs for all *n*-grams we stopped the training at 1,000 epochs. For more practical and efficient results we also provided the performance on 100 epochs. The figures illustrate the accuracy of 10-fold crossover, the mean values are indicated by solid black lines, and the individual precision values of each author are shown as dots with circles.

We performed an ANOVA and a Student's t-test with p=0.05 on performance of character *n*-gram embeddings to compare them. In Figure 5.2, the mean accuracies of 1-5 grams were 66.15%, 66.00%, 57.25%, 52.50%, and 49.65%, respectively. Based on the statistical tests, we observed that there was no significant difference between 1-gram (unigram) and 2-gram (bigram) embeddings. The performance of the *n*-grams when they are trained for 100 epochs are as follows, $1-gram = 2-gram > 3-gram > 4-gram = 5-gram$. The mean accuracies of 1-5 grams in Figure 5.3 are 78.45%, 75.70%, 72.40%, 65.35%, and 55.45%, respectively. Based on the statistical tests, the performance of the *n*-grams when they are trained

for 1,000 epochs as follows, $1-gram > 2-gram > 3-gram > 4-gram > 5-gram$. The performance gap becomes larger between character *n*-grams after increasing the number of epochs.

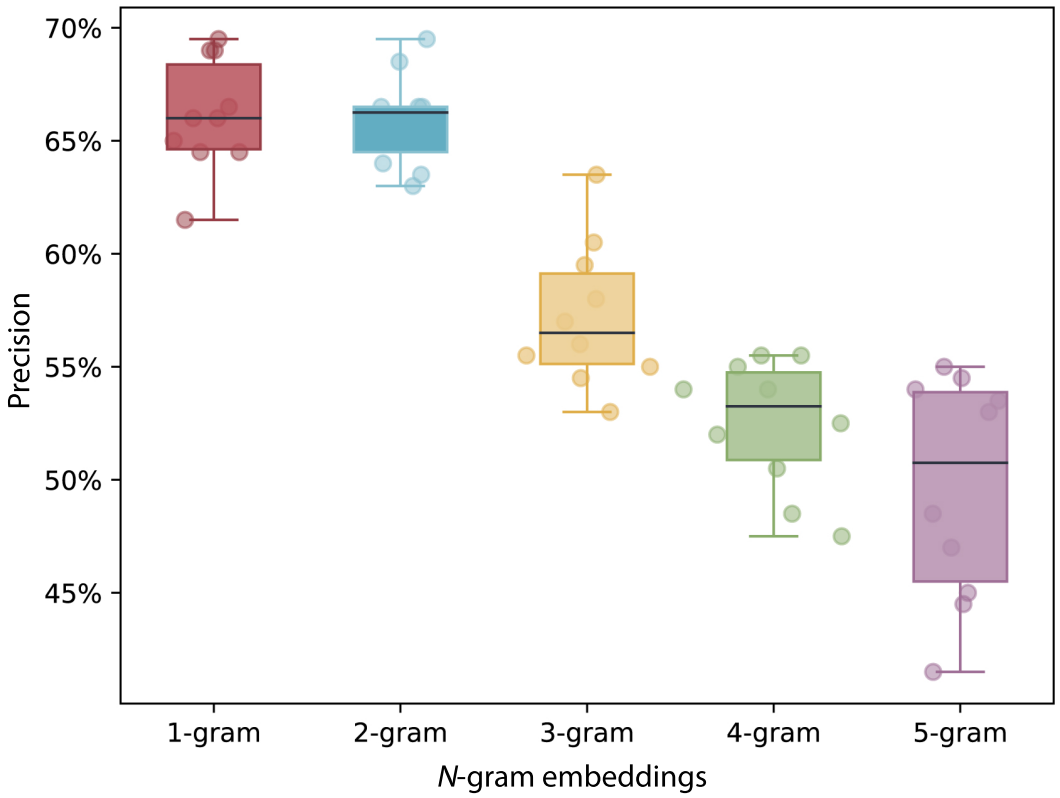

Figure 5.2: The box plot that illustrates the performances of character *n*-grams trained for 100 epochs on the development set. Precision is shown in the *y* axis. Five boxplots display the results of the 1-5 gram methods, respectively. In each box, the square box indicates the interquartile range from 25% to 75%; the black line inside the box represents the median value. The individual precision values of authors from each method are shown as small dots with circles in the same color as the boxplot.

5.5.3 Results of Experiment II: Varying Number of Authors

Table 5.1 shows the results of the approaches discussed in Section 5.4 with varying the number of authors. In Table 5.1, the first column lists the algorithm used in the experiment, where the baseline methods are listed above the double lines, and our proposed methods are listed below the double lines. The rest of the columns list the accuracies of the algorithms with 100, 200,

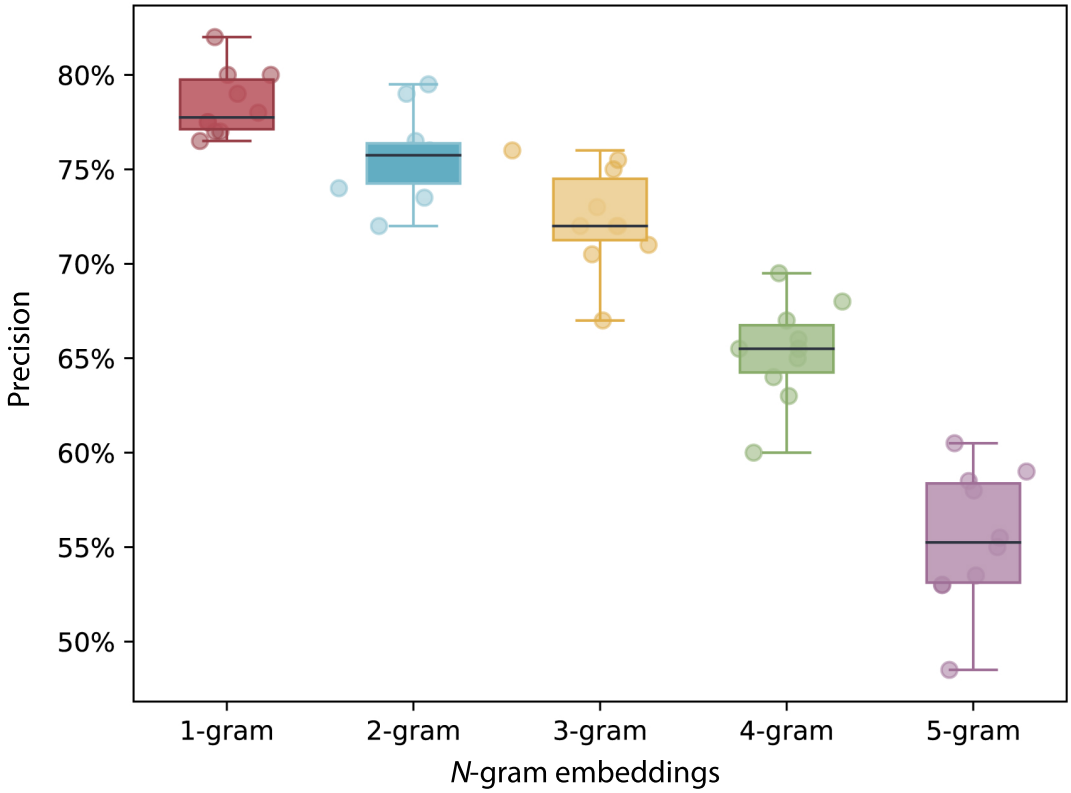

Figure 5.3: The box plot that illustrates the performance of character *n*-grams trained for 1,000 epochs on the development set. Five boxplots display the results of the 1-5 gram methods, respectively. In each box, the square box indicates the interquartile range from 25% to 75%; the black line inside the box represents the median value. The individual precision values of authors from each method are shown as small dots with circles in the same color as the boxplot.

500, and 1000 authors, respectively. The number in bold marks the highest accuracy in each column.

As expected, the accuracy was decreased as the number of authors was increased for all the methods listed. Among the seven baseline methods, $\text{CNN}_{FastText}$ performed surprisingly better than the others without needing character *n*-grams. While Word2Vec struggling with Out-of-vocabulary (OOV) words, FastText can obtain vectors for unseen words by summing up the vectors for character n-grams. This property of the FastText allows it to perform better compared to the other single-channel baseline CNNs. Our tests confirmed that using character *n*-grams performs better than the compared algorithms. CNN_{W2V}, which does not use character *n*-gram, performed worse than the CNN_{Char2} on all cases. We were unable to reproduce the

behavior of CNN$_{Char1}$ and CNN$_{Char2}$ reported in [57]. The reported results show that Character Unigrams perform better on 100 authors compared to Character Bigrams, with a small margin. In our tests, Character Bigrams performed better on all cases.

Comparing with CNN$_{W2V}$, the Multi-Channel architecture methods performed better with adding the character *n*-grams channel. The best performer in terms of the word embeddings pre-trained with Word2Vec was CNN$_{W2V+Char2}$, which performed better than the state-of-the-art results. The accuracies of the CNN$_{W2V+Char2}$ algorithms with 100, 200, 500, and 1000 authors are 52.67%, 50.53%, 43.79%, and 38.29%, respectively.

Similarly with Word2Vec, CNN$_{FastText+Char2}$ was the best performer for all authors' sets in Experiment II with FastText embeddings and the overall. The accuracies of the CNN$_{FastText+Char2}$ algorithms with 100, 200, 500, and 1000 authors were 55.20%, 53.14%, 46.90%, and 41.28%, respectively. On average, CNN$_{FastText+Char2}$ performed 10% better than the state-of-the-art results with 100, 200, 500, and 1000 authors.

The ANOVA results showed that CNN$_{FastText+Char2}$ significantly outperformed the best state-of-art algorithm in all numbers of writing samples.

CNN$_{FastText+Char2}$ was compared to all the best of the state-of-the-art algorithms using a Multivariate Analysis of Variance (MANOVA) with a Tukey post hoc comparison test. Table 5.2 presents the Multivariate Analysis of Variance (MANOVA) results for the mean difference between CNN$_{FastText+Char2}$ and CNN$_{FastTest}$, which was the best of the state-of-the-art algorithms. In Table 5.2, the confidence intervals were adjusted by Tukey's honestly significant difference test. The MANOVA results showed that CNN$_{FastText+Char2}$ significantly outperformed the best state-of-art algorithm in all numbers of writing samples.

5.5.4 Results of Experiment III: Varying Number of Writing Samples

Table 5.3 shows the results of the methods discussed in Section 5.4 for 50, 100, 200, and 500 writing samples per author. In Table 5.3, the first column lists the algorithm used in the experiment, where the baseline methods are listed above the double lines, and the proposed methods in this dissertation are listed below the double lines. The rest of the columns list the

Table 5.1: Performance of the algorithms with varying number of authors.

Algorithms	Authors			
	100	200	500	1000
CNN_{Char1} [57]	49.24%	47.68%	41.37%	35.60%
CNN_{Char2} [57]	49.96%	48.84%	42.92%	37.55%
k-signatures [56]	42.50%	41.10%	35.50%	30.30%
LSTM_{Char2}	33.80%	33.50%	29.80%	24.80%
$\text{CNN}_{W2VStatic}$	24.10%	20.80%	16.10%	12.70%
CNN_{W2V}	47.21%	45.52%	39.85%	34.73%
$\text{CNN}_{FastText}$	51.83%	50.25%	44.18%	38.74%
$\text{CNN}_{W2V \cup Char1}$	50.42%	48.12%	42.44%	37.48%
$\text{CNN}_{W2V + Char1}$	52.44%	49.74%	43.53%	37.71%
$\text{CNN}_{W2V \cup Char2}$	48.95%	47.06%	41.76%	36.57%
$\text{CNN}_{W2V + Char2}$	52.67%	50.53%	43.79%	38.29%
$\text{CNN}_{FastText \cup Char1}$	52.62%	51.36%	46.17%	40.61%
$\text{CNN}_{FastText + Char1}$	52.55%	50.89%	45.42%	40.25%
$\text{CNN}_{FastText \cup Char2}$	54.23%	52.23%	46.62%	40.84%
$\text{CNN}_{FastText + Char2}$	**55.20%**	**53.14%**	**46.90%**	**41.28%**

Table 5.2: Comparison of $\text{CNN}_{FastText + Char2}$ to $\text{CNN}_{FastTest}$ for the different number of writing samples.

Author	Mean Difference	Std. Error	F	Sig.	95% CI for Difference	
					Lower Bound	Upper Bound
100	0.034	0.0040	57.17	≤ 0.001	0.0243	0.0430
200	0.029	0.0030	68.67	≤ 0.001	0.0216	0.0363
500	0.027	0.0020	140.92	≤ 0.001	0.0224	0.0320
1000	0.025	0.0008	926.68	≤ 0.001	0.0236	0.0271

accuracy of the algorithms with 50, 100, 200, and 500 writing samples per author, respectively. The results marked in bold are the highest accuracy in each column.

As seen in Table 5.3, the accuracy values increase with the increasing number of writing samples for all methods. Among the baseline methods, CNN$_{FastText}$ performed surprisingly better than the others without needing character *n*-grams. The reported results in [57] showed that Character Bigrams performed better only on 500 writing samples compared to Character Unigrams. However, in our tests, Character Bigrams perform better on 50 and 100 writing samples, which suggests that character unigrams performed better with larger writing samples.

Our tests verified that using character *n*-grams perform better than the benchmark algorithms. CNN$_{W2V}$ without the use of character *n*-gram performed worse than the CNN$_{Char2}$ for all cases. After adding character *n*-grams channel, CNN$_{W2V+Char2}$ performed better than the CNN$_{W2V}$. The best performer in terms of the word embeddings pre-trained with Word2Vec is CNN$_{W2V+Char2}$, which performed better than the state-of-the-art algorithms. The accuracies of the CNN$_{W2V+Char2}$ algorithms with 50, 100, 200, and 500 writing samples are 54.91%, 61.52%, 67.11%,and 72.88%, respectively. From Table 5.3, we can see that CNN$_{W2V+Char2}$ performs the best among all methods with 50 writing samples.

Furthermore, we observed the same behavior with character *n*-grams, which improved the accuracy in every combination. Similarly with Word2Vec, CNN$_{FastText+Char2}$ was the best performer in all experiments with FastText embeddings. On average, CNN$_{FastText+Char2}$ performed 5% better than the state-of-the-art results with 50, 100, 200, and 500 writing samples per author. The accuracies of the CNN$_{FastText+Char2}$ algorithms with 50, 100, 200, and 500 writing samples were 54.36%, 62.17%, 68.34%, and 74.50%, respectively. The improvement in accuracy was less than the improvements observed in the experiment with varying the number of authors.

5.5.5 Result of Experiment IV: Impacts of Feature Map Merging Methods

Table 5.1 shows the comparison of different merging methods with varying the number of authors. The add operation performed better in all experiments except one (Table 5.1, CNN$_{FastText \cup Char1}$) with small margins. The accuracy of the CNN$_{FastText \cup Char1}$ algorithm with 100, 200, 500, and 1000 authors is 52.62%, 51.36%, 46.17%, and 40.61%, whereas the accuracy of CNN$_{FastText+Char1}$

algorithm with 100, 200, 500, and 1000 authors is 52.55%, 50.89%, 45.42%, and 40.25%, respectively.

Similarly, Table 5.3 shows the comparison of different merging methods with varying the number of writing samples. The add operation performed better in all experiments with varying the number of writing samples except one (Table 5.3, CNN$_{FastText \cup Char1}$ with small margins. The accuracy of CNN$_{FastText \cup Char1}$ algorithm with 500 writing samples is 72.96%, whereas the accuracy of CNN$_{FastText + Char1}$ algorithm with 500 writing samples is 72.89%.

From both Tables 5.1 and 5.3, we can see that the best performer in each experiment uses the add operation in the merge layer. In a general case for authorship attribution on micro-messages, the recommended merge operation is the add operation instead of the concatenation operation for our proposed Multi-Channel CNN architecture based on these experimental results.

Table 5.3: Performance of the algorithms with varying number of writing samples.

Algorithms	Writing Samples			
	50	100	200	500
CNN$_{Char1}$ [57]	51.40%	58.20%	64.07%	70.30%
CNN$_{Char2}$ [57]	51.56%	58.25%	63.59%	69.80%
CNN$_{W2VStatic}$	36.60%	41.70%	46.00%	50.90%
CNN$_{W2V}$	49.14%	56.68%	62.96%	69.70%
CNN$_{FastText}$	51.46%	59.14%	65.61%	72.46%
CNN$_{W2V \cup Char1}$	52.86%	60.10%	65.85%	71.89%
CNN$_{W2V + Char1}$	53.71%	60.55%	66.12%	72.30%
CNN$_{W2V \cup Char2}$	52.68%	60.12%	65.87%	71.86%
CNN$_{W2V + Char2}$	**54.91%**	61.52%	67.11%	72.88%
CNN$_{FastText \cup Char1}$	51.94%	59.83%	66.14%	72.96%
CNN$_{FastText + Char1}$	53.28%	60.56%	66.64%	72.89%
CNN$_{FastText \cup Char2}$	52.04%	60.58%	67.28%	73.87%
CNN$_{FastText + Char2}$	54.36%	**62.17%**	**68.34%**	**74.50%**

CNN$_{FastText+Char2}$ was compared to all the best of the state-of-the-art algorithms using a Multivariate Analysis of Variance (MANOVA) with a Tukey Post Hoc comparison test. Table 5.4 presents the MANOVA results for the mean difference between CNN$_{FastText+Char2}$ and CNN$_{FastTest}$, which was the best of the state-of-the-art algorithms. In Table 5.4, the confidence intervals were adjusted by Tukey's honestly significant difference test. The MANOVA results showed that although CNN$_{FastText+Char2}$ had higher mean accuracies, the differences between these two methods were not statistically significant.

Table 5.4: Comparison of CNN$_{FastText+Char2}$ to CNN$_{FastTest}$ for the different number of writing samples.

					95% CI for Difference	
W	Mean Difference	Std. Error	F	Sig.	Lower Bound	Upper Bound
50	0.029	0.0225	1.304	0.699	-0.03487	0.0928
100	0.030	0.0207	1.943	0.590	-0.02842	0.0889
200	0.027	0.0190	2.567	0.606	-0.02662	0.0813
500	0.020	0.0174	2.879	0.767	-0.02901	0.0698

5.6 Summary

In this chapter, we presented a Multi-Channel CNN approach that can be applied to identify the authors of micro-messages. We compared the performance of state-of-the-art CNNs that use character *n*-gram embeddings with the proposed CNNs that use a Multi-Channel architecture. The computational experiments showed that using Multi-Channel CNNs with pre-trained word embeddings performed better compared to the CNNs that use character *n*-gram embeddings on the Twitter dataset. Also, we showed that using the Add operation instead of the Concatenation operation in the merge layer increased the performance of the system on all of the cases but one ($CNN_{FastText \cup Char1}$).

Chapter 6

AARef: Exploiting Authorship Identifiers of Micro-Messages with Refinement Blocks

With the rise of web-based social networking, a great many short texts/micro-messages are exchanged daily. Although short texts/micro-messages are a powerful and efficient way to communicate among individuals, their anonymity and short-length attributes give rise to a real challenge for Author Identification studies. Here, we handle the Author Identification of short texts problem via Convolutional Neural Networks (CNNs). Specifically, we present a novel Multi-Channel CNN architecture that processes different features of text via word, character, and parts of speech (POS) embeddings. We examine the usefulness of different feature types and show that the combination of embeddings can capture different stylometric features. In addition, we add an identity mapping block in the convolutional layer to preserve the largest amount of information from features. Extensive experiments with a variety number of authors and writing samples per author were conducted using our proposed architecture. Based on the experiments, our proposed method outperforms the state-of-the-art system on a large Twitter dataset.

6.1 Introduction

Author Identification [60], also referred to as Authorship Attribution, is the task of capturing the stylistic information in a collection of writing samples and identifying the authors of unknown texts based on the captured information [43]. Author Identification research has substantially developed for the last decade with a focus on long texts [52, 63]. Recently, there is a growing interest in identifying authors of micro-messages due to the development of social media platforms and the emergence of social media as the primary mode of communication [52]. The

increase in micro-message traffic has attracted attention in many fields such as email author identification [44], blog author identification [48], and web message author identification [45].

The task of identifying authors of micro-messages has been shown to be more difficult than Author Identification using long texts [36]. Social media platforms, such as Twitter [56], limit the number of characters for each tweet (micro-message). Furthermore, many tweets are not as formal as book chapters or documents, and may lack the syntax of natural languages and contexts and may contain many typos and syntax errors. Here, we attempt to tackle these problems via a novel Multi-Channel CNN architecture.

The stylometric properties of a text can be captured by different feature sets [52]. Therefore we combine word embeddings, character embeddings, and part of speech embeddings within one CNN architecture to form a Multi-Channel CNN architecture, namely AARef. These embeddings can be considered as individual information channels for a CNN. We also add a novel multi-branch convolution block with identity mapping in the convolutional layer. The identity mapping links the layers in the network and provides an alternative path to preserve the gradient. This skip connection ensures that the initial information can be captured and passed to the next layer. We compare AARef with the best performing state-of-art methods within the literature [2] as well as several popular authorship attribution methods for micro-messages [56, 63]. Our preliminary results of AARef using shared convolutional filters and skip connections for all embeddings outperform the state-of-the-art methods [2].

We have designed a number of experiments with a varying number of authors (from 100 to 1,000) as well as a varying number of writing samples (from 50 to 500) per author. Each tweet was treated as one writing sample in the experiments. We showed that each tweet could be successfully identified using our proposed Multi-Channel CNN architecture. Comparing with the state-of-art baselines, our model achieved a minimum of 3.81% improvement in classification accuracy on the Twitter dataset.

6.2 Method

In this section, we present the details of AARef.

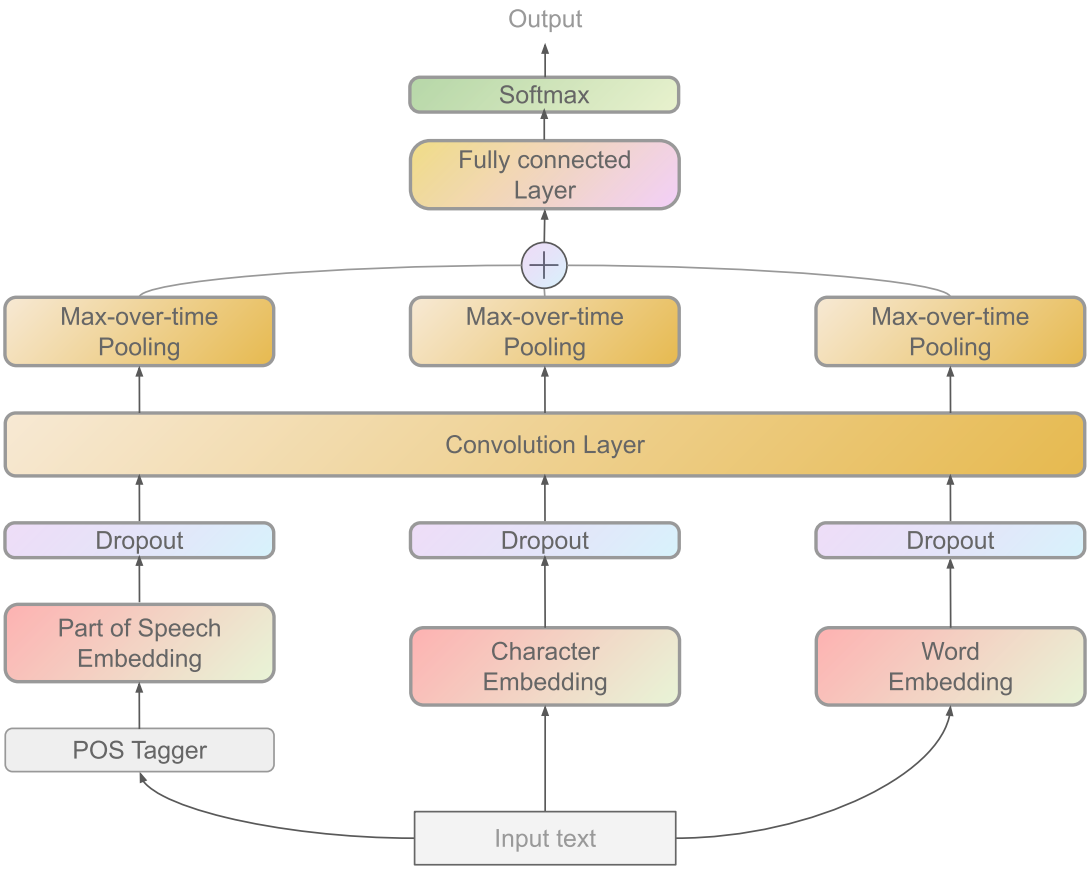

Figure 6.1: AARef architecture diagram. The writing samples are tokenized and fed to embeddings. Word embeddings, character bigram embeddings and POS embeddings are illustrated in the figure. The embeddings are then forwarded to convolutional layers with different window sizes. After convolution, max-over-time pooling is used in pooling layer, followed by the merge layer, which combines the features by the Add$(+)$ operation, then feed into a fully connected layer with a softmax output.

6.2.1 Model Construction

In light of the above discussion, we devised an AARef architecture utilizing word embeddings, character *n*-gram embeddings, and POS embeddings. The Multi-Channel convolutional network architecture is shown in Figure 6.1, and the details of the proposed model are as follows.

6.2.1.1 Embedding Layer

There are three embeddings used in parallel in our proposed network. For the word and character embeddings, the input text samples are tokenized and fed into each embedding separately. For the POS embedding, the input text is first tagged via TweetNLP Tagger [46]. After writing samples are converted into POS, they are tokenized and fed into POS embedding. All three embeddings are used as separate channels as shown in Figure 6.1. These feature embeddings are padded when necessary to have a constant size. Dropout is applied to the embeddings to avoid over-fitting. After dropout, the embeddings (inputs) are given to the convolutional layers with different window sizes in parallel.

6.2.1.2 Convolutional Layer

The three embedding channels are then separately forwarded to the convolutional layers with different window sizes. Figure 6.2 shows a detailed illustration of the multi branch convolutional refinement blocks. The convolutional blocks apply different sized filters to the input, feature embeddings in our case, and shift the filter until the end of the sequence. The window size of the convolution operation corresponds to the size of the filter that is applied to the input each time. Regarding the filters, two sets of filters are used with window sizes 3 and 4 separately.

For the convolutional branch with filter sizes 3 and 4, we used the Scaled Exponential Linear Unit (SELU) activation function [30] because of its self-normalizing properties. The convolutional layer can be represented as follows:

$$f(x) = \lambda \begin{cases} x, & \text{if } x > 0 \\ \alpha e^x - \alpha, & \text{if } x \leq 0 \end{cases} \tag{6.1}$$

where x is the output of the convolutional filters, and λ and α are pre-defined constants.

In addition to the convolution with window sizes 3 and 4, we added an identity mapping block on the left marked in a light gray color block. The features were upscaled to 500 first, then followed by two $L \times 3$ convolution operations with a batch normalization operation. An identity connection was added to the two convolution operations as shown in Figure 6.2 marked by a gray box. Given the output of the upscale layer, L, denoted as X_L, and the output of the identity block is X_{L+1}, the identity mapping block is represented by the following equation:

$$X_{L+1} = f(X_L) + X_L \tag{6.2}$$

This identity connection ensures that the information from the previous layer is not altered and can flow unimpeded to the next layer in the network.

For the convolutional layer identity mapping branch, we used a Leaky ReLU activation function [38]. The convolutional layer can be represented as follows:

$$f(x) = \begin{cases} x, & \text{if } x > 0 \\ 0.01x, & \text{if } x \leq 0 \end{cases} \tag{6.3}$$

where x is the output of the convolutional filters.

Since we used character embedding, word embedding, and part of speech embedding as individual input channels, each channel has an output, a feature map of size $1,536 (= 512 \times 3)$.

6.2.1.3 Pooling Layer

Convolutional layers are followed by max-over-time pooling [9], which takes the feature with the maximum value in a given filter. For the feature matrix output x by the convolutional layer, the max-over-time pooling operation outputs $\hat{x}$ as follows:

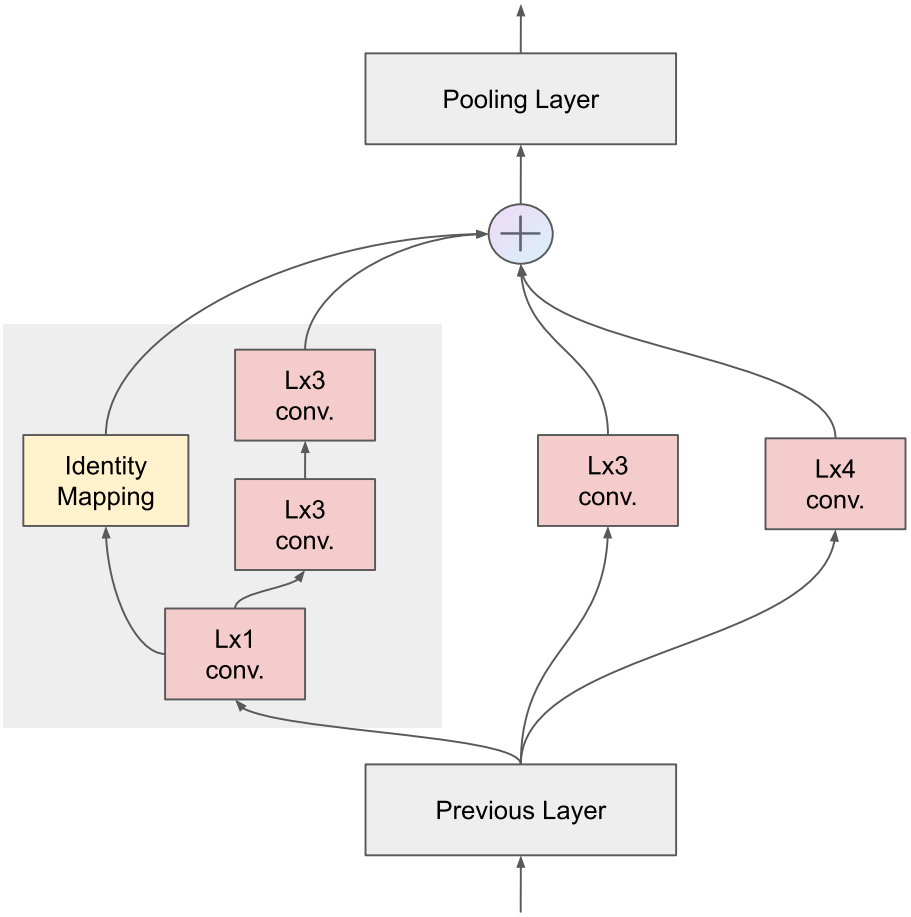

Figure 6.2: The detailed illustration of the multi branch convolution blocks.

$$\left[\hat{x}\right]_i = \max_t \left[x\right]_{i,t} \quad i = 1, \ldots, M \tag{6.4}$$

where t is the position in the sentence, i is the convolutional filter applied, and M is the number of the filters for each window size as shown in Figure 6.1.

In this way, we make the network keep the most salient features produced by convolutional layers. Thus, the number of selected values with max-over-time pooling is equal to the number of filters in the architecture per feature embedding.

6.2.1.4 Merge Layer

The resulting features are merged into a single feature map using add (+) operations in the merge layer as as shown in Figure 6.1. The add operation uses pairwise add operations between

feature vectors. The operation can be represented as follows:

$$x_{1..n} = a_{1..n} + b_{1..n} + c_{1..n} \tag{6.5}$$

where $x_{1..n}$ is the output of the merge layer. $a_{1..n}$, $b_{1..n}$, and $c_{1..n}$ represent the output of the last layer.

6.2.1.5 Fully Connected Layer with Softmax Output

After the merge layer, the last step is to feed the feature maps into the fully connected layer, followed by the softmax function for the classification. The softmax function, which is shown in equation 6.6, takes exponents of each element x_i from the input feature map x and normalizes them by dividing by the sum of all the exponentials. The equation is as follows:

$$S(x_i) = \frac{e^{x_i}}{\sum_{j=1}^{k} e^{x_j}} \tag{6.6}$$

where k is the total number of inputs. The softmax function scores the authors based on the closeness of the feature map to the class of an author. The network predicts the author of the writing samples with the highest score.

6.2.2 Hyper-parameters

Hyper-parameters were selected using grid search [7]. The dimension of the embeddings used was $d = 400$. The dropout was 25%. For the convolutional layer, there were 512 filters per window size, $M = 512$, which makes a total of 1,536 filters. Hence, only one feature was selected per filter. We use a batch size of 64 for the experiment. The epoch limit was 100 with an early stopping condition. Early stopping was based on validation accuracy improvement and stops after 20 epochs without an improvement. The learning rate updated during the training by the Adam optimizer [29]. The initial learning rate was $1e - 4$.

6.3 Experiments

In this section, we present our experiments and datasets in detail.

The parameters used in the experiments were the number of authors and the number of writing samples per author. One of these parameters was changed one at a time to investigate the effect of the given parameter on the performance of the system.

We used a preprocessing method similar to [56] in all datasets. We replaced the numbers, username references, date, time, and website URLs with pre-defined meta tags. As we need as much information as we can get from the tweets, we did not convert the text into lowercase to keep the case information.

Our experimental settings are summarized in the following.

6.3.1 Experiment I: Varying the Number of Authors

We explored how our proposed architecture performs with an increasing number of authors. We performed a set of performance evaluations with a different number of authors while keeping the number of writing samples constant. The number of writing samples per author (w) was 200. The number of authors (a) was used in experiments was 100, 200, 500, and 1,000. In this experiment, we used the same sampling used in [56]. We used 10-fold cross-validation on the experiments for each author group. Hence, we evaluated 40 train-test splits in total.

6.3.2 Experiment II: Varying the Number of Writing Samples

In this experiment, we investigated the impact of a different number of writing samples for author identification. We performed a set of performance evaluations with a varying number of writing samples per author where the number of authors was held to 50. The number of writing samples per author used in experiments was 50, 100, 200, and 500. We sampled 10 different disjoint sets of groups for each writing sample size. We used 10-fold cross-validation on the experiments for each author group. Hence, we evaluated 400 experiments in total.

6.4 Results

In addition to the baselines described in Chapter 5, we compared our proposed method with the following baselines:

- Rocha et al. [52]: Character 4-grams, word 1-5 *n*-grams, and POS 1-5 *n*-grams are extracted from the writing samples. The features only seen once in the dataset, hapax legomena, are removed to reduce the noise. Then, the extracted features are fed to the PMSVM for the classification.

- Theóphilo et al. [63]: The CNN architecture uses character 4-grams as input. The layers used in the architecture are dynamic k-max-pooling [24] to capture the most important features in each layer and uses a technique called folding [24] that applies an element-wise sum operation for every two rows. The same hyper-parameters that were used as in Theóphilo et al. [63] were used herein. The only difference in terms of hyper-parameters was the number of epochs. The number of epochs was increased to 100 epochs instead of 10 epochs. This change was necessary for fairness since the benchmark algorithms were allowed to go as high as 100 epochs. The best performing model in the validation set was selected and evaluated on the test set.

- CNN$_{WC}$: The architecture was discussed in Chapter 5. Word embeddings are initialized with pre-trained FastText embeddings and character bigrams are randomly initialized. Convolutional filters are shared with the embeddings. The embeddings and network parameters updated with backpropagation.

6.4.1 Results of Experiment I: Varying the Number of Authors

Table 6.1 shows the results of approaches discussed in Section 6.3 with varying the number of authors. In Table 6.1, the first column lists the algorithm used in the experiment, where the baseline methods are listed above the double lines and our proposed method is listed below the double lines. The rest of the columns list the accuracies of the algorithms with 100, 200,

500, and 1,000 authors, respectively. The number in bold marks the highest accuracy in each column.

The accuracies of the algorithm decreased with an increasing number of authors for all algorithms. CNN_{WC} was the best performer among the baseline methods. $\text{CNN}_{FastText}$ outperformed the other baseline methods without using character-level information. $\text{CNN}_{FastText}$ was followed by CNN_{Char2} which uses Character Bigrams.

The proposed method, AARef, was the best performer for all authors sets in Experiment I. The accuracies of the AARef algorithm with 100, 200, 500, and 1,000 authors were 57.18%, 55.31%, 47.99%, and 42.83%, respectively. On average, AARef performed better than the state-of-the-art results for all number of authors.

Table 6.1: Performance of the algorithms with varying the number of authors.

Algorithms	Authors			
	100	200	500	1,000
CNN_{Char1} [57]	49.24%	47.68%	41.37%	35.60%
CNN_{Char2} [57]	49.96%	48.84%	42.92%	37.55%
CNN_{W2V}	47.21%	45.52%	39.85%	34.73%
$\text{CNN}_{FastText}$	51.83%	50.25%	44.18%	38.74%
CNN_{WC} [2]	55.20%	53.14%	46.90%	41.28%
Rocha et al. [52]	43.99%	42.32%	36.63%	31.61%
Theóphilo et al. [63]	30.99%	31.44%	29.88%	27.76%
k-signatures [56]	42.50%	41.10%	35.50%	30.30%
LSTM_{Char2}	33.80%	33.50%	29.80%	24.80%
AARef	**57.18%*****	**55.31%*****	**47.99%*****	**42.83%*****

Asterisks denotes statistical significance of difference using MANOVA with p-values ≤ 0.001 (***).

The proposed AARef method was compared to the baseline methods CNN_{WC}, Theóphilo et al. [63], Rocha et al. [52] using a MANOVA. The results showed that the mean accuracies of the AARef method were statistically different than the baseline methods with $F = 1677.972$,

$F = 1867.684$, $F = 3212.107$, $F = 7411.100$ for 100, 200, 500, and 1000 authors, respectively (p-values ≤ 0.001 in all cases). The mean differences among the methods were further compared using Tukey's post hoc multiple comparison tests. The results of Tukey's post hoc multiple comparison tests were given in Table 6.2 only for the comparisons of the AARef to the baseline methods for the brevity of results. The Tukey's post hoc tests showed that the proposed AARef method outperformed all the baseline methods.

Table 6.2: Comparison of AARef to CNN_{WC}, Theóphilo et al. [63], Rocha et al. [52] on the different number of authors using a Multivariate Analysis of Variance with a Turkey's Post hoc multiple comparison tests.

					95% CI for Difference	
Author	Algorithm	Mean Difference	Std. Error	Sig.	Lower Bound	Upper Bound
100	CNN_{WC}	0.020	0.00416	0	0.008	0.0315
100	Theóphilo et al. [63]	0.262	0.00416	0	0.250	0.2736
100	Rocha et al. [52]	0.131	0.00427	0	0.119	0.1429
200	CNN_{WC}	0.022	0.00359	0	0.012	0.0317
200	Theóphilo et al. [63]	0.239	0.00359	0	0.229	0.2487
200	Rocha et al. [52]	0.131	0.00369	0	0.120	0.1409
500	CNN_{WC}	0.011	0.00216	0	0.005	0.0169
500	Theóphilo et al. [63].	0.181	0.00216	0	0.175	0.1871
500	Rocha et al. [52]	0.114	0.00221	0	0.108	0.1205
1000	CNN_{WC}	0.014	0.00119	0	0.011	0.0176
1000	Theóphilo et al. [63].	0.149	0.00119	0	0.146	0.1528
1000	Rocha et al. [52]	0.111	0.00130	0	0.108	0.1147

6.4.2 Results of Experiment II: Varying the Number of Writing Samples

Table 6.3 shows the results of the algorithms discussed in Section 6.3 with varying the number of writing samples per author. In Table 6.3, the first column lists the algorithm used in the experiment, where the baseline methods are listed above the double lines and our proposed

method is listed below the double lines. The rest of the columns list the accuracies of the algorithms with 50, 100, 200, and 500 writing samples per author, respectively. The number in bold marks the highest accuracy in each column.

As expected, accuracies increased as the number of writing samples increases for all the methods listed. Among the baseline methods, CNN_{WC} performed better than others. The performances of $\text{CNN}_{FastText}$, CNN_{Char1}, and CNN_{Char2} were very close. One can see that $\text{CNN}_{FastText}$ scaled better since $\text{CNN}_{FastText}$ was clearly better than CNN_{Char1}, and CNN_{Char2}.

The proposed method, AARef, was the best performer for all writing samples per author in Experiment II. The accuracies of the AARef algorithm with 50, 100, 200, and 500 writing samples per author are 56.10%, 63.51%, 70.05%, and 76.04%, respectively. On average, AARef performed better than the state-of-the-art results with 50, 100, 200, and 500 writing samples per author. It was noted that AARef had a smaller footprint compared to Rocha et al. [52], where the frequencies of each character 4-grams, word 1-5 *n*-grams, and POS 1-5 *n*-grams were used as inputs.

Table 6.3: Performances of the algorithms with varying number of writing samples.

Algorithms	Writing Samples			
	50	100	200	500
CNN_{Char1} [57]	51.40%	58.20%	64.07%	70.30%
CNN_{Char2} [57]	51.56%	58.25%	63.59%	69.80%
CNN_{W2V}	49.14%	56.68%	62.96%	69.70%
$\text{CNN}_{FastText}$	51.46%	59.14%	65.61%	72.46%
CNN_{WC} [2]	54.36%	62.17%	68.34%	74.50%
Rocha et al. [52]	42.88%	49.90%	57.43%	66.71%
Theóphilo et al. [63]	30.20%	38.32%	45.53%	56.00%
AARef	**56.10%*****	**63.51%*****	**70.05%*****	**76.04%*****

Asterisks denotes statistical significance of difference using MANOVA with p-values ≤ 0.001 (***).

The proposed AARef method was also compared to the baseline methods CNN_{WC}, Theóphilo et al. [63], Rocha et al. [52] using a Multivariate Analysis of Variance (MANOVA). The results

showed that the mean accuracies of the compared methods were statistically different than the baseline methods with $F = 98.47$, $F = 109.15$, $F = 140.56$, $F = 117.69$ for 50, 100, 200, and 500 authors, respectively (p-values ≤ 0.001 in all cases). However, the Tukey's post hoc multiple comparison tests showed that there was no statistical difference between the mean accuracies of the CNN$_{WC}$ and AARef methods as shown in Table 6.4 although the AARef method had higher level of mean accuracies. The AARef method outperformed the other the baseline methods.

Table 6.4: Comparison of AARef to CNN$_{WC}$, Theóphilo et al. [63], Rocha et al. [52] on the different number of authors using a Multivariate Analysis of Variance with a Turkey's Post hoc multiple comparison tests.

					95% CI for Difference	
Author	Algorithm	Mean Difference	Std. Error	Sig.	Lower Bound	Upper Bound
50	CNN$_{WC}$	0.017	0.017	0.74	-0.029	0.063
50	Theóphilo et al. [63]	0.259	0.017	0.00	0.213	0.305
50	Rocha et al. [52]	0.132	0.017	0.00	0.086	0.178
100	CNN$_{WC}$	0.013	0.016	0.83	-0.030	0.057
100	Theóphilo et al. [63]	0.252	0.016	0.00	0.209	0.295
100	Rocha et al. [52]	0.136	0.016	0.00	0.093	0.179
200	CNN$_{WC}$	0.017	0.014	0.59	-0.0193	0.054
200	Theóphilo et al. [63]	0.245	0.014	0.00	0.209	0.282
200	Rocha et al. [52]	0.126	0.014	0.00	0.090	0.163
500	CNN$_{WC}$	0.015	0.012	0.58	-0.017	0.048
500	Theóphilo et al. [63]	0.200	0.012	0.00	0.168	0.233
500	Rocha et al. [52]	0.093	0.012	0.00	0.061	0.125

6.5 Summary

We proposed a Multi-Channel CNN approach that takes advantage of multi-branch refinement convolution blocks. The architecture of proposed block includes different convolution

branches with identity mapping. The skipped connection in the identity mapping block helps the model capture salient information and passes the information on to the rest of the network. We compared the performance of AARef with state-of-the-art CNNs that use a Multi-Channel architecture. The results of the experiments show that the AARef performs better than the state-of-the-art CNNs on the Twitter dataset.

Chapter 7

Evaluating the Robustness of Machine Learning Methods

This chapter aims to evaluate the robustness of our proposed methods on a real-world dataset: Reddit dataset. The dataset was collected from the Reddit social media platform, which consists of 331 million posts from 811,930 unique authors. A number of filtering techniques are applied to the posts and authors to create a more focused dataset for our experiments, resulting in a micro-message dataset with only English content. Specifically, we excluded posts with unusually high or low character counts in order to avoid unidentifiable texts and ensure that the remaining writing samples were representative of the broader population of micro-messages.

7.1 Dataset Collection & Filtering

Several prepossessing and selection criteria were carefully designed and implemented to ensure the representative and reliability of our dataset. We imposed a maximum limit of 512 characters per post and excluded posts with fewer than 32 characters to ensure that the posts were sufficiently lengthy and distinctive for analysis. We also only included authors with a minimum of 100 and a maximum of 1,000 writing samples in our dataset. This was done to exclude authors who may not have produced enough writing to be representative, as well as to eliminate potentially automated accounts.

Our dataset collection was conducted over the course of one year, from January 2019 to January 2020, and the final dataset consisted of 331 million documents from 811,930 unique authors, with a raw size of 47 GB. We applied Near-Duplicate filtering and English content filtering to the dataset, as well as filtered and replaced self-identifying and noisy information

with special tokens. Reddit usernames were replaced with the same special token to avoid authorship leakage, and subreddits were replaced with subreddit special tokens to standardize the content among users.

We also applied filtering steps similar to those outlined in Section 3.3.1 for the Twitter dataset to reduce noise from URLs, numbers, and date-time information. These filtering steps can reduce the content length in terms of characters, which is why we removed posts with less than 32 characters to enforce the minimum character limit. This also resulted in the removal of some authors who had fewer than the minimum number of posts. In addition, we removed authors with less than 1,000 writing samples from the dataset.

These careful curation and selection steps allowed us to create a high-quality dataset for our experiments. The number of posts applied in the dataset after the filtering steps was 114,736,827. It was further reduced by limiting the maximum number of writing samples per author to 2,000. This limit was introduced to reduce the number of authors with low-quality posts and to eliminate automated accounts from the dataset. After removing authors with more than 2,000 posts, the total number of posts was reduced to 69,280,354, and the number of unique authors was reduced to 50,734.

The character count histogram of the final dataset is shown in Figure 7.2. The character count distribution in Figure 7.2 supports the claim of users' tendency to write micro-messages rather than longer messages. To further verify this statement, we illustrated the character count histogram to the raw Reddit dataset before applying preprocessing steps. The character count histogram of the raw Reddit dataset is shown in Figure 7.1. The raw Reddit dataset histogram shows a larger number of writing samples with fewer characters, similar to the processed dataset.

The histogram in Figure 7.3 shows the distribution of the number of posts per author in a one-year period from January 2019 to January 2020. Since we collected posts for one year period, limiting the number of writing samples to 2,000 per year equals 5.45 posts per day. This suggests that posting more than five comments per day is a demanding task that may lead to low-quality or automated content. The authors that have more than 2,000 posts in the one-year period were excluded from the dataset.

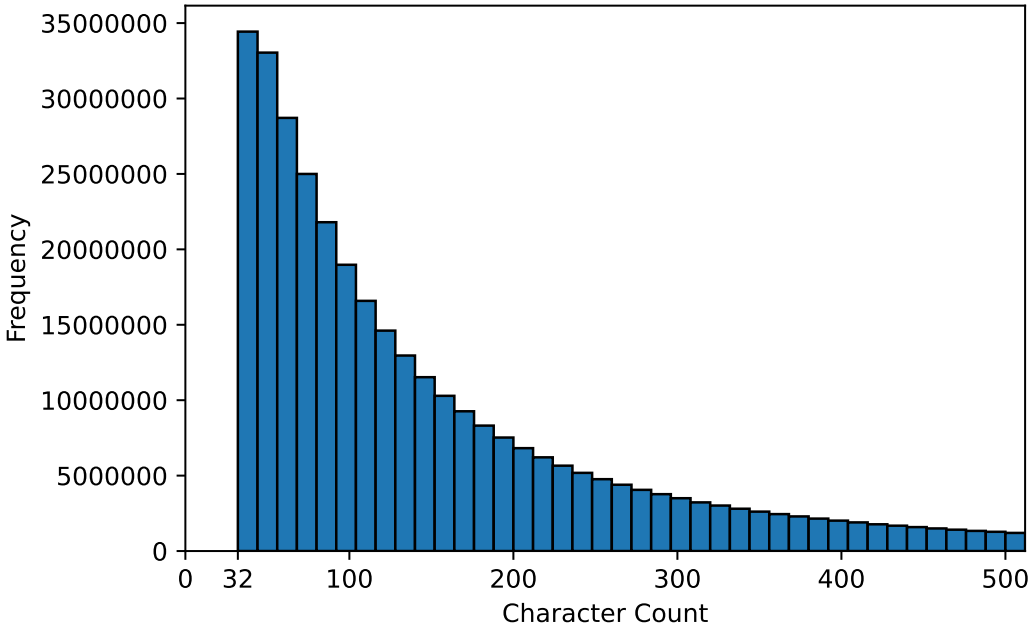

Figure 7.1: Character frequency histogram of Reddit dataset before applying preprocessing pipeline. The character counts are counted in buckets from 32 to 512. The character counts are grouped into 40 equally distributed ranges. Therefore, each bar represents the frequency of character counts within the range of twelve character count values.

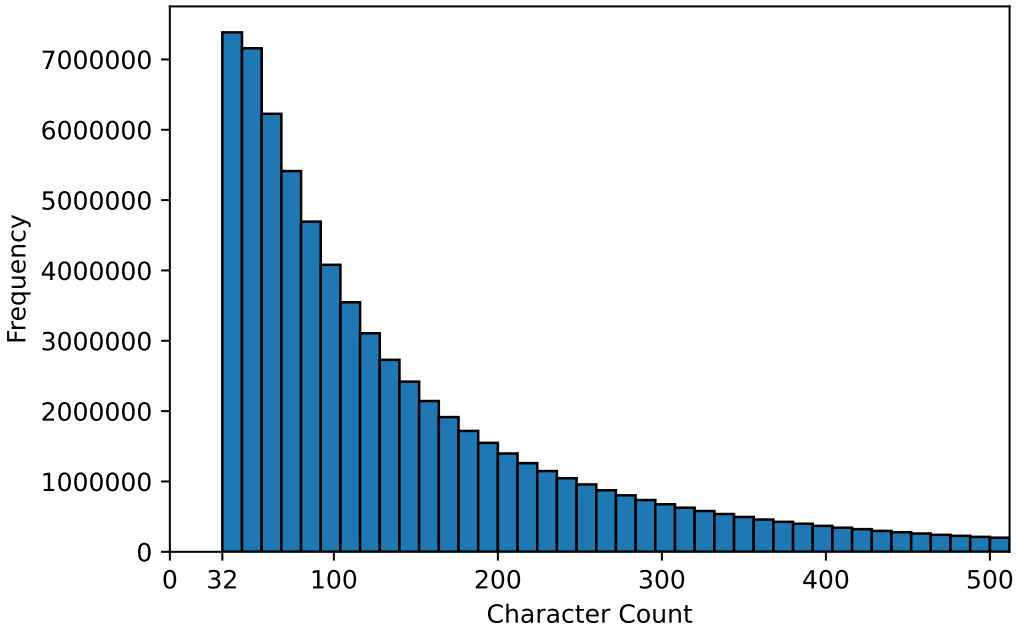

Figure 7.2: Character frequency histogram of Reddit dataset after preprocessing. The character counts are counted in buckets from 32 to 512. The character counts are grouped into 40 equally distributed ranges. Therefore, each bar represents the frequency of character counts within the range of twelve character count values.

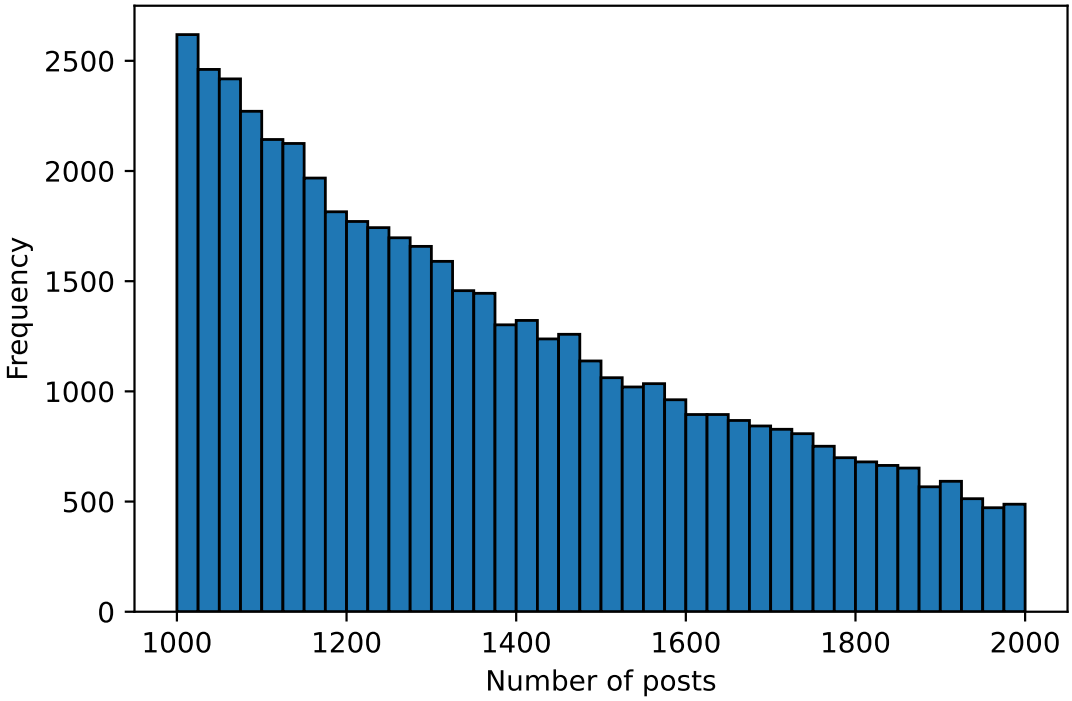

Figure 7.3: Histogram of the number of posts per author on Reddit dataset. The post counts are counted in buckets from 1,000 to 2,000. The number of posts are grouped into 40 equally distributed ranges. Therefore, each bar represents the frequency of character counts within the range of 25 posts. The frequency denotes the frequency of authors that posted a given number of posts.

7.1.1 Near-Duplicate Filtering

The near-duplicate documents are highly similar to one another and can be identified by their characteristic features, such as the inclusion of lengthy quotations with minimal original content. Including near-duplicate documents doesn't have enough stylistic information can introduce noise to performance evaluation metrics and the trained model. For example, authors that quote from the same book with different adjectives is very limited information for author identification systems. Also, such documents are prone to memorizing the training data. For this purpose, we filtered the near-duplicate writing samples from the dataset. However, detecting such documents in a large dataset presents a significant challenge, as the complexity of comparing each document pair scales quadratically with the number of samples. As a result, it is impractical to perform such comparisons using traditional methods. Therefore, we used a combination of two techniques used in the literature [34] to deduplicate the dataset.

The two techniques, suffix arrays and substring matching, are employed to identify duplicate content and filter the document in our dataset. As shown in Table 7.1 and 7.2, the duplicate content is highlighted with a yellow background. It should be noted that the duplicate substrings may come from different writing samples, but we have chosen to present examples of highly overlapping samples. For instance, the first two rows in Table 7.1 show a quote with only a small amount of unique content from the second author. Therefore, these two documents are highly similar, with the second post having particularly little unique content. To quantify this, we calculated the percentage of unique tokens in each writing sample and removed samples with a percentage below our predefined threshold of 70% from the dataset. This approach is great at detecting automated posts that follow similar patterns in their posts. For example, the posts demonstrated on the third to last rows of Table 7.1 capture the patterns of automated bad language detectors and in-game property exchange accounts. The detailed process of the suffix arrays and substring matching are as follows:

Username	Content
AcuteGryphon655	No I pointed out that cops stopping speeders is important because speeding is dangerous, so we shouldn't stop them from doing that. In fact, if anything, you're just trying to strawman me from saying, "stopping speeders is more important than you seem to accept," to, "his choice of wording is misleading for half of a sentence and therefore incorrect."
trameyes	No I pointed out that cops stopping speeders is important because speeding is dangerous, so we shouldn't stop them from doing that Great, but nobody said that we should... >In fact, if anything, you're just trying to strawman me from saying, "stopping speeders is more important than you seem to accept," to, "his choice of wording is misleading for half of a sentence and therefore incorrect."
Aylaviere	You have given **Pagoda Gate** to User '**aurorium**'. Click the button below to continue.
Aylaviere	You have given **Valentines Bruce Plushie** to User '**purpletamaninja**'. Click the button below to continue.
Aylaviere	You have given **Starry Chomby Plushie** to User '**ikiracake**'. Click the button below to continue.
profanitycounter	UH OH! Someone has been using stinky language and u/ya_boi_daelon decided to check u/profanitycounter's bad word usage. I have gone back one thousand posts and comments and reviewed their language usage.
profanitycounter	UH OH! Someone has been using stinky language and u/Bananplyte decided to check u/Bananplyte's bad word usage. I have gone back one thousand posts and comments and reviewed their language usage.
princesscatnip	IGN: Luna Deposited: "sBold" Drowzee — F — Level 8 — Poke Ball Requesting: Vaporeon
princesscatnip	IGN: Luna Deposited: "sTimid" Drowzee — F — Level 8 — Poke Ball Requesting: Lugia

Table 7.1: Detected near-duplicate documents I. The portion of the writing sample that was used more than once is highlighted with the yellow background.

7.1.1.1 Suffix Arrays

Comparing the parts of the documents with each other is impractical, as outlined before. It is computationally restrictive since the naive approach scales quadratically with the number of samples. For this purpose, we construct the suffix array of the entire dataset concatenated to a

Username	Content
professorbooty25	I'd argue that it's far better to spend the day feeling a big chocolate log brewing in your colon, slithering it's way into the launch chamber. You hold of going all day until you get home where you've been keeping Cleveland hostage for the last few hours. Then you finally the feeling of King Kong's finger sliding through you starfish and that satisfying plonk as fatboy drops into hiroshitma.
TempyMc101	I'd argue that it's far better to spend the day feeling a big chocolate log brewing in your colon, slithering it's way into the launch chamber. You hold of going all day until you get home where you've been keeping Cleveland hostage for the last few hours. Then you finally the feeling of King Kong's finger sliding through you starfish and that satisfying *plonk* as fatboy drops into hiroshitma.
Notaroboticfish	>Article 25. >(1) Everyone has the right to a standard of living adequate for the health and well-being of himself and of his family, **including food, clothing, housing and medical care** and necessary social services, and the right to security in the event of unemployment, sickness, disability, widowhood, old age or other lack of livelihood in circumstances beyond his control.
TheHess	Universal Declaration of Human Rights, Article 25: > Everyone has the right to a standard of living adequate for the health and well-being of himself and of his family, including food, clothing, housing and medical care and necessary social services, and the right to security in the event of unemployment, sickness, disability, widowhood, old age or other lack of livelihood in circumstances beyond his control.
crowdsourced	> Article 25 of the United Nations' 1948 Universal Declaration of Human Rights states that ""Everyone has the right to a standard of living adequate for the health and well-being of himself and of his family, including food, clothing, housing and medical care and necessary social services."

Table 7.2: Detected near-duplicate documents II. The portion of the writing sample that was used more than once is highlighted with the yellow background.

single sequence. Formally suffix array is computed with:

$$SA(D) = \text{arg\_sort}(\text{suffix}(\phi(D))) \tag{7.1}$$

where ϕ denotes the flattening operation of a dataset to generate a single sequence of concatenated all writing samples. The suffix operation generates all the suffixes from the input. Finally, arg_sort sorts the elements within the array based on their lexicographical order and

returns the indexes of the ordered list. For example, for input ”message” the suffix function produces following output:

$$\text{suffix}(\text{”message”}) = \{\text{”message”}, \text{”essage”}, \text{”ssage”}, \text{”sage”}, \text{”age”}, \text{”ge”}, \text{”e”}\} \tag{7.2}$$

Furthermore, the output of the SA(”message”) is computed as (5, 7, 2, 6, 1, 4, 3) for a 1-indexed representation. In our experiments, we constructed the suffixes based on the tokens rather than characters.

7.1.1.2 Substring matching

The use of constructed suffix arrays enables the identification of common substrings. To locate all repeated sequences, one can simply scan the suffix array sequentially and search for sequences that possess a shared prefix of a predetermined minimum length. The matching sequences are identified and preserved for further filtering.

7.2 Experiments

In this section, we conducted experiments on Reddit dataset with proposed methods in Chapter 6 and studied the performances of author identification frameworks in relation to two parameters: the number of writing samples per author and the number of authors. The same number of writing samples are sampled from each author across all experiments to ensure balance in the importance of the classes. This allowed us to evaluate the effects of specific parameters on the performance of author identification frameworks. Our experimental settings are summarized in the following.

7.2.1 Experiment I: Varying the Number of Authors

We explored how our proposed architecture performs with an increasing number of authors. We performed a set of performance evaluations with a different number of authors while keeping the number of writing samples constant. The number of writing samples per author was 200. The number of authors used in the experiments was 50, 100, 200, 500, and 1,000. The authors

in these subsets are randomly sampled from the Reddit dataset. A total of 200 writing samples are randomly selected for each author. The performance of the frameworks was assessed using accuracy as the evaluation metric, which was calculated as the number of correct predictions divided by the total number of predictions.

7.2.2 Experiment II: Varying the Number of Writing Samples

In this experiment, we investigated the impact of different numbers of writing samples for the author identification task. We performed a set of performance evaluations with varying writing samples per author, where the number of authors was held to 100. The number of writing samples per author used in the experiments was 50, 100, 200, 500, and 1,000. Same as the Experiment I, the framework's performance was assessed using accuracy as the evaluation metric.

7.3 Results

We use seven state-of-the-art models to evaluate the robustness of the proposed methods, including GPT-2 [50], BERT [64], XLNet [66] and RoBERTa [35]. The details of the baseline models used are as follows:

- GPT-2 [50]: Autoregressive model based on the transformer architecture. The model was pretrained on WebText (web pages from outgoing links in Reddit with 3 karmas or more). We used the GPT-2_{BASE} variation, which has a 12-layer transformer architecture and a hidden size of 768. It also has 12 attention heads, which allow the model to attend to different parts of the input and context simultaneously.

- BERT [64]: This model corrupts the inputs by using random masking. The model's objective is to predict the original sentence, but it also has a secondary objective: the inputs are two sentences A and B (separated by a token), and with probability 50%, the sentences are consecutive in the corpus. In the remaining 50%, the sentences are not related. The model must predict whether the sentences are consecutive or not. In our experiments, we used four variations of BERT:

- $\text{BERT}_{\text{SMALL}}$: This variation has 4 layers, 512 hidden units, and 8 attention heads. It is generally faster to train and has a smaller model size compared to the larger variations.
- $\text{BERT}_{\text{MEDIUM}}$: This variation has 8 layers, 512 hidden units, and 8 attention heads. It generally has better performance than the SMALL variation, but may take longer to train and have a larger model size.
- $\text{BERT}_{\text{BASE}}$: This is the standard BERT model, and it has 12 layers, 768 hidden units, and 12 attention heads. It strikes a balance between model performance and efficiency.
- $\text{BERT}_{\text{LARGE}}$: This is the largest BERT variation, with 24 layers, 1024 hidden units, and 16 attention heads. It generally has the best performance among all the variations, but may be slower to train and have a larger model size.

- XLNet [66]: XLNet is a unsupervised language representation learning method based on a generalized permutation language modeling objective. Additionally, XLNet employs Transformer-XL as the backbone model.
- RoBERTa [35]: This model is similar to BERT, but it includes improved pretraining tricks such as masking tokens differently at each epoch, and using a chunk of contiguous texts (up to 512 tokens) instead of just two sentences for the next sentence prediction (NSP) loss. It also uses byte pair encoding with bytes as the subunit instead of characters, and trains with larger batches. In our experiments, we used the $\text{RoBERTa}_{\text{BASE}}$ variation, which has a 12-layer transformer architecture with a hidden size of 768 and 12 attention heads.

7.3.1 Results of Experiment I: Varying the Number of Authors

Table 7.3 presents the performance of the algorithms in Experiment I, where the number of authors is varied while the number of writing samples per author is held constant at 200. The last column shows the number of trainable parameters. The model achieving the highest accuracy within each subset is emphasized with bold text, while the model with the second highest

accuracy is indicated with an underline. Among the algorithms, AARef achieves the highest accuracy, with 53.18% for 50 authors, 45.32% for 100 authors, 39.16% for 200 authors, 30.83% for 500 authors, and 24.20% for 1,000 authors. On the other hand, CNN_{Char1} has the lowest accuracy, with 40.00% for 50 authors. The performance gap between AARef and the second best baseline, $\text{RoBERTA}_{\text{BASE}}$ marginal at the smallest subset, 50 authors. This suggests that AARef struggles when there is limited data to train on. However, the performance gap increases when the data size gets larger.

It is also worth noting that the number of parameters has a positive correlation with the accuracy of the algorithms. Specifically, the algorithms with more parameters, such as $\text{BERT}_{\text{LARGE}}$, have higher accuracy than the BERT variants with fewer parameters. This suggests that performance scale with the model size leading larger models to have better performance. The AARef outperforms the more complex models while only optimizing 25 million parameters. Compared to transformer baselines, AARef is with the lowest number of parameters. While $\text{RoBERTA}_{\text{BASE}}$ performs similarly in some cases, it requires more than four times more parameters compared to AARef. This demonstrates the efficiency of the AARef.

7.3.2 Results of Experiment II: Varying the Number of Writing Samples

The results of Experiment II, presented in Table 7.4, illustrate the performance of the algorithms in terms of accuracy, with the number of authors held constant at 100 and the number of writing samples varied. The final column shows the number of trainable parameters. The model achieving the highest accuracy within each subset is emphasized with bold text, while the model with the second highest accuracy is indicated with an underline. The AARef outperforms the baseline models on 100, 200, 500, and 1,000 writing sample subsets with 41.52%, 51.05%, 54.12%, and 62.23% accuracy, respectively. However, on the smallest subset $\text{BERT}_{\text{LARGE}}$ is the best performer, followed by $\text{BERT}_{\text{BASE}}$. Although AAref is behind the $\text{BERT}_{\text{LARGE}}$ and $\text{BERT}_{\text{BASE}}$ models on 50 writing samples per author experiment, AAref has a lower computational complexity compared to both of these models, as it has significantly fewer trainable parameters: 4X fewer than $\text{BERT}_{\text{BASE}}$ and 13X fewer than $\text{BERT}_{\text{LARGE}}$.

Table 7.3: Performance of the algorithms with varying the number of authors on Reddit dataset. The model achieving the highest accuracy within each subset is emphasized with bold text, while the model with the second highest accuracy is indicated with an underline.

Algorithms	Authors					No. of Parameters
	50	100	200	500	1,000	
CNN_{Char1} [57]	40.00%	31.80%	25.78%	23.05%	21.18%	5M
CNN_{Char2} [57]	34.70%	29.45%	29.67%	26.21%	23.50%	6M
CNN_{Char3}	40.00%	32.65%	28.65%	23.83%	22.10%	10M
CNN_{Char4}	39.65%	31.55%	26.33%	22.05%	19.37%	29M
$\text{CNN}_{FastText}$	33.70%	29.25%	24.88%	20.39%	17.70%	13M
CNN_{WC} [2]	46.00%	39.50%	32.35%	27.95%	23.20%	32M
$\text{GPT-2}_{\text{BASE}}$ [50]	43.00%	37.85%	32.48%	28.14%	<u>24.76%</u>	125M
$\text{BERT}_{\text{SMALL}}$ [64]	41.10%	34.40%	25.15%	20.71%	16.58%	29M
$\text{BERT}_{\text{MEDIUM}}$ [64]	44.10%	35.50%	28.45%	21.24%	17.39%	41M
$\text{BERT}_{\text{BASE}}$ [11]	46.66%	38.25%	31.75%	25.49%	21.12%	110M
$\text{BERT}_{\text{LARGE}}$ [11]	48.10%	40.10%	32.51%	<u>27.72%</u>	23.34%	340M
$\text{XLNet}_{\text{BASE}}$ [66]	47.00%	40.55%	34.28%	26.91%	22.00%	125M
$\text{RoBERTa}_{\text{BASE}}$ [35]	<u>52.66%</u>	<u>42.40%</u>	<u>36.25%</u>	27.13%	23.86%	117M
AARef	**53.18%**	**45.32%**	**39.16%**	**30.83%**	**25.20%**	25M

Table 7.4: Performance of the algorithms with varying the number of writing samples on the Reddit dataset. The model achieving the highest accuracy within each subset is emphasized with bold text, while the model with the second highest accuracy is indicated with an underline.

Algorithms	Writing Samples					No. of Parameters
	50	100	200	500	1,000	
CNN_{Char1} [57]	18.40%	24.30%	31.25%	42.94%	50.16%	5M
CNN_{Char2} [57]	17.24%	23.40%	32.13%	44.26%	50.92%	6M
CNN_{Char3}	21.80%	29.10%	32.55%	41.74%	46.98%	10M
CNN_{Char4}	19.80%	27.70%	32.20%	40.66%	46.43%	29M
$\text{CNN}_{FastText}$	20.85%	26.95%	33.75%	44.26%	51.64%	13M
CNN_{WC} [2]	22.40%	36.60%	36.20%	49.58%	59.10%	32M
$\text{GPT-2}_{\text{BASE}}$ [50]	13.20%	21.60%	36.90%	48.76%	56.57%	125M
$\text{BERT}_{\text{SMALL}}$ [64]	21.20%	28.60%	33.70%	42.86%	50.01%	29M
$\text{BERT}_{\text{MEDIUM}}$ [64]	21.20%	30.80%	35.00%	43.90%	50.62%	41M
$\text{BERT}_{\text{BASE}}$ [11]	<u>26.60%</u>	37.15%	37.05%	48.98%	55.10%	110M
$\text{BERT}_{\text{LARGE}}$ [11]	**27.00%**	<u>40.80%</u>	<u>48.51%</u>	51.14%	57.08%	340M
$\text{XLNet}_{\text{BASE}}$ [66]	22.00%	33.00%	40.10%	50.26%	56.28%	125M
$\text{RoBERTa}_{\text{BASE}}$ [35]	25.60%	34.70%	46.75%	<u>53.90%</u>	<u>58.89%</u>	117M
AARef	25.87%	**41.52%**	**51.05%**	**54.12%**	**60.23%**	25M

As observed in Experiment I, the performance of the algorithms tends to improve as the model size increases. This trend is evident in the variations of BERT models, where the performance improves on all subset sizes as the model size increases from $\text{BERT}_{\text{SMALL}}$ to $\text{BERT}_{\text{LARGE}}$. This suggests that scaling Transformer-based models to larger configurations can lead to performance improvement.

One possible reason for the relatively lower performance of AARef on the smallest subset of 50 authors in the second experiment could be the limited amount of data available for learning. In comparison, the other models may be able to utilize the prior knowledge transferred from the pretraining phase to achieve better performance on this subset. However, when there is sufficient data available, AARef demonstrates its strength and outperforms the other baselines. This can be seen in the higher accuracy achieved by AARef on the larger subsets in the second experiment, where it outperforms all the other algorithms. Similar to the results of Experiment I, Experiment II results suggest that AARef is a highly efficient model that is able to achieve good performance with a relatively low number of parameters, especially when there is sufficient data for learning.

7.4 Summary

In this chapter, a large-scale Reddit dataset is introduced with various experimental settings. The dataset is filtered to avoid low-quality content by removing self-identifying information and near-duplicate content. The resulting dataset is split into two sets of experiments with a varying number of authors and writing samples per author. The baseline methods and proposed methods in this thesis are evaluated with these experimental configurations. The results of these experiments showed that AARef outperformed the baseline methods in nine out of ten experiments, demonstrating its effectiveness in the analysis of micro-messages from social media platforms.

Chapter 8

Exploring the Anonymity of Social Media Users using Micro-Messages

In this chapter, we study the anonymity of social media users in various conditions. Specifically, we investigate the effects of the number of writing samples per author and the number of authors on anonymity. To better reflect real-world scenarios, our experiments were conducted on a large scale, with a million tweets. Through this work, we aim to answer the question of how anonymous social media identities are, and how easily they can be tracked using different handles.

Our findings are essential foundations for understanding the applicability of the performant author identification systems we demonstrated in previous research [1, 2]. These insights are important for future research on the robustness of author identification models and have direct implications for related topics such as identifying cybercriminals.

8.1 Introduction

Social networks are hard to moderate because of their large size and unique properties. One such property is that one person can have multiple identities by creating multiple social media accounts under different handles. The same user with multiple handles introduces noise to the system because writing samples provided from different accounts are recognized as different users. The system will be penalized for predicting the wrong identity even though it might predict the correct identity that controls both accounts. Furthermore, some social media handles are controlled using automated software, also known as bots [17, 31]. The bots can follow simple rules and phrases or can have more complex sets of rules for posting micro-messages. They

introduce noise to author identifiers as detecting the writing style of bots does not generalize well to the rest of the users.

There are several works available on the Author Identification task of micro-messages. However, none of those mentioned earlier works explore the performance of author identification systems under various conditions to fully understand their capabilities under real-world scenarios. In this work, we conducted experiments to answer the following question: "How confidently can we Identify an author on social media with X number of posts?". For this purpose, we perform a series of experiments under various conditions concerning the size of the system and the number of writing samples available from each user.

We use a real-world dataset from Twitter to test the model and compare how likely the identities can be identified under different conditions. Our experimental results show exciting findings that increasing the number of authors sometimes helps the author identifiers when the number of writing samples is limited. However, as expected, when we have more writing samples from the authors, the performance of the author identifiers increases, and we can make confident predictions when there are more than fifty writing samples from the same author. This work answers the question of how anonymous social media identities are and how likely these identities can be tracked under different handles. We believe these findings are essential foundations for understanding the applicability of the performant Author Identification systems we demonstrated in [1, 2]. These insights are important for future research in how robust the author identification models are and directly impact topics such as identifying cybercriminals.

8.2 Experiments

In this section, experimental configurations are presented, which were designed to assess the impact of varying the number of authors and writing samples on the performance of our model. A series of experiments were conducted based on the results of the first experiment. We followed the preprocessing steps outlined in previous work [1] in all subsets. We replaced the numbers, username references, date, time, and website URLs with pre-defined meta tags. To maximize the ability to capture stylistic features from the tweets, we did not convert the text

into lowercase to keep the case information. This is preferred to capture the way users capitalize certain words.

8.2.1 Experiment I: Varying Number of Authors and Writing Samples

In the first experiment, the parameters used in the experiments were the number of authors and the number of writing samples per author. We conducted experiments with the following parameters 10, 20, 50, 100, 200, 500, and 1,000 for all pairs of writing samples and authors. Each pair of the number of writing samples and the number of authors are evaluated with a 10-fold crossover. Therefore, each pair is trained ten times to be evaluated against a distinct test set. There are seven possible values for both parameters. Therefore, there are 49 configurations to experiment with. Hence, we experimented on 490 different subsets. The models are trained from scratch without any pretraining on the Author Identification task.

8.2.2 Experiment II: Varying Number of Authors and Writing Samples with AARef+

In the second experiment, we used the same range of combinations of the number of authors and writing samples as in the first experiment, but with the addition of a pretraining step using a pretraining dataset. The pretraining dataset was tested with two variations. The first variation has 1,000 authors with 1,000 writing samples per author and is denoted as AARef+$_{1000}$. The second variation has 2,000 authors with 1,000 writing samples per author and is denoted as AARef+$_{2000}$. The model was first trained on the pretraining dataset and then fine-tuned on the evaluation dataset.

8.2.3 Experiment III: Evaluation of AARef+

In the third experiment, the pretrained approaches, AARef+$_{1000}$ and AARef+$_{2000}$, were evaluated on the varying number of authors and a varying number of writing samples introduced in Chapter 6 and Chapter 5. This experiment also compared the pretrained approaches with previous results and baselines.

8.2.4 Results

The performance of AARef and pretrained variation of AARef are evaluated with various parameters.

8.2.5 Results of Experiment I: Varying Number of Authors and Writing Samples

Figure 8.1 displays the performance of AARef under different numbers of writing samples and authors. Table 8.1 provides further details. The first column lists the number of authors used in the experiment, and the remaining columns list the number of writing samples per author.

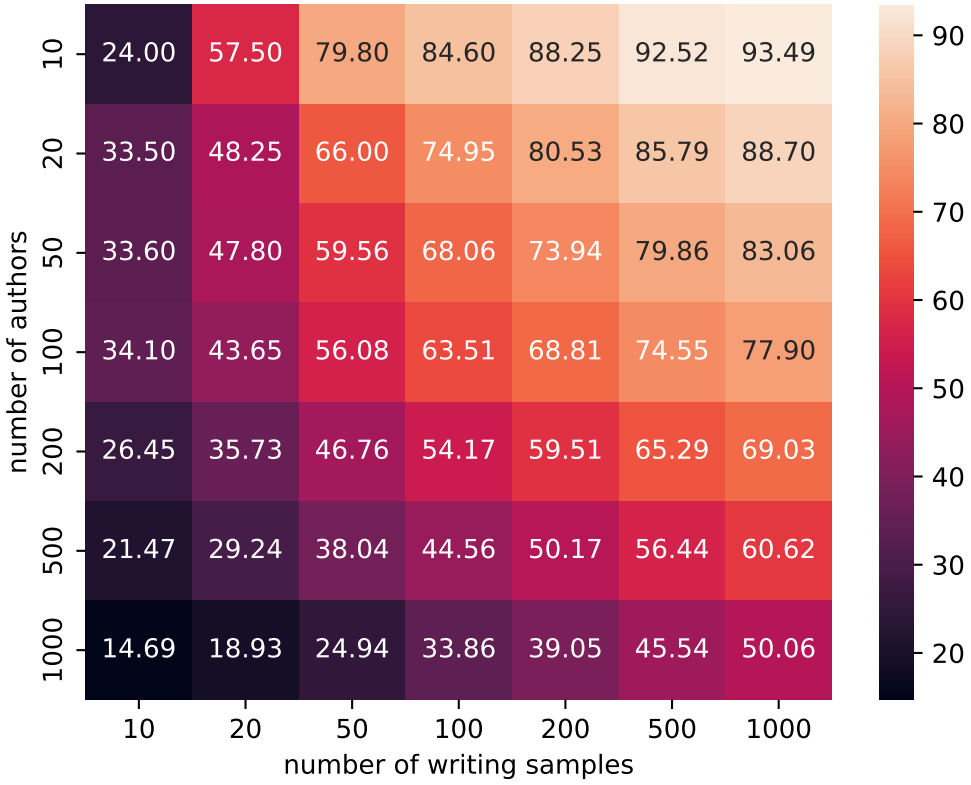

Figure 8.1: Evaluation with the varying number of authors and writing samples. The heatmap plot illustrates the performance of Author Identification methods with varying pairs of the number of writing samples and number of authors.

In the smallest setting of writing samples per author, AARef was unable to predict the authors confidently. However, accuracy increases with the number of authors when there are ten writing samples per author. This is not the case for the other writing sample per author

conditions, where accuracy decreases with the number of authors. This suggests that AARef cannot gather sufficient information about the environment in these conditions. Additionally, the performance of ten writing samples per author highly depends on the random sampling of authors and writing samples. Once the number of writing samples per author increases, AARef can achieve more than 90% accuracy. The performance improvement with large writing samples also indicates that AARef can scale with larger writing samples.

8.2.6 Results of Experiment II: Varying Number of Authors and Writing Samples with AARef+

In Figure 8.2, the performance of AARef+$_{1000}$ is illustrated similarly to the performance illustration of AARef in Figure 8.1. The heatmap plot shows the accuracy of the number of authors and the number of writing sample pairs. It is straightforward to see the performance of AARef+$_{1000}$ significantly improved on smaller subsets over AARef. For example, on the smallest subset with ten authors and ten writing samples per author accuracy of AARef increased from 24% to 71% with AARef+$_{1000}$, suggesting a 195% increase in performance. Although this is the extreme case with minimal training data, it shows how well AARef+ can perform in environments with low data availability.

Since the performance improvement of AARef+$_{1000}$ is promising, we explored the performance of the larger pretraining sets. For this purpose, 1,000 more authors were sampled from the Twitter dataset. Therefore, AARef+$_{2000}$ trained with a total of 2,000 authors with 1,000 writing samples per author. Table 8.2 shows the average accuracy and standard deviation of AARef+$_{2000}$. For comparison Table 8.1 shows the average accuracy and standard deviation of AARef+$_{1000}$. The difference between the two models is marginal, especially on larger subsets. For example, on 1,000 authors with 1,000 writing samples per author AARef+$_{1000}$ outperforms AARef+$_{2000}$ by less than 2%. On the smaller subsets, the best performer change between experiments. The smallest subset with ten authors and ten writing samples AARef+$_{1000}$ outperforms AARef+$_{2000}$. However, with 20 authors and ten writing samples AARef+$_{2000}$ outperforms AARef+$_{1000}$. The comparable performance of AARef+$_{1000}$ and AARef+$_{2000}$ suggests that 1,000 are sufficient on these experimental configurations to pretrain the AARef+ with identical settings.

Table 8.1: Performance evaluation of AARef+$_{1000}$ with the varying number of authors and writing samples on the Twitter dataset. The table shows the average and standard deviation (±) of the accuracy. The columns represent the number of writing samples per author. The first values in the rows represent the number of authors.

a\w	10	20	50	100	200	500	1,000
10	71.00±7.0	80.00±8.4	83.20±5.1	91.10±2.0	92.00±1.9	93.24±0.6	94.03±0.5
20	53.50±9.9	69.00±4.6	77.30±4.7	81.85±2.1	84.40±1.5	87.64±0.8	88.76±0.8
50	47.80±6.2	60.00±4.7	68.24±2.8	73.64±1.3	76.09±1.1	80.59±0.7	84.95±0.6
100	43.80±3.3	52.55±1.7	60.74±1.2	65.94±0.8	69.14±1.1	74.27±0.5	78.72±0.4
200	34.65±2.2	42.13±2.7	50.08±1.5	55.07±0.8	58.73±0.8	63.08±0.5	69.22±0.2
500	27.58±1.3	33.93±1.4	40.13±0.7	45.30±0.3	52.98±0.3	57.27±0.2	61.10±0.2
1,000	24.60±0.7	30.18±0.5	35.55±0.5	39.21±0.4	42.50±0.3	46.26±0.3	50.70±0.2

Table 8.2: Performance evaluation of AARef+$_{2000}$ with the varying number of authors and writing samples on the Twitter dataset. The table shows the average and standard deviation (±) of the accuracy. The columns represent the number of writing samples per author. The first values in the rows represent the number of authors.

a\w	10	20	50	100	200	500	1,000
10	68.00±9.7	78.00±9.3	86.60±3.6	89.60±2.5	91.95±1.0	93.48±0.6	94.16±0.6
20	57.50±6.8	69.00±9.6	76.80±4.2	81.65±2.3	84.57±1.2	88.02±0.7	89.41±0.7
50	47.60±4.7	58.50±4.2	69.00±2.5	73.98±1.0	76.74±0.9	80.75±0.6	82.36±0.4
100	43.60±4.5	52.05±2.6	61.96±2.0	66.71±1.4	70.39±1.3	74.17±0.4	79.31±0.3
200	34.50±3.1	42.10±2.4	52.05±1.5	56.38±0.7	59.57±0.8	63.61±0.4	68.14±0.3
500	27.20±1.3	34.36±1.4	41.05±0.9	45.08±0.4	50.60±0.3	57.53±0.2	61.00±0.2
1000	24.94±0.8	30.76±0.6	36.48±0.6	39.92±0.5	42.95±0.4	46.52±0.1	49.75±0.2

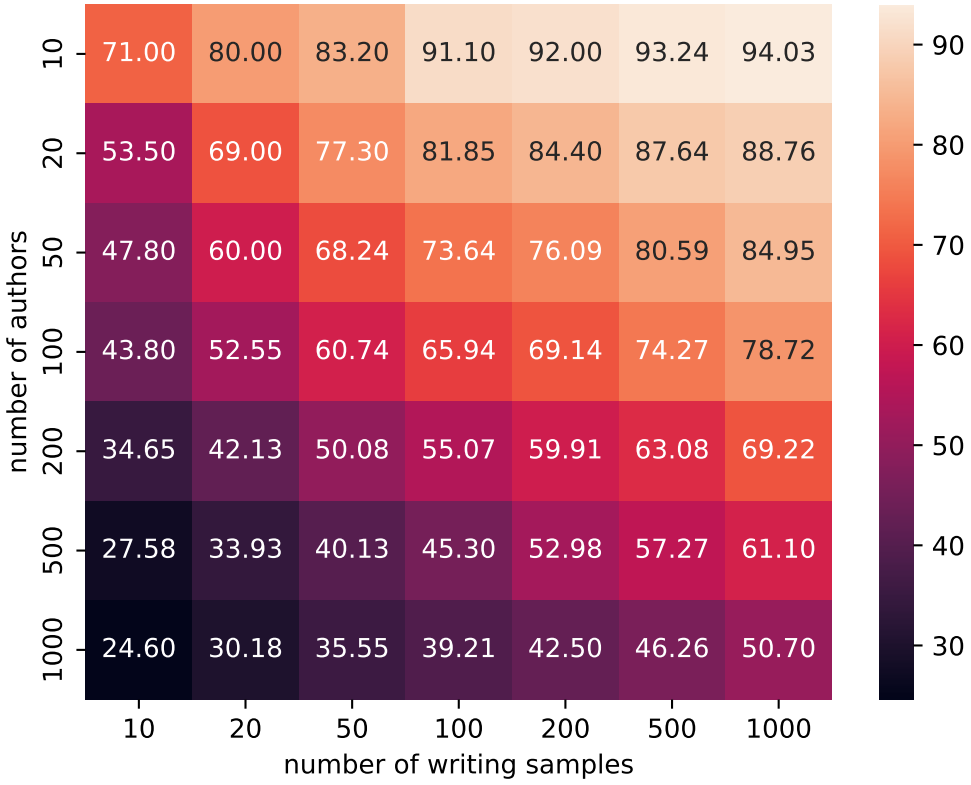

Figure 8.2: Evaluation of AARef+$_{1000}$ with the varying number of authors and writing samples. The heatmap plot illustrates the performance of Author Identification methods with varying pairs of the number of writing samples and number of authors.

8.2.7 Results of Experiment III: Evaluation of AARef+

Table 8.3 shows the results of approaches discussed in Section 6.3 with varying the number of authors. In Table 8.3, the first column lists the algorithm used in the experiment, where the baseline methods are listed above the double lines, and our proposed method is listed below the double lines. The rest of the columns list the accuracies of the algorithms with 100, 200, 500, and 1,000 authors, respectively. The number in bold marks the highest accuracy in each column.

The variations of the pretrained model AARef+$_{1000}$ and AARef+$_{2000}$ outperformed AARef on the varying number of authors experiment. The best performing model was AARef+$_{2000}$ which improved the performance of AARef+$_{1000}$. The performance improvement is larger on smaller subsets, and this supports the argument that the model requires sufficient task-related

data to perform its' best. For example, the improvement of AARef+$_{2000}$ on 100 authors over AARef is 5%, while an improvement on 200 authors is less than 3.1%. Furthermore, the improvement of AARef+$_{2000}$ on 500 authors over AARef is 1.8%, while an improvement on 1,000 authors is less than 1%.

Table 8.3: Performance of the algorithms with the varying number of authors.

Algorithms	Authors			
	100	200	500	1,000
CNN$_{Char1}$ [57]	49.24%	47.68%	41.37%	35.60%
CNN$_{Char2}$ [57]	49.96%	48.84%	42.92%	37.55%
CNN$_{W2V}$	47.21%	45.52%	39.85%	34.73%
CNN$_{FastText}$	51.83%	50.25%	44.18%	38.74%
CNN$_{WC}$ [2]	55.20%	53.14%	46.90%	41.28%
Rocha et al. [52]	43.99%	42.32%	36.63%	31.61%
k-signatures [56]	42.50%	41.10%	35.50%	30.30%
AARef	57.18%	55.31%	47.99%	42.83%
AARef+$_{1000}$	59.92%	56.57%	48.52%	42.89
AARef+$_{2000}$	**60.32%**	**57.02%**	**48.89%**	**42.91**

Table 8.4 shows the results of approaches discussed in Section 6.3 with varying the number of writing samples. In Table 8.4, the first column lists the algorithm used in the experiment, where the baseline methods are listed above the double lines, and our proposed method is listed below the double lines. The rest of the columns list the accuracies of the algorithms with 100, 200, 500, and 1,000 authors, respectively. The number in bold marks the highest accuracy in each column.

The variations of the pretrained model AARef+$_{1000}$ and AARef+$_{2000}$ outperformed AARef AARef on the varying number of writing samples experiment. The best performing model was AARef+$_{2000}$ which improved the performance of AARef+$_{1000}$. The performance improvement is larger on smaller subsets, and this supports the argument that the model requires sufficient

Table 8.4: Performances of the algorithms with the varying number of writing samples.

Algorithms	Writing Samples			
	50	100	200	500
CNN_{Char1} [57]	51.40%	58.20%	64.07%	70.30%
CNN_{Char2} [57]	51.56%	58.25%	63.59%	69.80%
CNN_{W2V}	49.14%	56.68%	62.96%	69.70%
$\text{CNN}_{FastText}$	51.46%	59.14%	65.61%	72.46%
CNN_{WC} [2]	54.36%	62.17%	68.34%	74.50%
Rocha et al. [52]	42.88%	49.90%	57.43%	66.71%
Theóphilo et al. [63]	30.20%	38.32%	45.53%	56.00%
AARef	56.10%	63.51%	70.05%	76.04
AARef+$_{1000}$	61.87%*	67.37%*	71.45%	76.12
AARef+$_{2000}$	**62.19%***	**68.36%***	**72.31%***	**76.19**

task-related data to perform its' best. For example, the improvement of AARef+$_{2000}$ on 100 authors over AARef is 9.7%, while an improvement on 200 authors is around 7.6%. Furthermore, the improvement of AARef+$_{2000}$ on 500 authors over AARef is 3.2%, while an improvement on 1,000 authors is less than 1%.

In Figure 8.3 the performances of the proposed approaches AARef, AARef+$_{1000}$, and AARef+$_{2000}$ are further compared. For each subset, and each variation, ten distinct groups with ten splits. Therefore, each box visualizes the accuracy of 100 runs.

To test the statistical significance of the performance improvement of AARef+$_{1000}$ and AARef+$_{2000}$ over AARef, two independent Student's t-test is conducted. AARef+$_{1000}$ significantly outperforms (p-values ≤ 0.001) AARef on subsets with 50 and 100 writing samples. AARef+$_{1000}$ significantly outperforms (p-values ≤ 0.001) AARef on subsets with 50, 100, and 200 writing samples.

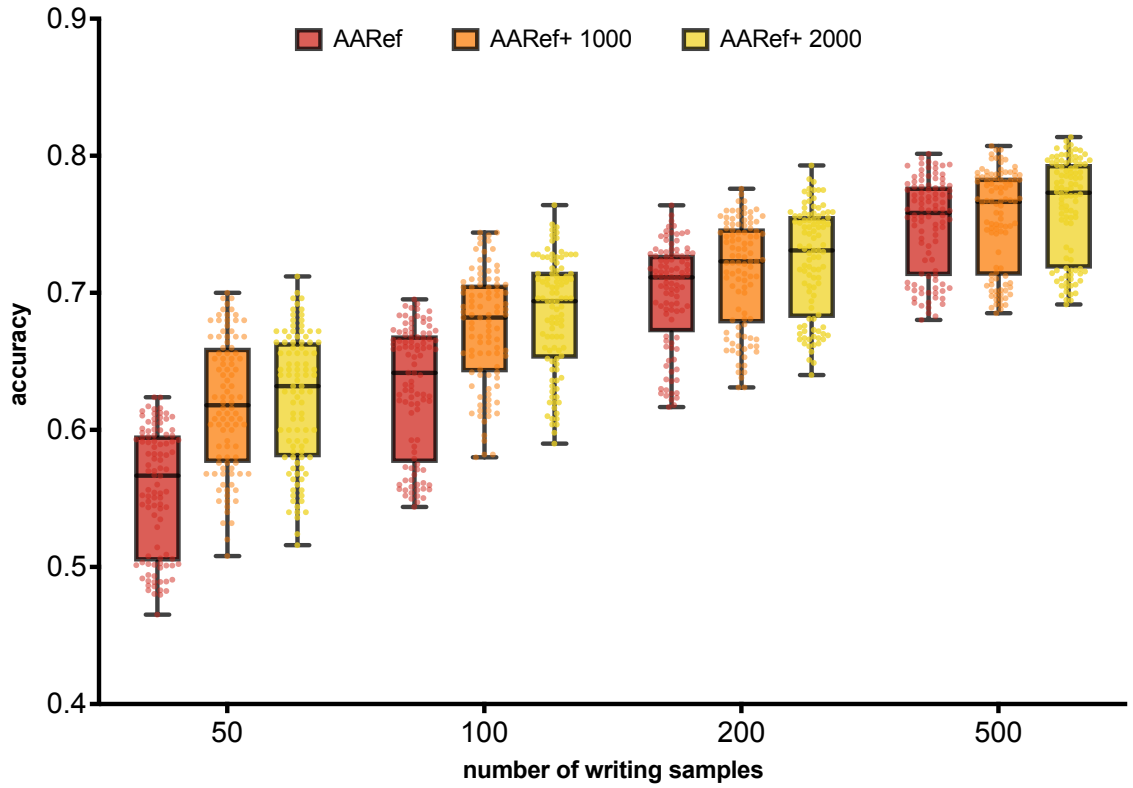

Figure 8.3: Performance visualization of AARef variations on the varying number of writing authors experiment. Each box plot is visualized by aggregating the 100 runs. Each point denotes the accuracy of a single experiment. The bottom and top of the box represent the first and third quartiles, respectively, and the line within the box represents the median. The whiskers extending from the box indicate the range of the data.

8.3 Summary

In this chapter, we investigated the anonymity of social media users using their writing styles. Our results show the capability of existing author identification systems under various conditions. Based on our findings, social media users and social media networks can use adversarial users' identities to avoid undesirable behaviors.

In future work, it would be interesting to analyze the ability to transfer authorship classifiers between different social media domains. This would allow larger datasets to train the authorship identification systems with more performant downstream datasets and less data availability. Another promising direction is incorporating various public information about the user and the post. For example, the time and the date of the post can provide helpful information about the characteristics and regular schedule of the author.

Chapter 9

Conclusion and Future Work

9.1 Conclusion

In this thesis, we investigated the machine learning methods for author identification on micro-messages. The machine learning methods are evaluated in various experimental settings on multiple real-world datasets.

In the first stage, we proposed three author identification methods. The first approach introduces an evolutionary approach to divide the task into author verification to improve the scalability with respect to the number of authors. The second approach proposes a multi-channel convolutional neural network to learn the authors' writing style with character n-gram and word unigram representations. Third, the approach proposed shared convolutional kernels and incorporated the part of speech tags.

In the second stage, a new large-scale Reddit dataset is introduced. The preprocessing pipeline, including duplicate content filtering, was implemented. With the Reddit dataset, various baselines were evaluated in two experimental settings. Proposed methods in the first stage of the dissertation were evaluated with the same experimental settings. The AARef method extensively experimented on various environments on the Twitter dataset. A task pretraining step was introduced to improve the performance in low-data environments. The models trained with task pretraining are referred to as AARef+. Furthermore, AARef+ improved the performance of AARef on smallest subset by 195%.

9.2 Future Work

- Incorporating the contextualized embeddings to AARef. Rather than using static embeddings (e.g., FastText), contextualized embeddings would allow the model to differentiate the representation of the words in different contexts.

- Investigating pretraining approaches on multiple domains. Pretraining the Author Identification models on multiple data sources to improve the generalization of the trained model. The trained model can adapt to environments with minimal data availability.

References

[1] Sarp Aykent and Gerry Dozier. Aaref: Exploiting Authorship Identifiers of Micro-Messages with Refinement Blocks. *2020 19th IEEE International Conference on Machine Learning and Applications*, pages 1044–1050, 12 2020.

[2] Sarp Aykent and Gerry Dozier. Author identification of micro-messages via multi-channel convolutional neural networks. *2020 IEEE International Conference on Systems, Man and Cybernetics*, pages 675–681, 10 2020.

[3] Douglas Bagnall. Author identification using multi-headed recurrent neural networks. In Linda Cappellato, Nicola Ferro, Gareth J. F. Jones, and Eric SanJuan, editors, *Working Notes of Conference and Labs of the Evaluation Forum 2015*, volume 1391 of *CEUR Workshop Proceedings*, 2015.

[4] Grzegorz Baron. Comparison of cross-validation and test sets approaches to evaluation of classifiers in authorship attribution domain. In *International Symposium on Computer and Information Sciences*, pages 81–89. Springer, 2016.

[5] Theodore C. Belding. The distributed genetic algorithm revisited. In *Proceedings of the 6th International Conference on Genetic Algorithms*, page 114–121, San Francisco, CA, USA, 1995.

[6] Dario Benedetto, Emanuele Caglioti, and Vittorio Loreto. Language trees and zipping. *Physical Review Letters*, 88(4):048702, 2002.

[7] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *Journal of Machine Learning Research*, 13(Feb):281–305, 2012.

[8] Donald E Brown, Christopher L Huntley, and Andrew R Spillane. A parallel genetic heuristic for the quadratic assignment problem. In *Proceedings of the 3rd International Conference on Genetic Algorithms*, pages 406–415, 1989.

[9] Ronan Collobert, Jason Weston, Léon Bottou, Michael Karlen, Koray Kavukcuoglu, and Pavel Kuksa. Natural language processing (almost) from scratch. *Journal of Machine Learning Research*, 12(Aug):2493–2537, 2011.

[10] L Davis, editor. *Handbook of Genetic Algorithms*. Van Nostrand Reinhold, 1991.

[11] Jacob Devlin, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. BERT: pre-training of deep bidirectional transformers for language understanding. *CoRR*, abs/1810.04805, 2018.

[12] Joachim Diederich, Jörg Kindermann, Edda Leopold, and Gerhard Paass. Authorship attribution with support vector machines. *Applied Intelligence*, 19(1-2):109–123, jul 2003.

[13] Christina Faust, Gerry Dozier, Jinsheng Xu, and Michael C. King. Adversarial authorship, interactive evolutionary hill-climbing, and author caat-iii. In *2017 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 1–8, 2017.

[14] Joshua Gaston, Mina Narayanan, Gerry Dozier, D. Lisa Cothran, Clarissa Arms-Chavez, Marcia Rossi, Michael C. King, and Jinsheng Xu. Authorship attribution vs. adversarial authorship from a liwc and sentiment analysis perspective. In *2018 IEEE Symposium Series on Computational Intelligence (SSCI)*, pages 920–927, 2018.

[15] Ian Goodfellow, Yoshua Bengio, and Aaron Courville. *Deep Learning*. MIT press, 2016.

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 770–778, 2016.

[17] Maryam Heidari, James H Jones, and Ozlem Uzuner. Deep contextualized word embedding for text-based online user profiling to detect social bots on twitter. In *2020 International Conference on Data Mining Workshops (ICDMW)*, pages 480–487. IEEE, 2020.

[18] Sepp Hochreiter and Jürgen Schmidhuber. Long Short-Term Memory. *Neural Computation*, 9(8):1735–1780, 11 1997.

[19] Yedid Hoshen, Ron J. Weiss, and Kevin W. Wilson. Speech acoustic modeling from raw multichannel waveforms. In *2015 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP)*, pages 4624–4628, 2015.

[20] John Houvardas and Efstathios Stamatatos. N-gram feature selection for authorship identification. In Jérôme Euzenat and John Domingue, editors, *Artificial Intelligence: Methodology, Systems, and Applications*, pages 77–86, Berlin, Heidelberg, 2006. Springer Berlin Heidelberg.

[21] Matthew L. Jockers and Daniela M. Witten. A comparative study of machine learning methods for authorship attribution. *Literary and Linguistic Computing*, 25(2):215–223, 2010.

[22] Armand Joulin, Edouard Grave, Piotr Bojanowski, and Tomas Mikolov. Bag of tricks for efficient text classification. In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 427–431, Valencia, Spain, April 2017. Association for Computational Linguistics.

[23] Patrick Juola. An overview of the traditional authorship attribution subtask. In *Proceedings of the Conference and Labs of the Evaluation Forum (Online Working Notes/Labs/Workshop)*, 2012.

[24] Nal Kalchbrenner, Edward Grefenstette, and Phil Blunsom. A convolutional neural network for modelling sentences. In *Proceedings of the 52nd Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 655–665, Baltimore, Maryland, June 2014. Association for Computational Linguistics.

[25] Vlado Kešelj, Fuchun Peng, Nick Cercone, and Calvin Thomas. N-gram-based author profiles for authorship attribution. *Pacific Association for Computational Linguistics*, pages 255–264, 2003.

[26] Mike Kestemont. Function words in authorship attribution. from black magic to theory? In *Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL)*, pages 59–66, Gothenburg, Sweden, April 2014. Association for Computational Linguistics.

[27] Iryna Khomytska and Vasyl Teslyuk. Authorship attribution by differentiation of phonostatistical structures of styles. In *2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT)*, volume 2, pages 5–8. IEEE, 2018.

[28] Yoon Kim. Convolutional neural networks for sentence classification. In *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, pages 1746–1751, 2014.

[29] Diederik P. Kingma and Jimmy Lei Ba. Adam: A method for stochastic optimization. In *3rd International Conference on Learning Representations*, pages 1–15, 2015.

[30] Günter Klambauer, Thomas Unterthiner, Andreas Mayr, and Sepp Hochreiter. Self-normalizing neural networks. In I. Guyon, U. V. Luxburg, S. Bengio, H. Wallach, R. Fergus, S. Vishwanathan, and R. Garnett, editors, *Advances in Neural Information Processing Systems*, pages 971–980. Curran Associates, Inc., 2017.

[31] Jürgen Knauth. Language-agnostic Twitter-bot detection. In *Proceedings of the International Conference on Recent Advances in Natural Language Processing (RANLP 2019)*, pages 550–558, Varna, Bulgaria, September 2019. INCOMA Ltd.

[32] Moshe Koppel, Jonathan Schler, and Shlomo Argamon. Authorship attribution in the wild. *Language Resources and Evaluation*, 45(1):83–94, 2011.

[33] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems*, pages 1097–1105, 2012.

[34] Katherine Lee, Daphne Ippolito, Andrew Nystrom, Chiyuan Zhang, Douglas Eck, Chris Callison-Burch, and Nicholas Carlini. Deduplicating Training Data Makes Language Models Better. In *Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics (Volume 1: Long Papers)*, pages 8424–8445, Dublin, Ireland, May 2022. Association for Computational Linguistics.

[35] Yinhan Liu, Myle Ott, Naman Goyal, Jingfei Du, Mandar Joshi, Danqi Chen, Omer Levy, Mike Lewis, Luke Zettlemoyer, and Veselin Stoyanov. Roberta: A robustly optimized BERT pretraining approach. *CoRR*, abs/1907.11692, 2019.

[36] Kim Luyckx. *Scalability issues in authorship attribution*. ASP/VUBPRESS/UPA, 2011.

[37] Kim Luyckx and Walter Daelemans. The effect of author set size and data size in authorship attribution. *Literary and Linguistic Computing*, 26(1):35–55, apr 2011.

[38] Andrew L Maas, Awni Y Hannun, and Andrew Y Ng. Rectifier nonlinearities improve neural network acoustic models. In *International Conference on Machine Learning Workshop on Deep Learning for Audio, Speech and Language Processing*, 2013.

[39] Laurens van der Maaten and Geoffrey Hinton. Visualizing Data using t-SNE. *Journal of Machine Learning Research*, 9(86):2579–2605, 2008.

[40] Nathan Mack, Jasmine Bowers, Henry Williams, Gerry Dozier, and Joseph Shelton. The best way to a strong defense is a strong offense: Mitigating deanonymization attacks via iterative language translation. *International Journal of Machine Learning and Computing*, 5(5):409, 2015.

[41] Tomás Mikolov, Ilya Sutskever, Kai Chen, Gregory S. Corrado, and Jeffrey Dean. Distributed representations of words and phrases and their compositionality. In Christopher J. C. Burges, Léon Bottou, Zoubin Ghahramani, and Kilian Q. Weinberger, editors, *Advances in Neural Information Processing Systems*, pages 3111–3119, 2013.

[42] Arvind Narayanan, Hristo Paskov, Neil Zhenqiang Gong, John Bethencourt, Emil Stefanov, Eui Chul Richard Shin, and Dawn Song. On the feasibility of internet-scale author identification. In *2012 IEEE Symposium on Security and Privacy*, pages 300–314, 2012.

[43] Tempestt Neal, Kalaivani Sundararajan, Aneez Fatima, Yiming Yan, Yingfei Xiang, and Damon Woodard. Surveying stylometry techniques and applications. *ACM Computing Surveys (CSUR)*, 50(6):86, 2018.

[44] Sarwat Nizamani and Nasrullah Memon. Ceai: Ccm-based email authorship identification model. *Egyptian Informatics Journal*, 14(3):239 – 249, 2013.

[45] Rebekah Overdorf and Rachel Greenstadt. Blogs, twitter feeds, and reddit comments: Cross-domain authorship attribution. *Proceedings on Privacy Enhancing Technologies*, 2016(3):155–171, 2016.

[46] Olutobi Owoputi, Brendan O'Connor, Chris Dyer, Kevin Gimpel, Nathan Schneider, and Noah A. Smith. Improved part-of-speech tagging for online conversational text with word clusters. In *Proceedings of the 2013 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 380–390, Atlanta, Georgia, June 2013. Association for Computational Linguistics.

[47] Fabian Pedregosa, Gaël Varoquaux, Alexandre Gramfort, Vincent Michel, Bertrand Thirion, Olivier Grisel, Mathieu Blondel, Peter Prettenhofer, Ron Weiss, Vincent Dubourg, et al. Scikit-learn: Machine learning in python. *Journal of Machine Learning Research*, 12:2825–2830, 2011.

[48] Shanta Phani, Shibamouli Lahiri, and Arindam Biswas. A machine learning approach for authorship attribution for Bengali blogs. In *International Conference on Asian Language Processing*, pages 271–274, 2016.

[49] Mitchell A Potter, Kenneth A De Jong, and John J Grefenstette. A Coevolutionary Approach to Learning Sequential Decision Rules. In *Proceedings of the 6th International Conference on Genetic Algorithms*, pages 366–372, 1995.

[50] Alec Radford, Jeff Wu, Rewon Child, David Luan, Dario Amodei, and Ilya Sutskever. Language models are unsupervised multitask learners. 2019.

[51] Dylan Rhodes. Author attribution with cnns. *Available online: https://www.semanticscholar.org/paper/Author-Attribution-with-Cnn-s-Rhodes/0a904f9d6b47dfc574f681f4d3b41bd840871b6f/pdf (accessed on 22 August 2016)*, 2015.

[52] Anderson Rocha, Walter J Scheirer, Christopher W Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne RB Carvalho, and Efstathios Stamatatos. Authorship attribution for social media forensics. *IEEE Transactions on Information Forensics and Security*, 12(1):5–33, 2016.

[53] Sebastian Ruder, Parsa Ghaffari, and John G Breslin. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv preprint arXiv:1609.06686*, 2016.

[54] Claude Sammut and Geoffrey I. Webb, editors. *TF–IDF*, pages 986–987. Springer US, Boston, MA, 2010.

[55] Upendra Sapkota, Steven Bethard, Manuel Montes, and Thamar Solorio. Not all character n-grams are created equal: A study in authorship attribution. In *Proceedings of the 2015 Conference of the North American Chapter of the Association for Computational Linguistics: Human Language Technologies*, pages 93–102, 2015.

[56] Roy Schwartz, Oren Tsur, Ari Rappoport, and Moshe Koppel. Authorship attribution of micro-messages. In *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing*, pages 1880–1891, Seattle, Washington, USA, October 2013. Association for Computational Linguistics.

[57] Prasha Shrestha, Sebastian Sierra, Fabio Gonzalez, Manuel Montes, Paolo Rosso, and Thamar Solorio. Convolutional neural networks for authorship attribution of short texts.

In *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers*, pages 669–674, 2017.

[58] Nitish Srivastava, Geoffrey Hinton, Alex Krizhevsky, Ilya Sutskever, and Ruslan Salakhutdinov. Dropout: A simple way to prevent neural networks from overfitting. *Journal of Machine Learning Research*, 15(56):1929–1958, 2014.

[59] Efstathios Stamatatos. Author identification using imbalanced and limited training texts. In *18th International Workshop on Database and Expert Systems Applications*, pages 237–241. IEEE, 2007.

[60] Efstathios Stamatatos. A survey of modern authorship attribution methods. *Journal of the American Society for information Science and Technology*, 60(3):538–556, 2009.

[61] Reiko Tanese. *Distributed genetic algorithms for function optimization*. PhD thesis, University of Michigan, 1989.

[62] William J. Teahan and David J. Harper. Using Compression-Based Language Models for Text Categorization. In *Language Modeling for Information Retrieval*, pages 141–165. Springer, Dordrecht, 2003.

[63] Antônio Theóphilo, Luís AM Pereira, and Anderson Rocha. A needle in a haystack? harnessing onomatopoeia and user-specific stylometrics for authorship attribution of micromessages. In *IEEE International Conference on Acoustics, Speech and Signal Processing*, pages 2692–2696. IEEE, 2019.

[64] Iulia Turc, Ming-Wei Chang, Kenton Lee, and Kristina Toutanova. Well-read students learn better: On the importance of pre-training compact models. *arXiv preprint arXiv:1908.08962*, 2019.

[65] Jianxin Wu. Power mean SVM for large scale visual classification. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2344–2351. IEEE, 2012.

[66] Zhilin Yang, Zihang Dai, Yiming Yang, Jaime G. Carbonell, Ruslan Salakhutdinov, and Quoc V. Le. Xlnet: Generalized autoregressive pretraining for language understanding. *CoRR*, abs/1906.08237, 2019.

[67] Xiang Zhang, Junbo Jake Zhao, and Yann LeCun. Character-level convolutional networks for text classification. In Corinna Cortes, Neil D. Lawrence, Daniel D. Lee, Masashi Sugiyama, and Roman Garnett, editors, *Advances in Neural Information Processing Systems*, pages 649–657, 2015.