

Authorship Attribution of Articles Using Linguistic Analysis

*Project report submitted
in partial fulfilment of the requirement for the degree of*

Bachelor of Engineering in Information Technology

By

Sumanta Kumar Paul (002011001063)

Sayan Maji (002011001079)

Soumyadeep Pal (002011001113)

Under the guidance of

Prof. Uttam Kumar Roy



**Department of Information Technology,
Faculty of Engineering and Technology,
Jadavpur University, Salt Lake Campus
2020 – 2024**

This page has been intentionally left blank

Department of Information Technology
Faculty of Engineering and Technology
Jadavpur University

BONAFIDE CERTIFICATE

This is to certify that this project report entitled “*Authorship Attribution of Articles using Linguistic Analysis*” submitted to **Department of Information Technology, Jadavpur University, Salt Lake Campus, Kolkata**, is a bonafide record of work done by **Sumanta Kumar Paul** (Regn. No.: 153790 of 2020-2021), **Sayan Maji** (Regn. No.: 153796 of 2020-2021) and **Soumyadeep Pal** (Regn. No.: 153815 of 2020-2021) under my supervision from 03/07/2023 to 15/03/2024.

Prof. Uttam Kumar Roy
Professor, Information Technology

Countersigned By:

Prof. Bibhas Chandra Dhara
HOD, Information Technology

Declaration

We, Sumanta Kumar Paul, Sayan Maji and Soumyadeep Pal, hereby declare that this project report titled “*Authorship Attribution of Articles using Linguistic Analysis*” contains only the work completed by us as a part of the Bachelor of Engineering course, during the year 2023-2024, under the supervision of Prof. Uttam Kumar Roy, Department of Information Technology, Jadavpur University, Kolkata, India.

It has not been submitted to this or any other university for a degree or professional qualification, in whole or in part.

All information, materials and methods that are not original to this work have been properly referenced and cited. All information in this document have been obtained and presented in accordance with academic rules and ethical conduct.

Acknowledgements

We would like to express our sincere gratitude to Prof. Uttam Kumar Roy for allowing us to pursue our final-year project under his supervision. In addition to his meticulous research guidance, we would like to acknowledge his encouraging support for our research motivations as well as the constructive criticisms during our experimental designs, drafting of our manuscripts, etc. without which this project would not have been possible.

We would also like to thank our parents for their support and encouragement in our academic pursuits.

**Department of Information Technology
Jadavpur University
Salt Lake Campus, Kolkata**

Sumanta Kumar Paul

Sayan Maji

Soumyadeep Pal



JADAVPUR UNIVERSITY

Dept. of Information Technology

Vision:

To provide young undergraduate and postgraduate students a responsive research environment and quality education in Information Technology to contribute in education, industry and society at large.

Mission:

M1: To nurture and strengthen professional potential of undergraduate and postgraduate students to the highest level.

M2: To provide international standard infrastructure for quality teaching, research and development in Information Technology.

M3: To undertake research challenges to explore new vistas of Information and Communication Technology for sustainable development in a value-based society.

M4: To encourage teamwork for undertaking real life and global challenges.

Program Educational Objectives (PEOs):

Graduates should be able to:

PEO1: Demonstrate recognizable expertise to solve problems in the analysis, design, implementation and evaluation of smart, distributed, and secured software systems.

PEO2: Engage in the engineering profession globally, by contributing to the ethical, competent, and creative practice of theoretical and practical aspects of intelligent data engineering.

PEO3: Exhibit sustained learning capability and ability to adapt to a constantly changing field of Information Technology through professional development, and self-learning.

PEO4: Show leadership qualities and initiative to ethically advance professional and organizational goals through collaboration with others of diverse interdisciplinary backgrounds.

Mission - PEO matrix:

Ms/ PEOs	M1	M2	M3	M4
PEO1	3	2	2	1
PEO2	2	3	2	1
PEO3	2	2	3	1
PEO4	1	2	2	3

(3 – Strong, 2 – Moderate and 1 – Weak)

Program Specific Outcomes (PSOs):

At the end of the program a student will be able to:

PSO1: Apply the principles of theoretical and practical aspects of ever evolving Programming & Software Technology in solving real life problems efficiently.

PSO2: Develop secured software systems considering constantly changing paradigms of communication and computation of web enabled distributed Systems.

PSO3: Design ethical solutions of global challenges by applying intelligent data science & management techniques on suitable modern computational platforms through interdisciplinary collaboration.

This page has been intentionally left blank

Abstract

The absence of reliable machine learning models to predict authors of anonymous text documents like threat letters, ransom notes and hoax messages from unidentified sources with sufficient accuracy is a major issue that limits the scope of linguistic analysis in criminal investigations by law enforcement agencies. Existing approaches like comparison of stylometric features such as lexicology, syntax, structure and idiosyncrasy across often prove to be dissatisfactory in formally written documents with similar structural formats and minimal grammatical or spelling errors. In critical domains such as investigations, the assumption on availability of prominently visible identifying features in text documents is unrealistic.

Our project explores the applicability of supervised machine learning methods in addressing these challenges of authorship attribution. Specifically, we propose two chapters (Chapters 3 and 4 of the report), to suggest a reliable model to address and overcome the aforementioned limitations.

Our first chapter is an exploration of the effectiveness of supervised machine learning methods to perform authorship attribution using a variety of linguistic analysis techniques. We seek to investigate the applicability of some supervised machine learning algorithms on text datasets of different scales. In particular, we evaluate them on two publicly available blog article corpora.

In our second chapter, we extend our exploration to the usage of global vector word embeddings as a linguistic feature. We systematically explore the applicability of multi-channel convolutional neural networks, conducting thorough experimentation of the proposed model across varying linguistic feature sets, datasets and dataset sizes and evaluation of its performance compared to other supervised machine learning algorithms.

Keywords: Authorship attribution; linguistic analysis; supervised machine learning; multi-channel convolutional neural networks; word embeddings

Contents

Acknowledgements	iii
Abstract	vi
1 Introduction	1
1.1 Background	1
1.2 Report outline	2
2 Literature Review	4
2.1 Linguistic Analysis	4
2.2 Authorship Attribution	7
2.3 Supervised Learning	7
2.4 Linguistic Feature Sets for Machine Learning	8
2.5 Multi-channel Convolutional Neural Networks	9
3 Analysis of the Applicability of Supervised Learning for Authorship Attribution using Linguistic Analysis	11
3.1 Overview	12
3.2 Materials and Methods	15
3.2.1 Supervised Learning Algorithms	15
3.2.2 Datasets	18
3.3 Experimental Setup	19
3.3.1 Implementation details	19
3.3.2 Evaluation	25
3.4 Conclusion	27

4	Evaluation of Multi-channel Convolutional Neural Networks for Word Embedding based Linguistic Analysis	28
4.1	Overview	29
4.2	Multi-channel Convolutional Neural Networks	30
	4.2.1 Global Vector Word Embeddings	30
	4.2.2 Multi-channel CNN Algorithms	32
4.3	Experimental Protocol	33
	4.3.1 Datasets	34
	4.3.2 Configuration and Implementation	35
	4.3.3 Evaluation metrics	45
4.4	Results and Discussion	45
	4.4.1 Across varying dataset sizes	46
	4.4.2 Further analysis	47
4.5	Conclusion	47
5	Conclusions and Future Work	48
5.1	Conclusions	48
5.2	Future research directions	48
	References	51

List of Figures

4.1	Contents of the downloadable GloVe file “glove.840B.300d.txt” . .	30
4.2	Representation of words as word embeddings and their visualization in a 2D space	31
4.3	Model diagram of proposed multi-channel CNN model	33

List of Tables

3.1	Optimal linguistic feature sets and test accuracies for authorship attribution using different machine learning models	26
4.1	Layers in multi-channel CNN model with output shapes, number of parameters, input channels and output channels (for implementation using Blog Authorship Dataset with 5 authors)	32
4.2	Dataset sizes across different number of authors in the Blog Authorship Corpus	46
4.3	Test accuracy values and approximate execution times across Different number of authors in the Blog Authorship Corpus	46

Chapter 1

Introduction

1.1 Background

The past decade has seen an increase in the frequency of anonymous threat letters to eminent personalities and institutions, hoax or prank letters to public facilities like airlines and banks as well as targeted misinformation and disinformation activities on online blogs and social media platforms. In situations such as these, there is often no evidence other than the content of the text document itself that can identify the sender or narrow down the list of suspects, which can lead to inefficient investigations based on trial and error.

However, the content of the text document itself can provide a sufficient amount of hints or clues to identify demographic information about its author such as age group, gender, education level and domain, area of work, locations of residence, community, etc. as well as other information like time and purpose of writing the document. Comparison of these attributes with existing databases of content written by people from different demographic criteria and can help narrow down the list of potential authors. Comparison with content written by a list of suspected authors can also help identify the author with an acceptance degree of accuracy by analysing writing patterns such as lexical, syntactic, structural, content-specific, idiosyncratic features, word frequencies, word pairs and other stylometric and grammatical feature sets.

It is important to note that this approach, commonly known as linguistic analysis or specifically forensic linguistics in investigations agencies, has

already been used to identify perpetrators in the past. The most popular example, perhaps, would be the identification and capture of Ted Kaczynski by the United States Federal Bureau of Investigation (F.B.I.) where manual linguistic analysis was used to identify the demographic attributes of author of the threat letters and eventually the author by comparing with previous letters.

In recent times, computational approaches have emerged to identify authors of anonymous documents using linguistic analysis techniques. Most of the early approaches relied on simple mathematical methods like comparing term frequencies and inverse document frequencies of words. Machine learning algorithms have been used to compare more complex attributes like stylometric and grammatical features. Supervised machine learning is used to train the model using other text content written by likely authors of the anonymous document and then help attribute it to one of these authors.

Some machine learning algorithms used for this task are logistic regression, principal component analysis, random forest, support vector machine, k-nearest neighbour, Naïve-Bayes classifiers, feed forward neural networks and convolutional neural networks. We investigate the effectiveness of the aforementioned supervised machine learning algorithms and suggest a multi-channel convolutional neural network model capable of more accurate authorship attribution using linguistic analysis.

1.2 Report outline

The report is organized as follows:

Chapter 2 presents the existing literature on linguistic analysis and authorship attribution. We also provide a background on the application of supervised learning algorithms along with linguistic feature sets primarily used for authorship attribution. We briefly summarise the applicability of multi-channel convolutional neural networks in this field which relates to our work.

Chapter 3 explores the applicability of supervised machine learning algorithms on two popular authorship attribution datasets. We analyse the performance of the methods across various linguistic feature sets and make further comparison among them along the lines of effectiveness and efficiency.

Chapter 4 combines the findings of the previously explored supervised learning algorithms and linguistic feature sets and explores the applicability of multi-channel convolutional neural networks with word embeddings as a linguistic feature. We also evaluate the performance of this approach across different datasets and dataset sizes, and perform further analysis over the relative performances of different approaches over the two datasets.

Chapter 5 provides a summary of the research outputs through a conclusion and discusses various potential future research directions that could be carried out as further works.

Chapter 2

Literature Review

This chapter starts by briefly covering the relevant works in linguistic analysis, gradually moving on to its application in authorship attribution, with a review of its literature as well. This is followed by shifting the focus to the paradigm of supervised machine learning and a discussion of the literature in this domain. We then move on to the overview of linguistic feature sets used in machine learning models, following up with an explanation of multi-channel convolutional neural networks. To keep it concise, we specifically focus on the relevant literature on the methods evaluated in this report.

2.1 Linguistic Analysis

Linguistic analysis is the systematic study of language to understand its structure, meaning, and use. It involves breaking down language into its fundamental components to explore how it functions and conveys meaning. This process is essential for a variety of applications including computational linguistics, text mining, natural language processing (NLP), and literary analysis. Key aspects of linguistic analysis in text documents include lexical analysis, syntactic analysis, semantic analysis, structural analysis, pragmatic analysis, discourse analysis, sentiment analysis, morphological analysis, idiosyncratic analysis, stylometric analysis, sociolinguistic or demographic analysis, and stylistic analysis. Combinations of these techniques can be used for performing authorship attribution of text documents.

- 2.1.1. **Lexical analysis** involves the study and processing of words in a text. This includes identifying individual words (tokenization), categorizing them (part-of-speech tagging), and understanding their meanings (lemmatization and stemming). It is fundamental for tasks like spell checking, keyword extraction, and search engine optimization.
- 2.1.2. **Syntactic analysis**, also known as parsing, examines the grammatical structure of sentences. It identifies the syntactic relationships between words and phrases, ensuring sentences are grammatically correct. This involves creating parse trees or dependency graphs, which are crucial for understanding sentence structure and meaning.
- 2.1.3. **Semantic analysis** focuses on understanding the meaning of words, phrases, and sentences. It involves interpreting the meanings of words in context, resolving ambiguities (word sense disambiguation), and identifying relationships between concepts. This is essential for tasks like machine translation and information retrieval.
- 2.1.4. **Structural analysis** examines the overall organization and arrangement of text. It looks at how sentences and paragraphs are constructed and how they contribute to the text's coherence and meaning. This includes analyzing the use of headings, subheadings, and other structural elements that guide the reader's understanding.
- 2.1.5. **Pragmatic analysis** goes beyond literal meaning to consider the context in which language is used. It involves understanding the speaker's intent, the relationship between speakers, and the situational context. This analysis is vital for interpreting implied meanings, politeness strategies, and speech acts.
- 2.1.6. **Discourse analysis** studies larger units of text, such as conversations, paragraphs, or entire documents, to understand how they cohere and convey meaning. It examines the structure and flow of

communication, including the roles of coherence, cohesion, and thematic progression. This analysis helps in understanding narrative structure and argumentation.

- 2.1.7. **Sentiment analysis**, also known as opinion mining, determines the emotional tone of a text. It classifies text as positive, negative, or neutral, and can even detect specific emotions like joy, anger, or sadness. This is widely used in social media monitoring, customer feedback analysis, and market research.
- 2.1.8. **Morphological analysis** studies the structure of words and how they are formed from morphemes, the smallest units of meaning. It involves identifying root words, prefixes, suffixes, and inflections. This analysis is crucial for understanding word formation, derivation, and inflection in different languages.
- 2.1.9. **Idiosyncratic analysis** examines unique language use and individual variations in language. This includes analyzing specific word choices, stylistic preferences, and personalized expressions. It is useful in authorship attribution, forensic linguistics, and understanding personal writing styles.
- 2.1.10. **Stylometric analysis** quantitatively examines linguistic features such as word frequency, sentence length, and syntax patterns to identify unique writing styles. It employs statistical and computational methods, aiding in authorship attribution, plagiarism detection, and textual analysis by uncovering distinctive stylistic signatures.
- 2.1.11. **Sociolinguistic analysis**, also known as demographic analysis, examines how language varies across different social groups and contexts. It studies factors such as regional dialects, sociolects, language change, and the influence of social variables like age, gender, ethnicity, and socioeconomic status on language use. This

analysis helps in understanding language diversity and social dynamics in communication.

- 2.1.12. Stylistic analysis** qualitatively explores an author's unique language choices, including vocabulary, sentence structure, and literary devices. It aims to understand the aesthetic and rhetorical effects of these choices, providing insights into the author's voice, intent, and the text's emotional and thematic impact.

2.2 Authorship Attribution

Authorship attribution is the process of identifying the author of a text by analyzing its linguistic characteristics. It combines techniques from linguistics, statistics, and computer science to discern unique writing styles and patterns that can distinguish one author from another. This technique is crucial in fields like literary studies, forensic linguistics, and digital humanities. It involves examining stylistic features such as vocabulary usage, sentence structure, and punctuation patterns, using both qualitative and quantitative methods. Key approaches include statistical analysis of linguistic features, and machine learning techniques that classify texts based on these features. Applications range from resolving historical authorship disputes and detecting plagiarism to verifying the authenticity of documents in legal contexts. Advances in computational tools have significantly enhanced the accuracy and reliability of authorship attribution, making it an invaluable tool for textual analysis.

2.3 Supervised Learning

Machine learning is a subfield of artificial intelligence that focuses on the development of algorithms and statistical models that enable computer systems to learn from and make predictions or decisions based on data. Instead of relying on explicit programming, machine learning systems use data to identify patterns, make predictions and improve their performance over time

through experience. It encompasses various methods such as supervised learning, unsupervised learning, and reinforcement learning.

Supervised machine learning involves training algorithms on labeled datasets, where each example has an associated output label. During training, the algorithm learns to map inputs to outputs by minimizing the error between its predictions and the actual labels. Common tasks include classification, where the algorithm predicts discrete categories, and regression, where it predicts continuous values. Supervised learning finds applications in diverse fields, including image and speech recognition, medical diagnosis, and recommendation systems. However, its effectiveness depends on the quality and quantity of labeled data, and the choice of appropriate algorithms and model architectures tailored to specific tasks.

2.4 Linguistic Feature Sets for Machine Learning

Some linguistic feature sets commonly used for authorship attribution by machine learning algorithms include word frequencies like term frequency and inverse document frequency, tag frequencies like part of speech, function and content word frequencies, N-grams like character N-grams and word N-grams, diversity metrics like vocabulary richness, readability metrics like text complexity and topic modelling features.

- 2.4.1. **Term frequency and inverse document frequency** measure how often a word occurs relative to the total number of words in that document and how often a document contains the word relative to the total number of documents. Their product evaluates the importance of the word in a particular document.
- 2.4.2. **Part of speech, function and content word frequencies** analyse the frequency distribution of parts of speech like pronouns and adverbs, function words like articles and prepositions, and content words like nouns and verbs to convey grammatical and thematic relations.

- 2.4.3. **Character and word N-grams** are sequential patterns of N characters or words extracted from a text. Character N-grams capture spelling variations and linguistic patterns, while word N-grams represent word sequences, aiding in tasks like text classification, authorship attribution, and language modelling.
- 2.4.4. **Vocabulary richness** measured by the type-token ratio, assesses the diversity of words in a text. It calculates the ratio of unique words (types) to the total number of words (tokens), providing insights into lexical variety and linguistic complexity.
- 2.4.5. **Text complexity** measured by the Flesch-Kincaid readability test evaluates the complexity of a text based on its average sentence length and average syllable count per word. It produces a numerical score corresponding to the approximate grade level required to comprehend the text, aiding in assessing readability and comprehension difficulty.
- 2.4.6. **Topic modelling** is a statistical technique for identifying latent themes or topics within a collection of documents. It uncovers underlying patterns in text data by clustering words that frequently co-occur, enabling exploratory analysis and content recommendation.

2.5 Multi-channel Convolutional Neural Networks

Neural networks are computational models inspired by the human brain's structure and function. They consist of interconnected nodes organized in layers. Each node applies a transformation to its input and passes the result to the next layer. Through training with labeled data, neural networks learn to recognize patterns and make predictions. Deep neural networks, with many layers, are particularly powerful for complex tasks like image and speech recognition, natural language processing, and reinforcement learning.

Convolutional neural networks are a specialized type of neural network designed for processing grid-structured data such as images. They use convolutional layers to automatically extract relevant features from input images, followed by pooling layers to reduce dimensionality and increase computational efficiency. CNNs are widely used in tasks like image classification, object detection, and image segmentation due to their ability to capture spatial hierarchies of features.

Multi-channel convolutional neural networks extend the capabilities of traditional CNNs by processing multiple input channels simultaneously. Each input channel represents different information or modalities, allowing the network to integrate diverse sources of data. This approach is beneficial for tasks like multi-sensor data fusion, where information from various sensors or data streams needs to be combined for analysis. Multi-channel CNNs are employed in applications such as multimodal learning, medical imaging, and sensor fusion in autonomous systems.

Chapter 3

Analysis of the Applicability of Supervised Learning for Authorship Attribution using Linguistic Analysis

Content

3.1	Overview	12
3.2	Materials and Methods	15
	3.2.1 Supervised Learning Algorithms	15
	3.2.2 Datasets	18
3.3	Experimental Setup	19
	3.3.1 Implementation details	19
	3.3.2 Evaluation	25
3.4	Conclusion	27

3.1 Overview

Over the past few decades machine learning algorithms like K-Nearest Neighbour (KNN), Support Vector Machine (SVM), Author's Multilevel N-gram Profile (AMNP), Logistic Regression, Principal component analysis and cluster analysis (PCA) & Random Forest (RF), Naïve-Bayes have been used in the field of Authorship Attribution and linguistic analysis. These have been used in various fields such as text forensics, email forensics, social media forensics, source code attribution, automatic recommendation for criminal responsibility, historical and authorial studies, news and journalism, attribution of anonymous works, etc.

From the various research references in the above practical example we can conclude that one machine learning algorithm cannot be the most suitable for all situations – every problem will have unique characteristics depending upon which we have to choose the machine learning algorithm. Here we will be looking at the authorship attribution algorithm for news articles and blogs.

NP-completeness is a concept in computational complexity theory that identifies a class of decision problems for which solutions can be verified quickly but for which there is no known efficient algorithm to find a solution. A problem is considered NP-complete if any problem in the class NP can be reduced to it in polynomial time, implying that if an efficient algorithm exists for any NP-complete problem, it can be used to solve all problems in NP efficiently. NP-complete problems are some of the most challenging computational problems, and while solutions to individual instances can be verified efficiently, finding solutions for large instances typically requires exhaustive search or heuristic approaches, making them computationally intractable for practical purposes.

Authorship attribution is not an NP-complete problem. While it involves analysing large sets of data and determining the likelihood of various authors being responsible for a given text, it does not require exhaustive search over all possible combinations of authors and texts, which is a characteristic of NP-complete problems. Instead, authorship attribution typically relies on statistical

and machine learning techniques to identify patterns and similarities in language use, making it computationally tractable within reasonable timeframes. Additionally, the problem can often be formulated as a supervised learning task, where models are trained on labelled datasets of known authorship, further aiding in efficiency and scalability. Therefore, authorship attribution does not belong to the class of NP-complete problems. Thus, we will be carrying out probabilistic and statistical approach instead of deterministic approach.

A probabilistic approach involves representing uncertainty in data using probability distributions and employing probabilistic models to make predictions or decisions. In this approach, uncertainty is quantified using probabilities, and predictions are made based on the likelihood of different outcomes given observed data. Probabilistic methods often involve techniques such as Bayesian inference, where prior knowledge and observed data are combined to update beliefs about uncertain quantities.

On the other hand, a statistical approach focuses on analyzing data to uncover patterns, relationships, and trends using statistical methods and models. This approach involves collecting and summarizing data, performing hypothesis tests, estimating parameters of interest, and making predictions based on statistical models. Statistical methods can include techniques such as regression analysis, hypothesis testing, and analysis of variance.

We will check now which statistical machine learning algorithm fits based for the news article and blog detection dataset. Multichannel Convolutional Neural Networks (CNNs) are often preferred over other machine learning algorithms for authorship attribution due to several key advantages:

- 1. Automatic Feature Extraction:** CNNs can automatically learn hierarchical representations of input data, eliminating the need for manual feature engineering. In authorship attribution, linguistic features can be extracted directly from textual data without the need for extensive preprocessing.

2. **Capture Local Dependencies:** CNNs are well-suited for capturing local dependencies in sequential data, such as the syntactic and semantic structure of text. By applying convolutional filters across input sequences, CNNs can detect patterns and motifs that are indicative of an author's writing style.
3. **Multichannel Architecture:** Multichannel CNNs leverage multiple input channels to incorporate different types of information simultaneously. In authorship attribution, this can include textual features, metadata (e.g., document length, publication date), and other contextual information. By integrating diverse sources of data, multichannel CNNs can improve model performance and robustness.
4. **Parameter Sharing and Weight Sharing:** CNNs exploit parameter sharing and weight sharing to efficiently learn representations from data, reducing the risk of overfitting, especially in cases of limited training data. This allows CNNs to generalize well to unseen texts and authors.
5. **Scalability and Parallelization:** CNNs can be highly scalable and amenable to parallelization, making them suitable for processing large volumes of text data efficiently. This scalability enables the training of deep CNN architectures on extensive corpora, further enhancing model performance.
6. **End-to-End Learning:** Multichannel CNNs enable end-to-end learning, where the model learns to perform authorship attribution directly from raw text data without the need for intermediate processing steps. This streamlined approach simplifies the modelling pipeline and can lead to more accurate and interpretable results.

Now we will verify our hypothesis by implementing the suitable machine learning algorithms as mentioned in the first paragraph on the Reuters dataset as elaborated in 3.2.2.

3.2 Materials and Methods

In this section, we explore the applicability of the aforementioned supervised machine learning algorithms and move to a description of the two datasets used to evaluate the performances of the machine learning models.

3.2.1 Supervised Learning Algorithms

We discuss the applicability of logistic regression, principal component analysis and random forest, support vector machine, K-nearest neighbour, author's multilevel N-gram profile, Naïve-Bayes, and convolutional neural networks in authorship attribution with a brief description of these models.

(1) Logistic Regression is a linear classification algorithm that models the probability of a binary outcome. It estimates the probability that a given input belongs to a particular class based on its features. logistic regression works by fitting a logistic function to the data, which maps input features to the probability of the input belonging to a particular class.

In authorship attribution, logistic regression can be utilized to classify texts into different author categories based on linguistic features extracted from the texts. By learning the relationship between linguistic patterns and authorship, logistic regression can provide probabilistic predictions about the likelihood of a text belonging to each author.

(2) Principal Component Analysis and Random Forest combines (a) principal component analysis, a dimensionality reduction technique that transforms high-dimensional data into a lower-dimensional space while preserving most of its variance, and accomplishes this by identifying the principal components, which are linear combinations of the original features that capture the most variation in the data, and (b) random forest, an ensemble learning method that builds multiple decision trees and combines their predictions to improve accuracy and robustness. Each tree

in the forest is trained on a random subset of the data and features, and the final prediction is done by aggregating the predictions of individual trees.

In authorship attribution, principal component analysis can be applied to reduce the dimensionality of linguistic features extracted from texts, making them more manageable for subsequent analysis. Random forest can then be trained on these reduced features to classify texts into different author categories. Random forest is particularly effective for authorship attribution tasks due to its ability to handle high-dimensional data and capture complex relationships between features.

- (3) **Support Vector Machine (SVM)** is a powerful supervised learning algorithm used for classification and regression tasks. It finds the hyperplane that best separates different classes in the feature space by maximizing the margin between classes. SVM can handle both linearly separable and non-linearly separable data by using different kernel functions to map the input data into higher-dimensional feature spaces.

In authorship attribution, SVM can be trained on linguistic features extracted from texts to classify them into different author categories. By finding the optimal hyperplane that separates texts authored by different authors, SVM can effectively distinguish between different writing styles and assign texts to their respective authors.

- (4) **K-Nearest Neighbours (KNN)** is a simple and intuitive classification algorithm that classifies data points based on the majority class among their k nearest neighbours in the feature space. KNN does not require explicit model training; instead, it stores all training data points and classifies new data points based on their proximity to existing data points.

In authorship attribution, KNN can be trained on linguistic features extracted from texts and classify them based on similarities to known authors. In authorship attribution, KNN can be particularly useful when there are clear clusters of data belonging to different authors, as it relies on the local structure of the data to make predictions.

- (5) **Author's Multilevel N-gram Profile (AMNP)** is a feature representation technique used in authorship attribution. It involves creating profiles of authors based on the frequencies of different n-grams (contiguous sequences of n items, typically words or characters) in their texts. These profiles capture the unique writing style and linguistic patterns of each author, which can be used as features for authorship attribution models.

In authorship attribution, AMNP is used to represent texts as high-dimensional feature vectors encoding the frequencies of various n-grams. These feature vectors are then fed into machine learning algorithms to classify texts into different author categories. By capturing the distinctive writing styles of authors, AMNP-based features can improve the accuracy and reliability of authorship attribution models.

- (6) **Naïve-Bayes** is a simple yet powerful probabilistic classifier based on Bayes' theorem with strong independence assumptions between features. Despite its simplicity, Naïve-Bayes often performs well in practice and is computationally efficient.

In authorship attribution, Naïve-Bayes can be used to classify texts into different author categories based on the probability of observing certain linguistic features given each author. By learning the conditional probability distribution of linguistic features for each author, Naïve-Bayes can make probabilistic predictions about the likelihood of a text belonging to each author.

- (7) **Convolutional Neural Network (CNN)** is a deep learning architecture commonly used for image recognition, but it can also be applied to sequential data like text. CNNs consist of multiple layers of convolutional and pooling operations, followed by fully connected layers for classification. CNNs can automatically learn hierarchical representations of data by applying convolutional filters across input sequences, capturing local dependencies and patterns.

In authorship attribution, CNNs can automatically learn hierarchical representations of linguistic features from texts, capturing local

dependencies and patterns indicative of an author's writing style. By processing texts at multiple levels of abstraction, CNNs can effectively extract informative features for authorship attribution tasks. Multichannel CNNs can further incorporate additional contextual information, such as metadata or linguistic features, for improved performance in authorship attribution.

3.2.2 Datasets

Two datasets, the Reuters Corpus and the Blog Authorship Corpus, have been used for evaluating the performance of supervised machine learning models in authorship attribution. We briefly describe their characteristic aspects.

(1) The Reuters Corpus is a collection of documents that appeared on Reuters newswire in 1987. The documents were assembled and indexed with categories. The original corpus has 10,369 documents and a vocabulary of 29,930 words. This collection of news articles is widely used in natural language processing and text classification tasks. It consists of documents covering various topics, each labelled with one or more categories. Researchers use it to develop and evaluate machine learning models for tasks like text categorization and information retrieval. The dataset is accessible through the Natural Language Toolkit (NLTK) and serves as a benchmark for NLP research. It is a multi-class, multi-label dataset. It has 90 classes, 7769 training documents and 3019 testing documents. It is a subset of the Reuters-21578 benchmark.

The Reuters Corpus can be downloaded from the URL:

<https://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>

(2) The Blog Authorship Corpus consists of the collected posts of 19,320 bloggers gathered from blogger.com in August 2004. The corpus incorporates a total of 681,288 posts and over 140 million words - or approximately 35 posts and 7250 words per person. Each blog is presented as a separate file, the name of which indicates a blogger id and the

blogger's self-provided gender, age, industry and astrological sign. For each age group there are an equal number of male and female bloggers.

The Blog Authorship Corpus can be downloaded from the URL:

<https://u.cs.biu.ac.il/~koppel/BlogCorpus.htm>

3.3 Experimental Setup


3.3.1 Implementation details

- (1) **Logistic Regression** is implemented by vectorizing the text using a simple bag of words approach with fewer features, initializing a count vectorizer with a specified maximum number of features, and converting the training and testing text data to bag of words vectors using the vectorizer.




```
vectorizer = CountVectorizer(max_features=1000)
X_train_vectorized = vectorizer.fit_transform(X_train)
X_test_vectorized = vectorizer.transform(X_test)
```

- (2) **Principal Component Analysis and Random Forest** is implemented by creating a list of documents and their corresponding categories, using TF-IDF vectorization to convert the text data into numerical features, setting `max_features` parameter to 5000 to limit the maximum number of features, applying PCA to reduce the dimensionality of the TF-IDF vectors, and initializing and training a Random Forest classifier with 100 trees.



```
documents = [" ".join(reuters.words(doc_id)) for doc_id in
               docs if set(reuters.categories(doc_id)) <=
               set(common_categories)]
labels = [reuters.categories(doc_id)[0] for doc_id in docs if
           set(reuters.categories(doc_id)) <= set(common_categories)]
tfidf_vectorizer = TfidfVectorizer(max_features=5000)
X_train_tfidf = tfidf_vectorizer.fit_transform(X_train)
X_test_tfidf = tfidf_vectorizer.transform(X_test)
num_components = 10
pca = PCA(n_components=num_components)
X_train_pca = pca.fit_transform(X_train_tfidf.toarray())
X_test_pca = pca.transform(X_test_tfidf.toarray())
clf = RandomForestClassifier(n_estimators=100,
                             random_state=42)
clf.fit(X_train_pca, y_train)
```

- (3) **Support Vector Machine (SVM)** is implemented by converting text data to TF-IDF features, initializing a vectorizer with parameters for feature extraction (maximum features, n-gram range and stop words), converting the training and testing text data to TF-IDF features, increasing the parameter grid, setting a parameter grid for hyperparameter tuning using grid search, increasing the number of cross-validation folds, initializes a GridSearchCV object with the Support Vector Classification (SVC) model, parameter grid, and cross-validation parameters, performing grid search to find the best hyperparameters, and training the SVM model with the best hyperparameters.



```
vectorizer = TfidfVectorizer(max_features=5000, ngram_range=
                             (1,3), stop_words='english')
X_train_tfidf = vectorizer.fit_transform(X_train)
X_test_tfidf = vectorizer.transform(X_test)
```




```
param_grid = {'C': [0.1, 1, 10], 'gamma': [1, 0.1, 0.01],  
              'kernel': ['rbf', 'poly']}  
grid_search = GridSearchCV(SVC(), param_grid, cv=5, n_jobs=2)  
grid_search.fit(X_train_tfidf, y_train)  
best_params = grid_search.best_params_  
svm_model = SVC(**best_params)  
svm_model.fit(X_train_tfidf, y_train)
```

- (4) **K-Nearest Neighbours (KNN)** is implemented by creating a pipeline with a TF-IDF vectorizer and KNN classifier, and the pipeline includes a TF-IDF vectorizer with specific settings and a K-Nearest Neighbors classifier with the number of neighbours set to 25 which can be adjusted as per requirement.



```
model = make_pipeline(TfidfVectorizer(stop_words='english',  
                                     max_features=200, sublinear_tf=True),  
                     KNeighborsClassifier(n_neighbors=25))
```

- (5) **Naïve-Bayes** is implemented by creating a pipeline with TF-IDF vectorizer and Multinomial Naïve-Bayes classifier, and the pipeline includes a TF-IDF vectorizer with specific settings and a Multinomial Naïve-Bayes classifier with adjustable alpha for regularization.



```
model = make_pipeline(TfidfVectorizer(stop_words='english',  
                                     max_features=2000, sublinear_tf=True),  
                     MultinomialNB(alpha=0.5))
```

- (6) **Author's Multilevel N-gram Profile (AMNP)** with Naïve-Bayes is implemented by performing feature extraction using AMNP where the function tokenizes a document, extracts unigrams, bigrams, and trigrams, concatenates them, and converts them into strings.


```
def amnp_features(document):  
    tokens = nltk.word_tokenize(document.lower())  
    unigrams = tokens  
    bigrams = list(ngrams(tokens, 2))  
    trigrams = list(ngrams(tokens, 3))  
    amnp_features = unigrams + bigrams + trigrams  
    amnp_features = [item if isinstance(item, str) else  
        ' '.join(item) for item in amnp_features]  
    return " ".join(amnp_features)
```

Then, AMNP feature extraction is applied to all documents in the dataset, the AMNP features are vectorized with a smaller vocabulary size, and it is used to train a Multinomial Naïve-Bayes classifier.

```
amnp_documents = [amnp_features(doc) for doc in documents]  
vectorizer = CountVectorizer(max_features=1000)  
X_train_vectorized = vectorizer.fit_transform(X_train)  
X_test_vectorized = vectorizer.transform(X_test)  
clf = MultinomialNB()  
clf.fit(X_train_vectorized, y_train)
```

- (7) **Convolutional Neural Network (CNN)**, which refers to a single-channel convolutional neural network approach here, is implemented in a sequence of steps as discussed ahead:

We import the required libraries from sklearn and keras.



```
import nltk
import numpy as np
from sklearn.model_selection import train_test_split
from sklearn.preprocessing import LabelEncoder
from tensorflow.keras.preprocessing.text import Tokenizer
from tensorflow.keras.preprocessing.sequence import
    pad_sequences
from tensorflow.keras.models import Sequential
from tensorflow.keras.layers import Embedding, Conv1D,
    GlobalMaxPooling1D, Dense, Dropout
from tensorflow.keras.callbacks import EarlyStopping
```

We create a set of all categories and filters for common categories with more than one file, a list X of documents, where each document is represented as a space-separated string of words, a list y of category labels for each document, and split the data into train and test sets.



```
docs = REUTERS.fileids()
categories = set(REUTERS.categories())
common_categories = [cat for cat in categories if
    len(REUTERS.fileids(cat)) > 1]
X = [" ".join(REUTERS.words(doc_id)) for doc_id in docs if
    set(REUTERS.categories(doc_id)) <= set(common_categories)]
y = [REUTERS.categories(doc_id)[0] for doc_id in docs if
    set(REUTERS.categories(doc_id)) <= set(common_categories)]
X_train, X_test, y_train, y_test = train_test_split(X, y,
    test_size=0.2, random_state=42)
```

We set the maximum number of words and sequence length, create a tokenizer with a specified number of words and fit it on the training text, convert the training and testing text to sequences, and pad the sequences to a specified length.



```
max_words = 20000
maxlen = 300
tokenizer = Tokenizer(num_words=max_words)
tokenizer.fit_on_texts(X)
X_train_seq = tokenizer.texts_to_sequences(X_train)
X_test_seq = tokenizer.texts_to_sequences(X_test)
X_train_pad = pad_sequences(X_train_seq, maxlen=maxlen)
X_test_pad = pad_sequences(X_test_seq, maxlen=maxlen)
```

We encode categorical labels into numerical format.



```
label_encoder = LabelEncoder()
y_encoded = label_encoder.fit_transform(y)
y_train_encoded = label_encoder.transform(y_train)
y_test_encoded = label_encoder.transform(y_test)
```

We build a more complex CNN model with an embedding layer, convolutional layer, global max-pooling layer, dense layer, dropout, and output layer, compile the model with the Adam optimizer, sparse categorical cross-entropy loss, and accuracy metric, configure early stopping to prevent overfitting, and train the model on the training data with early stopping and a specified batch size.




```
embedding_dim = 100
model = Sequential()
model.add(Embedding(input_dim=max_words,
output_dim=embedding_dim, input_length=maxlen))
```



```
model.add(Conv1D(128, 5, activation='relu'))
model.add(GlobalMaxPooling1D())
model.add(Dense(64, activation='relu'))
model.add(Dropout(0.5))
model.add(Dense(len(set(y)), activation='softmax'))
model.compile(optimizer='adam',
              loss='sparse_categorical_crossentropy',
              metrics=['accuracy'])
early_stopping = EarlyStopping(monitor='val_loss', patience=3,
                               restore_best_weights=True)
history = model.fit(X_train_pad, y_train_encoded, epochs=20,
                   validation_split=0.2, callbacks=[early_stopping],
                   batch_size=64)
```

We can load the pre-trained model from the specified file path. The file should be in HDF5 format (.h5). It uses the loaded model to make predictions on the user input. “predicted_probs” contains the predicted probabilities for each class, and “predicted_class” contains the predicted class, which is the one with the highest probability.



```
from tensorflow.keras.models import load_model
model = load_model('authorship_attribution_model.h5')
predicted_probs = model.predict(user_input_pad)
predicted_class = np.argmax(predicted_probs)
```

3.3.2 Evaluation

On comparing the results from the aforementioned machine learning models, we obtain the following sequence of the models arranged in ascending order of accuracy:

$$\text{Logistic Regression} < \text{PCA \& RF} < \text{SVM} < \text{KNN} < \text{AMNP \& Naïve-Bayes} \leq \text{Naïve-Bayes} < \text{CNN}$$

The best accuracy values for each of the aforementioned models have been provided in Table 3.1 along with the appropriate linguistic feature set which yields the highest accuracy value.

Model	Optimal Linguistic Feature Set	Test Accuracy
Logistic Regression	Count Vectorization	70.20 %
PCA & RF	TF-IDF Vectorization	74.79 %
SVM	TF-IDF Vectorization	75.14 %
KNN	TF-IDF Vectorization	79.47 %
AMNP & Naïve-Bayes	Count Vectorization	80.81 %
Naïve-Bayes	TF-IDF Vectorization	80.93 %
CNN	Count Vectorization	84.18 %

Table 3.1: Optimal linguistic feature sets and test accuracies for authorship attribution using different machine learning models.

The different linguistic feature sets across which they were tested include count vectorization, TF-IDF vectorization, write-print stylometry, basic-9 features and word embeddings.

- (1) **Count vectorization** involves tokenization of text documents into a matrix of token counts and comparing frequency of words.
- (2) **TF-IDF vectorization** represents importance of words by comparing for Term Frequency (as a proportion of total word count) and Inverse Document Frequency (proportion of documents containing the word).
- (3) **Write-print stylometry** involves lexical features (word counts, word lengths, bigrams and trigrams, vocabulary richness), syntactic features (function words and punctuations), structural features (sentence and paragraph arrangements), content-specific features (acronyms, keywords

and jargons) and idiosyncratic features (common mistakes).

- (4) **Basic-9 features** include character count, unique words count, lexical density, average syllables per word, sentence count, average sentence length and some readability metrics.
- (5) **Word embeddings** are based on co-occurrence of words and uses a combination of GloVe (Global Vector) embeddings and word-word embeddings to represent documents.

We observe that using convolutional neural networks yields the highest accuracy. Hence, it is justifiable to try and improve this approach to obtain higher accuracies.

3.4 Conclusion

In this paper, we explored and analysed the applicability of different supervised machine learning algorithms for authorship attribution using different linguistic analysis techniques. We holistically evaluated their performances across two datasets, and from our findings, it seems fair to conclude that using convolutional neural networks is the best approach for authorship attribution of articles using linguistic analysis. However, we also note that instead of using a single-channel CNN, implementing a multi-channel CNN based on special word embedding vectors may improve the performance of this approach. We discuss this modification in Chapter 4.

Chapter 4

Evaluation of Multi-channel Convolutional Neural Networks for Word Embedding based Linguistic Analysis

Content

4.1	Overview	29
4.2	Multi-channel Convolutional Neural Networks . . .	30
	4.2.1 Global Vector Word Embeddings	30
	4.2.2 Multi-channel CNN Algorithms	32
4.3	Experimental Protocol	33
	4.3.1 Datasets	34
	4.3.2 Configuration and Implementation	35
	4.3.3 Evaluation metrics	45
4.4	Results and Discussion	45
	4.4.1 Across varying dataset sizes	46
	4.4.2 Further analysis	47
4.5	Conclusion	47

4.1 Overview

Authorship attribution of articles using linguistic analysis is a topic of active research. It involves an array of different tasks like feature selection, feature extraction, configuring and training machine learning models, adjusting their parameters for increased accuracy, and testing with new data. Classically, different supervised machine learning algorithms have been applied to perform this task. Among such methods, using convolutional neural networks appears to be the best approach according to our evaluation in Chapter 3.

Although using a standard single-channel convolutional neural network achieves better performance than other machine learning models, there is still room for improvement. The proposed approach in our work modifies the previously discussed CNN approach. We use the convolutional neural network with word embeddings, where each word is mapped to a continuous-valued word vector using GloVe (global vector) embeddings and each input document is represented as a concatenation of word embeddings. This modified CNN model is trained using these document representations as input for authorship attribution. The multi-channel CNN consisting of a static word embedding channel (word vectors trained by GloVe embeddings) and a non-static word embedding channel (word vectors trained initially by GloVe embeddings then updated during training) is trained.

In this chapter, we briefly describe global vector word embeddings and multi-channel convolutional neural networks with details of the steps in our application of multi-channel CNN with global vector word embeddings, and present the experimental details of our approach with datasets, configuration and implementation, evaluation metrics, and follow it up with an evaluation of the results across different dataset sizes and different linguistic feature sets, before presenting further analysis of our work.

The experimental pipeline for our approach using a multi-channel CNN with GloVe embeddings is schematically represented in Figure 4.3.

4.2 Multi-channel Convolutional Neural Networks

In this section, we explore the structure of global vector word embeddings and the representation of words as word embeddings, and then move to a description of multi-channel CNNs with an understanding of the steps in our application of multi-channel CNN with global vector word embeddings.

4.2.1 Global Vector Word Embeddings

GloVe (Global Vectors for Word Representation) is a word embedding technique developed by researchers at Stanford University. Unlike other word embedding methods that rely solely on local context information (such as word2vec), GloVe combines both local context and global statistical information of a corpus to create word vectors. These embeddings are dense vector representations that capture semantic relationships between words by analyzing word co-occurrence statistics from a large corpus. Unlike traditional count-based models, GloVe constructs a global word-word co-occurrence matrix and leverages matrix factorization techniques to produce lower-dimensional word vectors.

```
, -0.082752 0.67204 -0.14987 -0.064983 0.056491 0.40228 0.0027747 -0.3311 -0.30691 2.0817 0.031819 0.013643
-0.13057 -0.054601 0.037083 -0.070552 0.5893 -0.30385 0.2898 -0.14653 -0.27052 0.37161 0.32031 -0.29125 0.00
-0.091386 0.40075 -0.17223 0.18145 0.37586 -0.28682 0.37289 -0.16185 0.18008 0.3032 -0.13216 0.18352 0.09575
0.27985 -0.074412 -0.13762 -0.21866 0.18138 0.040855 -0.113 0.24107 0.3657 -0.27525 -0.05684 0.34872 0.01188
-0.25626 0.17715 -0.54095 0.16596 -0.036058 0.08499 -0.64989 0.075549 -0.28831 0.40626 -0.2802 0.094062 0.32
0.27512 0.36012 0.16311 0.23964 -0.05923 0.3261 0.20559 0.038677 -0.045816 0.089764 0.43151 -0.15954 0.08532
-0.13919 0.56252 0.21457 -0.46443 -0.012211 0.029988 -0.051094 -0.20135 0.80788 0.47377 -0.057647 0.46216 0.
0.054056 0.046816 0.059539 0.046265 0.17754 -0.31094 0.28119 -0.24355 0.085252 -0.21011 -0.19472 0.0027297 -
0.24074 0.41923 0.13012 -0.17167 -0.37808 -0.23089 -0.019477 -0.29291 -0.30824 0.30297 -0.22659 0.081574 -0.
-0.091065 0.23438 -0.0041331 0.003232 0.0072134 0.008697 0.21614 0.049904 0.35582 0.13748 0.073361 0.14166 0.
-0.18251 0.20441 0.13319 0.1294 0.050594 -0.15612 -0.39543 0.12538 0.24881 -0.1927 -0.31847 -0.12719 0.4341 ...
-0.14023 0.028462 0.56923 -0.1649 -0.036429 0.010051 -0.17107 -0.042608 0.044965 -0.4393 -0.26137 0.30088 -0.
0.059389 0.074901 0.061068 -0.4662 0.40054 -0.19099 -0.14331 0.018267 -0.18643 0.20709 -0.35598 0.05338 -0.0
0.012001 0.20751 -0.12578 -0.59325 0.12525 0.15975 0.13748 -0.33157 -0.13694 1.7893 -0.47094 0.70434 0.266
0.31703 0.19025 -0.37505 0.035603 0.1181 0.012032 -0.037566 -0.5046 -0.049261 0.092351 0.11031 -0.073062 0.3
-0.19392 -0.11068 -0.014088 -0.17906 0.24509 -0.16878 -0.15351 -0.13808 0.02151 0.13699 0.0068061 -0.14915 -
0.48945 0.15388 0.05295 -0.049831 0.11207 0.14881 -0.37003 0.30777 -0.33865 0.045149 -0.18987 0.26634 -0.264
-0.36371 -0.25948 -0.42398 -0.14305 -0.10208 0.21498 -0.21924 -0.17935 0.21546 0.13801 0.24504 -0.2559 0.054
...
```

Figure 4.1: Contents of the downloadable GloVe file “glove.840B.300d.txt”.

These vectors effectively encode meaningful semantic relationships, allowing for analogies and similarities to be captured. GloVe embeddings are widely used in natural language processing tasks such as text classification, sentiment analysis, and machine translation due to their ability to represent words in a continuous vector space that reflects their contextual usage.

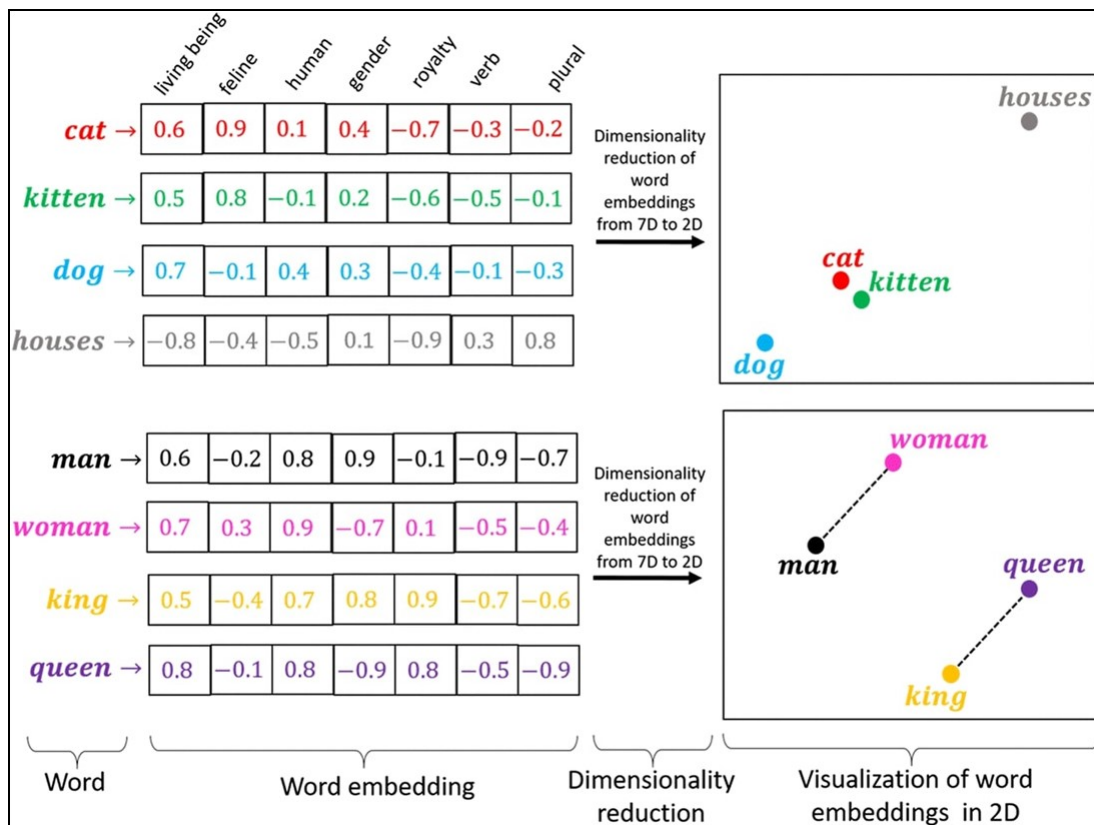


Figure 4.2: Representation of words as word embeddings and their visualization in a 2D space (Source: coderzcolumn.com)

The GloVe file, which has a size of 55,13,905 KB (or approximately 5.26 GB) can be downloaded from the following URL:

<http://nlp.stanford.edu/data/glove.840B.300d.zip>

4.2.2 Multi-channel CNN Algorithms

Multi-Channel Convolutional Neural Networks (CNNs) extend the capabilities of standard CNNs by processing data with multiple input channels, such as RGB images or multispectral data. Each channel represents different types of information, and the convolutional filters operate across all channels simultaneously, allowing the network to learn features that integrate information from each channel. This capability is particularly useful for tasks involving complex data, such as color image processing, multi-sensor data fusion, and medical imaging. By combining information from various channels, multi-channel CNNs can capture more detailed and nuanced features, leading to improved performance in applications like diagnostic imaging, where different types of scans are integrated, and in IoT, where data from multiple sensors are fused. This results in a more comprehensive and accurate analysis compared to single-channel CNNs.

Our implementation of a multi-channel CNN model with global vector word embeddings includes the following layers as shown in Table 4.1.

Layer	Output Shape	Number of Parameters	Number of Input Channels	Number of Output Channels
Input	(12998)	0	2	2
Embedding	(12998,300)	9331200	2	2
Concatenate	(12998,600)	0	2	1
1D-Convolution	(12994,64)	192064	1	1
1D-Max Pooling	(1,64)	0	1	1
Flatten	(64)	0	1	1
Dense	(256)	16640	1	1
Dropout	(256)	0	1	1
Dense	(5)	1285	1	1

Table 4.1: Layers in multi-channel CNN model with output shapes, number of parameters, input channels and output channels (for implementation using Blog Authorship Dataset with 5 authors).

The sequence of the different layers listed in Table 4.1 can also be represented schematically using a model diagram as shown in Figure 4.3.

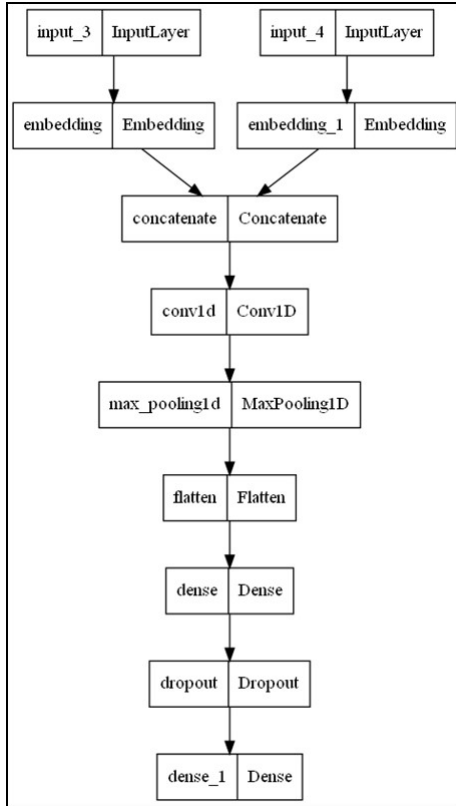


Figure 4.3: Model diagram of proposed multi-channel CNN model.

In the following, we discuss the configuration and implementation of the aforementioned model in more detail, and evaluate its performance across different dataset sizes before presenting further analysis of the approach.

4.3 Experimental Protocol

In this section, we present the experimental details of the implementation and evaluation of the multi-channel convolutional neural network model.

4.3.1 Datasets

Two datasets, the Reuters Corpus and the Blog Authorship Corpus, have been used for evaluating the performance of supervised machine learning models in authorship attribution. We briefly describe their characteristic aspects.

- (1) **The Reuters Corpus** is a collection of documents that appeared on Reuters newswire in 1987. The documents were assembled and indexed with categories. The original corpus has 10,369 documents and a vocabulary of 29,930 words. This collection of news articles is widely used in natural language processing and text classification tasks. It consists of documents covering various topics, each labelled with one or more categories. Researchers use it to develop and evaluate machine learning models for tasks like text categorization and information retrieval. The dataset is accessible through the Natural Language Toolkit (NLTK) and serves as a benchmark for NLP research. It is a multi-class, multi-label dataset. It has 90 classes, 7769 training documents and 3019 testing documents. It is a subset of the Reuters-21578 benchmark.

The Reuters Corpus can be downloaded from the URL:

<https://kdd.ics.uci.edu/databases/reuters21578/reuters21578.html>

- (2) **The Blog Authorship Corpus** consists of the collected posts of 19,320 bloggers gathered from blogger.com in August 2004. The corpus incorporates a total of 681,288 posts and over 140 million words - or approximately 35 posts and 7250 words per person. Each blog is presented as a separate file, the name of which indicates a blogger id and the blogger's self-provided gender, age, industry and astrological sign. For each age group there are an equal number of male and female bloggers.

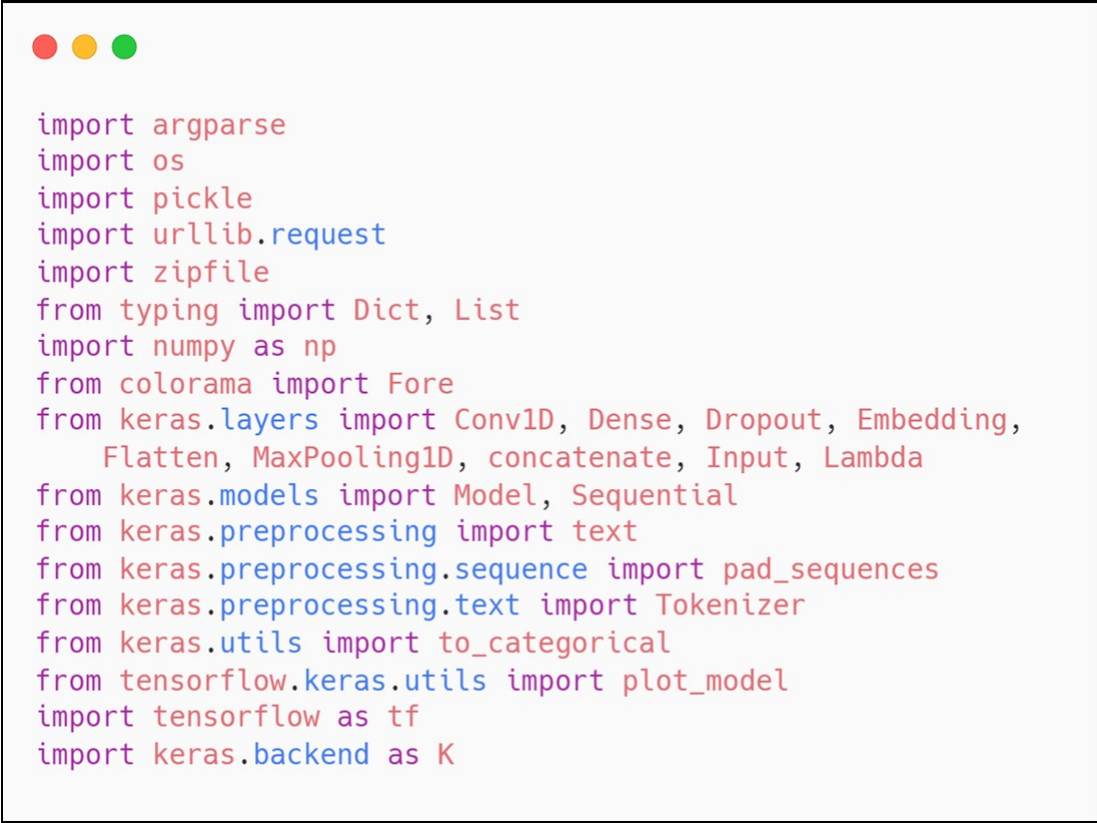
The Blog Authorship Corpus can be downloaded from the URL:

<https://u.cs.biu.ac.il/~koppel/BlogCorpus.htm>

4.3.2 Configuration and Implementation

Our suggested approach which modifies the traditional single-channel CNN model discussed in Chapter 3 by using a multi-channel convolutional neural network with GloVe (global vector) word embeddings, is implemented in a sequence of steps as discussed ahead:

We import the required libraries from keras.



```
import argparse
import os
import pickle
import urllib.request
import zipfile
from typing import Dict, List
import numpy as np
from colorama import Fore
from keras.layers import Conv1D, Dense, Dropout, Embedding,
    Flatten, MaxPooling1D, concatenate, Input, Lambda
from keras.models import Model, Sequential
from keras.preprocessing import text
from keras.preprocessing.sequence import pad_sequences
from keras.preprocessing.text import Tokenizer
from keras.utils import to_categorical
from tensorflow.keras.utils import plot_model
import tensorflow as tf
import keras.backend as K
```

Then, we define a function which checks for the presence of the GloVe model in the local directory and automatically downloads the model from the url <http://nlp.stanford.edu/data/glove.840B.300d.zip> only if not found, since it is a large file and can significantly slow down the execution.



```
def download_glove(filepath: str):
    print("GloVe model not found. Downloading from
          http://nlp.stanford.edu/data/glove.840B.300d.zip ...")
    url = "http://nlp.stanford.edu/data/glove.840B.300d.zip"
    urllib.request.urlretrieve(url, filepath)
```

We define a function to load the word embeddings from the GloVe model.



```
def load_model(filepath: str) -> dict:
    parent_dir = os.path.dirname(filepath)
    print(f"Loading GloVe model from {filepath} ...")
    if not os.path.exists(filepath):
        download_glove(filepath)
        with zipfile.ZipFile(filepath, "r") as zip_ref:
            zip_ref.extractall(parent_dir)
    embeddings_index = {}
    with open(filepath.replace(".zip", ".txt"), "r",
              encoding="utf-8") as file:
        for line in file:
            values = line.split()
            word = values[0]
            try:
                coefs = np.asarray(values[1:],
                                   dtype="float32")
                if len(coefs) == 300:
                    embeddings_index[word] = coefs
            except:
                continue
    print(f"{len(embeddings_index)} word vectors found.")
    return embeddings_index
```


We define a CNN model with two parallel embedding layers (trainable and non-trainable), and pass the word embeddings, word indices, dimensions of word embeddings and maximum length of input sequence as parameters.

```
def model_cnn_word_word(
    embedding_matrix: Dict[str, np.ndarray],
    word_index: Dict[str, int],
    embedding_dim: int,
    max_sequence_length: int,
    number_of_authors: int,
) -> Sequential:
    embed1_in = Input(shape=(None, max_sequence_length))
    embed1_in = tf.reshape(embed1_in, shape=
[tf.shape(embed1_in)[0]*tf.shape(embed1_in)
[1],max_sequence_length])
    embed1_out = Embedding(
        len(word_index) + 1,
        embedding_dim,
        weights=[embedding_matrix],
        input_length=max_sequence_length,
        trainable=False,
    )(embed1_in)
    embed2_in = Input(shape=(None, max_sequence_length))
    embed2_in = tf.reshape(embed2_in, shape=
[tf.shape(embed2_in)[0]*tf.shape(embed2_in)
[1],max_sequence_length])
    embed2_out = Embedding(
        len(word_index) + 1,
        embedding_dim,
        weights=[embedding_matrix],
        input_length=max_sequence_length,
        trainable=True,
    )(embed2_in)
```

We implement the remaining layers in the CNN model, compile and plot it.



```
conc1 = concatenate([embed1_out, embed2_out])
conc2 = Conv1D(64, 5, activation="relu")(conc1)
conc3 = MaxPooling1D(max_sequence_length - 5 + 1)(conc2)
conc4 = Flatten()(conc3)
conc5 = Dense(256, activation="relu")(conc4)
conc6 = Dropout(0.5)(conc5)
conc7 = Dense(number_of_authors, activation="softmax")
(conc6)
model = Model([embed1_in, embed2_in], conc7)
model.compile(loss="categorical_crossentropy",
optimizer="rmsprop", metrics=["acc"])
plot_model(model, to_file='model.png')
print("-----\n")

print(model.summary())
print("-----\n")

return model
```

We define functions to perform the following tasks as well:

- (1) A function that takes in a dictionary of word embeddings and a dictionary of word indices, and returns a matrix of word embeddings where any missing words are filled in with zeros
- (2) A function that creates a tokenizer object and fit it on the given lines of text, and returns a tokenizer object that has been fit on the given text.
- (3) A function that loads data from a pickle file, and returns data loaded from the pickle file.
- (4) A function that cleans the provided text by removing special characters and converting the text to a sequence of words, and returns cleaned text string.
- (5) A function that extracts the features for inputText and authorLabel from each row, and returns list of tuples containing the cleaned text and the author id.



```
def fill_in_missing_words_with_zeros(
    embeddings_index: Dict[str, np.ndarray],
    word_index: Dict[str, int],
    EMBEDDING_DIM: int,
) -> np.ndarray:
    embedding_matrix = np.zeros((len(word_index) + 1,
    EMBEDDING_DIM))
    for word, i in word_index.items():
        embedding_vector = embeddings_index.get(word)
        if embedding_vector is not None:
            embedding_matrix[i] = embedding_vector
    return embedding_matrix
```



```
def create_tokenizer(lines: List[str]) -> Tokenizer:
    tokenizer = Tokenizer()
    tokenizer.fit_on_texts(lines)
    return tokenizer
```



```
def load_pickle_data(path: str):
    with open(path, "rb") as file:
        data = pickle.load(file)
    return data
```



```
def get_clean_text(input_text: str) -> str:
    clean_text_list = text.text_to_word_sequence(
        input_text, filters="", lower=False, split=" "
    )
    clean_text = " ".join(clean_text_list)
    return clean_text
```



```
def prepare_data_for_classification(data: List[tuple]) ->
tuple:
    X_data = []
    y_data = []
    for input_text, author_label in data:
        clean_text = get_clean_text(input_text)
        X_data.append(clean_text)
        y_data.append(author_label)
    return X_data, y_data
```

Then, we define a function which trains a classifier using the provided training data, and returns trained classifier and StandardScaler used to scale the data.



```
def train_classifier(
    X_train: List[List[float]],
    y_train: List[int],
    glove_filepath: str,
    number_of_authors: int,
    number_of_epochs: int,
    embedding_dim: int = 300,
) -> tuple:
```



```
tokenizer = create_tokenizer(X_train)
sequences = tokenizer.texts_to_sequences(X_train)
max_sequence_length = max([len(article.split()) for
article in X_train])
vocab_size = len(tokenizer.word_index) + 1
print(f"\nMax document length : {max_sequence_length}")
print(f"Vocabulary size : {vocab_size}")
x_train = pad_sequences(sequences,
maxlen=max_sequence_length)
y_train = to_categorical(y_train)
print(f"Shape of data tensor : {x_train.shape}")
print(f"Shape of label tensor : {y_train.shape}")
print("-----\n")

glove_embedding_matrix = load_model(glove_filepath)
print("Filling non existing words...\n")
print("-----\n")

glove_embedding_matrix = fill_in_missing_words_with_zeros(
glove_embedding_matrix, tokenizer.word_index,
embedding_dim
)
model = model_cnn_word_word(
glove_embedding_matrix,
tokenizer.word_index,
embedding_dim,
max_sequence_length,
number_of_authors,
)
print("-----\n")

model.fit([x_train, x_train], y_train,
epochs=number_of_epochs, batch_size=50)
return tokenizer, model, max_sequence_length
```


We also define a function which tests the classifier using the test data.

```
def test_classifier(  
    X_test: List[List[float]],  
    y_test: List[int],  
    tokenizer: Tokenizer,  
    max_sequence_length: int,  
    model: Sequential,  
) -> None:  
    sequences = tokenizer.texts_to_sequences(X_test)  
    y_test = to_categorical(y_test)  
    x_test = pad_sequences(sequences,  
maxlen=max_sequence_length)  
    _, acc = model.evaluate([x_test, x_test], y_test,  
verbose=0)  
    print("-----  
-----\n")  
    print("Test Accuracy : %f" % (acc * 100))  
    print("-----  
-----\n")
```

Finally, we define the main function to process the dataset, train a classifier, and test it.

```
def main(  
    number_of_authors: int,  
    dataset_type: str,  
    train_test_data_path: str,  
    glove_model_path: str,  
    number_of_epochs: int  
) -> None:
```



```

print("-----\n")
print(f"Number of Authors      : {number_of_authors}")
print(f"Dataset Type           : {dataset_type}")
print(f"Train-Test Data Path      : {train_test_data_path}")
train_data_path = (
    f"{train_test_data_path}/{str(number_of_authors)}"
    f"_train_files_{dataset_type}.pkl"
)
test_data_path = (
    f"{train_test_data_path}/{str(number_of_authors)}"
    f"_test_files_{dataset_type}.pkl"
)
train_data = load_pickle_data(train_data_path)
test_data = load_pickle_data(test_data_path)
X_train, y_train =
prepare_data_for_classification(train_data)
X_test, y_test =
prepare_data_for_classification(test_data)
print(f"Total Train Articles   : {len(train_data)}")
print(f"Total Test Articles      : {len(test_data)}")
print("-----\n")

tokenizer, model, max_sequence_length = train_classifier(
    X_train, y_train, glove_model_path, number_of_authors,
    number_of_epochs
)
test_classifier(X_test, y_test, tokenizer,
max_sequence_length, model)
print(Fore.RESET)

```

The main program for the implementation of this approach is given as:



```
if __name__ == "__main__":
    parser = argparse.ArgumentParser(description="Train and
test CNN classifier.")
    parser.add_argument(
        "--number_of_authors", type=int, help="Number of
authors to keep for classification.",
    )
    parser.add_argument(
        "--dataset", type=str, choices=["blogs"], help='Choose
"blogs" dataset.',
    )
    parser.add_argument(
        "--train_test_data_path", type=str, help="Path to the
train/test data."
    )
    parser.add_argument(
        "--glove_model_path", type=str, help="Path of
glove.840B.300d.zip model."
    )
    parser.add_argument(
        "--number_of_epochs", type=int, help="Number of
epochs."
    )
    args = parser.parse_args()
    main(
        args.number_of_authors,
        args.dataset,
        args.train_test_data_path,
        args.glove_model_path,
        args.number_of_epochs,
    )
```

The format of the execution sequence is given with an example, where N, T, G and E denote the number of authors, train-test data path, GloVe model path and number of epochs.



```
python classifier.py --number_of_authors <N> --dataset blogs -  
-train_test_data_path <T> --glove_model_path <G> --  
number_of_epochs <E>  
  
python classifier.py --number_of_authors 5 --dataset blogs --  
train_test_data_path ../train_test_data --glove_model_path  
../../glove.840B.300d.zip --number_of_epochs 8
```

The above code illustrates the execution of the “classifier.py” program from shell with different values for number of authors and number of epochs.

4.3.3 Evaluation metrics

The performance of the suggested approach is evaluated in terms of test accuracy values and execution time across different dataset sizes and number of epochs. We will associate higher test accuracy values and lower execution times with higher effectiveness and higher efficiency to evaluate the performance of the model. We observe that performance drops gradually with increasing dataset sizes, whereas while increasing number of epochs, it only increases upto a certain limit and does not increase indefinitely. The results are discussed in more detail in the next section.

4.4 Results and Discussion

We report the findings of the experimentation across the datasets as metrics averaged over multiple executions. As discussed earlier, our experiments comprise evaluating the performance of a multi-channel convolutional neural network model using word embeddings, and comparing it with those of other machine learning algorithms discussed in Chapter 3. The exhaustive results across dataset sizes have been tabulated in Table 4.3.

4.4.1 Across varying dataset sizes

We implemented varying dataset sizes by varying the number of authors in the corpus. For example, in the Blog Authorship Corpus, we obtained three different dataset sizes with 5, 10 and 20 authors. Table 4.2 illustrates the varying dataset sizes across varying number of authors.

Number of Authors	Size of Dataset
5	3900 KB
10	6060 KB
20	8940 KB

Table 4.2: Dataset sizes across different number of authors in the Blog Authorship Corpus.

In the following table, we list the test accuracy values and approximate execution times across the different dataset sizes.

Number of Authors	Test Accuracy	Execution Time
5	94.99 %	17 minutes
10	93.87 %	44 minutes
20	92.13 %	125 minutes

Table 4.3: Test accuracy values and approximate execution times across different number of authors in the Blog Authorship Corpus.

From these observations, it is reasonable to infer that multi-channel convolutional neural network models using word embeddings are far more effective than other machine learning algorithms for performing authorship attribution of articles using linguistic analysis, and thus, are worthy of greater attention and future research.

4.4.2 Further analysis

On analysing the results obtained in Table 4.2 and Table 4.3, we observe that execution time increases at a rate higher than the geometric rate of increase with a geometric increase in the dataset size. We also observe that the test accuracy values drop faster as dataset size increases but remain above 90%, as we increase the number of epochs accordingly.

On comparing execution times of the other machine learning approaches discussed in Chapter 3, we further observed that using a multi-channel CNN yields higher accuracy values but consumes more time for similar dataset sizes. Thus, there appears to be an increasing trade-off between effectiveness and efficiency as we increase the dataset size, while choosing a multi-channel CNN model over models using other machine learning algorithms.

4.5 Conclusion

Despite the great progress in authorship attribution of articles using linguistic analysis in recent times, the concerns regarding the availability of a reliable and efficient machine learning algorithm for fast and accurate authorship attribution are yet to be fully resolved. In this study, we take a step forward in this direction and try to come up with an approach more effective than the traditional methods by combining the use of multi-channel convolutional neural networks with global vector word embeddings for performing the task of authorship attribution. Our results suggest the high applicability of multi-channel CNN models for this task. With a systematic analysis of our findings, we also show the competitive performance of this approach with respect to the other approaches in this field. We feel our holistic evaluation should be of great assistance to other researchers working in related domains. The inferences drawn from our findings, as well as some of the limitations, should be of great interest to fellow researchers for further exploration. In future, we plan to expand the horizon of our investigations by digging deeper into the nuances of each linguistic feature and combine them to get better results.

Chapter 5

Conclusions and Future Work

5.1 Conclusions

This thesis focuses on the application of supervised machine learning algorithms for performing authorship attribution of articles using linguistic analysis techniques. Our contributions to this field are twofold. Firstly, we perform a holistic evaluation of multiple machine learning algorithms using different linguistic feature sets on multiple datasets. To the best of our knowledge, this is a unique evaluation of its kind, which makes our analysis of practical use to the law enforcement community for criminal investigation purposes. Secondly, we demonstrate the configuration and implementation of an improved convolutional neural network approach using global vector word embeddings in a multi-channel model. We further demonstrate the competitive performance of this approach which exceeds those of traditional supervised machine learning algorithms, hence proving its immense applicability to the task of authorship attribution of articles using linguistic analysis.

5.2 Future research directions

While considerable research has already taken place in the fields of linguistic analysis and authorship attribution using machine learning algorithms, relatively less exploration has been performed at their combination. While linguistic analysis had vastly remained a topic of research in the liberal arts

domain, research on authorship attribution using machine learning remained primarily focused on optimizing simple linguistic feature sets like vectorization for supervised learning. However, recent research has increased the understanding of the effectiveness of their combined usage. Our work represents an investigation of popular supervised machine learning algorithms and their performances across different linguistic feature sets, and suggests a modified approach using word embeddings and multi-channel convolutional neural networks. As a result, our analysis raises a number of intriguing questions regarding its applicability in more complex problems involving authorship attribution of articles using linguistic analysis.

Firstly, the primary motivation behind our work has been the possibility of using linguistic analysis techniques to identify the authors of anonymous text documents like threat letters, ransom notes and hoax messages received from unidentified sources for the purpose of criminal investigations. It is important to note that this usage can be extended to include (1) determining the true authors of historical texts, unfinished and disputed literary works, (2) detecting plagiarism by identifying the original authors of copied texts, (3) moderating social media and online content by detecting fake news and automated reviews, as well as identifying the original sources behind provocative and defamatory content in pseudonymous or anonymous online posts.

Secondly, our work focuses exclusively on linguistic analysis of text written in English language only. Thus a potential direction for future research emerges in the identification of language agnostic features for improved cross-language attribution in vernacular or multilingual environments.

Thirdly, although mentioned in Chapter 2, our work does not explore the applicability of sociolinguistic features or the performance of our approach across these features. Attribution of demographic specifics of the author like age, gender, country, education, profession, personality type, political views, etc. may provide faster approaches of authorship attribution as the list of authors for comparison can be reduced in every step. This warrants further investigation into the impact of sociolinguistic analysis on performance.

Next, it is important to note that every approach discussed or suggested in our work requires all training and testing data to be available beforehand, and

are not suitable for situations where the model has to adapt to include new articles and authors during its execution. Thus, authorship attribution during text generation to enhance real-time monitoring of online content can be another potential direction for future research.

Apart from these, our work assumes that the writing style and other identifiable features of every author remain constant over time and across different genres of articles, which may not be true in all cases. This opens up an intriguing line of research into the development of dynamic and evolving models capable of adapting to gradual changes in writing styles and patterns over time for long-term authors.

Lastly, addressing ethical issues is of utmost importance to ensure responsible and unbiased use of authorship attribution. Compliance with ethical considerations and mitigation of bias in the case of inaccurate results provide directions into a relatively new but significant field of research.

In conclusion, the directions of investigation performed in our study have led to a number of intriguing questions about the applicability of our approach in areas hitherto not studied. We hope these prove to be of interest and provide future directions of research to the community at large.

References

- [1] Sreenivas Mekala, Vishnu Vardan Bulusu, and Raghunadha Reddy T., “A Survey On Authorship Attribution Approaches” in *International Journal of Computational Engineering Research (IJCER)*, vol. 8, issue 9, pp. 2250 – 3005, 2018.
- [2] Moshe Koppel, Jonathan Schler, and Shlomo Argamon, “Authorship Attribution: What’s Easy and What’s Hard” in *Journal of Law and Policy*, vol. 63, pp. 23 – 68, 2013.
- [3] Farkhund Iqbal, Rachid Hadjidj, Benjamin C.M. Fung, and Mourad Debbabi, “A Novel Approach of Mining Write-prints for Authorship Attribution in E-mail Forensics” in *Digital Investigation Journal*, vol. 16, pp. 42 – 51, 2008.
- [4] Anderson Rocha, Senior Member, IEEE, Walter J. Scheirer, Senior Member, IEEE, Christopher W. Forstall, Thiago Cavalcante, Antonio Theophilo, Bingyu Shen, Ariadne R. B. Carvalho, and Efstathios Stamatatos, “Authorship Attribution for Social Media Forensics” in *IEEE Transactions on Information Forensics and Security*, vol. 108, pp. 85 – 110, 2016.
- [5] K. A. Apoorva, and S. Sangeetha, “Deep Neural Network and Model based Clustering Technique for Forensic Electronic Mail Author Attribution” in *SN Applied Sciences*, research article, pp. 3 – 348, 2021.

-
- [6] Tim Grant (Aston University), “The Idea of Progress in Forensic Authorship Analysis”, *University Printing House, Cambridge, United Kingdom*, 2022.
- [7] Mashael M. AlAmr, and Eric Atwell, “WEKA in Forensic Authorship Analysis” in *A Corpus-based Approach of Saudi Authors- Proceedings of the 17th International Conference on Natural Language Processing, NLP Association of India (NLP AI)*, pp. 257 – 260, 2020.
- [8] Asif Karim, Sami Azam, Bharanidharan Shanmugam, Krishnan Kannoorpatti, and Mamoun Alazab, “A Comprehensive Survey for Intelligent Spam Email Detection” in *IEEE ACCESS*, vol. 7, pp. 230 – 278, 2019.
- [9] 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, Seattle, Washington, USA, Association for Computational Linguistics*, pp. 1880 – 1891, 2013.
- [10] Xuan Li, “Authorship Attribution on the Enron Email Corpus”, *Duquesne University*, masters thesis, 2013.
- [11] Omar González Brito, José Luis Tapia Fabela, and Silvia Salas Hernández, “New Approach to Feature Extraction in Authorship Attribution” in *International Journal of Combinatorial Optimization Problems and Informatics*, pp. 10 – 34, 2021.
- [12] Joseph Rudman, “The State of Authorship Attribution Studies, Some Problems and Solutions – Computers and the Humanities”, *Kluwer Academic Publishers, Netherlands*, pp. 351 – 365, 1998.

-
- [13] Abdulaziz Altamimi, Saud Alotaibi, and Abdulrahman Alruban, “Surveying the Development of Authorship Identification of Text Messages” in *International Journal of Intelligent Computing Research (IJICR)*, vol. 10, issue 1, pp. 109 – 121, 2019.
- [14] Parvez Mahbub, Naz Zarreen Oishie, and S.M. Rafizul Haque, “Authorship Identification of Source Code Segments Written by Multiple Authors Using Stacking Ensemble Method”, *arXiv:2212.05610v1*, 2022.
- [15] Yunpeng Yangb, Leilei Konga, Zhongyua Hana, Yong Hana, and Haoliang Qia, “N-gram-based Authorship Identification of Source Code”, *Forum for Information Retrieval Evaluation, Hyderabad, India*, 2020.
- [16] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, and Carole E. Chaski, “Identifying Authorship by Byte-Level N-Grams: The Source Code Author Profile (SCAP) Method” in *International Journal of Digital Evidence Spring*, vol. 6, issue 1, 2007.
- [17] Georgia Frantzeskou, Efstathios Stamatatos, Stefanos Gritzalis, and Sokratis Katsikas, “Source Code Author Identification Based on N-gram Author Profiles”, *IFIP International Federation for Information Processing, Artificial Intelligence Applications and Innovations, (Boston: Springer)*, vol. 204, pp. 508 – 515, 2006.
- [18] Anđelka Zečević, “N-gram Based Text Classification According To Authorship” in *Proceedings of the Student Research Workshop associated with RANLP 2011*, pp. 145 – 149, 2011.
- [19] George K. Mikros, and Kostas A. Perifanos, “Authorship Attribution in Greek Tweets Using Author’s Multilevel N-Gram Profiles, Analyzing Microtext” in *Papers from the 2013 AAAI Spring Symposium*, 2013.

-
- [20] Jianyu Miaoa, and Lingfeng Niub, “A Survey on Feature Selection” in *Procedia Computer Science*, pp. 919 – 926, 2016.
- [21] Sicong Shao, “Machine Learning-Based Author Identification for Social Media Forensics”, *University of Arizona*, pp. 32 – 45, 2021.
- [22] Manuel Sage, Pietro Cruciata, Raed Abdo, Jackie Chi Kit Cheung, and Yaoyao Fiona Zhao, “Investigating the Influence of Selected Linguistic Features on Authorship Attribution using German News Articles” in *IEEE International Conference on Information Reuse and Integration (IRI)*, pp. 269 – 276, 2018.
- [23] AhmedAbbasi, Abdul Rehman Javed, Farkhund Iqbal, Zunera Jalil, Thippa Reddy Gadekallu, and Natalia Kryvinska, “Authorship Identification using Ensemble Learning” in *Scientific Reports*, pp. 1 – 16, 2022.
- [24] Hossam Ahmed, “The Role of Linguistic Feature Categories in Authorship Verification”, *The 4th International Conference on Arabic Computational Linguistics (ACLing 2018), Dubai, United Arab Emirates*, pp. 10 – 46, 2018.
- [25] Abiodun Modupe, Turgay Celik, Vukosi Marivate, and Oludayo O. Olugbara, “Post-Authorship Attribution Using Regularized Deep Neural Network” in *Applied Sciences*, pp. 5 – 17, 2022.
- [26] Amelec Vilorio, Omar Bonerge Pineda Lezamab, and Eduardo Chang, “Classification of Authors for an Automatic Recommendation Process for Criminal Responsibility”, *The 7th International Symposium on Emerging Inter-networks, Communication and Mobility (EICM), Leuven, Belgium*, 2020.

-
- [27] Javier Calle-Martín, and Antonio Miranda-García, “Stylometry and Authorship Attribution” in *Introduction to the Special Issue, English Studies*, pp. 251 – 258, 2012.
- [28] Łukasz Gągała, “Authorship Attribution with Neural Networks and Multiple Features” in *Notebook for PAN at CLEF 2018*, 2018.
- [29] Yunita Sari, “Neural and Non-neural Approaches to Authorship Attribution”, *University of Sheffield*, thesis, 2018.
- [30] David Wright, “Stylistics versus Statistics: A Corpus Linguistic Approach to Combining Techniques in Forensic Authorship Analysis using Enron emails”, in *Scientific Reports*, pp. 5 – 22, 2014.
- [31] D. Jurafsky, and J. H. Martin, “Speech and Language Processing – An Introduction to Natural Language Processing, Computational Linguistics and Speech Recognition”, *Prentice Hall PTR, Upper Saddle River, New Jersey, USA*, edition 1, pp. 340 – 375, 2000.
- [32] Julian Hitschler, Esther van den Berg, Ines Rehbein, “Authorship Attribution with Convolutional Neural Networks and POS-Eliding”, in *Proceedings of the Workshop on Stylistic Variation, Association for Computational Linguistics*, pp. 53 – 58, 2017.
- [33] Mohammed Abuhamad, Ji-su Rhim, Tamer AbuHmed, Sana Ullah, Sanggil Kang, and DaeHun Nyang, “Code Authorship Identification using Convolutional Neural Networks” in *Future Generation Computer Systems*, pp. 104 – 115, 2019.
- [34] Prasha Shrestha, Sebastian Sierra, Fabio A. González, Paolo Rosso, Manuel Montes-y-Gómez, and Tamar 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, Short Papers, Association for Computational Linguistics, vol. 2, pp. 669 – 674, 2017.

- [35] Dylan Rhodes, “Author Attribution with CNN: Overview of the Author Identification Task” at *PAN 2014*, analysis, pp. 13 – 31, 2014.
- [36] Ali S. Razavian, Hossein Azizpour, Josephine Sullivan, and Stefan Carlsson, “CNN Features Off The Shelf: An Astounding Baseline for Recognition” at *Computer Vision and Pattern Recognition Workshops (CVPRW), 2014 IEEE Conference*, pp. 512 – 519, 2014.
- [37] Antonio Miranda-García, and Javier Calle-Martín, “A Survey of Non-traditional Authorship Attribution Studies”, in *Ecdotica* 5, pp. 147 – 168, 2008.
- [38] Moshe Koppel, Jonathan Schler, Shlomo Argamon, and Eran Messeri, “Authorship Attribution with Thousands of Candidate Authors” in *Proceedings of the 29th ACM SIGIR Conference on Research and Development in Information Retrieval, New York: ACM Press*, pp. 659 – 660, 2006.
- [39] A. M. U. D. Khanday, Q. R. Khan, and S. T. Rabani, “Identifying Propaganda from Online Social Networks during COVID-19 using Machine Learning Techniques” in *International Journal of Information Technology*, pp. 115 – 122, 2021.
- [40] S. Akuma, T. Lubem, and I. T. Adom, “Comparing Bag of Words and TF-IDF with Different Models for Hate Speech Detection from Live Tweets” in *International Journal of Information Technology*, pp. 3629 – 3635, 2022.
- [41] A. E. Kah, A. E. Airej, and I. Zeroual, “Arabic Authorship Attribution on Twitter: What Really Matters” in *Indonesian Journal of Electric*

Engineering and Computer Science, pp. 1730 – 1737, 2022.

- [42] C. Suman, A. Raj, S. Saha, and P. Bhattacharyya, “Authorship Attribution of Microtext using Capsule Networks”, *IEEE Trans Computer Society System*, pp. 1038 – 1047, 2022.
- [43] R. Schwartz, O. Tsur, A. Rappoport, and M. Koppel, in *Proceedings of the 2013 Conference on Empirical Methods in Natural Language Processing, Association for Computational Linguistics*, pp. 1880 – 1891, 2013.
- [44] S. M. Nirkhi, and R. V. Dharaskar, “Comparative Study of Authorship Identification Techniques for Cyber Forensics Analysis” in *International Journal of Advanced Computer Science and Applications*, pp. 32 – 35, 2013.
- [45] A. Abbasi, and H. Chen, “A Stylometric Approach to Identity-level Identification and Similarity Detection” in *ACM Transactions on Information Systems*, pp. 1 – 29, 2008.
- [46] S. Argamon, C. Whitelaw, P. Chase, S. Hota, N. Garg, and S. Levitan, “Stylistic Text Classification using Functional Lexical Features” in *Journal of the American Society for Information Science and Technology*, pp. 802 – 821, 2007.
- [47] H. Baayen, H. van Halteren, A. Neijt, and F. Tweedie, “An Experiment in Authorship Attribution” in *Journees internationales d'Analyse statistique des Donnees Textuelles*, 2002.
- [48] H. Baayen, H. van Halteren, A. Neijt, and F. Tweedie, “Outside the Cave of Shadows: Using Syntactic Annotation to Enhance Authorship Attribution” in *Literary and Linguistic Computing*, pp. 121 – 131, 1996.

-
- [49] E. Brill, “A Simple Rule-based Part-of-speech Tagger” in *Proceedings of the 3rd Conference on Applied Natural Language Processing, East Stroudsburg, PA: ACL*, pp. 152 – 155, 1992.
- [50] R. D. Peng, and H. Hengartner, “Quantitative Analysis of Literary Styles” in *The American Statistician*, vol. 56, no. 3, pp. 15 – 38, 2007.
- [51] A. S. Romanov, M. I. Vasilieva, A. V. Kurtukova, and R. V. Meshcheryakov, “Sentiment Analysis of Text Using Machine Learning Techniques” in *Proceedings of the 2nd International Conference, R. Piotrowski’s Readings LE & AL, Saint Petersburg, Russia*, pp. 86 – 95, 2020.
- [52] A. Khomenko, Y. Baranova, A. Romanov, and K. Zadvornov, “Linguistic Modeling as a Basis for Creating Authorship Attribution Software” in *Proceedings of the Computational Linguistics and Intellectual Technologies Dialogue, Moscow, Russia*, pp. 1063 – 1074, 2021.
- [53] H. Baayen, H. van Halteren, A. Neijt, and F. Tweedie, “An Experiment in Authorship Attribution” in *6es Journées Internationales d'Analyse Statistique de Données Textuelles*, pp. 69 – 79, 2002.
- [54] J. J. Alviar, “Recent Advances in Computational Linguistics and their Application to Biblical Studies”, in *New Testament Studies*, pp. 139 – 150, 2008.
- [55] K. Luyckx, W. Daelemans, “Authorship Attribution and Verification with Many Authors and Limited Data” in *Proceedings of 22nd International Conference on Computational Linguistics*, vol. 1, pp. 513 – 520, 2008.

