Integrating Stylometry with DeBERTa for Neural Authorship Attribution

Tanatip Timtong

Master of Science

Computer Science

School of Informatics

University of Edinburgh

2024

Abstract

Neural authorship attribution, a subproblem of artificial text detection, involves identifying the LLM that generated a given text. This task is typically solved by training a classifier on a corpus of text generated by multiple LLMs. Through this training, it learns to extract the features that distinguish each LLM's writing style. Despite these efforts, the subtle nuances in these styles make it difficult to achieve good performance. This paper's main aim is to improve the robustness of these classifiers by incorporating stylometric features. These features are robust signals for this task because they tend to remain consistent across text generated by the same LLM [1, 2]. We trained a novel model based on DeBERTa [3] that uses both textual and stylometric features to attribute text to the correct LLM that generated it. Our model achieved state-of-the-art performance on the proposed dataset, with a test accuracy and an F1-score of 83.1% and 0.825 respectively. Secondly, we analysed the writing styles of LLMs using SHAP [4] to identify the most distinctive linguistic features that characterise each LLM's writing style. Our analysis revealed that most LLMs follow a systematic approach to sentence composition. These findings establish a foundation for developing more effective text attribution methods and provide valuable insights into the artificial cognitive processes underlying natural language generation.

Research Ethics Approval

This project was planned in accordance with the Informatics Research Ethics policy. It did not involve any aspects that required approval from the Informatics Research Ethics committee.

Declaration

I declare that this thesis was composed by myself, that the work contained herein is my own except where explicitly stated otherwise in the text, and that this work has not been submitted for any other degree or professional qualification except as specified.

(*Tanatip Timtong*)

Acknowledgements

I would like to express my gratitude to those who have contributed to the success of this project. First and foremost, I extend my sincere thanks to Professor Rik Sarkar for his invaluable guidance and expertise throughout the project. Additionally, I am deeply grateful to my mother, brother, partner, and friends for their unwavering love and support during this time.

Table of Contents

Chapter 1

Introduction

Recent advancements in large language models (LLMs) have enabled the generation of text that closely resembles human writing. As LLMs continue to improve, there is an increased risk that artificially generated text may be mistaken for human-written ones. This raises concerns about the potential misuse of LLMs to generate misinformation. For instance, adversaries can use these models to create and spread large volumes of artificial fake news at little cost [5, 6]. This highlights the need to develop methods that can detect artificial text.

Even for some security applications such as copyright protection, distinguishing between artificially generated and human-written text may not be enough. A more critical solution would be to identify the LLM that generated the artificial text. This process, known as neural authorship attribution [7], can improve accountability and help uncover the characteristics of malicious actors who exploit these models.

Neural authorship attribution is commonly addressed by training a classifier on a corpus of text generated by multiple LLMs. During this process, the classifier learns to accurately identify the LLM that generated a given text. It does so by extracting the features that discriminate each LLM's writing style. This task is particularly challenging because the LLMs' writing styles are typically subtle and nuanced, especially when they use similar architectures. Furthermore, LLMs can adopt different writing styles depending on the prompts, further complicating the task. These factors hinder the classifier's ability to consistently differentiate between the writing styles of the LLMs, negatively impacting performance.

This paper's primary goal is to improve the robustness of classifiers for neural authorship attribution by incorporating stylometric features. Stylometric features are quantifiable stylistic characteristics of a text that can be analysed to determine an

author's writing style [8]. Traditional classifiers for this task heavily relied on stylometric features [9, 10, 11]; however, recent advancements have shifted focus towards transformer-based models, which instead uses learned textual features [12, 13]. Despite this trend, stylometric features remain robust signals for distinguishing between the writing styles because they are often consistent across texts produced by the same author [1, 2]. Therefore, these features are still valuable for neural authorship attribution, and we hypothesise that combining them with transformer-based models may improve their effectiveness in learning the writing styles of LLMs.

Our secondary goal is to conduct a preliminary study analysing the features that best characterise each LLM's writing style. Since a classifier for neural authorship attribution must learn to extract the distinctive linguistic features of an author's writing style, we hypothesise that these features can be identified by evaluating each feature's contribution to our model's ability to predict a particular author. A simple metric for measuring this contribution is feature importance. The more important a feature is, the greater its impact on the model's performance in correctly attributing texts to that author, suggesting that it is likely a key discriminative characteristic of the author's writing style. The main research questions addressed in this work are :

1. Does the incorporation of stylometric features improve the performance of classifiers for neural authorship attribution?

2. Does our model perform better than the current state-of-the-art classifier for neural authorship attribution?

3. What linguistic features are useful for distinguishing between the writing styles of different LLMs?

Our contributions are as follows. Firstly, in response to the lack of a high-quality dataset for neural authorship attribution, we developed a novel dataset that combines both stylometric and textual features. The initial corpus was sourced from MAGE [14], a comprehensive testbed for deepfake text detection. This dataset contains 15k news articles written by humans or generated by one of 8 state-of-the-art LLMs. Each news article is represented by its content and 57 stylometric features that have been standardised using a robust scaler. We used three types of stylometric features: lexical, syntactical and structural, each capturing a different linguistic aspect of the text.

Secondly, we introduced a novel model DeBERTa$_{stylo}$ that combines stylometric features with the learned textual features from DeBERTa [3] for neural authorship

attribution. Using the proposed dataset, we conducted an ablation study to select the optimal hyperparameters for our model. The hyperparameters we tuned included the number of unfrozen layers in DeBERTa, the learning rate, weight decay, and the dropout inclusion rate. The best-performing configuration, as determined by the validation accuracy, featured a learning rate of 5×10^{-5}, a weight decay of 0.05, a dropout inclusion rate of 0.9, and 1 unfrozen layer. This setup achieved a validation accuracy and F1-score of 80.2% and 0.789 respectively. With these hyperparameters, our model achieved a 1.1% increase in test accuracy over the original DeBERTa model , and an improvement of at least 8.3% in test accuracy compared to the current state-of-the-art classifiers such as RoBERTa [15] and BERT [16]. These findings suggest that incorporating stylometric features improves the classifier's performance for neural authorship attribution. Furthermore, DeBERTa's architecture is inherently better at capturing the subtle nuance of the author's writing style than other masked language modelling architectures.

Finally, we identified the linguistic features that distinguish the writing style of the LLMs in our dataset. Building on a similar work [17], we applied SHAP [4] on our best-performing model to calculate the importance of every feature with respect to each LLM. Among the eight LLMs in our dataset, only two have writing styles that are characterised solely by their semantics. This highlights the relatively powerful generative capabilities of these models, as their writing styles cannot be fully captured by stylometric features alone. In contrast, the remaining LLMs exhibit a systematic approach to sentence construction. The syntactic features were the most effective for discriminating between their writing styles, consistently showing the highest feature importance across these models.

The paper is organised as follows: Section 2 covers the related work and background knowledge required to understand the paper. Section 3 discusses problem formulation, including task definition, dataset creation, and evaluation metrics. Section 4 describes our methodology, while Section 5 discusses the experimental setup and results. Finally, in Section 6, we conclude the paper and make suggestions for future research.

Chapter 2

Related Work

In this chapter, we first present a brief overview of current natural language generation technologies and explain their underlying mechanisms. We then review the literature on authorship attribution and identify its relationship with neural authorship attribution. Following this, we discuss the progress made in solving neural authorship attribution. Finally, we provide a detailed overview of the transformer architecture, which serves as the foundation for both our model and the current state-of-the-art classifier for this task.

2.1 Natural Language Generation

Natural language generation (NLG) is a broad term that refers to AI techniques for producing high-quality, human-like text in natural language [7]. It includes tasks like text summarisation, machine translation, and open-ended text generation [18, 19]. In the context of neural authorship attribution, we concentrate on open-ended text generation. This task can be solved with three main approaches: non-neural, non-transformer neural, and transformer methods [20].

Early approaches to natural language generation used non-neural methods, which were primarily rule-based. Hidden Markov Models (HMMs) [21] and reinforcement learning [22] were commonly featured in this paradigm. Non-transformer neural methods have also been demonstrated to be highly effective for this task. Previous research explored various recurrent neural architectures, such as long short-term memory (LSTM) [23] and gated recurrent units (GRUs) [24]. However, these architectures are prone to the vanishing gradient problem, which limits their ability to generate coherent and contextually accurate text [25]. More recent work has addressed this limitation using self-attention in the transformer architecture [26]. Consequently, transformer

language models currently represent the state-of-the-art in natural language generation across different tasks. Among the transformer language models, the unidirectional GPT-2 [27] and GPT-3 [28] have received the most attention for natural language generation because of their superior performance in both conditional and unconditional open-ended text generation. Other transformer models that performed well on this task include T5 [29], GPT-J [30], GPT-Neo-X [31] and LLama [32].

In transformer language models, text is generated using left-to-right decoding. Decoding is the task of choosing a token to generate based on the probabilities that the model assigns to the possible tokens [33]. At each decoding step, the next token y_t is sampled from the probability distribution of all possible next tokens. This distribution is conditioned on both the previously decoded tokens and the input sequence. The decoding process continues iteratively, generating tokens until either a stop token is encountered or a predefined maximum length is reached [34].

Suppose the current time step is t, the previous output sequence is $Y_{t-1} = \{y_1, y_2, ..., y_{t-1}\}$, and the input sequence is $X_N = \{x_1, x_2, ..., x_N\}$. The predicted next token y_t is given by the Equation 2.1. Here, h_t is the hidden state of the model at time step t, and w_o is the output matrix. The softmax function is used to generate a probability distribution over the model's vocabulary.

$$y_t \sim P(y_t|Y_{t-1}, X_N) = softmax(w_o \cdot h_t) \tag{2.1}$$

2.2 Authorship Attribution

Authorship Attribution (AA) involves identifying the author of a given text from a pool of potential authors based on their distinct writing style [7]. Formally, a writing style refers to the distinctive manner in which an entity expresses thoughts through language [35]. It encompasses the entirety of the author's word choices, sentence structures, and use of literary devices.

Numerous studies have addressed the AA problem with feature extraction techniques such as n-grams [36, 37], topic modelling [38], LIWC [39], POS-Noise [40] and POS tags [41]. In a separate study, Zheng et al. used stylometric features (i.e., lexical, syntactic and structural features) to capture an author's writing style [42]. Their findings revealed that structural features were most effective for authorship attribution. Furthermore, Shao et al. used readability scores as features and produced impressive results, highlighting the importance of readability in distinguishing between authors [43].

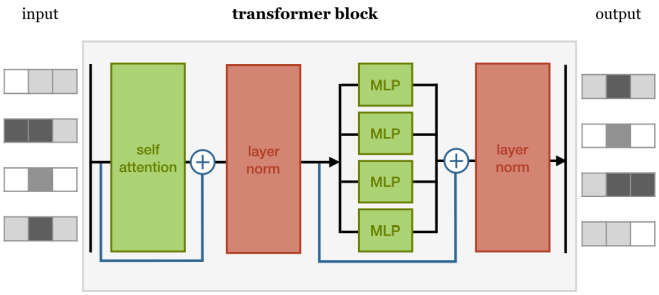


Figure 2.1: Structure of a transformer block [50]

Several classical machine-learning classifiers, including Naive Bayes [44], SVM [45], Random Forest [46], and KNN [47], have been applied to the AA problem. However, with the recent advancements in neural networks, CNN and RNN have shown to be more effective at representing the characteristics of an author's writing style [41, 48].

With the introduction of transformer language models, a new type of author, known as neural authors, has emerged in the AA landscape. Early attempts to attribute texts to neural authors were based on stylometric and statistical features [8, 9]. However, recent research has shifted towards fine-tuning pre-trained language models, as this method has demonstrated superior performance [12, 13]. Among these models, RoBERTa [15] and BERT [16] stand out as the state-of-the-art for this task [49]. Both models are based on the transformer architecture [26]. Therefore, to better understand how these models work, we will detail the fundamental principles of this architecture in the next section.

2.3 Transformer Architecture

A transformer is designed to capture the context of each token in an input sequence $(\mathbf{x}_1, ..., \mathbf{x}_N)$ using a stack of transformer blocks, each of which is a multilayer neural network. These blocks work together to develop richer contextualised representations of the tokens' meanings [33]. Both the RoBERTa [15] and BERT [16] base models contain 12 transformer blocks. Within each block, a self-attention layer is applied first, followed by a pointwise feed-forward layer, which is a single MLP applied to each vector individually. Residual connections and layer normalisations are applied after each layer. Figure 2.1 depicts this structure.

2.3.1 Text Preprocessing

Before processing an input text, the transformer first tokenises it into a sequence of N tokens. After this, it computes the input embedding for each token by combining their token and absolute positional embeddings. These input embeddings $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N) \in \mathbb{R}^{N \times d}$ are then used as the input to the transformer's blocks.

A token embedding is a d-dimensional vector that captures the semantics of a token, regardless of its context. These static embeddings are indexed from an embedding matrix $\mathbf{E} \in \mathbb{R}^{|V| \times d}$, which stores the token embeddings for all the tokens in the model's vocabulary. In contrast, an absolute positional embedding is a d-dimensional vector that encodes a token's position within the input sequence. These embeddings are usually learned alongside the token embeddings during pre-training, or they can be computed using a static function $f : \mathbb{N} \to \mathbb{R}^k$ that maps positions to real-valued vectors of dimension k . The function must be well chosen to ensure it can capture the inherent relationships between the positions.

RoBERTa [15] and BERT [16] share similar text preprocessing steps. Both models compute the input embeddings by summing the token embedding with the absolute positional embedding. They also use learned position embeddings with a maximum sequence length of 512 tokens. However, they slightly differ as RoBERTa utilises a larger vocabulary size of 50k tokens compared to BERT's 30k tokens. Furthermore, BERT uses WordPiece tokenisation [51], whereas RoBERTa uses byte-pair encoding (BPE) tokenisation [52].

2.3.2 Self Attention

The self-attention layer uses the self-attention mechanism to map a sequence of input vectors $\mathbf{X} = (\mathbf{x}_1, ..., \mathbf{x}_N) \in \mathbb{R}^{N \times d}$ to an output sequence of the same length $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_N) \in \mathbb{R}^{N \times d}$, where d denoting the embedding dimension [33]. This mapping is performed such that each output vector has been contextualised using information from the input sequence.

The fundamental operation of self-attention involves comparing an attended token $\mathbf{x}_i$ to all other tokens $\mathbf{x}_j$ in the input sequence to determine their relevance in the current context. These comparisons are then used to compute the output vectors $\mathbf{y}_i$ from the input sequence. During the attention process, each input vector $\mathbf{x}_i \in \mathbb{R}^{1 \times d}$ serves three different roles [50]:

1. Query q: Compared to every other vector $\mathbf{x}_j$ to compute the attention weights for

its output $\mathbf{y}_i$

2. Key k: Compared to every other vector $\mathbf{x}_j$ to compute the attention weights for the other outputs $\mathbf{y}_j$

3. Value v: Used as part of the weighted sum to compute each output vector once the attention weights have been established.

To capture these roles, we first project each $\mathbf{x}_i$ into the query, key and value vectors using the respective learnable weight matrices: $\mathbf{W}_q, \mathbf{W}_k, \mathbf{W}_v \in \mathbb{R}^{d\times d}$. This computation can be performed simultaneously for all input vectors $\mathbf{X}$ using matrix multiplication to generate the query, key and value vectors $\mathbf{Q}, \mathbf{K}, \mathbf{V} \in \mathbb{R}^{N\times d}$.

Next, we compute the attention score matrix $\mathbf{S} \in \mathbb{R}^{N\times N}$ by taking the dot product between the query vector $\mathbf{q}_i$ and the key vector $\mathbf{k}_j$ for every pair of $\mathbf{x}_i$ and $\mathbf{x}_j$. These scores are then normalised using the softmax function to generate the attention weight matrix $\mathbf{A} \in \mathbb{R}^{N\times N}$, where each $\mathbf{a}_i$ is a vector of attention weights indicating the proportional relevance of other input tokens $\mathbf{x}_j$ to the attended token $\mathbf{x}_i$. Since softmax is sensitive to large input values, it can kill the gradients and slow down learning; hence, the dot products in $\mathbf{S}$ are scaled down by k to prevent the inputs to softmax from growing too large. Finally, each output vector $\mathbf{y}_i$ is computed as the weighted sum over all value vectors $\mathbf{v}$, with the weights determined by $\mathbf{a}_i$

$$\mathbf{Q} = \mathbf{X}\mathbf{W}_q; \quad \mathbf{K} = \mathbf{X}\mathbf{W}_k; \quad \mathbf{V} = \mathbf{X}\mathbf{W}_v$$

$$\mathbf{S} = \frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}; \quad \mathbf{A} = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}}) \tag{2.2}$$

$$\mathbf{Y} = SelfAttention(\mathbf{Q}, \mathbf{K}, \mathbf{V}) = softmax(\frac{\mathbf{Q}\mathbf{K}^\top}{\sqrt{d}})\mathbf{V}$$

In models that utilise causal self-attention, the upper-triangular portion of the matrix $\mathbf{S}$ is masked to eliminate information about future words. This ensures the model can only attend to input tokens that precede $\mathbf{x}_i$ when computing $\mathbf{y}_i$. In contrast, models that use bidirectional self-attention do not apply masking. This approach allows the model to contextualise each output vector using information from the entire input sequence. Both RoBERTa [15] and BERT [16] use bidirectional self-attention to compute the output of the self-attention layer.

Chapter 3

Problem Formulation

In this chapter, we first outline the task definition of neural authorship attribution. Next, we describe our data collection and preprocessing steps, followed by an exploratory data analysis of our dataset. Finally, we describe the metrics used to evaluate and compare our model's performance to the baselines.

3.1 Task Definition

Neural authorship attribution is typically framed as a multi-class classification problem, where each class represents a distinct author. Suppose we have texts generated by humans and k LLMs $\{LLM_1, ..., LLM_k\}$. In this scenario, the entity that generated a text is referred to as its author (i.e., humans or one of the k LLMs), resulting in a total of $k+1$ authors. For simplicity, we refer to the human author as LLM_0. Let $\mathcal{X} \subseteq \mathbb{R}^d$ and $\mathcal{Y} \subseteq \{0,1\}^{k+1}$ be the feature and label space respectively, where d denotes the number of features representing a given text. The label of each text $\mathbf{x} \in \mathcal{X}$ is represented by a one-hot encoding $\mathbf{y} = (y_0, ..., y_{k+1}) \in \mathcal{Y}$ such that :

$$y_i = \begin{cases} 1 & \text{if } LLM_i \text{ generated } \mathbf{x} \\ 0 & \text{otherwise} \end{cases} \tag{3.1}$$

During training, a classifier $C : \mathcal{X} \to \mathcal{Y}$ learns to define a probabilistic decision boundary that best separates each author's data points within the feature space, based on a specific loss function. Given an input text $\mathbf{x}$, the classifier assigns a probability to each author to indicate how likely they created the text. It then predicts the author of $\mathbf{x}$ based on these probabilities :

$$C(\mathbf{x}) = \underset{i \in \{0,\ldots,k\}}{\arg\max}\, p_i(\mathbf{x}) \tag{3.2}$$

where $p_i(\mathbf{x})$ denotes the probability of $\mathbf{x}$ being generated by LLM_i. The parameters of the classifiers are optimised so that the decision boundary assigns the highest probability to the correct author. During testing, the classifier applies the learned decision boundary to assign a new input text $\mathbf{x}$ to one of the known authors from the training data using the same equation.

3.2 Dataset

In this section, we provide a detailed explanation of the steps involved in creating our dataset. We begin by discussing the rationale for creating a new dataset for neural authorship attribution, highlighting the limitations of existing ones. We then explain how we gathered our initial corpus from MAGE, a comprehensive testbed for deepfake text detection [14]. Finally, we describe the preprocessing steps applied to the corpus to produce the final dataset, and then perform an exploratory data analysis on it.

3.2.1 Limitations of Existing Dataset

In the literature, only a few datasets have been specifically created for neural authorship attribution. Among these, TURINGBENCH [49] emerges as the pioneering benchmark because it has been extensively used to train and evaluate many classifiers for this task [53]. This benchmark contains news articles generated by 19 different LLMs. However, these generated articles often lack coherency, fluency and grammaticality because they contain many fragmented and nonsense words and symbols. Table 3.1 provides examples from this benchmark that illustrate these issues and highlight the poor quality of the generated text. Consequently, the dataset's effectiveness in evaluating a classifier's performance for this task is limited, underscoring the need to create a new dataset.

3.2.2 Data Collection

The MAGE testbed combines human-written text from ten benchmark datasets covering various writing tasks such as question answering and story generation [14]. It also features artificial text generated by 27 different LLMs. We chose to source our initial

News Article	Label
car bomb kills more than 100 during killing at least 50 or to::::::::::::::::	XLNet
ing treasury advantage intitutional plan-growth debt mutual fund information: scheme objective, manager, house, financial institutions, financial	GPT 2

Table 3.1: Some examples from the TURINGBENCH benchmark

corpus from this testbed for two reasons. Firstly, the paper for this testbed comprehensively details the steps involved in collecting and preprocessing the human-written and artificial text. This explanation clarifies the testbed's composition and limitations, ensuring transparency in our data collection process. Secondly, the testbed only uses state-of-the-art LLMs to generate the artificial text. Consequently, the quality of the generated text is significantly higher than that of the TURINGBENCH benchmark, demonstrating the most recent advancement in natural language generation. This ensures that our dataset remains relevant and up-to-date for training and evaluating classifiers in neural authorship attribution.

Continuation, topical and specified prompts were used to generate the artificial text from each human-written text. The first prompt type was used with all LLMs, whereas the two last prompt types were only used with the three OpenAI models :

1. Continuation prompts: LLMs were asked to continue the generation based on the first 30 words of the original human-written text
2. Topical prompts: LLMs were asked to generate texts based on the provided topic
3. Specified Prompts: A topical prompt was provided along with metadata about the text sources

Since each author's writing style may differ depending on the writing task, we focused solely on news article writing to ensure a consistent analysis. This restriction helps normalise the linguistic features and reduce the variability introduced by task-specific conventions. Consequently, this allows their writing styles to be distinguished and assessed with greater reliability and accuracy.

To collect data for this task, Li et al. [14] sampled 1,777 human-written news from the XSum [54] and TLDR [55] datasets. Continuation prompts were then fed

to all models to generate 47,979 artificial news. Moreover, the three OpenAI models generated an additional 10,662 artificial news because they were also given topical and specified prompts. In total, this resulted in 58,641 artificially generated articles.

Our study only examined nine different authors from MAGE: humans, LLama 65B [32], GPT-3.5 Turbo [28], Text Davinci-003[28], Flan-T5 XXL [29], GPT-J [30], GPT-NeoX [31], OPT 30B [56], BigScience T0 11B [57]. These LLMs were selected because they are the most prominent among all models. Furthermore, we only included artificial news articles generated with the continuation prompt to ensure a consistent text generation style across all models.

Li et al. [14] normalised the punctuation and removed line breaks to reduce the effects beyond the text content. They also removed articles that were either too long or too short. Following this, we refined the dataset to include only the articles created by our chosen authors while excluding those generated by topical and specified prompts. This process resulted in a final dataset comprising 15,353 articles generated by nine different authors. Table 3.2 shows a few chosen examples from our dataset.

News Article	Label
Most flights coming out of or landing in Chile's main airports were cancelled or delayed. The strike hit Chile in the busy run-up to the Christmas holiday period, leaving thousands stranded across much...	GPT-NeoX
SVGs have cool benefits like crisp image quality with a single file, support for emojis, inline icons, and dark mode detection. This article discusses how to add SVG favicons to a project. Code and examples are provided.	Humans

Table 3.2: Some examples from our dataset

3.2.3 Data Preprocessing

Since Li et al. [14] had already conducted the majority of text preprocessing, our only additional step was whitespace reduction. We intentionally avoided standard text preprocessing practices, such as removing punctuation and special characters, and text normalisation (e.g., stemming and lowercasing). This decision was made to preserve the stylometric features that inherently contribute to the author's unique writing

style. Performing these practices would result in the loss of valuable information that could compromise the authenticity of the writing style and the performance of our model. Additionally, each article was labelled with a string representing its author. To standardise the labels into a format our model can understand, we converted each label into a one-hot encoding vector using Equation 3.1.

Next, we extracted 57 stylometric features from the preprocessed news articles. Stylometric features are used to identify the different stylistic signals present in a given text [8]. Detailed information on how each feature was calculated and what it measures will be further discussed in the Methodology section. There are three main types of stylometric features that we used:

1. Lexical Features: These characterise the author's writing style at the word level, capturing the meaning and usage of words.

2. Syntactical Features: These characterise the author's writing style at the sentence level, capturing patterns from sentence structure and the arrangement of words and phrases to create grammatically correct sentences.

3. Structural Features: They characterise the author's writing style level at the paragraph level, capturing the overall organisation and format of the text.

We divided the dataset into the train, validation and test sets with an 80%/10%/10% split using stratified sampling and a seed of 2543673. This sampling method was chosen to ensure each set contained roughly the same proportion of articles for each author.

Some stylometric features have highly varying ranges with many outliers. For example, the Flesch Reading Ease score [58] ranges from 0 to 100, whereas Gunning Fog Index [59] ranges from 0 to 20. This discrepancy can cause our model to give more weight to features with a larger scale, introducing feature bias and negatively impacting performance. Hence, we used a robust scaler to standardise each feature so that they have comparable ranges, ensuring all features contribute equally to our model.

A robust scaler standardises each feature using its median and interquartile range (IQR). The IQR is defined as the difference between the 25th percentile and the 75th percentile of the feature values. By using the median and IQR, this method is robust against outliers because it prevents them from disproportionately affecting the scaling process. Consequently, the scaled data remains more representative of the majority of the feature values.

We started by fitting the scaler on the training data to calculate the medians and IQRs of each feature. After this, these values were used to transform the feature values

Author	Train Instances	Validation Instances	Test Instances
Humans	1,437 (11.6%)	160 (11.6%)	178 (11.6%)
LLama 65B	1,420 (11.4%)	158 (11.4%)	175 (11.4%)
GPT-3.5 Turbo	1,436 (11.5%)	159 (11.5%)	178 (11.6%)
Text Davinci-003	1,436 (11.5%)	159 (11.5%)	177 (11.5%)
Flan-T5 XXL	1,418 (11.4%)	158 (11.4%)	175 (11.4%)
GPT-J	1,216 (9.8%)	135 (9.8%)	150 (9.8%)
GPT-NeoX	1,225 (9.9%)	136 (9.8%)	151 (9.8%)
OPT 30B	1,411 (11.3%)	157 (11.4%)	174 (11.3%)
BigScience T0 11B	1,436 (11.5%)	160 (11.6%)	178 (11.6%)
Total	12,435 (100%)	1382 (100%)	1536 (100%)

Table 3.3: Author instances distribution in the train, validation and test set

in the training, validation, and test set. Only the training data was used to fit the scaler to prevent data leakage. If the scaler was also fitted on both the validation and test data, information from these datasets would leak into the training data via the medians and IQRs. This leakage could negatively impact the training process, resulting in an overly optimistic performance and compromising the model's reliability and accuracy.

Let $\mathbf{x}_{median} \in \mathbb{R}^{1\times d}$ and $\mathbf{x}_{IQR} \in \mathbb{R}^{1\times d}$ denote the vectors containing the median and IQR of all features, computed from the training data, respectively. We can then scale a feature vector $\mathbf{x} \in \mathbb{R}^{1\times d}$ as follows:

$$\mathbf{x}_{new} = \frac{\mathbf{x} - \mathbf{x}_{median}}{\mathbf{x}_{IQR}} \tag{3.3}$$

3.2.4 Exploratory Data Analysis

After creating our dataset, we conducted exploratory data analysis to uncover underlying patterns and insights in the data. Firstly, we analysed the distribution of the authors in the training, validation and test sets. Table 3.3 shows the proportion of each author across these sets. The roughly equal representation of each author across all sets ensures that the model is trained and evaluated on a representative sample of the data. Furthermore, having an equal proportion of each author in the training set prevents the model from becoming biased towards any single author during training.

Secondly, we visualised the contextual embeddings generated by DeBERTa for all articles in our dataset. To achieve this, we first extracted the embeddings of the [CLS]

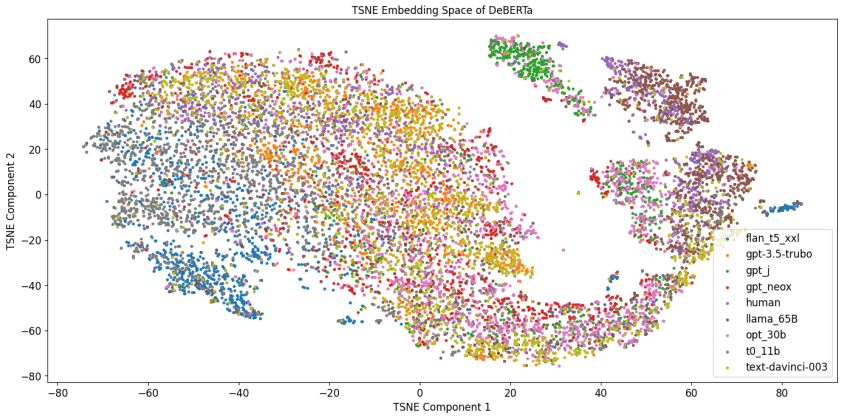
token for each article, which captures the summary representation of the entire text. We then projected them into a 2D vector space using t-SNE. Additionally, we visualised the stylometric features of all articles by applying t-SNE to project them into a 2D vector space as well.

Figure 3.1a shows the embedding space of the articles in our dataset. The article embeddings for some authors, such as Flan-T5 XXL [29] , GPT-J [30] and LLama 65B [32], are well-separated in the embedding space, forming distinct clusters. This implies that sentence embeddings alone are sufficient to characterise the writing style of these models, as they can be used to accurately classify the articles written by these authors. In contrast, the embeddings for the remaining authors were dispersed across a single, massive cluster. This distribution suggests that while articles written by the same author show significant variation in embedding patterns, these representations remain similar to those of other authors in the cluster. As a result, sentence embeddings may not be reliable features for distinguishing between these authors.

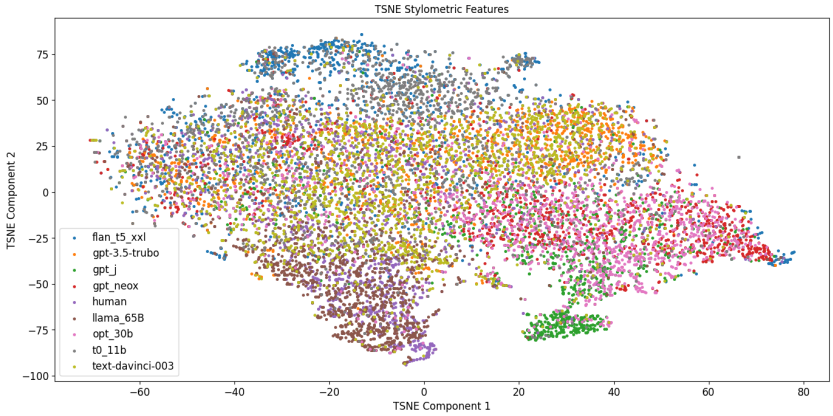
Figure 3.1b illustrates the projected stylometric features of the articles in our dataset. In comparison to the article embeddings produced by DeBERTa, the stylometric features for each author are less distinctly separated, forming one massive cluster. Within this cluster, the stylometric features of authors such as Text-Davinci-003 [28], OPT 30B [56], LLama 65B [32], GPT-J [30], Flan-T5 XXL [29], and BigScience T0 11B [57] are relatively close together, creating subclusters. This suggests that although the writing style of different authors may be more difficult to distinguish based on stylometric features, there is minimal variation in these features across articles produced by the same author. Consequently, this shows that the stylometric features are relatively consistent for the same author, serving as a strong signal for neural authorship attribution.

3.3 Evaluation Metric

We evaluated our model's performance primarily using accuracy. This metric was chosen because it is more intuitive and easier to interpret than other classification metrics. Given that our dataset is balanced, accuracy provides a meaningful and reliable measure of our model's performance without introducing bias toward any particular author. However, when evaluating a multi-class classification problem, relying solely on accuracy has a limitation because it does not reveal whether all authors were predicted equally well. Some authors might be accurately predicted, while others could be neglected. To address this issue, we also monitored the F1 score to provide a more



(a) DeBERTa's embedding space



(b) Stylometric features

Figure 3.1: t-SNE plots

comprehensive understanding of the model's performance in predicting each author.

3.3.1 Accuracy

Intuitively, accuracy is a metric ranging from 0 to 1 that measures the proportion of news articles that have been correctly attributed to their respective authors. The higher the accuracy, the better the performance of our model. Given a 9×9 confusion matrix M , where M_{ij} represents the number of articles whose actual and predicted author was LLM_i and LLM_j respectively. We can compute accuracy as follows:

$$Accuracy(M) = \frac{\sum_{i=0}^{8} \sum_{j:i=j} M_{ij}}{\sum_{i=0}^{8} \sum_{j=0}^{8} M_{ij}} \tag{3.4}$$

3.3.2 F1-Score

The F1-score is derived from precision and recall, so it is essential to define these two metrics first. For each author LLM_i, precision measures the ratio of the correctly attributed articles to the total number of articles that have been attributed to that author by our model. Intuitively, it tells us the quality of the model's prediction, reflecting how much confidence we can place in the model when it attributes an article to the author. A high precision indicates a low false positive rate for that author. The best possible precision score is 1, while the worst score is 0.

On the other hand, recall measures the ratio of the correctly attributed articles to the total number of articles generated by that author. Intuitively, recall measures the ability of our model to find all articles created by that author. A high recall indicates that the model can identify most of the articles generated by that author, indicating a low false negative rate. The best possible recall score is 1, while the worst score is 0.

Given the confusion matrix M_i for the author LLM_i, containing the true positive (TP_i), true negative, (TN_i), false positive (FP_i) and false negative (FN_i) observations. We can compute the precision and recall for the author as follows:

$$\text{Precision } (P_i) = \frac{TP_i}{TP_i + FP_i}; \quad \text{Recall } (R_i) = \frac{TP_i}{TP_i + FN_i} \tag{3.5}$$

The F1-score calculates a simple weighted average of the precision and recall, taking both the false positive and false negative rates into account. An F1 score achieves its best value at 1, indicating perfect precision and recall, and the worst score at 0. Given an author LLM_i, the F1-score of the author can be computed as follows:

$$\text{F1-Score } (F1_i) = \frac{2P_iR_i}{P_i + R_i} \tag{3.6}$$

To calculate the F1 score for a multi-class dataset, we first compute the individual F1 scores for each author using the one-vs-all approach. Once we have these individual scores, we can average them to get a single metric for our model's overall performance. There are several methods for calculating this average, but we chose macro-averaging. This method is especially appropriate for our dataset because it treats all classes as equally important, which is ideal given that our dataset contains approximately equal numbers of articles for each author. The macro F1 score is calculated as follows:

$$\text{Macro F1-Score} = \frac{1}{9}\sum_{i=0}^{8} F1_i \tag{3.7}$$

Chapter 4

Methodology

This chapter outlines our paper's methodological framework. We begin by discussing how we computed the stylometric features from the news article. Next, we describe the architecture of our proposed model for neural authorship attribution, as well as the ablation study that will be conducted on it. Finally, we discuss our approach to analyse the writing style of the authors in our dataset. Figure 4.1 shows our workflow's pipeline.

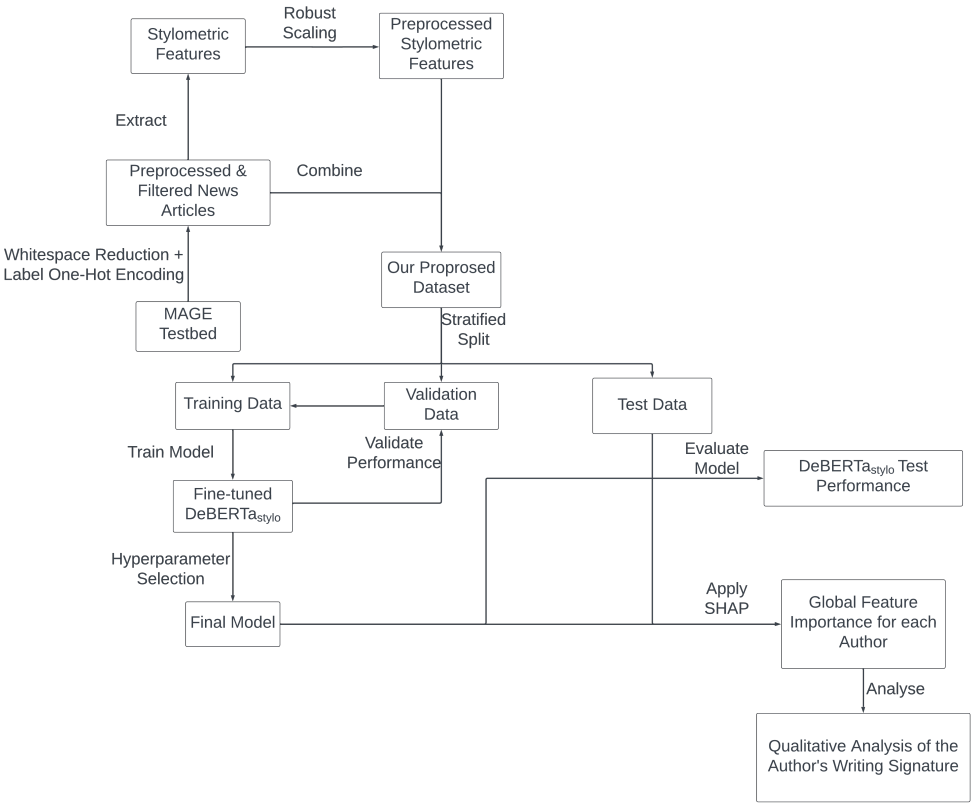


Figure 4.1: Workflow pipeline for our project

4.1 Stylometric Features Extraction

Stylometric features were widely used in traditional classifiers for authorship attribution. Early efforts to solve this task focused on training ensembles of machine learning models on extracted stylometric features [9]. Later work adopted convolutional and recurrent neural networks to perform authorship attribution using these same features [10, 11]. However, recent approaches have shifted towards transformer-based models due to their superior performance [12, 13]. Unlike traditional classifiers, these models rely on learned textual features for more accurate attribution rather than manually engineered features.

Despite this trend, stylometric features continue to serve as robust signals for distinguishing between different authors' writing styles, as they tend to remain consistent across texts produced by the same author [1, 2]. Hence, these features are still valuable for neural authorship attribution , and we hypthosise that combining them with a transformer-based model can improve its ability to differentiate between the writing styles. Our hypothesis is supported by a previous work that has combined stylometric features with the textual features learned by RoBERTa for this task [12]. Their results have demonstrated that incorporating both types of features improved the classifier's performance compared to using only textual features. However, the specific features used and their computation methods have not been clearly outlined, making it challenging for other researchers to understand and evaluate the effectiveness of this approach.

To improve transparency, this section outlines the stylometric features that we have used and explains how they are computed for each news article. There are 57 stylometric features in total, which can be divided into three categories: lexical, syntactical, and structural. Each category captures a different linguistic aspect of a text, as detailed in the previous chapter. We first describe the lexical features, followed by the syntactical features, and finally the structural features.

4.1.1 Lexical Features

Table 4.1 shows the 16 lexical features we extracted from the news articles. Computing the first 11 lexical features was trivial. The NLTK's list of English stopwords [60] and the Python library "functionwords" [61] were used to identify all the stopwords and function words in the article respectively, while SpaCy [62] was used to count the syllables of each word.

Function words express grammatical relationships among other words in the same

Feature ID	Feature Name
1	Word Count
2	Stopword Count
3	Function Word Count
4-5	Average Character Per Word (Mean + Std)
6-7	Average Syllable Per Word (Mean + Std)
8-9	Average Stopword Per Sentence (Mean + Std)
10-11	Average Function Word Per Sentence (Mean + Std)
12	Hapax Legomenon Count
13	Hapax Dislegemena Count
14	Honore's Measure [63]
15	Sichel's Measure [58]
16	Moving Average Type-Token Ratio (MATTR) [64]

Table 4.1: Lexical features extracted from the news articles

sentence, and analysing their usage can reveal patterns in the sentence complexity and structure. Similarly, stopwords are a set of commonly used words such as "I", "you", and "that", and analysing their usage can reveal the stylistic preference and writing habits of the authors. Both the average syllable and number of characters per word measure the complexity of the vocabulary, which can indicate the level of sophistication of the writing.

The remaining lexical features measure vocabulary richness, which indicates the diversity of words used in a text. A text with low vocabulary richness uses few words that are frequently repeated, while a text with high vocabulary richness continually introduces new words. Hapax legomenon are the words that pnly appear once in a text, whereas Hapax dislegemena are the words that appear exactly twice in the text.

Let N, V, V_1, V_2 denote the number of words, word types, Hapax legomenon and Hapax dislegemena in an article respectively. We can compute the Honore's measure R [63] and Sichel's measure S [58] of the article as follows:

$$R = 100 \times \log\left(\frac{N}{1 - (\frac{V_1}{V})}\right) \tag{4.1}$$

$$S = \frac{V_2}{V} \tag{4.2}$$

The type-token ratio (TTR) [65] is another useful metric for assessing vocabulary

diversity. However, its value is significantly affected by the text's length, with shorter texts generally producing higher TTR. Given an article, we can compute its TTR as follows:

$$TTR = \frac{V}{N} \tag{4.3}$$

We can use the moving-average type-token ratio (MATTR) [64] to address this issue. MATTR uses a moving window of 50 words to calculate the TTRs, then averages these values to obtain the final result. Let $TTR_1, ..., TTR_W$ denote the TTRs computed from an article using W different windows, its MATTR can be calculated as follows:

$$MATTR = \frac{1}{W} \sum_{i=1}^{W} TTR_i \tag{4.4}$$

4.1.2 Syntactical Feature

Table 4.2 shows the 36 syntactical features we extracted from the news articles. The average word per sentence reveals the author's typical sentence structure and the complexity of their syntactical constructions. Some authors prefer concise sentences, whereas others prefer more complex and longer sentences. Additionally, authors have different preferences for grammatical constructs, such as the use of passive vs active voices and complex noun phrases. These preferences are reflected in the sentence length and inherently contribute to the author's writing style.

We used SpaCy [62] to calculate the average number of tags per sentence for each POS in the universal POS tagset [66]. These features provide insights into the typical grammatical composition of sentences. Different writing styles may have distinctive POS tag distributions. For instance, frequent use of adjectives and adverbs may indicate a more descriptive style, while a predominant use of nouns and verbs could indicate a more direct and formal style.

Feature ID	Feature Name
17-18	Average Word Count Per Sentence (Mean + Std)
19-52	Average POS Tags Count Per Sentence (Mean + Std)

Table 4.2: Syntactical features extracted from the news articles

4.1.3 Structural Feature

Table 4.3 shows the five structural features we extracted from the news articles. Calculating the sentence and punctuation count was straightforward. Both features provide insight into how an author organises their ideas in the writing. Some authors may present information using long, complex sentences, resulting in a low sentence and punctuation count. In contrast, others might break their points into multiple, simpler sentences, resulting in a higher sentence and punctuation count.

Feature ID	Feature Name
53	Sentence Count
54	Punctuation Count
55	Flesh Reading Ease (FR Score) [58]
56	Flesh-Kincaid Grade Level (FKG Score) [67]
57	Gunning Fog Index [59]

Table 4.3: Structural features extracted from the news articles

The remaining structural features measure readability. The readability of a text refers to how easily a reader can understand its content. We discuss three metrics for measuring readability, each of which employs different methods to calculate the score. First, the Flesch Reading Ease [58] measures readability on a scale from 1 to 100, where a higher score indicates better readability. For instance, a score of 100 implies that the text is straightforward, whereas a score between 0 and 30 indicates that the content is more suited for university-level reading. Given an article, we can compute its Flesh Reading Ease score as follows:

$$\text{FR Score} = 206.835 - 1.015 \times (\frac{\text{Total Words}}{\text{Total Sentences}}) - 84.6 \times (\frac{\text{Total Syllables}}{\text{Total Words}}) \quad (4.5)$$

Second, the Flesch-Kincaid Grade Level [67] measures readability in terms of the US education grade level needed to understand the text. This metric ranges between 0 to 18, with higher numbers indicating a more difficult text. The Flesh-Kincaid Grade Level of an article can be calculated as follows:

$$\text{FKG Score} = 0.39 \times (\frac{\text{Total Words}}{\text{Total Sentences}}) + 11.8 \times (\frac{\text{Total Syllables}}{\text{Total Words}}) - 15.59 \quad (4.6)$$

Finally, the Gunning Fox index [59] measures readability based on the years of education required to understand a text. The index ranges from 0 to 20, with a higher number implying a more difficult text. By defining complex words as those containing three or more syllables, we can compute the Gunning Fox index of an article as follows:

$$G = 0.4 \times ((\frac{\text{Total Words}}{\text{Total Sentences}}) + 100 \times (\frac{\text{Complex Words}}{\text{Total Words}})) \tag{4.7}$$

4.2 Model Architecture & Training

Current state-of-the-art classifiers for neural authorship attribution [7, 49] predominantly rely on fine-tuning RoBERTa [15] and BERT [16]. Although these models are still effective, they are relatively older in the field of NLP. Recent advancements in NLP technologies have introduced numerous new architectures designed to learn and understand human language more effectively than these earlier models. This presents an opportunity to improve the classifier's performance by utilising these more powerful architectures. In this paper, we propose a novel model that integrates stylometric features with DeBERTa [3], a bidirectional transformer language model that improves on RoBERTa and BERT.

4.2.1 Model Architecture Overview

Given a news article, our model first prepends a special [CLS] token to its input sequence. This modified sequence is then fed into the pre-trained DeBERTa model and processed through all its layers to generate a d-dimensional output vector for each input token. The output vector corresponding to the [CLS] token $\mathbf{x}_{cls} \in \mathbb{R}^{1\times d}$ acts as the sentence embeddings, capturing the representation of the entire sequence. These embeddings represent the textual features of an article.

Next, the model generates the final set of linguistic features for the article $\mathbf{x}_{linguistic} \in \mathbb{R}^{1\times(d+57)}$ by concatenating the sentence embeddings with its 57 stylometric features $\mathbf{x}_{stylo} \in \mathbb{R}^{1\times 57}$. This resulting vector, which captures both the article's semantic and stylistic information, is then input into the classifier head. The classifier head multiplies this vector by a set of learnable weights $\mathbf{W}_c \in \mathbb{R}^{(d+57)\times 9}$ and applies the softmax function to generate a probability distribution $\hat{\mathbf{y}} \in [0,1]^9$ over all the nine possible authors in our dataset. The values in $\mathbf{W}_c$ are learned through supervised training of our model. Each input sequence in the training data is labelled with its corresponding

author, represented as a one-hot encoding vector $\mathbf{y} \in \{0,1\}^9$. Figure 4.2 illustrates our model's architecture.

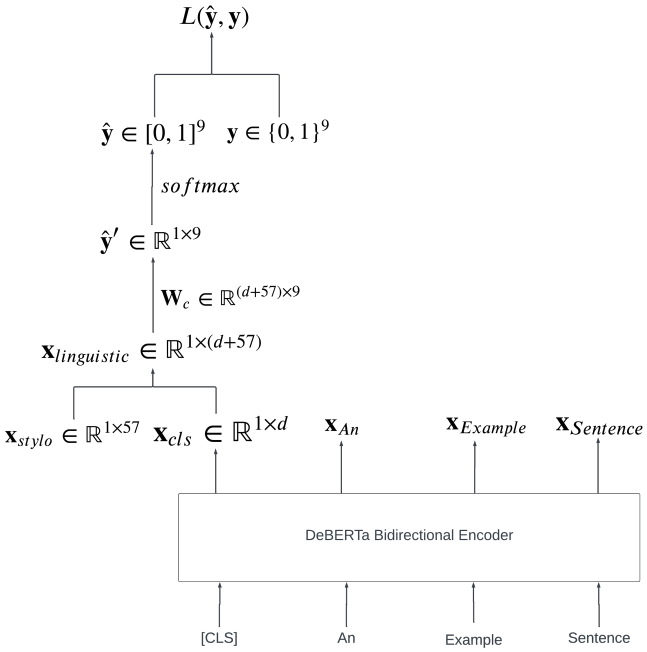


Figure 4.2: DeBERTa$_{stylo}$ model architecture

4.2.2 DeBERTa Architecture

Both RoBERTa [15] and BERT [16] encode each input token as a single vector, which is computed by summing its token embeddings with the corresponding absolute positional embeddings. This approach has a limitation: the conventional self-attention mechanism cannot differentiate whether a token's content or its absolute position contributes more to a certain embedded vector component, potentially resulting in information loss.

To overcome this limitation, DeBERTa [3] introduces the disentangled self-attention mechanism. Each token is represented by two vectors: one for its content and another for its position in the sequence. This separation enables the calculation of attention weights between the tokens using disentangled matrices that consider both their content and relative positions. Consequently, this capability improves the model's understanding of the relationship between the tokens and their positions within the sequence. Experimental results showed that DeBERTa outperformed state-of-the-art models on various NLP benchmarks, highlighting the effectiveness of this mechanism [68].

Given a token at position i in the input sequence, we represent it as two vectors $\mathbf{h}_i \in \mathbb{R}^{1\times d}$ and $\mathbf{p}_{i|j} \in \mathbb{R}^{1\times d}$. $\mathbf{h}_i$ captures the token's semantic, irrespective of its context,

while $\mathbf{p}_{i|j}$ captures its relative position to the token at position j. The attention score S_{ij} between the tokens at positions i and j is then calculated as the sum of four components using Equation 4.8. These components are derived from the disentangled matrices, and they account for the interaction between the content and position in the following ways: content-to-content, content-to-position, position-to-content and position-to-position.

$$S_{ij} = \{\mathbf{h}_i, \mathbf{p}_{i|j}\} \times \{\mathbf{h}_j, \mathbf{p}_{j|i}\}^\top = \mathbf{h}_i\mathbf{h}_j^\top + \mathbf{h}_i\mathbf{p}_{j|i}^\top + \mathbf{p}_{i|j}\mathbf{h}_j^\top + \mathbf{p}_{i|j}\mathbf{p}_{j|i}^\top \tag{4.8}$$

The content-to-content term $\mathbf{h}_i\mathbf{h}_j^\top$ captures the semantic relationship between the meaning of the tokens at both positions. The content-to-position term $\mathbf{h}_i\mathbf{p}_{j|i}^\top$ describes how the context of the token at position i relates to its relative position to the token at position j. Conversely, the position-to-content term $\mathbf{p}_{i|j}\mathbf{h}_j^\top$ maps how the context of the token at position j relates to its relative position to the token at position i. Finally, the position-position term $\mathbf{p}_{i|j}\mathbf{p}_{j|i}^\top$ describes the relationship between relative position vectors for the tokens at both positions. The last component is removed in future calculations since it does not provide valuable information on the tokens' content.

Let k be a hyperparameter that denotes the maximum relative distance, we can define the relative distance $\delta(i,j) \in [0, 2k)$ from a token i to token j as follows:

$$\delta(i,j) = \begin{cases} 0 & \text{if } i-j \leq -k \\ i-j+k & \text{if } -k \leq i-j \leq k \\ 2k-1 & \text{otherwise} \end{cases} \tag{4.9}$$

Given the content matrix of all tokens in the input sequence $\mathbf{H} \in \mathbb{R}^{N\times d}$, we can compute the disentangled self-attention score S_{ij} from the token at position i to the token at position j using Equation 4.10. $\mathbf{Q}_c, \mathbf{K}_c, \mathbf{V}_c \in \mathbb{R}^{N\times d}$ are the projected content vectors generated using the projection matrices $\mathbf{W}_{q,c}, \mathbf{W}_{k,c}, \mathbf{W}_{v,c} \in \mathbb{R}^{d\times d}$. $\mathbf{P} \in \mathbb{R}^{2k\times d}$ represent the relative position embedding vectors shared across all layers, and $\mathbf{Q}_r, \mathbf{K}_r \in \mathbb{R}^{N\times d}$ represent the projected relative position vectors generated using the projection matrices $\mathbf{W}_{q,r}, \mathbf{W}_{k,r} \in \mathbb{R}^{d\times d}$.

$$\mathbf{Q}_c = \mathbf{H}\mathbf{W}_{q,c}; \quad \mathbf{K}_c = \mathbf{H}\mathbf{W}_{k,c}; \quad \mathbf{V}_c = \mathbf{H}\mathbf{W}_{v,c}; \quad \mathbf{Q}_r = \mathbf{H}\mathbf{W}_{q,r}; \quad \mathbf{K}_r = \mathbf{H}\mathbf{W}_{k,r}$$

$$S_{ij} = \mathbf{q}_c^i \cdot \mathbf{k}_c^j + \mathbf{q}_c^i \cdot \mathbf{k}_r^{\delta(i,j)} + \mathbf{k}_c^j \cdot \mathbf{q}_r^{\delta(j,i)} \tag{4.10}$$

$$\mathbf{Y} = softmax(\frac{\mathbf{S}}{\sqrt{3d}})\mathbf{V_c}$$

$\mathbf{S} \in \mathbb{R}^{N \times N}$ stores the attention scores for all pairs of tokens in the input sequence. $\mathbf{q}_c^i$ represents the i^{th} row of $\mathbf{Q}_c$, whereas $\mathbf{k}_c^j$ represents the j^{th} row of $\mathbf{K}_c$. $\mathbf{k}_r^{\delta(i,j)}$ is the $\delta(i,j)^{th}$ row of $\mathbf{K}_r$, and $\mathbf{q}_r^{\delta(j,i)}$ is the $\delta(j,i)^{th}$ row of $\mathbf{Q}_r$. All these vectors are an element of $\mathbb{R}^{1 \times d}$. Like BERT [16], the attention scores have been scaled down by $\sqrt{3d}$ to prevent the inputs to softmax from growing too large.

4.2.3 Task Output Activation

The final layer in our model consists of nine nodes, each representing one of the nine possible authors in our training data. The outputs of this layer $z_0, ..., z_8$ are normalised using softmax to generate a probability distribution over all authors. The normalised output $\hat{y}_i$ represents the predicted probability that LLM_i generated the given article. The value of $\hat{y}_i$ can be computed as follows:

$$\hat{y}_i = softmax(z_i) = \frac{e^{z_i}}{\sum_{j=0}^{8} e^{z_j}} \tag{4.11}$$

Softmax was chosen over the sigmoid function because it ensures that the output probabilities add up to one for all authors, resulting in a valid probability distribution. In contrast, the sigmoid function treats each output probability independently and does not guarantee this property.

4.2.4 Task Loss Function

Since we use softmax for the output activation, we train our model with cross-entropy loss. This loss is ideal for multi-class classification because it quantifies the difference between the predicted and true probability distribution of the authors for a given article. The cross-entropy loss guides the learning process by optimising the parameters in $\mathbf{W}_c$ to ensure the model assigns the highest probability to the correct author. Given the predicted probabilities $\hat{\mathbf{y}}$ and ground truth label $\mathbf{y}$ of a news article, we can compute the loss for this instance as follows :

$$\hat{\mathbf{y}} = softmax(\mathbf{x}_{linguistic}\mathbf{W}_c) \tag{4.12}$$

$$L(\hat{\mathbf{y}}, \mathbf{y}) = -\sum_{i=0}^{8} y_i \log(\hat{y}_i) \tag{4.13}$$

4.3 Ablation Study

We performed an ablation study to select the optimal hyperparameters for our model. The purpose of an ablation study is to determine how removing or replacing specific parts of the model affects its performance and behaviour. By analysing the results of this study, we can identify and eliminate weaker models and choose the hyperparameter values that maximise our model's accuracy on the validation set. Several ablations were conducted on the number of unfrozen layers in DeBERTa, the learning rate, weight decay, and dropout.

4.3.1 Dropout

Dropout is a regularisation technique that mitigates the effect of overfitting [69]. It is typically represented as an additional layer inserted between the linear and activation layers. During training, it randomly deletes a fraction of the hidden units from the linear layer based on a hyperparameter called the inclusion rate (i.e., the rate at which a unit is included). In our study, we applied dropout to every transformer block in the pre-trained DeBERTa model [3]. We also explored only inclusion rates close to one, as recommended by a previous study [69].

Suppose $\mathbf{y}, \mathbf{y}' \in \mathbb{R}^d$ denote the output of the linear layer before and after applying dropout respectively, and $mask \in \mathbb{R}^d$ is a mask vector randomly sampled from a Bernoulli distribution with inclusion rate p. The forward pass during training is defined as follows:

$$mask \sim bernoulli(p) \tag{4.14}$$

$$\mathbf{y}' = mask \odot \mathbf{y} \tag{4.15}$$

By randomly dropping some units in training, dropout reduces the dependency of the hidden units between the layers and forces the model to extract diverse features. This approach can be viewed as bagging different sub-networks and averaging their outputs. During inference, dropout is not applied because we do not want stochasticity in the prediction. To account for the change in expectations of the output values, we scale $\mathbf{y}$ down by the inclusion rate:

$$\mathbf{y}' = p * \mathbf{y} \tag{4.16}$$

4.3.2 Learning Rate

Secondly, we investigated how changing the learning rate of the optimiser affects the model's performance. The learning rate is used during backpropagation. Specifically, when updating the weights, it determines the step size the optimiser takes along the negative gradient direction with respect to the loss [70]. By updating the weights over several iterations, the model is guided towards a local minimum of the loss function.

The choice of learning rate significantly affects the model's convergence to a set of weights that achieves this minimum. A small learning rate allows the model to learn a more optimal set of weights, but it will take longer to reach this solution because weights are updated more slowly, resulting in a longer training process. In contrast, a high learning rate allows the model to learn faster, but it can also lead to oscillations around the local minimum because the weights are updated more aggressively, resulting in a suboptimal set of weights. Based on the findings of previous studies [15, 16, 68], we evaluated learning rates in the range of 5×10^{-5} to 5×10^{-4}.

4.3.3 Number of Unfrozen Layers

Thirdly, we investigated the impact of freezing layers in the pre-trained DeBERTa model. Freezing layers is a common practice when fine-tuning a model on task-specific data to mitigate overfitting [71]. When a layer is frozen, its parameters are not updated during training. We chose to freeze the layers in the first few blocks of the model because they learn the low-level features of the text from pre-training. These features are generic and can be effectively applied to various NLP tasks, including sequence classification. Given that neural authorship attribution is a form of sequence classification, we wanted to preserve and use the learned low-level features.

Only the layers in the final x transformer blocks were unfrozen to allow the pre-trained model to adapt to our specific task, where x is the hyperparameter we modified. By doing this, these layers can learn and capture the task-specific features during training, improving the model's performance on our dataset. Previous studies have shown that the optimal performance during fine-tuning is typically achieved with minimal adjustments to the pre-trained parameters [71, 72]; hence, we kept the value of x smaller than 3.

4.3.4 Weight Decay

Finally, we investigated the impact of weight decay on the model's performance. Weight decay, also known as L2 regularisation, is a technique that reduces overfitting on the training data [73]. This approach works by adding a regularisation term to the loss function to penalise the magnitude of the model's weights. Given the predictions $\hat{\mathbf{Y}} = (\hat{\mathbf{y}}_1, ..., \hat{\mathbf{y}}_N) \in [0,1]^{N\times 9}$ and labels $\mathbf{Y} = (\mathbf{y}_1, ..., \mathbf{y}_N) \in \{0,1\}^{N\times 9}$ for a batch of N articles, the modified loss function of the entire batch is given in Equation 4.17, where $\mathbf{W}$ and λ are the model's weights and regularisation parameter respectively.

$$L_{batch}(\mathbf{Y}, \hat{\mathbf{Y}}, \mathbf{W}) = -\frac{1}{N}\sum_{i=1}^{N} L(\hat{\mathbf{y}}_i, \mathbf{y}_i) + \lambda ||\mathbf{W}||_2^2 \tag{4.17}$$

The regularisation term explicitly constrains the magnitude of the weights using L2 norms to prevent them from becoming too large, thus avoiding overly sensitive behaviour on unseen data. The regularisation parameter λ is the hyperparameter we adjust in our ablation study. It controls the relative importance between the loss and the regularisation penalty. A small λ indicates that we prioritise minimising the cross-entropy loss, while a high λ suggests a preference for smaller weights. We experimented with weight decay values between 0.01 to 0.1, as suggested by previous work [70, 73].

4.4 Author Writing Style Analysis

After conducting the ablation study, we analysed each author's writing style using the best-performing model. This analysis aims to identify the linguistic features that discriminate each author's writing style from the others. Our findings will improve the model's interpretability and deepen our understanding of how these authors comprehend and produce text in natural language. Consequently, this provides valuable insights into the artificial and human cognitive processes related to natural language generation.

4.4.1 Linguistic Feature Identification

Our model solves neural authorship attribution by being trained on a dataset of news articles generated by nine authors (i.e., humans and 8 LLMs). During this process, the model learns to identify the author that generated a given text accurately. It does so by extracting the linguistic features, either textual or stylometric, that discriminate each author's writing style.

Based on this idea, we hypothesise that our model could be used to identify the linguistic features that characterise each author's writing style. This identification is achieved by assessing each feature's contribution to our model's ability to predict a particular author. A simple metric to measure contribution is feature importance. The more important a feature is, the greater its impact on the model's performance in correctly attributing texts to that author. Subsequently, such a feature is likely a key discriminative characteristic of the author's writing style. It is important to note that this discrimination is relative to the writing styles of other authors in the dataset, not an absolute measure of the author's writing style in general.

In news article writing, no previous work has examined the writing styles of the authors in our dataset using both stylometric and textual features. Hence, our analysis and findings are novel.

4.4.2 Shapley Additive Explanations

Building on a similar work [17], we chose Shapley Additive Explanations (SHAP) [4] to compute the feature importance because it provides a unified framework for interpreting the results of machine learning models. Its model-agnostic nature allows it to be easily applied to many black-box models without requiring knowledge of their internal representation. This property makes SHAP ideal for interpreting our transformer-based classifier that inherently lacks interpretability. Additionally, we performed our analysis on the test set because it provides an unbiased evaluation of the model, ensuring that the feature importance values assigned by SHAP are not influenced by the data our model has already seen.

First, we use SHAP to examine the local interpretation of our model. Local interpretation aims to explain the model's prediction for each instance in our test set. Given an article-author pair, SHAP assigns a Shapley value to each feature of the article, reflecting its importance in influencing the model's prediction for that author. A positive value implies that a feature favourably impacts the model's prediction for the author, whereas a negative value suggests a negative effect. Following this, we assessed the global interpretation of our model to better understand its overall behaviour on the test set and improve the generalisability of the results. This entails averaging the Shapley values of each feature for an author across the test set to determine its overall importance to that author.

Chapter 5

Experiments

In this chapter, we first explain the motivation for the experiments, the experimental setup, and the baselines used. Next, we interpret and discuss the findings, including the results of the ablation study and the analysis of each author's writing style. The main research questions addressed in this work are:

1. Does the incorporation of stylometric features improve the performance of classifiers for neural authorship attribution?
2. Does our model perform better than the current state-of-the-art classifier for neural authorship attribution?
3. What linguistic features are useful for distinguishing between the writing styles of different LLMs?

5.1 Motivation

The experiments aim to address three objectives. The first objective is to investigate whether incorporating stylometric features improves the accuracy of our classifier, thereby answering our first research question. The second objective is to determine if our model outperforms the current state-of-the-art classifier for this task, which would answer the second research question. Lastly, the third objective is to identify the linguistic features that distinguish each author's writing style, allowing us to answer the third research question.

5.2 Baselines

To evaluate our model's performance, we compare its results with those from state-of-the-art bidirectional language models that were fine-tuned on our dataset. These models include DeBERTa [3], RoBERTa [15], BERT [16], Electra [74], and XLNet [75]. We address the first objective by comparing our model's performance to DeBERTa, as our model is an adaptation of DeBERTa that integrates stylometric features with textual features. To achieve the second objective, our model's performance is compared to RoBERTa and BERT, the current state-of-the-art classifiers for neural authorship attribution [49]. We included Electra and XLNet in our baselines to provide a comprehensive benchmark for evaluating our model against other language modelling architectures.

In news article writing, no literature analysis has examined the writing style of LLMs and humans using stylometric and textual features, so no direct baselines exist for comparison.

5.3 Experiment Description

Our experiment is divided into two parts. In the first part, we trained our model and the baselines on our dataset. In the second part, we applied SHAP to the best-performing model to compute the global importance of each linguistic feature for every author.

5.3.1 Model & Baseline Training

All training runs were performed using the AdamW optimiser [76] and a linear learning rate scheduler with no warmup phase. By gradually decreasing the learning rate with a scheduler, we achieved faster convergence to an optimal solution, reducing training time. We trained the models for 100 epochs using a batch size of 64. To prevent overfitting, we implemented early stopping with a patience of five epochs to monitor the validation loss. Finally, to ensure the results were reproducible, we set the seed to 2543673 before we started training.

During model training, we experimented with different learning rates, weight decay values, number of unfrozen layers in DeBERTa [3], and dropout inclusion rates. Table 5.1 shows the search space of each hyperparameter. The choice of these hyperparameters was justified in the ablation study. The best-performing model, as determined by the validation accuracy, was selected for analysis in the second part of the experiment.

Each baseline was fine-tuned using the same hyperparameter configuration as the best-performing model. We did this to evaluate whether our model performed better under the same setting. Furthermore, we used the base version of all models to ensure they had a similar number of parameters. This approach allowed us to attribute any differences in performance to the model's architecture rather than its size.

When comparing the results, we used the performance of each model at the epoch where it had attained the lowest validation loss, rather than at the end of the training. This ensured that we evaluated each model when it best generalised to the validation data before it started overfitting. Consequently, the weights of each model from these specific epochs were saved and used to assess their performance on the test set.

Hyperparameter	Search Space
Dropout Inclusion Rate	0.8,0.9
Number of Unfrozen Layers	1,2,3
Weight Decay	0.01,0.05,0.1
Learning Rate	5×10^{-5}, 1×10^{-4}, 5×10^{-4}

Table 5.1: Search space for each hyperparameter in our model

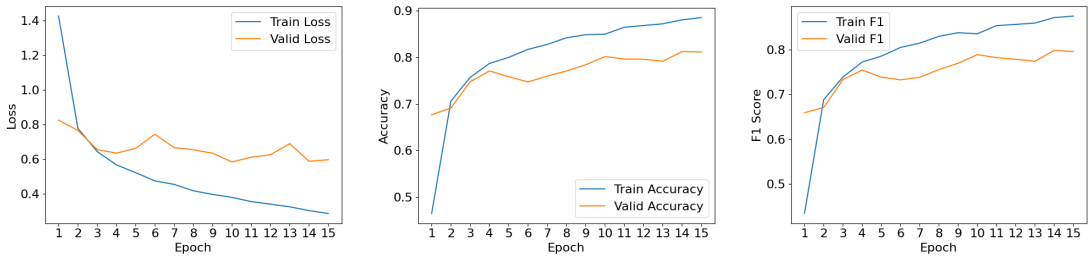
5.3.2 SHAP Feature Importance Estimation

Using our best-performing model, we applied SHAP's DeepExplainer [77] on its classification head to estimate the global importance of each linguistic feature for every author. A seed of 2543673 was used to randomly select 1000 samples from the training data to serve as our background dataset for feature integration.

Given a news article from the test set, we first fed its text through DeBERTa [3] to generate the sentence embeddings. These embeddings were then concatenated with the stylometric features of the articles to create the complete set of linguistic features. The DeepExplainer then assigns a Shapley value to each feature from this set, indicating its contribution to the classification head's prediction of the article's author. We repeat this process for every article in our test set and then compute the global importance of each linguistic feature for every author, as detailed in the Methodology chapter. Since the individual dimensions of the sentence embeddings are not directly interpretable, we summed the global importance of all embedding dimensions into a single value that reflects the overall significance of the article's semantics.

5.4 Ablation Study Results

The ablation study revealed that the hyperparameter configuration that achieved the highest validation accuracy and F1-score was a learning rate of 5×10^{-5}, a weight decay of 0.05, a dropout inclusion rate of 0.9, and 1 unfrozen layer. Figure 5.1 illustrates our model's training and validation performance with this configuration.



(a) Cross-Entropy Loss (b) Accuracy (c) F1-Score

Figure 5.1: Our model's performance on the best hyperparameter configuration

Figure 5.1a shows that our model is progressively learning the authors' writing style from the training data, as evidenced by the decreasing training loss. Additionally, the model's ability to generalise to unseen articles is improving because the validation loss gradually decreased. Figure 5.1b and 5.1c further support this claim because both the training and validation accuracy and F1-score steadily increased with the number of training epochs. These results suggest that, as training progresses, the model makes increasingly more correct predictions for all authors, both on the training and validation data, while reducing each author's false positive and negative rates.

Our model attained the lowest validation loss of 0.802 at epoch 10. It also achieved a validation accuracy of 80.2% and an F1-score of 0.789 at this epoch. Following this point, the validation loss increased, indicating the onset of overfitting. Consequently, we ended training at epoch 15 with early stopping.

Table 5.2 shows that our model outperformed all baselines on both the test accuracy and F1-score. This result has two significant implications. Firstly, incorporating stylometric features into our classifier improved its performance. This improvement is evident from our model's 1.1% increase in test accuracy compared to DeBERTa [3]. Additionally, our model achieved state-of-the-art performance in neural authorship attribution on our dataset, surpassing the previously leading RoBERTa [15] and BERT [16] models by at least 8.3% in test accuracy. This suggests that DeBERTa's architecture is better suited than other language modelling architectures for capturing

and distinguishing the nuanced linguistic features of each author.

Model	Accuracy	F1-Score
BERT [16]	74.8%	0.735
Electra [74]	74.5%	0.727
XLNET [75]	67.6%	0.660
RoBERTa [15]	74.4%	0.730
DeBERTa [3]	82.0%	0.815
DeBERTa$_{stylo}$ (Our model)	**83.1%**	**0.825**

Table 5.2: Our model and the baseline performance on the test set

5.5 Writing Style Analysis Results

This section examines the writing styles of the authors in our dataset. For each author, we plotted and analysed a bar chart displaying the top five features with the highest global importance for that author. These features are the most significant characteristics that differentiate their writing style from the others.

5.5.1 Humans

According to Figure 5.2a, the human writing style is primarily characterised by its semantics because this feature has significantly higher global importance compared to others. Subsequently, stylometric features are less effective for identifying human-written articles. This result is logical because human writing often expresses emotions, thoughts, or viewpoints that typically reflect the writer's knowledge, cultural background, and temperament; hence, the writing style of one individual would vary significantly from another based on these factors. Since our dataset includes human-written articles created by multiple individuals, the stylometric features for the human authors will be inconsistent. This inconsistency renders them ineffective in training our model to learn the human writing style.

5.5.2 GPT-J & OPT 30B

Similar to human writing, Figure 5.2a shows that the writing styles of OPT 30B [56] and GPT-J [30] are characterised solely by their semantics. This observation

suggests that the articles generated by these models lack consistent stylometric signals. Such inconsistency highlights the powerful generative capabilities of these LLMs that allow them to create high-quality text in many styles, each with different stylometric characteristics. Consequently, these models do not adhere to a single, identifiable writing style, making it difficult to detect the texts they generate using only stylometric features.

Although the global importance of the semantics is comparable for both humans and GPT-J, it is significantly lower, by at least half, for OPT 30B. This suggests that semantics play a more crucial role in identifying the articles generated by humans and GPT-J compared to those produced by OPT 30B. As a result, the writing style of GPT-J is likely more sophisticated and closer to human writing than OPT 30B.

5.5.3 GPT-NeoX

From Figure 5.2b, the writing style of GPT-NeoX [31] is mainly characterised by several stylometric features: the Honore's measure, MATTR, Hapax legomena count, and the standard deviation in the number of nouns and adpositions per sentence. Each feature holds comparable predictive power for this author because they share similar global importance. Since the first three features assess vocabulary richness, their high importance suggests that articles produced by this model display a relatively distinct pattern in vocabulary usage. Furthermore, the importance of the last two features indicates that the model has a predictable pattern in how adpositions and nouns are used throughout the sentences. This suggests that the articles produced by GPT-NeoX tend to have a more consistent syntactic structure.

5.5.4 LLama 65B

Figure 5.2b depicts the top five features that differentiate LLama 65B's [32] writing style from others. Since these features have approximately equal global importance, they hold comparable predictive power for this author. The high importance of the average number of determiners, function words, and adpositions per sentence suggests a consistent pattern in their usage across the articles generated by LLama. This highlights the model's tendency to produce text with a distinctive syntactic structure, indicating a more methodical writing style. Secondly, the high importance of the standard deviation in the number of particles and auxiliary words per sentence implies that these articles display predictable patterns in how these words are used throughout the sentences.

5.5.5 GPT-3.5 Turbo

According to Figure 5.2b, the writing style of GPT-3.5 Turbo [28] is primarily characterised by the standard deviation in the number of adpositions per sentence, as its global importance almost doubles that of the second most important feature. This result suggests that the articles generated by this model follow a consistent pattern in the usage of adpositions across the sentences. From this, we can deduce that the model maintains a uniform approach to sentence construction. The model's writing style could also be differentiated using other stylometric features such as the average number of characters per word. The high importance of this feature reveals a discernible pattern in the length of the words generated by this model. This suggests that the model follows a systematic approach to word formation and vocabulary usage.

Compared to GPT-NeoX and LLama 65B, GPT-3.5 Turbo has a more defined writing style, as evidenced by the significantly higher global importance of its most prominent feature relative to others. This implies that a single feature is sufficient to accurately characterise the model's writing style. In contrast, the top five features for the other two models have relatively equal global importance, suggesting that our model relies on a broader set of features to identify the articles generated by these models. This broader reliance indicates a more nuanced writing style that is harder to identify.

5.5.6 Flan-T5 XXL

The writing style of Flan-T5 XXL [29] is mainly characterised by the features shown in Figure 5.2c. The top two most significant features suggest that the model maintains a relatively predictable structure, particularly in its use of auxiliary verbs and nouns across sentences. Consequently, the model's writing style appears to have a uniform syntactic pattern. Additionally, the high importance of the average number of pronouns, adpositions and proper nouns per sentence suggests a predictable pattern in their usage across the articles generated by this model.

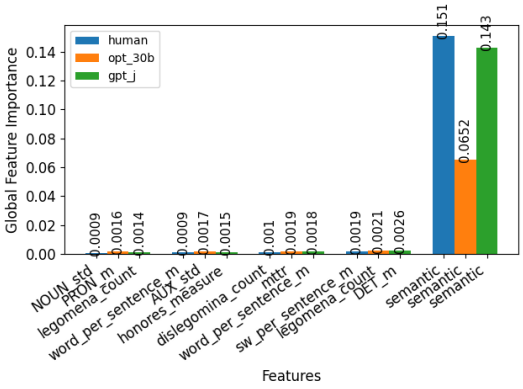
5.5.7 BigScience T0 11B

According to Figure 5.2c, the writing style of T0 11B [57] is characterised by both its semantics and stylistic features, as they share similar global importance. The high importance of its semantics indicates a strong underlying pattern in the thematic focus and overall meaning of the generated text. Additionally, the high emphasis placed on

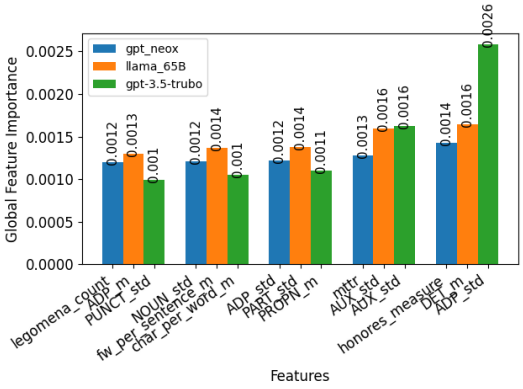
the average number of function words per sentence and character per word implies that the model has a systematic and uniform approach to word formation, vocabulary usage and sentence construction.

5.5.8 Text Davinci-003

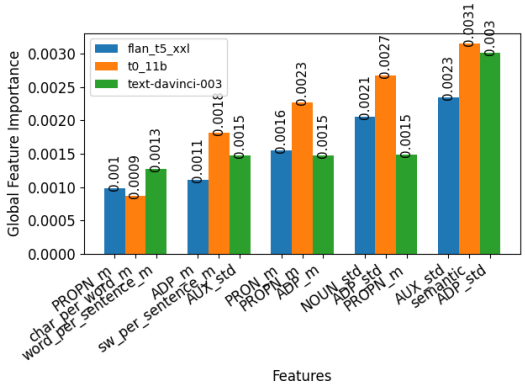
From Figure 5.2c, the writing style of Davinci [28] is predominantly characterised by the standard deviation in the number of adpositions per sentence, as its global importance doubles that of the second most important feature. This suggests a discriminative pattern in the usage of adpositions across the sentences. Based on this observation, the model appears to exhibit a systematic approach to sentence construction. There is also a consistent pattern in the number of words per sentence, as evidenced by the high global importance of this feature. This implies that the model maintains a structured control of the sentence length during text generation.



(a) Humans, OPT 30B and GPT-J



(b) GPT-NeoX, LLama 65B and GPT-3.5 Turbo



(c) Flan-T5 XXL, T0 11B and Text Davinci-003

Figure 5.2: Top 5 most discriminative features for each author's writing style

Chapter 6

Conclusions

In this paper, we make several key contributions. First, we present a novel dataset for neural authorship attribution derived from the MAGE testbed [14]. This dataset focuses on news article writing and contains 15k news articles that were either written by humans or generated by one of eight state-of-the-art LLMs. Each article is represented by its content and 57 stylometric features. Secondly, we proposed a novel model DeBERTa$_{stylo}$ that incorporates stylometric features with the DeBERTa architecture [3] for neural authorship attribution. An ablation study was conducted using our dataset to select the optimal hyperparameters for the model. Finally, we analysed our best-performing model, as determined by the validation accuracy, using SHAP [4] to identify the linguistic features that discriminate each author's writing style.

Our ablation study results demonstrated that our model outperformed DeBERTa, achieving a 1.1% increase in test accuracy. This suggests that incorporating stylometric features improves the classifier's performance by providing additional information that helps the model better distinguish between the writing styles. Furthermore, our model outperformed the current state-of-the-art classifiers such as RoBERTa [15] and BERT [16], achieving at least an 8.3% increase in test accuracy. This finding suggests that DeBERTa's architecture is inherently more effective at capturing the subtle nuance of the author's writing styles than other masked language modelling architectures. Lastly, our analysis of the writing styles revealed that most LLMs use a systematic approach to sentence construction. The semantic and syntactic features of the text were the most effective for distinguishing between the different writing styles.

Although our results indicate that incorporating stylometric features improved our model's performance, more work is needed to determine whether this improvement has practical significance. This significance can be assessed by conducting statistical

tests to compare our model's performance with the baseline. Furthermore, our model's success might be partially attributed to our specific dataset rather than the inherent advantages of combining stylometric features with the learned textual features from DeBERTa. To improve the reliability and validity of our findings, further investigations should evaluate our model's performance on datasets from other domains.

Another significant issue with our approach was that we evaluated our model solely on data from a single domain: news article writing. This was done to eliminate domain-specific conventions that can alter the writing styles, thereby improving the reliability and accuracy of our writing style analysis. However, for a classifier to effectively address neural authorship attribution, it must generalise well to out-of-distribution text generated by previously unencountered LLMs. Studies have demonstrated that classifiers often exhibit substantial performance degradation when evaluated on out-of-distribution data [14, 78]. Therefore, further work should focus on exploring solutions to address this out-of-distribution challenge for our model.

Bibliography

[1] H. Ramnial, S. Panchoo, and S. Pudaruth, "Authorship attribution using stylometry and machine learning techniques," in *Intelligent Systems Technologies and Applications* (S. Berretti, S. M. Thampi, and P. R. Srivastava, eds.), (Cham), pp. 113–125, Springer International Publishing, 2016.

[2] I. N. Bozkurt, O. Baghoglu, and E. Uyar, "Authorship attribution," in *2007 22nd international symposium on computer and information sciences*, pp. 1–5, 2007.

[3] P. He, X. Liu, J. Gao, and W. Chen, "Deberta: Decoding-enhanced bert with disentangled attention," 2021.

[4] S. Lundberg and S.-I. Lee, "A unified approach to interpreting model predictions," 2017.

[5] J. Goldstein, J. Chao, S. Grossman, A. Stamos, and M. Tomz, "Can ai write persuasive propaganda?," 04 2023.

[6] M. Subbiah, A. Bhattacharjee, Y. Hua, T. Kumarage, H. Liu, and K. McKeown, "Towards detecting harmful agendas in news articles," 2023.

[7] A. Uchendu, T. Le, and D. Lee, "Attribution and obfuscation of neural text authorship: A data mining perspective," 2023.

[8] T. Kumarage, J. Garland, A. Bhattacharjee, K. Trapeznikov, S. Ruston, and H. Liu, "Stylometric detection of ai-generated text in twitter timelines," 2023.

[9] A. Uchendu, T. Le, K. Shu, and D. Lee, "Authorship attribution for neural text generation," in *Proceedings of the 2020 Conference on Empirical Methods in Natural Language Processing (EMNLP)* (B. Webber, T. Cohn, Y. He, and Y. Liu, eds.), (Online), pp. 8384–8395, Association for Computational Linguistics, Nov. 2020.

[10] F. Jafariakinabad, S. Tarnpradab, and K. A. Hua, "Syntactic recurrent neural network for authorship attribution," 2019.

[11] P. Shrestha, S. Sierra, F. González, M. Montes, P. Rosso, and T. Solorio, "Convolutional neural networks for authorship attribution of short texts," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (M. Lapata, P. Blunsom, and A. Koller, eds.), (Valencia, Spain), pp. 669–674, Association for Computational Linguistics, Apr. 2017.

[12] T. Kumarage and H. Liu, "Neural authorship attribution: Stylometric analysis on large language models," 2023.

[13] L. Fröhling and A. Zubiaga, "Feature-based detection of automated language models: tackling gpt-2, gpt-3 and grover," *PeerJ Computer Science*, vol. 7, 2021.

[14] Y. Li, Q. Li, L. Cui, W. Bi, Z. Wang, L. Wang, L. Yang, S. Shi, and Y. Zhang, "Mage: Machine-generated text detection in the wild," 2024.

[15] Y. Liu, M. Ott, N. Goyal, J. Du, M. Joshi, D. Chen, O. Levy, M. Lewis, L. Zettlemoyer, and V. Stoyanov, "Roberta: A robustly optimized bert pretraining approach," 2019.

[16] J. Devlin, M.-W. Chang, K. Lee, and K. Toutanova, "Bert: Pre-training of deep bidirectional transformers for language understanding," 2019.

[17] S. Mitrović, D. Andreoletti, and O. Ayoub, "Chatgpt or human? detect and explain. explaining decisions of machine learning model for detecting short chatgpt-generated text," 2023.

[18] J. Li, T. Tang, W. X. Zhao, and J.-R. Wen, "Pretrained language models for text generation: A survey," 2021.

[19] H. Zhang, H. Song, S. Li, M. Zhou, and D. Song, "A survey of controllable text generation using transformer-based pre-trained language models," 2023.

[20] E. Crothers, N. Japkowicz, and H. Viktor, "Machine generated text: A comprehensive survey of threat models and detection methods," 2023.

[21] L. E. Baum and T. Petrie, "Statistical inference for probabilistic functions of finite state markov chains," *Annals of Mathematical Statistics*, vol. 37, pp. 1554–1563, 1966.

[22] S. Janarthanam and O. Lemon, "Learning lexical alignment policies for generating referring expressions for spoken dialogue systems," in *Proceedings of the 12th European Workshop on Natural Language Generation (ENLG 2009)* (E. Krahmer and M. Theune, eds.), (Athens, Greece), pp. 74–81, Association for Computational Linguistics, Mar. 2009.

[23] D. Y. Pawade, A. Sakhapara, M. Jain, N. Jain, and K. Gada, "Story scrambler - automatic text generation using word level rnn-lstm," *International Journal of Information Technology and Computer Science*, 2018.

[24] V.-K. Tran and L.-M. Nguyen, "Semantic refinement gru-based neural language generation for spoken dialogue systems," in *Computational Linguistics: 15th International Conference of the Pacific Association for Computational Linguistics, PACLING 2017, Yangon, Myanmar, August 16–18, 2017, Revised Selected Papers 15*, pp. 63–75, Springer, 2018.

[25] S. Hochreiter, "The vanishing gradient problem during learning recurrent neural nets and problem solutions," *International Journal of Uncertainty, Fuzziness and Knowledge-Based Systems*, vol. 6, no. 02, pp. 107–116, 1998.

[26] A. Vaswani, N. Shazeer, N. Parmar, J. Uszkoreit, L. Jones, A. N. Gomez, L. Kaiser, and I. Polosukhin, "Attention is all you need," 2023.

[27] A. Radford, J. Wu, R. Child, D. Luan, D. Amodei, and I. Sutskever, "Language models are unsupervised multitask learners," 2019.

[28] T. B. Brown, B. Mann, N. Ryder, M. Subbiah, J. Kaplan, P. Dhariwal, A. Neelakantan, P. Shyam, G. Sastry, A. Askell, S. Agarwal, A. Herbert-Voss, G. Krueger, T. Henighan, R. Child, A. Ramesh, D. M. Ziegler, J. Wu, C. Winter, C. Hesse, M. Chen, E. Sigler, M. Litwin, S. Gray, B. Chess, J. Clark, C. Berner, S. McCandlish, A. Radford, I. Sutskever, and D. Amodei, "Language models are few-shot learners," 2020.

[29] C. Raffel, N. Shazeer, A. Roberts, K. Lee, S. Narang, M. Matena, Y. Zhou, W. Li, and P. J. Liu, "Exploring the limits of transfer learning with a unified text-to-text transformer," 2023.

[30] B. Wang and A. Komatsuzaki, "Gpt-j-6b: A 6 billion parameter autoregressive language model." https://github.com/kingoflolz/mesh-transformer-jax, May 2021.

[31] S. Black, S. Biderman, E. Hallahan, Q. Anthony, L. Gao, L. Golding, H. He, C. Leahy, K. McDonell, J. Phang, M. Pieler, U. S. Prashanth, S. Purohit, L. Reynolds, J. Tow, B. Wang, and S. Weinbach, "Gpt-neox-20b: An open-source autoregressive language model," 2022.

[32] H. Touvron, T. Lavril, G. Izacard, X. Martinet, M.-A. Lachaux, T. Lacroix, B. Rozière, N. Goyal, E. Hambro, F. Azhar, A. Rodriguez, A. Joulin, E. Grave, and G. Lample, "Llama: Open and efficient foundation language models," 2023.

[33] D. Jurafsky and J. H. Martin, *Speech and Language Processing: An Introduction to Natural Language Processing, Computational Linguistics, and Speech Recognition*. USA: Prentice Hall PTR, 1st ed., 2000.

[34] J. Wu, S. Yang, R. Zhan, Y. Yuan, D. F. Wong, and L. S. Chao, "A survey on llm-generated text detection: Necessity, methods, and future directions," 2024.

[35] J.-J. Weber *et al.*, *The Stylistics Reader From Roman Jakobson to the Present*. Arnold, London, Unknown/unspecified, 1996.

[36] A. Sharma, A. Nandan, and R. Ralhan, "An investigation of supervised learning methods for authorship attribution in short hinglish texts using char word n-grams," 2018.

[37] Y. Sari, A. Vlachos, and M. Stevenson, "Continuous n-gram representations for authorship attribution," in *Proceedings of the 15th Conference of the European Chapter of the Association for Computational Linguistics: Volume 2, Short Papers* (M. Lapata, P. Blunsom, and A. Koller, eds.), (Valencia, Spain), pp. 267–273, Association for Computational Linguistics, Apr. 2017.

[38] Y. Seroussi, I. Zukerman, and F. Bohnert, "Authorship attribution with topic models," *Computational Linguistics*, vol. 40, pp. 269–310, June 2014.

[39] A. Uchendu, J. Cao, Q. Wang, B. Luo, and D. Lee, "Characterizing man-made vs. machine-made chatbot dialogs," 10 2019.

[40] O. Halvani, L. Graner, R. Regev, and P. Marquardt, "An improved topic masking technique for authorship analysis," 04 2020.

[41] E. Ferracane, S. Wang, and R. Mooney, "Leveraging discourse information effectively for authorship attribution," in *Proceedings of the Eighth International Joint Conference on Natural Language Processing (Volume 1: Long Papers)* (G. Kondrak and T. Watanabe, eds.), (Taipei, Taiwan), pp. 584–593, Asian Federation of Natural Language Processing, Nov. 2017.

[42] R. Zheng, J. Li, H. Chen, and Z. Huang, "A framework for authorship identification of online messages: Writing-style features and classification techniques," *Journal of the American Society for Information Science and Technology*, vol. 57, no. 3, pp. 378–393, 2006.

[43] J. Shao, A. Uchendu, and D. Lee, "A reverse turing test for detecting machine-made texts," in *Proceedings of the 10th ACM Conference on Web Science*, WebSci '19, (New York, NY, USA), p. 275–279, Association for Computing Machinery, 2019.

[44] F. Howedi, "Text classification for authorship attribution using naive bayes classifier with limited training data," 12 2014.

[45] T. Solorio, S. Pillay, S. Raghavan, and M. Montes y Gómez, "Modality specific meta features for authorship attribution in web forum posts," in *Proceedings of 5th International Joint Conference on Natural Language Processing* (H. Wang and D. Yarowsky, eds.), (Chiang Mai, Thailand), pp. 156–164, Asian Federation of Natural Language Processing, Nov. 2011.

[46] R. Hou and C.-R. Huang, "Stylometric studies based on tone and word length motifs," in *Pacific Asia Conference on Language, Information and Computation*, 2017.

[47] H. Alshaher and J. Xu, "A new term weight scheme and ensemble technique for authorship identification," in *Proceedings of the 2020 4th International Conference on Compute and Data Analysis*, ICCDA '20, (New York, NY, USA), p. 123–130, Association for Computing Machinery, 2020.

[48] B. Alsulami, E. Dauber, R. Harang, S. Mancoridis, and R. Greenstadt, "Source code authorship attribution using long short-term memory based networks," pp. 65–82, 08 2017.

[49] A. Uchendu, Z. Ma, T. Le, R. Zhang, and D. Lee, "Turingbench: A benchmark environment for turing test in the age of neural text generation," 2021.

[50] Peter Bloem, "Transformer From Scratch." `https://peterbloem.nl/blog/transformers`. Accessed: 2024-07-01.

[51] Y. Wu, M. Schuster, Z. Chen, Q. V. Le, M. Norouzi, W. Macherey, M. Krikun, Y. Cao, Q. Gao, K. Macherey, *et al.*, "Google's neural machine translation system: Bridging the gap between human and machine translation," *arXiv preprint arXiv:1609.08144*, 2016.

[52] R. Sennrich, B. Haddow, and A. Birch, "Neural machine translation of rare words with subword units," 2016.

[53] A. Uchendu, Z. Ma, T. Le, R. Zhang, and D. Lee, "The Turing Test Benchmark Environment," 2021. `hhttps://turingbench.ist.psu.edu/` [Accessed: 01/08/2024].

[54] S. Narayan, S. B. Cohen, and M. Lapata, "Don't give me the details, just the summary! Topic-aware convolutional neural networks for extreme summarization," in *Proceedings of the 2018 Conference on Empirical Methods in Natural Language Processing*, (Brussels, Belgium), 2018.

[55] Jules Belveze, "tldr_news." `https://huggingface.co/datasets/JulesBelveze/tldr_news`.

[56] S. Zhang, S. Roller, N. Goyal, M. Artetxe, M. Chen, S. Chen, C. Dewan, M. Diab, X. Li, X. V. Lin, T. Mihaylov, M. Ott, S. Shleifer, K. Shuster, D. Simig, P. S. Koura, A. Sridhar, T. Wang, and L. Zettlemoyer, "Opt: Open pre-trained transformer language models," 2022.

[57] V. Sanh, A. Webson, C. Raffel, S. Bach, L. Sutawika, Z. Alyafeai, A. Chaffin, A. Stiegler, A. Raja, M. Dey, M. S. Bari, C. Xu, U. Thakker, S. S. Sharma, E. Szczechla, T. Kim, G. Chhablani, N. Nayak, D. Datta, J. Chang, M. T.-J. Jiang, H. Wang, M. Manica, S. Shen, Z. X. Yong, H. Pandey, R. Bawden, T. Wang, T. Neeraj, J. Rozen, A. Sharma, A. Santilli, T. Fevry, J. A. Fries, R. Teehan, T. L. Scao, S. Biderman, L. Gao, T. Wolf, and A. M. Rush, "Multitask prompted training enables zero-shot task generalization," in *International Conference on Learning Representations*, 2022.

[58] R. Flesch, "A new readability yardstick.," *Journal of applied psychology*, vol. 32, no. 3, p. 221, 1948.

[59] R. Gunning, "The technique of clear writing," *(No Title)*, 1952.

[60] S. Bird, E. Klein, and E. Loper, *Natural language processing with Python: analyzing text with the natural language toolkit*. " O'Reilly Media, Inc.", 2009.

[61] Wang Haining, "functionwords." `https://github.com/Wang-Haining/functionwords`.

[62] M. Honnibal and I. Montani, "spaCy 2: Natural language understanding with Bloom embeddings, convolutional neural networks and incremental parsing." To appear, 2017.

[63] A. Honoré *et al.*, "Some simple measures of richness of vocabulary," *Association for literary and linguistic computing bulletin*, vol. 7, no. 2, pp. 172–177, 1979.

[64] M. A. Covington and J. D. McFall, "Cutting the gordian knot: The moving-average type–token ratio (mattr)," *Journal of Quantitative Linguistics*, vol. 17, no. 2, pp. 94–100, 2010.

[65] B. Richards, "Type/token ratios: what do they really tell us?," *Journal of Child Language*, vol. 14, no. 2, p. 201–209, 1987.

[66] S. Petrov, D. Das, and R. McDonald, "A universal part-of-speech tagset," *arXiv preprint arXiv:1104.2086*, 2011.

[67] J. P. Kincaid, R. P. Fishburne Jr, R. L. Rogers, and B. S. Chissom, "Derivation of new readability formulas (automated readability index, fog count and flesch reading ease formula) for navy enlisted personnel," 1975.

[68] P. He, J. Gao, and W. Chen, "Debertav3: Improving deberta using electra-style pre-training with gradient-disentangled embedding sharing," 2023.

[69] N. Srivastava, G. Hinton, A. Krizhevsky, I. Sutskever, and R. Salakhutdinov, "Dropout: A simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 56, pp. 1929–1958, 2014.

[70] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016. `http://www.deeplearningbook.org`.

[71] J. Howard and S. Ruder, "Universal language model fine-tuning for text classification," 2018.

[72] J. Yosinski, J. Clune, Y. Bengio, and H. Lipson, "How transferable are features in deep neural networks?," 2014.

[73] A. Y. Ng, "Feature selection, l1 vs. l2 regularization, and rotational invariance," in *Proceedings of the Twenty-First International Conference on Machine Learning*, ICML '04, (New York, NY, USA), p. 78, Association for Computing Machinery, 2004.

[74] K. Clark, M.-T. Luong, Q. V. Le, and C. D. Manning, "Electra: Pre-training text encoders as discriminators rather than generators," 2020.

[75] Z. Yang, Z. Dai, Y. Yang, J. Carbonell, R. Salakhutdinov, and Q. V. Le, "Xlnet: Generalized autoregressive pretraining for language understanding," 2020.

[76] I. Loshchilov and F. Hutter, "Decoupled weight decay regularization," 2019.

[77] A. Shrikumar, P. Greenside, and A. Kundaje, "Learning important features through propagating activation differences," 2019.

[78] E. Mitchell, Y. Lee, A. Khazatsky, C. D. Manning, and C. Finn, "Detectgpt: Zero-shot machine-generated text detection using probability curvature," 2023.