information

Review

Authorship Attribution Methods, Challenges, and Future Research Directions: A Comprehensive Survey

Xie He [1], Arash Habibi Lashkari [1,*], Nikhill Vombatkere [1] and Dilli Prasad Sharma [2]

[1] Behaviour-Centric Cybersecurity Center (BCCC), School of Information Technology, York University, 4700 Keele Street, Toronto, ON M3J 1P3, Canada; hazel636@my.yorku.ca (X.H.); nvombatkere@gmail.com (N.V.)
[2] The Edward S. Rogers Sr. Department of Electrical & Computer Engineering, University of Toronto, 10 King's College Road, Toronto, ON M5S 3G4, Canada; dilli.sharma@utoronto.ca
* Correspondence: ahabibil@yorku.ca

Abstract: Over the past few decades, researchers have put their effort and paid significant attention to the authorship attribution field, as it plays an important role in software forensics analysis, plagiarism detection, security attack detection, and protection of trade secrets, patent claims, copyright infringement, or cases of software theft. It helps new researchers understand the state-of-the-art works on authorship attribution methods, identify and examine the emerging methods for authorship attribution, and discuss their key concepts, associated challenges, and potential future work that could help newcomers in this field. This paper comprehensively surveys authorship attribution methods and their key classifications, used feature types, available datasets, model evaluation criteria and metrics, and challenges and limitations. In addition, we discuss the potential future research directions of the authorship attribution field based on the insights and lessons learned from this survey work.

Keywords: authorship attribution; authorship verification; author profiling; source code analysis

Citation: He, X.; Lashkari, A.H.; Vombatkere, N.; Sharma, D.P. Authorship Attribution Methods, Challenges, and Future Research Directions: A Comprehensive Survey. *Information* **2024**, *15*, 131. https://doi.org/10.3390/info15030131

Academic Editor: Leandros Maglaras

Received: 2 February 2024
Revised: 19 February 2024
Accepted: 23 February 2024
Published: 28 February 2024

1. Introduction

Authorship attribution identifies the author of an unknown document, text, source code, executable binary, disputed provenance, or dubious or unknown origin [1]. Historically, research in this field dates back to the early 19th century [2]. The subject area of stylometry or authorship recognition/detection has several related fields of research, such as: (i) Authorship Attribution (AA) or identification, i.e., identifying the author(s) of a set of different texts; (ii) Authorship Characterization (AC), i.e., the detection of sociolinguistic attributes like gender, age, occupation, and educational level of the author; (iii) Authorship Verification (AV), i.e., checking whether a text or set of similar texts is written or not written by an author; (iv) Authorship Discrimination (AD), i.e., checking whether two different texts are written by the same author; (v) Plagiarism Detection (PD), i.e., searching for the sentence(s) or paragraph(s) that are reproduced from the text(s) of one or more than one author; (vi) Text Indexing and Segmentation (TIS), i.e., when several texts are issued from different authors, which are concatenated in a form of a global book or forum text; and (vii) Authorship De-Identification (ADI), i.e., identifying features that can adequately capture an author's writing style [3]. Also, one of the major threats to modern society is the propagation of misinformation (e.g., fake news, science denialism, hate speech) via social media. Authorship attribution techniques can help fight against disinformation by recognizing false identities. Several authorship attribution methods are applied to social media forensics analysis [4,5].

Applications of authorship attribution play a significant role in software forensics analysis [6], plagiarism detection [7,8], security attacks detection, and protection of trade secrets, patent claims, copyright infringement, or cases of software theft. Code authorship

attribution methods prevent and mitigate various security attacks, such as malware and adversarial attacks. Authors create many malware variants [9] and use them to attack target systems. The automatic coding style transformation method can help to reduce advanced adversarial examples and coding style manipulation attacks [10]. Learning an author's unique coding style patterns increases diversity, making it hard for attackers to manipulate or imitate. Several emerging methods for the authorship attribution of text/documents or source code have been proposed. This paper presents a comprehensive survey and critique of the state-of-the-art authorship attribution methods.

Over the past few decades, researchers have put their effort and paid significant attention to this emerging field that helps new researchers understand the state-of-the-art works on authorship attribution methods. We compared our survey work against four other existing related survey papers (e.g., [4,9,11,12]) in terms of key contributions based on some criteria such as model taxonomy, feature set taxonomy, datasets, evaluation metrics, synthesizing with shortcomings, challenges, and future research directions. We present the comparison of critical contributions of our work with other existing survey papers in Table 1.

Table 1. Comparison of key contributions of our survey paper compared to other existing survey papers.

Criteria	Our Work	Zheng and Jin [11]	Kalgutkar et al. [9]	Rocha et al. [4]	Stamatatos [12]
Model taxonomy	✓	✓(discussion only)	✓	✓(discussion only)	✓
Features taxonomy	✓	✓(discussion only)	✓	✓(social media only)	✓
Datasets	✓	✗	✓(partially)	✗	✗
Evaluation metrics	✓	✗	✗	✓(partially)	✓(partially)
Synthesizing with shortcomings	✓	✗	✗	✗	✗
Authorship verification	✓	✗	✗	✓(partially)	✗
Challenges	✓	✓	✓	✗	✗
Future research directions	✓	✗	✗	✓(social media only)	✗

1.1. Key Contributions

In this survey, we examined the existing authorship attribution methods, classified them based on their model characteristics, used feature sets, and discussed their performance, available datasets, challenges, and future work that would help a newcomer. This work has made the following key contributions:

- We extensively surveyed the state-of-the-art authorship attribution methods and classified them into five categories based on their model characteristics: stylistic, statistical, language, machine learning, and deep learning. We synthesized the characteristics of the model types and also listed their shortcomings.
- We investigated what feature sets are used in the state-of-the-art authorship attribution models. In particular, we surveyed the state-of-the-art and feature extraction methods and feature types they used.
- We developed an authorship attribution features taxonomy for text-based and code-based feature sets. In addition, we synthesized the critical properties of each feature type and discussed their shortcomings.
- We surveyed the available datasets and summarized their applicability, size, and shortcomings.
- We also surveyed various evaluation methods used to validate the performance of authorship attribution methods. We discussed the evaluation metrics used to assess the performance of authorship attribution methods.

- We discussed the key distinctive tasks of the source code or text authorship attribution models compared to authorship verification models. We briefly discussed the state-of-the-art authorship attribution methods and their shortcomings.
- Lastly, we discussed the challenges and limitations obtained from this extensive survey and suggested future work directions.

1.2. Paper Structure

This paper is organized as follows:

- Section 2 discusses the key model types and their classifications by synthesizing their characteristics with shortcomings.
- Section 3 presents the existing feature types and their taxonomy.
- Section 4 discusses available datasets primarily used in the source code and/or text authorship attribution research. Additionally, we discuss existing datasets and their shortcomings.
- Section 5 surveys metrics used to evaluate the effectiveness of the authorship attribution approaches in terms of their accuracy, precision, recall, and computational complexity.
- Section 6 provides a short review of the existing authorship verification methods. This section helps readers know how the authorship attribution task differs from the authorship verification task. This section presents the state-of-the-art authorship verification methods with their shortcomings.
- Section 7 discusses the challenges and limitations of the existing authorship attribution methods covered in this work.
- Section 8 discusses potential future works and suggests future research directions in text, documents, and/or source code authorship attribution.
- Section 9 concludes this survey work.

2. Review on Models and Their Classification

This section comprehensively reviews state-of-the-art works on authorship attribution models for text and source code. Based on their characteristics, we classify the models into five different categories. The authorial style was studied in depth by Juola et al. [13], who proposed that attribution analysis can either be supervised (documents categorized before analysis) or unsupervised (looks for superficial patterns in the data). Instead of this, we propose a different breakdown of approaches. Figure 1 shows our taxonomy of available models, and the rest of this section will explain the five proposed categories.

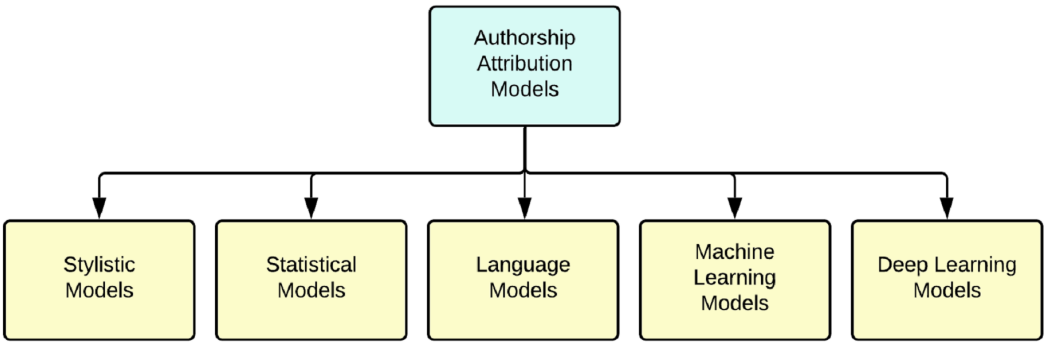

Figure 1. Model taxonomy.

2.1. Stylistic Models

Stylistic models consider each author's unique style—a sort of authorial fingerprint. Stylometric analysis of data, a prevalent approach to attributing authors to documents of unknown authorship, comprises analysis tasks, writing-style features used, and the techniques incorporated to analyze these features [14,15].

Mechti and Almansour [2] portray the evolutionary timeline of authorship attribution, starting with the controversies surrounding the authorship of French and English literature. Also, Jin and Jiang [16] proposed a method based on extracting the writing characteristics of various authors based on their usage of punctuation marks using text clustering. A comparative analysis was performed between the proposed and character Bi-gram methods using 200 articles by five well-known modern writers. In addition, Stuart et al. [17] proposed measuring certain expressible writing style features. Its uses include author characterization for recognition in text whose authorship is disputed or unknown. Hinh et al. [18] presented a technique that incorporated semantic frames as a method for authorship attribution. They concluded that it provided a deeper view into the semantic level of texts, which influences a writer's style.

A group of researchers proved that the linguistic profiling of chatbots' texts is feasible using the Java Graphical Authorship Attribution Program (JGAAP) by Ali et al. [19]. Numerous features were extracted using chatbot conversation log files and tested against a wide range of classifiers, which allowed for the testing of each feature's performance on the data [20]. The Juola Wyner (JW) cross entropy classifier and Vowel 2–3 Letter words yielded the most promising results. Further, it was observed that a combination of features or stacking classifiers could help improve the accuracy of the results. A research team at Pace University, Goodman et al. [21], also explored a stylometric analysis of biometric keystrokes for authorship attribution. This study provided a starting point for testing and evaluating the use of stylometry for email author identification; however, implementing various methods to determine and apply additional features will need more research [21]. Additionally, this method was not tested by Ali et al. [19], as it is simple to visualize hundreds of extracted statistics but often problematic to resolve which ones are more relevant for authorship attribution.

The use of Word Adjacency Networks (WANs) to attribute the authorship of unknown texts was explored by Segarra et al. [22] using the frequency and relational structure of function words (combining the two methods halves the error rate) to build stylometric fingerprints. Also, the dissimilarity between texts was measured using the relative entropy of Markov chains [23–26] and multi-dimensional scaling [27]. They used a variety of factors, such as "length of the profile", "length of texts", "number of authors", and similarities in writing styles to test the hypothesis. It proved that WANs carry information about periods, the genre of the composition, the gender of the authors, and collaboration between authors. Stylistic features are the attributes or writing-style markers that are the most effective discriminators of authorship [28]. Authors make their choices in various ways and for multiple reasons. These choices are observable to the reader as stylistic variations of the choice between items in a vocabulary. A consistent and distinguishable tendency to make some linguistic choices can be called a style [29]. Stylistic Models allow documents to be analyzed based on stylistic traits that uniquely distinguish one author from another.

Oman and Cook [30] explained that "effective writing is more than observing established conventions for spelling, grammar and sentence structure", similar to how effective programming is more than following style guides, and showed many similarities between writing and programming styles. Just like how an author "exercises perception and judgment in selecting the best suited to his material, audience, and intention from equally correct expressions", a programmer must choose the appropriate operators, keywords, and library functions from which many equally correct options could be selected [30,31].

The research goal in Ref. [32] was to show that it is possible to identify the author of a program by examining programming style characteristics. They defined programmers' coding style as 60 metrics in layout, style, and structure characteristics, then used classifiers to identify the source code authors based on these style metrics. Their result showed that these characteristics could classify programs with less than 2% error rates (Gaussian Likelihood Classifiers and Neural Networks). Macdonell et al. [33] designed 26 measures to show code style and used them in case-based reasoning, Feedforward Neural Networks, and multiple discriminant analysis for discriminating between program authors. The

experiment results showed that their style features worked well; they achieved a high classification rate of 81.1% accuracy with Feedforward Neural Networks (FFNN) and Multiple Discriminant Analysis (MDA) and an 88.0% accuracy with Case-Based Reasoning (CBR). Ding and Samadzadeh [34] analyzed programmers' coding style in 56 metrics and also measured the contribution of 56 metrics by stepwise discriminant analysis. They classified the source code to one author through canonical discriminant analysis. The identification rate could be 62.6–67.2% with the extracted metrics and 85.8% with derived canonical variate. The work of the researchers Lange and Mancoridis [35] designed 38 traits to profile developers' coding styles. They successfully classified 55% of the programs into the correct authors based on these measures using the nearest neighbor with ranking similarity. In another work [36], the authors defined six metrics of programmer style and used a Decision Tree to detect outsourced program submissions. All these researchers extending text authorship attribution and attempting to extract developers' coding style laid a solid foundation for source code authorship attribution.

2.2. Language Models

Language modeling is crucial in modern NLP applications. A language model is a probability distribution over words or word sequences. In practice, a language model gives the probability of a particular word sequence being "valid". Language models are either simple probabilistic or neural network-based.

Document Embeddings, a traditional semi-supervised neural language model, was compared with the Bag of Words (BOW) approach on the PAN 2011 dataset by Agun and Yilmazel [37] to represent a document and create meta-profiles for authors. Document embeddings are superior, as they can be adjusted to other unlabeled domains and re-trained with external resources. These data (PAN 2011 dataset by Agun and Yilmazel [37]) are stored in a much more effective manner, thus allowing them to run computationally complex algorithms efficiently. Another research team proposed a distributed memory model (PV-DM) and a distributed bag of words model (PV-DBOW) [38]. They compared the classification performance of four learning algorithms (Logistic Regression, Naïve Bayes, SVM, and Multi-layer Perceptron classifiers). These classifiers were selected because Naïve Bayes and BOW are usually the baseline measures, and SVMs have been reported as the best-performing classifiers for any text classification task. PV-DBOW significantly improved and created better document representations than BOW in a well-known dataset overall classifier types. It makes tiny, dense vectors independent of the classification model and can be used to construct author fingerprints with efficient distance-based methods in web-scale systems.

By extracting features such as length of sentences and common n-grams that effectively differentiate authors' writing styles over time, Tamboli and Prasad [39] found that authors' writing styles change over time. Also, they found that different authors evolve differently, and each feature can infer temporal changes in the style. They used three approaches: a Profile-Based Approach (all samples of the known text of a author are compiled into a single file), an Instance-Based Approach (each known text sample is considered a separate document), and a Hybrid Approach [12]. They also used a Decay Function with Linear Regression to analyze how different features behave concerning time series data available for authors. It was brought out that several features show temporal changes in increasing and decreasing order, and the language model for each author may differ. Also, Ge et al. [40] explored language models using Neural Network Language Models (NNLMs) and compared their performance with the n-gram models (i.e., 4-gram). NNLM-based work achieves promising results compared with the N-gram models.

A group of researchers, Pratanwanich and Lio [41], compared the Supervised Author-Topic (SAT) model with the Unsupervised Author Topic (AT) Model to learn about author contributions, interests, and topic-specific word distributions. The authors used a collapsed Gibbs sampling to understand model parameters and attribute anonymous documents to authors. The SAT outperforms the AT [42] because of the additional Dirichlet before

representing the author contributions, allowing it to deal with multi-author documents effectively. It also allows for the discovery of topics and author interests. The SAT model outperformed the AT model (superior to the Latent Dirichlet Allocation model [43]), displayed better fitness, could predict writers from 2865 candidates correctly, and ranked the contributions precisely. It was best for single-author documents, and the random forest [44] with five trees was best for multiple-author documents based on ROCs and AUCs; however, random forests do not provide information discovery (author interests and underlying topics). Furthermore, McCallum [45] discusses a Bayesian classification approach, where a Mixed Membership Topic Model represents the multiple classes that comprise a document.

Seroussi et al. [46] explore the Disjoint Author-Document Topic (DADT) Model on several datasets containing formal texts written by a few authors and informal texts generated by thousands of online users. They present experimental results that demonstrate the applicability of topical author representations to inferring the sentiment polarity of texts and predicting the ratings that users would give to items, such as movies. They prove that the DADT yields better results than LDA-based approaches and methods based on the AT models.

2.3. Statistical Models

A statistical model is subjected to a variety of inferential statistics, ranging from the simple analysis of event distributions to a complex pattern-based analysis, where the results of this inference determine the final results (and confidence) of the experiment [13].

In 1887, Mendenhall [47] applied a quantitative analysis of stylistic traits to verify whether a certain number of literary works written by Shakespeare, Francis Bacon, or John Fletcher were correctly attributed. Mendenhall proposed a method to analyze a composition by forming a "word spectrum," or "characteristic curve," which represents an arrangement of words according to their length and the relative frequency of their occurrence. Furthermore, Labbé, Cyril and Labbé, Dominique [48] and Marusenko and Rodionova [49] used a combination of stylistic and statistical methods to solve the authorship attribution and identification problem for the works of Moliere and Corneille. This problem was solved using mathematical modeling methods, and quantitative descriptions of individual author styles on a syntactic level. The attribution of the 'Federalist Papers' was also performed using function words, Poisson, and binomial data distributions, along with a Bayesian approach and discrimination methods by Mosteller and Wallace [50] and Mosteller [51], which eventually became arguably the most famous and widely cited statistical analysis of authorship [13]. Mosteller and F. Wallace's study was based on discrimination with Bayesian analysis methods. Khomytska and Teslyuk [52] used the Java programming language to attribute authors on the phonological language level with a stricter structure, and it proved to be more effective because it allows only a strict function that always evaluates its argument. They differentiated the authorial style based on the mean frequency of eight consonant phoneme groups, each having its capability of style differentiation. Style dominance, established by several consonant groups by which the compared texts differed essentially, further defined the distance between the two styles. Also, three research groups [53–55] used the method of determination of functional style effect and authorial style effect (MFSASE) and the Method of Complex Analysis (MCA) with a combination of ranking and style distance determination methods to test the feasibility of the research on Byron's and Moore's poetry. They showed that utilizing a combination of complex analysis and the multifactor method of determining the degree of style along with the authorial factor improves the efficiency of style and authorship attribution [56,57]. The mean frequency of occurrence was considered a criterion of style differentiation, assuming that the frequency of occurrence of consonant phoneme groups follows normal Gaussian distribution. Also, the Chi-square test, Student's t-test, Kolmogorov-Smirnov test, and Pearson's test proved that constrictive phonemes have the most robust differentiating capability. Additionally, they found that authorship attribution is based on the differentiation of phonostatistical structures.

Inches et al. [58] worked on authorship attribution for a novel set of documents, namely, online chats. It was the first approach to conversational documents using statistical models. This research experimentally demonstrated the unsuitability of the classical statistical models for conversational documents. They proposed a novel approach that could achieve a high accuracy rate of up to 95% for hundreds of authors. Statistical analysis was further explored using probabilistic distance measurements, such as inter-textual distance [48], the Delta Rule [59,60], Latent Dirichlet Distributions [61], and KL Divergence Distance [62,63]; however, statistical analysis has the disadvantage of ignoring the style of the editor claims [2]. Most methods currently employed in computational models rely on multivariate statistical comparisons between features and rules. The features used as statistical variables comprise the relative frequencies of various simple phenomena, such as alphabetic characters, strings of characters, whole words, or common grammatical forms [59]. Thus, statistical models provide a mathematical approach to the authorship attribution problem, using the laws of statistics and information theory to help solve such tasks.

In the field of source code authorship attribution, information retrieval and similarity ranking techniques are widely used. As an extension to Ref. [64] and Ref. [65], source code authorship identification using a fully automated and reliable Source Code Author Profiling (SCAP) approach includes a new simplified profile and a similarity measure performed by Frantzeskou et al. [66]. Byte-level n-gram profiles were better able to capture the idiosyncrasies of the source code authors and were used to show that this method is language-independent. It was tested on a varied dataset with differing characteristics to prove that the method is efficient for small code samples and samples with multiple candidate authors. It also emphasized the significance of comments in classification. Burrows et al. [67] used an information retrieval technique with mean reciprocal ranking and correctly classified 76.78% of query documents in a 10-class classification experiment. Similar approaches were used in the works of Refs. [31,68,69].

2.4. Machine Learning Models

Research on automatic attribution has benefited from the performance of machine learning techniques in the context of automatic classification. It involves inferring an author classification function based on the stylistic and statistical traits extracted from the writings of these authors [2]. This section presents an overview of the machine learning algorithms that have appeared in previous authorship attribution studies with empirical evaluations; however, not all classification algorithms have been covered, as there are far too many (version 3.5.8 of the Weka machine learning toolkit has 114 classification algorithms available [70,71]).

Kothari et al. [72] used the Bayes classifier and Voting Feature Intervals classifier to attribute the authorship of source code by computing the similarity of the developers' profiles. This approach achieved 76% accuracy with 12 authors. Rosenblum et al. [73] used a Markov Random Field [74], which allows for the incorporation of heterogeneous features to classify the function entry points (FEPs) in gaps of binaries. Locating the FEPs is a very first step to binary code analysis. FEPs in gaps of binaries were used with Unary and Binary features, along with local and global byte offset information that allow the model to understand the location of FEPs on a Linux and Windows Binary Files dataset. Shevertalov et al. [75] discretized four metrics using Genetic Algorithms (GA) in order to improve the source code for author classification, then fed them to the nearest neighbor classifier. In this experiment, 75% of 20 projects were correctly classified. Rosenblum et al. [76] used SVM to identify authorship and the k-means algorithm to cluster authorship among stylistically similar programs. This technique can identify the correct author from 20 candidates with 77% accuracy.

Layton and Azab [77] analyzed the Zeus Botnet Source Code using Evidence Accumulation Clustering (EAC) [78–80]. EAC is an ensemble clustering method, where initial clustering using Re-centered Local Profiles [81], specific parameter values and features is

followed by final clustering using the NUANCE Algorithm [79,82] (where features such as common n-gram and distinctiveness help in clustering). It is further paired with manual analysis and the use of algorithms to compute the authorship distance between pairs of source code files to prove that the majority of the code was written by a single author. However, the code had portions that varied in programming style, proving that multiple authors wrote these portions. They proved that analyzing the code by breaking it into smaller segments can effectively improve the output of attribution. They further provided valuable insights into the structure of the botnet, its evolution over time, and its potential to be used to link the malware with other malware written by the same authors. It was shown that clustering is used to aid manual analysis by employing a hierarchical clustering algorithm using a single linkage (which forms a tree-like structure, where samples are combined into nested sets of clusters). The authors also state that Commodity Malware is harder to attribute but easier to block by generating signatures that block this type of attack.

Caliskan-Islam et al. [83] utilized a random forest classifier to identify the most likely author of a code fragment based on the syntactic features obtained only from Abstract Syntax Trees (ASTs) and along with lexical and layout features obtained from source code. This method reached 94% accuracy in classifying 1600 authors. The follow-up work on binaries using the same method [84] obtained attribution accuracy of up to 96% with 100 and 83% with 600 candidate programmers. Using a fine-grained technique, Meng [85] attributed the authors of program binaries with the help of a linear SVM [86]. After extracting features that optimally capture the code properties after compilation, extracting author contribution percentages using Git-author [87], and using basic blocks of code as a unit of testing, the identification of authors was performed to an accuracy of 52% for a dataset with 282 authors. They also found that 88% of basic blocks have a major author who contributes more than 90% of the basic block, while only 67% of functions have such a major author. Meng et al. [88] applied two models to identify multiple authors of a binary program and compared the outcomes of them. On a dataset of 284 authors, the CRF models achieved an accuracy of 65% compared with the accuracy of 58% for SVM models. Zhang et al. [89] first extracted programming layout/style/structure features and programming logic features based on continuous and discrete word-level n-grams. Further, an SVM was used to identify the authorship of source codes or programs. The approach with proposed features achieved an accuracy of 80.28% on 53 authors.

Dauber et al. [90] show that short, incomplete, and typically uncompilable fragments of code belonging to accounts of open-source contributors can be effectively attributed. It used calibration curves that help attribute open-world cases and cases with lower overall accuracy. The calibration curve is obtained by fitting an appropriate equation to a set of experimental data. They achieved an accuracy of 73% overall for attributing single samples, 95% for pairs of samples, and 99% for sets of 15 samples using the Google Code Jam dataset and a random forest classifier with 500 trees [44]. Most samples fell into the lower confidence levels and showed that mid-confidence attributions could be trusted to a high degree.

Cyber attacks recorded by the AWS Honeypot dataset are attributed using K-Nearest Neighbors (KNN) [91], Naive Bayes [92], SVM [93,94], Bayesian Network [95], and Decision Tree [28,96,97] classifiers by Alkaabi and Olatunji [98]. They explored the models and attribute selection techniques even though they restricted the attack to Saudi Arabia. They found that the Bayesian Network classifier showed the best result (98.84%) with all features while selecting the best four features based on the correlation, and the best result was 90.2% achieved by KNN. The SVM model achieved the highest accuracy among the other classifiers—94.79%.

Machine learning techniques were also broadly used in the textual authorship attribution field and comparatively studied. For example, Zhao and Zobel [62] showed that Bayesian Classification outperformed Decision Trees, and Li et al. [99] concluded that support vector machines (SVM) and neural networks offered performances comparable to those obtained by the Naive Bayes algorithm.

Pillay and Solorio [100] demonstrated a two-stage approach for combining unsupervised and supervised learning approaches for performing authorship attribution on web forum posts. During the first stage, the approach focused on using clustering techniques to make an effort to group the datasets into stylistically similar clusters. The second stage involved using the resulting clusters from stage one as features to train different machine learning classifiers. Donais et al. [101] presented a modified Bayesian classifier used internally by ChatSafe (an author attribution system intended for use with short message-based communication, i.e., instant messaging or SMS), which improves the accuracy of a standard Bayesian classifier. Khonji et al. [102] also presented an evaluation of Random Forests [103] on the problem domain of authorship attribution. They took advantage of the classifier's robustness against noisy features by extracting diverse features from evaluated e-texts. Pinho et al. [104] illustrated the performance of a compression-based measure that relies on relative compression, compared with recent approaches that use multiple discriminant analysis and support vector machines. The authors of this paper attained 100% correct classification of the datasets used, showing consistency between the compression ratio and the classification performance.

Machine learning-based authorship attribution systems extract features like the average length of words, the frequency of digits used, the frequency of letters used, etc., and use them to classify documents.

2.5. Deep Learning Models

Over the last decade, representation learning has gained increasing attention in the machine learning community. It has become a field in and of itself dedicated to enabling easier and more distinctive feature extraction processes [105]. Deep learning has successfully captured more useful representations through multiple non-linear data transformations with a higher predictive power among several representation learning methods [106].

Abuhamad et al. [106] utilized TF-IDF-based deep representations and examined the learning process of large-scale, language-oblivious, and obfuscation-resilient code authorship attribution. They used robust Recurrent Neural Network (RNN) layers that were fully connected and dedicated to authorship attribution learning followed by a scalable Random Forest Classifier [44] covering the entire Google Code Jam dataset over four languages. The training process followed a supervised learning approach using TensorFlow's Adaptive Moment estimation with the help of mini-batches for optimization [107], and Dropout Regularization [108], L2 Regularization, and k-fold cross validation [109] to overcome the problem of over-fitting. This method achieved an accuracy of 92.3% with 8903 C++ authors from GCJ. Also, it achieved an accuracy of 93.42% for 120 C authors under obfuscation. For the binaries [110], it achieved 95.74% for 1500 programmers; the performance dropped to 93.86% with obfuscation. They proved that deep code authorship features using deep LSTMs or GRUs [107] enable large-scale authorship identification, even with limited code samples per author. The impact of temporal effects on authorship identification is insignificant. Further, this technique has language-agnostic identification capabilities and applies to a real-world dataset. Zafar et al. [111] used a character-level CNN to output source code vectors, which mapped the source codes from the same author closer, then fed them into the K-Nearest Neighbor (KNN) classifier to identify the author. Their method achieved an accuracy of 84.94% across 20,458 authors. White and Sprague [112] input SentencePiece tokens into a bi-directional LSTM to generate embeddings; then, an SVM and KNN were used to classify the authors on 1600 C/C++ programmers. In closed-world tasks, SVM had an accuracy of 91.50%, while KNN had an accuracy of 90.38%.

Bogdanova [113] constructed an interpretable model using a neural network for source code embedding generation, then fed the embeddings to the KNN classifier. For the visualization of the embeddings, the t-SNE algorithm was used. They also selected the Saliency Map approach to show the focus of the model. The accuracy of this method was 42% on 400 Python authors. Bogdanova and Romanov [114] addressed the inability to add new authors without retraining and the lack of interpretability in source code

authorship attribution. They trained a convolutional neural network to generate the vector representation for files. Then, a KNN classifier to predict the author was given an embedding vector. They also presented the Saliency map algorithm, which assigns an importance value to each input parameter for explainability. They reached an accuracy higher than 70% on datasets consisting of 82–230 programmers with different languages.

In the domain of authorship of textual content, Bagnall [115] used multi-headed RNNs to construct a character language model that was one of the best-performing models on the PAN2015 author identification task. Ruder et al. [116] used multi-channel CNNs and analyzed the influences of different CNN input channels on authorship attribution and identification performance. Yavanoglu [117] introduced a new authorship identification process based on an Artificial Neural Network (ANN) model using embedded stylistic features. The primary dataset contained 22,000 Turkish newspaper articles that belonged to different genres. The results indicated a 97% success rate with the Levenberg Marguardt-based classifier. Ge et al. [40] used a Feedforward Neural Network to create a lightweight language model that performed better than the baseline n-gram method on a limited dataset. Shrestha et al. [118] presented a new model using Convolutional Neural Networks (CNN), which focused on the authorship attribution of short texts. Zhao et al. [119] also uses RNNs to attribute article fragments because of their powerful abilities to represent and extract features. Zhao concluded that deep learning methods overcome the issue of humans choosing features they can represent and extract very effectively. Deep learning representations have enabled several advancements in many machine learning applications, including code authorship attribution, and can solve complex attribution problems.

In addition to using deep learning to learn a different representation of source code, some researchers also used it to classify the authors of source code. Yang et al. [120] utilized a back propagation neural network trained with a particle swarm optimization algorithm to analyze a predefined set of feature metrics, and determined its weights then constructed a neural network-based classification model to attribute the source code authors. The proposed method can achieve an accuracy of 91.06% on 40 JAVA programmers within a reasonable time range. Abuhamad et al. [121] explored the adequacy of different source code representations, as well as different CNN-based deep learning architectures in the performance of the code authorship identification task. This method can distinguish up to 1000–1600 programmers of different languages with different accuracies ranging from 94.6% to 96.2%. Ullah et al. [122] used a deep learning model to attribute the real authors based on Program Dependence Graph (PDG) features. The proposed research gave an average accuracy of 99% on three groups consisting of 1000 programmers in different languages. Kurtukova et al. [123] used deep NN to highlight informative features that were fed to a hybrid CNN and BiGRU (HNN) architecture to identify the author. The average accuracy obtained for all programming languages was 86% with 20 authors. Bogomolov et al. [124] suggested two language-agnostic models, working with path-based representations of code. The first model was a random forest, and the second PbNN model was an adapted version of the code2vec neural network. The PbNN model reached 98.5% accuracy, which was higher than the random forest on 40 programmers. Li et al. [10] worked on making deep learning-based code authorship attribution robust. Their approach incorporated data augmentation and gradient augmentation to learn robust coding style patterns and reduced the success rate of targeted and untargeted attacks by 22.8% and 41.0%. When it came to accuracy, this method eventually improved the average accuracy of the DL-CAIS model from 88.2% to 92.1% on 204 C++ programmers, while lowering the accuracy of the PbNN model on the same dataset.

2.6. Synthesis

Based on this literature review, we can categorize the existing techniques employed for authorship attribution research into the following groups:

1. Stylistic models, which take into account the styles unique to each author.
2. Statistical models that use computational, statistical, and mathematical methods on data to make inferences on their authorship and also on the confidence levels of the results.
3. Language models that work with the probability distributions over words and phrases.
4. Machine learning models that convert the authorship attribution problem into a multi-class classification problem and aim to use the most effective classifier from a set of classifiers to solve the said problem.
5. Deep learning models can capture more distinct features and provide better results than certain machine learning methods for more complex attribution problems.

The following are the shortcomings that have been identified with these existing methods:

1. Statistical methods have the disadvantage of taking neither the lexicon used nor the theme of the suspect document into account.
2. For selecting optimal values of n in n-gram-based language models, a large value is better for lexical and contextual information. Still, it would significantly increase the representation size, and a small value will represent the syllables but will not be enough to describe contextual information.
3. Some feature extraction techniques, such as n-grams, might not be able to capture all dependency features over a long distance in the text.
4. Most research in the field assumes that documents and codes are written by a single author, which is not valid in most cases. Another case is that many files might have a primary author who contributed most of the code, with other authors making minor additions and modifications. These scenarios can affect the accuracy negatively.
5. Finding an adequately large and balanced dataset has always been a challenge in this field. It is much harder to attribute short texts accurately.
6. About the extraction of the style of source code authors, some approaches are language-dependent; therefore, they cannot be used in other languages.
7. For some statistical models and machine learning models, the selection of metrics was not an easy process, for example, thresholds needed to be set to eliminate those less contributive metrics to classifiers.
8. The model accuracy decreased while the number of authors increased, but this issue was more obvious for some machine learning models. For example, the degradation of the accuracy of models such as random forest and SVM is worse than neural networks such as LSTM and BiLSTM.
9. Neural networks cannot generalize well when the number of source code samples is insufficient since there is not enough information that can be analyzed.
10. Most of the existing models cannot provide any explanations for the attribution results.

3. Review on Feature Types and Their Classification

This section discusses the different types of features that are part of the attribution problem involving text and source code-related documents. We discuss the features extracted by various authors and their use in authorship attribution. Figure 2 shows our taxonomy of the types of features, and the rest of this section will explain the proposed categories.

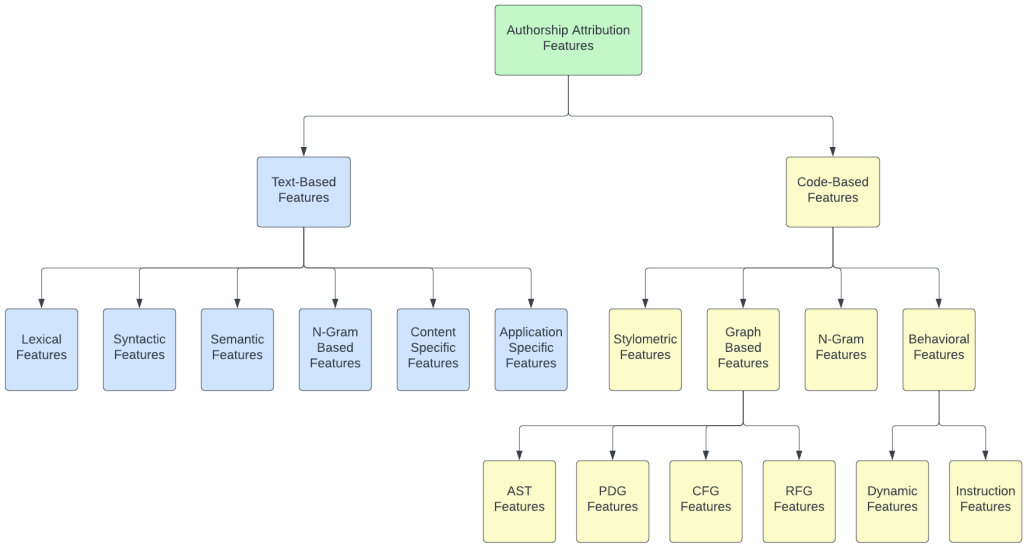

Figure 2. Feature taxonomy.

3.1. Text-Based Features

This section analyzes different linguistic aspects that may help represent style in text-based documents, where text-based features are characteristics extracted from these documents. They are lexical, syntactic, semantic, n-gram-based, content-specific, and application-specific. In the sections below, we discuss each of them in greater detail.

3.1.1. Lexical Features

We discuss the role of lexical words (nouns, verbs, adjectives, and adverbs), among others, in representing style. Mechti and Almansour [2] describe lexical features as those that carry information about writing style and stylistic relationships between words. Tamboli and Prasad [39] identify features (Table 2) that show significant deviation over time to attribute newspaper articles spanning over five years and discuss the suitability of specific features as effective lexical features. The main focus, however, was on cognitive errors, which are critical errors made by the author due to casual bias, uncertain conditions, and unexpected outcomes from specific issues [125]. It was demonstrated that character 4-gram is most promising because its usage patterns imply contextual and thematic variations and provide promising results compared to the bi-gram and tri-gram features.

Function words (articles, pronouns, prepositions, conjunctions, and auxiliary verbs) were explored by Argamon and Levitan [126], Zhao and Zobel [127], Yu [128], and Kestemont [129]. Segarra et al. [22] also stated that they could carry information about the author that is not biased by the topic of the text being analyzed because they are content-independent and explore other useful features (Table 2). Additionally, Koppel et al. [130] used function words, word length distributions, and word contents to attribute a large set of e-mail messages. Yule [131], Mendenhall [47], and Ahmed et al. [132] used lexical features (Table 2) as input data for Markov Chain methods for the attribution. Holmes [133] and Yavanoglu [117] discuss the evolution of stylometry and attribute Turkish newspapers using metadata with the help of lexical features (Table 2). Ali et al. [19] further proved the effectiveness of lexical features in the linguistic profiling of chatbots (Table 2). Zhao and Zobel [62] also explored the effectiveness of lexical features (Table 2) as useful stylistic markers [62,134]. Ramezani et al. [135] studied the effects of textual features on authorship attribution accuracy. The evaluation results showed that the information about the used words and verbs were the most reliable criteria for the authorship attribution tasks; however, lexical information does not give insight into the structure of the sentences and the textual complexity [136]. Abbasi and Chen [28] define lexical features as word or character-based statistical measures of lexical variation, and these are extremely useful in the authorship attribution problem.

Table 2. Lexical features.

Paper	Features
Tamboli and Prasad [39]	Character n-grams (including space, special characters, punctuation), word-to-character ratio, average word length (AWL), average word length per sentence (AWLS), average character length per sentence (ACLS), function word distributions, total lines, words, punctuation, alphabet cases, short words, big words, cognitive error [125]
Segarra et al. [22]	Function words, directed proximity between two words, window length, sentence delimiters
Koppel et al. [130]	Function words, word length distributions, word contents
Argamon and Levitan [126], Zhao and Zobel [127], Yu [128], Kestemont [129]	Function words
Ahmed et al. [132]	Characters, sentence length [131], word length [47], rhyme, first word in a sentence
Holmes [133], Yavanoglu [117]	Number of nouns, verbs, adjectives and adverbs
Ali et al. [19]	2–3 Letter Words, 2–4 Letter Words, 3–4 Letter Words, Character Bigrams, Unicode Character frequencies, Character Tetragrams, Character Trigrams, Dis Legomena, Hapax Legomena, Hapax-Dis Legomena, MW function Words, Words frequencies, Vowel 2–3 letters words, Vowel 2–4 letters words, Vowel 3–4 letters words, Vowel initial words, Word Bigrams, Word Length, Syllables per word, Word Tetragrams, Word Trigrams
Zhao and Zobel [62]	Frequent words
Can and Patton [134]	Number of tokens, token types, average token length, average type length

3.1.2. Syntactic Features

Syntax states the rules for using words, phrases, clauses, and punctuation to form sentences. Syntactic features focus on the syntax and structure of the text. Varela et al. [137] explored authorship attribution of literary texts in a multilingual environment using features grouped by syntactic functions (Table 3). Varela et al. [138] also developed a new approach based on syntactic features of the language (Table 3) for the resolution of cases involving authorship attribution. Wu et al. [139] discussed a Multi-Channel Self-Attention Network (MCSAN) incorporating both the inter-channel and inter-positional interaction to extract character n-grams among other syntactic features to distinguish different authors. The use of syntactic n-grams was researched by Sidorov et al. [140] to prove their effectiveness as features in a machine learning model for NLP applications. Cutting et al. [141], Solorio et al. [142] also explored effective syntactic features (Table 3) for attribution tasks. Baayen [143] analyzed the frequencies of syntactic writing rules and discovered the features important for determining the complexity of the text. Further, Jin and Jiang [16] and Wanner et al. [136] used syntactic features for the characterization of the personal style of an author and discussed its effect on the attribution of authors [117].

From the above works, it is clear that syntactic analysis is a significant part of stylometry and linguistic analysis of texts and can be extremely helpful in showcasing each author uniquely.

Table 3. Syntactic features.

Paper	Features
Varela et al. [137]	Morphological (nouns, determinants, pronouns, adjectives, adverbs, verbs, prepositions, conjunctions, etc.), flexion (number (singular, plural), gender (male, female), a person (first, second, third), time (past, present, future), mode (imperative, indicative, subjunctive), etc.); Syntactic (subject, predicate, direct object, indirect object, main verb, auxiliary verb, adjunct, complementary object, passive agent (of the subject, of the predicate), etc.); Syntactic Auxiliary (article (definite, indefinite), pronouns (demonstrative, quantitative, possessive, personal, reflective), etc.); and Distance Between Key Syntactic Elements (absolute distances between the main verb, subject, predicate, direct object, indirect object, supplement, pronouns, adverbs, adjectives, and conjunctions) and essential terms, accessories of a sentence, such as subject, predicate, and accessories
Wu et al. [139]	N-grams of the characters, words, parts of speech (POS), phrase structures, dependency relationships, and topics from multiple dimensions (style, content, syntactic and semantic features)
Sidorov et al. [140]	Syntactic n-grams
Cutting et al. [141], Solorio et al. [142]	Parts of speech and parts-of-speech taggers
Wanner et al. [136]	Syntactic dependency trees
Baayen [143]	Frequencies of syntactic writing rules, frequency of appearance of coordination or subordination relationships
Jin and Jiang [16], Yavanoglu [117]	Punctuation

3.1.3. Semantic Features

Semantic features, derived from the analysis of the grammatical format of sentences, include the arrangement of words, phrases, and clauses and determine relationships between independent terms in a specific context. They give computers and systems the ability to understand, interpret, and derive meanings from sentences, paragraphs, and any document of a similar kind [144]. McCarthy et al. [145] proposed a method using the WordNet [146] resource for extracting semantic measures (Table 4) to analyze text on over 200 indices of cohesion and difficulty. Yule [147], Holmes [148], and Tweedie and Baayen [149] also explored the use of semantic features pertaining to vocabulary (Table 4) in the attribution of authors [150,151]. Hinh et al. [18] explored incorporating semantic frames to provide a deeper view into the semantic level of texts, which is an influencing factor in a writer's style, by using a variety of online resources in a pipeline fashion to extract information about frames within the text.

Table 4. Semantic features.

Paper	Features
McCarthy et al. [145]	Implicit meaning carried by the individual words—synonyms, hypernyms, hyponyms of words—and the identification of causal verbs to analyze text on over 200 indices of cohesion and difficulty
Yule [147], Holmes [148], Tweedie and Baayen [149]	Vocabulary richness, yule measures, entropy measures such as positive and negative emotion words, cognitive words, stable words (those that can be replaced by an equivalent) [150], and frequencies [151]
Hinh et al. [18]	Semantic frames

Semantic features provide deep insight into the implicit meaning of the text and help model authors better on specific topics and profile authors based on what they are trying to convey.

3.1.4. N-Gram-Based Features

N-grams are continuous sequences of words, symbols, or tokens in a document and can be defined as the neighboring sequences of items.

Holmes [133], Grieve [63], and Stamatatos [12] conclude through their research of the analysis of literary style, evaluation of techniques for quantitative authorship attribution, and a survey of features and techniques used in authorship attribution that n-grams produce better results than lexical characteristics in author assignment. They further conclude that writing styles vary over time or according to the literary genre of the texts, which makes the style detection process, based on linguistic methods, a bit complex [143]. Ragel et al. [152] explored authorship detection of SMS messages using unigrams with the specific aim of comparing how well the algorithms performed under a large number of candidate authors and less testing data against controlled tests. Laroum et al. [153] used the frequency of character n-grams while classifying digital documents. Ouamour and Sayoud [154] used several n-gram features (Table 5) [155] for the attribution of short texts in the Arabic language, and Vazirian and Zahedi [156] used similar features (Table 5) for the attribution of two Persian poem corpora.

Table 5. N-gram-based features.

Paper	Features
Holmes [133], Grieve [63], Stamatatos [12]	N-Grams
Ragel et al. [152]	Uni-grams
Laroum et al. [153]	Frequency of character n-grams
Ouamour and Sayoud [154]	Characters n-grams [155], word n-grams
Vazirian and Zahedi [156]	Uni-grams, bi-grams, tri-grams
Escalante et al. [157]	Character level local histograms

On the other hand, Escalante et al. [157] worked with local histograms that preserve sequential information of documents that are better than character-based n-grams for authorship attribution and are great at modeling writing style. Furthermore, Mechti and Almansour [2] state that the problem with the n-gram features remains the determination of the value of n. A large n can better understand the lexical and contextual information. In fact, a large n would significantly increase the representation size, while a small n (i.e., 2 or 3) would be able to represent the syllables. Still, it would not be enough to represent contextual information.

3.1.5. Content-Specific Features

Content-specific features comprise important keywords and phrases on certain topics [158]. Agun and Yilmazel [37] extracted Document Embedding Representations and Bag of Words Representations of the documents being attributed using a One-Hot-Vector. Pratanwanich and Lio [41] used Word Distributions per topic, Latent Topics, Author Contributions per document, and Vector of Word Counts (Feature Vector), among other features, to predict the authors of the documents being tested.

These features represent the essence of the work that authors work on, focusing more on the content of the work rather than the structure and style. Table 6 shows content specific work with their features.

Table 6. Content-specific features.

Paper	Features
Agun and Yilmazel [37]	Document Embedding Representations, Bag of Words Representations
Pratanwanich and Lio [41]	Word Distributions per topic, Latent Topics, Author Contributions per document, Vector of Word Counts (Feature Vector)

3.1.6. Application-Specific Features

This section discusses features pertaining to specific applications that cannot be generalized. These features serve the specific purpose of aiding in attributing some text for a particular scenario, event, or application. Abbasi and Chen [28] proposed 29 metrics for stylometric analysis of online texts. Marinho et al. [159] extracted the absolute frequencies of thirteen directed motifs with three nodes from the co-occurrence networks and used them as classification features. Khomytska and Teslyuk [52,56] used the mean frequencies of occurrence of eight consonant phoneme groups (Table 7) to attribute documents on the phonological language level. Alkaabi and Olatunji [98] used Packet Arrival Date, Packet Arrival Time, Honeypot Server (Host), Packet Source, Packet Protocol Type, Source Port, Destination Port, Source Country, and Time as the main features to attribute cyber attacks on the AWS Honeypot dataset. The features were selected and grouped based on Correlation Attribute Evaluation and Info Gain Attribute Evaluation. Packet source was ranked as the first attribute that should be selected and is also one of the attributes that attackers can manipulate. These features do not fit into any other category because they are unique to a specific application, task, or problem and comprise multiple types of features.

Kešelj et al. [65] states that techniques used for style marker extraction are almost always language-dependent. Text-based features focus on language's lexical, structural, and semantic constructs. Whether it is punctuation or the number of parts of speech in a text, they bring out the implicit and explicit linguistic properties in a piece of text. They then represent those markers in a manner that helps use such features for further analysis.

Table 7. Application-specific features.

Paper	Features
Abbasi and Chen [28]	Word length, frequency of different length words, count of letters and letter n-gram, count of digits and digit n-gram, bag of words, word n-gram, vocabulary richness, message features, paragraph features, frequency of function words, occurrence of punctuation and special characters, frequency of POS tags, POS tag n-gram, technical structure features, misspellings
Marinho et al. [159]	Absolute frequencies of thirteen directed motifs from co-occurrence networks
Khomytska and Teslyuk [52], Khomytska and Teslyuk [56]	Frequencies of occurrence of eight consonant phoneme groups (Labial—ll, Forelingual—fl, Mesiolingual—ml, Backlingual—bl, Sonorant—st, Nasal—nl, Constrictive—cv, and Occlusive—ov)
Alkaabi and Olatunji [98]	Packet Arrival Date, Packet Arrival Time, Honeypot Server (Host), Packet Source, Packet Protocol Type, Source Port, Destination Port, Source Country, Time

3.2. Code-Based Features

Code-based features can better represent the essence of a programmer's style. They are practical markers of an author's programming fingerprint. Ref. [160] states that differences in coding style make identification of the author possible since the coding style expressed in source code can be quantified. [9] states that the underlying notion of style is that programs written by different programmers are strongly distinguished by quantifiable features of the

programs, known as style markers, which may include various software metrics as well as language metrics, such as n-grams and lexical diversity. We use this section to analyze and review the various code-based metrics extracted for the authorship attribution problem.

3.2.1. Stylometric Features

From 1989 to 2017, the mainstream of research in the source code authorship attribution domain worked on source code stylometric features and proposed different feature sets. Stylometric features can be intuitively understood. The coding habit of indentation is a simple example—it represents a programmer's coding style. Most of the stylometric features were language-dependent, which is specific to a particular programming language, like C++, Python, or Java. Language-dependent features focus on properties such as keywords, syntax, style, and structure, which are variable to different languages [32].

For stylometric features, the most common categories were defined as syntactic (or called structural in many earlier papers), lexical, and layout features. Syntactic features express the structural characters that represent the author's preference for different statements, keywords, nesting sizes, etc. Lexical features demonstrate the author's habit of using certain keywords and identifiers. Layout features focus mainly on code formats, such as code indentation, empty lines, and placement of braces, spaces, and tabs. These features were demonstrated through statistical information, such as count, percentage, frequencies, log of the feature count to total count, and TF/IDF.

Oman and Cook [161] extracted completely typographic Boolean metrics using programs taken from textbook example code for tree-traversal algorithms and one simple sorting algorithm. The researchers further listed 236 style rules from programming style guidebooks and code analyzers that could be used as a base to derive metrics for programming style in their work [162]. Sallis et al. [163] talked about the use of cyclomatic complexity and the use of layout conventions in software forensics. [32] proposed a set of features for C programming language, which contained 60 characteristics captured from programmers' style, structure, and layout features. Cyclomatic complexity is a software metric used to measure the complexity of a software program.

In 1999, Macdonell et al. [33] proposed 26 metrics used in C++ language. Ding and Samadzadeh [34] extracted 56 Java features, out of which 48 were used to build effective Java program fingerprints for software authorship attribution. Lange and Mancoridis [35] discussed the use of histogram distributions to represent a combination of text-based and software-based metrics for author identification in software forensics of the Java programming language. Elenbogen and Seliya [36] talked about six features to establish programmers' profiles based on personal experience and heuristic knowledge. Caliskan-Islam et al. [83] proposed a Code Stylometry Feature Set (CSFS), which contained 29 metrics and was categorized under 3 types: lexical features, layout features, and syntactic features. In 2017, Zhang et al. [89] proposed 18 new features as a supplement to the previous work by other researchers.

Here, we list the features and details of the literature mentioned above in Table 8.

Table 8. Stylometric features.

Paper	Features
Oman and Cook [161]	Typographic Boolean metrics—inline comments on the same line as source code, blocked comments, bordered comments, keywords followed by comments, one or two space indentation occurred more frequently, three or four space indentation occurred more frequently, five space indentation or greater occurred more frequently, lower-case characters only, upper-case characters only, case used to distinguish between keywords and identifiers, underscore used in identifiers, BEGIN followed by a statement on the same line then followed by a statement on the same line, multiple statements per line, blank lines

Table 8. *Cont.*

Paper	Features
Sallis et al. [163]	Cyclomatic complexity of the control flow, use of layout conventions
Krsul and Spafford [32]	Curly brackets position, different style of indentation, different comments style and their percentage, comments and code agree, separator style, percentage of blank lines, line len, len of function, len of function name, parameter name style, len of parameter name, parameter declaration type, number of parameter, variable name style, percentage and number of different style variable names, len of different type of variables, ratio of different type of variables, number of live variables per statement, lines of code between variable references, use of ”ifdef”, most common type of statement, ratio of different type of statement, number of statement output, types of error, cyclomatic complexity, program volume, use of “go to”, use of internal representation of data objects, use of debugging symbols, use of macro, use of editor, use of compiler, use of revision control systems, use of development tools, use of software development standards.
Macdonell et al. [33]	Percentage of blank lines, non-whitespace lines, percentage of different style of comments, percentage of operators with different whitespace position, the ratio of different compilation keywords, the ratio of different decision statements, the ratio of gotos, line len, percentage of uppercase letters, use of debug variables, cyclomatic complexity
Ding and Samadzadeh [34]	Percentage of different curly brackets position, number of space in a different position, percentage of blank lines, number of indentation in a different position, different comments style and their percentage, percentage of comment lines, percentage of different format of statement, percentage of different type of statement, len of different type of variables, ratio of different type of variables, len of function name, percentage of different style of identifier, lines of class or interface, percentage of interfaces, number of variables per class or interface, number of functions per class or interface, ratio of different keywords, number of characters
Lange and Mancoridis [35]	Histogram distributions, relative frequencies of curly brace positions, relative frequencies of the class member access protection scopes, the relative frequency of uses of the three different types of commenting, the relative frequency of the use of various control-flow techniques, the indentation whitespace used at the beginning of each line, the whitespace that occurs on the interior areas of each non-whitespace line, the trailing whitespace at the end of a line, the use of the namespace member and class member operator, the use of the underscore character in identifiers, the complexity of switch statements used in the code, how densely the developer packs code constructs on a single line of text, the frequency of the first characters used in identifiers and the len of identifiers
Elenbogen and Seliya [36]	Number of comments, lines of code, variables count, variable Name len, use of for-loop, program compression size

Table 8. *Cont.*

Paper	Features
Caliskan-Islam et al. [83]	Term frequency of word unigrams, len of line, the ratio of white space, if newline before the majority of the open brace, if the majority of indented lines begin with spaces or tabs, depth of nesting, branching factor, number of parameters, the standard deviation of the number of parameters, log of the number of different elements divided by file len where the elements are keywords, operators, word tokens, comments, literals, functions, preprocessors, tabs, spaces, empty lines, AST features
Zhang et al. [89]	Frequency of comments, number of comment lines, len of comments, frequency of different statements and different format statements, percentage of different variables, variables len, variable name style, percentages of different keywords, percentages of different operators, the format of the operator, percentage of different types of methods, len of methods, use of "return 0", use of "go to", frequency of class and interface, percentage of blank lines, len of lines, the number of leading whitespaces of lines, use of the two-dimensional array

3.2.2. N-Gram Features

N-gram is a contiguous sequence consisting of an n byte, character, or word extracted from the source code. A sliding window with a fixed length of n generates this sequence. Source code authorship attribution through n-gram features can be considered traditional textual authorship attribution. N-gram features can represent part of stylometric features. Language-independent and automatic construction are the two most significant advantages of n-gram features. Language-independent features do not depend on the language but on other properties that can extract the stylistic traits of an author by focusing more on the face value of code rather than the inherent meaning of each term.

Since 2006, Frantzeskou et al. [66] worked on profile generation using n-gram to demonstrate programmers' style and presented the SCAP method, which was considered a milestone. His approach was based on byte-level n-gram profiles, a set of the L most frequent n-grams of the source code. The profile was compared with the candidate authors' profiles based on a similarity measure: the size of the intersection of the two profiles to classify it into an author. The accuracy of their method exceeded 95% in most cases. Frantzeskou stated that n-gram could better capture author idiosyncrasies and may be more effective in authorship attribution, where the dataset contains works written in various programming languages.

From 2007 to 2010, Burrows gradually developed his approach based on n-gram features. Burrows and Tahaghoghi [69] started his approach from the techniques for detecting plagiarism using token-based n-grams and a similarity measure. The source code was scanned, and the token stream was broken into n-grams using a sliding window approach. Then, they used the query document as a search engine query and produced a list of all documents ranked by similarity. So far, Burrows' method can attribute authorship in up to 67% of cases based on mean reciprocal rank.

In 2009, Burrows et al. [67] created 6-gram program representations where the 6-gram was possible with combinations of six feature classes. Since this representation was suitable for retrieval systems, they queried each document against all works from different combinations of ten randomly selected authors. Eventually, their approach correctly classified 76.78% of query documents. Burrows et al. [68] evaluated the effectiveness and robustness of their method through the timeline. They still tokenized code into overlapping 6-grams but used a TF×IDF approach to rank the query code against all other documents. They discovered features robust to the evolution of coding style and used them to improve the method. Eventually, the effectiveness of the method jumped to 82.35%. In

2010, Burrows [31] extended his approach using the Okapi BM25 similarity measure. This approach had an accuracy of around 90% on student programming assignments.

Approaches proposed by Frantzeskou and Burrows had a considerable influence on source code authorship attribution. Some researchers attempted to improve and compare these two methods. In 2013, Tennyson and Mitropoulos conducted a replicated experiment to compare Burrow's approach and the SCAP approach in Ref. [164]. Their conclusion was that when employing anonymized data, the SCAP method surpassed the Burrows method by a slight margin on 15 authors from Planet Source Code, whereas with non-anonymized data, the SCAP method outperforms the Burrows method by a significant margin. In 2014, Burrows compared several approaches in this domain directly with the same evaluation methods on the same datasets in his paper [31]. The results agreed with Tennyson's conclusion, although Burrows expanded the dataset size, showing that Burrows's approach was slightly less accurate than the SCAP approach on two freelance collections consisting of 100 authors and 76 authors. In 2014, Tennyson and Mitropoulos [164] worked on a selection of parameter L's value in the SCAP method, which defined the profile's maximum length. Instead of eliminating entire quantiles of n-grams, only the n-grams that occurred less than four times were eliminated and evaluated separately in the experiment. The results showed that the "Omit n-grams with frequency = 1" approach performed the best, and the accuracy was improved from 91.0% to 97.2%.

In addition to the works of Burrows and Frantzeskou, other researchers also explored the use of n-gram features. Zhang et al. [89] extracted programming logic features based on continuous and discrete word-level n-grams, then constructed the author profile using logic features along with layout, style, and structure features. Further, a method based on sequential minimal optimization for SVM training was used to identify the authorship of source codes or programs. This approach with proposed features achieves an accuracy of 98.4% on eight authors and 80.28% on 53 authors.

Layton and Azab [77] used the Recentred Local Profiles (RLP) method, which profiled an author's coding style by using the L most distinctive n-grams. RLP stated that the features are used to determine at what level clusters could be considered coming from the same author or to show evidence against such a hypothesis.

3.2.3. Graph-Based Features

Graph-based features can reflect structural traits underlying the code lines, such as an author's tendency to create deeply nested code. Program dependency, control flow, data flow, and syntactic entity construction styles can be fetched through graph-based features. One merit of graph-based features is that they are robust against obfuscation. In the following sections, we will elaborate on several types of graph-based features.

AST Features

Abstract syntax tree (AST) is extracted from the source code by the compiler, which keeps syntactic entities. The nodes of an AST correspond to different code constructs, such as statements, expressions, operations, and variables, while the leaves correspond to identifiers, keywords, and operators. Since AST only keeps the structural and content details and ignores formatting details, it captures the syntactic features well, such as a programmer's tendency to use long chains of assignments. Depth of AST nodes, node n-gram, and node types are the most commonly used AST features.

Due to the nature and advantages of AST, Pellin [165] utilized AST to determine code authorship. They broke down the AST translated from source code into functions, and each function tree was considered a document. This collection was fed to an SVM to predict the author from two authors. But this early attempt only achieves between 67% and 88% classification accuracy. Later on, Caliskan's work on AST properties drew much attention from researchers in this domain. Caliskan-Islam et al. [83] worked on the coding characteristics derived from AST. Besides lexical and layout features obtained from source code, the features obtained from ASTs were categorized as syntactic features. With the

feature set, source code fragments were expressed as numerical vectors to perform feature selection using information gain, then fed to a random forest classifier to identify the author. Their method reached 94% accuracy in classifying 1600 authors.

Alsulami et al. [166] worked on how to automatically extract AST features instead of manually extracting them. They implemented Long Short-Term Memory and Bidirectional Long Short-Term Memory models to automatically extract AST features and used a random forest classifier for authorship attribution. Their method's accuracy reached 88.86% on a dataset of 70 Python programmers. Dauber et al. [90] extracted the AST feature vector after breaking down the files into smaller fragments, fed them into a random forest to predict the author and averaged the classifier probability of linked samples.

Bogomolov et al. [124] suggested path-based representations of code and used two language-agnostic models to classify. In this representation, path contexts consisted of a path between two leaves in an AST and tokens corresponding to start and end leaves. The path features were denoted as term frequencies of tokens and paths and then fed to the classifiers. The experiment showed that the PbNN model reached an accuracy of 97.9%, while the PbRF model achieved 98.5% on the existing JAVA dataset. However, the accuracy went down to 22% on GitHub because similarity values decreased.

Program Dependence Graph (PDG) Features

Another graphical representation of the source code is the Program Dependence Graph. In a PDG, vertices correspond to programming expressions, variables, conditions, and method calls. The data dependencies were edges between two vertices; control dependencies could be extracted when v1 controlled v2. Ullah et al. [122] worked on PDG. They extracted control flow and data variations features from the graph to identify authors. Then, they used a deep learning algorithm to analyze the data and control features. This method reached a 99% accuracy on 1000 programmers.

CFG Features

A Control Flow Graph is a directed graph comprised of the basic block nodes, edges representing control flow, and the type of the edge denoted as a labeling function. In 2011, Rosenblum et al. [76] presented an algorithm that leverages CFG and the instruction sequence to automatically extract stylistic features from binary codes. They then utilized SVM (support vector machine) for authorship identification and employed the k-means algorithm to cluster authorship among programs exhibiting similar styles. With a 77% accuracy, these techniques successfully determined the correct author from a set of 20 candidates. Meng et al. [88] devised four types of new features based on binary code, including control flow features that described the incoming and outgoing CFG edges and context features, for example, the width and depth of a function's CFG. These features with a CRF model can discriminate 284 authors with 65% accuracy.

RFG Features

Flow and dependencies between the registers can show the behavior of a program and capture the semantic style of code. Alrabaee et al. [167] described how to construct a Register Flow Graph and use it in binary code authorship attribution. They devised a novel layered approach, where one layer was the Register Flow Analysis Layer. This layer captured features corresponding to the register to manipulate patterns from the Register Flow Graph. The graph-based features and details of the literature mentioned above are in Table 9.

Table 9. Graph-based features.

Paper	Features
Pellin [165]	AST function tree
Caliskan-Islam et al. [83]	Maximum depth of an AST node, term frequency AST node bigrams, term frequency of 58 possible AST node type excluding leaves, term frequency-inverse document frequency of 58 possible AST node type excluding leaves, average depth of 58 possible AST node types excluding leaves, term frequency of code unigrams in AST leaves, term frequency-inverse document frequency of code unigrams in AST leaves, average depth of code unigrams in AST leaves
Alsulami et al. [166]	AST features generated by neural network
Dauber et al. [90]	AST features from small fragments
Bogomolov et al. [124]	AST-path-based representations
Ullah et al. [122]	Control flow and data variations features from PDG
Rosenblum et al. [76]	CFG Features
Meng et al. [88]	Control flow features from CFG, context features from CFG (for example, the width and depth of a function's CFG)
Alrabaee et al. [167]	RFG Features (register manipulate patterns)

3.2.4. Behavioral Features

One type of feature that makes source code different than text is that source code has behaviors when it communicates with the computer, such as instructions and the use of hardware.

Dynamic Features

Dynamic features are generated in the stage of code compilation and execution, which collects runtime measurements, such as memory use, CPU use, and data structure use. These features can capture programmers' tendencies; for example, memory access patterns can reflect the author's use of an array over the heap. Dynamic features can also reveal the hidden functionalities and behaviors from source code. Therefore, this brings an advantage for dynamic features—they are hard to obfuscate. Another advantage is that most dynamic features are universal across various languages. Dynamic features are considered a good complement to static features. Most dynamic analyses were based on mobile applications, Ref. [168] summarized several common approaches based on dynamic features, such as power consumption, CPU and memory features, and system call features.

On the other hand, a limited number of studies worked on dynamic features from source codes other than mobile applications. Wang et al. [169] introduced four classes of dynamic features—memory dynamics, CPU dynamics, disk dynamics, and network dynamics—and analyzed seven dynamic features. After evaluation of all the extracted static and dynamic features, they found out that most of the dynamic features were not considered high-valued features, except the memory access patterns and function memory usage. The dynamic features and details of the literature mentioned above are in Table 10.

Table 10. Dynamic features.

Paper	Features
Wang et al. [169]	Function call, module running time, total running time, function memory usage, total memory usage, memory access patterns (range of memory addresses and operation frequencies), len of disassembled code

Instruction Features

Another class of behavioral features is instruction features, which are retrieved from binaries. These features reflect valuable code properties, such as operated data types, data access patterns, memory numbers, registers, and immediate operands. Rosenblum et al. [76] used idiom templates to capture instruction sequence underlying a program; they also considered byte n-grams as a relaxation of the idiom instruction abstractions, which captured specific instruction opcodes and immediate and memory operands. Meng et al. [88] used instruction features that describe instruction prefixes, operand sizes, and operand addressing modes to identify authors of binary programs. The detailed information about instruction features is listed in Table 11.

Table 11. Instruction features.

Paper	Features
Meng et al. [88]	Instruction features: instruction prefixes, instruction operands, constant value in instructions. Control flow features, data flow features, and context features.
Rosenblum et al. [76]	Idiom feature capturing low-level details of the instruction sequence.

3.2.5. Neural Networks Generated Representations of Source Code or Features

In recent years, there has been a trend that applies neural networks to produce representations of source code or features. Abuhamad et al. [106] designed a feature learning and extraction method using deep learning architecture with RNNs (TF-IDF as input since it represents code files) and proved that TF-IDF features described an author's preferences on using certain terms and also help minimize the effect of non-useful frequent terms. The feature selection operators considered were the order of term frequency, chi-squared value, mutual information, and univariate feature selection. Finally, from n features, the top k features that reduce the dimensionality were selected. The system took the selected TF-IDF as the initial representation for code files and generated TF-IDF-based deep representation using an architecture with multiple RNN layers and fully connected layers. The resulting representation was then passed into a random forest classifier, ensuring scalability and the ability to de-anonymize the author. This system achieved an accuracy of 96% with 1600 authors and 92.3% with 8903 C++ authors for GCJ. Also, it achieved an accuracy of 93.42% for 120 authors under obfuscation. In the following year, Abuhamad et al. [121] kept exploring how to use neural networks to generate representations of source code and features. They employed a deep learning architecture based on convolutional neural networks (CNN) to generate TF-IDF and word embedding representations that were both high-quality and distinctive. In the final layer of the architecture, a softmax classifier was utilized to classify authors. This method can distinguish up to 1600 C++ programmers with 96.2% accuracy, 1000 Java programmers with 95.8% accuracy, and 1500 Python programmers with 94.6% accuracy.

In 2020, Zafar et al. [111] conducted research focusing on using deep metric learning to generate a new representation. They used a character-level CNN to convert the input character stream into a compact vector, ensuring that source codes from the same author were mapped closely together. They then fed these vectors into the KNN classifier to

determine the author. Their approach achieved an accuracy of 84.94% across 20,458 authors. Similarly, in 2021, White and Sprague [112] worked with metric learning to address the problem of authorship identification and verification. It leverages a bi-directional LSTM to generate embeddings that group similar and separate dissimilar samples. The metric learning approach makes it possible to measure similarity in the learned embedding space. Afterward, a support vector machine is trained to classify the authors based on these embeddings. In closed-world tasks, SVM had an accuracy of 91.50%, while KNN had an accuracy of 90.38%.

3.2.6. Select Influential Features

Among all the features proposed by different researchers, what are the most informative and contributive features? In 2004, Ding and Samadzadeh [34] investigated the software metrics of Java source code. The contributions of the 56 metrics to authorship identification were measured by canonical discriminant analysis. Eventually, 48 metrics were identified as the most contributive ones. Kothari et al. [72] utilized Shannon's information entropy as a selection criterion to find the 50 most contributive metrics. Lange and Mancoridis [35] used a genetic algorithm to select optimal metrics.

In 2008, Frantzeskou et al. [170] identified high-level features used in the SCAP method. The significance of each feature was assessed through a series of experiments, where one feature was eliminated at a time. The findings revealed that comments had a significant impact, followed by layout as the next influential factor. The classification accuracy did not appear to be influenced by variable and function names defined by the programmer. Burrows et al. [67] generated 63 possible combinations of 6 classes of features and created different 6-gram source code representations based on the feature combinations. Then, they used these program representations to predict the programmers. Eventually, they found the most contributive feature sets from the highest accuracy classification. Rosenblum et al. [76] performed feature selection by ranking features through mutual information (MI) between the feature and the true author label. To reduce the feature set size and its sparsity, Caliskan-Islam et al. [83] only kept features with non-zero information gain and eliminated other features. This feature set showed robustness and consistency in different source code files.

Wisse and Veenman [171] empirically determined the most frequent node n-grams, then greedily eliminated infrequent n-grams. Wang et al. [169] accessed each feature's importance by calculating the absolute average weights on the 400 outgoing connections of the corresponding input neuron in the DNN. In 2020, Kurtukova et al. [123] used a deep neural network to highlight informative features in vector form. These features work with a hybrid CNN and BiGRU (HNN) architecture, which could obtain an accuracy of 86% with 20 authors in the simple case and exceeded 80% in the complicated ones. Bogomolov et al. [124] employed feature filtering based on mutual information (MI) when the size of path-based representations was significant.

3.2.7. Explanation of Attribution through Features

One trend in source code authorship attribution is to explain the decisions of classifiers through features. Recently, some new techniques were introduced. In 2014, a detection tool, DREBIN, was introduced by Arp et al. [172]. DREBIN inputs the feature vector into a linear SVM to calculate the result, which determines the classification of the source code. In the calculation process, each feature vector will be assigned a weight; some weights will be high enough to determine or shift the classification. They used the highest k-weight features to generate explanations. Melis et al. [173] extended Drebin to any black-box machine-learning algorithm. They computed another approximate feature-relevance vector to replace the local gradient and used it to find the most influential features.

Ribeiro et al. [174] proposed The Local Interpretable Model-Agnostic Explanations (LIME) technique, which was defined as an explanation of a potentially interpretable model. When given a specific instance that requires an explanation, it starts to take samples from instances around it, and then obtain predictions using the original model. These predictions

were input to an explanatory model, which generated explanations for the original model. LIME is considered model-agnostic, meaning it can be applied to any model, and it aims to provide locally faithful interpretations. Murenin et al. [175] utilized the LIME technique and proposed a solution that could display the most and the least used keywords and features, so as to identify the authorship in Android applications and provide the explanations for the attribution decisions made by classifiers.

Bogdanova and Romanov [114] presented the Saliency map algorithm for the Source Sode Authorship Attribution (SCAA), which is interpretable. It assigned an importance value to each input parameter. The inputs are dimensional vectors for each token. As a result, each token is assigned d importance values. They normalize these values using maximum values and then average these scores. The output of the Saliency map algorithm is color maps of the source code, which highlight the most important features; therefore, they had to gain the explanation through manual analysis.

3.3. Synthesis

Text-based features

The limitations of the text-based features include:

1. Lexical information tends to overlook the structure of the sentences and the textual complexity.
2. For n-gram features, a large n would significantly increase the representation size.
3. N-gram features fall short when representing the contextual information.
4. Content-specific features focus more on the essence content of the work; the weakness lies in their inability to consider the structure and style of the text.

Code-Based Features

Several downsides of code-based features are concluded as below.

1. Stylometric features are usually language-dependent; some traits contribute to one programming language but cannot be used in another language.
2. Available keywords, statements, and structures in source code are limited and restrained by guidelines. Stylometric features are not very distinguishing for coding style.
3. Some style-related and layout-related characteristics are distinguishing but easily changed by code formatting tools, so they lack robustness to obfuscators.
4. Stylometric features have to be extracted manually or by tools, which is time-consuming. Stylometric features have to be extracted manually or by tools—manually is time-consuming, and by tools is usually quick.

One limitation of n-gram features for source code is that the number of n-gram features grows rapidly with the increasing size of n, so a portion of n-gram features might be easily cut off because of the large size.

For graph-based features, extracting graph-based features is a difficult process; they have to be extracted by certain tools, and some tools are language-dependent or work only for complete code.

The notable weaknesses of dynamic features include:

1. An extra procedure is needed to retrieve dynamic and instruction features instead of static analysis, and it needs to be compiled and executed under a specific running environment.
2. They can be obtained only from executable source code, which is hard to extract from code fragments.

In investigating sets of features, most researchers worked on small groups of features. Some researchers examined limited combinations of feature categories, for example, the combination of AST features and n-gram features, and claimed to have an outstanding performance. But wider categories of features needed to be covered.

Most works cannot provide feature-wise explanations for the attribution results. Only very few researchers attempted to work on the explainability of their classifiers and profile of the authors, and their effectiveness needs to be examined.

4. Review on Available Datasets

This section presents a review of available datasets that are mostly used in authorship attribution research. We also discussed popular data sources and their features with their shortcomings. Most of the research in source code authorship attribution used programs from Google Code Jam (GCJ: it is an international programming competition) and GitHub (a code hosting platform for version control and collaboration). Other sources include Codeforces (a contest-based programming website), Freecode, formerly Freshmeat (a website hosting open-source software for programmers), Planet Source Code (a free source code database and sharing website), SourceForge (a complete business software and services comparison platform), student programming assignments from universities, and program examples from programming textbooks.

For a better view of datasets used in the literature, a list of datasets from several researchers that used datasets of a larger size are in Table 12.

Table 12. Datasets used in the literature.

Paper	Features	Classification Model	Language	Author Number	Files Per Author	Data Source	Accuracy
Abuhamad et al. [121]	NN representation	3S-CNN	C++	1600	9	GCJ	96.20%
Abuhamad et al. [121]	NN representation	3S-CNN	JAVA	1000	9	GCJ	95.80%
Abuhamad et al. [121]	NN representation	3S-CNN	Python	1500	9	GCJ	94.60%
Abuhamad et al. [121]	NN representation	3S-CNN	C	745	10	Github	95.00%
Abuhamad et al. [121]	NN representation	3S-CNN	C++	142	10	Github	97.00%
Ullah et al. [122]	PDG features	Deep learning	C#	1000		GCJ	98%
Ullah et al. [122]	PDG features	Deep learning	JAVA	1000		GCJ	98%
Ullah et al. [122]	PDG features	Deep learning	C++	1000		GCJ	100%
Zafar et al. [111]	NN representation	KNN	multiple	20458	9	GCJ	84.94%
Zafar et al. [111]	NN representation	KNN	C++	10280	9	GCJ	90.98%
Zafar et al. [111]	NN representation	KNN	Python	6910	9	GCJ	84.79%
Zafar et al. [111]	NN representation	KNN	JAVA	3911	9	GCJ	79.36%
Abuhamad et al. [110]	NN representation	RFC	Binary	1500	9+	GCJ	95.74%
Abuhamad et al. [110]	NN representation	RFC	C	566	7	GCJ	94.80%
Caliskan-Islam et al. [83]	Stylometric features	RFC	C,C++	1600,	9	GCJ	92.83%
Caliskan-Islam et al. [83]	Stylometric features	RFC	Python	229	9	GCJ	53.91%
Abuhamad et al. [176]	NN representation	RFC	C, C++	562		Github	93.18%
Abuhamad et al. [176]	NN representation	RFC	C, C++	608	6	Github	91.24%
Abuhamad et al. [176]	NN representation	RFC	C, C++	479	10	Github	92.82%
Abuhamad et al. [110]	NN representation	RFC	JAVA	1952	7	GCJ	97.24%

Table 12. *Cont.*

Paper	Features	Classification Model	Language	Author Number	Files Per Author	Data Source	Accuracy
Abuhamad et al. [110]	NN representation	RFC	Python	3458	7	GCJ	96.20%
Abuhamad et al. [110]	NN representation	RFC	C++	8903	7	GCJ	92.30%
Abuhamad et al. [110]	NN representation	RFC	Binary	241	9+	Github	92.13%
Abuhamad et al. [110]	NN representation	RFC	C++	147		Github	95.21%
Abuhamad et al. [110]	NN representation	RFC	C	745		Github	94.38%
Abuhamad et al. [176]	NN representation	RFC	C, C++	346	20	Github	95.12%
Abuhamad et al. [176]	NN representation	RFC	C, C++	282	30	Github	96.14%
Caliskan et al. [84]	Stylometric/ instruction features	SVM	Binary	600	8	GCJ	71%
Caliskan et al. [84]	Stylometric/ instruction features	RFC	Binary	600	8	GCJ	83%
Caliskan et al. [84]	Stylometric/ instruction features	RFC	Binary	50		Github	65%
White and Sprague [112]	NN representation	SVM	C,C++	1600		GCJ	91.50%
White and Sprague [112]	NN representation	KNN	C,C++	1600		GCJ	90.38%
Bogdanova [113]	NN representation	KNN	Python	400		GCJ	42%
Bogdanova and Romanov [114]	NN representation	KNN	C++	230		GCJ	74.4%
Bogdanova and Romanov [114]	NN representation	KNN	JAVA	82		GCJ	77%
Bogdanova and Romanov [114]	NN representation	KNN	Python	115		GCJ	73%
Meng et al. [88]	Instruction, control/data flow, context features	CRF	Binary	284		3 open source projects	65%
Meng et al. [88]	Instruction, control/data flow, context features	SVM	Binary	284		3 open source projects	58%

4.1. Sizes of Datasets

Most researchers using statistical models in their works conducted their experiments on datasets with a size of less than 100 authors. The largest dataset used in these works was 272 authors in Ref. [31], who developed his method on information retrieval techniques. The sizes of datasets the researchers used in most works utilizing machine learning models were less than 300. However, Caliskan-Islam et al. [83] used a dataset with a maximum of 1600 authors. Deep learning models work with much bigger size of datasets. Zafar et al. [111] used the most extensive dataset, covering 20,458 authors.

4.2. Sample Size Per Author

The number of samples per author influences the success rate of authorship attribution. Although there is no consensus on the sample size per author, most researchers used around seven files per author to implement their approaches. Caliskan-Islam et al. [83]

suggested using nine files per author to obtain satisfying accuracy. On the other hand, Abuhamad et al. [106] examined the accuracies of 5/7/9 files per author with different languages, and the result showed that it obtained the highest accuracy with nine files per author. However, the accuracy degradation between nine and seven files was minor; they claimed that only seven files per author are enough for a high attribution accuracy. Abuhamad et al. [176] concluded that under Request for Comments (RFC), the accuracy was improved from 92.12% to 94.4% when the sample size per author was increased from 10 to 30, but when the sample size per author kept expanding, the accuracy decreased. Meanwhile, more code lines per sample (from 4 to 10) led to more successful attribution.

4.3. Dataset from "Wild"

The dataset in the wild is a set of known candidates that are extremely large (possibly many thousands), but it might not even include the actual author. Abuhamad et al. [106] stated that "reported results using the GitHub dataset show some accuracy degradation compared to the results obtained using GCJ dataset given the same number of programmers". In the work [84], the random forest (RF) classifier obtained an accuracy of 83–96% on GCJ datasets sized from 100 to 600 authors but only achieved an accuracy of 65% on 50 programmers from GitHub. Abuhamad et al. [121] used the three-slices convolutional neural network (3S-CNN) method and showed 99.4% accuracy on 150 programmers from GCJ, but the accuracy dropped to 97% on 142 programmers from GitHub. The lower success rate on GitHub might be the reuse of the source code from other authors, multiple authors collaborating on the same project, and the authors being out of the training data, also known as an open-world problem.

To deal with the problem of specifying the authors of source code, Meng et al. [87] brought up a Git built-in tool named Git-author that derived from the structural authorship model and weighted authorship model, which calculated authorship weights given code change measurement between commits. Similarly, Bogomolov et al. [124] suggested collecting data from GitHub through Git-commit because each commit had a single author, and this made Git a rich source of data.

For other issues of authorship attribution "in the wild", Dauber et al. [90] proposed another solution. Firstly, they identified the author of code fragments instead of the entire file in case more than one author wrote the file. After breaking the files into smaller fragments, their experiment extracted the AST feature vector, fed them into a random forest to predict the author, and averaged the classifier probability of linked samples. Secondly, to tackle the open-world problem, they employed the calibration curves to set a threshold below which to identify which samples are written by unknown authors or determine the trustworthiness of the attributions. Unfortunately, their attempt did not display a satisfying result.

4.4. Synthesis

Based on the dataset information we summarized, the following are the shortcomings of the use of datasets:

1. There are no benchmark datasets with large sizes, and they are widely accepted by researchers in the domain of source code authorship attribution.
2. There is no consensus on the sample size per author.
3. Some researchers claimed to obtain high accuracies, but the experiments were conducted on small-size datasets, so the results lacked soundness.
4. Google Code Jam data are more reliable than GitHub due to the fact that all the files are solutions for limited coding questions and the author labels to the files are more reliable. However, most authors have more advanced skills and experiences, so GCJ data are also more imbalanced.
5. GitHub data have a large size of the source file that contains the most realistic programs, but they come with issues relating to the collaboration of multiple authors and code reuse. Although some researchers brought up solutions to these issues, such

as the use of Git commit and Git-author, most authorship attribution tasks are for single-author attribution, and this makes the authorship attribution on the GitHub dataset problematic.

5. Review on Model Evaluation Metrics

In this section, we present the various evaluation metrics used for the performance assessment of the authorship attribution models. We grouped the evaluation metrics and discussed them.

5.1. Accuracy

Accuracy is one of the common model evaluation metrics used for the evaluation of the authorship attribution models. Based on the analysis of 12 studies that utilized datasets containing a substantial number of authors, we compiled findings regarding the combination of feature sets and classifiers and their corresponding accuracies.

The neural network usually works with code representation, displaying accuracy fluctuations spanning from 62% to 99.5%, influenced by representation types and neural network architectures. Notably, Abuhamad et al. [121] revealed that TF-IDF outperformed word embedding in accuracy, while the three-slices convolutional neural network (3S-CNN) exhibited good performance compared to a single slices convolutional neural network (S-CNN). Abuhamad et al. [176] employed RNN alongside TF-IDF, yielding 80% accuracy, whereas word2vec representation achieved 76% accuracy.

Neural networks can work along with manually extracted features, their performance contingent upon the quality of the features employed. For instance, when PDG features were utilized as input for neural networks, the accuracy reached an impressive 99% [122]. In contrast, using AST features yielded an accuracy of 88% in Ref. [166].

Random forest classifiers (RFC) demonstrated that accuracy varies from 72% to 98%, determined by feature quality. For instance, RFC's accuracy was 72% with AST features [166], 83% with stylometric features [84], and 92%–98% in the study of [83] employing the same stylometric features. RFC combined with code representation gained popularity for effectively handling large inputs while maintaining satisfactory accuracy, achieving over 90% accuracy in studies [106,110,176].

K-Nearest Neighbors (KNN) paired with neural network-generated code representations emerged as a prevalent approach. There is a large accuracy variation ranging from 42% to 94% in Refs. [111–114]. Its highest performance slightly trails behind RFC paired with NN-generated code representations.

Support vector machines (SVM) were widely used, especially in the early studies in this domain. The accuracy of support vector machines (SVM) varies from 61% to 91%, which is relatively lower than another approach. In Ref. [166], SVM obtained a lower accuracy of 61.28% using the same AST feature set compared with RFC and neural network. Caliskan et al. [84] showed that SVM with the same stylometric features had a lower success rate of 71% compared to 83% accuracy of RFC.

5.2. Mean Reciprocal Rank and Mean Average Precision

Mean reciprocal rank and mean average precision are another group of metrics that are used to evaluate the performance of the methods in information retrieval techniques, including authorship attribution. Burrows and Tahaghoghi [69] used these metrics to evaluate their approach since it used information retrieval techniques. Their approach can attribute authorship in up to 67% of cases based on mean reciprocal rank, and the mean average precision was 15.55% on 100 authors.

5.3. Precision, Recall, and F-Measure

Some other performance evaluation metrics are precision, recall, and F-measure, which usually come with accuracy. Precision, recall, and F-measure metrics are the classification metrics that are also used in the authorship attribution methods, where precision finds

the number of positive class predictions that actually belong to the positive class, Recall measures the number of positive class predictions made out of all positive classes, and F-measure provides a single score that balances both precision and recall.

Alrabaee et al. [177] investigated several techniques for malware binaries. Since the application domain targeted by binary authorship attribution works is much more sensitive to false positives than false negatives, they employed F-measure to evaluate the performance. The multi-layer approach proposed by Alrabaee et al. [167] that used RFG and other features achieved 58% for the F-measure score on 50 authors and 28% for the F-measure score on 179 authors. The method proposed by Rosenblum et al. [76] that utilized SVM with CFG and the instruction sequence reached 43% for the F-measure score on 50 authors. Caliskan et al. [84] used RFC working with stylometric features, which obtained 60% for the F-measure score on 140 authors. Ullah et al. [122] also evaluated their method besides accuracy. Their method of PDG features with neural network reached 97% for precision, 100% for recall, and 99% for F-measure on the C++ dataset; 99% for precision, 96% for recall, and 98% for F-measure on the JAVA dataset; and 100% for precision, 98% for recall, and 99% for F-measure on the C# dataset.

5.4. Algorithm Complexity

The other performance evaluation is complexity, although few studies have explored this factor. Zafar et al. [111] attempted to enhance the complexity by employing a KNN that utilized the ball trees data structure instead of naive KNN or RFC. Their implementation had a complexity of $O(d \log n)$, while the complexity of naive KNN was O(dn) and the complexity of RFC was $O(dt)$ for prediction and $O(n^2 dt)$ for training. Here, n was the number of samples, d was the number of dimensions, and t was the number of trees.

Some researchers working on neural networks are more concerned about the complexity since the complexity of neural networks increases, and more running time is needed as the problem scale becomes larger. Consequently, distillation has emerged as a solution to reduce the computational complexity of neural network architectures.

5.5. Attribution Accuracy under Obfuscation

Only some of the works assessed the attribution accuracy under obfuscation, and the performance under obfuscation is determined not only by the robustness of the approaches against obfuscation but also affected by the types of obfuscators since different obfuscators work on different languages and different aspects of code features. Therefore, it is hard to compare the performance under obfuscation of all the works. The methods used in Ref. [83] fed stylometric features, including AST features into RFC to attribute authorship, and eventually obtained 94% accuracy in classifying 1600 authors. In the scenario of obfuscation, the accuracy reached 98.89% under the Stunnix obfuscator on 20 C++ authors and 67.22% under the Tigress obfuscator on 20 C authors.

Alrabaee et al. [177] examined the performance under refactoring and obfuscation of several binary authorship attribution works. Regarding OBA2 [167], they concluded that the accuracy decreased from 81% to 62% under refactoring to 58% under obfuscation. From this, we can see that OBA2 was vulnerable to some techniques, such as variable rename. Regarding Caliskan-Islam et al.'s method in binaries [84], which used RFC with stylometric and AST features, the accuracy decreased from 79% to 70% under refactoring to 24% under obfuscation. Their method utilized AST features that displayed robustness to refactoring, but they extracted features through a decompilation process, which can be affected by Flatten Control Flow graph obfuscation techniques and therefore displayed vulnerability to obfuscation. Regarding Rosenblum's method [76], which leverages CFG and the instruction sequence to work with SVM, the accuracy decreased from 66% to 40% under refactoring to 27% under obfuscation. The reason for their vulnerability to refactoring was that they utilized many idioms from assembly files, which were easily changed by refactoring tools. Regarding Caliskan-Islam's method in binaries, Caliskan et al. [84] provided some different results, with the conclusion mentioned above in Ref. [177]. They claimed that the accuracy

of their approach decreased from 96% to 88% on 100 authors under obfuscation using Obfuscator-LLVM and showed robustness against obfuscation because "stylistic features are still preserved to a large degree" through basic obfuscation.

Kurtukova et al. [123] employed the HNN model for author attribution. They noted a significant variance in performance degradation with different languages and obfuscators. For example, the accuracy of their method on JavaScript dropped from 76% to 70% under JS obfuscator Tool to 63% under JS-obfuscator. Their accuracy on Python dropped from 92% to 48% under Opy obfuscator to 38% under Pyarmor obfuscator. The accuracy on PHP dropped from 86% to 63% under Yakpro-po obfuscator to 61% under PHP Obfuscator. They concluded that their model was resistant to lexical obfuscation, such as removing whitespace characters, transforming strings, etc. The accuracy dropped 7% under lexical obfuscation. The decrease in the accuracy was as high as 30% under more complex obfuscation, such as the addition of pseudo-complex code, etc.

Zafar et al. [111] utilized CNN-generated representation and KNN as a classifier. To evaluate the robustness of obfuscation, they applied randomized obfuscation tools in their work. Their approach achieved 83.41% accuracy for 14,100 authors under obfuscation, and the average difference for mixed-language scenarios between with and without obfuscation was -7.04%. Abuhamad et al. [110] used RNN-generated representation to work with RFC. This method achieved 95.74% for 1500 programmers of software; the performance dropped to 93.86% under binary code obfuscation using Obfuscator-LLVM. For 8903 non-binary programmers, they achieved 92.3% on accuracy; the accuracy reached 98.90% under Stunnix obfuscator on 120 C++ authors and 93.42% under Tigress obfuscator on 120 C authors.

6. Review on Authorship Verification Methods

This section presents a short review of authorship verification (AV) methods, including their differences with authorship attribution (AA), state-of-the-art works on AV approaches, and common shortcomings.

AA and AV are two important tasks in the field of digital text forensics analysis. AA deals with the problem of identifying the most likely author of a document or text with unknown authorship, given a set of texts of candidate authors, but AV focuses on the question of whether a document or text was in fact written by a known author, where only a set of reference documents or texts of this author is given [178]. Simply, AV tries to answer the question if two documents with unknown authors were written by the same author or not. In AA, a learner attributes a new previously unseen text of unknown authorship to one of these known authors. In contrast, in AV, the learner predicts whether the same author wrote two given texts/documents or not [19,179].

Authorship verification problems are considered a unary classification or a binary classification. An AV method is unary if and only if its decision criterion is determined solely on the basis of the target class, whereas in a binary classification, the decision criterion is determined on a training or testing corpus-labeled data. A binary model can be further categorized into intrinsic or extrinsic [180]. An intrinsic binary classification relies on only the training datasets and does not use any external documents to verify the authorship, whereas extrinsic classification determines their decision criteria, with testing data from external documents that represent the outlier class [178].

In the past two decades, researchers from different disciplines, including linguistics, psychology, computer science, and mathematics, proposed a range of techniques and concepts for this task [181–184]. Stylometric learning is a computational approach for many linguistic tasks, such as authorship verification [185]. State-of-the-art research on AV focuses on stylometric feature learning and/or deep neural network-based approaches.

- Stylometric features-based approaches extract the stylometric features and categorize them into several distinct groups, e.g., lexical features, character features, syntactic features, semantic features, and application-specific features or compression-based features [186,187].

- Deep neural network-based AV approaches integrate the feature extraction task into a deep-learning framework for authorship verification [115,188,189]. Similarity learning and attention-based neural network models were developed to verify authorship for social media messages [190,191]. An *attention-based* Siamese network topology approach learns linguistic features using the bidirectional recurrent neural network. Recently, Hu et al. [189] proposed a topic-based stylometric representation learning for authorship verification.

Based on the short review of AV methods, we summarize the shortcomings of the existing AV work, which are as follows:

- Lack of publicly available appropriate corpora that are required to train and evaluate the models. The metadata with the existing publicly available corpora lack features such as precise time, location, and topic category of the texts.
- There is no more research on how the length of the text message affects the results of the AV model. There is a dramatic shift in the author's characteristics of their writing according to the situation and posting small text messages on social media platforms.
- Limited investigation on the effect of topical influence. In AV problems, the topic of the documents is not always known beforehand, which can lead to a challenge regarding the recognition of the writing style. In addition, the existing AV methods are not robust against topical influence [178].

7. Challenges and Limitations

This section summarizes the challenges and limitations of authorship attribution research through the comprehensive survey conducted in this work.

- Limitations with models: Various techniques have been employed in this field, categorized into stylistic, statistical, language, machine, and deep learning models. Based on our review, these methods face several challenges and limitations as below:
  - Statistical Methods: They do not consider vocabulary or document themes, limiting their effectiveness.
  - Language Models: Selecting optimal 'n' values in n-grams is challenging, as larger values increase representation size, while smaller values lack contextual information.
  - Feature Extraction: Some techniques struggle to capture long-distance dependency features in text.
  - Multiple Authors: Many documents and codes involve multiple authors, impacting accuracy.
  - Data Availability: Finding large and balanced datasets, especially for short texts, is challenging.
  - Language Dependency: Some approaches are language-specific.
  - Metric Selection: Determining useful metrics for models can be complex.
  - Model Scalability: Some models suffer accuracy degradation with increased authors.
  - Generalization: Neural networks may struggle with limited samples.
  - Explainability: Most models lack explanations for results.
- Limitations with datasets: There are several key issues and challenges related to datasets in the authorship attribution research domain. The key challenges and limitations of datasets are as follows:
  - Lack of benchmark and standardization: State-of-the-art research lacks widely accepted large benchmark datasets, hindering standardization and robust evaluation of models.
  - Variability in data sample size per author: There is no consensus on the ideal sample size per author, leading to variability in research approaches.
  - Data validity and imbalance issues: The reliability of small datasets raises concerns about result validity. Even reliable datasets can suffer from imbalance

issues, affecting the representativeness of the data and potentially biasing attribution models.

These limitations underscore the need for more comprehensive and standardized datasets.

- Limitations with feature sets: In this survey, we found the following several limitations and challenges for the feature sets:
  - Limitations with text-based feature sets: These feature sets suffer from lexical oversights, potentially missing structural nuances in the text. The choice of the 'n' value in n-gram features impacts representation size and context, with larger values increasing size but potentially lacking contextual information. Moreover, while content-specific features focus on content, they may overlook the text's essential structural and stylistic elements.
  - Limitations with code-based feature sets: In code-based feature sets, stylometric features are language-dependent and may not generalize effectively across programming languages. The limited availability of keywords and language elements restricts the capabilities of these features. Code-based features are also vulnerable to alterations caused by code formatting, reducing their robustness. Furthermore, extracting stylometric features can be a complex and time-consuming process. Additionally, other code-based features, such as graph-based and dynamic features, face challenges in extraction, language dependence, and additional procedures. Finally, there is a limited exploration of feature categories, and many existing models lack feature-wise explanations for their results.

8. Future Research Directions

We suggest the following future research directions in the authorship attribution research domain:

- Code reuse detection and dynamic author profiling: Authors of both malware and legitimate software reuse software codes and libraries that save time and effort because they have already been written and created [9]. Dynamic author profiling methods can adapt authorship attribution that tracks the changing or evolving coding styles of the author over time. Identifying the reuse of codes, clone detection techniques, and dynamic profiling of the authors can be a potential direction of future work.
- Multi-author attribution: Multi-author attribution focuses on determining who wrote a particular program code or document among multiple authors. In multi-author attribution, it is challenging to divide the code boundaries of different authors in a sample code and identify the code segments belonging to different authors. Investigation into the development of multi-author attribution models for effectively and accurately identifying code segment that belongs to an author among multiple authors can be a potential future work.
- Explainable and interpretable models for authorship attribution: An explainable and interpretable model helps to understand why the model made a particular prediction with the given input variables. Identifying the most influential features or predictor variables and their role in prediction is challenging with linguistic or stylometric features. Design and development of explainable and interpretable methods for authorship attribution can be a potential future research topic.
- Advancement in deep learning models: There are numerous works that leverage deep learning-based models for the classification of authorship attribution. The performance of these models can be further improved by considering the combination of advanced deep neural architectures and loss functions. Investigation into the recent advances in deep learning models (e.g., LSTM, Autoencoder, CNNs, BERT, GPT, etc.) and their possible combinations with different loss functions can be an interesting potential future research topic.
- Against obfuscation or adversarial attack: Authorship obfuscation is a protective countermeasure that aims to evade authorship attribution by obfuscating the writing

style in a text [192]. Most of the existing authorship obfuscation approaches are rule- or learning-based, and these approaches do not consider the adversarial model and cannot defend against adversarial attacks on authorship attribution systems. The development of the obfuscation approaches considering the adversarial setup could be a potential future research direction in authorship attribution.

- Open world problem: An open-world classification deals with scenarios in which the set of classes is not known in advance. In authorship attribution, if a set of potential authors is not known in advance, the model requires making a generalization with the set of known other authors and predicting the authors that have never been seen before. Dynamic authorship modeling and incremental learning, ensemble models, etc., can adapt to the changing authorship styles and characteristics over time. Also, research on authorship attribution methods in blockchain and quantum computing technology can be an interesting potential research direction.
- Dynamic and incremental authorship attribution: Dynamic and incremental authorship attribution methods are used to identify authors in a dynamic situation where they change their writing or coding styles over time. It is a more crucial and challenging task because authorship patterns evolve and work environments change dynamically over time. Adaptive and incremental machine learning models can capture changing authorship patterns by continuously training and updating the models with the new data. Similarly, time series analysis and change-point detection methods can adapt to change authorship patterns that allow for the recognition of new coding or writing styles. The continuous evaluation and validation of the models are crucial to ensure their performance over time.

9. Conclusions

In recent decades, considerable research effort has been dedicated to the field of authorship attribution. This area holds substantial importance in various domains, such as software forensics analysis, plagiarism detection, security attack detection, and safeguarding intellectual property like trade secrets, patents, and copyrights. This survey aims to familiarize new researchers with the current state-of-the-art techniques in authorship attribution. By identifying and scrutinizing emerging methods in this field, the survey delves into their key concepts, challenges, and potential avenues for future exploration. This comprehensive overview serves as a valuable resource for newcomers seeking to navigate the landscape of authorship attribution. In this paper, we conducted a comprehensive survey on authorship attribution methods, their key classifications, feature types, and their classifications, available datasets, evaluation metrics, authorship verification methods, and challenges and limitations considered in the authorship attribution literature. In addition, we discussed the potential future research directions in authorship attribution based on the insights and lessons learned from this survey work.

Author Contributions: X.H.: methodology, investigation, and writing; A.H.L.: conceptualization, methodology, supervision, investigation, writing, reviewing, and editing; N.V.: methodology, investigation, and writing; D.P.S.: conceptualization, methodology, investigation, writing, reviewing, and editing. All authors have read and agreed to the published version of the manuscript.

Funding: This research received the Dean's Award for Research Excellence (DARE) 2023 at York University for the first author and the Mitacs Globalink Research Internship (MGRI) fund in Canada for the internship of the third author.

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data sharing does not apply to this article, as no datasets were generated.

Acknowledgments: The authors acknowledge the Dean's Award for Research Excellence (DARE) 2023 at York University and the Mitacs Globalink Research Internship (MGRI) program in Canada for supporting this research by providing funds.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Al-Sarem, M.; Saeed, F.; Alsaeedi, A.; Boulila, W.; Al-Hadhrami, T. Ensemble Methods for Instance-Based Arabic Language Authorship Attribution. *IEEE Access* **2020**, *8*, 17331–17345. [CrossRef]
2. Mechti, S.; Almansour, F. An Orderly Survey on Author Attribution Methods: From Stylistic Features to Machine Learning Models. *Int. J. Adv. Res. Eng. Technol.* **2021**, *12*, 528–538.
3. Swain, S.; Mishra, G.; Sindhu, C. Recent approaches on authorship attribution techniques—An overview. In Proceedings of the 2017 International conference of Electronics, Communication and Aerospace Technology (ICECA), Coimbatore, India, 20–22 April 2017; IEEE: Piscataway, NJ, USA, 2017; Volume 1, pp. 557–566.
4. Rocha, A.; Scheirer, W.J.; Forstall, C.W.; Cavalcante, T.; Theophilo, A.; Shen, B.; Carvalho, A.R.B.; Stamatatos, E. Authorship Attribution for Social Media Forensics. *IEEE Trans. Inf. Forensics Secur.* **2017**, *12*, 5–33. [CrossRef]
5. Theophilo, A.; Giot, R.; Rocha, A. Authorship Attribution of Social Media Messages. *IEEE Trans. Comput. Soc. Syst.* **2023**, *10*, 10–23. [CrossRef]
6. Spafford, E.H.; Weeber, S.A. Software forensics: Can we track code to its authors? *Comput. Secur.* **1993**, *12*, 585–595. [CrossRef]
7. Bull, J.; Collins, C.; Coughlin, E.; Sharp, D. *Technical Review of Plagiarism Detection Software Report*; Computer Assisted Assessment Centre: Luton, UK, 2001.
8. Culwin, F.; MacLeod, A.; Lancaster, T. *Source Code Plagiarism in UK HE Computing Schools, Issues, Attitudes and Tools*; Technical Report SBU-CISM-01-02; South Bank University: London, UK, 2001.
9. Kalgutkar, V.; Kaur, R.; Gonzalez, H.; Stakhanova, N.; Matyukhina, A. Code authorship attribution: Methods and challenges. *ACM Comput. Surv. CSUR* **2019**, *52*, 1–36. [CrossRef]
10. Li, Z.; Chen, G.Q.; Chen, C.; Zou, Y.; Xu, S. RoPGen: Towards Robust Code Authorship Attribution via Automatic Coding Style Transformation. In Proceedings of the 44th International Conference on Software Engineering, Pittsburgh, PA, USA, 21–29 May 2022; pp. 1906–1918. [CrossRef]
11. Zheng, W.; Jin, M. A review on authorship attribution in text mining. *Wiley Interdiscip. Rev. Comput. Stat.* **2023**, *15*, e1584. [CrossRef]
12. Stamatatos, E. A survey of modern authorship attribution methods. *J. Am. Soc. Inf. Sci. Technol.* **2009**, *60*, 538–556. [CrossRef]
13. Juola, P. Authorship attribution. *Found. Trends Inf. Retr.* **2008**, *1*, 233–334. [CrossRef]
14. Mosteller, F.; Wallace, D.L. *Applied Bayesian and Classical Inference: The Case of The Federalist Papers*; Springer: Berlin/Heidelberg, Germany, 2012.
15. Zheng, R.; Li, J.; Chen, H.; Huang, Z. A framework for authorship identification of online messages: Writing-style features and classification techniques. *J. Am. Soc. Inf. Sci. Technol.* **2006**, *57*, 378–393. [CrossRef]
16. Jin, M.; Jiang, M. Text clustering on authorship attribution based on the features of punctuations usage. In Proceedings of the 2012 IEEE 11th International Conference on Signal Processing, Beijing, China, 21–25 October 2012; IEEE: Piscataway, NJ, USA, 2012; Volume 3, pp. 2175–2178.
17. Stuart, L.M.; Tazhibayeva, S.; Wagoner, A.R.; Taylor, J.M. Style features for authors in two languages. In Proceedings of the 2013 IEEE/WIC/ACM International Joint Conferences on Web Intelligence (WI) and Intelligent Agent Technologies (IAT), Atlanta, GA, USA, 17–20 November 2013; IEEE: Piscataway, NJ, USA, 2013; Volume 1, pp. 459–464.
18. Hinh, R.; Shin, S.; Taylor, J. Using frame semantics in authorship attribution. In Proceedings of the 2016 IEEE International Conference on Systems, Man, and Cybernetics (SMC), Budapest, Hungary, 9–12 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 004093–004098.
19. Ali, N.; Hindi, M.; Yampolskiy, R.V. Evaluation of authorship attribution software on a Chat bot corpus. In Proceedings of the 2011 XXIII International Symposium on Information, Communication and Automation Technologies, Sarajevo, Bosnia and Herzegovina, 27–29 October 2011; IEEE: Piscataway, NJ, USA, 2011; pp. 1–6.
20. Evaluating Variation in Language (EVL) Lab. Java Graphical Authorship Attribution Program Classifiers. 2010. Available online: https://github.com/evllabs/JGAAP/tree/master/src/com/jgaap/classifiers (accessed on 20 December 2023).
21. Goodman, R.; Hahn, M.; Marella, M.; Ojar, C.; Westcott, S. The use of stylometry for email author identification: a feasibility study. In Proceedings of the Student/Faculty Research Day (CSIS) Pace University, White Plains, NY, USA, 4 May 2007; pp. 1–7.
22. Segarra, S.; Eisen, M.; Ribeiro, A. Authorship attribution through function word adjacency networks. *IEEE Trans. Signal Process.* **2015**, *63*, 5464–5478. [CrossRef]
23. Zhao, Y.; Zobel, J.; Vines, P. Using relative entropy for authorship attribution. In Proceedings of the Asia Information Retrieval Symposium, Singapore, 16–18 October 2006; Springer: Berlin/Heidelberg, Germany, 2006; pp. 92–105.
24. Kesidis, G.; Walrand, J. Relative entropy between Markov transition rate matrices. *IEEE Trans. Inf. Theory* **1993**, *39*, 1056–1057. [CrossRef]
25. Khmelev, D.V.; Tweedie, F.J. Using Markov chains for identification of writer. *Lit. Linguist. Comput.* **2001**, *16*, 299–307. [CrossRef]
26. Sanderson, C.; Guenter, S. Short text authorship attribution via sequence kernels, Markov chains and author unmasking: An investigation. In Proceedings of the 2006 Conference on Empirical Methods in Natural Language Processing, Sydney, Australia, 22–23 July 2006; pp. 482–491.

27. Cox, M.A.; Cox, T.F. Multidimensional scaling. In *Handbook of Data Visualization*; Springer: Berlin/Heidelberg, Germany, 2008; pp. 315–347.
28. Abbasi, A.; Chen, H. Writeprints: A stylometric approach to identity-level identification and similarity detection in cyberspace. *ACM Trans. Inf. Syst.* **2008**, *26*, 1–29. [CrossRef]
29. Argamon, S.; Burns, K.; Dubnov, S. *The Structure of Style: Algorithmic Approaches to Understanding Manner and Meaning*; Springer: Berlin/Heidelberg, Germany, 2010.
30. Oman, P.W.; Cook, C.R. A paradigm for programming style research. *ACM Sigplan Not.* **1988**, *23*, 69–78. [CrossRef]
31. Burrows, S. Source Code Authorship Attribution. Ph.D. Thesis, RMIT University, Melbourne, Australia, 2010.
32. Krsul, I.; Spafford, E.H. Authorship analysis: Identifying the author of a program. *Comput. Secur.* **1997**, *16*, 233–257. [CrossRef]
33. Macdonell, S.; Gray, A.; MacLennan, G.; Sallis, P. Software forensics for discriminating between program authors using case-based reasoning, feedforward neural networks and multiple discriminant analysis. In Proceedings of the ICONIP'99 & ANZIIS'99 & ANNES'99 & ACNN'99 6th International Conference on Neural Information Processing, Perth, WA, Australia, 16–20 November 1999; Volume 1, pp. 66–71. [CrossRef]
34. Ding, H.; Samadzadeh, M.H. Extraction of Java program fingerprints for software authorship identification. *J. Syst. Softw.* **2004**, *72*, 49–57. [CrossRef]
35. Lange, R.C.; Mancoridis, S. Using code metric histograms and genetic algorithms to perform author identification for software forensics. In Proceedings of the 9th Annual Conference on Genetic and Evolutionary Computation, London, UK, 7–1 July 2007; pp. 2082–2089.
36. Elenbogen, B.S.; Seliya, N. Detecting outsourced student programming assignments. *J. Comput. Sci. Coll.* **2008**, *23*, 50–57.
37. Agun, H.V.; Yilmazel, O. Document embedding approach for efficient authorship attribution. In Proceedings of the 2007 2nd International Conference on Knowledge Engineering and Applications (ICKEA), London, UK, 21–23 October 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 194–198.
38. Le, Q.; Mikolov, T. Distributed representations of sentences and documents. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 1188–1196.
39. Tamboli, M.S.; Prasad, R.S. Feature selection in time aware authorship attribution. In Proceedings of the 2018 International Conference on Advances in Communication and Computing Technology (ICACCT), Sangamner, India, 8–9 February 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 534–537.
40. Ge, Z.; Sun, Y.; Smith, M. Authorship attribution using a neural network language model. In Proceedings of the AAAI Conference on Artificial Intelligence, Phoenix, AZ, USA, 12–17 February 2016; Volume 30.
41. Pratanwanich, N.; Lio, P. Who wrote this? Textual modeling with authorship attribution in big data. In Proceedings of the 2014 IEEE International Conference on Data Mining Workshop, Shenzhen, China, 14 December 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 645–652.
42. Blei, D.M.; Ng, A.Y.; Jordan, M.I. Latent dirichlet allocation. *J. Mach. Learn. Res.* **2003**, *3*, 993–1022.
43. Seroussi, Y.; Zukerman, I.; Bohnert, F. Authorship attribution with latent Dirichlet allocation. In Proceedings of the Fifteenth Conference on Computational Natural Language Learning, Portland, OR, USA, 23–24 June 2011; pp. 181–189.
44. Breiman, L. Random forests. *Mach. Learn.* **2001**, *45*, 5–32. [CrossRef]
45. McCallum, A.K. Multi-label text classification with a mixture model trained by EM. In Proceedings of the AAAI 99 Workshop on Text Learning, Orlando, FL, USA, 18–19 July 1999.
46. Seroussi, Y.; Zukerman, I.; Bohnert, F. Authorship attribution with topic models. *Comput. Linguist.* **2014**, *40*, 269–310. [CrossRef]
47. Mendenhall, T.C. The characteristic curves of composition. *Science* **1887**, 237–246. [CrossRef] [PubMed]
48. Labbé, C.; Labbé, D. Inter-textual distance and authorship attribution Corneille and Moliére. *J. Quant. Linguist.* **2001**, *8*, 213–231. [CrossRef]
49. Marusenko, M.; Rodionova, E. Mathematical methods for attributing literary works when solving the "Corneille–Molière" problem. *J. Quant. Linguist.* **2010**, *17*, 30–54. [CrossRef]
50. Mosteller, F.; Wallace, D.L. Inference in an authorship problem: A comparative study of discrimination methods applied to the authorship of the disputed Federalist Papers. *J. Am. Stat. Assoc.* **1963**, *58*, 275–309.
51. Mosteller, F.; Wallace, D.L. *Inference and Disputed Authorship: The Federalist*; CSLI: Stanford, CA, USA, 1964.
52. Khomytska, I.; Teslyuk, V. Authorship attribution by differentiation of phonostatistical structures of styles. In Proceedings of the 2018 IEEE 13th International Scientific and Technical Conference on Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 11–14 September 2018; IEEE: Piscataway, NJ, USA, 2018; Volume 2, pp. 5–8.
53. Khomytska, I.; Teslyuk, V. Modelling of phonostatistical structures of English backlingual phoneme group in style system. In Proceedings of the 2017 14th International Conference The Experience of Designing and Application of CAD Systems in Microelectronics (CADSM), Lviv, Ukraine, 21–25 February 2017; IEEE: Piscataway, NJ, USA, 2017; pp. 324–327.
54. Khomytska, I.; Teslyuk, V. Specifics of phonostatistical structure of the scientific style in English style system. In Proceedings of the 2016 XIth International Scientific and Technical Conference Computer Sciences and Information Technologies (CSIT), Lviv, Ukraine, 6–10 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 129–131.
55. Khomytska, I.; Teslyuk, V. The method of statistical analysis of the scientific, colloquial, belles-lettres and newspaper styles on the phonological level. In *Advances in Intelligent Systems and Computing*; Springer: Berlin/Heidelberg, Germany, 2017; pp. 149–163.

56. Khomytska, I.; Teslyuk, V. Authorship and style attribution by statistical methods of style differentiation on the phonological level. In Proceedings of the 2018 Conference on Computer Science and Information Technologies, Lviv, Ukraine, 11–14 February 2018; Springer: Berlin/Heidelberg, Germany, 2018; pp. 105–118.
57. Khomytska, I.; Teslyuk, V.; Holovatyy, A.; Morushko, O. Development of Methods, Models, and Means for the Author Attribution of a Text. *East. Eur. J. Enterp. Technol.* **2018**, *3*, 41–46. [CrossRef]
58. Inches, G.; Harvey, M.; Crestani, F. Finding participants in a chat: Authorship attribution for conversational documents. In Proceedings of the 2013 International Conference on Social Computing, Alexandria, VA, USA, 8–14 September 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 272–279.
59. Burrows, J. 'Delta': A measure of stylistic difference and a guide to likely authorship. *Lit. Linguist. Comput.* **2002**, *17*, 267–287. [CrossRef]
60. Savoy, J. Authorship attribution based on a probabilistic topic model. *Inf. Process. Manag.* **2013**, *49*, 341–354. [CrossRef]
61. Gal, Y.; Ghahramani, Z. Pitfalls in the use of parallel inference for the Dirichlet process. In Proceedings of the International Conference on Machine Learning, Beijing, China, 21–26 June 2014; pp. 208–216.
62. Zhao, Y.; Zobel, J. Searching with style: Authorship attribution in classic literature. In Proceedings of the ACM International Conference Proceeding Series, Ballarat, VIC, Australia, 30 January–2 February 2007; Volume 244, pp. 59–68.
63. Grieve, J. Quantitative authorship attribution: An evaluation of techniques. *Lit. Linguist. Comput.* **2007**, *22*, 251–270. [CrossRef]
64. Gray, A.; Sallis, P.; MacDonell, S. Identified: A dictionary-based system for extracting source code metrics for software forensics. In Proceedings of the Software Engineering: Education and Practice, International Conference on, Dunedin, New Zealand, 26–29 January 1998; IEEE: Piscataway, NJ, USA, 1998; p. 252.
65. Kešelj, V.; Peng, F.; Cercone, N.; Thomas, C. N-gram-based author profiles for authorship attribution. In Proceedings of the Conference Pacific Association for Computational Linguistics (PACLING 2003), Halifax, Canada, 22–25 August 2003; Volume 3, pp. 255–264.
66. Frantzeskou, G.; Stamatatos, E.; Gritzalis, S.; Katsikas, S. Effective identification of source code authors using byte-level information. In Proceedings of the 28th International Conference on Software Engineering, Shanghai, China, 20–28 May 2006; pp. 893–896.
67. Burrows, S.; Uitdenbogerd, A.L.; Turpin, A. Application of Information Retrieval Techniques for Source Code Authorship Attribution. In Proceedings of the Database Systems for Advanced Applications, Brisbane, Australia, 21–23 April 2009; Springer: Berlin/Heidelberg, Germany, 2009; pp. 699–713.
68. Burrows, S.; Uitdenbogerd, A.L.; Turpin, A. Temporally Robust Software Features for Authorship Attribution. In Proceedings of the 2009 33rd Annual IEEE International Computer Software and Applications Conference, Seattle, WA, USA, 20–24 July 2009; Volume 1, pp. 599–606. [CrossRef]
69. Burrows, S.; Tahaghoghi, S.M. Source code authorship attribution using n-grams. In Proceedings of the Twelth Australasian Document Computing Symposium, Melbournem, Australia, 10 December 2007; pp. 32–39.
70. Holmes, G.; Donkin, A.; Witten, I.H. Weka: A machine learning workbench. In Proceedings of the ANZIIS'94-Australian New Zealnd Intelligent Information Systems Conference, Brisbane, QLD, Australia, 29 November–2 December 1994; IEEE: Piscataway, NJ, USA, 1994; pp. 357–361.
71. Witten, I.H.; Frank, E.; Hall, M.A.; Pal, C.J. Practical machine learning tools and techniques. In *Data Mining*; Elsevier: Amsterdam, The Netherlands, 2005; Volume 2.
72. Kothari, J.; Shevertalov, M.; Stehle, E.; Mancoridis, S. A Probabilistic Approach to Source Code Authorship Identification. In Proceedings of the 4th International Conference on Information Technology (ITNG'07), Las Vegas, NV, USA, 2–4 April 2007; pp. 243–248. [CrossRef]
73. Rosenblum, N.; Zhu, X.; Miller, B.; Hunt, K. Machine learning-assisted binary code analysis. In Proceedings of the NIPS Workshop on Machine Learning in Adversarial Environments for Computer Security, Whistler, BC, Canada, 3–4 December 2007.
74. Kindermann, R.; Snell, J. *Contemporary Mathematics: Markov Random Fields and their Applications*; American Mathematical Society: Providence, RI, USA, 1980.
75. Shevertalov, M.; Kothari, J.; Stehle, E.; Mancoridis, S. On the Use of Discretized Source Code Metrics for Author Identification. In Proceedings of the 2009 1st International Symposium on Search Based Software Engineering, Windsor, UK, 13–15 May 2009; pp. 69–78. [CrossRef]
76. Rosenblum, N.; Zhu, X.; Miller, B.P. Who wrote this code? identifying the authors of program binaries. In Proceedings of the Computer Security—ESORICS 2011, Leuven, Belgium, 12–14 September 2011; Springer: Berlin/Heidelberg, Germany, 2011; pp. 172–189.
77. Layton, R.; Azab, A. Authorship analysis of the Zeus botnet source code. In Proceedings of the 2014 5th Cybercrime and Trustworthy Computing Conference, Auckland, New Zealand, 24–25 November 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 38–43.
78. Fred, A.; Jain, A.K. Evidence accumulation clustering based on the k-means algorithm. In Proceedings of the Joint IAPR International Workshops on Statistical Techniques in Pattern Recognition (SPR) and Structural and Syntactic Pattern Recognition (SSPR), Windsor, ON, Canada, 6–9 August 2002; Springer: Berlin/Heidelberg, Germany, 2002; pp. 442–451.
79. Layton, R.; Watters, P.; Dazeley, R. Automated unsupervised authorship analysis using evidence accumulation clustering. *Nat. Lang. Eng.* **2013**, *19*, 95–120. [CrossRef]

80. Alazab, M.; Layton, R.; Broadhurst, R.; Bouhours, B. Malicious spam emails developments and authorship attribution. In Proceedings of the 2013 4th Cybercrime and Trustworthy Computing Workshop, Sydney, NSW, Australia, 21–22 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 58–68.
81. Layton, R.; Watters, P.; Dazeley, R. Recentred local profiles for authorship attribution. *Nat. Lang. Eng.* **2012**, *18*, 293–312. [CrossRef]
82. Layton, R.; Perez, C.; Birregah, B.; Watters, P.; Lemercier, M. Indirect information linkage for OSINT through authorship analysis of aliases. In Proceedings of the Pacific-Asia Conference on Knowledge Discovery and Data Mining, Golden Coast, QLD, Australia, 14–17 April 2013; Springer: Berlin/Heidelberg, Germany, 2013; pp. 36–46.
83. Caliskan-Islam, A.; Harang, R.; Liu, A.; Narayanan, A.; Voss, C.; Yamaguchi, F.; Greenstadt, R. De-anonymizing programmers via code stylometry. In Proceedings of the 24th USENIX security symposium (USENIX Security 15), Washington, DC, USA, 12–14 August 2015; pp. 255–270.
84. Caliskan, A.; Yamaguchi, F.; Dauber, E.; Harang, R.; Rieck, K.; Greenstadt, R.; Narayanan, A. When Coding Style Survives Compilation: De-anonymizing Programmers from Executable Binaries. In Proceedings of the 2018 Network and Distributed System Security Symposium, San Diego, CA, USA, 18–21 February 2018. [CrossRef]
85. Meng, X. Fine-grained binary code authorship identification. In Proceedings of the 2016 24th ACM SIGSOFT International Symposium on Foundations of Software Engineering, Seattle, WA, USA, 13–18 November 2016; pp. 1097–1099.
86. Cortes, C.; Vapnik, V. Support vector machine. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
87. Meng, X.; Miller, B.P.; Williams, W.R.; Bernat, A.R. Mining software repositories for accurate authorship. In Proceedings of the 2013 IEEE International Conference on Software Maintenance, Eindhoven, The Netherlands, 22–28 September 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 250–259.
88. Meng, X.; Miller, B.P.; Jun, K.S. Identifying multiple authors in a binary program. In Proceedings of the European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; Springer: Berlin/Heidelberg, Germany, 2017; pp. 286–304.
89. Zhang, C.; Wang, S.; Wu, J.; Niu, Z. Authorship Identification of Source Codes. In Proceedings of the Asia-Pacific Web (APWeb) and Web-Age Information Management (WAIM) Joint Conference on Web and Big Data, Beijing, China, 7–9 July 2017; Springer International: Cham, Switzerland, 2017; pp. 282–296.
90. Dauber, E.; Caliskan, A.; Harang, R.; Greenstadt, R. Poster: Git blame who?: Stylistic authorship attribution of small, incomplete source code fragments. In Proceedings of the 2018 IEEE/ACM 40th International Conference on Software Engineering: Companion (ICSE-Companion), Gothenburg, Sweden, 27 May–3 June 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 356–357.
91. Zhang, S.; Li, X.; Zong, M.; Zhu, X.; Cheng, D. Learning k for knn classification. *ACM Trans. Intell. Syst. Technol.* **2017**, *8*, 1–19. [CrossRef]
92. Ewais, A.; Samara, D.A. Adaptive MOOCs based on intended learning outcomes using naive bayesian technique. *Int. J. Emerg. Technol. Learn.* **2020**, *15*, 4–21. [CrossRef]
93. Dai, T.; Dong, Y. Introduction of SVM related theory and its application research. In Proceedings of the 2020 3rd International Conference on Advanced Electronic Materials, Computers and Software Engineering (AEMCSE), Shenzhen, China, 24–26 April 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 230–233.
94. Sapkota, U.; Solorio, T.; Montes-y-Gómez, M.; Ramírez-de-la-Rosa, G. Author Profiling for English and Spanish Text. In Proceedings of the Working Notes for CLEF 2013 Conference, Valencia, Spain, 23–26 September 2013.
95. Das, M.; Ghosh, S.K. Standard Bayesian network models for spatial time series prediction. In *Enhanced Bayesian Network Models for Spatial Time Series Prediction*; Springer: Berlin/Heidelberg, Germany, 2020; pp. 11–22.
96. Zheng, Q.; Li, L.; Chen, H.; Loeb, S. What aspects of principal leadership are most highly correlated with school outcomes in China? *Educ. Adm. Q.* **2017**, *53*, 409–447. [CrossRef]
97. Argamon, S.; Whitelaw, C.; Chase, P.; Hota, S.R.; Garg, N.; Levitan, S. Stylistic text classification using functional lexical features. *J. Am. Soc. Inf. Sci. Technol.* **2007**, *58*, 802–822. [CrossRef]
98. Alkaabi, M.; Olatunji, S.O. Modeling Cyber-Attribution Using Machine Learning Techniques. In Proceedings of the 2020 30th International Conference on Computer Theory and Applications (ICCTA), Alexandria, Egypt, 12–14 December 2020; IEEE: Piscataway, NJ, USA, 2020; pp. 10–15.
99. Li, J.; Zheng, R.; Chen, H. From fingerprint to writeprint. *Commun. ACM* **2006**, *49*, 76–82. [CrossRef]
100. Pillay, S.R.; Solorio, T. Authorship attribution of web forum posts. In Proceedings of the 2010 eCrime Researchers Summit, Dallas, TX, USA, 18–20 October 2010; IEEE: Piscataway, NJ, USA, 2010; pp. 1–7.
101. Donais, J.A.; Frost, R.A.; Peelar, S.M.; Roddy, R.A. A system for the automated author attribution of text and instant messages. In Proceedings of the 2013 IEEE/ACM International Conference on Advances in Social Networks Analysis and Mining, Niagara, ON, Canada, 25–28 August 2013; pp. 1484–1485.
102. Khonji, M.; Iraqi, Y.; Jones, A. An evaluation of authorship attribution using random forests. In Proceedings of the 2015 International Conference on Information and Communication Technology Research (ICTRC), Abu Dhabi, United Arab Emirates, 17–19 May 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 68–71.
103. Pacheco, M.L.; Fernandes, K.; Porco, A. Random Forest with Increased Generalization: A Universal Background Approach for Authorship Verification. In Proceedings of the CLEF Working Notes 2015, Toulouse, France, 8–11 September 2015.
104. Pinho, A.J.; Pratas, D.; Ferreira, P.J. Authorship attribution using relative compression. In Proceedings of the 2016 Data Compression Conference (DCC), Snowbird, UT, USA, 30 March 2016–1 April 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 329–338.

105. Bengio, Y.; Courville, A.; Vincent, P. Representation learning: A review and new perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef]
106. Abuhamad, M.; AbuHmed, T.; Mohaisen, A.; Nyang, D. Large-scale and language-oblivious code authorship identification. In Proceedings of the 2018 ACM SIGSAC Conference on Computer and Communications Security, Toronto, ON, Canada, 15–19 October 2018; pp. 101–114.
107. Shin, E.C.R.; Song, D.; Moazzezi, R. Recognizing functions in binaries with neural networks. In Proceedings of the 24th USENIX security symposium (USENIX Security 15), Washington, DC, USA, 12–15 August 2015; pp. 611–626.
108. Srivastava, N.; Hinton, G.; Krizhevsky, A.; Sutskever, I.; Salakhutdinov, R. Dropout: a simple way to prevent neural networks from overfitting. *J. Mach. Learn. Res.* **2014**, *15*, 1929–1958.
109. Kohavi, R. A study of cross-validation and bootstrap for accuracy estimation and model selection. In Proceedings of the International Joint Conference on Artificial Intelligence, Montreal, QB, Canada, 20–25 August 1995; Volume 14, pp. 1137–1145.
110. Abuhamad, M.; Abuhmed, T.; Mohaisen, D.; Nyang, D. Large-Scale and Robust Code Authorship Identification with Deep Feature Learning. *ACM Trans. Priv. Secur.* **2021**, *24*, 23. [CrossRef]
111. Zafar, S.; Sarwar, M.U.; Salem, S.; Malik, M.Z. Language and Obfuscation Oblivious Source Code Authorship Attribution. *IEEE Access* **2020**, *8*, 197581–197596. [CrossRef]
112. White, R.; Sprague, N. Deep Metric Learning for Code Authorship Attribution and Verification. In Proceedings of the 2021 20th IEEE International Conference on Machine Learning and Applications (ICMLA), Pasadena, CA, USA, 13–16 December 2021; pp. 1089–1093. [CrossRef]
113. Bogdanova, A. Source Code Authorship Attribution Using File Embeddings. In Proceedings of the 2021 ACM SIGPLAN International Conference on Systems, Programming, Languages, and Applications: Software for Humanity, Chicago, IL, USA, 17–22 October 2021; pp. 31–33. [CrossRef]
114. Bogdanova, A.; Romanov, V. Explainable source code authorship attribution algorithm. *J. Phys. Conf. Ser.* **2021**, *2134*, 012011. [CrossRef]
115. Bagnall, D. Author identification using multi-headed recurrent neural networks. *arXiv* **2015**, arXiv:1506.04891.
116. Ruder, S.; Ghaffari, P.; Breslin, J.G. Character-level and multi-channel convolutional neural networks for large-scale authorship attribution. *arXiv* **2016**, arXiv:1609.06686.
117. Yavanoglu, O. Intelligent authorship identification with using Turkish newspapers metadata. In Proceedings of the 2016 IEEE International Conference on Big Data (Big Data), Washington, DC, USA, 5–8 December 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 1895–1900.
118. Shrestha, P.; Sierra, S.; González, F.A.; Montes-y Gómez, M.; Rosso, P.; Solorio, T. 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, Valencia, Spain, 3–7 April 2017; pp. 669–674.
119. Zhao, C.; Song, W.; Liu, X.; Liu, L.; Zhao, X. Research on Authorship Attribution of Article Fragments via RNNs. In Proceedings of the 2018 IEEE 9th International Conference on Software Engineering and Service Science (ICSESS), Beijing, China, 23–25 November 2018; IEEE: Piscataway, NJ, USA, 2018; pp. 156–159.
120. Yang, X.; Xu, G.; Li, Q.; Guo, Y.; Zhang, M. Authorship attribution of source code by using back propagation neural network based on particle swarm optimization. *PLoS ONE* **2017**, *12*, e0187204. [CrossRef]
121. Abuhamad, M.; su Rhim, J.; AbuHmed, T.; Ullah, S.; Kang, S.; Nyang, D. Code authorship identification using convolutional neural networks. *Future Gener. Comput. Syst.* **2019**, *95*, 104–115. [CrossRef]
122. Ullah, F.; Wang, J.; Jabbar, S.; Al-Turjman, F.; Alazab, M. Source Code Authorship Attribution Using Hybrid Approach of Program Dependence Graph and Deep Learning Model. *IEEE Access* **2019**, *7*, 141987–141999. [CrossRef]
123. Kurtukova, A.; Romanov, A.; Shelupanov, A. Source Code Authorship Identification Using Deep Neural Networks. *Symmetry* **2020**, *12*, 2044. [CrossRef]
124. Bogomolov, E.; Kovalenko, V.; Rebryk, Y.; Bacchelli, A.; Bryksin, T. Authorship Attribution of Source Code: A Language-Agnostic Approach and Applicability in Software Engineering. In Proceedings of the 29th ACM Joint Meeting on European Software Engineering Conference and Symposium on the Foundations of Software Engineering, Athens, Greece, 23–28 August 2021; pp. 932–944. [CrossRef]
125. Burns, K. Bayesian inference in disputed authorship: A case study of cognitive errors and a new system for decision support. *Inf. Sci.* **2006**, *176*, 1570–1589. [CrossRef]
126. Argamon, S.; Levitan, S. Measuring the usefulness of function words for authorship attribution. In Proceedings of the Joint Conference of the Association for Computers and the Humanities and the Association for Literary and Linguistic Computing, Victoria, BC, Canada, 15–18 June 2005; pp. 1–3.
127. Zhao, Y.; Zobel, J. Effective and scalable authorship attribution using function words. In Proceedings of the Asia Information Retrieval Symposium, Jeju Island, Republic of Korea, 13–15 October 2005; Springer: Berlin/Heidelberg, Germany, 2005; pp. 174–189.
128. Yu, B. Function words for Chinese authorship attribution. In Proceedings of the NAACL-HLT 2012 Workshop on Computational Linguistics for Literature, Montreal, Canada, 8 June 2012; pp. 45–53.
129. Kestemont, M. Function words in authorship attribution. From black magic to theory? In Proceedings of the 3rd Workshop on Computational Linguistics for Literature (CLFL), Gothenburg, Sweden, 27 April 2014; pp. 59–66.

130. Koppel, M.; Schler, J.; Argamon, S. Computational methods in authorship attribution. *J. Am. Soc. Inf. Sci. Technol.* **2009**, *60*, 9–26. [CrossRef]
131. Yule, G.U. On sentence-length as a statistical characteristic of style in prose: With application to two cases of disputed authorship. *Biometrika* **1939**, *30*, 363–390.
132. Ahmed, A.F.; Mohamed, R.; Mostafa, B.; Mohammed, A.S. Authorship attribution in Arabic poetry. In Proceedings of the 2015 10th International Conference On Intelligent Systems: Theories and Applications (SITA), Rabat, Morocco, 20–21 October 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–6.
133. Holmes, D.I. The evolution of stylometry in humanities scholarship. *Lit. Linguist. Comput.* **1998**, *13*, 111–117. [CrossRef]
134. Can, F.; Patton, J.M. Change of writing style with time. *Comput. Humanit.* **2004**, *38*, 61–82. [CrossRef]
135. Ramezani, R.; Sheydaei, N.; Kahani, M. Evaluating the effects of textual features on authorship attribution accuracy. In Proceedings of the ICCKE 2013, Mashhad, Iran, 31 October 2013–1 November 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 108–113.
136. Wanner, L. Authorship Attribution Using Syntactic Dependencies. In *Artificial Intelligence Research and Development*; IOS Press: Amsterdam, The Netherlands, 2016; pp. 303–308.
137. Varela, P.; Justino, E.; Britto, A.; Bortolozzi, F. A computational approach for authorship attribution of literary texts using syntactic features. In Proceedings of the 2016 International Joint Conference on Neural Networks (IJCNN), Vancouver, BC, Canada, 24–29 July 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 4835–4842.
138. Varela, P.J.; Justino, E.J.R.; Bortolozzi, F.; Oliveira, L.E.S. A computational approach based on syntactic levels of language in authorship attribution. *IEEE Lat. Am. Trans.* **2016**, *14*, 259–266. [CrossRef]
139. Wu, H.; Zhang, Z.; Wu, Q. Exploring syntactic and semantic features for authorship attribution. *Appl. Soft Comput.* **2021**, *111*, 107815. [CrossRef]
140. Sidorov, G.; Velasquez, F.; Stamatatos, E.; Gelbukh, A.; Chanona-Hernández, L. Syntactic n-grams as machine learning features for natural language processing. *Expert Syst. Appl.* **2014**, *41*, 853–860. [CrossRef]
141. Cutting, D.; Kupiec, J.; Pedersen, J.; Sibun, P. A practical part-of-speech tagger. In Proceedings of the 3rd Conference on Applied Natural Language Processing, Trento, Italy, 31 March–3 April 1992; pp. 133–140.
142. Solorio, T.; Pillay, S.; Montes-y Gómez, M. Authorship identification with modality specific meta features. In Proceedings of the CLEF 2011, Amsterdam, The Netherlands, 17–20 September 2011.
143. Baayen, R. *Analyzing Linguistic Data: A Practical Introduction to Statistics Using R*; Cambridge University Press: Cambridge, UK, 2008; Volume 4, pp. 32779–32790.
144. Kanade, V. What Is Semantic Analysis? Definition, Examples, and Applications in 2022. Available online: https://www.spiceworks.com/tech/artificial-intelligence/articles/what-is-semantic-analysis/ (accessed on 10 December 2023).
145. McCarthy, P.M.; Lewis, G.A.; Dufty, D.F.; McNamara, D.S. Analyzing Writing Styles with Coh-Metrix. In Proceedings of the Flairs Conference, Melbourne Beach, FL, USA, 11–13 May 2006; pp. 764–769.
146. Miller, G.A. WordNet: a lexical database for English. *Commun. ACM* **1995**, *38*, 39–41. [CrossRef]
147. Yule, G.U. *The Statistical Study of Literary Vocabulary*; Cambridge University Press: Cambridge, UK, 1944; Volume 42, pp. b1–b2.
148. Holmes, D.I. Vocabulary richness and the prophetic voice. *Lit. Linguist. Comput.* **1991**, *6*, 259–268. [CrossRef]
149. Tweedie, F.J.; Baayen, R.H. How variable may a constant be? Measures of lexical richness in perspective. *Comput. Humanit.* **1998**, *32*, 323–352. [CrossRef]
150. Koppel, M.; Akiva, N.; Dagan, I. Feature instability as a criterion for selecting potential style markers. *J. Am. Soc. Inf. Sci. Technol.* **2006**, *57*, 1519–1525. [CrossRef]
151. Cheng, N.; Chandramouli, R.; Subbalakshmi, K. Author gender identification from text. *Digit. Investig.* **2011**, *8*, 78–88. [CrossRef]
152. Ragel, R.; Herath, P.; Senanayake, U. Authorship detection of SMS messages using unigrams. In Proceedings of the 2013 IEEE 8th International Conference on Industrial and Information Systems, Peradeniya, Sri Lanka, 17–20 December 2013; IEEE: Piscataway, NJ, USA, 2013; pp. 387–392.
153. Laroum, S.; Béchet, N.; Hamza, H.; Roche, M. Classification automatique de documents bruités à faible contenu textuel. *Rev. Des Nouv. Technol. Inf.* **2010**, *18*, 25.
154. Ouamour, S.; Sayoud, H. Authorship attribution of ancient texts written by ten arabic travelers using a smo-svm classifier. In Proceedings of the 2012 International Conference on Communications and Information Technology (ICCIT), Hammamet, Tunisia, 26–28 June 2012; IEEE: Piscataway, NJ, USA, 2012; pp. 44–47.
155. Spitters, M.; Klaver, F.; Koot, G.; Van Staalduinen, M. Authorship analysis on dark marketplace forums. In Proceedings of the 2015 European Intelligence and Security Informatics Conference, Manchester, UK, 7–9 September 2015; IEEE: Piscataway, NJ, USA, 2015; pp. 1–8.
156. Vazirian, S.; Zahedi, M. A modified language modeling method for authorship attribution. In Proceedings of the 2016 Eighth International Conference On Information and Knowledge Technology (IKT), Hammamet, Tunisia, 7–8 September 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 32–37.
157. Escalante, H.J.; Solorio, T.; Montes, M. Local histograms of character n-grams for authorship attribution. In Proceedings of the 49th Annual Meeting of The Association for Computational Linguistics: Human Language Technologies, Portland, OR, USA, 19–24 June 2011; pp. 288–298.

158. Martindale, C.; McKenzie, D. On the utility of content analysis in author attribution: The Federalist. *Comput. Humanit.* **1995**, *29*, 259–270. [CrossRef]
159. Marinho, V.Q.; Hirst, G.; Amancio, D.R. Authorship attribution via network motifs identification. In Proceedings of the 2016 5th Brazilian conference on intelligent systems (BRACIS), Recife, Brazil, 9–12 October 2016; IEEE: Piscataway, NJ, USA, 2016; pp. 355–360.
160. Bayrami, P.; Rice, J.E. Code authorship attribution using content-based and non-content-based features. In Proceedings of the 2021 IEEE Canadian Conference on Electrical and Computer Engineering (CCECE), Virtual, 12–17 September 2021; IEEE: Piscataway, NJ, USA, 2021; pp. 1–6.
161. Oman, P.W.; Cook, C.R. Programming style authorship analysis. In Proceedings of the 17th Conference on ACM Annual Computer Science Conference, Louisville, KY, USA, 21–23 February 1989; pp. 320–326.
162. Oman, P.W.; Cook, C.R. A taxonomy for programming style. In Proceedings of the 1990 ACM Annual Conference on Cooperation, Washington, DC, USA, 20–22 February 1990; pp. 244–250.
163. Sallis, P.; Aakjaer, A.; MacDonell, S. Software forensics: old methods for a new science. In Proceedings of the 1996 International Conference Software Engineering: Education and Practice, Dunedin, New Zealand, 24–27 January 1996; IEEE: Piscataway, NJ, USA, 1996; pp. 481–485.
164. Tennyson, M.F.; Mitropoulos, F.J. Choosing a profile length in the SCAP method of source code authorship attribution. In Proceedings of the IEEE SOUTHEASTCON 2014, Lexington, KY, USA, 13–16 March 2014; IEEE: Piscataway, NJ, USA, 2014; pp. 1–6.
165. Pellin, B.N. *Using Classification Techniques to Determine Source Code Authorship*; White Paper; Department of Computer Science, University of Wisconsin: Madison, WI, USA, 2000.
166. Alsulami, B.; Dauber, E.; Harang, R.; Mancoridis, S.; Greenstadt, R. Source code authorship attribution using long short-term memory based networks. In Proceedings of the Computer Security ESORICS 2017—22nd European Symposium on Research in Computer Security, Oslo, Norway, 11–15 September 2017; pp. 65–82.
167. Alrabaee, S.; Saleem, N.; Preda, S.; Wang, L.; Debbabi, M. Oba2: An onion approach to binary code authorship attribution. *Digit. Investig.* **2014**, *11*, S94–S103. [CrossRef]
168. Ferrante, A.; Medvet, E.; Mercaldo, F.; Milosevic, J.; Visaggio, C.A. Spotting the Malicious Moment: Characterizing Malware Behavior Using Dynamic Features. In Proceedings of the 2016 11th International Conference on Availability, Reliability and Security (ARES), Salzburg, Austria, 31 August–2 September 2016; pp. 372–381. [CrossRef]
169. Wang, N.; Ji, S.; Wang, T. Integration of Static and Dynamic Code Stylometry Analysis for Programmer De-Anonymization. In Proceedings of the 11th ACM Workshop on Artificial Intelligence and Security. Association for Computing Machinery, Toronto, Canada, 15–19 October 2018; pp. 74–84.
170. Frantzeskou, G.; MacDonell, S.; Stamatatos, E.; Gritzalis, S. Examining the significance of high-level programming features in source code author classification. *J. Syst. Softw.* **2008**, *81*, 447–460. [CrossRef]
171. Wisse, W.; Veenman, C. Scripting DNA: Identifying the JavaScript programmer. *Digit. Investig.* **2015**, *15*, 61–71. [CrossRef]
172. Arp, D.; Spreitzenbarth, M.; Hubner, M.; Gascon, H.; Rieck, K. DREBIN: Effective and Explainable Detection of Android Malware in Your Pocket. In Proceedings of the Network and Distributed System Security Symposium 2014, San Diego, CA, USA, 23–26 February 2014.
173. Melis, M.; Maiorca, D.; Biggio, B.; Giacinto, G.; Roli, F. Explaining Black-box Android Malware Detection. *arXiv* **2018**, arXiv:1803.03544. http://arxiv.org/abs/1803.03544.
174. Ribeiro, M.T.; Singh, S.; Guestrin, C. "Why Should I Trust You?": Explaining the Predictions of Any Classifier. *arXiv* **2016**, arXiv:1602.04938. http://arxiv.org/abs/1602.04938.
175. Murenin, I.; Novikova, E.; Ushakov, R.; Kholod, I. Explaining Android Application Authorship Attribution Based on Source Code Analysis. In Proceedings of the Internet of Things, Smart Spaces, and Next Generation Networks and Systems: 20th International Conference, NEW2AN 2020, and 13th Conference, RuSMART 2020, St. Petersburg, Russia, 26–28 August 2020; Springer: Berlin/Heidelberg, Germany, 2020; pp. 43–56.
176. Abuhamad, M.; Abuhmed, T.; Nyang, D.; Mohaisen, D. Multi-χ: Identifying Multiple Authors from Source Code Files. *Proc. Priv. Enhanc. Technol.* **2020**, *2020*, 25–41. [CrossRef]
177. Alrabaee, S.; Shirani, P.; Debbabi, M.; Wang, L. On the Feasibility of Malware Authorship Attribution. In Proceedings of the 9th International Symposium FPS 2016, Quebec City, QC, Canada, 24–25 October 2016; pp. 256–272. [CrossRef]
178. Halvani, O.; Winter, C.; Graner, L. Assessing the applicability of authorship verification methods. In Proceedings of the 14th International Conference on Availability, Reliability and Security, Canterbury, UK, 26–29 August 2019; pp. 1–10.
179. Tyo, J.; Dhingra, B.; Lipton, Z.C. On the state of the art in authorship attribution and authorship verification. *arXiv* **2022**, arXiv:2209.06869.
180. Potha, N.; Stamatatos, E. Intrinsic author verification using topic modeling. In Proceedings of the 10th Hellenic Conference on Artificial Intelligence, Patras, Greece, 9–12 July 2018; pp. 1–7.
181. Koppel, M.; Schler, J. Authorship verification as a one-class classification problem. In Proceedings of the 21st International Conference on Machine Learning, Alberta, Canada, 4–8 July 2004; p. 62.
182. Koppel, M.; Winter, Y. Determining if two documents are written by the same author. *J. Assoc. Inf. Sci. Technol.* **2014**, *65*, 178–187. [CrossRef]

183. Ding, S.H.; Fung, B.C.; Iqbal, F.; Cheung, W.K. Learning stylometric representations for authorship analysis. *IEEE Trans. Cybern.* **2017**, *49*, 107–121. [CrossRef] [PubMed]
184. Halvani, O.; Winter, C.; Graner, L. Unary and binary classification approaches and their implications for authorship verification. *arXiv* **2018**, arXiv:1901.00399.
185. Luyckx, K.; Daelemans, W. Authorship attribution and verification with many authors and limited data. In Proceedings of the 22nd International Conference on Computational Linguistics (COLING 2008), Manchester, UK, 18–22 August 2008; pp. 513–520.
186. Veenman, C.J.; Li, Z. Authorship Verification with Compression Features. In Proceedings of the Working Notes for CLEF 2013 Conference, Valencia, Spain, 23–26 September 2013.
187. Hernández-Castañeda, Á.; Calvo, H. Author verification using a semantic space model. *Comput. Sist.* **2017**, *21*, 167–179. [CrossRef]
188. Litvak, M. Deep dive into authorship verification of email messages with convolutional neural network. In Proceedings of the Information Management and Big Data: 5th International Conference, SIMBig 2018, Lima, Peru, 3–5 September 2018; Springer: Berlin/Heidelberg, Germany, 2019; pp. 129–136.
189. Hu, X.; Ou, W.; Acharya, S.; Ding, S.H.; Ryan, D.G. TDRLM: Stylometric learning for authorship verification by Topic-Debiasing. *Expert Syst. Appl.* **2023**, *233*, 120745. [CrossRef]
190. Boenninghoff, B.; Nickel, R.M.; Zeiler, S.; Kolossa, D. Similarity Learning for Authorship Verification in Social Media. In Proceedings of the ICASSP 2019—2019 IEEE International Conference on Acoustics, Speech and Signal Processing (ICASSP), Brighton, UK, 12–17 May 2019; pp. 2457–2461. [CrossRef]
191. Boenninghoff, B.; Hessler, S.; Kolossa, D.; Nickel, R.M. Explainable Authorship Verification in Social Media via Attention-based Similarity Learning. In Proceedings of the 2019 IEEE International Conference on Big Data (Big Data), Los Angeles, CA, USA, 9–12 September 2019; pp. 36–45. [CrossRef]
192. Zhai, W.; Rusert, J.; Shafiq, Z.; Srinivasan, P. Adversarial Authorship Attribution for Deobfuscation. In Proceedings of the 60th Annual Meeting of the Association for Computational Linguistics 2022, Dublin, Ireland, 22–27 May 2022; pp. 7372–7384.