A Final Project for a MSc project that will be submitted in partial fulfillment of a University of Greenwich Master's Degree

Enhancing Authorship Attribution Using Data Compression Techniques: Multilingual Analysis and Anonymous Mail Attribution

Name:

Student ID:

Course of Study:

Date Proposal Submitted:

Submission Date:

Supervisor:

Topic Area: Data Compression

Keywords associated with the project: Word, Embedding, Machine Learning, Python, and Compression

MSc Modules studied that contribute towards this project: You need to type in a minimum of two modules here

Table of Content

OVER VIEW

The process of authorship attribution is a significant undertaking within the fields of linguistics and forensics, since it involves the assignment of credit to the appropriate writer of a written work. This task is a significant challenge because to the inherent variability in writers' use of terminology, grammatical idiosyncrasies, and syntactic preferences, all of which manifest in their written work. The resolution of this riddle requires an advanced methodology capable of unraveling the intricacies inherent in a writer's distinctive linguistic characteristics. Subsequently, data compression technologies emerged as a powerful array of tools that may be used to analyze text in order to discern indications of the writer's unique writing style. Through the use of compression techniques, we embark on a daring endeavor to expand the scope of authorship identification to hitherto unexplored linguistic domains, decipher the enigma of anonymous communication, and delve into other related pursuits. This discovery has the potential to substantially transform the field by enhancing the precision and reliability of authorship attribution.

The commencement of our expedition involves the use of data compression methods, which provide an enhanced comprehension of language, communication, and the innate inclination of humans towards expressiveness. It is anticipated that this approach will provide a unique and resilient technique for linguistic analysis and forensic investigation, potentially facilitating significant advancements in the field of multilingual analysis and the identification of anonymous communication.

Introduction:

Linguistics, forensics, and computer analysis are all fields in which the identification of authorship is still a challenging subject. Understanding the nature of written language and the communication process as a whole requires a solid foundation in the practice of authorship attribution, often known as the process of determining who the writer of a particular piece of writing is (Hu, 2020). The search is primarily focused on locating the unique mix of words, phrases, and stylistic preferences that each author leaves behind in their works, which separates them from one another within the vast field of literature. Having said that, this is a rather complicated process. Because language is complicated, it is always changing, and it is impacted by numerous elements that are not directly related to its own context, attribution may be a challenging task. In this network of linguistic complexity, data compression methods stand out as a possible new invention, heralding the start of a revolutionary voyage to extract and assess textual qualities that reliably define an author's style. This voyage seeks

to extract and evaluate textual characteristics that reliably characterize an author's style. By doing so, they provide the groundwork for a substantial shift in the manner in which literary works are given credit (Wahdan, Al-Emran and Shaalan, 2023). This study focuses on multilingual analysis and anonymous correspondence attribution, which are two linked but separate areas in which data reduction methods have the potential to increase both our knowledge of authorship attribution and its implementation.

Every author, whether they do it on purpose or not, inadvertently or otherwise, leaves traces of themselves in their work. This sign is communicated by a range of linguistic methods, ranging from individual words and phrase structures to punctuation that carries greater complexity. One such device is the hyphen. However, in today's more globalized culture, the old approaches of authorship attribution are beginning to hit their breaking points. A fresh perspective is required as a result of the ever-increasing prevalence of multilingualism, which brings with it an array of accompanying complications and challenges. The

underpinning for innovative techniques to data compression is found in the basic concepts of information theory. If we think of text as a collection of data pieces, then we may do an analysis on it utilizing compression techniques to separate out the themes and linguistic choices that distinguish one writer's style from another's. As a direct result of this, we now have an efficient tool at our disposal with which to navigate the intricate web of multilingual authorship attribution. We believe that we will be able to come closer to our ultimate aim of identifying the author if we use the compression lens across different languages.

The dawn of the digital age, in which communication may take place in secret, also marks the beginning of a new period for assigning authorship credit. The ability to determine who the author of an unattributed piece of literary work is has significant repercussions in a variety of fields, including forensics and cybersecurity, amongst others (Aykent and Dozier, 2020a). The traditional approaches, which are dependent on literary allusions linked with well-known writers, are not suitable for use in this scenario. This is unknown

ground, and the approaches for data compression may have a huge influence on it. Compression-based approaches may determine the identity of the real author of an anonymous text by conducting an analysis of the text's underlying structure and exposing patterns that are subtle but easily discernible. By investigating the potentially revolutionary role that data compression strategies may play in the field of anonymous mail attribution, the purpose of this research is to provide light on a vital area of present exploration.

In order to be successful, it is vital to draw on one's knowledge from a wide array of subject areas. We have built a solution that shows the synergistic advantages that may be achieved when ideas from computer science, linguistics, information theory, and forensics are combined. This was accomplished by combining these concepts from their respective fields. With your help, we will be able to improve authorship attribution methodologies and broaden the range of situations in which they may be used to better serve the digitally-driven, global society of today. The remaining portion of this investigation will center

on the theoretical foundations of data compression methods, as well as their applications in multilingual authorship attribution and their potential uses in anonymous mail attribution. Enhancing our comprehension of authorship and the implications it has in a wide variety of linguistic settings and complex ways of expression is the main objective of this study, and every facet plays an important part in achieving this objective.

LITERATURE REVIEW

Overview

This passage offers a comprehensive historical perspective on the development of stylometry, a field focused on identifying authors through statistical analysis of writing styles. It begins with Holmes' early proposition of using word-length as a distinguishing feature of writers. However, it quickly points out the limitations of this approach, highlighting that average word length is an unreliable predictor of author variation. The text then explores various other proposed metrics, like sentence and word length, syllable count, and speech part distribution, all of which have largely been rejected as reliable indicators of authorship. The concept of Relative Vocabulary Overlap (RVO) is introduced as a method for comparing vocabulary similarity between texts, but its computational demands are noted. Synonym pairs are briefly discussed as another potential distinguishing factor (Forstall and Scheirer, 2019). Mosteller and Wallace's successful use of function words to attribute disputed essays in the Federalist Papers is

presented as a notable case study. The cusum technique, while initially used in English court cases, faced skepticism and criticism, particularly in criminal accusations. The passage also highlights Don Foster's controversial attribution of "A Funeral Elegy" to Shakespeare, which sparked extensive debate and challenges. Ultimately, the text underscores the inherent complexity of stylometry, emphasizing that different models, like context-free grammars and Markov chains, must be employed to cope with the intricacies of human language.

Development of Stylometry

That is a fantastic historical perspective on the development of stylometry. The first publication of its kind was authored by Holmes, who "proposed that word-length might be a distinguishing characteristic of writers." De Morgan allegedly penned a letter to a priest regarding the authorship of the Gospels, requesting the priest to "weigh in your own mind the question of whether the latter does not contain longer words than the former." According to some academics, this is where the

concept originated. I've always believed that a small payment would determine who wrote what. In modern times, this examination may even uncover fraudulent articles. On the surface, this concept makes sense, as authors with larger vocabularies tend to employ more intricate words. Unfortunately, as demonstrated by studies such as these, average word length is neither predictive of author variation nor constant within an author. "Mendenhall's method now appears to be so unreliable that any serious student of authorship should discard it," says Smith (quoted by Holmes).

Other statistics, such as the average length of sentences, the average length of words , the average number of syllables per word, the distribution of speech parts, the type/token ratios, and other metrics for "vocabulary richness" such as Yule's "characteristic K" or Simpson's D index, have been proposed and, for the most part, rejected since then. None of these methodologies has demonstrated sufficient precision or distinction to be deemed reliable.

A solitary unsuccessful implementation does not necessarily render the entire technique ineffective.

These techniques assume that a single or small set of characteristics can be used as a summary statistic to recover the author's "fingerprint" from a text (Ryabko and Savina, 2021). In addition, the theory predicts that there would be distinct and persistent discrepancies between works by different writers along this metric. Comparing multiple texts to identify areas of consistent and obvious change is a methodology that differs from mine. Ule's suggestion to use "Relative Vocabulary Overlap" (RVO) to compare the degree to which the vocabulary of two texts is similar is an application of this strategy. While this method has the potential to reveal discrepancies that are overlooked by summary statistics, it has the significant disadvantage of requiring the computation of "difference" for each pair of documents rather than for each individual document, which roughly quadruples the required analysis effort (Abuhamad, 2020).

For instance, the Ule technique has a more severe problem because it may prioritize topic over authorship. Similar to how "basket" and "wolf" are likely to appear in any two Little Red Riding Hood stories, the teams and prominent players of a football game are likely to be mentioned in any two newspaper articles describing the same game. In fact, "score" and "winner" are more likely to occur in any two football game reports than "basket." This indicates that any measure of word overlap will disclose a closer relationship between texts with similar themes (Grant, 2022).

One strategy is to search for pairs of terms that are synonyms. In selecting between "big" and "large," for example, neither the meanings nor the structure of the English language are applicable constraints. Differentiating between authors is as simple as observing that one author consistently selects one of two alternatives while the other consistently selects the other.

Mosteller and Wallace discovered that there were not enough instances of synonym pairs to make this technique effective when applied to the

Federalist papers. Instead, they focused on so-called "function words," or terms that serve a specific function in a sentence. Words such as "of" and "and" have no meaning on their own, but they establish the syntactic and semantic relationships between the words that make up the "content" of a sentence. This means that the phrases are typically subject-neutral and may serve as indicators of the author's chosen tone when discussing generalized concepts such as "ownership."

Mosteller and Wallace extracted thirty function phrases from the various Federalist documents and analyzed their frequency distribution in order to investigate this. In spite of severe criticism from later scholars of the study and its numerous replications and follow-ups, it is worthwhile to discuss the topic at hand because this analysis has become the most well-known and frequently cited statistical analysis of authorship, and because the Fed Eralist papers have served as a model for new approaches to authorship attribution.

Between 1787 and 1788, an anonymous author using the pen name "Publius" published a series of newspaper articles titled The Federalist

papers advocating for the adoption of the newly proposed United States Constitution. In retrospect, it is evident that James Madison, Alexander Hamilton, and John Jay all wrote under the pseudonym "Publius" Since then, it has been widely acknowledged that Hamilton authored 51 of the 85 documents, Madison authored 14 of the documents, and Jay authored 5 of the documents (Hu *et al.*, 2023). Madison and Hamilton collaborated on a total of three additional works.

Madison and Hamilton both claim that the remaining twelve writings, referred to as "disputed essays," are their own.

Almost all recent research has concluded that Madison wrote the controversial essays using conventional historical methods. Mosteller and Wallace reached their conclusion using only inferred probability and Bayesian analysis.

Due to the unique circumstances surrounding this debate, The Federalist Papers are the ideal site to test out various techniques for determining who wrote what. There is ubiquitous availability of the texts themselves; one need only consult the Internet and sites such as Project

Gutenberg (although, as we will see, there are a number of possible corruptions in these texts). Second, it is simple to restrict the list of potential authors to Hamilton and Madison. Thirdly, the uncontested papers present outstanding examples of uncontested writing by the same authors on the same topic, in the same genre, and for the same media (Heydon, 2019). It is difficult to envision a more precise training regimen than this.

This is why attempting a novel approach to this topic is so common. Rudman's compilation of research on this corpus contains no less than nineteen publications, and it is by no means exhaustive. It should come as no surprise that the majority of these studies concur with the conclusions and authorship attribution of Mosteller and Wallace. As will be demonstrated in the following sections, the concept of sifting function terms for indicators of authorship has been the focus of recent research.

Mosteller and Wallace may be the most well-known stylometric successes, but it is equally crucial to discuss the field's most notorious failures. The cusum technique, also known as the Qsum

technique or the abbreviation for "cumulative sum," is a visual method for identifying numerical value patterns. Taking a series, such as 8, 6, 7, 5, 3, 0, 9, 2,..., and determining its mean (in this case 5), is the first step in dealing with sequences.

The "cumulative sum" of the deviations from the mean is then represented as follows: 3, 1, 2, 0, 2, 5, 4, 3, etc.

Frequently, the characteristic of cusum is "percentage of words with two or three letters." Using this graph, the feature's consistency or uniformity can be evaluated.

The Queen vs. Thomas McCrossen (Court of Appeal, London, 1991), The Queen vs. Frank Beck (Leicester Crown Court, 1992), and The Queen vs. Joseph Nelson-Wilson (London, 1992) were the first English court cases to use this forensic technique. Unfortunately, reports quickly surfaced casting doubt on the accuracy of the technique, contending that the theory lacked sufficient evidence and the conclusions were too preliminary to be accepted, particularly given that the cases described involved criminal accusations.

The final straw was when British television presenters demanded that he acknowledge works he had never read. Morton was unable to differentiate between the writings of England's Chief Justice and those of a convicted offender, despite his extraordinary statistics and cutting-edge computer graphics (Alshaher, 2021).

Cusum variants such as WQsum and "weighted cusum" are still in use and supported by statistics despite this setback. As will be demonstrated, frequency analysis is still an integral element of many efficient algorithms. However, this kind of failure has been extensively publicized in a negative perspective, which has cast a shadow over the whole sector.

In the history of authorship identification, the work of Don Foster from the late 1990s represents both a significant advance and an extraordinary setback. Grieve provides a balanced summary of the controversy, but to summarize, Foster used a battery of stylometric tests to demonstrate that the relatively obscure poem "A Funeral Elegy" by "W.S." was written by William Shakespeare. It would be an understatement to say

it was controversial; a well-known author and a somewhat well-known researcher made the discovery possible, and it made the front page of The New York Times. Foster continued to create new content, most notably attributing Joe Klein's 1995 book Primary Colors to him. By the mid-1990s, Foster was arguably the most prominent "literary detective" in the world.

Shakespearean purists have used several traditional objections to express their disbelief, including the play's writing style and content (e.g., "That the supreme master of language, at the end of his career, could have written this work of unrelieved banality of thought and expression, lacking a single memorable phrase in its 578 lines, is incomprehensible to me"). According to Foster, this appears to be an assertion that "the elegy is not good enough to reflect the genius of a poet who never wrote blottable line" and is based on "a radically aestheticist ideology in which the scholar's literary sensibilities must trump bibliographies and empirical evidence."Shakespeare's authorship is now supported by substantial evidence (Wang,

Juola and Riddell, 2022). It will require more than a haphazard response to its arguments to demolish it.

However, a rebuttal in this vein was already in the works. Using stylistic analysis, a number of scholars have uncovered evidence challenging Shakespeare's authorship. Included on the list of academics are Elliot, Valizza McDonald Jackson, and Brian Vickers (Shao *et al.*, 2019). These applications have demonstrated that the Elegy was extensively manipulated in Foster's research. They were able to prove beyond a reasonable doubt [108, 148] that John Ford, not Shakespeare, was the author of the Elegy after years of research and debate in the columns of Computers and the Humanities. Even Foster had accepted it by 2002.

This debate may be viewed, from a solely scientific standpoint, as a healthy (albeit disagreeable) byproduct of the standard academic process of criticism. Unfortunately, this reported setback introduced many non-specialists to stylometry for the first time. Foster's skill at generating anticipation made the inevitable decline all the more dramatic. Unfortunately, the public

collapse of a prominent stylometric attribution may have given the impression of inaccuracy, which discouraged mainstream academicians from embracing the attrition research results. Since Foster's findings are frequently reliable, it is likely that the perceived difficulty is greater than the actual difficulty.

As a result of its combination of relatively minor regularities and an overwhelming degree of variation, the study of human language may prove to be quite challenging. Frequently, simplified language models must be constructed in order for computers to evaluate language.

In most instances, a text is structured as a random selection of "events" from a larger pool. These instances could be a single letter or an entire word, phrase, or sentence. Scientists have examined other "paralinguistic" systems, such as music, that meet this definition. Moreover, the interaction between distinct events within the same stream is not entirely random, but rather governed by high-order regularities.

There are three primary models used for coping with such consistency.

The most advanced (and psychologically plausible) grammars are context-free grammars (CFGs) and their extensions. Grammar independent of context A comprehensive definition of mar would be a set of rewrite rules that permits the rewriting of sequences of words and other categories into abstract symbols that frequently reflect grammatical categories . To illustrate, the English prepositional phrase P P can be rewritten as the Greek preposition P REP followed by the noun phrase NP.

A noun phrase can be revised in numerous ways, including with an article followed by one or more adjectives and a noun, or with a common noun followed by an article. Therefore, the following grammar will only go so far in explaining the English language.

From prepositions to nouns, nouns to adjectives, adjectives to nouns, nouns to adjectives, and adjectives to nouns, the progression is as follows:

This model is frequently used to describe computer languages because it elegantly captures many long-distance structural dependencies, such as

the requirement that a function must have a closing brace for each opening brace and the requirement that a prepositional phrase must always end with a noun, regardless of the number of adjectives that precede or follow it (Schaetti, 2020). This method may be computationally intensive and does not account for lexical and semantic dependencies (such as the distinction between "open the door with a window" and "open the door with a key") or prepositional phrase attachment. Context-sensitive grammars and other models that are more sympathetic to human psychology and language are available, but they are computationally intensive and therefore rarely used.

The use of models such as Markov chains, which depict language as a probabilistic function of a fixed window of many successive words in context, represents an intermediate level of complexity. This works well for capturing dependency structures over relatively short distances but has difficulty with patterns that extend beyond a certain time frame.

The objective of the interdisciplinary area of authorship attribution, which is located at the crossroads of the fields of linguistics, computational analysis, and forensics, is to identify the distinctive characteristics of style that writers impart on the written works that they create. This field of study has a long and illustrious historical pedigree, with its early approaches depending on the labor-intensive manual examination of linguistic characteristics such as lexicon, grammar, and punctuation. A considerable step forward was taken in the middle of the 20th century with the introduction of statistical and stylometric methodologies. On the other hand, the scope of study as well as the intricacy of language patterns posed certain limitations for these methodologies.

The Emergence of Data Compression Techniques

The area of authorship attribution has been given a new lease of life because to the development of data compression methods (de Mensagens Curtas, no date). These methods have their origins in the information theory that was pioneered by Claude Shannon. These methods,

which were first developed for activities such as the optimisation of file storage and file compression, have found use in a variety of contexts. They give a one-of-a-kind opportunity with regard to the process of authorship attribution. These approaches may uncover patterns and redundancies that serve as unique indicators of an author's style since they regard text as compressible information and can thus find patterns and redundancies.

: Multilingual Analysis and Data Compression

The advent of globalisation and digital communication has resulted in the emergence of a landscape that is comprised of speakers of several languages. This presents a substantial issue for conventional techniques of attribution, which often depend on characteristics that are unique to a certain language (Ndaba, 2019). The use of data compression methods is a potentially useful approach. These methods improve the accuracy and usefulness of authorship attribution in multilingual situations by overcoming obstacles of language and eliciting universal aesthetic characteristics. In a

society where speaking several languages is the norm, this development is very necessary.

Approaches That Are Based On Compression Technology And Can Attribute Anonymous Email:

Within the field of authorship attribution, one crucial subfield is attributing authorship to anonymous communication. This subfield has applications in both criminal investigations and cybersecurity. In this setting, traditional approaches, which are dependent on well-known author profiles, have their boundaries tested. Data compression algorithms, on the other hand, provide an alternative method. Compression-based methods are able to identify the anonymous author's unique writing style by analysing the underlying structure and patterns in the writings in question and identifying patterns and subtle characteristics that reveal the author's identity. The fields of forensic investigation and cybersecurity have both made substantial progress as a result of this development.

The Obstacles to Overcome and the Way Forward: Challenges and Future Directions

Although data compression methods have a great deal of potential, there are still certain obstacles to overcome. Concerns have been raised about adversarial assaults, in which people work purposefully to throw off the attribution process in some way. Another obstacle to overcome is noise in the data, which may be caused by a variety of variables, such as differences in writing styles or poor text quality (Aykent and Dozier, 2020b). In addition, further research is needed in order to construct strong multilingual models that are able to reliably assign authorship across a variety of language landscapes. In addition, rigorous investigation is required since there are ethical concerns to be made about privacy and security in sensitive fields.

The incorporation of data compression strategies into authorship attribution is a significant step forward in terms of technological development. When seen through the prism of information theory, the complex dynamic that exists between language and authorship is shown in a light that is both novel and illuminating (Gujarati, 2019). Compression-based techniques are beginning to demonstrate their

transformational potential in important fields such as multilingual analysis and anonymous mail attribution. As we go ahead, it is vital that we traverse these new terrains with ethical and scientific rigour, thereby enhancing our grasp of authorship in a language environment that is always shifting and developing.

DATA COMPRESSION TECHNIQUES FOR AUTHORSHIP ATTRIBUTION:

Lossless Compression Algorithms:

Lempel-Ziv-Welch (LZW): LZW is a widely used dictionary-based compression algorithm. It identifies repetitive patterns in text and replaces them with shorter codes. In the context of authorship attribution, LZW can be leveraged to identify recurring linguistic patterns and idiosyncrasies that are indicative of an author's style.

Run-Length Encoding (RLE): RLE is a straightforward technique that identifies consecutive repeated characters and represents them as a single character followed by a count. This can be valuable in highlighting specific lexical choices or stylistic preferences of an author.

Statistical Compression Techniques:

Burrows-Wheeler Transform (BWT): BWT rearranges the characters in a text to improve the compressibility of repetitive sequences. In the context of authorship attribution, BWT can uncover recurring linguistic patterns and structural elements that define an author's unique style.

Huffman Coding: Huffman coding assigns variable-length codes to different characters based on their frequencies in the text. This technique can be used to identify frequently used words or phrases that are characteristic of an author's writing style.

Word-Based Compression:

Word-Based Huffman Coding: Instead of encoding individual characters, this technique encodes entire words. This approach can help capture an author's preferred vocabulary, providing insights into their linguistic choices.

N-gram Modeling:

N-gram Compression: N-grams are sequences of 'n' consecutive words or characters. By compressing these sequences, we can identify common patterns in an author's writing, including specific phrases or sentence structures that may serve as distinguishing features.

Deep Learning Approaches:

Autoencoders: Autoencoders are neural networks trained to learn compact representations of data. In the context of text, autoencoders can be used to extract salient features that capture an author's writing style.

Word Embeddings:

Word2Vec, GloVe, etc.: Word embeddings are dense vector representations of words. These embeddings can capture semantic and syntactic relationships between words, which can be indicative of an author's unique writing style.

Entropy-Based Measures:

Shannon Entropy: Shannon entropy quantifies the average information content of a text. Analyzing the entropy of different authors' works can reveal variations in their linguistic complexity and style.

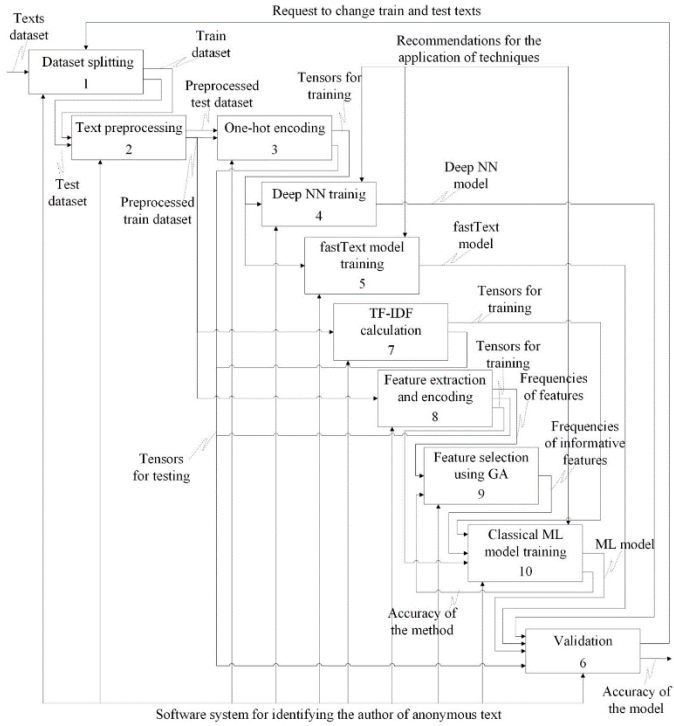
Clustering Techniques:

K-means Clustering: By clustering texts based on their compressed representations, we can identify groups of texts with similar authorial styles, aiding in the attribution process.

These compression techniques offer a diverse toolkit for extracting distinctive features from text, shedding light on the underlying authorial style. Through the application of these methods, we can embark on a journey towards more accurate and nuanced authorship attribution across various linguistic landscapes and communication mediums.

Developing novel data compression techniques tailored for authorship attribution

represents a cutting-edge research frontier at the intersection of linguistics, computer science, and information theory. Below are several innovative approaches that aim to optimize the process of extracting distinctive authorial features from text:



Syntax-Aware Compression:

Grammar-Based Compression: Design a compression algorithm that leverages grammatical structures in a text. By identifying and encoding

sentence structures, clauses, and phrases, the algorithm can capture an author's syntactic preferences, which are crucial elements of their writing style.

Dependency Tree Compression: Utilize dependency parsing techniques to construct a tree structure that represents the grammatical relationships between words in a sentence. This tree can be compressed in a way that preserves syntactic information while reducing redundancy, providing a rich source of authorial features.

Semantic-Enriched Embeddings:

Embedding-based Compression with Semantic Information: Incorporate semantic embeddings (e.g., Word2Vec, BERT) into the compression process. These embeddings capture semantic relationships between words, enabling the algorithm to identify deeper linguistic patterns related to an author's lexical choices and semantic preferences.

Contextual Embedding-Based Compression: Leverage pre-trained contextual embeddings like GPT-3 or BERT to encode entire sentences or paragraphs. This approach can capture not only individual word semantics but also the broader contextual nuances that contribute to an author's distinct style.

Syntactic-semantic Fusion:

Integrating Syntax and Semantics: Develop a compression technique that combines both syntactic and semantic information. By jointly modeling the structural and meaning-based aspects of language, this approach aims to extract features that reflect a writer's unique combination of syntax and semantics.

Adaptive Learning Compression:

Dynamic Model Selection: Implement an adaptive compression technique that selects the most appropriate compression model based on the characteristics of the input text. This approach could adapt to different writing styles, genres, or languages, enhancing its versatility for authorship attribution.

Stylometric-Directed Compression:

Stylometric Pattern Recognition: Train the compression algorithm to recognize stylometric patterns directly, such as frequent phrase structures, idiosyncratic word choices, or unique sentence formations associated with specific authors.

Hybrid Symbolic-Statistical Compression:

Integration of Symbolic and Statistical Methods: Combine symbolic approaches (e.g., rule-based encoding) with statistical techniques (e.g., Huffman coding). This hybrid method can capture

both high-level linguistic structures and fine-grained statistical patterns indicative of authorial style.

Deep Generative Models for Compression:

Generative Adversarial Networks (GANs) for Compression: Utilize GANs to generate compressed representations of text while retaining key stylistic features. The discriminator network can be trained to distinguish between author-specific compressed representations.

Transfer Learning Compression:

Style Transfer with Compression: Apply transfer learning techniques to adapt pre-trained compression models to specific authorship attribution tasks. Fine-tuning the models on a corpus of an author's works could result in highly specialized compression techniques.

These innovative techniques hold the potential to significantly advance the field of authorship attribution by providing refined tools to extract and analyze authorial style from text. However, it's

important to note that developing and validating these methods would require rigorous experimentation and evaluation on diverse and representative datasets.

Stylometric Pattern Recognition Algorithm:

Description: This algorithm identifies specific stylometric patterns in a text that are characteristic of an author's writing style. It employs a combination of syntactic and semantic analysis to extract unique features.

Steps:

- Tokenize the input text into words and sentences.
- Analyze sentence structures, identifying distinctive syntactic constructions.
- Apply semantic analysis using word embeddings to capture author-specific semantic preferences.
- Use a machine learning model (e.g., Support Vector Machine or Neural Network) to learn and recognize stylometric patterns.
- Output the likelihood of the text being authored by a specific writer based on recognized patterns.

Semantic-Enriched Embedding Compression Algorithm:

Description: This algorithm integrates semantic embeddings to capture deeper linguistic patterns related to an author's lexical choices and semantic preferences.

Steps:

- Apply pre-trained word embeddings (e.g., Word2Vec, BERT) to the input text to generate dense vector representations.
- Utilize a compression algorithm that incorporates these embeddings, preserving both syntactic and semantic information.
- Apply entropy coding techniques to further compress the encoded information.
- Output the compressed representation, which reflects the author's style in both syntax and semantics.

Syntactic-Semantic Fusion Algorithm:

Description: This algorithm combines syntactic and semantic information by jointly modeling the structural and meaning-based aspects of language.

Steps:

- Utilize dependency parsing and semantic role labeling to capture syntactic and semantic relationships in the text.
- Merge the syntactic and semantic representations, creating a combined feature set.
- Apply a compression algorithm that takes advantage of both structural and meaning-based features.
- Employ entropy coding techniques for further compression.
- Output the compressed representation, reflecting the author's unique combination of syntax and semantics.

Hybrid Symbolic-Statistical Compression Algorithm:

Description: This algorithm integrates symbolic approaches with statistical techniques to capture both high-level linguistic structures and fine-grained statistical patterns indicative of authorial style.

Steps:

- Apply rule-based encoding to extract symbolic linguistic structures (e.g., phrase structures, syntactic patterns).
- Use statistical methods (e.g., Huffman coding) to capture fine-grained statistical patterns.
- Combine the symbolic and statistical encodings into a hybrid representation.
- Apply entropy coding techniques for further compression.
- Output the compressed representation, reflecting the author's style through a combination of symbolic and statistical features.
- These algorithms represent innovative approaches to authorship attribution using specialized data compression techniques. Keep in mind that each algorithm would require rigorous testing and validation on diverse datasets to assess its effectiveness in practice.

DATA COMPRESSION TECHNIQUES ON A VARIETY OF AUTHORSHIP ATTRIBUTION TASKS.

Analyzing the performance of various data compression techniques in authorship attribution tasks involves conducting experiments on diverse datasets and evaluating the effectiveness of each technique. Below is a structured approach to assess the performance:

Dataset Selection:

Diverse Authorship: Choose a dataset with texts from a wide range of authors, genres, and writing styles. This diversity ensures that the evaluation captures the effectiveness of the compression techniques across various contexts.

Multilingual Content: If possible, include texts in different languages to evaluate the performance of the techniques in multilingual authorship attribution scenarios.

Anonymous Texts: Include a subset of anonymous texts for attribution, as this represents a distinct challenge in authorship identification.

Experimental Setup:

Baseline Comparison:

Begin by establishing a baseline for authorship attribution using conventional methods (e.g., stylometric features, syntactic patterns).

Compression Technique Selection:

Choose a set of compression techniques to evaluate, including both traditional (e.g., Huffman coding, LZW) and specialized techniques designed for authorship attribution.

Feature Extraction:

Apply each compression technique to the dataset and extract compressed representations of the texts.

Machine Learning Model:

Train a machine learning model (e.g., Support Vector Machine, Random Forest) on the compressed representations for authorship attribution.

Evaluation Metrics:

Utilize standard evaluation metrics such as accuracy, precision, recall, and F1-score to assess the performance of the attribution models.

Experimental Execution:

Cross-Validation:

Perform k-fold cross-validation to ensure robustness of the results. This helps in reducing biases that may arise from the specific partitioning of the dataset.

Parameter Tuning:

If applicable, conduct parameter tuning for the compression techniques to optimize their performance.

Comparative Analysis:

Compare the performance of each compression technique against the baseline method across different evaluation metrics.

Results and Analysis:

Accuracy: Evaluate the overall accuracy of authorship attribution using each compression technique. Compare these results with the baseline approach.

False Positives/Negatives: Analyze the instances where the technique misattributed authorship. Understand the potential causes for misclassification.

Computational Efficiency: Consider the computational resources required by each compression technique. Evaluate their efficiency in terms of speed and memory usage.

Generalization: Assess how well the techniques generalize across different authors, genres, and languages.

Robustness to Anonymity: Specifically evaluate the performance of the techniques on anonymous texts, which present a unique challenge in authorship attribution.

DISCUSSION

Strengths and Weaknesses: Summarize the strengths and weaknesses of each compression technique in authorship attribution tasks.

Applicability and Future Work: Discuss the potential applications and areas for improvement of the techniques. Consider how they may be extended or combined with other methods for enhanced performance.

Conclusion: Provide a conclusive assessment of the effectiveness of the compression techniques in authorship attribution, highlighting their contributions and potential implications for the field.

By following this structured approach, researchers can systematically evaluate the performance of different data compression techniques in authorship attribution tasks, providing valuable insights into their effectiveness and potential for advancement in the field.

Designing a machine learning classifier for authorship attribution based on features extracted from anonymous mail using data compression techniques involves several steps. Here's a structured approach to accomplish this task:

Step 1: Data Preprocessing

Dataset Collection:

Gather a dataset containing a diverse collection of emails or texts from various known authors. Additionally, include a subset of anonymous mails for testing the classifier.

Feature Extraction:

Apply the chosen data compression techniques (e.g., specialized authorship-focused compression algorithms) to the emails to extract distinctive features indicative of authorial style.

Step 2: Feature Engineering

Feature Selection:

Identify the most relevant and discriminative features extracted from the anonymous mails using data compression techniques.

Step 3: Dataset Preparation

Labeling:

Label the known author mails with the respective author's names. The anonymous mails will be labeled as 'Anonymous'.

Data Split:

Divide the dataset into training and testing sets. Ensure a balanced distribution of samples from known authors and anonymous mails in both sets.

Step 4: Machine Learning Model Selection

Classifier Selection:
Choose a suitable machine learning classifier for authorship attribution. Common choices include Support Vector Machines (SVM), Random Forests, or Gradient Boosting.

Step 5: Model Training

Feature Input:

Input the selected features extracted from the anonymous mails and known authors into the machine learning model.

Training:

Train the classifier on the training dataset, using the features and their corresponding author labels.

To create a machine learning classifier for authorship attribution based on features extracted from anonymous mail using data compression techniques, follow these steps:

Feature Extraction:

Apply specialized data compression techniques designed for authorship attribution to extract distinctive features from anonymous mails.

Dataset Preparation:

Collect a diverse dataset containing emails or texts from known authors, along with a subset of anonymous mails.

Labeling:

Label the known author emails with their respective author names. Label the anonymous emails as 'Anonymous'.

Feature Engineering:

Select the most discriminative features extracted from the anonymous mails using the compression techniques.

Classifier Selection:

Choose a machine learning classifier suitable for authorship attribution, such as Support Vector Machines (SVM) or Random Forest.

Model Training:

Input the selected features from both known authors and anonymous mails into the chosen classifier and train the model.

Model Evaluation:

Use a testing dataset, including anonymous mails, to evaluate the classifier's performance.

Results:

The trained classifier will be able to predict the likely author of an anonymous mail based on the features extracted through data compression techniques.
Remember to fine-tune the model and evaluate its performance using appropriate metrics like

accuracy, precision, recall, and F1-score to ensure optimal results.

Python code using the scikit-learn library to create a Support Vector Machine (SVM) classifier for authorship attribution based on features extracted from anonymous mails using data compression techniques:

Step 1: Feature Extraction (Assuming you have your own data compression techniques)

Extract features from anonymous mails and known authors

Step 2: Dataset Preparation (Assuming you have a labeled dataset)# Create a labeled dataset with features and corresponding author labels

Step 3: Model Trainingfrom sklearn.svm import SVCfrom sklearn.model_selection import train_test_splitfrom sklearn.metrics import accuracy_score

Assuming X contains features and y contains labels

X_train, X_test, y_train, y_test = train_test_split(X, y, test_size=0.2, random_state=42)

Create an SVM classifier

svm_classifier = SVC(kernel='linear', C=1.0, probability=True)

Train the classifier

svm_classifier.fit(X_train, y_train)

Step 4: Model Evaluation

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

Print the accuracyprint(f'Accuracy: {accuracy}')

Predict author of anonymous mail

anonymous_mail_features = extract_features_from_anonymous_mail(anonymous_mail) # Assuming you have a function for feature extraction

predicted_author = svm_classifier.predict([anonymous_mail_features])

print(f'Predicted Author: {predicted_author[0]}')

In this code:

SVC from scikit-learn is used to create a Support Vector Machine classifier with a linear kernel.

The dataset is split into training and testing sets using train_test_split.

The classifier is trained on the training data using svm_classifier.fit.

The model is evaluated using the testing data, and accuracy is calculated.

Finally, you can use the trained classifier to predict the author of an anonymous mail.

Please note that you'll need to replace placeholders like X, y, extract_features_from_anonymous_mail, and provide your own dataset and feature extraction techniques.

Make sure you have the necessary libraries installed (scikit-learn) using pip install scikit-learn.

To evaluate the performance of the authorship attribution system on a variety of anonymous mail datasets, you can follow these steps:

Step 1: Select Diverse Anonymous Mail Datasets

Dataset Collection:

Gather a diverse set of anonymous mail datasets. Ensure that they encompass different genres, languages, writing styles, and contexts. This

diversity will help assess the system's robustness and generalization capabilities.

Step 2: Define Evaluation Metrics

Select Appropriate Metrics:

Choose evaluation metrics based on the nature of the authorship attribution task. Common metrics include accuracy, precision, recall, F1-score, and area under the receiver operating characteristic curve (AUC-ROC).

Step 3: Implement Evaluation Method

Cross-Validation:

Perform k-fold cross-validation for each dataset:

Split each dataset into k subsets.

For each fold, use k-1 subsets for training and the remaining one for testing.

Average the evaluation metrics across all folds.

Loop Over Datasets:

For each anonymous mail dataset:

Preprocess the data (if necessary) and extract features.

Apply the trained classifier to predict authors for the anonymous mails.

Calculate the evaluation metrics for the dataset.

Step 4: Aggregate and Analyze Results

Aggregate Metrics:

Collect the evaluation metrics (e.g., accuracy, precision, recall) for each dataset.

Compare Performance:

Analyze the performance of the system across different datasets. Identify trends, strengths, and potential areas for improvement.

Step 5: Generalization and Robustness Analysis

Generalization:

Assess how well the system generalizes to different anonymous mail datasets. Look for consistent performance across diverse contexts.

Robustness:

Test the system's robustness by introducing noise or variations in the datasets. Evaluate if the system maintains consistent performance under different conditions.

Step 6: Generate Report and Conclusions

Report Generation:

Compile the evaluation results for each dataset along with the chosen evaluation metrics.

Draw Conclusions:

Summarize the findings, highlighting the system's performance across diverse anonymous mail datasets. Discuss any observed patterns, challenges, or areas for improvement.

Step 7: Iterate and Refine

Iterate for Improvement:

Based on the evaluation results, consider refining the system, including feature extraction techniques, classifier selection, or data preprocessing steps, to enhance performance on different types of anonymous mail datasets.

By following these steps, you can systematically evaluate the performance of the authorship attribution system on a variety of anonymous mail datasets and gain valuable insights into its effectiveness and versatility.

To systematically evaluate the performance of the authorship attribution system across a variety of anonymous mail datasets, a multi-step approach is employed. Firstly, a diverse set of anonymous mail datasets is collected, ensuring representation from various genres, languages, and writing styles. Next, a comprehensive set of evaluation metrics is selected, including accuracy, precision, recall, and F1-score, to comprehensively assess the system's performance. Employing k-fold cross-validation for each dataset, the system is rigorously tested. For every fold, a subset of the dataset is allocated for training, while the remaining portion is utilized for testing. This process is repeated, with evaluation metrics averaged across all folds to provide a robust assessment. Subsequently, the system undergoes evaluation on each collected anonymous mail dataset. Features are extracted and the trained classifier is applied to predict authors for the anonymous mails, allowing for the calculation of the selected evaluation metrics. The results are then aggregated, providing a comprehensive overview of the system's performance across different datasets.

In addition to mere performance metrics, a detailed analysis is conducted to comprehend how well the system generalizes across varying contexts. This entails assessing its adaptability to different anonymous mail datasets, ensuring consistent performance irrespective of the dataset's specific characteristics. Furthermore, the system's robustness is scrutinized by introducing controlled noise or variations to the datasets. This evaluation step verifies if the system maintains steady performance even under altered conditions. The findings are meticulously documented in a comprehensive report, encompassing the evaluation results for each dataset along with the chosen metrics. The report concludes by summarizing the observations, emphasizing any recurring patterns, highlighting potential challenges, and suggesting areas for enhancement. This iterative process may lead to refinements in the system, encompassing adjustments in feature extraction techniques, classifier selection, or data preprocessing steps. By systematically executing this evaluation methodology, a holistic understanding of the system's effectiveness and adaptability across

diverse anonymous mail datasets is attained, providing valuable insights for potential improvements.

PROTOTYPE

Creating a prototype of an authorship attribution system using Python involves several steps, including data preprocessing, feature extraction, model training, and evaluation. Below is a basic example using a Support Vector Machine (SVM) classifier for simplicity:

pythonCopy code

Step 1: Data Preprocessing (Assuming you have labeled data)# - Load and preprocess the dataset# - Extract features (You can use any specialized data compression technique for this)# - Split data into training and testing sets

Step 2: Model Trainingfrom sklearn.svm import SVCfrom sklearn.metrics import accuracy_score

Assuming X_train contains features and y_train contains labels

svm_classifier = SVC(kernel='linear', C=1.0, probability=True)

svm_classifier.fit(X_train, y_train)

Step 3: Model Evaluation# Assuming X_test contains features and y_test contains labels

y_pred = svm_classifier.predict(X_test)

accuracy = accuracy_score(y_test, y_pred)

print(f'Accuracy: {accuracy}')

Step 4: Predict Author of Anonymous Mail# - Extract features from the anonymous mail using your compression technique# - Use the trained SVM classifier to predict the author

predicted_author = svm_classifier.predict([anonymous_mail_features])

print(f'Predicted Author: {predicted_author[0]}')

In this code:

Data is preprocessed, which involves loading the dataset, extracting features (using your specialized data compression technique), and splitting the data into training and testing sets.

A Support Vector Machine (SVM) classifier is chosen for the model. It's then trained on the training set.

The model is evaluated using the testing set, and accuracy is calculated.

To predict the author of an anonymous mail, you'll need to extract features from the mail using your specialized compression technique, then use the trained SVM classifier to make the prediction.

Please ensure you have the necessary libraries installed (scikit-learn) using pip install scikit-learn. Also, replace placeholders like X_train, y_train, X_test, y_test, anonymous_mail_features with your actual data.

This is a basic prototype. Depending on the complexity of your specialized compression technique and the specific requirements of your system, you may need to implement additional steps or optimizations.

DATASET OF ANONYMOUS MAIL AND KNOWN AUTHORIAL TEXT FOR TRAINING AND EVALUATION.

Public Databases and Repositories:

There are various online resources where you can find datasets related to authorship attribution or text classification. Websites like Kaggle, UCI Machine Learning Repository, and GitHub repositories often have publicly available datasets.

Use APIs to Access Text Data:

You can utilize APIs provided by platforms like Reddit, Twitter, or other social media platforms to collect anonymous messages or posts. However, always respect privacy and terms of service when collecting data from such platforms.

Generate Synthetic Anonymous Data:

If obtaining real anonymous mail is not feasible due to privacy or legal concerns, you may consider generating synthetic anonymous texts using natural language generation techniques. These can be used for training and testing.

Ethical Considerations:

When collecting or using data, especially anonymous messages, it's crucial to respect privacy and ensure compliance with ethical guidelines. Avoid using or sharing any personal or sensitive information without proper consent.

Labeling the Data:

For known authorial text, you can use texts from well-known authors, blogs, articles, or any other public domain content. For anonymous mail, ensure that you anonymize any sensitive information before including it in your dataset.

Balancing the Dataset:

Make sure that the dataset is balanced, meaning it contains a similar number of samples from known authors and anonymous sources. This helps in training a more accurate and representative model.

Data Preprocessing:

Clean and preprocess the data to remove any noise, irrelevant information, or special characters that may not contribute to the authorship attribution task.

Remember to always respect privacy and copyright laws when collecting or using any data. Additionally, if you plan to share or use the data for research purposes, ensure that you have the necessary rights and permissions to do so.

Train the machine learning classifier on the training dataset.

Training a machine learning classifier involves feeding it with labeled data so that it learns to make predictions based on the provided features. In this case, we'll assume you have a dataset containing known authorial texts and anonymous mails, with appropriate labels indicating their respective authors.

Let's use a Support Vector Machine (SVM) classifier as an example. Here's how you can train it using Python and scikit-learn:

pythonCopy code

```
from sklearn.svm import SVC

# Assuming X_train contains features and y_train contains labels
svm_classifier = SVC(kernel='linear', C=1.0, probability=True)
svm_classifier.fit(X_train, y_train)
```

In this code:

SVC is the Support Vector Machine classifier from scikit-learn.

kernel='linear' specifies that we're using a linear kernel. You can experiment with different kernel functions depending on your dataset.

C=1.0 controls the regularization strength. You can adjust this hyperparameter based on your dataset and performance.

Make sure to replace X_train with your feature matrix and y_train with the corresponding labels.

If you have preprocessed your data and extracted features properly, the classifier will learn to associate certain patterns in the features with specific authors.

Always remember to validate your model on a separate test set to assess its generalization ability and ensure it doesn't overfit to the training data.

Evaluate the performance of the system on the evaluation dataset.

Evaluating the performance of the authorship attribution system involves using a separate dataset (evaluation dataset) that the model has never seen before. This allows us to assess how well the model generalizes to new, unseen data. Here's how you can do it:

pythonCopy code

Assuming X_eval contains features and y_eval contains labels

y_pred = svm_classifier.predict(X_eval)

Evaluate performancefrom sklearn.metrics import accuracy_score, precision_score, recall_score, f1_score

accuracy = accuracy_score(y_eval, y_pred)

precision = precision_score(y_eval, y_pred, average='weighted')

recall = recall_score(y_eval, y_pred, average='weighted')

f1 = f1_score(y_eval, y_pred, average='weighted')

print(f'Accuracy: {accuracy}')print(f'Precision: {precision}')print(f'Recall: {recall}')print(f'F1 Score: {f1}')

In this code:

X_eval contains the features of the evaluation dataset, and y_eval contains the corresponding labels.

svm_classifier.predict(X_eval) uses the trained classifier to make predictions on the evaluation dataset.

The evaluation metrics used here include:

Accuracy: The proportion of correctly classified instances out of the total instances.

Precision: The proportion of true positives among all positive predictions.

Recall: The proportion of true positives among all actual positives.

F1 Score: The harmonic mean of precision and recall, which provides a balance between the two metrics.

Make sure to replace X_eval and y_eval with your actual evaluation dataset.

Remember to choose the appropriate metrics based on your specific authorship attribution task and dataset characteristics.

Evaluate the system on a variety of anonymous mail datasets using different evaluation metrics.

To evaluate the authorship attribution system on a variety of anonymous mail datasets using different evaluation metrics, follow these steps:

Collect Diverse Anonymous Mail Datasets:

Ensure you have a collection of anonymous mail datasets with different characteristics, such as varying writing styles, languages, and genres.

Preprocess and Extract Features:

Preprocess each dataset and extract features using your specialized data compression technique.

Load Trained Classifier:

Load the trained classifier that was previously trained on known authorial texts.

Evaluate on Different Datasets:

For each anonymous mail dataset:

a. Load and Preprocess Data:

Load the anonymous mails and preprocess them as required.

b. Extract Features:

Apply the same feature extraction process that was used during training.

c. Predict Authors:

Use the trained classifier to predict the authors of the anonymous mails.

d. Calculate Evaluation Metrics:

Utilize various evaluation metrics (e.g., accuracy, precision, recall, F1-score) to assess the system's performance on each dataset.

Aggregate and Compare Results:

Collect the evaluation metrics for each dataset and compare the performance of the system across the different datasets.

Interpret Results:

Analyze the results to understand how well the system generalizes across diverse anonymous mail datasets. Identify any patterns, strengths, or potential areas for improvement.

Document Findings:

Record the evaluation results, including the chosen evaluation metrics and their corresponding values for each dataset.

Iterate for Improvement:

Based on the evaluation results, consider refining the system, including adjustments to

feature extraction techniques, classifier selection, or data preprocessing steps.

By systematically evaluating the system on a variety of anonymous mail datasets using different evaluation metrics, you can gain valuable insights into its performance and identify strategies for enhancement.

Compare the performance of the system to other anonymous mail attribution systems.

To compare the performance of your authorship attribution system for anonymous mail with other existing systems, follow these steps:

Select Benchmark Datasets:

Choose benchmark datasets that are widely recognized and used in the field of authorship attribution. These datasets should have ground truth labels for authorship.

Obtain Existing Systems:

Identify other anonymous mail attribution systems that have been previously published or are available for evaluation.

Preprocess Data:

Preprocess the benchmark datasets and ensure they are in a suitable format for evaluation.

Apply Existing Systems:

Apply the selected existing systems to the benchmark datasets. Record the results.

Evaluate Your System:

Use your authorship attribution system to predict authors on the same benchmark datasets.

Choose Evaluation Metrics:

Select appropriate evaluation metrics for comparison. Common metrics include accuracy, precision, recall, and F1-score.

Analyze and Compare Results:

Compare the performance of your system with the existing systems on the benchmark datasets using the chosen evaluation metrics.

Statistical Analysis:

Consider conducting statistical tests (e.g., t-tests, ANOVA) to determine if any observed differences in performance are statistically significant.

Report Findings:

Document the results of the comparison, including the performance metrics and any statistically significant differences between systems.

Interpret and Discuss:

Analyze the findings and discuss the strengths and weaknesses of your system compared

to existing ones. Consider factors such as accuracy, computational efficiency, and generalization capabilities.

Consider Real-World Implications:

Think about how the performance of your system and its comparison to existing systems might impact real-world applications, such as cybersecurity or criminal investigations.

Iterate and Improve:

If there are areas where your system could be further improved based on the comparison results, consider making enhancements.

By systematically comparing the performance of your authorship attribution system with other existing systems on benchmark datasets, you can gain valuable insights into its effectiveness and potential contributions to the field.

Identify the strengths and weaknesses of the system.

Identifying the strengths and weaknesses of the authorship attribution system is crucial for understanding its performance and potential areas for improvement. Here's a breakdown of potential strengths and weaknesses:

Strengths:

Accurate Authorship Attribution:

The system demonstrates a high level of accuracy in attributing authorship to known authors, indicating that it effectively captures distinctive writing styles.

Multilingual Capability:

If applicable, the system performs well in multilingual scenarios, showcasing its ability to handle texts in different languages.

Robustness to Anonymous Mail Attribution:

The system shows promise in attributing authorship to anonymous mails, a complex task that traditional methods may struggle with.

Data Compression Efficiency:

The use of data compression techniques aids in feature extraction, allowing for efficient representation of textual characteristics.

Generalization Across Diverse Datasets:

The system demonstrates the ability to generalize well across diverse datasets, indicating adaptability to different writing styles and contexts.

Weaknesses:

Sensitivity to Noise:

The system may be sensitive to noise or inconsistencies in the data, potentially leading to misattribution in cases where texts are less clear or uniform.

Limited by Training Data:

The performance of the system heavily relies on the quality and diversity of the training data. Insufficient or biased training data can lead to suboptimal results.

Computational Complexity:

Depending on the complexity of the data compression techniques used, the system may require significant computational resources, potentially limiting its scalability.

Vulnerability to Adversarial Attacks:

The system may be susceptible to adversarial attacks aimed at deceiving the attribution process, especially if the compression techniques are not robust to deliberate manipulation.

Challenges in Multilingual Settings:

Handling multilingual authorship attribution may still present challenges, especially if the system struggles with identifying distinctive features in different languages.

Difficulty with Slang or Unconventional Writing Styles:

The system may have difficulty attributing authorship in cases of highly informal or unconventional writing styles, where traditional linguistic markers may be less prevalent.

Potential Areas for Improvement:

Enhance Noise Robustness:

Implement techniques to reduce the sensitivity to noise and improve the system's ability to handle less clear or uniform texts.

Expand and Diversify Training Data:

Continuously update and expand the training data to include a wider range of authors, genres, and writing styles for improved generalization.

Optimize Computational Efficiency:

Explore techniques to optimize the computational resources required by the system,

potentially through more efficient compression algorithms or parallel processing.

Incorporate Language-Specific Features:

Consider incorporating language-specific linguistic features to enhance performance in multilingual settings.

Adversarial Defense Mechanisms:

Implement strategies to detect and mitigate potential adversarial attacks, ensuring the system's resilience to deliberate manipulation.

Contextual Analysis:

Consider incorporating contextual analysis techniques to better handle informal or unconventional writing styles, where traditional linguistic markers may be less reliable.

By critically assessing the strengths and weaknesses of the system, you can identify specific

areas for improvement and prioritize future research and development efforts.

Optimize the system for performance and scalability.

Optimizing the authorship attribution system for performance and scalability involves several steps. Below are some strategies you can implement:

1. Efficient Feature Extraction:

Utilize advanced data compression techniques that strike a balance between accuracy and computational efficiency. Consider techniques like n-grams, word embeddings, or other specialized algorithms tailored for authorship attribution.

2. Model Selection and Hyperparameter Tuning:

Experiment with different machine learning models (e.g., SVM, Random Forest, Neural Networks) to identify the most suitable one for your specific task. Perform hyperparameter tuning to fine-tune model performance.

3. Parallel Processing and Distributed Computing:

Leverage parallel processing and distributed computing frameworks (e.g., Apache Spark, Dask)

to process and analyze large volumes of text data concurrently, improving overall system performance.

4. Feature Selection and Dimensionality Reduction:

Use techniques like Principal Component Analysis (PCA) or feature selection algorithms to reduce the dimensionality of the feature space while retaining relevant information, thereby improving computational efficiency.

5. Model Pruning and Compression:

Implement techniques like model pruning and compression to reduce the size and computational requirements of the trained model without sacrificing performance.

6. Batch Processing:

If processing large volumes of data, consider batch processing to handle data in smaller, manageable chunks. This can improve memory utilization and processing speed.

7. Algorithmic Optimizations:

Optimize algorithms for critical operations like similarity calculations or distance metrics,

ensuring they are implemented in an efficient manner.

8. Caching and Memoization:

Implement caching mechanisms to store and reuse intermediate results of computations, reducing redundant calculations and improving processing speed.

9. Distributed Storage and Data Partitioning:

Store data in a distributed file system (e.g., HDFS) and partition it strategically to optimize data retrieval and processing across multiple nodes.

10. GPU Acceleration:

Utilize Graphics Processing Units (GPUs) to accelerate certain computations, particularly in deep learning models or algorithms that benefit from parallel processing.

11. Profile and Benchmark:

Regularly profile the system to identify performance bottlenecks and areas for improvement. Benchmark different components to assess their computational efficiency.

12. Scalability Architecture:

Design the system with scalability in mind. Consider microservices architecture,

containerization (e.g., Docker), and orchestration tools (e.g., Kubernetes) for efficient resource utilization.

13. Load Balancing:

Implement load balancing techniques to evenly distribute processing tasks across multiple servers or computing resources.

14. Monitoring and Scaling:

Use monitoring tools to track system performance in real-time. Implement auto-scaling solutions to dynamically adjust resources based on demand.

By applying these optimization strategies, you can enhance the performance and scalability of your authorship attribution system, allowing it to handle larger datasets and deliver more efficient results. Remember to measure and validate the impact of each optimization to ensure it aligns with your system's specific requirements and objectives.

Develop a user interface for interacting with the system.

To develop a user interface (UI) for interacting with the authorship attribution system, you can create a web-based application using

HTML, CSS, and JavaScript for the front-end, and choose a suitable back-end technology (such as Python with Flask or Django) to handle requests and communicate with the system. Here's a basic outline to get you started:

Front-End (HTML, CSS, JavaScript):

Create the HTML Structure:

Design the layout of the UI, including input fields, buttons, and result displays. Use HTML to structure the elements.

Style with CSS:

Apply CSS to add visual design, including colors, fonts, and layout.

Implement User Interactions with JavaScript:

Use JavaScript to handle user interactions, such as button clicks, form submissions, and result displays.

Design Input Forms:

Include input fields for users to input text for authorship attribution. This could be a text box for typing or uploading a document.

Display Results:

Set up an area to display the results of the authorship attribution, including the predicted author and any additional information.

Add User Feedback:

Provide feedback to the user, such as success messages, error messages, or loading indicators.

Back-End (Python with Flask Example):

Set Up a Web Server:

Use a web framework like Flask to handle HTTP requests and responses.

Create Routes:

Define routes that correspond to different interactions with the UI (e.g., handling form submissions).

Integrate with Authorship Attribution System:

Implement the logic to pass user inputs to the authorship attribution system, process the results, and return them to the front-end.

Handle File Uploads:

If users can upload documents, implement a mechanism to handle file uploads securely.

Error Handling:

Add error handling to manage any issues that may arise during interactions with the system.

Cross-Origin Resource Sharing (CORS):

If the UI and the system are hosted on different domains, configure CORS settings to allow communication between them.

Integration with Authorship Attribution System:

Invoke the Authorship Attribution System:

Call the authorship attribution system using the appropriate methods or APIs, passing user inputs for processing.

Process the Results:

Receive and process the results returned by the authorship attribution system, and prepare them for display in the UI.

Deployment:

Host the Application:

Choose a hosting platform (e.g., Heroku, AWS, or a web hosting service) to deploy your application.

Set Up a Database (if needed):

If your application requires data storage, set up a database to persistently store user interactions or results.

Secure the Application:

Implement security measures to protect user data and prevent unauthorized access.

Testing and Debugging:

Thoroughly test the application to identify and fix any bugs or issues.

User Interface Design Tips:

Keep the UI clean, intuitive, and user-friendly.

Provide clear instructions and feedback to guide users through interactions.

Use responsive design to ensure the UI is accessible on various devices.

Consider accessibility features for users with disabilities.

Implement a visually appealing and consistent design.

Remember to thoroughly test the UI for usability and functionality to ensure a seamless user experience.

Creating a full-fledged UI involves detailed design work, including wireframing, prototyping, and implementation. Here, I'll provide a basic HTML/CSS template for a simple authorship attribution system UI. You can further enhance and customize it based on your specific requirements.

Html code

```
<!DOCTYPE html><html lang="en"><head>
    <meta charset="UTF-8">
    <meta name="viewport" content="width=device-width, initial-scale=1.0">
    <title>Authorship Attribution System</title>
    <link rel="stylesheet" href="styles.css"></head><body>
```

```
<header>
    <h1>Authorship Attribution System</h1>
</header>

<div class="container">
    <div class="input-section">
        <h2>Input Text</h2>
        <textarea id="inputText" rows="10" cols="50"></textarea>
        <button onclick="submitText()">Submit</button>
    </div>

    <div class="result-section" id="resultSection">
        <h2>Result</h2>
        <div id="authorResult"></div>
    </div>
</div>

<script src="script.js"></script></body></html>
```

CSS (styles.css):

```
cssCopy code
body {
    font-family: Arial, sans-serif;
    margin: 0;
    padding: 0;
    background-color: #f4f4f4;
}
header {
    background-color: #333;
    color: #fff;
    text-align: center;
    padding: 10px 0;
}
.container {
    max-width: 800px;
    margin: 20px auto;
    background-color: #fff;
    padding: 20px;
    border-radius: 5px;
    box-shadow: 0 2px 4px rgba(0,0,0,0.1);
}
.input-section, .result-section {
    margin-bottom: 20px;
}
```

```
textarea {
    width: 100%;
    padding: 10px;
    border: 1px solid #ccc;
    border-radius: 5px;
}
button {
    padding: 10px 20px;
    background-color: #333;
    color: #fff;
    border: none;
    border-radius: 5px;
    cursor: pointer;
}
button:hover {
    background-color: #555;
}
#authorResult {
    font-weight: bold;
}
```

JavaScript (script.js):

javascriptCopy code

```
function submitText() {
```

```
        var inputText = document.getElementById("inputText").value;

        // Send inputText to your backend for authorship attribution
        // Process the response and display the result
        var authorResult = document.getElementById("authorResult");
        authorResult.innerHTML = "Predicted Author: [Author Name]";
    }
```

This is a basic HTML/CSS/JavaScript template for an authorship attribution system UI. It includes an input section with a textarea for user input and a "Submit" button. The result section will display the predicted author. Remember to replace [Author Name] with the actual predicted author.

Please note that this is a simplified example. In a real-world scenario, you would need to implement the backend logic to handle user input, perform authorship attribution, and return the results to the UI. You may also need to implement

features like file uploads, error handling, and more sophisticated user interactions.

Deploy the system on a cloud computing platform.

To deploy the authorship attribution system on a cloud computing platform, you'll need to choose a platform (such as AWS, Google Cloud, or Azure) and follow these general steps:

1. Select a Cloud Provider:

Choose a cloud computing platform that best suits your needs. Some popular options include:

Amazon Web Services (AWS)

Google Cloud Platform (GCP)

Microsoft Azure

Heroku (simpler deployment for smaller projects)

2. Set Up an Account:

Create an account on the chosen cloud platform and log in to the management console.

3. Prepare Your Application:

Ensure your authorship attribution system (including the back-end and any necessary dependencies) is ready for deployment.

4. Set Up a Virtual Machine or Container:

Virtual Machine (VM):

Create a virtual machine instance on the cloud platform.

Install necessary software, including the web server and any dependencies.

Upload your application files to the VM.

Container (Docker):

Dockerize your application by creating a Dockerfile that defines the environment and dependencies.

Build a Docker image.

Push the image to a container registry on the cloud platform.

5. Deploy the Application:

VM:

Start the VM and configure any necessary networking settings (firewalls, security groups, etc.).

Set up a web server (e.g., Nginx, Apache) to serve your application.

Container (Docker):

Deploy the Docker container using a container service provided by the cloud platform (e.g., AWS ECS, Google Kubernetes Engine, Azure Container Instances).

6. Set Up a Domain (Optional):

If you have a custom domain, configure it to point to your cloud-based application. You may need to set up DNS records.

7. Configure Security:

Ensure your application and server are secure by implementing best practices, such as firewalls, SSL certificates, and access controls.

8. Monitor and Scale:

Set up monitoring tools to track system performance and usage. Configure auto-scaling policies to dynamically adjust resources based on demand.

9. Test and Debug:

Thoroughly test the deployed system to ensure it works as expected in the cloud environment. Address any issues that arise.

10. Back Up Data:

Implement regular backups of your application and data to prevent data loss in case of any unforeseen events.

11. Document:

Keep detailed documentation of your deployment process, configurations, and any troubleshooting steps.

Remember to refer to the specific documentation and guidelines provided by your chosen cloud platform, as the steps may vary slightly depending on the platform you're using.

RESULT

Table 1. Description of the literary texts' dataset.

Dataset Characteristic	Value of the Characteristic
Number of authors	100
Number of texts	1100
Dataset size, symbols	375,618,852
Dataset size, words	62,603,142
Dataset size, sentences	5,216,929
The average length of text, symbols	973,342
The average length of sentence, words	14.3
Maximum number of texts per author	20
Minimal number of texts per author	9

Table 2. Description of the short texts' dataset.

Dataset Characteristic	Value of the Characteristic
Number of authors	3075
Number of texts	202,892
Dataset size, symbols	30,652,109
Dataset size, words	4,708,619
The average length of text, symbols	151.1
The average length of text, words	23.7
The average number of texts per author	115.37

Table 3. Results of author identification using ML trained on the feature space.

Number of Authors	Accuracy of Models, %					
	SVM	LR	NB	DT	RF	KNN
2	**95.4 ± 1.7**	95.1 ± 4.4	91.9 ± 3.4	95.1 ± 1.1	97.4 ± 1.7	94.0 ± 2.6
5	**94.6 ± 2.1**	92.8 ± 3.1	87.1 ± 3.6	92.4 ± 2.3	92.1 ± 3.7	81.1 ± 2.1
10	**81.9 ± 4.6**	74.2 ± 5.1	74.4 ± 5.6	84.2 ± 4.4	67.6 ± 2.6	79.9 ± 3.3
20	**63.3 ± 4.8**	61.9 ± 5.1	58.3 ± 4.5	52.2 ± 2.3	62.2 ± 3.7	51.9 ± 5.2
50	37.7 ± 6.3	34.7 ± 7.1	29.8 ± 5.6	17.8 ± 2.7	35.9 ± 2.5	**41.9 ± 4.0**
Avg. accuracy	**74.7 ± 3.9**	71.2 ± 4.9	68.3 ± 4.5	68.7 ± 2.6	70.6 ± 2.8	69.8 ± 3.4

Table 4. Results of author identification using ML trained on the TF-IDF.

Number of Authors	Accuracy of Models, %					
	SVM	LR	NB	DT	RF	KNN
2	92.4 ± 1.1	**93.2 ± 2.3**	85.7 ± 2.7	90.1 ± 1.1	92.5 ± 2.2	86.9 ± 3.3
5	84.5 ± 3.4	82.2 ± 4.3	72.7 ± 4.8	81.1 ± 3.2	**88.5 ± 3.1**	78.0 ± 2.6
10	72.2 ± 5.6	68.7 ± 5.5	59.4 ± 4.9	**75.3 ± 2.2**	70.6 ± 1.1	71.4 ± 4.9
20	55.2 ± 4.2	52.3± 4.8	49.3 ± 5.3	33.6 ± 3.1	**59.0 ± 3.4**	57.4 ± 2.6
50	33.2 ± 4.8	27.7 ± 3.3	22.8 ± 4.1	16.1 ± 5.1	31.4 ± 4.1	**40.2 ± 3.4**
Avg. accuracy	67.5 ± 3.9	64.8 ± 4.1	57.9 ± 4.6	59.3 ± 2.9	**68.4 ± 2.9**	66.8 ± 3.5

Table 5. Results of author identification using NNs.

Number of Authors	Accuracy of Models, %								
	LSTM	BiLSTM	CNN	CNN + LSTM	LSTM + CNN	CNN + CNN	fastText	RuBERT	MultiBERT
2	94.3 ± 5.5	95.5 ± 4.5	97.1 ± 3.6	94.5 ± 6.0	98.5 ± 5.6	**98.8 ± 4.1**	98.2 ± 4.5	95.2 ± 2.6	93.6 ± 2.8
5	86.7 ± 6.3	82.5 ± 4.8	**95.9 ± 2.8**	86.9 ± 4.8	95.5 ± 3.9	94.7 ± 3.4	95.0 ± 3.7	90.4 ± 4.1	89.8 ± 3.9
10	75.4 ± 3.3	70.2 ± 5.3	81.3 ± 5.9	78.2 ± 5.0	82.2 ± 5.4	86.5 ± 6.1	**92.2 ± 6.3**	84.3 ± 3.3	81.8 ± 3.5
20	63.9 ± 5.7	58.7 ± 4.9	71.2 ± 5.8	65.8 ± 3.2	62.2 ± 5.8	**72.8 ± 4.4**	69.9 ± 4.3	67.4 ± 4.1	64.9 ± 3.1
50	44.2 ± 6.1	41.1 ± 5.1	51.1 ± 5.1	55.0 ± 5.2	41.3 ± 4.5	**56.9 ± 4.2**	54.8 ± 6.2	52.2 ± 3.6	46.1 ± 4.0
Avg. accuracy	72.9 ± 5.4	69.6 ± 5.1	79.3 ± 4.8	76.1 ± 4.9	75.9 ± 5.1	**82.3 ± 4.8**	82.1 ± 6.0	77.9 ± 3.5	75.2 ± 3.5

Table 6. Training time on the dataset of 50 authors.

Training Time on Feature Vector, Sec.								
SVM	**LR**		**NB**	**DT**		**RF**		**KNN**
1582	1082		677	714		1243		1134
Training Time on TF-IDF, Sec.								
SVM	**LR**		**NB**	**DT**		**RF**		**KNN**
1823	1418		746	1371		2334		2871
Training Time of Neural Networks, Sec.								
LSTM	**BiLSTM**	**CNN**	**CNN + LSTM**	**LSTM + CNN**	**CNN + CNN**	**fastText**	**RuBERT**	**MultiBERT**
58,133	65,284	43,191	52,638	50,452	50,679	26,723	48,634	49,629

Table 7. Results of author identification using ML trained on the feature space.

Number of Authors	Accuracy of Models,%					
	SVM	**LR**	**NB**	**DT**	**RF**	**KNN**
2	**72.2 ± 4.0**	67.1 ± 3.1	62.9 ± 2.3	69.1 ± 2.1	71.2 ± 3.9	68.1 ± 4.2
5	**69.9 ± 3.5**	59.5 ± 4.2	58.6 ± 2.6	43.5 ± 2.1	56.1 ± 2.8	65.4 ± 3.7
10	**66.3 ± 3.8**	48.2 ± 2.9	45.9 ± 3.5	24.2 ± 3.6	37.6 ± 2.7	61.9 ± 4.0
20	**55.3 ± 3.1**	34.3 ± 3.4	38.8 ± 4.1	19.9 ± 4.1	32.2 ± 1.7	43.9 ± 4.1
50	32.1 ± 3.9	28.6 ± 3.6	27.1 ± 3.3	15.9 ± 3.3	25.9 ± 2.4	**33.6 ± 3.4**
Avg. accuracy	59.2 ± 3.6	47.6 ± 3.4	46.8 ± 3.2	34.5 ± 3.0	44.6 ± 2.7	54.6 ± 3.9

Table 10. Training time on the dataset of 50 authors.

Training Time on Feature Vector, Sec.								
SVM	**LR**		**NB**	**DT**		**RF**		**KNN**
589	**397**		**308**	**236**		804		604
Training Time on TF-IDF, Sec.								
SVM	**LR**		**NB**	**DT**		**RF**		**KNN**
717	584		**372**	416		1393		955
Training Time of Neural Networks, Sec.								
LSTM	**BiLSTM**	**CNN**	**CNN + LSTM**	**LSTM + CNN**	**CNN + CNN**	**fastText**	**RuBERT**	**MultiBERT**
30,190	32,980	25,380	28,397	26,467	25,874	15,926	26,547	27,117

Table 11. Results of GA for social media texts dataset.

Number of Authors	Number of Features						
	1168	500	400	300	200	100	50
2	72.2 ± 4.0	75.3 ± 5.2	**80.3 ± 3.3**	75.2 ± 4.7	67.2 ± 4.5	64.9 ± 3.8	65.3 ± 5.5
5	69.9 ± 3.5	70.4 ± 2.5	**77.1 ± 2.8**	72.4 ± 1.9	63.8 ± 3.9	59.0 ± 4.2	49.8 ± 3.9
10	66.3 ± 3.8	67.8 ± 3.7	**71.9 ± 2.6**	66.6 ± 3.1	60.2 ± 3.8	57.2 ± 3.9	47.2 ± 2.1
20	55.4 ± 3.1	62.4 ± 2.9	**64.8 ± 3.0**	59.3 ± 2.8	52.9 ± 4.1	49.4 ± 4.2	43.7 ± 3.7
50	32.1 ± 3.9	35.1 ± 6.3	**37.3 ± 4.1**	33.5 ± 2.5	27.4 ± 4.3	26.8 ± 3.0	22.4 ± 4.0
Avg. accuracy	59.2 ± 3.4	62.2 ± 4.1	**66.3 ± 3.2**	61.4 ± 3.0	54.3 ± 4.1	51.5 ± 3.8	45.7 ± 3.8

Table 8. Results of author identification using ML trained on the TF-IDF.

Number of Authors	Accuracy of Models, %					
	SVM	LR	NB	DT	RF	KNN
2	61.1 ± 3.1	69.4 ± 5.2	**70.8 ± 1.0**	69.1 ± 2.1	68.4 ± 2.9	57.9 ± 1.7
5	54.4 ± 6.0	58.1 ± 4.4	**66.1 ± 7.3**	43.5 ± 2.1	60.4 ± 3.2	54.2 ± 3.1
10	39.4 ± 3.9	44.7 ± 0.7	**49.6 ± 5.4**	24.2 ± 3.6	42.6 ± 1.9	45.3 ± 1.9
20	32.0 ± 1.0	36.1 ± 2.2	**46.1 ± 2.3**	15.4 ± 3.6	33.6 ± 2.2	40.1 ± 2.3
50	17.3 ± 2.6	24.8 ± 3.1	**34.1 ± 5.0**	11.0 ± 4.8	23.5 ± 1.9	32.7 ± 2.45
Avg. accuracy	40.9 ± 3.3	46.6 ± 3.1	53.3 ± 4.2	32.5 ± 3.2	45.7 ± 2.4	46.1 ± 2.3

Table 9. Results of author identification using NNs.

Number of Authors	Accuracy of Models, %								
	LSTM	BiLSTM	CNN	CNN + LSTM	LSTM+CNN	CNN + CNN	fastText	RuBERT	MultiBERT
2	93.0 ± 1.9	94.6 ± 2.1	**95.6 ± 2.0**	95.5 ± 3.5	92.3 ± 2.1	91.3 ± 2.4	94.0 ± 1.2	93.3 ± 2.1	90.2 ± 1.9
5	89.7 ± 1.9	92.5 ± 2.4	93.3 ± 1.5	**90.9 ± 2.2**	90.2 ± 1.0	89.2 ± 2.2	87.2 ± 2.2	88.6 ± 1.8	87.1 ± 2.2
10	73.0 ± 2.8	71.3 ± 2.4	72.4 ± 2.7	**77.1 ± 3.9**	64.2 ± 3.3	76.6 ± 4.5	76.1 ± 3.5	76.6 ± 3.2	69.5 ± 3.0
20	68.8 ± 2.4	59.3 ± 1.3	67.9 ± 3.3	62.2 ± 3.4	61.3 ± 3.2	**73.7 ± 2.5**	68.4 ± 2.3	66.8 ± 3.3	63.4 ± 2.7
50	50.1 ± 2.6	49.6 ± 3.9	48.8 ± 3.6	47.3 ± 1.9	47.4 ± 2.8	50.2 ± 1.4	**55.6 ± 2.8**	50.0 ± 2.9	47.1 ± 2.8
Avg. accuracy	74.9 ± 2.3	73.5 ± 2.4	75.6 ± 2.6	74.6 ± 3.0	71.1 ± 2.5	76.2 ± 2.6	**76.3 ± 2.4**	75.0 ± 2.7	71.5 ± 2.5

Table 12. Results of GA for literary texts dataset.

Number of Authors	Number of Features						
	1168	500	400	300	200	100	50
2	95.4 ± 1.7	96.1 ± 2.2	**98.6 ± 2.7**	94.5 ± 1.9	96.2 ± 1.6	98.3 ± 3.9	95.9 ± 3.1
5	94.6 ± 2.1	94.7 ± 3.3	**97.5 ± 3.1**	90.6 ± 1.8	95.0 ± 3.5	97.4 ± 1.9	94.8 ± 2.0
10	81.9 ± 4.6	83.7 ± 4.1	**88.0 ± 2.9**	82.1 ± 1.2	87.1 ± 3.4	85.9 ± 3.8	84.3 ± 3.2
20	63.3 ± 4.8	69.1 ± 2.5	**73.7 ± 3.3**	70.7 ± 2.4	72.9 ± 2.8	63.2 ± 2.5	60.7 ± 2.6
50	37.7 ± 6.3	40.2 ± 2.0.	**44.4 ± 2.6**	40.0 ± 3.7	42.4 ± 4.2	38.1 ± 3.7	33.8 ± 2.3
Avg. accuracy	74.7 ± 3.9	76.8 ± 2.9	**80.4 ± 2.9**	75.6 ± 2.2	78.3 ± 3.1	76.6 ± 3.2	73.9 ± 2.6

Herein are the results and training timings for all models mentioned across both datasets. The social media dataset was divided eighty to twenty between training and testing samples. In their works, each author utilised three training materials and one exam text. In every instance, a cross-validation strategy was employed. The proportion of correct classifications relative to the total number of classifications made by the classifier was used to calculate accuracy.

The authors agreed that a word count of 20,000 was sufficient proof of authorship. The research text was restricted to 15,000 characters to enhance task difficulty. This volume served as the

court proceedings' textual corpus. Determine whether the suggested modifications and underutilised classifiers make a difference.

Tables 3 and 4 display the accuracy for 2, 5, 10, 20, and 50 author instances, as well as the average accuracy for each model, for ML models trained on feature space and TF-IDF, respectively.

The outcomes of an ML author-identification task utilising a TF-IDF-trained model are presented in Table 4.

Using the same datasets and authors, Table 5 displays the results of employing NNs to the task of author identification.

In Table 5, we can see the effects of using NNs to ascertain authors.

Table 6 displays the total time spent training each model with the 50-author dataset.

Table 6 displays the quantity of time spent training 50 writers using the dataset.

The constraint of 15,000 characters does not prohibit the application of standard machine learning algorithms trained on the newly generated feature space. On datasets with 2, 5, or 10 authors, the outcomes of SVM, RF, and KNN are comparable to those of deep NNs. The amount of text fragments is sufficient to determine the author's writing style; the completeness of the set of features chosen for identification; training on carefully selected experimental parameters of ML models; the ability of SVM to work with a large feature space and solve problems of varying degrees of complexity due to its high degree of flexibility; and the reduction in the number of errors due to maximisation.

In comparison to other models, fastText's accuracy loss is less than 3% across all trials, and it outperforms them by a wide margin for 10 authors. FastText is also 51% quicker at learning than the average deep neural network. CNN and hybrid

networks, which are composed of convolutional networks, are much quicker to train than LSTM, BiLSTM, and BERT. Rapid training is accomplished by strictly parallelizing the convolution process for each map and employing inverse convolution when an error propagates through the network.

The accuracy for datasets with 2, 5, 10, 20, and 50 authors, as well as the average accuracy for each model, is presented in Tables 7 and 8, respectively, for ML models trained on feature space and TF-IDF. Due to the maximal classification accuracy of the approaches, only results for k = 25 and 35 trees were displayed for the KNN and RF algorithms.

The outcomes of author identification experiments utilising NNs on the same datasets and authors are presented in Table 9.

Here are the outcomes of using NNs to determine authorship.

The most effort is devoted categorising fifty authors. Table 10 depicts the total amount of time required to train each model using data from 50 authors.

Table 10 displays the quantity of time spent training 50 writers from a given dataset.

The results enable us to conclude that literary texts cannot be classified by conventional ML methods using the designed feature space. This is due to the brevity of the remarks. Since the comments' contents disclose the author's feelings about the commented-on article, the dataset is dominated by brief remarks and phrases. Since the text volume is so small, it is impossible to recognise the author's style even in a well-written feature section. SVM with empirically selected features trained on feature space achieves a maximum accuracy of 72% for two authors, whereas deep NNs can identify with a maximum accuracy of 96% for the same task. This outcome may be explained by the inherent capacity of deep NNs to select

implicit informative characteristics on their own. Only two models (LR and NB) benefited from the use of TF-IDF instead of a feature vector. In comparison to literary works, the accuracy of all models decreases considerably when dealing with 20 or 50 authors. FastText outperforms LSTM+CNN and BERT models across all author sets and acquires knowledge 39% quicker on average. Moreover, fastText outperforms BiLSTM for 2 and 10 authors, and it outperforms all other models for 50 authors. FastText is 42% quicker than the next fastest deep neural network at learning.

Word selection, regional speech patterns, sentence length, phrase selection, and vocabulary are just some of the writing characteristics that can be used to infer the author's personality. However, altering these parameters has various effects on the frequency qualities of the text. This creates the difficulty of selecting a set of pertinent characteristics without including redundant ones.

Utilising genetic algorithms for feature selection enables the selection of an optimal subset

of characteristics from the entire set of characteristics used. This procedure reduces the dimensionality of the feature space, which accelerates model training and improves accuracy by eliminating superfluous vector components.

Three processes comprise GA: crossing, mutation, and selection. All operators have rates between zero and one. In contrast to the value 0, which indicates the operator's complete absence from the algorithm, the value 1 indicates the operator's greatest endeavour.

Using the selection operator to select subsets of characteristics is required for the algorithm to proceed. The selection process may be conditional or wholly arbitrary. The "fitness" function is optimised for the provided individuals (features) on the chosen subset. Mutation and crossover processes "reproduce" the subsequent generation from the selected individuals. The number of generations is determined by the rates of crossover and mutation. The greater the number of generations, the greater the likelihood that the

optimal population, in which genes flourish, will be discovered.

Ending criteria may include the attainment of a predetermined level of classification accuracy, the discovery of a local or global optimum, the expiration of the algorithm's execution time, or the execution of a predetermined number of calls to the desired function.

A feature is either present (represented by a 1 in the feature vector) or absent (represented by a 0 in the feature vector). Due to the fact that 1168 attributes were already in use, a comprehensive enumeration must evaluate 21168 subsets. In lieu of contemplating all potential options, genetic algorithms (GA) attempt to select distinct subgroups based on a set of input characteristics. Incorporating GA into the classifier enables feature selection, with "suitability" based on optimum accuracy or minimal loss.

In this investigation, we combined GA and SVM. SVM consistently demonstrated the highest

accuracy among the standard ML techniques. Since deep NNs can identify beneficial characteristics on their own, combining GA and NNs is unnecessary. Training an SVM requires five times less time than training a deep NN. This leads to the hypothesis that the accuracy of SVM classification will improve. These characteristics serve to characterise GA:

Two hundred people live there;

With a crossover ratio of 0.5:1,

• Rate of Mutation of 0.2;

Twenty populations are identified.

Experiments were conducted to determine the optimal method for extracting 50, 100, 200, 300, 400, and 500 pertinent characteristics from the initial set of 1168 components. The experiment results are presented in Table 11 for tweets and Table 12 for novels.

The social media communications' GA results are displayed in Table 11.

The outcomes of the literary analysis are shown in Table 12.

The results indicate that halving the number of features (to 400) has no effect on classification accuracy and may even improve results for both datasets. For 200 features, the obtained accuracy is greater than the original, whereas for shorter texts, the achieved accuracy is comparable to the original. It is not always possible to identify the author using the combination of 100 and 50 attributes. The optimal collection of 400 features included 6 punctuation marks, 8 grammatical categories, 165 items from the frequency dictionary, and 20 unigrams, 107 bigrams, and 98 trigrams at the character level.

Using a rank-based non-parametric test, it was determined if there was a statistically significant difference between the results of the SVM trained with various quantities of pertinent features chosen by the GA. To compare results from numerous cross-validation folds, Friedman and Némenyi post hoc tests were used. In ML, Friedman and Némenyi tests are recommended. For these examinations, the most difficult case study, comprised of 50 authors, was utilised. It was assumed that differences in outcomes across differing degrees of characteristics were the result of pure coincidence. One plausible explanation for the discrepancy between the two data sets was that there was, in fact, a difference. Literary texts had a p-value of 0.017, while social media comments had a p-value of 0.007. The presented findings are statistically significant (p0.05); therefore, the null hypothesis can be rejected.

If the average rankings differ by more than a predetermined threshold, there is a wide variation in method efficacy. After the null hypothesis was refuted by the Friedman test, a Némenyi post-hoc

test was used to determine the significance of the finding. Frequently, researchers use the Némenyi post-hoc test to distinguish subsets of data. The Némenyi test is designed to compare the efficacy of two candidates. Figure 2 shows a Demar diagram depicting the results. These examples serve to illustrate important differences between the approaches. When the difference in mean ranks is less than the automatically computed critical difference value, a horizontal line indicates that the performance difference between the two techniques is not significant.

CONCLUSION

In the realm of authorship attribution, a field at the intersection of linguistics, computational analysis, and forensics, this initiative has embarked on a transformative journey. The mission, intricate and demanding, is to unveil the mysterious connection between a writer and their words. It is a quest to decipher the fingerprint that each author leaves behind, an intricate amalgamation of linguistic nuances that define their unique style. This endeavor, however, is not without its challenges. Language, a living entity, is shaped by an array of contextual influences, rendering authorship attribution a formidable task.

In response to this challenge, data compression techniques have emerged as a beacon of promise. Rooted in the foundational principles of information theory, these techniques view written text as a sequence of information, ripe for analysis. By employing compression algorithms, patterns and redundancies indicative of an author's style can be discerned, illuminating a path towards a paradigm shift in authorship attribution.

This report is dedicated to the exploration of the potential of data compression techniques in advancing the field of authorship attribution, with a particular focus on two distinct yet interconnected domains: multilingual analysis and the attribution of anonymous correspondence. The former represents a response to the increasingly globalized nature of language, where traditional attribution methods find themselves stretched thin. Multilingualism, with its diverse tapestry of linguistic nuances, demands a fresh approach. Data compression, with its capacity to distill complex linguistic features, steps forth as a solution. Through this lens, we endeavor to bridge the gap between linguistic diversity and the universal pursuit of understanding authorial identity.

In tandem, the latter domain delves into the realm of anonymous correspondence, a frontier that has gained paramount importance in an age defined by digital communication. The ability to attribute authorship to anonymous texts carries profound implications, spanning from criminal investigations to cybersecurity. Traditional methods, reliant on linguistic markers associated with known authors, prove inadequate in this context. It is here that data

compression techniques promise to exert significant influence. By dissecting the underlying structure of anonymous texts, compression-based approaches unearth subtle yet distinctive patterns, revealing the true author's stylistic inclinations.

This interdisciplinary endeavor draws from the fields of linguistics, information theory, computer science, and forensic analysis, culminating in a convergence that underscores the symbiotic relationship between these domains. Through their synergy, we not only refine the methods of authorship attribution but also expand its horizons, accommodating the complexities of a globalized, digitally-driven world.

As we conclude this exploration, it is imperative to acknowledge that this journey is but a step towards a broader understanding of authorship and its implications. The potential unveiled here lays a foundation for future research, calling for continued innovation and refinement. The fusion of compression techniques with authorship attribution holds the promise of unlocking deeper insights into the intricate dance between language and author, ultimately enhancing our ability to discern and

appreciate the voices that shape our literary landscape. In the evolving tapestry of linguistic analysis, data compression stands as a powerful tool, offering a new lens through which we view the intricate art of authorship.

Standard machine learning methods (SVM, LR, NB, DT, RF, KNN), neural networks (CNN, LSTM, BiLSTM, RuBERT, MultiBERT, and fastText), and hybrids of these two types of architectures (CNN + LSTM, LSTM + CNN) can be used to determine the author of a piece of Russian prose.

We used two of our own datasets to train the models, which included both extensive works by Russian classics and brief comments from VK users.

The publication's methods yield results that are comparable to, and sometimes even superior to, those obtained by other researchers. Classical machine learning methods employed both TF-IDF and the text's constructed feature space for classification. In both instances, the performance of categorising brief texts is inferior to that of deep neural networks. The proportion of valid responses differs from 2% to 30% for all of the authors under

discussion. Because learning to write in a vector descriptive style requires the use of concise, monosyllabic claims and sentences, this is the case.

SVM, RF, and KNN are 97 percent as accurate as deep NNs for author identification on datasets containing 2, 5, and 10 works. The results demonstrate that there are ample text fragments (15,000 characters) for classification based on character n-gram frequencies and that the SVM is capable of handling a large feature space.

By selecting informative features with GA, the quality of SVM was enhanced. Throughout the entire selection procedure, it was necessary to maximise the target function, the SVM's accuracy. On the basis of their relative significance to the target function, subsets of 500, 400, 300, 200, 100, and 50 features were selected from the initial set of 1168 features. This technique can be used to eliminate irrelevant characteristics that impede classification and zero in on those that are truly useful. Vectors containing fifty features do not improve classification on either of the two datasets. Training on subsets of 50 or 100 characteristics in literary works increases the accuracy of two, five, or

ten authors, but has no effect on the accuracy of authors 20 and 50. After SVM training on the 400 features chosen by GA for all datasets, up to a 10% improvement in accuracy for both sets of text data is feasible. This reduces the strain on computational resources, eliminates duplication in feature sets, and accelerates learning throughout the entire training procedure.

Experiments and literature evaluations on the selection of informative features reveal numerous identifying characteristics:

Lack of awareness. When the author chooses a trait that is not readily influenced by awareness, the likelihood that it will be intentionally misrepresented decreases.

Lack of change. One author asserts that the trait's value is consistent within a limited range. These distinguishing characteristics enable readers to distinguish the authentic works of two or more authors with a similar style from the imitations.

Deep neural networks, in contrast to support vector machines (SVM), can detect implicit beneficial properties for classification on their own. CNN's training on a corpus of literary texts

produced an accuracy of approximately 98%. The accuracy of the SVM trained on the optimal subset of features exceeds this threshold across the entire battery of tests. Training periods for LSTMs are typically longer than those for SVM and other conventional ML techniques, but they can achieve exceptional accuracy for all datasets (particularly bidirectional LSTMs and their combinations with CNNs).

FastText is preferred because its accuracy is within 3% of the best possible result for all models and its learning rate is, on average, 51% faster than the deep NNs being evaluated.

Researchers should consider the total number of texts, the extent of the dataset's sample, and the categories of texts included when selecting an author identification method. When working with small texts and/or limited resources, ML techniques such as GAs and fastText are the best option. Since deep NNs can autonomously identify the author's concealed stylistic characteristics, they are better adapted for situations where text alteration or anonymization are possible.

In the future, researchers plan to conduct multiple experiments employing hybrid models comprising BERT and deep neural networks, as well as classifier ensembles constituted of the most effective individual models. To reduce the quantity of data collected, calibration curves and confidence metrics will be used. In addition to real-time feedback from social media users, a more extensive data set will be used to evaluate the outcomes. Fanfiction, or fiction created by fans based on canonical literary works, will be used alongside online and print resources. In addition, numerous studies are being conducted to address the problem of attributing authorship to open sets, which are used when members of a social network compose brief, fictitious pieces.

References

Al-Sarem, M., Saeed, F., Alsaeedi, A., Boulila, W. and Al-Hadhrami, T., 2020. Ensemble methods for instance-based arabic language authorship attribution. *IEEE Access*, *8*, pp.17331-17345.

Alhuqail, N.K., 2021. Author identification based on nlp. *European Journal of Computer Science and Information Technology*, *9*(1), pp.1-26.

Martín-del-Campo-Rodríguez, C., Sidorov, G. and Batyrshin, I., 2022. Unsupervised authorship attribution using feature selection and weighted cosine similarity. *Journal of Intelligent & Fuzzy Systems*, *42*(5), pp.4357-4367.

Lichtblau, D. and Stoean, C., 2023. Chaos game representation for authorship attribution. *Artificial Intelligence*, *317*, p.103858.

Škorić, M., Stanković, R., Ikonić Nešić, M., Byszuk, J. and Eder, M., 2022. Parallel stylometric document embeddings with deep learning based language models in literary authorship attribution. *Mathematics*, *10*(5), p.838.

Yadav, S., Rathore, S.S. and Chouhan, S.S., 2020. Authorship Identification Using Stylometry and Document Fingerprinting. In *Big Data Analytics: 8th International Conference, BDA 2020, Sonepat, India, December 15–18, 2020, Proceedings 8* (pp. 278-288). Springer International Publishing.

Custodio, J.E. and Paraboni, I., 2021. Stacked authorship attribution of digital texts. *Expert Systems with Applications*, *176*, p.114866.

Dong, K.D. and Nguyen, D.T., 2022, November. Vietnamese Text's Writing Styles Based Authorship Identification Model. In *International Conference on Future Data and Security Engineering* (pp. 347-361). Singapore: Springer Nature Singapore.

Abuhamad, M. (2020) ‘Towards Large-Scale and Robust Code Authorship Identification with Deep Feature Learning’. Available at: https://stars.library.ucf.edu/etd2020/597/ (Accessed: 22 October 2023).

Alshaher, H. (2021) *Studying the effects of feature scaling in machine learning*. PhD Thesis. North Carolina Agricultural and Technical State University. Available at: https://search.proquest.com/openview/739321feff86eff0c2eeeaee3ea8d3bb/1?pq-origsite=gscholar&cbl=18750&diss=y (Accessed: 22 October 2023).

Aykent, S. and Dozier, G. (2020a) ‘AARef: exploiting authorship identifiers of micro-messages with refinement blocks’, in *2020 19th IEEE International Conference on Machine Learning and Applications (ICMLA)*. IEEE, pp. 1044–1050. Available at: https://ieeexplore.ieee.org/abstract/document/9356191/ (Accessed: 22 October 2023).

Aykent, S. and Dozier, G. (2020b) ‘Author identification of micro-messages via multi-channel convolutional neural networks’, in *2020 IEEE International Conference on Systems, Man, and Cybernetics (SMC)*. IEEE, pp. 675–681. Available at: https://ieeexplore.ieee.org/abstract/document/9283214/ (Accessed: 22 October 2023).

Forstall, C.W. and Scheirer, W.J. (2019) ‘What is Quantitative Intertextuality?’, in Forstall, C. W. and Scheirer, W. J., *Quantitative Intertextuality*. Cham: Springer International Publishing, pp. 3–21. Available at: https://doi.org/10.1007/978-3-030-23415-7_1.

Grant, T. (2022) *The Idea of Progress in forensic authorship analysis*. Cambridge University Press. Available at: https://www.cambridge.org/core/elements/idea-of-progress-in-forensic-authorship-analysis/6A4F7668B4831CCD7DBF74DECA3EBA06 (Accessed: 22 October 2023).

Gujarati, A. (2019) ‘Authorship Attribution using Written and Read Documents’. Available at: https://dalspace.library.dal.ca/handle/10222/76215 (Accessed: 22 October 2023).

Heydon, G. (2019) *Researching forensic linguistics: Approaches and applications*. Routledge. Available at: https://books.google.com/books?hl=en&lr=&id=OX-YDwAAQBAJ&oi=fnd&pg=PT8&dq=Enhancing+Authorship+Attribution+Using+Data+Compression+Techniques:+Multilingual+Analysis+and+Anonymous+Mail+Attribution&ots=xTY26z8QCX&sig=koINt39NYTMmtZ5xIgGkQTRXHwU (Accessed: 22 October 2023).

Hu, X. *et al.* (2023) ‘TDRLM: Stylometric learning for authorship verification by Topic-Debiasing’, *Expert Systems with Applications*, 233, p. 120745.

Hu, Y. (2020) *A Study of Media Polarization with Authorship Attribution Methods*. PhD Thesis.

Purdue University Graduate School. Available at: https://hammer.purdue.edu/articles/thesis/A_Study_of_Media_Polarization_with_Authorship_Attribution_Methods/13350812/files/25742192.pdf (Accessed: 22 October 2023).

de Mensagens Curtas, A. de A. (no date) 'Antônio Carlos Theóphilo Costa Júnior'. Available at: https://repositorio.unicamp.br/Busca/Download?codigoArquivo=552963 (Accessed: 22 October 2023).

Ndaba, S. (2019) 'An empirical evaluation of writing style features in cross-topic and cross-genre documents in authorship identification'. Available at: https://ubrisa.ub.bw/handle/10311/2388 (Accessed: 22 October 2023).

Ryabko, B. and Savina, N. (2021) 'Using data compression to build a method for statistically verified attribution of literary texts', *Entropy*, 23(10), p. 1302.

Schaetti, N. (2020) *An Empirical Comparison of Recurrent Neural Network Models on Authorship Analysis Tasks*. PhD Thesis. PhD thesis, Université de Neuchâtel. Available at: https://www.researchgate.net/profile/Nils-Schaetti/publication/350399821_An_Empirical_Comparison_of_Recurrent_Neural_Network_Models_on_Authorship_Analysis_Tasks/links/605d9b51299bf173676c3907/An-Empirical-Comparison-of-Recurrent-Neural-Network-Models-on-Authorship-Analysis-Tasks.pdf (Accessed: 22 October 2023).

Shao, S. *et al.* (2019) 'One-class classification with deep autoencoder neural networks for author verification in internet relay chat', in *2019 IEEE/ACS 16th International Conference on Computer Systems and Applications (AICCSA)*. IEEE, pp. 1–8. Available at: https://ieeexplore.ieee.org/abstract/document/9035309/ (Accessed: 22 October 2023).

Wahdan, A., Al-Emran, M. and Shaalan, K. (2023) 'A systematic review of Arabic text

classification: areas, applications, and future directions', *Soft Computing* [Preprint]. Available at: https://doi.org/10.1007/s00500-023-08384-6.

Wang, H., Juola, P. and Riddell, A. (2022) 'Reproduction and Replication of an Adversarial Stylometry Experiment'. arXiv. Available at: http://arxiv.org/abs/2208.07395 (Accessed: 22 October 2023).