Authorship Analysis of Arabic Long Texts Through Style Comparisons

Qutaiba Olayyan
Electrical and Computer Engineering Department
Birzeit University, Palestine
1190760@student.birzeit.edu

Eyab Ghifari
Electrical and Computer Engineering Department
Birzeit University, Palestine
1190999@student.birzeit.edu

Prof. Adnan Yahya
Electrical and Computer Engineering Department
Birzeit University, Palestine
yahya@birzeit.edu

***Abstract*—This research paper discusses the implementation of an authorship attribution project aimed at generating writer-specific fingerprints indicative of their writing styles. The study focuses on long Arabic texts written in classical Arabic, sourced from newspapers and news channels. Personal files were created for each writer to capture their distinct writing style, enabling differentiation through machine learning models. The validity of this approach was verified through a series of experiments utilizing various methodologies and algorithms, resulting in a classification accuracy of approximately 84%. Additionally, a model for authorship verification was developed, allowing the calculation of text similarity to determine if two texts were authored by the same individual, even if they were not part of the original group of writers. Finally, a simple user interface was created to facilitate the exploration of the aforementioned processes. This research contributes to the field of authorship attribution in classical Arabic texts and offers valuable insights for further studies in computational linguistics and machine learning.**

I. Introduction

In an era defined by the proliferation of websites, blogs, and social media platforms, the digital realm has evolved into an expansive repository of data, estimated at approximately 80 zettabytes [1] by the end of 2021. The accessibility and enormity of this data realm have inadvertently facilitated intellectual and literary theft, with numerous individuals exposed to the risk of plagiarism. Consequently, there has been a growing urgency for methodologies capable of accurately attributing a text to its author, leading to a renewed interest in the concept of authorship attribution.

Authorship attribution, the discipline focused on deducing the originator of written content, is critical across various sectors including forensic linguistics and literary studies. The procedure involves pinpointing and examining subtle linguistic details and stylistic markers within a work. Often, the frequency analysis of function words and Part-Of-Speech N-grams is used. Nevertheless, these methods somewhat lean on the disputed bag-of-words concept that sees the text as a collection of independent, unrelated words. The refinement of these techniques will investigate stylistic markers other than the bag-of-words assumption, namely sequence-based patterns of function words and Part-Of-Speech tags. Upon examination, it's found that the direct application of function word and Part-Of-Speech n-gram frequencies tend to be more effective than their sequence-based patterns, providing a richer understanding of the complexities involved in authorship attribution. [2]

Our research, titled "Authorship Analysis through Style Comparisons," builds upon this concept and seeks to innovate further by developing a mechanism for comparing the stylistic attributes of unknown documents with the writings of known authors. This mechanism quantifies stylistic similarity or dissimilarity between two texts and extends its utility to author verification. This involves measuring the stylistic distance between a piece of text and a set of known authors' works to ascertain common authorship.

The focus of our study is long-form Modern Standard Arabic text, as extracted from magazines and newspapers. The choice of Modern Standard Arabic was influenced by its standing as the fourth most commonly used language on the [3], coupled with a relative dearth of comparative research on the language. Analysis of Arabic texts presents unique challenges due to the complexity of its linguistic rules and distinctive features such as diacritical marks, inflections, and elongations, which are absent in languages such as English.

Throughout our research, we have utilized an assortment of efficient text-processing tools, alongside machine learning classifiers. The performance of these classifiers has been assessed based on their accuracy. The culmination of our data collection and the application of authorship attribution techniques resulted in the creation of a novel model for authorship verification. Finally, a streamlined user interface has been developed to facilitate the use and discussion of our results in subsequent presentations.

Our research holds promising implications for both computational linguistics and digital forensics. Beyond its potential for combating plagiarism and literary theft, it offers intriguing insights into the unique 'linguistic fingerprints' inherent in each author's style. This endeavor aims to revolutionize our understanding of authorship and redefine how we verify and attribute written content in an increasingly digitized world.

II. METHODOLOGY

In the section delineated as Methodology, the procedural underpinnings of the conducted research are systematically outlined. It bifurcates into two imperative subsections—Data Collection and Preprocessing of the Data. The Data Collection subsection sheds light on the methodical approach employed for the collation of pertinent textual data, elucidating the criteria for source selection and the strategies for data acquisition. Following this, the Preprocessing of the Data subsection expounds upon the assortment of techniques and tools leveraged to purify, transform, and normalize the raw data, thereby rendering it suitable for ensuing analysis. The intention behind this exposition is to proffer a comprehensive understanding of the concrete steps operationalized throughout our research, tracing the journey from raw data procurement to its refinement for methodical examination.

A. *Data collection*

Data for this project was collected from multiple sources. The primary data source was the University of Bizerte's Fada website [1], from which we sourced data on articles for a research project titled "An Arabic Author Attribution Dataset" [4]. This dataset encompasses works from seven distinct authors, with ten Arabic articles per author derived from various electronic sources. Accompanying each author's set of articles is a guide containing the textual content of each of the ten articles as well as the associated metadata, all of which was retrieved from the "Taamolat" website [2].

In addition, we utilized data from another research project, "A Dataset for Arabic Author Identification" [5], which comprises nine authors with ten articles each. This set of data also included metadata and statistics for each article, all of which were procured from both the Alaraby.co.uk (The New Arab) and Al Jazeera websites [3].

With the inclusion of these additional sources, our dataset expanded to incorporate the works of fifteen authors, each contributing ten articles. This resulted in a rich collection of approximately 150 articles of various types. By synthesizing these various sources, our research draws from a broad and representative cross-section of contemporary Arabic literature.

To provide a graphical representation of our data, we included a bar graph in Figure 1. The graph depicts the correlation between the number of words and authors, clearly illustrating the textual contribution of each author to our comprehensive dataset. This visual aid enhances our understanding of the data and aids in interpreting the statistical distribution of words across different authors.

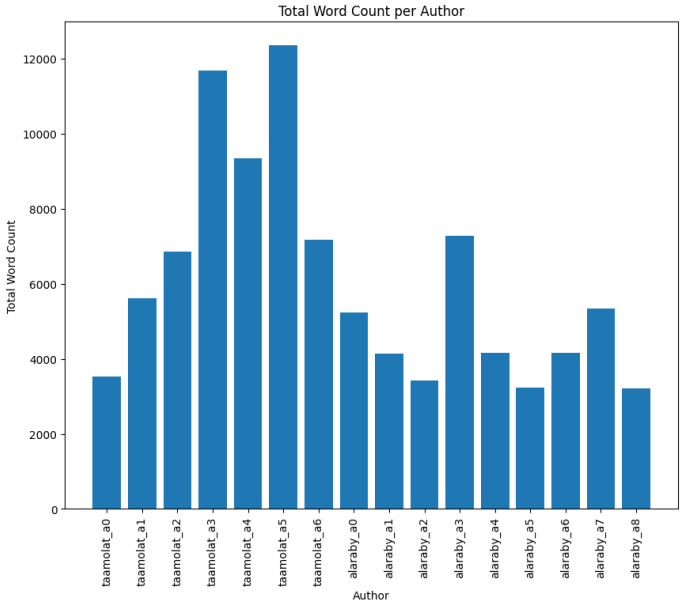

Fig. 1. Number of Words per Author

B. *Preprocessing*

The Preprocessing of the Data section elucidates the sequential methodology employed to prepare the raw data for in-depth analysis. As an indispensable phase in our research, preprocessing encompasses a variety of steps, each designed to refine, streamline, and enhance the quality of the data. The process starts with the normalization of the text, followed by stemming, tokenization, and ends with the removal of Arabic stop words. Each stage serves a critical function in our analytical pipeline, collectively contributing to the efficiency and accuracy of our research. Detailed descriptions of each step follow in the subsequent paragraphs, providing a comprehensive understanding of our preprocessing techniques. The forthcoming elaboration comprises a sequential delineation of the key steps incorporated into our data preprocessing regimen. Each step is fundamental in transforming the raw data into a refined form, which is conducive to a more effective and accurate analysis. Herein, these integral processes are described in detail:

- **Normalization:** The normalization process for Arabic text consists of numerous cleaning and standardization procedures, preparing the text for following data analysis or processing tasks. These procedures include the removal of non-Arabic characters and spaces, the elimination of Arabic diacritical marks, and the standardization of certain Arabic letters. Moreover, it involves getting rid of special characters, English words, HTML tags, hyperlinks, Twitter metadata, and elongated Arabic letters.
  Also, the process transforms text elements to a unified canonical form. Specifically, in our case, all variations of hamza "آ أ ء إ" become an alef "ا". In the same manner,

[1]Bizerte's Fada website: https://fada.birzeit.edu/
[2]Taamolat: http://www.taamolat.com/
[3]Al-Jazeera: https://www.aljazeera.net/blogs/

Taa marboota "ة ، ـة" is converted to "ه ، ـه".
Two Python libraries, PyArabic [6] and Tnkeeh [7] , support this process. PyArabic is employed for tasks such as eliminating diacritical marks and normalizing Arabic letters like Alef and Teh. The Tnkeeh library is used for additional cleaning procedures, offering an array of features specifically designed for Arabic text cleaning and preprocessing.
The end result of this normalization process is a clean, standardized Arabic text, devoid of irrelevant elements. This significantly enhances the quality of data for further processing. The combined utilization of PyArabic and Tnkeeh libraries in this operation underscores their value in maintaining the integrity of Arabic text data.

- **Stemming:** The process of stemming is an integral part of text preprocessing, particularly for languages such as Arabic that have a rich morphological structure. Stemming reduces words to their root form by removing affixes, which can significantly simplify subsequent text analysis tasks. In the implemented function, this process is facilitated by the ISRI stemmer, a part of the Natural Language Toolkit (NLTK) library [8]. The ISRI stemmer is specifically designed for Arabic language processing, considering the unique features and complexities of Arabic morphology. By applying the ISRI stemmer, words are effectively truncated to their root form. For instance, the word 'المدرسة' is reduced to its root 'درس', demonstrating the efficacy of the stemming process. The use of the ISRI stemmer from the NLTK library underscores its utility in Arabic language processing tasks, particularly in accurately identifying word roots.
- **Tokenization:** The tokenization process divides a sequence of text into individual words or terms, termed as "tokens". This process is particularly crucial in the context of text analysis as it allows for a detailed, individual examination of each word in the text.
The tokenization function in the given context is implemented using the PyArabic library [6], a Python library tailored for Arabic text processing. The 'tokenize' function from this library is specifically employed to segment the input text into separate tokens, effectively breaking down the text into individual words. This granular view of the text data enables precise and in-depth analysis of the text. The use of the PyArabic library in this process highlights its significance in Arabic text analysis and processing.
- **Arabic Stop Words:** The process outlined involves creating a comprehensive list of Arabic stop words, which are typically common words that are removed during text preprocessing due to their low semantic value. The stop words are sourced from a file named 'stop_words.txt', retrieved from the ArabicStopWords [9] repository on GitHub, renowned for hosting the largest collection of Arabic stop words. Each line from this file is read, stripped of leading and trailing spaces, and added to a list. To ensure the uniqueness of each stop word, the list is converted into a set and then back into a list. Additionally, the list is supplemented with the Arabic stop words from the Natural Language Toolkit (NLTK) [8], a widely used Python library for natural language processing. The final result is a comprehensive and unique list of Arabic stop words, integral to text preprocessing tasks, enabling effective filtering of words that contribute minimal meaningful information for text analysis.

III. Implementation

In our pursuit to construct distinctive markers tied to an author's individual writing styles, particularly focusing on comprehensive texts in classical Arabic, we applied two specialized feature extraction techniques in natural language processing (NLP):

- **Term Frequency-Inverse Document Frequency (TF-IDF):**TF-IDF is a widely-used technique in natural language processing (NLP) that assigns a weight to each word in a document, indicating its significance within the document and the broader corpus. The TF-IDF score of a term within a document is calculated as the product of its term frequency (TF) and its inverse document frequency (IDF). The term frequency is the number of times the term appears in the document, while the inverse document frequency is a measure of how much information the term provides, that is, whether the term is common or rare across all documents.
The TF-IDF score is given by:

$$\text{TF-IDF} = \text{TF} \times \text{IDF}$$

The IDF is computed as:

$$\text{IDF} = \log\left(\frac{N}{DF}\right)$$

where N is the total number of documents in the corpus, and DF is the document frequency of the term, which is the number of documents that contain the term.
To create a TF-IDF vector for a document, we first need to create a document-term matrix (DTM) that represents the frequency of each term in each document of the corpus. Then, we calculate the TF-IDF score for each term in each document. However, since the magnitude of the TF-IDF vector could affect the computation of similarity between documents, we often normalize the TF-IDF vector by its Euclidean norm:

$$||v|| = \sqrt{\sum_i v_i^2}$$

where v is the TF-IDF vector for the document, and v_i is the TF-IDF score for the i-th term in the document.
The normalized TF-IDF vector is given by:

$$v_{\text{norm}} = \frac{v}{||v||}$$

where v_{norm} is the normalized TF-IDF vector for the document. This normalization ensures that the length of the vector is 1, facilitating the comparison of documents using cosine similarity.
In our implementation, we used the sklearn library [10] to generate the TF-IDF vectors for our documents. This approach enabled us to capture the unique vocabulary usage of the authors.

- **Word Embeddings:** Word embeddings represent words in a high-dimensional space, where semantically similar words have close representations. This property is crucial for capturing the semantic context of words. To generate these embeddings, our methodology employs the FastText model from the gensim library. FastText, an extension of the Word2Vec model, incorporates subword information, making it particularly effective for languages like Arabic, characterized by a high degree of morphological complexity. The model performs exceptionally well with small datasets and excels at handling rare or out-of-vocabulary words.
For implementing this technique, we designed a class in Python that initializes specific parameters for the FastText model. The vector size is set to 100, providing a balance between computational complexity and the depth of the semantic and syntactic relationships captured in the vectors. The window size is defined as 10, allowing the model to consider a broad context when learning the embeddings. A minimum count threshold of 1 ensures that even rare words are included during training. The use of four worker threads, facilitated by the multiprocessing capabilities of Python, accelerates the training process.
In the application of this class, the FastText model is trained on a given corpus, with each sentence in the corpus transformed into a feature vector. The feature vector for a sentence is computed as the mean of the word vectors for the words in the sentence, or a zero vector if the sentence contains no words present in the model's vocabulary. The class also includes methods for retrieving the feature names and author names. These chosen parameters for the FastText model, informed by both theoretical considerations and practical constraints, effectively capture the unique vocabulary usage and semantic context of words in the texts. This approach, enabled by the power of the Python libraries gensim [11] and numpy [12], is instrumental in creating writer-specific fingerprints, a key objective of our research.
The integration of TF-IDF and FastText allowed us to capture both the unique vocabulary usage and the semantic context of words in the texts. This combination was instrumental in creating writer-specific fingerprints and in distinguishing authors using machine learning models. Our methodology provides a substantial contribution to the field of authorship attribution in classical Arabic texts and serves as a valuable reference for future research in computational linguistics and machine learning.

IV. Evaluation

Various machine-learning models were applied to the feature vectors produced by the TF-IDF and FastText techniques. The principal objective was to distinguish between authors using their writing styles. The models were evaluated on precision, recall, F1 score, and accuracy, which collectively offer a comprehensive assessment of a model's performance.

These models were implemented using the sklearn library [10] in Python. They include Logistic Regression [13], an often effective statistical model for binary and multiclass classification tasks; Random Forest [14], an ensemble learning method known for its flexibility and performance without requiring extensive parameter tuning; Support Vector Machines (SVM) [15], which is effective in high-dimensional spaces and is particularly suitable for text classification problems; K-Nearest Neighbors (KNN) [16], one of the simplest of all machine learning algorithms where the function is only approximated locally and all computation is deferred until function evaluation; and Multi-Layer Perceptron (MLP) [17], a class of feedforward artificial neural network utilizing backpropagation for training.

TABLE I
Model Evaluation Metrics

Model	Precision	Recall	F1 Score	Accuracy
Logistic Regression	0.7839	0.5731	0.5900	0.6520
Random Forest	0.6656	0.4586	0.4610	0.5329
SVM	0.8869	0.7649	0.7993	0.7931
KNN	0.7834	0.7266	0.7367	0.7335
MLP	0.9001	0.8197	0.8471	0.8339

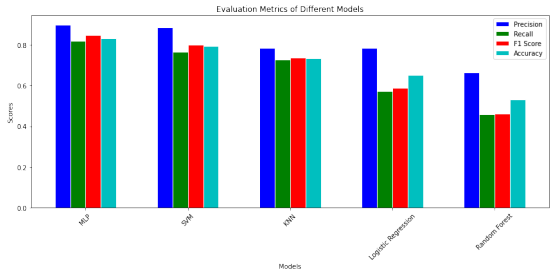

Fig. 2. Comparison of Precision, Recall, F1 Score, and Accuracy across Different Machine Learning Models.

Performance results, as tabulated above and visualized in Figure 2, demonstrate that the Multi-Layer Perceptron (MLP) and Support Vector Machines (SVM) models outperformed others in terms of precision, recall, F1 score, and accuracy. The findings underscore the effectiveness of these models in capturing distinct authorial styles, the primary objective of the research, and highlight the importance of model selection in machine learning tasks. Depending on the specific characteristics of the text and the problem at hand, the choice of the most suitable model may vary, emphasizing the importance of experimenting with different models to find the best performer for the specific task.

V. Conclusions and Possible Improvements

This research paper presents an authorship attribution project focused on analyzing writing styles in long Arabic texts written in classical Arabic. The study utilizes machine learning models to generate writer-specific fingerprints and differentiate between authors. The approach was validated through experiments, achieving a classification accuracy of approximately 84%. The research also developed a model for authorship verification and created a user-friendly interface. The findings contribute to the field of authorship attribution in classical Arabic texts and have implications for computational linguistics and digital forensics. The methodology involved data collection, preprocessing, and the use of TF-IDF and Word Embeddings techniques. Evaluation results showed that MLP and SVM models performed best in distinguishing authors. Overall, the research offers insights into authorship analysis and helps combat plagiarism in a digitized world.

Despite the significant strides we have made in authorship attribution, there remains room for further enhancement in the accuracy of our model. An exciting direction for future improvements lies in the incorporation of a pair-wise feature extraction methodology. In this innovative approach, a pair of texts forms the unit of analysis, with a binary label indicating whether the two texts originate from the same author.

There are several strategies for implementing this pair-wise methodology:

1) Both texts could be merged before preprocessing. The subsequent stages of feature extraction and model training would then apply to the combined text, treating it as a single unit.
2) Alternatively, preprocessing could be applied to each text separately before the texts are merged. Feature extraction and model training would then apply to this combined, preprocessed text. This approach allows each text to undergo individual preprocessing before they are considered collectively.
3) In the third approach, preprocessing and feature extraction would apply to each text separately. The resulting feature vectors would then be combined and fed into the machine learning models. This approach maintains the distinctiveness of each text through the feature extraction stage, potentially preserving more unique information about the texts.

These improvements hold the potential to elevate the accuracy of our model by capturing more nuanced information about the differences and similarities between authors. However, each method involves trade-offs and would require thorough testing and validation to ascertain its effectiveness. As we move forward, exploring these strategies could lead to the development of more sophisticated and accurate models for authorship attribution.

References

[1] O. Djuraskovic, *30+ big data statistics (2023) - amount of data generated in the world*, Apr. 2023. [Online]. Available: https://firstsiteguide.com/big-data-stats/.

[2] M. A. Boukhaled and J.-G. Ganascia, "Stylistic features based on sequential rule mining for authorship attribution," in *Cognitive Approach to Natural Language Processing*, 2017, pp. 159–175. [Online]. Available: https : / / www . sciencedirect . com / topics / computer - science/authorship-attribution.

[3] A. Khilji, *10 most common languages used on the internet for 2023 — mars translation*, https : / / www . marstranslation.com/blog/10-most-common-languages-used-on-the-internet, Feb. 2023.

[4] M. Hijja, A. Yahya, and A. Yahya, "An arabic author attribution dataset," *Fada: Birzeit University Repository*, Jun. 2019. [Online]. Available: http://hdl.handle.net/20.500.11889/6022.

[5] B. Anini, Y. Anini, and A. Yahya, "A dataset for arabic author identification," *Fada: Birzeit University Repository*, Jun. 2019. [Online]. Available: http://hdl.handle.net/20.500.11889/6023.

[6] T. Zerrouki, *Pyarabic, an arabic language library for python*, 2010. [Online]. Available: https://pypi.python.org/pypi/pyarabic.

[7] Z. Alyafeai and M. Saeed, *Tkseem: A preprocessing library for arabic.* https://github.com/ARBML/tnkeeh, 2020.

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

[9] M. T. Alrefaie, N. Hussein, and T. Bazine, *Arabic-stop-words*, GitHub repository, Jan. 2017. [Online]. Available: https : / / github . com / mohataher / arabic - stop-words.

[10] F. Pedregosa, G. Varoquaux, A. Gramfort, *et al.*, "Scikit-learn: Machine learning in Python," *Journal of Machine Learning Research*, vol. 12, pp. 2825–2830, 2011.

[11] R. Rehurek and P. Sojka, "Gensim–python framework for vector space modelling," *NLP Centre, Faculty of Informatics, Masaryk University, Brno, Czech Republic*, vol. 3, no. 2, 2011.

[12] C. R. Harris, K. J. Millman, S. J. van der Walt, *et al.*, "Array programming with NumPy," *Nature*, vol. 585, no. 7825, pp. 357–362, Sep. 2020. DOI: 10 . 1038 / s41586 - 020 - 2649 - 2. [Online]. Available: https : / / doi . org/10.1038/s41586-020-2649-2.

[13] D. R. Cox, "The regression analysis of binary sequences," *Journal of the Royal Statistical Society: Series B (Methodological)*, vol. 20, no. 2, pp. 215–232, 1958.

[14] T. K. Ho, "Random decision forests," in *Proceedings of 3rd international conference on document analysis and recognition*, IEEE, vol. 1, 1995, pp. 278–282.

[15] C. Cortes and V. Vapnik, "Support-vector networks," *Machine learning*, vol. 20, no. 3, pp. 273–297, 1995.

[16] T. Cover and P. Hart, "Nearest neighbor pattern classification," *IEEE Transactions on Information Theory*, vol. 13, no. 1, pp. 21–27, 1967.

[17] S. Haykin, *Neural networks: a comprehensive foundation*. Prentice Hall PTR, 1994.