BRAUDE
College of Engineering, Karmiel

Software Engineering Department

Capstone Project Phase A – 61998

Analysis of Medieval Arabic Creations

25-1-R-19

https://github.com/YosraDaso/CapstoneProject.git

Yosra Fhamne	204138879	yosra.fhamne@e.braude.ac.il
Omar Saleh	323004895	omar.saleh@e.braude.ac.il

Supervisors:

Prof. Zeev Volkovich

Dr. Renata Avros

Abstract

Authorship attribution is a challenging task in text classification, particularly in the context of medieval Arabic literature, due to the language's stylistic diversity, morphological richness, and the lack of standardized orthography. This project focuses on distinguishing authentic works of Abu Hamid Al-Ghazali from pseudo-authentic texts using advanced machine learning techniques. A computational framework is proposed, integrating Siamese Networks, AraBERT embeddings, and signal representation to analyze and capture distinct stylistic markers of authorship. The framework includes a systematic preprocessing pipeline tailored for Arabic texts and applies clustering methods to classify texts into authentic and pseudo-authentic groups. By leveraging techniques such as CNN and BiLSTM in the core model, the approach is designed to enhance accuracy and scalability in authorship attribution. This research contributes to the fields of computational linguistics and historical text analysis, offering a robust solution to the complexities of Arabic manuscript authenticity.

Keywords: Authorship Attribution, Arabic Medieval Literature, Machine Learning, Siamese Networks, AraBERT, Stylistic Analysis.

Table of Contents

1. Introduction

This project focuses on the authorship attribution of medieval Arabic texts, with particular attention to the works attributed to Al-Ghazali. Authorship attribution in Arabic texts presents unique challenges due to the language's morphological richness, stylistic variability, and the lack of standardized orthography in historical manuscripts. These factors complicate efforts to distinguish genuine works from those falsely attributed to an author, particularly in the context of medieval texts. Traditional methods of manual stylistic analysis often lack the scalability and objectivity needed to handle such complexities [2,12]. The study aims to define the challenges in verifying authorship and propose a computational framework using advanced machine learning techniques. While the project centers on Al-Ghazali's works, the proposed approach is designed to be adaptable to various authorship attribution tasks, contributing to the broader understanding of Arabic texts and their intellectual heritage.

1.1 Authorship Attribution

Authorship attribution is the process of identifying the author of a text by analyzing its linguistic and stylistic features. This field has gained significant attention across various domains, including literary studies, historical research, and forensic linguistics, due to its ability to verify textual authenticity and resolve questions of disputed authorship. It plays a crucial role in uncovering intellectual contributions, preserving cultural heritage, and enhancing the accuracy of historical records [10]. Traditionally, authorship attribution relied on manual stylistic analysis, which involved examining features such as word choice, sentence structure, and thematic patterns. However, these methods were often subjective and lacked scalability when applied to large datasets or texts with stylistic variability. The limitations of manual approaches necessitated the development of computational methods to systematically and objectively analyze texts at scale [12]. Modern approaches use statistical and algorithmic techniques to capture subtle patterns in text. Machine learning models, such as Support Vector Machines (SVM) and ensemble classifiers, have been widely adopted to analyze linguistic features and distinguish between authors. These models excel at identifying complex patterns and can handle the variability inherent in large and diverse datasets, making authorship attribution more robust and adaptable to diverse textual sources [2]. By leveraging computational techniques, authorship attribution not only aids in identifying the authors of disputed texts but also contributes to broader applications, such as detecting plagiarism, analyzing historical documents, and supporting forensic investigations [10,12]. These developments form the foundation for addressing the complexities of authorship attribution in both modern and historical contexts.

1.2 Abu Hamid Al-Ghazali

Abu Hamid Al-Ghazali (1058–1111) is a renowned philosopher, theologian, and jurist of the Islamic Golden Age. Born in Tus, in present-day Iran, Al-Ghazali profoundly influenced Islamic thought through his works, including *Ihya' Ulum al-Din* (*The Revival of the Religious Sciences*) and *Tahafut al-Falasifa* (*The Incoherence of the Philosophers*). These writings addressed key theological, ethical, and philosophical questions, and his efforts to harmonize Sufism with Sunni orthodoxy established him as one of the most influential figures in Islamic history. Al-Ghazali's works continue to shape contemporary Islamic scholarship [25]. Several texts attributed to Al-Ghazali are subjects of scholarly debate, with questions regarding their authenticity arising from stylistic and thematic inconsistencies [22]. Verifying the authenticity of these works is critical for accurately understanding his contributions to Islamic thought and for situating his writings within the broader intellectual and cultural developments of his time. This project aims to address these challenges by examining Al-Ghazali’s works, contributing to a systematic understanding of authorship attribution in medieval Arabic texts.

1.3 Book's Structure

A short introduction to authorship attribution, the challenges in analyzing medieval Arabic texts, and the proposed computational framework are presented in Chapter 1. Background and related work, including key concepts and prior studies in the field, are detailed in Chapter 2. Chapter 3 outlines the expected achievements and success criteria of the project. Chapter 4 explains the research process and provides an in-depth description of the proposed approach, including preprocessing, embedding generation, and model architecture. The evaluation plan is discussed in Chapter 5, covering the testing strategies and criteria for validating the framework.

2. Background and Related Work

This section presents the essential background and related work necessary to understand the concepts, methodologies, and techniques underpinning the proposed solution.

2.1 Related Work

Authorship attribution has been a prominent area of research, with several studies exploring the application of BERT-based architectures to this task. Alzahrani and Al-Yahya [3], applied BERT to Arabic text authorship attribution, demonstrating its ability to capture nuanced linguistic features. Similarly, Bauersfeld [5], highlighted BERT's proficiency in identifying distinct writing patterns across diverse genres. Earlier, Fabien [9] demonstrated the model's capacity to isolate precise authorial markers, achieving high accuracy. More recently, Huang and Iwaihara [11], and Silva[19], reinforced BERT's effectiveness in detecting subtle stylometric features that distinguish individual authors, further validating its robustness in analyzing complex linguistic patterns. Advances in deep learning have also propelled research into historical text analysis. Reisi and Farimani [17], introduced an advanced convolutional neural network (CNN) model integrated with a self-attention mechanism across four architectural components. Their work focused on the classification of Persian literary texts from the fourth to seventh centuries, highlighting the model's capability to handle intricate historical manuscripts effectively. Building on these innovations, Silva [19], proposed the GANBERT model, which integrates BERT with Generative Adversarial Networks (GANs). This model, enhanced with advanced sampling techniques, was tailored to analyze 19th-century literary works. By achieving 88% accuracy in authorship attribution tasks, GANBERT represents a significant milestone in computational literary analysis, combining the strengths of BERT and GANs to address challenges in the field effectively.

2.1.1 Paper's Contribution

As discussed in the introduction, authorship attribution in Arabic texts presents unique challenges due to stylistic variability, morphological richness, and a lack of standardized orthography. Existing methods can be broadly categorized into intrinsic and extrinsic approaches. Intrinsic methods focus on linguistic features within a single text, while extrinsic approaches compare texts against external datasets to identify patterns. Among these, the Impostors' method has gained attention for its robustness in scenarios with limited labeled data, introducing linguistic variability to simulate real-world conditions [22]. Authorship attribution methodologies have primarily dealt with Englishlanguage texts while paying less attention to other languages, including ancient ones. These approaches often involve modifications of word embeddings to capture linguistic patterns more effectively (e.g.,[4]). In the context of Arabic texts, previous studies have demonstrated the potential of deep learning techniques in analyzing complex linguistic and stylistic features. For instance, short-pattern analysis techniques have effectively captured authorial markers in Arabic texts, particularly those attributed to Al-Ghazali [22]. Another study employed deep learning models for sentiment analysis on medieval Arabic documents, highlighting the applicability of computational methods to historical text classification [4]. Building on these foundations, this project introduces a computational framework that integrates advanced machine learning techniques, such as Siamese Networks, transformers and signal representation, tailored explicitly for authorship attribution in Arabic texts. By broadening the application of the Impostors' method to include a wider range of unknown impostors, this framework enhances adaptability and scalability for broader authorship attribution tasks. The study in [22] involved using pre-selected impostors, such as verified Al-Ghazali texts and pseudo-Ghazali works, focusing on a classification task with known data points. In contrast, this study introduces diverse impostors unknown to the author, simulating real-world variability. This approach aims to improve scalability and robustness, providing a comprehensive framework for addressing authorship challenges in Arabic texts.

2.2 Background

2.2.1 Word Embedding

Word embedding is a mathematical representation of words in a continuous vector space where words with similar meanings are positioned closer together. This spatial arrangement captures both semantic and syntactic relationships, enabling models to interpret language in a structured and meaningful way. Word embedding serves as the foundation for many computational applications, including text classification, information retrieval, sentiment analysis, and machine translation, by transforming textual data into numerical forms that deep learning models can process efficiently. Models such as Word2Vec (by Google) [14], and GloVe (by Stanford) [16], have pioneered this area by focusing on distributional semantics, while contextualized models like BERT and GPT have further advanced the field by incorporating surrounding context for more nuanced representations.

The importance of word embedding in deep learning lies in its ability to reduce the dimensionality of language data while preserving essential linguistic relationships, thereby enabling models to identify patterns and connections in text. This step is crucial for tasks that require understanding the meaning and context of words, sentences, or entire documents. By serving as the initial layer of input for deep learning architectures, word embedding forms the foundation upon which models like transformers build their advanced capabilities [20, 27].

2.2.2 Bidirectional Encoder Representations from Transformers (BERT)

BERT is a deep learning model designed to process natural language with exceptional contextual understanding. Built on the Transformer architecture, BERT distinguishes itself through bidirectional training, allowing it to predict text that comes both before and after a target word in a sentence. This approach enables BERT to analyze the broader context of words within sentences, addressing the limitations of earlier unidirectional models that process text in a single direction. These capabilities make BERT particularly useful for tasks such as question answering, text classification, and language inference [7].

It is important to explore the foundational concepts and mechanisms underpinning its architecture and operation to fully understand how this model functions. The following sections will examine these elements in detail to provide a comprehensive understanding of BERT's design and applications.

- Encoder

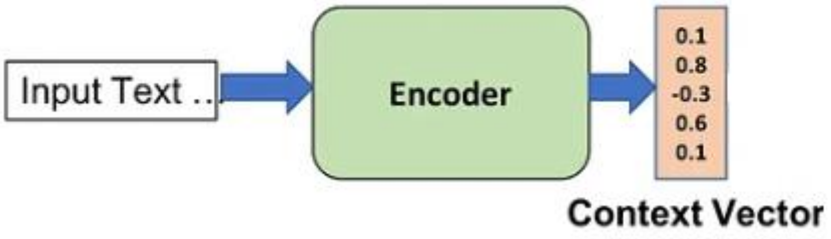

Figure 1. Overview of the Encoder. Adapted from *Exploring the Power of Encoder-Decoder Models: Pros, Cons, and Applications*.

Encoders are neural network components designed to handle sequence-to-sequence (Seq2Seq) tasks that involve transforming an input sequence into an intermediate representation, which is then decoded into the desired output sequence [26]. They are particularly valuable for tasks such as machine translation, text summarization, and speech recognition. Encoders are commonly implemented using recurrent neural networks (RNNs) or long short-term memory (LSTM) networks, with further details about these architectures provided later in this document. As illustrated in Figure 1, the encoder processes the input sequence and compresses it into a fixed-length vector, known as the context vector, which serves as an abstract representation of the input data, capturing its essential features and enabling efficient summarization. By focusing solely on encoding, the encoder establishes a robust foundation for analyzing and transforming sequential data effectively [29].

- Transformers

First introduced by Vaswani [21], transformers are a neural network architecture designed for sequence-to-sequence tasks, relying entirely on attention mechanisms rather than recurrent or convolutional layers to process data. Unlike RNNs and LSTMs, which process inputs sequentially, transformers process the entire input sequence simultaneously, allowing for efficient parallelization and the handling of extremely large inputs. This approach eliminates the limitations of sequential models and makes transformers particularly effective for tasks such as language modeling and translation. Additionally, transformers leverage the concept of attention, a mechanism that enables the model to focus on the most relevant parts of the input sequence, further enhancing their ability to capture complex dependencies in data. For the purpose of our study, we will focus exclusively on the encoder component of the Transformer architecture, as it aligns with our analysis objectives.

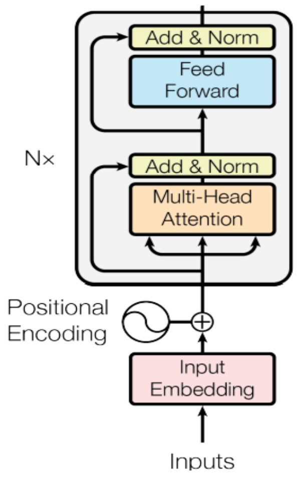

Figure 2. Overview of the Encoder Architecture in a Transformer Model. Adapted from *Attention Is All You Need*.

As depicted in Figure 2, the encoder processes input data through a series of interconnected components, each playing a critical role in generating meaningful representations. In the following sections, we will explore each encoder component in detail.

- Input embedding:

Input embedding is the initial step in the encoder, where the input tokens (words) are transformed into dense vector representations. As described in Section 2.2.1.

- Positional Encoding:

Positional encoding is a crucial component in Transformer architectures, designed to capture the positional information of words within a sequence. Unlike recurrent models, Transformers process input data in parallel, which means they lack inherent knowledge of word order. This limitation is addressed by introducing positional encoding to ensure that the sequence structure is preserved, which is vital for understanding context and relationships between words. By embedding positional information directly into the word embeddings, the model can discern not only the content of the words but also their relative order within the sequence [21].
Positional encoding is mathematically represented using sine and cosine functions, which assign unique positional values to each word. These values are calculated using the following equations:

$$PE_{(pos,2i)} = \sin\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$$

$$PE_{(pos,2i+1)} = \cos\left(\frac{pos}{1000^{\frac{2i}{d_{model}}}}\right)$$

Where pos is the position of the word in the sentence, i is the index of the dimension within the embedding vector, and d_{model} is the total dimensionality of the embedding vector. These functions ensure that each position is assigned a unique encoding. The sine function is applied to even dimensions (2i), while the cosine function is applied to odd dimensions (2i+1). The term $1000^{\frac{2i}{d_{model}}}$ scales the positional values, enabling the encoding to represent relationships between words across both short and long distances. This encoding is then added to the word embeddings, allowing the Transformer model to process input sequences with an understanding of both word content and position. This encoding is then added to the word embeddings, allowing the Transformer model to process input sequences with an understanding of both word content and position.

- Multi-Head Attention:

The multi-head attention mechanism is a core component of Transformer architectures, designed to enhance the model's ability to focus on distinct parts of the input sequence simultaneously. To understand multi-head attention, it is essential to first define the concept of attention and its variants. Attention is a mechanism that enables the model to weigh the importance of each input word in relation to others in a sequence, allowing it to focus on the most relevant parts when making predictions. In self-attention, the model computes attention scores for each word in the sequence relative to every other word, capturing dependencies and relationships within the same sequence. This operation is particularly effective for tasks requiring a deep understanding of contextual relationships, such as language modeling and translation [21].

Self-attention operates using three key vectors derived from the input embeddings: the query (Q), key (K), and value (V) matrices. The query matrix (Q) represents the current word, the key matrix (K) represents the words being compared against, and the value matrix (V) contains the information to be attended to. Additionally, d_k, the dimensionality of the key vectors, scales the dot product to ensure numerical stability when calculating attention scores. The scaled dot-product attention formula is defined as follows:

$$Attention(Q, K, V) = softmax\left(\frac{QK^T}{\sqrt{d_k}}\right)V$$

The attention mechanism computes the dot product between Q and K to measure the similarity between words. The resulting scores are normalized using the softmax function, producing weights prioritizing the sequence's most relevant parts. These weights are then used to calculate a weighted sum of the V values, ensuring that the model focuses on the most important input features.

While self-attention effectively captures relationships within a sequence, a single attention head may miss certain features due to limited representational capacity. To address this, as shown in Figure 3, multi-head attention performs multiple parallel attention operations with distinct query, key, and value projections. This enables the model to focus on different parts of the sequence simultaneously and learn diverse representations of the input data. The outputs from each attention head are concatenated and linearly transformed to produce the final representation, as defined by the formula:

$$MultiHead(Q, K, V) = Concat(head_1, \ldots, head_h)W^o$$

Here, h denotes the number of attention heads and W^orepresents the weight matrix used for the final linear transformation. By leveraging multi-head attention, Transformers can capture complex dependencies within sequences while enhancing the model's expressiveness and overall performance across various tasks [21].

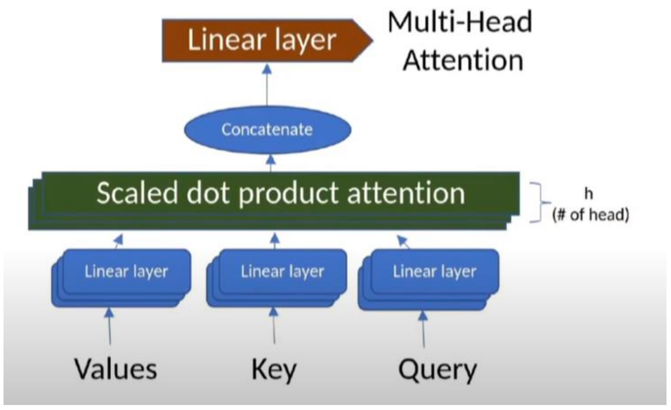

Figure 3. Multi-Head Attention mechanism in Transformers. Reprinted from *Transformers Model | شرح بالعربي*

- Add & Norm:

This step is crucial for stabilizing and enhancing the training process. This step involves two primary operations. First, it introduces a residual connection by adding the original input (such as the word embedding or output from the preceding layer) to the output of the multi-head attention mechanism. This addition ensures the retention of original input information, which facilitates better gradient flow during backpropagation. Second, it applies layer normalization to the summed output. Layer normalization ensures that the resulting values have a mean of zero and a variance of one, thereby standardizing the data distribution. This normalization step helps improve the convergence speed and stability of the model, especially during training. By combining residual connections with normalization, the "Add & Norm" block enhances the model's ability to learn complex relationships efficiently while maintaining numerical stability throughout the training process [21].

- Feed Forward:

The feed-forward network (FFN) applies two consecutive linear transformations to the input, interspersed with a non-linear activation function. The process is mathematically represented as:

$$FFN(x) = ReLU(W_1x + b_1)W_2 + b_2$$

Here, x is the input vector, W_1 and W_2 are weight matrices for the first and second linear transformations, respectively, and b_1 and b_2 are the corresponding bias vectors. The non-linear activation function, ReLU ($ReLU(x) = \max(0, x)$) introduces non-linearity, enabling the network to model complex relationships. This architecture transforms the intermediate representations produced by the multi-head attention mechanism, enriching them for further processing [21].

- Add & Nom:

The last step is the same as previous Add & Nom layer. The output of the feed-forward neural network is again added to the input of FFN and further normalized.

The encoder produces a sequence of embeddings, with each embedding corresponding to a specific position. These embeddings include not only the representation of the original word at that position but also incorporate information about other words in the sequence, learned through the attention mechanism. This enriched sequence is then passed on to subsequent components of the Transformer architecture for further processing [21].

- BERT

BERT's model architecture is a multi-layer bidirectional Transformer encoder, designed to fully capture context by processing text bidirectionally. Its training process involves two distinct steps: pre-training and fine-tuning. During pre-training, BERT is optimized using two unsupervised tasks. The first task, Masked Language Modeling (MLM), randomly masks a subset of input tokens, requiring the model to predict these tokens based on their surrounding context. This approach enables the model to learn deep, bidirectional representations of language. The second task, Next Sentence Prediction (NSP), trains the model to understand sentence relationships by predicting whether one sentence logically follows another. This dual-task training allows BERT to excel in understanding both word-level and sentence-level semantics. The fine-tuning step customizes the pre-trained BERT model for specific downstream tasks, such as text classification or question answering. This is achieved by adding minimal task-specific layers while retaining the pre-trained parameters. Fine-tuning enables BERT to adapt effectively to a wide range of natural language processing (NLP) applications with minimal adjustments to its architecture [7].

2.2.3 Deep Learning Architectures

- Siamese Neural Network (SNN)

Siamese Neural Network is a specialized neural network architecture designed to compare two inputs and determine their similarity or dissimilarity. Unlike traditional networks that produce a single output, SNNs consist of two or more identical subnetworks with shared architecture, configuration, and weights. Each subnetwork processes one input independently, transforming the inputs into embeddings in the same feature space. As visually represented in Figure 4, this design enables the network to focus on the relationship between the inputs rather than their individual characteristics by ensuring consistent feature extraction through shared weights.

Siamese networks offer significant advantages, particularly in applications where distinguishing relationships between inputs is essential. Their architecture is highly efficient in learning similarity functions, making them ideal for facial recognition, signature verification, and text similarity tasks. Moreover, their ability to generalize well from limited data reduces the dependency on large labeled datasets, which is a key advantage in scenarios where data collection is challenging. By focusing on relationships between inputs rather than individual data points, Siamese Neural Networks have become a critical tool in similarity-based machine learning tasks [6,18].

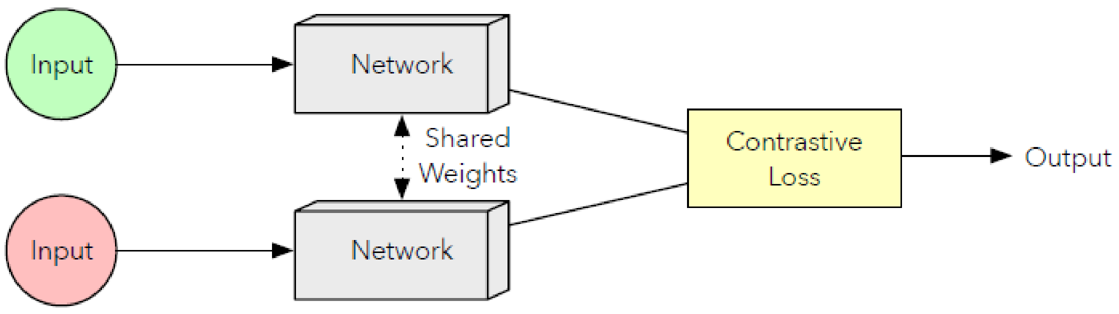

Figure 4. Example of Siamese network Architecture. Reprinted from *Learning to Weight Similarity Measures with Siamese Networks: A Case Study on Optimum-Path Forest*.

- Convolutional Neural Network (CNN)

Convolutional Neural Networks (CNNs) are deep learning models initially designed for computer vision but effectively adapted for natural language processing (NLP) tasks like text classification, sentiment analysis, and named entity recognition [23]. As illustrated in Figure 5, this structure processes sentences as one-dimensional data, utilizing convolutional layers with learnable filters to extract hierarchical features and identify patterns like word sequences or phrases. The Rectified Linear Unit (ReLU) activation function adds non-linearity, enhancing the model's ability to learn complex patterns. Pooling layers further refine the feature maps by reducing their dimensions, improving computational efficiency. Finally, fully connected layers combine the extracted features to make predictions. This architecture provides robust solutions for modeling complex relationships within textual data.

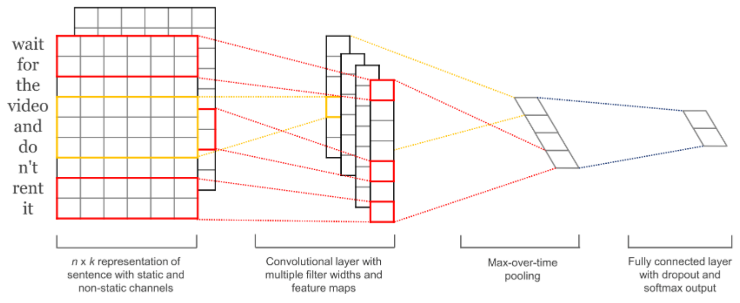
wait
for
the
video
and
do
n't
rent
it
n x k representation of sentence with static and non-static channels
Convolutional layer with multiple filter widths and feature maps
Max-over-time pooling
Fully connected layer with dropout and softmax output

Figure 5. Model architecture with two channels for an example sentence. Reprinted from *Convolutional Neural Networks for Sentence Classification*.

- Bidirectional Long Short-Term Memory (BiLSTM)

To provide a comprehensive understanding of BiLSTM networks, it is essential to first explain the foundational concepts and components upon which they are built.

○ RNN:

In natural language processing (NLP), processing sequential data is essential for understanding word relationships and the contextual meaning of sentences. While Bidirectional Encoder Representations from Transformers (BERT) excels in bidirectional contextual analysis, Recurrent Neural Networks (RNNs) are also highly effective for sequential data due to their unique architecture. RNNs consist of interconnected neural network cells designed to process temporal or ordered information, making them well-suited for tasks like language translation, speech recognition, and image captioning. RNNs utilize feedback loops to maintain and update a hidden state, allowing information from previous steps to influence current outputs. This enables the model to retain context across sequences, effectively capturing dependencies over time. As illustrated in Figure 6, RNNs process data sequentially, providing robust solutions for tasks requiring a deep understanding of sequential relationships [24].

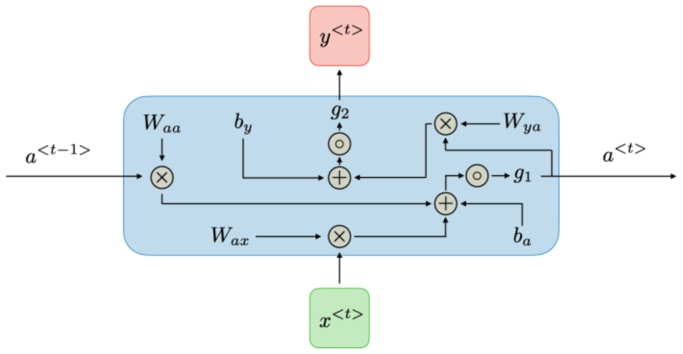

Figure 6. Illustration of a Recurrent Neural Network (RNN) architecture demonstrating sequential data flow. Reprinted from *CS 230 - Deep Learning Cheatsheet: Recurrent Neural Networks*.

There are five types of RNN:

- One-to-one $T_x = T_y = 1$ Traditional neural network.
- One-to-many $T_x = 1,\ T_y > 1$ Music generation.
- Many-to-one $T_x > 1,\ T_y = 1$ Sentiment classification.
- Many-to-many $T_x = T_y$ Name entity recognition.
- Many-to-many $T_x \neq T_y$ Machine translation.

Where T_x represents the length of the input sequence and T_y represents the length of the output sequence. For each timestep t, the input $x^{<t>}$, the activation $a^{<t>}$ and the output $y^{<t>}$ are expressed as follows:

$$a^{<t>} = g_1(W_{aa}a^{<t-1>} + W_{ax}x^{<t>} + b_a) \quad and \quad y^{<t>} = g_2(W_{ya}a^{<t>} + b_y)$$

where W_{ax} , W_{aa} , W_{ya} , b_a , b_y are coefficients that are shared temporally and g_1 , g_2 are activation functions.

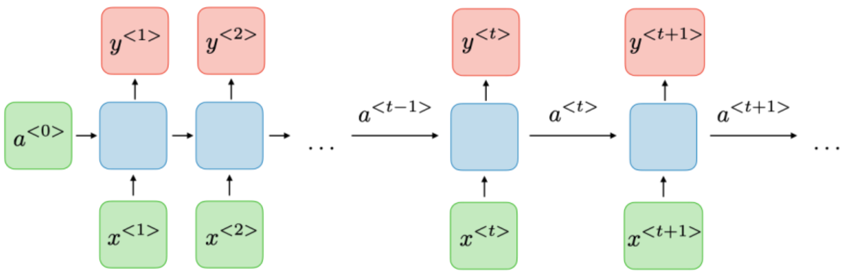

Figure 7. Internal structure of an RNN cell, showing input $x^{<t>}$, weights, activations, and output $y^{<t>}$. Reprinted from *CS 230 - Deep Learning Cheatsheet: Recurrent Neural Networks*.

Recurrent Neural Networks (RNNs) utilize prior inputs to influence current outputs, giving them a "memory" that makes them effective for processing sequential data. Unlike traditional deep neural networks, this architecture assumes dependencies between inputs and outputs. However, RNNs face challenges like exploding gradients, where weights become excessively large, and vanishing gradients, where weights become too small, hindering learning during backpropagation. Advanced architectures like Long Short-Term Memory (LSTM) networks and Gated Recurrent Units (GRUs) address these issues. For tasks like author attribution, a Many-to-One RNN model is applied, where sequential data is processed to produce a single output [24].

○ Long Short-Term Memory (LSTM):

Advanced architectures such as Long Short-Term Memory (LSTM) networks were introduced to address the vanishing gradient problem encountered in traditional RNNs. LSTMs incorporate gates that regulate the flow of information, enabling them to learn long-term dependencies effectively. These gates help determine which information to retain, discard, or output at each step, making LSTMs particularly effective for tasks involving sequential data, such as natural language processing, time series analysis, and speech recognition. To remedy the vanishing gradient problem, specific gates are used in some types of RNNs and usually have a well-defined purpose. They are usually noted as Γ and are equal to:

$$\Gamma = \sigma(Wx^{<t>} + Ua^{<t-1>} + b)$$

where W, U, b are coefficients specific to the gate and σ is the sigmoid function. The main ones are summed up in the table below:

Table 1. Types of Gates in LSTM Networks and Their Functions.

Type of gate	**Role**
Update gate Γ_u	How much past should matter now?
Relevance gate Γ_r	Drop previous information?
Forget gate Γ_f	Erase a cell or not?
Output gate Γ_o	How much to reveal of a cell?

The cell state candidate computation introduces a more complex interaction through the equation:

$$\tilde{c}^{<t>} = \tanh(W_c[\Gamma_r \star a^{<t-1>}, x^{<t>}] + b_c)$$

This equation employs the hyperbolic tangent activation function to process a combination of the current input and the gated previous state. The actual cell state update mechanism is captured in the equation:

$$c^{<t>} = \Gamma_u \star \tilde{c}^{<t>} + \Gamma_f \star c^{<t-1>}$$

This crucial step determines how information flows through the network over time. The final output computation represented by:

$$a^{<t>} = \Gamma_o \star c^{<t>}$$ (1)

(1)The sign ⋆ denotes the element-wise multiplication between two vectors.

determines what information is exposed to subsequent layers or time steps.

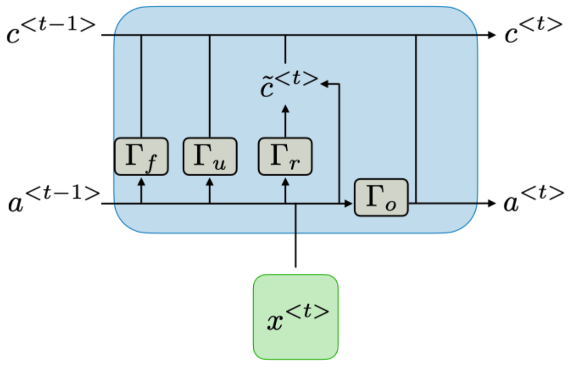

Figure 8. Diagram of an LSTM cell illustrating the flow of information through gates and cell states for sequence modeling. Reprinted from *CS 230 - Deep Learning Cheatsheet: Recurrent Neural Networks*.

○ BiLSTM

Bidirectional Long Short-Term Memory (BiLSTM) networks enhance the traditional LSTM architecture by processing input sequences in both forward and backward directions. Unlike unidirectional LSTMs, which capture context from either past or future inputs, BiLSTMs analyze sequences simultaneously in both directions, allowing them to utilize richer contextual information. This is particularly valuable in natural language processing (NLP) tasks such as sequence labeling, machine translation, and sentiment analysis, where understanding the complete context of a sequence is critical [30]. BiLSTM layers combine outputs from forward and backward passes through concatenation, summation, or averaging, enabling the network to effectively model complex dependencies. By leveraging this dual-direction processing, BiLSTMs achieve state-of-the-art performance across various NLP applications, including named entity recognition and part-of-speech tagging [21].

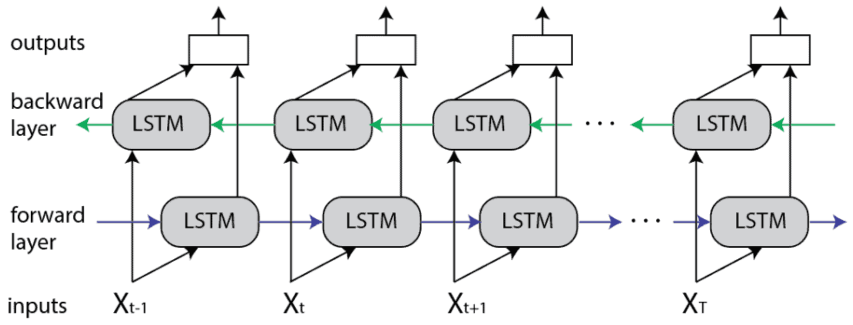

Figure 9. Diagram of a Bidirectional LSTM, showing forward and backward layers processing input sequences to capture context from both directions. Reprinted from *Bidirectional vs. unidirectional LSTM*.

2.2.4 Post-Processing Techniques

- Isolation Forest

The Isolation Forest algorithm is an unsupervised outlier detection method that identifies anomalies by leveraging their rarity and distinctness. Instead of relying on density or distance metrics, it uses randomly generated binary trees to isolate data points. The algorithm measures the ease of isolation through the path length in these trees, with anomalies typically having shorter path lengths due to their unique characteristics [13].

The expected path length, denoted as $E(h(x_i))$, is the average number of splits required to isolate a data point x_i across all trees in the forest. To adjust for variations in tree height, the effective path length, $h(x_i)$, incorporates the expected depth of an unsuccessful search in a binary tree, given N, the number of instances in the training set. This is expressed as:

$$c(N) = 2H(N-1) - \frac{2(N-1)}{N}$$

where $H(i) \approx \ln(i) + \gamma^{(2)}$ is the harmonic number approximation, and $c(N)$ normalizes the path length. The anomaly score for a data point x_i is computed using the following formula:

$$s(x_i, N) = 2^{-\frac{(h(x_i))}{c(N)}}$$

where $s(x_i, N)$ ranges between 0 and 1, with higher scores indicating a greater likelihood of being an anomaly. This score reflects the ease with which a data point is isolated compared to the rest of the dataset [13]. Isolation Forest's efficiency and scalability stem from its non-parametric nature and the use of random splits. It eliminates the need for predefined thresholds or assumptions about data distributions, making it well-suited for unsupervised outlier detection in high-dimensional datasets [8].

- Clustering

Clustering is an unsupervised machine-learning technique that organizes data points, objects, or observations into groups or clusters based on shared characteristics or patterns. It is widely used in exploratory data analysis, anomaly detection, and preprocessing steps to uncover underlying trends, reduce dimensionality, or segment data into smaller, more meaningful groups. Clustering applications include market research for customer segmentation, image segmentation to divide images into distinct regions, and document processing for grouping or summarizing textual content. Clustering techniques can be categorized into hard clustering, where data points belong to a single cluster, or soft clustering, where probabilities indicate the likelihood of a data point belonging to each cluster. This technique enables a better understanding of relationships within data, improves visualization, and can even identify anomalies by detecting data points that do not fit within clusters [28].

(2)Euler-Mascheroni Constant: γ = 0.5772156649.

3. Expected Achievements

This section outlines the expected achievements of the project and the anticipated outcomes, providing an overview of the goals and measures of success.

3.1 Objectives

The main goal of this project is to improve authorship attribution techniques for Arabic texts by creating a computational framework that leverages advanced machine-learning approaches.
This framework incorporates Siamese Networks, AraBERT post-training, and signal representation to analyze and capture distinctive stylistic features of authors effectively. By addressing the inherent complexities of Arabic texts, such as morphological richness and stylistic diversity, the project aims to generate robust numerical embeddings for words, sentences, and documents. These embeddings will visually represent an author's unique writing style within a high-dimensional feature space. Additionally, the project seeks to evaluate the framework's adaptability and reliability by incorporating diverse impostors to mimic real-world authorship scenarios. The proposed model will also be applied to historical Arabic texts, particularly those attributed to Al-Ghazali, to distinguish authentic works from pseudo-authentic ones. This application is expected to contribute valuable insights to the fields of literary analysis and historical document classification.

3.2 Expected Outcomes

1. An adapted computational framework for authorship attribution in Arabic texts, leveraging pre-trained models like AraBERT and advanced techniques such as Siamese Networks and signal representation.
2. Robust numerical embeddings that represent the stylistic features of authors in Arabic texts, applicable at the word, sentence, and document levels.
3. Authorship attribution incorporates diverse unknown impostors to simulate real-world scenarios, enhancing the model's adaptability and scalability.
4. Experiments validating the framework's ability to distinguish authentic works from pseudo-authentic ones, focusing on historical texts attributed to Al-Ghazali.

3.3 Success Criteria

Success for this project will be determined by its ability to distinguish between texts authored by Al-Ghazali accurately and those falsely attributed to him (Pseudo-Ghazali) within the test group. A critical evaluation point will be the framework's capability to correctly identify Iḥyāʾ ʿulūm al-dīn, a work definitively known to be written by Al-Ghazali, serving as a benchmark for validating the results. The project aims to group authentic Al-Ghazali texts into a distinct cluster while separating pseudo-Ghazali texts, leveraging stylistic markers unique to his writing. This will demonstrate the model's effectiveness in capturing the underlying style and reliability in accurate authorship attribution.

4. Research Process

4.1 Process

The research process for this project began with a comprehensive study of the authorship attribution field, focusing on challenges specific to medieval Arabic texts. Recognizing the complexity of this task, we prioritized understanding the linguistic and stylistic intricacies of Arabic while exploring computational methods tailored to text analysis. Much of this effort involved reviewing advanced machine learning algorithms, particularly those based on Transformer architectures like BERT. Our initial phase emphasized understanding the theoretical foundations of Transformer-based models and their components, such as attention mechanisms and positional encoding. This was crucial for designing a model that aligns with our task's unique requirements.

Additionally, we examined the latest studies on preprocessing Arabic texts to ensure our approach is informed and reflective of the field's current state. Key considerations in our process included the scalability of Transformer models for handling large textual datasets and the adaptation of pre-training and fine-tuning strategies to suit historical and stylistic variations in the data. These challenges necessitated careful evaluation of computational constraints and the selection of models that balance efficiency with effectiveness. The next phase will involve practically implementing the insights gained thus far. This includes fine-tuning Transformer models, testing configurations, and evaluating their performance on authorship attribution tasks. By combining theoretical rigor with practical application, we aim to address the challenges of Arabic text analysis and contribute to the broader field of authorship attribution.

4.2 Proposed Approach

The following diagram, Figure 10, illustrates the workflow of the proposed algorithm. The process begins with the preprocessing phase (left), where Arabic documents are prepared for analysis. The training and classification phases (right) follow. Each step in the workflow is explained in detail below.

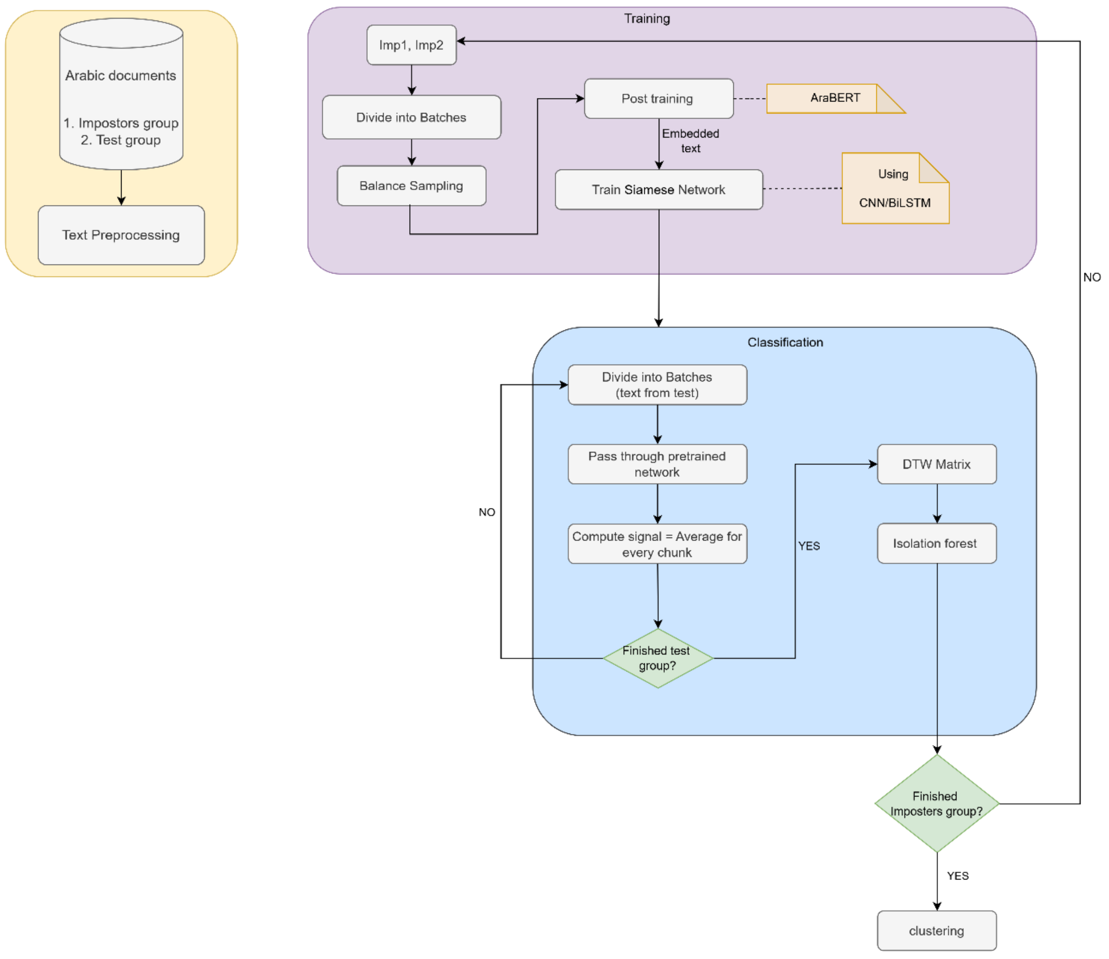

Figure 10. Algorithm explanation diagram.

- Preparing the Dataset

1. Impostors: This group consists of texts authored by individuals different from Al-Ghazali. These texts include works from Al-Ghazali's contemporaries, religious texts from the Islamic Golden Age, philosophical texts from the same era, and non-matching genres such as poetry or historical narratives. The diversity of this group ensures the model can handle stylistic variability and distinguish Al-Ghazali’s unique writing style from a wide range of other styles, simulating real-world authorship attribution scenarios.
2. Test group: This consists of texts being evaluated to distinguish the authentic works of Al-Ghazali from those falsely attributed to him (Pseudo-Ghazali). This includes confirmed texts, such as *Iḥyāʾ ʿulūm al-dīn*, which serve as benchmarks for validating the accuracy of authorship attribution, as well as unknown authorship texts that stylistically resemble his works but lack definitive evidence of authorship.

- Preprocess the Dataset

Arabic text preprocessing is critical in natural language processing (NLP). It ensures that data is clean, consistent, and ready for downstream tasks such as classification, clustering, and sentiment analysis. Preprocessing is particularly significant for Arabic due to its unique morphology, syntax, and orthographic complexities. The key steps include:

1. Tokenization: Splits text into smaller units ("tokens"). CAMeL Tools is employed for Arabic tokenization because it effectively handles the language's morphology.
2. Stopword Removal: Eliminates common, uninformative words such as pronouns and prepositions. Tools such as the Khoja list, El-Khair list, and ASL list (containing 67,153 words) are used to reduce dimensionality and computational load.
3. Normalization: Simplifies the text by:
   - Removing non-Arabic letters and special characters.
   - Replacing variations of "إ" ,"أ", and "آ" with "ا".
   - Converting "ة" (ta'a marbota) to "ه" (ha'a).
   - Removing "ال" (definite article) from the beginning of words.
   - Replacing the final "ى" with "ا".
   - Removing diacritics (tashkeel) to standardize text structure.
4. Stemming: Removes prefixes and suffixes while retaining the root of the word (e.g., "المعلم" → "معلم"). ARLSTem is used for light stemming, effectively reducing dimensionality while preserving stylistic and lexical details.

- Training

The first step involves dividing the impostors group into pairs. Each pair is then processed through the subsequent steps as outlined below:

1. Dividing the impostors into batches
2. This step involves balancing the data to ensure equal representation, enhancing the model's reliability. This procedure includes two techniques:
   - Undersampling: In cases where one document is larger than the other. Random undersampling (RU) is applied. This involves randomly removing samples from the majority class to match the number of documents in the minority class.
   - Oversampling: In cases where one document is smaller than the other. Oversampling is

performed using the SMOTE (Synthetic Minority Over-sampling Technique) algorithm. This method generates synthetic samples to increase the size of the minority class, ensuring balanced representation [1].

3. Post-training (Embedding Layer with AraBERT): This layer bridges the core layer and the input, using AraBERT, a pre-trained Arabic transformer. It processes tokenized text sequences (up to 512 tokens, with a 64,000 vocabulary) and outputs a multidimensional matrix [sequence_length, embedding_dim], where each token is a 768-dimensional vector. AraBERT applies Arabic-specific preprocessing and fine-tunes embeddings.
4. Siamese network:

The core layer: It is the central layer where the contextual information and all the semantic are extracted. We will conduct our experiments by instantiating this layer with various neural models:

- CNN: Captures spatial and sequential patterns in text embeddings by applying convolutional layers, as text data is represented as a sequence. The 1D kernel operates along the temporal dimension of the sequence to extract features from n-grams or local patterns. Multiple filters are used to capture diverse features, emphasizing local dependencies and contextual relationships within the text. The output is a set of feature maps that preserve stylistic and structural nuances.
- BiLSTM: Takes the feature maps generated by the CNN as input and processes them in both forward and backward directions to capture sequential dependencies. It models long-term contextual relationships in the data, integrating the local features extracted by the CNN with broader global dependencies. The resulting context-aware representation is then passed to the Siamese network for similarity assessment.

The outputs of those subnetworks are compared using the Euclidean distance equation: ($d = \sqrt{\sum_{i=1}^{n}(z_{1i} - z_{2i})^2}$), which calculates the distance between the embeddings z_1 and z_2 . A smaller distance indicates higher similarity, while a larger distance indicates dissimilarity. During training, parameter updates occur simultaneously across the subnetworks, maintaining identical weights and ensuring uniform feature extraction. This is achieved by using a loss function that evaluates similarity by minimizing the distance between embeddings for similar pairs while maximizing the distance for dissimilar pairs. A contrastive loss function is employed to optimize the performance of Siamese networks. The function is defined as:

$$L = (1 - Y)\frac{1}{2}(D)^2 + (Y)\frac{1}{2}\{\max(0, m - D)\}^2$$

where Y is a binary label indicating whether the inputs are similar (Y =0) or dissimilar (Y =1), D is the Euclidean distance between the embeddings, and m is a margin that ensures a minimum separation for dissimilar pairs (by default, it is set to 1. And will be adjusted based on validation performance).

- Classification

For each text in the Test group, the following procedure is performed:

1. Dividing the text into batches.
2. Process it through the trained network retained from the training phase.
3. A chunk becomes a mean score of its sub-batches, as illustrated in Figure 11. The output of this step is a signal representation for each text, composed of the scores of its respective chunks.

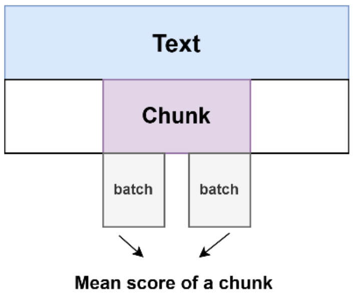

Figure 11. Illustration of Text Segmentation and Batch Assignment. Inspired by [Deep Sentimental Analysis of the Arabic Medieval Documents]().

We then proceed with the following steps:

- Computation of a pairwise Dynamic Time Warping (DTW) [15] distance matrix.
- Application of the Isolation Forest algorithm, as outlined in Section 2.2.4.

The training and classification processes are repeated iteratively for all pairs of impostors. Once all pairs have been processed, we proceed as follows:

- the final step involves clustering using the k-means algorithm with k=2. This step is crucial for distinguishing between texts definitively authored by Al-Ghazali and those falsely attributed to him.

4.3 Sequence diagram

The sequence diagram below illustrates the step-by-step flow of the proposed algorithm, capturing the sequential interactions and processes leading to the final results.

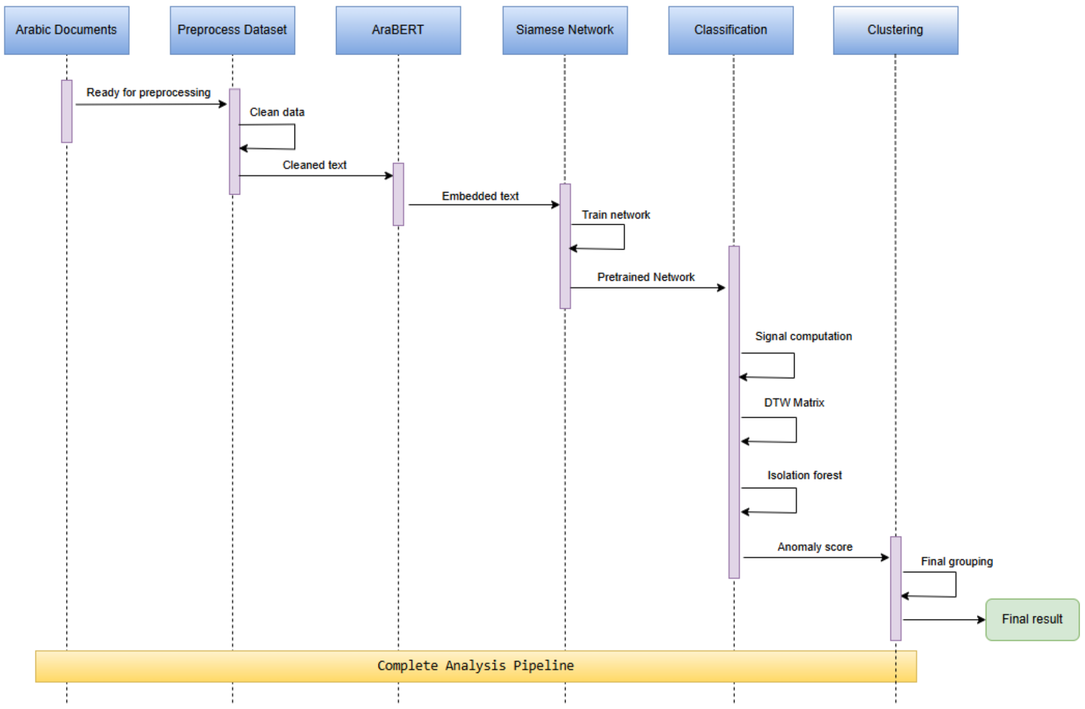

Figure 12. Sequence diagram of the proposed algorithm.

5. Evaluation Plan

The evaluation plan for this project is designed to thoroughly assess the proposed computational framework for authorship attribution of medieval Arabic texts. The primary focus is on testing the framework’s ability to preprocess, analyze, and classify texts accurately while maintaining robustness against noisy or irrelevant data. Each component of the algorithm will undergo systematic evaluation to ensure its functionality and reliability. The tests will include validating preprocessing steps such as tokenization and normalization, assessing the quality of AraBERT embeddings, evaluating the Siamese network's similarity scoring, and verifying the framework's capacity to classify texts into authentic and pseudo-authentic clusters. A detailed outline of the evaluation plan is presented in Table 2.

Table 2. Detailed Evaluation Plan for the Proposed Algorithm.

Test No.	Test Subject	Expected Result
1	Validate tokenization and normalization: Test Arabic text splitting and standardization.	Accurate and standardized tokenization and normalization across datasets.
2	Verify stopword removal: Ensure that uninformative words are removed effectively.	Reduced dimensionality while retaining key textual features.
3	Assess AraBERT embeddings: Test semantic and stylistic accuracy of generated embeddings.	High-quality embeddings that capture linguistic and stylistic nuances.
4	Evaluate similarity scoring: Use paired texts in the Siamese network to test similarity/dissimilarity.	Accurate similarity or dissimilarity scores reflecting true relationships.
5	Test robustness against noisy data: Introduce misspellings and incomplete sentences.	Consistent and reliable performance despite noisy or incomplete input.
6	Process incorrect documents: Test the system's handling of unrelated or irrelevant texts.	Proper rejection of irrelevant inputs with appropriate error handling.
7	Validate clustering: Test clustering of texts into authentic and pseudo-authentic categories.	Clear and accurate categorization of texts into predefined clusters.
8	Evaluate end-to-end framework: Assess the combined system's ability to identify authentic works.	Reliable authorship attribution with minimal errors across the test dataset.

References

[1] Al-Khazaleh, M. J., Alian, M., & Jaradat, M. A. (2024). Sentiment analysis of imbalanced Arabic data using sampling techniques and classification algorithms. *Bulletin of Electrical Engineering and Informatics*, 13(1), 607–618. https://doi.org/10.11591/eei.v13i1.5886

[2] Al-Sarem, M., Saeed, F., Alsaeedi, A., Boulila, W., & Al-Hadhrami, T. (2020). Ensemble methods for instance-based Arabic language authorship attribution. *IEEE Access, 8*, 17331–17342. https://doi.org/10.1109/ACCESS.2020.2964952

[3] AlZahrani, F. M., & Al-Yahya, M. (2023). A Transformer-Based Approach to Authorship Attribution in Classical Arabic Texts. *Applied Sciences, 13*(12), 7255. https://doi.org/10.3390/app13127255

[4] Avros, R., & Volkovich, Z. (2022). Deep sentimental analysis of the Arabic medieval documents. *Procedia Computer Science, 207*, 709–715. https://doi.org/10.1016/j.procs.2022.09.126

[5] Bauersfeld, L., Romero, A., Muglikar, M., & Scaramuzza, D. (2023). Cracking double-blind review: Authorship attribution with deep learning. *PLOS ONE, 18*(6), e0287611. https://doi.org/10.1371/journal.pone.0287611

[6] De Rosa, G. H., & Papa, J. P. (2022). Learning to weight similarity measures with Siamese networks: A case study on optimum-path forest. In *Optimum-Path Forest*. https://doi.org/10.1016/B978-0-12-822688-9.00015-3

[7] Devlin, J., Chang, M. W., Lee, K., & Toutanova, K. (2019). BERT: Pre-training of deep bidirectional transformers for language understanding. *Proceedings of NAACL-HLT 2019*, 4171–4186. https://doi.org/10.18653/v1/N19-1423

[8] Domingues, R., Filippone, M., Michiardi, P., & Zouaoui, J. (2018). A comparative evaluation of outlier detection algorithms: Experiments and analyses. *Pattern Recognition, 74*, 406–421. https://doi.org/10.1016/j.patcog.2017.09.037

[9] Fabien, M., Villatoro-Tello, E., Motlicek, P., & Parida, S. (2020). BertAA: BERT fine-tuning for Authorship Attribution. In P. Bhattacharyya, D. M. Sharma, & R. Sangal (Eds.), *Proceedings of the 17th International Conference on Natural Language Processing (ICON)* (pp. 127–137). NLP Association of India (NLPAI). https://aclanthology.org/2020.icon-main.16

[10] He, X., Lashkari, A. H., Vombatkere, N., & Sharma, D. P. (2024). Authorship attribution methods, challenges, and future research directions: A comprehensive survey. *Information, 15*(3), 131. https://doi.org/10.3390/info15030131

[11] Huang, Z., & Iwaihara, M. (2022). Authorship Attribution Based on Pre-Trained Language Model and Capsule Network. *DEIM2022*. https://proceedings-of-deim.github.io/DEIM2022/papers/H33-4.pdf

[12] Juola, P. (2008). Authorship attribution. *Foundations and Trends in Information Retrieval, 1*(3), 233–334. https://doi.org/10.1561/1500000005

[13] Liu, F. T., Ting, K. M., & Zhou, Z.-H. (2008). Isolation Forest. *2008 Eighth IEEE International Conference on Data Mining*, 413–422. https://doi.org/10.1109/ICDM.2008.17

[14] Mikolov, T., Chen, K., Corrado, G., & Dean, J. (2013). Efficient estimation of word representations in vector space. *Proceedings of the 1st International Conference on Learning Representations (ICLR)*. https://arxiv.org/abs/1301.3781

[15] Müller, M. (2007). *Dynamic time warping*. In *Information Retrieval for Music and Motion* (pp. 69–84). Springer, Berlin, Heidelberg. https://doi.org/10.1007/978-3-540-74048-3_4

[16] Pennington, J., Socher, R., & Manning, C. D. (2014). GloVe: Global vectors for word representation. *Proceedings of the 2014 Conference on Empirical Methods in Natural Language Processing (EMNLP)*, 1532–1543. https://doi.org/10.3115/v1/D14-1162

[17] Reisi, E., & Mahboob Farimani, H. (2020). Authorship Attribution in Historical and Literary Texts by a Deep Learning Classifier. *Journal of Applied Intelligent Systems and Information Sciences, 1*(2). https://doi.org/10.22034/jaisis.2021.269735.1018

[18] Serrano, N., & Bellogín, A. (2023). Siamese neural networks in recommendation. *Neural Computing and Applications*, 35, 13941–13953. https://doi.org/10.1007/s00521-023-08610-0

[19] Silva, K., Can, B., Blain, F., Sarwar, R., Ugolini, L., & Mitkov, R. (2023). Authorship Attribution of Late 19th Century Novels using GAN-BERT. In V. Padmakumar, G. Vallejo, & Y. Fu (Eds.), *Proceedings of the 61st Annual Meeting of the Association for Computational Linguistics (Volume 4: Student Research Workshop)* (pp. 310–320). Association for Computational Linguistics. https://doi.org/10.18653/v1/2023.acl-srw.44

[20] Springer. (2023). Word embeddings are an n-dimensional representation of words in a continuous space. *Artificial Intelligence Review*. https://doi.org/10.1007/s10462-023-10419-1

[21] Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A. N., Kaiser, Ł., & Polosukhin, I. (2017). Attention is all you need. *Proceedings of the 31st International Conference on Neural Information Processing Systems,* 6000–6010. https://user.phil.hhu.de/~cwurm/wp-content/uploads/2020/01/7181-attention-is-all-you-need.pdf
[22] Volkovich, Z. (2020). A short-patterning of the texts attributed to Al-Ghazali: A "Twitter look" at the problem. *Mathematics*, *8*(11), 1937. https://doi.org/10.3390/math8111937
[23] Yoon, K. (2014). Convolutional neural networks for sentence classification. *arXiv preprint arXiv:1408.5882*. https://arxiv.org/pdf/1408.5882.pdf
[24] https://stanford.edu/~shervine/teaching/cs-230/cheatsheet-recurrent-neural-networks
[25] https://www.britannica.com/biography/al-Ghazali
[26] https://www.geeksforgeeks.org/seq2seq-model-in-machine-learning/
[27] https://www.ibm.com/think/topics/word-embeddings#:~:text=Word%20embeddings%20capture%20semantic%20relationships,more%20nuanced%20understanding%20of%20language
[28] https://www.ibm.com/think/topics/clustering
[29] https://www.ibm.com/think/topics/encoder-decoder-model
[30] https://www.baeldung.com/cs/bidirectional-vs-unidirectional-lstm