Computer Systems Science and Engineering

Tech Science Press

DOI: 10.32604/csse.2024.054615

ARTICLE

An Expert System to Detect Political Arabic Articles Orientation Using CatBoost Classifier Boosted by Multi-Level Features

Saad M. Darwish[1,*], Abdul Rahman M. Sabri[2], Dhafar Hamed Abd[2] and Adel A. Elzoghabi[1]

[1]Department of Information Technology, Institute of Graduate Studies and Research, Alexandria University, El Shatby, P.O. Box 832, Alexandria, 21526, Egypt

[2]Department of Computer Science, College of Computer Science and Information Technology, University of Anbar, Baghdad, 55431, Iraq

*Corresponding Author: Saad M. Darwish. Email: saad.darwish@alexu.edu.eg

Received: 03 June 2024 Accepted: 08 August 2024 Published: 22 November 2024

ABSTRACT

The number of blogs and other forms of opinionated online content has increased dramatically in recent years. Many fields, including academia and national security, place an emphasis on automated political article orientation detection. Political articles (especially in the Arab world) are different from other articles due to their subjectivity, in which the author's beliefs and political affiliation might have a significant influence on a political article. With categories representing the main political ideologies, this problem may be thought of as a subset of the text categorization (classification). In general, the performance of machine learning models for text classification is sensitive to hyperparameter settings. Furthermore, the feature vector used to represent a document must capture, to some extent, the complex semantics of natural language. To this end, this paper presents an intelligent system to detect political Arabic article orientation that adapts the categorical boosting (CatBoost) method combined with a multi-level feature concept. Extracting features at multiple levels can enhance the model's ability to discriminate between different classes or patterns. Each level may capture different aspects of the input data, contributing to a more comprehensive representation. CatBoost, a robust and efficient gradient-boosting algorithm, is utilized to effectively learn and predict the complex relationships between these features and the political orientation labels associated with the articles. A dataset of political Arabic texts collected from diverse sources, including postings and articles, is used to assess the suggested technique. Conservative, reform, and revolutionary are the three subcategories of these opinions. The results of this study demonstrate that compared to other frequently used machine learning models for text classification, the CatBoost method using multi-level features performs better with an accuracy of 98.14%.

KEYWORDS

Political articles orientation detection; CatBoost classifier; multi-level features; context-based classification; social networks; machine learning; stylometric features

1 Introduction

Detecting the orientation of political articles involves analyzing the content to determine its ideological leaning or bias. It's important to note that bias detection can be subjective, and different people may interpret the same content differently [1,2]. Detecting the orientation of political articles has several applications across various domains that include media monitoring and analysis, content recommendation systems, fact-checking and fake news detection, policy impact assessment, public opinion analysis, investment decision-making, and AI-powered journalism [3–5]. Detecting the orientation of political articles comes with several challenges due to the complexity and subjectivity of political ideologies. Some of the key challenges associated with this task include: (1) Political bias is often subjective, and individuals may interpret the same content differently based on their own perspectives. (2) Political ideologies vary widely, and articles may represent different factions within a broader ideology. Determining the specific political orientation can be challenging when an article contains a mix of perspectives. (3) Language and terminology evolve over time. New phrases or terms may emerge, making it difficult to rely solely on predefined lists of biased words [1–3,6].

Overcoming these challenges often requires a combination of advanced natural language processing (NLP) techniques, machine learning models, and continuous refinement based on real-world feedback [7–9]. Detecting the orientation of political articles using machine learning techniques involves training models on labelled datasets to classify articles into different political orientations [10]. Models can be trained using features such as word embeddings, sentiment scores, and other linguistic patterns. It's important to note that the effectiveness of the model depends on the quality and representativeness of the labelled dataset, the chosen features, and the model architecture [11].

Detecting the political orientation of articles using machine learning involves several steps, from data collection and preprocessing to model training and evaluation; see References [12–14] for more details regarding these main steps. In general, the quality and diversity of the collected dataset, as well as the choice of features (e.g., bag-of-words, TF-term frequency-inverse document frequency, N-grams, word embeddings, sentiment scores, and other linguistic patterns) and model architecture, greatly influence the model's effectiveness. One of the key steps involves extracting relevant features from the text to feed into a machine-learning model. In this context, multi-level feature selection involves selecting a subset of relevant features from different levels or types of data [15]. By selecting features that are representative across multiple levels, the model may generalize better to new, unseen data as it learns more robust and transferable patterns [16].

The difficulty of understanding political Arabic articles can vary depending on several factors, including your familiarity with the Arabic language, your level of proficiency, and the complexity of the political content [12,13]. Here are some specific challenges that learners of Arabic may encounter when dealing with political articles: (1) Arabic has multiple dialects, and the language used in political articles may include formal or classical Arabic. Understanding these variations can be challenging. (2) Arabic words often share a common root, and understanding the root system can aid in comprehending new words. (3) Arabic often uses complex sentence structures, and political articles may contain lengthy and intricate sentences. Finally, (4) Arabic has various regional dialects, and political articles may use vocabulary or expressions specific to certain regions. Overcoming these challenges requires a comprehensive approach to Arabic language learning, including targeted vocabulary building, exposure to formal language, practice with complex sentence structures, and ongoing engagement with political and cultural content [12].

1.1 Problem Statement and Motivation

Detecting the orientation of political Arabic articles involves determining the stance or perspective expressed in the text, such as whether it is conservative, reform, or revolutionary towards a particular political ideology or entity. This task is challenging due to the complex nature of political discourse, the use of nuanced language, and the need to discern the author's position on various issues. Current machine learning algorithms for detecting the political orientation of articles, while powerful and versatile, face various challenges that can impact their effectiveness and performance. Choosing relevant features and extracting meaningful information from raw data can be challenging (feature engineering). Inadequate feature selection can impact the model's performance. Furthermore, selecting the right set of hyperparameters for a model is often an iterative process and can be time-consuming. Grid search or random search methods are commonly used, but they require computational resources.

1.2 Contribution

This study focuses on utilizing CatBoost classifier [17–20] boosting with a multi-level feature vector to classify political Arabic text orientation. Incorporating multi-level features provides a more comprehensive and effective representation of the underlying patterns in the data. The suggested model considers creating multi-level feature vectors by combining (concatenating) existing ones commonly used for feature extraction and representation in text processing tasks, which include Bag of Word (BOW), Term Frequency-Inverse Document Frequency (TF-IDF), and Hashing Vectorization. The model improves its performance by gaining more discriminatory power by adding features at different levels. While many text-based political orientation classifiers rely on single-layer features, using concatenated features can offer substantial advantages. Different features capture different aspects of the text, such as frequency, context, and syntactic structure. Combining them enriches the model's understanding. Furthermore, multiple feature types can help reduce ambiguity in the text, leading to more precise predictions [21,22].

The remainder of the paper is structured as follows: Recent relevant works are discussed in Section 2. Section 3 provides a comprehensive description of the suggested model. Discussions and simulation results for the Arabic article dataset are presented in Section 4. Section 5 concludes the paper with results and implications.

2 Literature Review

Gradient boosting is a powerful machine-learning technique used for regression and classification tasks. It builds models in a sequential manner to correct the errors of the previous models. The most common implementation of gradient boosting uses decision trees as the base learners. Its key concepts include: (1) Ensemble Learning: Gradient boosting combines multiple weak learners (decision trees) to create a strong learner. Weak learners are models that perform slightly better than random guessing. (2) Boosting: Boosting is a sequential technique where each new model aims to reduce the errors made by the previous models. It focuses on difficult-to-predict cases by assigning them higher weights. (3) Gradient Descent: Gradient boosting uses gradient descent to minimize a loss function. It optimizes the model by iteratively adding trees that point in the direction of the negative gradient of the loss function [17,18]. The key hyperparameters include the number of trees, learning rate, tree depth, and minimum samples per leaf. The model steps are as follows: (1) Start with an initial prediction, often the log odds for classification. (2) Calculate the difference between the actual values and the predictions of the current model. (3) Train a new decision tree to predict the residuals. This tree will focus on correcting the errors made by the current model. (4) Add the new tree to the ensemble. Update the

model's predictions by adding the predictions of the new tree, scaled by a learning rate. (5) Finally, repeat Steps 2–4 for a predefined number of iterations or until the model's performance no longer improves.

CatBoost is based on the gradient boosting algorithm, a popular ensemble learning technique. It builds a series of decision trees, each correcting the errors of the previous one (see Fig. 1) and includes built-in support for regularization techniques, helping prevent overfitting and improving model generalization [17–19]. Comparing machine learning classifiers, including CatBoost, involves considering various factors such as performance, ease of use, handling of different types of data, and computational efficiency [20,21]. CatBoost stands out with its built-in support for categorical features, which simplifies preprocessing. Extreme Gradient Boosting (XGBoost) and Light Gradient Boosting Machines (LightGBM) also support categorical features but may require additional encoding steps to handle them. CatBoost may have faster training times than random forests on large datasets due to its optimized implementation and parallelization. Ensemble methods like CatBoost are capable of capturing non-linear relationships, which may be challenging for linear models like logistic regression. CatBoost models are generally more interpretable than complex neural networks. Neural networks may outperform CatBoost on very large datasets with complex patterns. However, neural networks often require more computational resources and may be more challenging to train effectively [17–20].

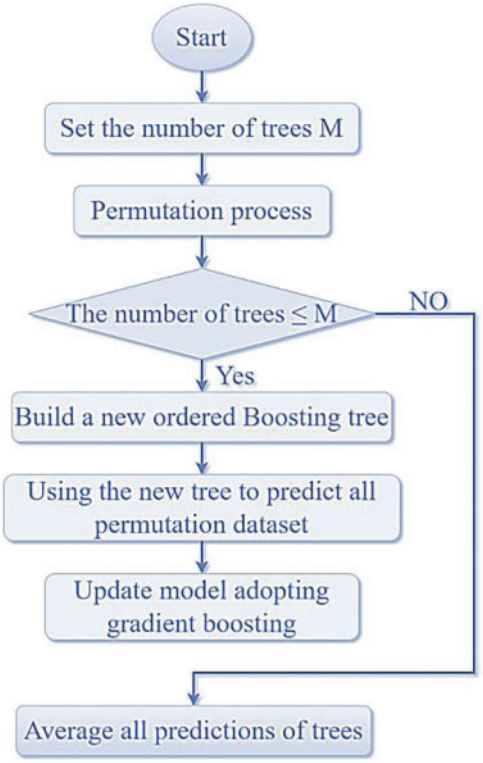

Figure 1: Flowchart of the CatBoost regression

Detecting the political orientation of Arabic articles is a complex task that involves various natural language processing (NLP) and machine learning techniques. This section discusses state-of-the-art related works that address political orientation detection in Arabic articles. These works span various aspects of political orientation detection in Arabic text, including sentiment analysis, bias detection,

and the use of machine learning and deep learning techniques. Each work contributes to advancing our understanding of handling political content in the Arabic language [23].

Approaches for determining the political orientation of Arabic articles can be divided into four categories in the academic literature [12,13,22]. (1) Lexicon-based: Analyze words and phrases associated with specific political stances. It requires building and maintaining sentiment lexicons tailored to the Arabic political language. (2) Machine learning: Train models on labelled datasets of articles with known orientations. Models can utilize features like n-grams, syntactic features, word embeddings, and topic modelling. (3) Deep learning: Convolutional neural networks (CNNs) and recurrent neural networks (RNNs) excel at capturing complex relationships in text data. (4) Rule-based: Define specific linguistic and stylistic rules indicative of certain orientations (e.g., use of rhetorical questions for criticism, specific pronouns for affiliation). Can be effective but requires expert knowledge and might struggle with subtle nuances [23–25].

In Reference [12], the contribution is divided into two parts. Collecting and manually categorizing a corpus of articles and comments from various Arab political perspectives and creating several versions of it is the first stage. Secondly, use these synthesized datasets to test different classifiers and feature reduction techniques. A model for detecting articles with a political orientation in Arabic was presented in Reference [13]. They highlight the difficulties that still need to be investigated to further our knowledge of subjective sentences, present and discuss the obtained results, and establish the model's main assumptions. To improve the models' ability to identify the articles' orientation, a novel method based on rough set (RS) theory is being used. Their model outperforms competing techniques, as seen in the comprehensive simulation results they provide. The results demonstrate that adding discriminating features considerably enhances the performance of the suggested method.

In Reference [25], a hybrid vector was created for the purpose of classifying political Arabic articles by combining supervised machine learning with feature extraction techniques. Term Frequency (TF) and five grams (unigram, bigram, trigram, 4-gram, and 5-gram) were utilized to build feature vectors that represent the articles. Support Vector Machine (SVM), Naive Bayes (NB), K-Nearest Neighbor (KNN), and Decision Tree (DT) were used for supervised machine learning in the research. In order to conduct sentiment classification, the authors of Reference [26] used state-of-the-art machine learning methods based on artificial neural networks on a main dataset consisting of 1533 words and 2122 phrases extracted from 206 publicly accessible internet postings. Using artificial neural networks and multiple hasher vector sizes, their work was able to classify people's opinion posts. This was in contrast to lexicon-based techniques, which have poor accuracy owing to their computational nature and parameter configuration.

By optimizing the CNN parameters using genetic algorithm-based convolutional neural networks, a novel hybrid classification model for Arabic text is built in Reference [27]. We compare the model to state-of-the-art research and test it on two big datasets. By investigating categories from a text and sorting them into the appropriate class, the study provided in Reference [14] applied Naive Bayes (NB) to analyze opinions. The study looks into how four different Naïve Bayes classifiers—Gaussian, multinomial, complement, and Bernoulli—along with two feature extraction methods—term frequency (TF) and term frequency-inverse document frequency (TF-IDF)—impact the accuracy of Arabic article classification. The applied classifiers have been assessed using precision, recall, F1-score, and the number of accurate predictions.

The authors presented a method in Reference [24] that is mostly based on a collection of local grammars created for the purpose of identifying various patterns of political opinion phrases. Each element in the opinion lexicon has a semantic marker (polarity and intensity) that corresponds to

one of the many opinion words (verbs, adjectives, and nouns) used in these grammars. Their method can determine who is expressing an opinion, who that opinion is directed towards, and whether the statement is good or negative based on its polarity. To identify ideological content in articles, the author in Reference [28] used deep neural networks. The research presents an examination and assessment of the efficacy of deep neural networks in identifying political ideology in news items, authors of articles, and news sources.

2.1 The Need to Extend the Related Work

Classifying political Arabic articles poses several challenges, and while there have been advancements in text classification models, there are still some disadvantages and limitations associated with the previously mentioned classification models. Some advanced classifiers, especially those based on deep learning, can be computationally intensive and may require significant resources for training and inference. This may limit their applicability in resource-constrained environments. Furthermore, many of these models may struggle to capture intricate relationships and dependencies in the data and involve tuning parameters. Finding the optimal values can be challenging and may require careful experimentation. Finally, the difficulty also depends on the nature of the single feature. If it contains sufficient information to discriminate between classes, then the task might be easier. However, if the feature lacks discriminatory power or is noisy, the classifier may struggle.

In many real-world scenarios, using only a single feature for classification may lead to suboptimal results, and it is often beneficial to consider multiple features to improve the model's performance and robustness. Feature engineering-creating new features that capture relevant information-can also play a crucial role in improving the performance of a classifier. When working with multi-level features, CatBoost's ability to efficiently handle categorical variables, prevent overfitting and provide excellent generalization makes it a powerful choice for classification tasks [19,20]. CatBoost automatically scales the features during training, reducing the sensitivity of the model to the scale of input features. This can be beneficial when dealing with multi-level features with varying magnitudes [29,30].

Still, most of the related work on detecting the orientation of political articles is directed to the English language, not Arabic, due to several key factors. NLP tools and libraries are more developed for languages like English, making it easier for researchers to work with these languages. The linguistic and morphological complexity of Arabic makes it more challenging to analyze. This includes issues with tokenization, stemming, and handling different dialects. Yet, understanding political orientation in Arabic requires a deep understanding of cultural and political nuances, which might be more challenging for researchers not familiar with the region. Besides, there are fewer publicly available, annotated datasets for Arabic compared to languages like English, limiting the training data for machine learning models. Finally, the community of researchers working on Arabic NLP is smaller compared to those working on languages like English, Spanish, or Chinese. There are fewer international collaborations focused on Arabic, partly due to geopolitical and linguistic barriers.

3 Methodology

Let D be the set of documents, and C be the set of predefined categories or topics. Each document $d \in D$ is represented as a feature vector X_d in a high-dimensional space. X_d is a numerical representation obtained through feature extraction techniques that convert text data into a format suitable for machine learning algorithms, Y represents the set of possible labels or categories that documents can be assigned to, T be the training set, consisting of labeled examples $(X_d, y_d), y_d \in Y$. A text classification model is represented by a function $f: X \rightarrow Y$ that maps feature vectors to labels. The goal is to learn

this function from the training data. The model is trained by optimizing its parameters based on the training set. This optimization involves minimizing a suitable loss or objective function that measures the difference between the predicted labels and the true labels in the training set.

Once the model is trained, it can be used to predict the category of new, unseen documents. Given a new document with features X_{new}, the model predicts its label y_{pred} by applying the learned function f. Mathematically, text classification can be expressed as finding the optimal parameters θ of the function f that minimizes the loss function:

$$\theta^* = argmin_\theta \mathcal{L}\left(f\left(X_d;\theta\right), y_d\right), \tag{1}$$

where $\mathcal{L}$ is the loss function, $f\left(X_d;\theta\right)$ is the predicted label, and y_d is the true label. The goal is to generalize the learned model to make accurate predictions on new, unseen text documents [10,11].

Political Arabic orientation classification is a challenging task due to the complexities of the Arabic language and the diverse nature of political expressions. In this study, we propose a widespread approach leveraging multi-level features to accurately classify Arabic text into different political orientations. It aids in categorizing content based on the political ideologies it reflects, enabling better analysis and comprehension of public discourse. Our model integrates a CatBoost classifier and multi-level features that combine Bag of Word (BOW), Term Frequency-Inverse Document Frequency (TF-IDF), and Hashing Vectorization, providing a nuanced understanding of political content. The political orientation will be categorized into specific labels, such as conservative, reform, revolutionary, or any other relevant categories [12,22,31,32].

The advantages of using multi-level features in this domain stem from their ability to capture diverse aspects of the data, handle heterogeneous information, and enhance the model's generalization and discriminative power [33]. The merging approach to building multi-level features provides a powerful means of leveraging diverse information sources and enhancing model capabilities in terms of interpretability and adaptability to complex datasets. The model gains the ability to understand intricate patterns and dependencies that may not be apparent when considering each feature type in isolation. This approach is particularly beneficial in tasks where the text data is complex and exhibits multiple levels of information. Fig. 2 depicts the main steps of the suggested model and how they interact with each other. The following subsection discusses each step in detail.

Step 1: Data Collection

Data collection is the process of gathering and acquiring information from various sources for analysis, interpretation, and decision-making. It involves systematic and organized methods to collect relevant data that can be used to answer specific questions or solve problems. Let N be the entire group or set of individuals, items, or events that are of interest for a particular study or analysis, n is the subset of the population selected for observation or measurement. The goal is often to collect data from a representative sample that accurately reflects the characteristics of the entire population $X = \{x_1, x_2, \ldots, x_n\}$. Various methods can be used to collect data, including surveys, experiments, interviews, observations, and more. In our case, the historical data-based collection technique, data that has been previously collected and labelled for similar classification tasks, is utilized. This can be valuable for training models when benchmark datasets are not feasible.

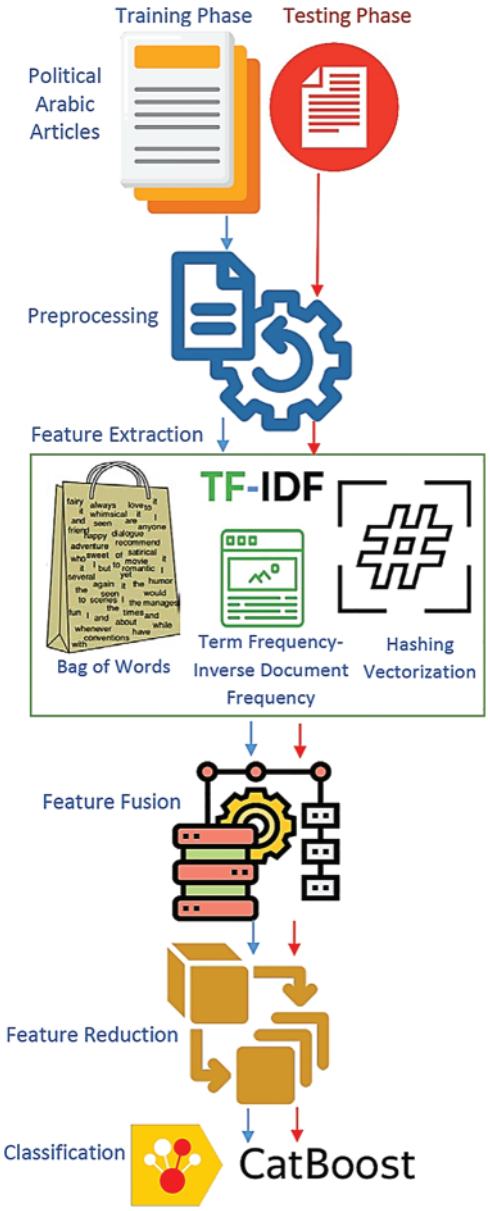

Figure 2: Flowchart of the proposed multilevel features–based CatBoost classifier to detect political Arabic articles orientation

We use a curated dataset containing Arabic text samples labelled with their corresponding political orientations. This study uses the same dataset described in [31], which includes 511 Arabic political articles, to form a dataset from a number of sources. The collected articles are classified into three domains, which are: Conservative (C) 158 articles, Reform (R) 180 articles, and Revolutionary (Re) 173 articles. In general, building a high-quality dataset for classification is crucial to the success of machine learning models. The dataset's quality directly influences the model's performance, generalization, and ability to make accurate predictions. Here are key criteria and considerations, as stated in [31], for building datasets for classification: (1) Balanced Classes: aim for a balanced distribution of classes

within the dataset. An imbalanced dataset can lead to biased models that perform poorly for minority classes. (2) Sufficient Size: the dataset should be large enough to capture the variability of the real-world problem. Insufficient data may result in overfitting, where the model performs well on training data but poorly on new, unseen data. (3) Representativeness: The dataset should be representative of the overall population or distribution you are trying to model. Avoid biases that may skew the model's predictions. (4) Cross-Validation: Consider using techniques like k-fold cross-validation to assess the model's robustness and performance across different subsets of the data.

Step 2: Data Preprocessing

Before applying a classification model, data preprocessing is crucial to ensure that the text data is in a suitable format for training and testing. Data preprocessing helps to clean and remove these noisy elements, making the text more focused and easier for the model to interpret. Cleaned and preprocessed data accelerates the training process by reducing the complexity and size of the feature space. This efficiency is especially important when working with large datasets [34,35]. Let D represents the raw dataset, organized as a matrix where rows correspond to instances or observations and columns correspond to features or variables that can be expressed as:

$$D = \begin{bmatrix} x_{11} & x_{12} & \cdots & x_{1m} \\ x_{21} & x_{22} & \cdots & x_{2m} \\ \vdots & \vdots & \ddots & \vdots \\ x_{n1} & x_{n1} & \cdots & x_{nm} \end{bmatrix}, \tag{2}$$

Each x_{ij} represents the value of the j-th feature for the i-th instance. Data preprocessing involves a sequence of operations on the original dataset D to obtain a processed dataset $D_{processed}$ in which:

$$D_{processed} = Operation_k(Operation_{k-1}(\ldots.Operation_2\,(Operation_1\,(D)\ldots.)), \tag{3}$$

Each $Operation_i$ represents a specific data preprocessing step. The choice of preprocessing steps depends on the nature of the data and the requirements of the specific analysis or machine learning task. In our case, there are a number of key preprocessing steps (operations) for Arabic text that include [25]:

- Tokenization: The first step in preprocessing text is to break it down into individual words, or tokens. For Arabic language tokenization, the process is influenced by the specific characteristics of the language, which include the presence of root-based morphology and the right-to-left script. Arabic script does not rely on explicit spaces between words, making word tokenization non-trivial. Traditional white-space-based tokenization may not be sufficient. The suggested model utilizes specialized Arabic language tokenization libraries or tools. For example, the Farasa Arabic NLP tool provides a word segmenter for Arabic, and the Natural Language Toolkit (NLTK) does provide some support for Arabic language processing, but it's important to note that its coverage may not be as extensive as for some other languages. However, you can still leverage NLTK for certain tasks in Arabic text processing [36].
- Cleaning: This step focused on removing any non-Arabic characters or punctuation, URLs, special characters, irrelevant symbols, or email addresses that are contained in the Arabic text datasets. Arabic text may contain diacritics (small markings above or below characters) that indicate vowels. You may choose to remove diacritics to simplify the text while still preserving the essential information.

- Normalization: normalization in the context of text classification typically refers to the process of standardizing or transforming the text data in a way that allows for consistent and meaningful comparisons across different documents. Arabic has different forms for the same letter, such as the final and isolated forms. So, this study used normalizing to improve the consistency of Arabic text datasets.
- Stop word removal: Arabic has a set of common words, such as conjunctions, prepositions, and pronouns. Removing these stop words can help reduce the noise in the Arabic text datasets and improve the performance of the selected model.
- Stemming: Stemming is a text normalization process that involves reducing words to their base or root form. For the Arabic language, stemming is particularly important due to the rich morphology of Arabic words. In our case, the Information Science Research Institute's (ISRI) root stemmer and light stemmer [37] are used to stem Arabic words in the given text. In summary, the main differences between an ISRI root stemmer and a light stemmer lie in their complexity, accuracy, and the linguistic nuances they capture. The ISRI Root Stemmer tends to be more accurate and linguistically sophisticated, while a light stemmer sacrifices some accuracy for simplicity and efficiency. The choice between them depends on the specific requirements of the NLP task and the trade-offs between accuracy and performance.

Step 3: Feature Extraction

Feature extraction is a critical step in natural language processing (NLP) and machine learning tasks, as it involves converting raw text data into a format suitable for machine learning models. The choice of features can significantly impact the model's performance, so it's often a beneficial practice to experiment with different feature sets and techniques to find the most effective combination for your specific use case [38]. The criteria for choosing features depend on the specific task, dataset characteristics, and the nature of the text. The criteria for determining the relevance of features in the context of text analysis include: (1) Information Gain: Evaluate how much information a feature provides about the target variable. Features with higher information gain are more relevant. (2) Correlation: Analyze the correlation between features and the target variable. Features with a higher correlation may have more predictive power, and (3) Feature Redundancy: Remove redundant features that do not add new information to the model. High feature redundancy can lead to overfitting [39]. Herein, the suggested system employs three commonly used and effective approaches for feature representation in Arabic text classification [38,39]:

- Bag of Words (BOW): This is a popular approach for text feature extraction. It simplifies text data by treating it as an unordered set of words and their frequencies in a document. Let's consider a collection of N documents, and each document is represented as a sequence of words. The goal is to represent each document as a vector based on the frequency of words in a predefined vocabulary. Create a vocabulary V, which is the set of all unique words across all documents. Create a Document-Term Matrix X, where each row represents a document and each column represents a word in the vocabulary:

$$X_{ij} = tf_{ij}, \tag{4}$$

X_{ij} represents the term frequency of word j in document i, and tf_{ij} is the count of word j in document i. Each document is now represented as a vector in a high-dimensional space, where the dimensionality

is equal to the size of the vocabulary.

$$v_i = [X_{i1}, X_{i2}, \ldots\ldots X_{i|V|}], \tag{5}$$

v_i is the BoW vector for the i-th document. In this case, each document is represented by a fixed-length vector, regardless of its length. BOW has limitations, such as the loss of word order information and the inability to capture semantic relationships between words.

– Term Frequency-Inverse Document Frequency (TF-IDF): TF-IDF is a numerical statistic that reflects the importance of a word in a document relative to its importance in a collection of documents, often a corpus. The TF-IDF score for a term t in document d is the product of its term frequency and inverse document frequency:

$$TF - IDF\,(t, d, D) = TF(t, d) \times IDF(t, D), \tag{6}$$

$$TF\,(t, d) = \frac{Number\ of\ occurrences\ of\ termtin\ document\ d}{Total\ number\ of\ terms\ in\ document\ d}, \tag{7}$$

$$IDF\,(t, D) = log\left(\frac{Total\ number\ of\ documents\ in\ Corpus\ D}{Number\ of\ documents\ containing\ term\ t + 1}\right). \tag{8}$$

This score reflects both how often the term appears in the document and how rare the term is across the entire corpus. In large-scale corpora, the computation of IDF values can become computationally expensive. Managing and updating the IDF values for a growing corpus might pose challenges. Furthermore, synonyms (different words with similar meanings) and polysemy (a word having multiple meanings) may not be appropriately addressed by TF-IDF. Variations of a word might be treated as different terms. Like the bag-of-words model, TF-IDF does not consider the order of words in a document. It treats documents as unordered sets of words, losing information about word sequences and relationships.

– Hashing Vectorization (HV): HV is a technique used to convert text data into fixed-size vectors using hash functions. This method is particularly useful for high-dimensional data, such as text, where the number of unique features (words) can be very large. Let's consider a document d with a set of features F, where each feature is a word. The goal is to represent the document as a fixed-size vector using a hash function h. Apply a hash function h to each feature in the document. The hash function maps each feature to an index in a fixed-size vector space V:

$$h\colon F \to \{1, 2, \ldots, |V|\}, \tag{9}$$

The resulting vector space V has a fixed number of dimensions, determined by the desired size of the vector. Create a vector v_d representing the document d based on the hashed features. Initialize the vector with zeros and increment the values at the indices corresponding to the hashed features $v_d = [0, 0, \ldots 0]$. For each f in F, update the vector:

$$v_d\,[h(f)] = v_d\,[h(f)] + 1, \tag{10}$$

The resulting vector v_d is a fixed-size representation of the document. While hashing vectorization is a useful technique for handling high-dimensional data, it comes with certain limitations. In the context of hashing vectorization, collisions can lead to different features being represented by the

same index in the vector, introducing ambiguity. The effectiveness of hashing vectorization depends on the quality of the chosen hash function. A poorly chosen hash function may increase the likelihood of collisions, diminishing the performance of the vectorization. In our case, MD5 is employed as it is very fast to compute.

Step 4: Feature Fusion

Feature fusion, also known as feature combination or feature concatenation, is a process where multiple features are combined to create a new feature. This technique is often used in machine learning and data analysis to create more informative or comprehensive representations of the data [40,41]. Let X be a dataset with n samples, and let $X^{(1)}, X^{(2)}, \ldots, X^{(m)}$ be individual feature matrices, where each X^i has n samples and v_i features. The feature fusion, denoted as X_{fusion}, is obtained by concatenating the individual feature matrices along the feature dimension. If X^i is of shape $\times v_i$, the feature fusion is of shape $\times \sum_{i=1}^{m} v_i$.

$$X_{fusion} = \left[X^{(1)}, X^{(2)}, \ldots, X^{(m)}\right] \in \mathbb{R}^{n \times m}. \tag{11}$$

Boosting features using combination, or feature fusion, can offer several advantages in the context of machine learning and data analysis. Different features may carry complementary information. Combining features from different sources or types can enhance the model's ability to discriminate between different classes or categories. Each feature type may contribute unique discriminative information. Combining features allows the model to avoid redundancy in the dataset. Redundant or highly correlated features can be merged to prevent multicollinearity issues, which might affect model stability.

Step 5: Feature Reduction using SVD

Singular Value Decomposition (SVD) is a matrix factorization technique that can be used for feature reduction or dimensionality reduction. It is commonly applied to decompose a matrix into three other matrices, which can then be truncated to retain only a subset of the most significant features. This process is particularly useful in scenarios where the original dataset has a large number of features and one wants to reduce dimensionality while preserving the most important information [42]. Let's consider a matrix X of dimensions $n \times m$, where n is the number of samples and m is the number of features. The SVD of X is given by:

$$X = U \sum \xi^T, \tag{12}$$

U is an $n \times n$ orthogonal matrix, Σ is an $n \times m$ diagonal matrix with singular values on the diagonal, and ξ^T is the transpose of an $m \times m$ orthogonal matrix. The diagonal entries of Σ represent the singular values, and they are arranged in descending order. The singular values capture the importance of the corresponding singular vectors in reconstructing the original matrix. To reduce features using SVD, only the top k singular values and their corresponding columns in U and ξ^T are retained. The reduced representation of X, denoted as X_k, is obtained as:

$$X_k = U_k \sum_k \xi_k^T, \tag{13}$$

U_k is a $n \times k$ matrix, Σ_k is a $k \times k$ diagonal matrix with top k singular values, and ξ_k^T is the transpose of an $K \times m$ matrix. The reduction to k features retains the most significant information in the original dataset while discarding less important components.

Step 6: CatBoost Classifier

The CatBoost algorithm is an extension of traditional gradient boosting methods with specific optimizations for handling categorical features [17,18]. The objective function in CatBoost is a combination of a loss function and a regularization term. The goal is to minimize this objective function during the training process.

$$\mathcal{L}(\theta) = \sum_{i=1}^{n} l(y_i, F(X_i, \theta)) + \sum_{j=1}^{T} \Omega(f_i), \tag{14}$$

θ is the model parameters, n is the number of training samples, X_i is the feature vector for the i-th sample, y_i is the true label for the i-th sample, $F(X_i, \theta)$ is the model prediction for the i-th sample, l is the loss function measuring the difference between true labels and predictions, T is the number of trees in the ensemble, f_i is the j-th tree, and $\Omega(f_i)$ is the regularization term for the j-th tree. Herein, log-likelihood loss (logloss) is employed as a loss function for classification. CatBoost employs gradient boosting, building trees sequentially to minimize the objective function. The optimization process involves the following steps:

(1) Initialize Model: Initialize the model with a constant prediction (the log odds for classification).
(2) Iterative Tree Building: Iteratively build decision trees to correct the errors made by the existing ensemble. The trees are added to the ensemble one at a time.
(3) Gradient Calculation: Calculate the negative gradient of the loss function with respect to the current ensemble's predictions. This is the "gradient" part of gradient boosting.
(4) Tree Building: Build a new tree that predicts the negative gradient. CatBoost uses an ordered boosting strategy and builds symmetric trees to improve training efficiency.
(5) Regularization: Apply regularization to control the complexity of the trees and prevent overfitting.
(6) Update Model: Update the model by adding the newly built tree to the ensemble with a certain weight.
(7) Repeat: Repeat the process for a specified number of iterations or until a convergence criterion is met. Algorithm 1 summarizes the classification steps for training and testing data.

Algorithm 1: Classification procedure for training and testing data

Input: Set of training and testing samples
$Tr = \{(x_i, l_j) \mid i = \{1, 2, 3, \ldots, n\}, j = \{1, 2, 3\}\}$, set of training samples and class
$Te = \{t_i \mid i = \{1, 2, 3, \ldots, e\}$, set of e test samples
Initialization:
Initial predictions for sample
Create dataset with categorical features
Boosting:
For each iteration:
- Fit decision tree to negative gradient
- Update prediction with learning rate correction
- Update categorical feature statistics

Apply L2 regularization to tree leaf value

(Continued)

Algorithm 1 (continued)

Prediction:

Initialize tree prediction

For each tree:

Add trees to prediction to test prediction

Output: Set of predicted class

$Y = \{ y_i | i \in \{1, 2, 3, \ldots, e\}\}$ − the test samples in Tr with the set of predicted class lables.

Gradient boosting, including popular implementations like XGBoost, LightGBM, and CatBoost, allows users to define custom loss functions based on the specific requirements of the problem. This flexibility makes it adaptable to a wide range of tasks, including regression, classification, and ranking [43–46]. Within this category, the CatBoost ensemble classifier has the following key advantages: (1) CatBoost often provides good default hyperparameters, reducing the need for extensive tuning compared to some other gradient boosting frameworks. This can be advantageous for users who want to quickly build a model without delving deeply into hyperparameter optimization. (2) CatBoost uses a method to automatically select the learning rate, which can simplify the hyperparameter tuning process compared to other algorithms where selecting an appropriate learning rate might be more challenging. (3) CatBoost incorporates regularization techniques to prevent overfitting. It has built-in support for handling overfitting, making it more robust in situations where overfitting might be a concern. (4) One of the significant distinctions is CatBoost's ability to handle categorical features efficiently. Unlike traditional ensemble classifiers that may require one-hot encoding of categorical variables, CatBoost can directly process categorical features without the need for extensive preprocessing. (5) Feature Importance Estimation: CatBoost offers built-in methods for feature importance estimation, aiding in model interpretation.

4 Results and Discussions

The Python (Release 3.10.10) implementation was used to examine the proposed system's efficiency. A modularly constructed prototype verification approach was used and evaluated on a Dell PC computer, which had the following characteristics: The system requirements are Intel (R), Core (TM) i7 CPU, L640 @ 2.31 GHz 2.31 GHz with RAM: 4 GB. System type: 64-bit operating system. Microsoft Windows 8.1 Enterprise as the running operating system, and a Hard Disk: 500 GB. The used dataset, which is a collection of 511 Arabic political articles sourced from various sources, is detailed in [31]. The collected articles are classified into three domains, which are: conservative (C), reform (R), and revolutionary (Re). Table 1 shows the data distribution among the three categories: C, R, and Re. It lists the overall count for each category and the number of samples or instances available for testing and training within each class. Furthermore, Fig. 3 shows the t-SNE projection of 511 documents. t-SNE (t-distributed stochastic neighbor embedding) is a dimensionality reduction technique commonly used for visualizing high-dimensional data in a lower-dimensional space. It's often used as a preprocessing step for data visualization before applying machine learning algorithms.

Table 1: Training and testing sample size for each class

Class	Training samples number	Testing samples number	Total samples
C	110	48	158
R	130	50	180
Re	117	56	173
Total	357	154	511

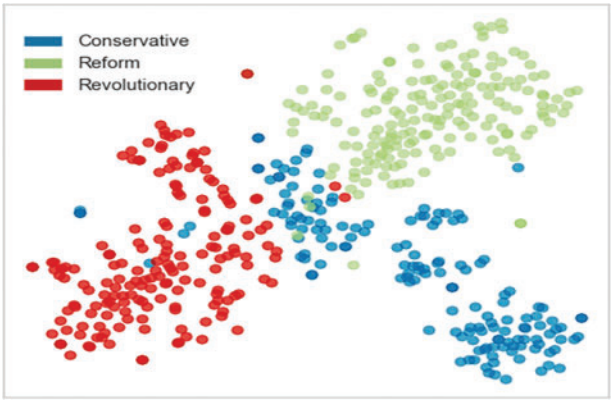

Figure 3: t-SNE projection of 511 documents

In our case, four versions of the dataset are employed to assess the role of the preprocessing phase for accuracy enhancement. D1 version for the original dataset; D2 version for the dataset with preprocessing that includes tokenization, stop word elimination, and normalization; D3 version for the dataset with pre-processing as in D2 with Root Stemmer; and finally D4 version for the dataset with preprocessing as in D2 with Light Stemmer. In the context of classification issues, typical performance assessment measures include accuracy, precision, recall, and F1-score; see Table 2. Kappa is a statistic that measures inter-rater agreement for categorical items. In the context of classification tasks, it's often used to assess the agreement between the predicted and actual classifications. It provides a measure of agreement that goes beyond simple accuracy, particularly when the classes are imbalanced. The parameters employed by the CatBoost algorithm in our research are shown in Table 3.

Table 2: Evaluation metrics [25]

Metric	Equation	Parameters
Accuracy	$\frac{TN+TP}{TN+TP+FN+FP}$	True negatives (TN), False negatives (FN) True positives (TP) False positives (FP)
Recall	$\frac{TP}{TP+FN}$	
Precision	$\frac{TP}{FP+TP}$	

(Continued)

Table 2 (continued)

Metric	Equation	Parameters
F1-score	$2 * \frac{Recall * Precision}{Recall + Precision}$	
Kappa	$k = \frac{p_o - p_e}{1 - p_o}$	p_o is the observed agreement, i.e., the accuracy of the classifier; p_e is the expected agreement, i.e., the agreement that would be expected by chance

Table 3: CatBoost algorithm parameters [46]

Parameter	Value	Description
Iterations	250	Max count of trees
Learning rate	0.03	Used in update to prevents over fitting
Depth	6	Depth of a tree. This is according to experiments
l2_leaf_reg	3	Coefficient at the L2 regularization term of the cost function
Loss function	Log loss	The metric to use in training and also selector of the machine learning problem to solve
Feature border type	Greedy log sum	The binarization mode in numeric features binarization. Used in the preliminary calculation. Possible values
Leaf estimation iterations	1	The number of steps in the gradient when calculating the values in the leaves
Leaf estimation method	Gradient	The method used to calculate the values in the leaves
Random seed	25	Random number seed
Max leaves	31	The maximum leaf count in resulting tree. This parameter is used only for Loss guide growing policy

4.1 Check Classifier Performance Using Training Set

Overall, classifier learning using training data sets forms the foundation of many machine learning applications, enabling automated decision-making, pattern recognition, and generalization from data. However, it's crucial to ensure high-quality training data, appropriate feature engineering, and careful model evaluation to maximize the benefits of classifier learning. Table 4 shows the proposed model performance metrics for varying feature types for different training datasets. As expected, utilizing

the fused feature vector outperforms the other individual feature vectors by an average of 3%–5% for all performance metrics for all training datasets. Overall, the BOW vectorization technique offers simplicity, efficiency, flexibility, and interpretability, making it a valuable tool in various NLP applications and text analysis tasks. TF-IDF offers improved feature representations compared to simple BOW techniques by emphasizing important terms while downplaying common ones. Hashing vectorizer offers several advantages for text classification, including scalability, online learning support, dimensionality reduction, and parallelization capabilities. However, feature fusion offers several advantages in machine learning, including the integration of complementary information, enhanced discriminative power, reduction of feature redundancy, improved robustness, efficient model training and inference. These advantages make it a valuable technique for building more effective and robust machine learning models.

Table 4: Proposed model performance metrics for varying feature types for different training datasets

Feature type	Dataset	Precision	Recall	F1-score	Accuracy
BOW vector	D1	94.70	94.83	94.74	94.76
	D2	94.43	94.62	94.48	94.48
	D3	96.91	96.93	96.90	96.88
	D4	95.55	95.67	95.60	95.59
TF-IDF vector	D1	95.86	95.93	95.89	95.88
	D2	95.85	95.97	95.89	95.88
	D3	96.81	96.83	96.80	96.83
	D4	96.14	96.18	96.16	96.15
Hash vector	D1	94.71	94.90	94.76	94.76
	D2	94.99	95.16	95.06	95.04
	D3	96.51	96.56	96.47	96.49
	D4	95.57	95.63	95.60	95.51
Fused feature vector	D1	97.69	97.82	97.74	97.75
	D2	97.42	97.61	97.48	97.47
	D3	97.52	97.71	97.58	97.67
	D4	98.54	98.67	98.59	98.59

4.2 Check Classifier Performance Using Test Set

Overall, using a test set in classifier learning is essential for evaluating generalization performance, assessing model performance, identifying overfitting, tuning parameters and selecting models, validating results, improving generalization, and guiding future iterations of the classifier. These benefits are crucial for building effective and reliable classifiers for machine learning tasks. As revealed from Table 5, utilizing individual feature types for classification, the system exhibits lower accuracy in the case of test data in all performance measures compared to a training set, reflecting issues with generalization by on average 5%. Generalization refers to how well a machine learning model performs on new, unseen data. In contrast, using a fused feature vector for classification, the model's performance metrics are almost the same, with very slight differences.

Table 5: Proposed model performance metrics for varying feature types for different test datasets

Feature type	Dataset	Precision	Recall	F1-score	Accuracy
BOW vector	D1	93.59	93.39	93.39	93.51
	D2	94.79	94.58	94.61	94.81
	D3	93.59	93.37	93.45	93.51
	D4	**96.04**	**96.09**	**96.06**	**96.10**
TF-IDF vector	D1	92.49	92.06	92.05	92.21
	D2	94.19	93.92	93.93	94.15
	D3	94.88	94.65	94.74	94.80
	D4	**94.05**	**94.06**	**94.04**	**94.15**
Hash vector	D1	89.77	89.38	89.44	89.61
	D2	88.75	88.15	88.16	88.31
	D3	91.91	91.33	91.53	91.56
	D4	**93.42**	**93.42**	**93.34**	**93.51**
Fused feature vector	D1	97.54	97.61	97.44	97.35
	D2	97.12	97.21	97.38	97.27
	D3	97.22	97.51	97.38	97.37
	D4	**98.24**	**98.47**	**98.49**	**98.32**

Overall, lower accuracy on the test set compared to the training set can be attributed to factors such as overfitting, data distribution discrepancies, imbalanced classes, and noise and variability. Overfitting occurs when a model learns to memorize the training data rather than capturing the underlying patterns. Models that are too complex or have too many parameters are prone to overfitting. When a model overfits, it performs well on the training set but fails to generalize to new, unseen data, resulting in lower accuracy on the test set. The test set may have a different distribution of data compared to the training set. If the test set contains samples from different populations or is collected under different conditions, the model's performance may suffer. Models trained on one distribution may not generalize well to data from a different distribution. Addressing these factors through techniques such as regularization, cross-validation, data preprocessing, and model selection can help improve the model's performance on unseen data.

Furthermore, preprocessing and stemming play critical roles in improving classification accuracy by standardizing, simplifying, and enhancing the quality of the input data, which enables the classification algorithm to better identify relevant patterns and make more accurate predictions. The results in Table 5 confirm that utilizing preprocessing steps followed by streaming based on Light Stemmer software on the dataset yields the best performance metrics. In general, overstemming occurs when a stemming algorithm removes too many characters from a word, resulting in the loss of meaningful information or ambiguity. Light stemmers tend to be more conservative in their stemming rules, reducing the risk of overstemming and preserving the semantic meaning of words more effectively. Light stemmers tend to be less aggressive in their stemming approach compared to root stemmers. They often preserve more of the original word's morphology, which can be beneficial for tasks where retaining some level of linguistic information is important, such as sentiment analysis or topic modeling [37].

The reasons why certain types of classes obtain the best classification accuracy in machine learning models can vary depending on class separability in which some classes might be inherently more separable from others in the feature space. Classes that have clearer boundaries in the feature space make it easier for the model to distinguish between them, resulting in higher accuracy. Depending on the document's political class and feature type, Tables 6 to 9 provide the suggested models' performance metrics for various datasets. The results reveal that, on average, the Re class obtains the best classification metrics, with an average rise of 2% within each dataset. One possible justification for these results stems from the distinctiveness of revolutionary texts. Revolutionary texts often contain unique language, rhetoric, and themes that distinguish them from other types of political discourse. These texts may frequently mention overthrowing existing systems, advocating for radical change, or promoting revolutionary ideologies. The distinctiveness of such language makes it easier for machine learning models to classify them accurately.

Table 6: Proposed model performance metrics for each political class for different test datasets based on BOW feature vector

Dataset	Class	Precision	Recall	F1-score	Accuracy
D1	C	0.87	0.94	0.90	93.506
	R	0.98	0.90	0.94	
	Re	0.96	0.96	0.96	
D2	C	0.90	0.94	0.92	94.805
	R	0.98	0.90	0.94	
	Re	0.97	100	0.98	
D3	C	0.96	0.92	0.94	93.506
	R	0.92	0.92	0.92	
	Re	0.93	0.96	0.95	
D4	C	0.96	0.96	0.96	96.104
	R	0.94	0.96	0.95	
	Re	0.98	0.96	0.97	

Table 7: Proposed model performance metrics for each political class for different test datasets based on TF-IDF feature vector

Dataset	Class	Precision	Recall	F1-score	Accuracy
D1	C	0.83	0.94	0.88	92.208
	R	0.98	0.86	0.91	
	Re	0.96	0.96	0.96	
D2	C	0.88	0.94	0.91	94.156
	R	0.98	0.88	0.93	
	Re	0.97	100	0.98	
D3	C	0.98	0.94	0.96	94.805
	R	0.92	0.92	0.92	
	Re	0.95	0.98	0.96	

(Continued)

Table 7 (continued)

Dataset	Class	Precision	Recall	F1-score	Accuracy
D4	C	0.92	0.94	0.93	94.156
	R	0.94	0.92	0.93	
	Re	0.96	0.96	0.96	

Table 8: Proposed model performance metrics for each political class for different test datasets based on hash feature vector

Dataset	Class	Precision	Recall	F1-score	Accuracy
D1	C	0.81	0.88	0.84	89.61
	R	0.96	0.86	0.91	
	Re	0.93	0.95	0.94	
D2	C	0.77	0.90	0.83	88.312
	R	0.93	0.82	0.87	
	Re	0.96	0.93	0.95	
D3	C	0.98	0.90	0.93	91.558
	R	0.88	0.88	0.88	
	Re	0.90	0.96	0.93	
D4	C	0.88	0.96	0.92	93.506
	R	0.94	0.88	0.91	
	Re	0.98	0.96	0.97	

Table 9: Proposed model performance metrics for each political class for different test datasets based on fused feature vector

Dataset	Class	Precision	Recall	F1-score	Accuracy
D1	C	0.88	0.95	0.91	97.31
	R	0.95	0.93	0.94	
	Re	0.97	0.94	0.96	
D2	C	0.84	0.92	0.90	97.24
	R	0.93	0.90	0.94	
	Re	0.97	0.97	0.97	
D3	C	0.98	0.95	0.93	97.62
	R	0.87	0.88	0.89	
	Re	0.97	0.97	0.97	
D4	C	0.88	0.94	0.92	98.30
	R	0.94	0.94	0.91	
	Re	98.24	98.47	98.49	

Another possible explanation for these results is that the boundaries between revolutionary texts and other political classes (e.g., Conservative and Reform) may be more well-defined. This clear distinction makes it easier for models to learn to differentiate between revolutionary texts and others, leading to higher accuracy. Finally, the features used for classification may be particularly well-suited for distinguishing revolutionary texts. For example, certain keywords, phrases, or topics commonly associated with revolutionary movements may be strong indicators for this class type, leading to better accuracy.

The Kappa values for different feature extraction methods on different datasets, used for training and testing phases, are shown in Table 10. As expected, in the learning phase, we can observe that the suggested model based on a fused feature vector performs better than other feature vectors. In the testing phase, the Kappa values for individual feature vectors are lower than their corresponding values in the learning phase. In contrast, utilizing our model, the values in both phases are almost equal. It can be observed from the table that Kappa values are closer to 1, indicating a high level of agreement between the raters beyond what would be expected by chance. This suggests that the raters are consistent in their judgments or classifications. It's often interpreted as a measure of the reliability of the raters in categorizing the items.

Table 10: Proposed model Kappa metric for different datasets based on different types of feature vectors for training and testing phases

Dataset	BOW		TF-IDF		Hash vector		Fused feature vector	
	Learning	Testing	Learning	Testing	Learning	Testing	Learning	Testing
D1	95.63	90.24	96.31	88.29	96.63	84.38	97.64	97.61
D2	95.21	92.18	97.31	91.21	97.05	82.46	97.89	97.83
D3	96.50	90.22	97.11	92.18	97.56	87.28	97.56	97.52
D4	97.89	94.14	98.73	91.21	97.89	90.24	98.89	98.83

From the results shown in Table 11, we can notice that the time taken for training and testing varied across datasets and feature extraction techniques. The size of the dataset can greatly influence the training time. Larger datasets generally require more time to train on, especially for algorithms that need to process the entire dataset multiple times. The execution time for building a feature vector can vary depending on several factors: (1) Dimensionality of the feature vector: The number of features in the vector directly affects the time it takes to build it. Building a feature vector with a large number of features generally takes longer than building one with fewer features. (2) Complexity of Feature Extraction: If the process of extracting features from raw data is computationally intensive or involves complex calculations, it can increase the execution time. The proposed system takes more time to classify as a result of building a reduced fused feature vector. Typically, the testing phase is faster than the training phase because the model has already been trained, and it's primarily a process of inference or prediction on new data. Utilizing parallel processing capabilities can significantly reduce training time.

Table 11: Proposed model classification time for different datasets based on different types of feature vectors for training and testing phases

Dataset	BOW		TF-IDF		Hash vector		Fused feature vector	
	Learning	Testing	Learning	Testing	Learning	Testing	Learning	Testing
D1	21.69	0.70	42.40	0.71	15.90	0.05	46.42	0.77
D2	15.89	0.62	22.18	0.59	13.83	0.06	27.19	0.69
D3	7.23	0.14	12.34	0.14	30.31	0.06	33.45	0.25
D4	13.26	0.37	17.32	0.36	19.15	0.17	20.76	0.43

The relationship between dataset size and overfitting is a fundamental concept in machine learning. Overfitting occurs when a model learns to memorize the training data rather than generalize from it. This typically happens when a model becomes overly complex relative to the amount of training data available. Fig. 4 shows the proposed model's accuracy as a function of dataset size (D4 as an example) to validate its ability to deal with overfitting problems. In our experiment, in each cycle during the training, a new 20% of the training data is added to perform training on the model before applying the model to the testing data. The results reveal that accuracy improves with an increasing number of rounds of training, which involves adding more samples to the dataset. As the dataset size increases, the risk of overfitting typically decreases. With more data, the model has a better chance of learning the true underlying patterns rather than just memorizing noise. Larger datasets provide more diverse examples, which can help the model generalize better to unseen data. However, it's important to note that overfitting can still occur with large datasets if the model architecture is too complex relative to the amount of data or if there are other issues such as insufficient regularization (early stopping is commonly used to prevent models from becoming overly complex and memorizing the training data) [1–3].

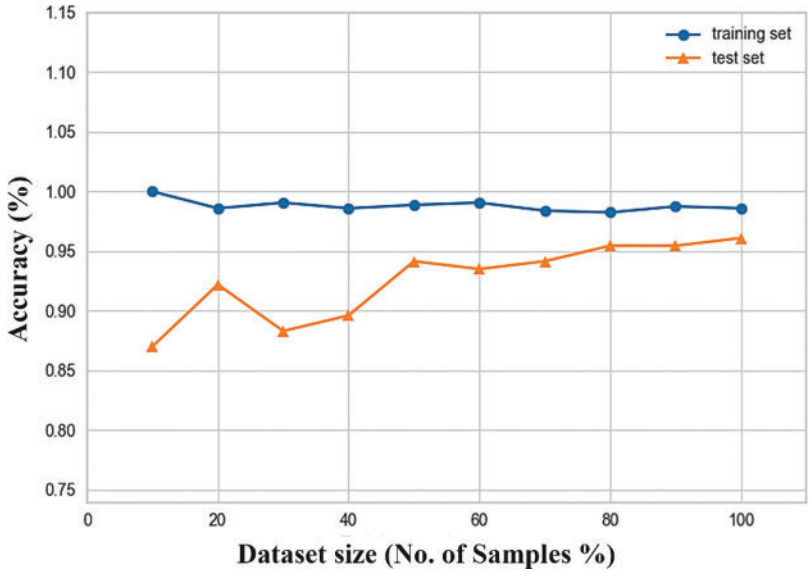

Figure 4: Proposed model performance in terms of accuracy as a function of dataset size

Fig. 5 shows the receiver operating characteristic curve (ROC) of the suggested model for the D4 dataset. The ROC curve plots the trade-off between sensitivity (true positive rate) and specificity (true negative rate) for different threshold values. A model with perfect classification would have a curve that passes through the top-left corner (0, 1) of the plot, indicating 100% sensitivity and 100% specificity. The closer the ROC curve follows the left-hand border and then the top border of the ROC space, the more accurate the test. A curve that is closer to the diagonal line (random guessing) suggests a less accurate model. The area under the ROC curve (AUC-ROC) is often used as a summary measure of a model's performance. An AUC-ROC value of 0.5 indicates that the model is performing no better than random guessing, while a value closer to 1 indicates better performance. The three classes achieved a high value of ROC with 0.98.

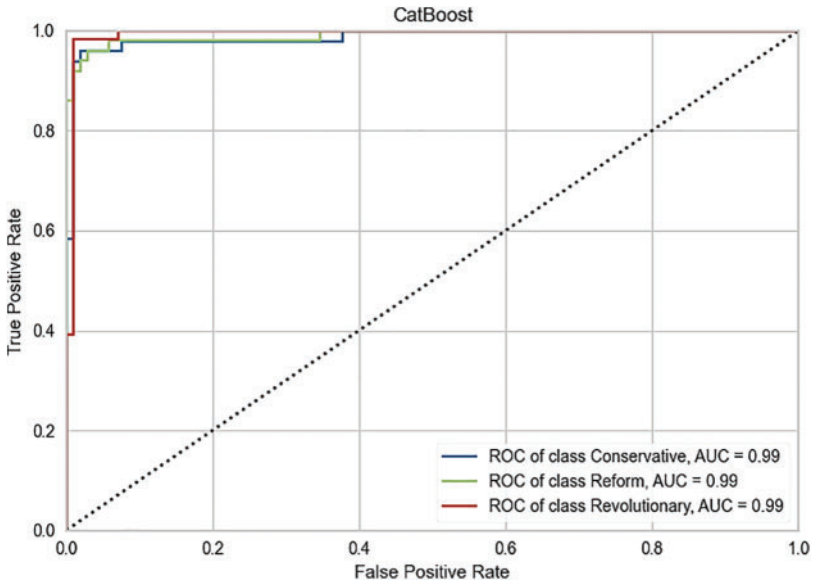

Figure 5: ROC curve for the utilized fused feature vector applied to the D4 dataset

The prediction rate represents the proportion of incorrectly classified instances out of the total number of predictions made by the model. It directly reflects how often the model is making mistakes in its predictions. A lower prediction error rate indicates higher accuracy, meaning that the model makes fewer mistakes in its predictions. Conversely, a higher prediction error rate suggests that the model is less accurate and is making more errors. When classes in the dataset exhibit significant overlap in feature space, distinguishing between them becomes challenging for the model. In regions of overlap, misclassifications are more likely to occur, particularly if the decision boundaries are not well-defined. As revealed in Fig. 6, the reform (R) class had the worst prediction rate. A possible reason for this result is that the R class may have sparse or lack diversity in the dataset, making it difficult for the model to learn discriminative features for this class.

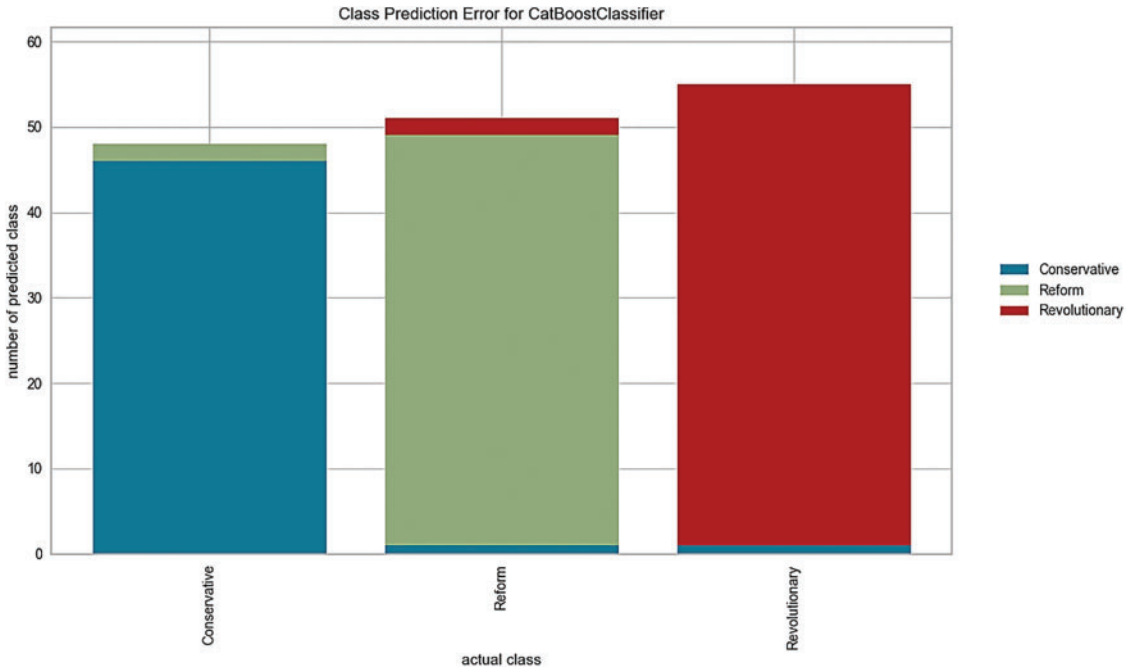

Figure 6: Error prediction for each class applied on D4 using fused feature vector

Table 12 shows the results of using k-fold cross-validation with k = 5 and k = 10 for different feature extraction methods on different datasets. The metrics used for evaluation are mean, and standard deviation (SD). These measures provide insights into the typical performance (mean) and the variability (standard deviation) of the accuracy values. Unlike a single train-test split, k-fold cross-validation reduces the risk of bias in the performance estimate. Since the model is tested on multiple subsets of the data, the evaluation is less dependent on a particular random split. Furthermore, the variability in performance metrics across different folds can provide insights into how the model behaves with different subsets of data. In general, as k increases, the bias of the model performance estimate decreases while the variance increases. A higher k value leads to a lower bias because the model is evaluated on more diverse subsets of the data. However, it also increases the variance because each fold's test set becomes smaller, potentially leading to higher variability in the performance estimates. The results confirm the superiority of the suggested model in both accuracy metrics (higher mean and smaller SD). A higher mean accuracy indicates better overall performance, while a smaller variance or standard deviation suggests more consistent performance across folds.

Table 12: Proposed model accuracy with different k-fold cross-validation as a function for type of features and dataset

Feature extraction	Dataset	Accuracy (k = 5)		Accuracy (k = 10)	
		Mean	SD	Mean	SD
BOW	D1	0.89	0.07	0.92	0.06
	D2	0.91	0.08	0.92	0.06

(Continued)

Table 12 (continued)

Feature extraction	Dataset	Accuracy (k = 5)		Accuracy (k = 10)	
		Mean	SD	Mean	SD
	D3	0.92	0.08	0.95	0.03
	D4	0.92	0.07	0.93	0.05
TF-IDF	D1	0.89	0.08	0.90	0.07
	D2	0.91	0.07	0.92	0.05
	D3	0.94	0.07	0.97	0.03
	D4	0.92	0.08	0.94	0.05
Hash vector	D1	0.87	0.07	0.88	0.06
	D2	0.87	0.09	0.88	0.09
	D3	0.93	0.07	0.96	0.03
	D4	0.92	0.08	0.92	0.07
Fused feature vector	D1	0.95	0.07	0.97	0.05
	D2	0.95	0.08	0.96	0.05
	D3	0.94	0.08	0.95	0.04
	D4	0.97	0.07	0.98	0.05

Table 13 displays the performance metrics of several classifiers. For this experiment, we run it on a whole dataset (511 documents) and swap out the CatBoost classifier from our model—which uses a fused feature vector—with one of the classifiers from Table 13. The proposed model outperforms all other classifiers with an accuracy of 98.10%. The XGBoost and SVM algorithms are also performing well, with an accuracy of 95.45% and 96.10%, respectively. On the other hand, the KNN and Gaussian-NB (Naïve Bayes) classifiers have the lowest accuracy. In general, ensemble methods can capture complex relationships in the data that individual classifiers may struggle to learn. By combining multiple models with different perspectives or modelling techniques, ensemble methods can effectively capture nonlinearities and interactions in the data.

Table 13: Comparative analysis with other classifiers

Classifier	Precision	Recall	F1-score	Kappa	Accuracy
XGBoost	95.41	95.42	95.41	93.17	95.46
k-Nearest Neighbors (KNN)	86.69	82.94	82.75	73.78	82.47
Decision Tree (DT)	84.97	84.85	84.87	77.59	85.07
Random Forest (RF)	95.43	95.27	95.29	93.16	95.46
Support Vector Machine (SVM)	96.04	95.94	95.95	94.14	96.10
Artificial Neural Network (ANN)	96.04	95.94	95.95	94.14	96.10

(Continued)

Table 13 (continued)

Classifier	Precision	Recall	F1-score	Kappa	Accuracy
Stochastic gradient descent (SGD)	95.57	95.37	95.38	93.17	95.46
AdaBoost	92.55	92.08	92.25	88.27	92.21
Bagging	92.37	92.16	92.07	88.30	92.21
Gaussian-NB	80.60	79.23	78.99	69.62	79.87
Bernoulli-NB	88.28	88.02	87.58	81.54	87.66
Multinomial-NB	93.86	93.08	93.14	90.21	93.51
Proposed model	97.04	96.09	96.06	94.16	98.10

While XGBoost and AdaBoost are also powerful algorithms with their own strengths, CatBoost's unique features, such as handling categorical variables, automatic handling of missing values, and efficient optimization techniques, contribute to its ability to achieve high accuracy in many classification tasks. CatBoost incorporates regularization techniques such as L2 regularization and early stopping to prevent overfitting. It dynamically adjusts the tree structure during training to find the optimal balance between model complexity and generalization performance. This helps reduce overfitting and improves the model's ability to generalize to unseen data.

Deep neural networks (DNNs) typically do not explicitly use multi-level feature concatenation because they inherently learn hierarchical feature representations through their layered structure. Through the process of backpropagation and optimization, DNNs automatically learn which features to extract and how to combine them to minimize the loss function. This eliminates the need for manually concatenating features from different levels. Explicitly concatenating features from multiple levels would increase the computational complexity and memory usage of the network [47,48]. This explains why multi-level feature-based classification is better with legacy classifiers than multi-level feature classification with deep models.

In general, the type of language, especially Arabic with its complex morphology and dialectal variations, significantly affects classifiers for political orientation. Addressing these linguistic challenges through utilizing multi–level features are essential for developing effective classifiers for Arabic political text. Effective feature extraction is crucial for text classifiers. In Arabic, extracting meaningful features is more challenging due to the language's morphology and script. Standard techniques like bag-of-words or TF-IDF individually might not capture the nuances of Arabic political discourse effectively. Utilizing supervised target-based text feature augmentation will allow the model to leverage additional information derived from the relationship between text data and the target variable, with the aim of potentially improving its performance [49–51].

5 Conclusions

A new approach to political Arabic text classification utilizing the multi-level feature-boosted CatBoost classifier was suggested in this work. Fusing features from different sources or modalities allows the classifier to capture a more comprehensive representation of the data. This comprehensive representation often contains complementary information that may not be present in individual features alone, enhancing the classifier's ability to discriminate between classes. CatBoost employs

gradient-based optimization techniques to train ensemble models efficiently. By utilizing gradients, it can navigate the search space more effectively, leading to faster convergence and improved performance compared to traditional boosting algorithms. CatBoost typically requires less data to achieve good performance compared to deep neural networks, especially in scenarios where the dataset is small or noisy. Deep neural networks, with their high parameter counts, tend to be more data-hungry and may overfit if the dataset is limited. The results of the study show that the proposed model outperforms other commonly used classifiers, such as ANN for text classification, achieving an accuracy of 98.104%. This suggests that the proposed method is effective for political Arabic text classification and has the potential to be applied to other types of text classification tasks in Arabic. One limitation of the suggested model is the size of the dataset used for evaluation. While the dataset is diverse, larger datasets could provide more robust results. Future research will explore context-aware fusion techniques and scalable fusion techniques to enable efficient fusion across diverse application domains and deployment scenarios.

Acknowledgement: Not applicable.

Funding Statement: The authors received no specific funding for this study.

Author Contributions: Conceptualization, Saad M. Darwish, Abdul Rahman M. Sabri and Dhafar Hamed Abd; methodology, Saad M. Darwish, Abdul Rahman M. Sabri, and Dhafar Hamed Abd; Software, Abdul Rahman M. Sabri, and Dhafar Hamed Abd; validation, Saad M. Darwish, Dhafar Hamed Abd, and Adel A. Elzoghabi; formal analysis, Saad M. Darwish, Dhafar Hamed Abd, and Adel A. Elzoghabi; investigation, Saad M. Darwish, and Dhafar Hamed Abd; resources, Abdul Rahman M. Sabri, and Dhafar Hamed Abd; data curation, Abdul Rahman M. Sabri, and Dhafar Hamed Abd; writing—original draft preparation, Saad M. Darwish, and Abdul Rahman M. Sabri; writing—review and editing, Saad M. Darwish Dhafar Hamed Abd, and Adel A. Elzoghabi; visualization, Abdul Rahman M. Sabri; supervision, Saad M. Darwish, and Abdul Rahman M. Sabri; project administration, Abdul Rahman M. Sabri, and Dhafar Hamed Abd; funding acquisition, Abdul Rahman M. Sabri, and Dhafar Hamed Abd. All authors reviewed the results and approved the final version of the manuscript.

Availability of Data and Materials: The benchmark dataset is available in https://data.mendeley.com/datasets/spvbf5bgjs/2, DOI: 10.17632/spvbf5bgjs.2, accessed on 1 June 2024.

Ethics Approval: Not applicable.

Conflicts of Interest: The authors declare that they have no conflicts of interest to report regarding the present study.

References

[1] J. Lee, J. Cha, and E. Park, "Data-driven approaches into political orientation and news outlet discrimination: The case of news articles in South Korea," *Telematics Inform.*, vol. 85, no. 2, pp. 1–11, 2023. doi: 10.1016/j.tele.2023.102066.

[2] S. Lee, J. Kim, D. Kim, K. Kim, and E. Park, "Computational approaches to developing the implicit media bias dataset: Assessing political orientations of nonpolitical news articles," *Appl. Math. Comput.*, vol. 458, pp. 1–12, 2023. doi: 10.1016/j.amc.2023.128219.

[3] S. Da-Silva and I. Paraboni, "Politically-oriented information inference from text," *J. Univ. Comput. Sci.*, vol. 29, no. 6, pp. 1–27, 569, 2023. doi: 10.3897/jucs.96652.

[4] A. Matsumura, R. Garg, M. Hussain, and M. Matsumura, "Political orientation of online media sources and reporting of COVID-19 vaccine myocarditis," *PLos One*, vol. 19, no. 1, pp. 1–15, 2024. doi: 10.1371/journal.pone.0296295.

[5] A. Gohary, F. Madani, E. Chan, and S. Tavallaei, "Political ideology and fair-trade consumption: A social dominance orientation perspective," *J. Bus. Res.*, vol. 156, no. 4, pp. 1–11, 2023. doi: 10.1016/j.jbusres.2022.113535.

[6] J. Jiang, X. Ren, and E. Ferrara, "Retweet-BERT: Political leaning detection using language features and information diffusion on social networks," in *Proc. Int. AAAI Conf. Web Soc. Med.*, vol. 17, pp. 459–469, 2023. doi: 10.1609/icwsm.v17i1.22160.

[7] J. Valtonen, V. Ilmarinen, and J. Lönnqvist, "Political orientation predicts the use of conventional and complementary/alternative medicine: A survey study of 19 European countries," *Soc. Sci. Med.*, vol. 331, pp. 1–10, 2023. doi: 10.1016/j.socscimed.2023.116089.

[8] S. Kamal, B. Little, J. Gullic, T. Harms, K. Olofsson and A. Bagavathi, "Modeling political orientation of social media posts: An extended analysis," 2023, *arXiv:2311.12323*.

[9] S. Bestvater and B. Monroe, "Sentiment is not stance: Target-aware opinion classification for political text analysis," *Polit. Anal.*, vol. 31, no. 2, pp. 235–256, 2023. doi: 10.1017/pan.2022.10.

[10] A. Olteanu, A. Cernian, and S. Gâgă, "Leveraging machine learning and semi-structured information to identify political views from social media posts," *Appl. Sci.*, vol. 12, no. 24, pp. 1–17, 2022. doi: 10.3390/app122412962.

[11] Z. Indra, A. Setiawan, and Y. Jusman, "Implementation of machine learning for sentiment analysis of social and political orientation in Pekanbaru City," *J. Phy.: Conf. Series*, vol. 1803, no. 1, pp. 1–10, 2021.

[12] R. Abooraig, S. Al-Zu'bi, T. Kanan, B. Hawashin, M. Al- Ayoub and I. Hmeidi, "Automatic categorization of Arabic articles based on their political orientation," *Digit. Invest.*, vol. 25, no. 6, pp. 24–41, 2018. doi: 10.1016/j.diin.2018.04.003.

[13] J. Alwan, A. Hussain, D. Abd, A. Sadiq, M. Khalaf and P. Liatsis, "Political Arabic articles orientation using rough set theory with sentiment lexicon," *IEEE Access*, vol. 9, no. 1, pp. 24475–24484, 2021. doi: 10.1109/ACCESS.2021.3054919.

[14] D. Abd, A. Sadiq, and A. Abbas, "Political articles categorization based on different Naïve Bayes models," in *Proc. Int. Conf. Appl. Comput. Support Ind.: Innov Technol.*, Cham, Springer International Publishing, 2019, pp. 286–301.

[15] F. Zhao, J. Zhang, Z. Chen, X. Zhang, and Q. Xie, "Topic identification of text-based expert stock comments using multi-level information fusion," *Expert Syst.*, vol. 40, no. 2, pp. 1–15, 2023. doi: 10.1111/exsy.12641.

[16] H. Bhuyan, M. Saikiran, M. Tripathy, and V. Ravi, "Wide-ranging approach-based feature selection for classification," *Multimed. Tools Appl.*, vol. 82, no. 15, pp. 23277–23304, 2023. doi: 10.1007/s11042-022-14132-z.

[17] J. Hancock and T. Khoshgoftaar, "CatBoost for big data: An interdisciplinary review," *J. Big Data*, vol. 7, no. 1, pp. 1–45, 2020. doi: 10.1186/s40537-020-00369-8.

[18] S. Krishnan, S. Aruna, K. Kanagarathinam, and E. Venugopal, "Identification of dry bean varieties based on multiple attributes using catboost machine learning algorithm," *Sci. Program.*, vol. 2023, no. 4, pp. 1–21, 2023. doi: 10.1155/2023/2556066.

[19] D. Wang and H. Qian, "CatBoost-based automatic classification study of river network," *ISPRS Int. J. Geo-Inf.*, vol. 12, no. 10, pp. 1–20, 2023. doi: 10.3390/ijgi12100416.

[20] L. Cao, X. He, S. Chen, and L. Fang, "Assessing forest quality through forest growth potential, an index based on improved catboost machine learning," *Sustainability*, vol. 15, no. 11, pp. 1–18, 2023. doi: 10.3390/su15118888.

[21] S. Makridakis, E. Spiliotis, V. Assimakopoulos, A. Semenoglou, G. Mulder and K. Nikolopoulos, "Statistical, machine learning and deep learning forecasting methods: Comparisons and ways forward," *J. Oper. Res. Soc.*, vol. 74, no. 3, pp. 840–859, 2023. doi: 10.1080/01605682.2022.2118629.

[22] D. Abd, A. Sadiq, and A. Abbas, "Classifying political Arabic articles using support vector machine with different feature extraction," in *Proc. Int. Conf. Appl. Comput. Support Ind.: Innov. Technol.*, Cham, Springer International Publishing, 2019, pp. 79–94.
[23] A. Wahdan, M. Al-Emran, and K. Shaalan, "A systematic review of Arabic text classification: Areas, applications, and future directions," *Soft Comput.*, vol. 28, no. 2, pp. 1545–1566, 2024. doi: 10.1007/s00500-023-08384-6.
[24] D. Najar and S. Mesfar, "Opinion mining and sentiment analysis for Arabic on-line texts: Application on the political domain," *Int. J. Speech Technol.*, vol. 20, no. 9, pp. 575–585, 2017. doi: 10.1007/s10772-017-9422-4.
[25] D. Abd, A. Sadiq, and A. Abbas, "Political Arabic articles classification based on machine learning and hybrid vector," in *Proc. 5th Int. Conf. Innov. Technol. Intell. Syst. Ind. Appl.*, 2020, pp. 1–7.
[26] D. Abd, W. Khan, B. Khan, N. Alharbe, D. Al-Jumeily and A. Hussain, "Categorization of Arabic posts using artificial neural network and hash features," *J. King Saud Univ.-Sci.*, vol. 35, no. 6, pp. 1–7, 2023. doi: 10.1016/j.jksus.2023.102733.
[27] D. Alsaleh and S. Larabi-Marie-Sainte, "Arabic text classification using convolutional neural network and genetic algorithms," *IEEE Access*, vol. 9, no. 6, pp. 91670–91685, 2021. doi: 10.1109/ACCESS.2021.3091376.
[28] K. Alzhrani, "Political ideology detection of news articles using deep neural networks," *Int. Autom. Soft Comput.*, vol. 33, no. 1, pp. 483–500, 2022. doi: 10.32604/iasc.2022.023914.
[29] L. Zhang and D. Jánošík, "Enhanced short-term load forecasting with hybrid machine learning models: CatBoost and XGBoost approaches," *Expert Syst. Appl.*, vol. 241, 2024, Art. no. 122686. doi: 10.1016/j.eswa.2023.122686.
[30] B. Dhananjay and J. Sivaraman, "Analysis and classification of heart rate using CatBoost feature ranking model," *Biomed. Signal Process. Control*, vol. 68, no. 16, pp. 1–9, 2021. doi: 10.1016/j.bspc.2021.102610.
[31] D. Abd, A. Sadiq, and A. Abbas, "PAAD: Political Arabic articles dataset for automatic text categorization," *Iraqi J. Comput. Inform.*, vol. 46, no. 1, pp. 1–11, 2020. doi: 10.25195/ijci.v46i1.246.
[32] O. Oueslati, E. Cambria, M. HajHmida, and H. Ounelli, "A review of sentiment analysis research in Arabic language," *Future Gener. Comput. Syst.*, vol. 112, no. 4, pp. 408–430, 2020. doi: 10.1016/j.future.2020.05.034.
[33] S. Marie-Sainte and N. Alalyani, "Firefly algorithm based feature selection for Arabic text classification," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 32, no. 3, pp. 320–328, 2020. doi: 10.1016/j.jksuci.2018.06.004.
[34] A. Gasparetto, M. Marcuzzo, A. Zangari, and A. Albarelli, "A survey on text classification algorithms: From text to predictions," *Information*, vol. 13, no. 2, pp. 1–39, 83, 2022. doi: 10.3390/info13020083.
[35] L. Zhu and D. Luo, "A novel efficient and effective preprocessing algorithm for text classification," *J. Comput. Commun.*, vol. 11, no. 3, pp. 1–4, 2023. doi: 10.4236/jcc.2023.113001.
[36] I. Guellil, H. Saâdane, F. Azouaou, B. Gueni, and D. Nouvel, "Arabic natural language processing: An overview," *J. King Saud Univ.-Comput. Inf. Sci.*, vol. 33, no. 5, pp. 497–507, 2021. doi: 10.1016/j.jksuci.2019.02.006.
[37] D. Abd, W. Khan, K. Thamer, and A. Hussain, "Arabic light stemmer based on ISRI stemmer," in *Proc. Int. Conf. Intell. Comput. Theories Appl.*, Springer International Publishing, 2021, pp. 32–45.
[38] N. Nagendra and J. Chandra, "A systematic review on features extraction techniques for aspect based text classification using artificial intelligence," *ECS Trans.*, vol. 107, no. 1, pp. 1–20, 2022.
[39] P. Prihatini, K. Indah, G. Sukerti, I. Indrayana, and I. Sudiartha, "Feature extraction performance on classified methods for text sentiment analysis," in *Proc. 4th Int. Conf. Appl. Sci. Technol. Eng. Sci.*, 2021, pp. 1235–1243.
[40] S. Li, M. Deng, Z. Shao, X. Chen, and Y. Zheng, "Automatic classification of interactive texts in online collaborative discussion based on multi-feature fusion," *Comput. Electr. Eng.*, vol. 107, no. 2, pp. 1–10, 2023. doi: 10.1016/j.compeleceng.2023.108648.
[41] H. Yang *et al.*, "A multi-layer feature fusion model based on convolution and attention mechanisms for text classification," *Appl. Sci.*, vol. 13, no. 14, pp. 1–19, 2023. doi: 10.3390/app13148550.

[42] K. Singh, S. Devi, H. Devi, and A. Mahanta, "A novel approach for dimension reduction using word embedding: An enhanced text classification approach," *Int. J. Inform. Manage. Data Insights*, vol. 2, no. 1, pp. 1–10, 2022. doi: 10.1016/j.jjimei.2022.100061.

[43] P. Florek and A. Zagdański, "Benchmarking state-of-the-art gradient boosting algorithms for classification," 2023, *arXiv:2305.17094*.

[44] A. Ibrahim, R. Ridwan, M. Muhammed, R. Abdulaziz, and G. Saheed, "Comparison of the CatBoost classifier with other machine learning methods," *Int. J. Adv. Comput. Sci. Appl.*, vol. 11, no. 11, pp. 1–15, 2020. doi: 10.14569/issn.2156-5570.

[45] B. Hasan, S. A. Shaikh, A. Khaliq, and G. Nadeem, "Data-driven decision-making: Accurate customer churn prediction with Cat-Boost," *Asian Bull. Big Data Manage.*, vol. 4, no. 2, pp. 239–250, Jun. 7, 2024. doi: 10.62019/abbdm.v4i02.175.

[46] M. Luo *et al.*, "Combination of feature selection and CatBoost for prediction: The first application to the estimation of aboveground biomass," *Forests*, vol. 12, no. 2, pp. 1–21, 2021. doi: 10.3390/f12020216.

[47] S. Akpatsa, X. Li, and H. Lei, "A survey and future perspectives of hybrid deep learning models for text classification," in *Proc. 7th Int. Conf. Artif. Intell. Secur.*, Dublin, Ireland, Springer International Publishing, Jul. 19–23, 2021, pp. 358–369.

[48] M. Zhang, "Applications of deep learning in news text classification," *Sci. Program.*, vol. 2021, no. 1, pp. 1–9, 2021. doi: 10.1155/2021/6095354.

[49] J. Attieh and J. Tekli, "Supervised term-category feature weighting for improved text classification," *Knowl.-Based Syst.*, vol. 261, no. 2, pp. 1–28, 2023. doi: 10.1016/j.knosys.2022.110215.

[50] R. Rathi and A. Mustafi, "The importance of term weighting in semantic understanding of text: A review of techniques," *Multimed. Tools Appl.*, vol. 82, no. 7, pp. 9761–9783, 2023. doi: 10.1007/s11042-022-12538-3.

[51] C. Li, W. Li, Z. Tang, S. Li, and H. Xiang, "An improved term weighting method based on relevance frequency for text classification," *Soft Comput.*, vol. 27, no. 7, pp. 3563–3579, 2023. doi: 10.1007/s00500-022-07597-5.