Iraqi Journal of Information and Communications Technology(IJICT)
Vol. 7, Issue 2, August 2024
www.ijict.edu.iq
ISSN:2222-758X
e-ISSN: 2789-7362

CUNEIFORM TEXT DIALECT IDENTIFICATION USING MACHINE LEARNING ALGORITHMS AND NATURAL LANGUAGE PROCESSING (NLP)

Elaf A. Saeed [1], **Ammar D. Jasim** [2], **Munther A. Abdul Malik** [3]
[1,2] Department of Systems Engineering, College of Information Engineering, Al-Nahrain University, Jadriya, Baghdad, Iraq
[3] Department of History, Baghdad University, College of Literature, Baghdad, Iraq
elafe1888@gmail.com[1], ammaralaythawy@gmail.com[2], munther@coart.uobaghdad.edu.iq[3]
Corresponding Author: **Ammar D. Jasim**
Received: 14/09/2023; Accepted: 26/12/2023
DOI:10.31987/ijict.7.2.265

***Abstract*- Due to a lack of resources and the tokenization issue, it is challenging to identify the languages inscribed in cuneiform symbols. Sumerian and six dialects of the Akkadian language-Old Babylonian, Middle Babylonian Peripheral, Standard Babylonian, Neo-Babylonian, Late Babylonian, and Neo-Assyrian-are among the seven languages and dialects written in cuneiform that need to be identified. This problem is addressed by the Cuneiform Language Identification task in VarDial 2019. This paper presents ten machine learning algorithms derived from four types of machine learning that were used (supervised, ensemble, instance-based, and Artificial Neural Network) learnings. The Support Vector Machine (SVM), NaÃ¯ve Bayes (NB), Logistic Regression (LR), and Decision Tree (DT) algorithms within supervised learning, the K-Nearest Neighbors algorithm (KNN) within instance-based learning, the Random Forest (RF), Adaptive Boosting (Adaboost), Extreme Gradient Boosting (XGBoost), and Gradient Boosting (GB) algorithms within ensemble learning. Also, one of the natural language processing algorithms, n-gram, is used to identify the cuneiform dialect. The best result belongs to an ensemble of Random Forest classifiers working on character-level features with a macro averaged F1 score of 96%, and the best outcome for the n-grams algorithm is 0.82% of di-gram.**

keywords: **Cuneiform, unigram, CLI, Over-sampling, SVM, RF, DT, KNN, DNN.**

I. INTRODUCTION

The upstream language identification process benefits a wide range of Natural Language Processing (NLP) operations, including Machine Translation (MT), speech recognition, information retrieval, data mining, and the creation of text resources for low-resource languages.

Automatic language identification is the process of identifying a text's language based on hints provided by the text itself. The computer techniques used to identify languages range from basic wordlists to state-of-the-art deep learning techniques [1]. Dealing with closely related languages or multiple dialects of a single language presents a different problem for language identification.

The profession of Cuneiform Language Identification (CLI) in VarDial 2019 [2] aims to address the challenge of discerning languages and dialects inside texts composed using cuneiform symbols. Identifying the languages and dialects employed in cuneiform texts might provide challenges because to limited resources and difficulties associated with tokenization. Due to the dual nature of the cuneiform writing system, which combines syllabic and logographic elements, the tokenization of cuneiform texts lacks a standardized approach or instrument. Although certain efforts have been made to address the tokenization challenge in specific languages or dialects within the cuneiform script, a general solution still needs to be discovered.

ISSN:2222-758X
e-ISSN: 2789-7362

One notable difficulty encountered in the process of cuneiform language treebanking arises because of the fragmented nature of the texts. It is fairly unusual for extended portions of texts to be damaged to the point where they cannot be restored, resulting in syntactic trees that are badly deformed. Various approaches have been suggested by researchers in the field of cuneiform treebanking to address this challenge. One such approach involves evaluating sentences based on their coherence and eliminating excessively fragmented texts from the treebank. Another approach involves establishing clear and consistent guidelines for labelling and connecting fragmented elements in a manner that leaves no room for ambiguity. The presence of numerous problems in cuneiform writing significantly hinders the processes of analysis and recognition. One concern pertains to the deformation of characters and the variability of fonts and patterns. An additional complexity arises from the presence of symbol shadows that may exhibit variations between different images of the same character. These variations can be attributed to the various angles of reflected light resulting from the three-dimensional geometry of the cuneiform sign [3].

The dataset that was used was downloaded from Kaggle. The VarDial2019 workshop required the gathering of this dataset. There are 139,421 fragments of cuneiform text in the data file. The Open Richly Annotated Cuneiform Corpus (Oracc), was processed to produce the dataset. This dataset was utilized in the VarDial2019 workshop's multiclass classification exercise with the aim of precisely identifying the language and dialect being spoken.

This study examines various machine learning techniques that have been demonstrated to be successful in classifying text and compare them to the precision, accuracy, and sensitivity that was attained and one of the natural language processing algorithms, which is n-gram, to identify the cuneiform dialect.

In this paper, we first review the literature on language identification and the work on languages written using the cuneiform writing system in section two. Section three, the Dataset Description, introduces the Architecture of the proposed model used to tackle the problem of identifying such languages and dialects in section four, proposes a model in section five, section six, the experimental results, and section seven, Conclusion and future work.

II. RELATED WORK

Textual data has been the focus of the majority of language identification studies. However, [4] (Shervin Malmasi and Marcos Zampieri, 2016), A technique was created to detect a collection of four regional Arabic dialects (Egyptian, Gulf, Levantine, North African) and Modern Standard Arabic (MSA) within a transcribed speech corpus. The system attained an F1-score of 0.51 on the closed training track, securing the top position among the 18 participating teams in the sub-task. [5] (Ahmed Ali. et al., 2015) This paper examined several methods for distinguishing Arabic radio dialects. The i-vector structure extracts phonetic, lexical, and bottleneck features from voice recognition systems. Researchers mixed generative and discriminative classifiers in a multi-class SVM. We assessed them on an Arabic-English identification assignment to ensure accuracy. We also tried a binary classifier that could distinguish Dialectal Arabic from Modern Standard Arabic (MSA) 100% of the time. [6] (Cyril Goutte et al. 2016) [7] (Shervin Malmasi and Marcos Zampieri, 2017) Analysed machine learning classifiers' ability to distinguish related languages and language variants. The results of the two DSL shared task editions were used in several experiments. Aimed to compare progress between the two tasks, establish an

upper bound on ensemble and oracle performance, and offer learning curves to identify complex languages.

Human annotation is used to analyze some problematic statements. [8] (Francisco Rangel, 2017) Language variety identification labels texts in a native language (e.g., Spanish, Portuguese, English) with its distinctive variation. This study presented a low-dimensionality representation (LDR) for five Spanish varieties: Argentina, Chile, Mexico, Peru, and Spain. Compared LDR to state-of-the-art representations and found a 35% accuracy boost. [9] (Abualsoud Hanani et al. 2017) proposed many algorithms for recognizing brief Arabic or Swiss-German dialect samples for the 2017 DSL Workshop. Our best run combines Arabic text and audio files. Swiss-German data is text-only. Coincidentally, our top Swiss-German and Arabic dialect runs had around 63% accuracy.

Additionally, [9] & [10] since 2014, the VarDial workshop has been conducted annually and covers computational techniques and linguistic resources for dialects and closely related languages. [11] (Ahmet Yavuz and Gerold Schneider, 2023) was presented as the first Turkish NLI application in this paper. NLI analyses a writer's writing in multiple languages to predict their first language. Most NLI research has concentrated on English, but this paper covers Turkish. [12] (Serhiy BYKH and Detmar MEURERS, 2012) Native Language Identification identifies an author's native language from a second-language material. This paper describes how to train a native language classifier using recurring n-grams of any length. Investigated two degrees of abstraction using parts-of-speech starting with surface n-grams achieving 89.71% accuracy. [13] (Artur Kulmizev et al., 2017) examines the Native Language Identification (NLI) Shared Task 2017 performance of a linear SVM trained on language-independent character features. This simple system (GRONINGEN) performs best (87.56 F1-score) on the evaluation set utilizing 1-9-character n-grams. [14] (Tommi Jauhiainen et al.2022) The shared effort for 2022's Nuanced Arabic Dialect Identification (NADI) involved the SUKI team's language identification technology. An F1 score of 0.1963 on test set A and 0.1058 on B.

To the best of our knowledge, no effort has been made to address the issue of identifying the language and dialect of cuneiform texts. These languages, like Sumerian and Akkadian, are regarded as low-resource languages because there aren't many electronic resources available for processing cuneiform. Some of these datasets are [15] and[16] , which are annotated cuneiform corpora with morphological, syntactic, and semantic tags. [14] generated a handwritten cuneiform character picture set. [15] analyzes various machine learning techniques that have been shown to be successful at classifying text based on their achieved F1 scores, accuracy, and training times.

III. DATASET DESCRIPTION

The data of CLI shared tasks were described in [17] consisting of Sumerian (SUX), Old Babylonian (OLB), Middle Babylonian peripheral (MPB), Standard Babylonian (STB), Neo-Babylonian (NEB), Late Babylonian (LTB), and Neo-Assyrian (NEA) are the seven groups that make up this material was downloaded from Kaggle website https://www.kaggle.com/datasets/wilstrup/cuneiform-language-identification/data. The number of samples in the training data for each label is displayed in Fig.1. There are 139,421 samples of text and 550 signs in the complete set of training data.

ISSN:2222-758X
e-ISSN: 2789-7362

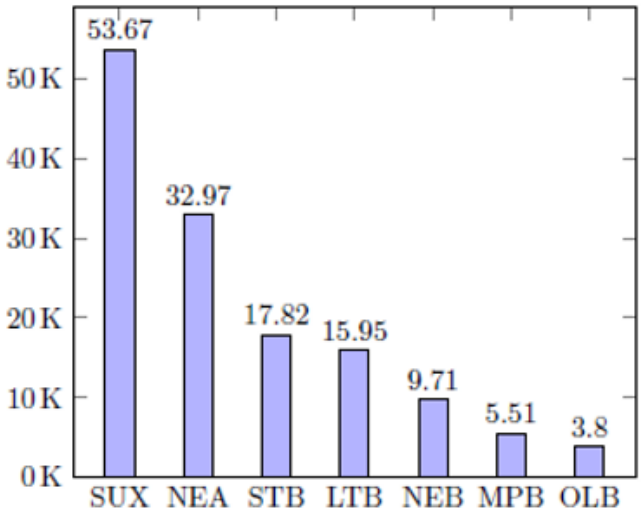


Figure 1: Number of samples for each label in the training set. [16]

Fig.1 demonstrates that the SUX and NEA classes comprise most of the training data. The data's most troublesome feature is how brief the majority of the text parts are. Only 7 characters, on average, are present in each section in the training set. Over 50% of the parts have 5 characters or fewer, while more than 10% have just one. Twelve segments in the tail of the distribution have more than 64 characters, whereas just one has more than 128. The low frequency of many of the cuneiform symbols in this data presents another challenging problem. There are 550 distinct characters in the training data. 128 of these only appear ten times or fewer, and 39 only appear once.

The training data's abundance of duplicates, both within and across classes, is another crucial characteristic. Only 86,454 of the 139,421 segments in the training set are unique. In six out of the seven classes, the most prevalent part can be found 3223 times. 1460 times, always in the same class (LTB), is where the second most often appears. To solve these challenges, we need to reduce data misalignment. There are several methods for converting from an imbalance to a balanced dataset, but this paper used a resample with different ratios method without changes in the dataset, the ratios for training and testing were 30-70, 20-80, and 10-90. then was changed in the dataset by using Over-sampling (up-sampling) by duplicating the sample from the minority classes for each class. With the new samples added to the original dataset, there are now 53,673 examples in each class, making the dataset more even.

More information about the data is provided in Table I, which reveals that only 13.65% of the data belongs to the other three classes, with the remaining 86.35% falling into the four categories of SUX, NEA, STB, and LTB before balancing. After balancing the dataset all class contains 53,673 samples.

IV. Architecture of the Proposed Model

Emulation of the decision-making computer system that can recognize cuneiform signs using a multi-class classification technique is part of the construction of an intelligent system. An intelligent system analyzes inputs such as signs in an

ISSN:2222-758X
e-ISSN: 2789-7362

TABLE I
LISTS THE NUMBER OF SAMPLES IN EACH LABEL'S TRAINING SET AND THEIR PROPORTION OF A TOTAL OF 139,421 SAMPLES, RANKED FROM HIGHEST TO LOWEST.

Label	# of samples	% of all	# of samples after balance
SUX	53,673	38.49%	53,673
NEA	32,966	23.64%	53,673
STB	17,817	12.78%	53,673
LTB	15,947	11.44%	53,673
NEB	9,707	6.96%	53,673
MPB	5,508	3.95%	53,673
OLB	3,803	2.72%	53,673

effort to determine the language. After completing several crucial steps, the intelligent system produces outcomes. The user provides data in the form of a file (.csv), and pre-processing is done to handle missing values and codded data in order to refine the data. Data is delivered to multiclass classification algorithms for training and validation after preprocessing. The accuracy and sensitivity of the output from the LR[18], K-NN[19], NB[20], DT[21], RF[22], SVM[23], ANN[24], Adaboost[25], gradient boost[26], XGBoost[27], and n-Gram[28] NLP[29] algorithms are compared. The general intelligent system of the complete architecture proposal is shown in Fig.2. The intelligent system comprises different phases for the identification of cuneiform language.

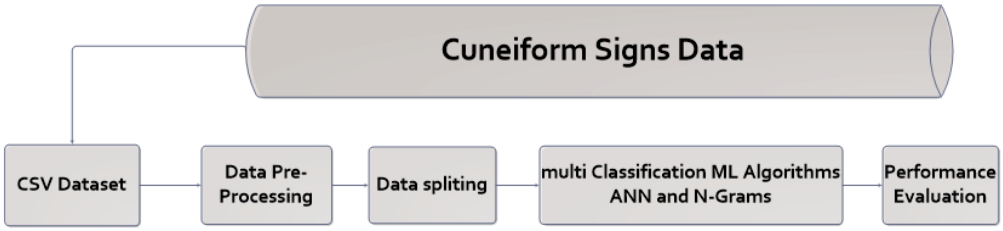


Figure 2: Diagram of General Architectural.

V. PROPOSED MODEL

Fig.3 displays the process for identifying intelligent cuneiform languages using the Adam optimization algorithm. The suggested model gathers data using the dataset that is housed in the Kaggle machine repository. The training phase and the test phase, which can be utilized for validation and testing, are the two sections of the provided model. Model design goes through several stages, starting from reading the data to evaluating the classification. Reading the obtained data from a CSV file is the first step. The preprocessing section outlines the steps undertaken to prepare the datasets for training algorithms, including Unigram extraction and counting, class balancing, and data splitting. N-gram extraction is a method used in natural language processing to identify and extract contiguous sequences of n items from a given text. For example, if n = 1, then the extracted sequences are called "unigram"; if n = 2, then the extracted sequences are called "bigrams";

ISSN:2222-758X
e-ISSN: 2789-7362

and if n = 3, then the extracted sequences are called "trigrams." In this work, a unigram was applied. For example, if the "cuneiform" column contained the string, applying the unigram would result in the "cuneiform split" column containing the list. Once the n-grams have been extracted, they can be counted to provide helpful information about the text. The count of each n-gram can be used to calculate the frequency of occurrence of the n-gram in the text. Additional samples were included in the original dataset in order to achieve class balance, resulting in a total of 53,673 samples in each class. The data was split directly without adding anything to it into three division ratios (30-70, 20-80, 10-90) %. This is one of the ways to balance the data. After making the addition to the dataset is another way to balance the dataset, it was split into (20-80). creating the neural network's layers (550 input, 164 hidden, and 7 outputs) [30] After improving the hidden layer, the number was increased from 62 to 164 to improve the results. The last stage evaluates the classification.

The system's accuracy, sensitivity, specificity, and precision are assessed after training. This test data is used as input by the trained model to determine the language. scikit-learn, a Python machine learning library that was used to design machine learning algorithms. The parameters that were changed and the others still in the default values of the library with each algorithm during training were chosen after several experiments until these values were reached. The KNN (n-neighbours = 5),SVC(gamma = auto, kernel = rbf),NB(var-smoothing = $1e^{-09}$), DT (criterion = gini), RF (n-estimator = 150 and criterion = gini), LR (max-iter = 100), GraidentBoost (n-estimator = 150), Adaboost (n-estimator = 300), and XGBoost (the defaults values).

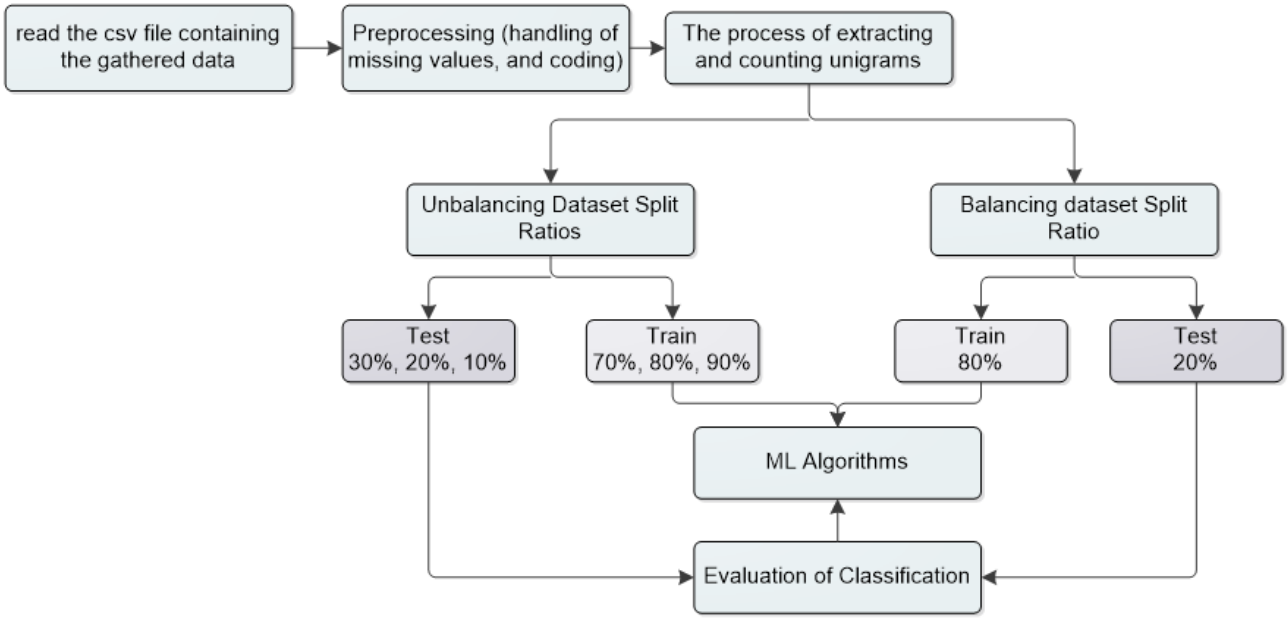


Figure 3: Diagram of the research approach for the suggested system.

ISSN:2222-758X
e-ISSN: 2789-7362

The N-gram model in Fig.4 was created from scratch as most NLP tools are designed for Latin letters and are challenging to adapt to cuneiform text. I created my own version using N-gram models with Markov chains and the Markov assumption. In this theory, the possibility of an n-gram being in place t depends exclusively on the k n-grams before it. Multiplying the conditional probabilities of n-grams gives the sequence's chance. The code calculates bigram and trigram probabilities for each language's training set. The likelihood of a series fitting each language is calculated. Language having the best possibility of occurrence is predicted.

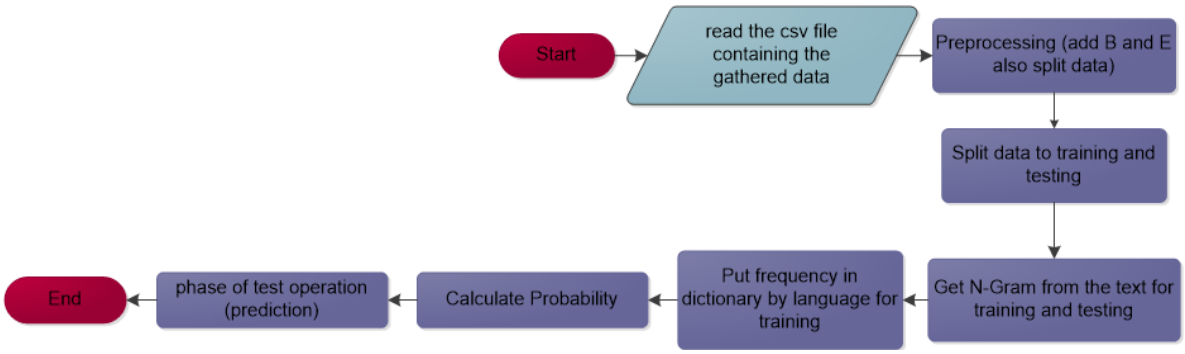


Figure 4: N-Gram suggested system.

The construction process occurred in a series of sequential stages. The initial procedure involves the reading of the CSV file. The subsequent stage involves data pre-processing, which entails the inclusion of a beginning (B) and an end (E) marker in each sentence, as well as the separation of individual characters. The third stage involves the partitioning of the data into distinct subsets for the purposes of testing and training. The next phase in the process involves the computation of the frequency. The subsequent stage involves the computation of probabilities. Lastly, it is necessary to make predictions on the test results.

VI. EXPERIMENTAL RESULTS

For more accurate cuneiform language recognition, a novel model called "Intelligent Identification of Cuneiform Language and Dialects Empowered with Adam Optimization" has been developed. The same dataset is also employed by several machine learning algorithms [30], including ANN, K-NN, NB, LR, DT, RF, SVM, Adaboost, gradient boost, XGBoost, and n-Gram NLP algorithms. The proposed model is found to identify the cuneiform language through experiments more accurately. Additionally, accuracy, sensitivity, and specificity measurements for classification are used to evaluate performance[31]. These measures include counting True Positive and False Positive elements and making a comparison

ISSN:2222-758X
e-ISSN: 2789-7362

graph for them. Table II includes the confusion matrix in various training-test ratios with 100 epochs. Accuracy can be defined as:

$$\text{Accuracy} = \frac{Tr_{Positive} + Tr_{Negative}}{Tr_{Positive} + Tr_{Negative} + Fa_{Positive} + Fa_{Negative}} \quad (1)$$

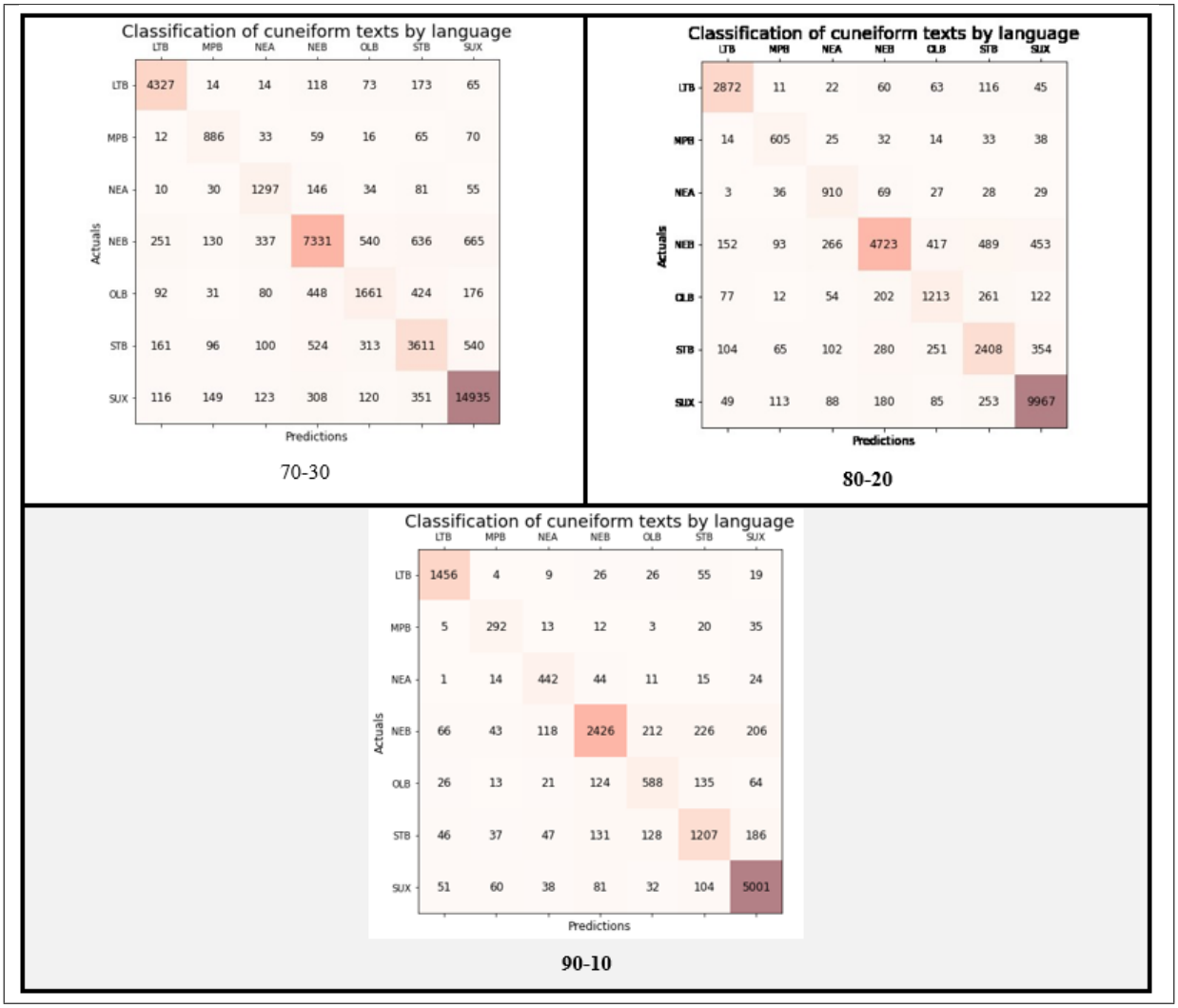
Sensitivity or recall can be calculated through the following equation

$$\text{Sensitivity} = \frac{Tr_{Negative}}{Tr_{Negative} + Fa_{Positive}} \quad (2)$$

Specificity can be calculated through the following equation:

$$\text{Specificity} = \frac{Tr_{Positive}}{Tr_{Positive} + Fa_{Negative}} \quad (3)$$

TABLE II
ANN CONFUSION MATRIX IN VARIOUS TRAINING-TEST RATIO AND 100 EPOCHS



ISSN:2222-758X
e-ISSN: 2789-7362

Intelligent cuneiform language identification performance evaluation Table III displays the Adam Optimization model with the various train-test ratios, the best epoch result. The best result in 80-20 train-test splitting. The maximum number of epochs that was reached was 150, and after this number the result stabilized and the change became within the same range.

TABLE III
THE ADAM OPTIMIZATION MODEL WITH THE VARIOUS TRAIN-TEST RATIOS, THE BEST EPOCH RESULT

Training-Testing	**Epoch**	**Accuracy (%)**	**Sensitivity (%)**	**Precision (%)**
70-30	50	0.80	0.7457	0.7257
	100	0.81	0.7557	0.74
	150	0.81	0.7542	0.7457
80-20	50	0.82	0.7585	0.8342
	100	0.81	0.7542	0.7385
	150	0.81	0.7557	0.7414
90-10	50	0.80	0.7457	0.7257
	100	0.82	0.7557	0.74
	150	0.80	0.7428	0.7242

For the purpose of intelligent coniform language detection, accuracy, sensitivity, and specificity, the Adam Optimization is used to calculate the ANN, K-NN, NB, LR, DT, RF, SVM, Adaboost, gradient boost, XGBoost, and n-Gram NLP algorithms as shown in Table IV. The best results in the Random Forest algorithm.

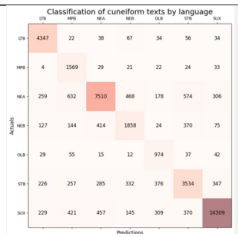
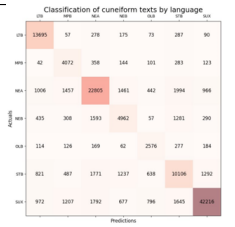
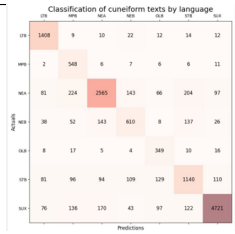
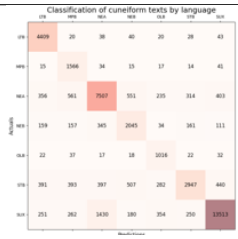
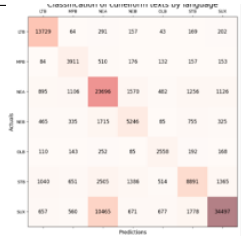
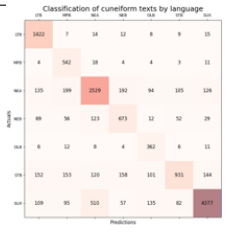
TABLE IV
THE ACCURACY OF THE ML AND N-GRAM ALGORITHMS

Train-Test	Algorithm	Accuracy (%)	Sensitivity (%)	Precision (%)
70-30	Support Vector Machine	0.71	0.5828	0.7528
	K-Nearest Neighbor	0.77	0.68	0.7771
	Naive Bayes	0.61	0.52	0.58
	Random Forest	**0.83**	0.7614	0.8385
	Gradient Boosting	0.73	0.58	0.76
	AdaBoost	0.72	0.63	0.60
	XGBoost	0.81	0.81	0.71
	Decision Tree	0.76	0.6614	0.6657
	Logistic Regression	0.73	4.53	0.7114
80-20	Support Vector Machine	0.72	0.5857	0.75
	K-Nearest Neighbor	0.77	0.7385	0.7171
	Naive Bayes	0.61	0.58	0.53
	Random Forest	**0.84**	0.7614	0.85
	Gradient Boosting	0.72	0.75	0.58
	AdaBoost	0.72	0.63	0.60
	XGBoost	0.81	0.82	0.71
	Decision Tree	0.76	0.6642	0.6742
	Logistic Regression	0.73	0.65	0.7142
90-10	Support Vector Machine	0.74	0.6857	0.78
	K-Nearest Neighbor	0.78	0.6914	0.7871
	Naive Bayes	0.62	0.55	0.59
	Random Forest	**0.84**	0.77	0.8485
	Gradient Boosting	0.73	0.75	0.58
	AdaBoost	0.71	0.58	0.64
	XGBoost	0.82	0.73	0.83
	Decision Tree	0.77	0.6714	0.6814
	Logistic Regression	0.73	0.6528	0.7142

Table V shows the result of the di-gram and tri-gram accuracy and confusion matrix for different ratios of split datasets. After balancing the dataset by using the up-sampling method, some results increased and others decreased. Upon analyzing the outcomes, it becomes evident that the random forest algorithm yields the most favorable results. This algorithm falls under the category of ensemble learning, which aims to enhance performance by partitioning the dataset into several samples and aggregating predictions from diverse models. The performance of the methods Artificial Neural Network (ANN), K-Nearest Neighbours (KNN), Decision Tree (DT), Support Vector Machine (SVM), and Gradient Boosting (GBoost) showed improvement when the size of the dataset was increased. Regarding the remaining algorithms, namely AdaBoost, XGBoost, Naive Bayes (NB), and Logistic Regression (LR), it is observed that their outcomes have exhibited a slight decline, as shown in Table VI the results.

ISSN:2222-758X
e-ISSN: 2789-7362

TABLE V
THE RESULT OF THE DI-GRAM AND TRI-GRAM ACCURACY AND CONFUSION MATRIX FIRST METHOD BALANCING

n-gram	ratio	Accuracy (%)	Confusion Matrix
Di-gram	70-30	0.81%	
	80-20	0.78%	
	90-10	0.81%	
Tri-gram	70-30	0.79	
	80-20	0.72	
	90-10	0.77	

ISSN:2222-758X
e-ISSN: 2789-7362

TABLE VI
THE ACCURACY OF THE ML AND N-GRAM ALGORITHMS AFTER UP-SAMPLING

Train-Test	Algorithm	Accuracy (%)	Sensitivity (%)	Precision (%)
80-20	Support Vector Machine	0.88	0.88	0.88
	K-Nearest Neighbor	0.88	0.88	0.88
	Naive Bayes	0.55	0.55	0.62
	Random Forest	**0.96**	**0.96**	**0.96**
	Gradient Boost	0.79	0.79	0.79
	AdaBoost	0.62	0.62	0.62
	XGBoost	0.79	0.80	0.79
	Decision Tree	0.94	0.94	0.94
	Logistic Regression	0.69	0.69	0.69
	ANN	0.92	0.92	0.92
	Di-Gram	0.82	0.81	0.82
	Tri-Gram	0.79	0.78	0.79

A. *Mean Square Error (MSE) Analysis*

The MSE function both defines and influences its learning performance. The effectiveness of the system depends on the elimination of errors. The mean square error is determined by comparing the desired and actual outputs. The mean square error, root square error, and mean square error values for the testing stages are displayed in Table VII, along with the number of epochs for each phase. Note that the results of MAE are lower than the results of MSE and RMSE in all algorithms. As for ANN, the best thing is to divide 80-20 epoch = 50. As for the machine learning algorithms, the best is the Random Forest algorithm. For the results after up-sampling the less error in the MAE method and the RF has the less MAE results as shown in Table VIII.

ISSN:2222-758X
e-ISSN: 2789-7362

TABLE VII
THE MSE, RMSE, AND MAE VALUES FOR THE TESTING STAGES

Train-Test	Algorithm	Epochs	MSE	RMSE	MAE
70-30	ANN	50	1.2631	1.1238	0.4258
		100	1.2636	1.1241	0.4143
		150	1.2502	1.1181	0.4142
80-20		50	1.1991	1.0950	0.3991
		100	1.2313	1.1096	0.4093
		150	1.1891	1.0770	0.8812
90-10		50	1.2191	1.1041	0.4044
		100	1.2751	1.1292	0.4187
		150	1.2838	1.1330	0.4292
70-30	Support Vector Machine		2.3824	1.5435	0.7224
	K-Nearest Neighbor		1.6622	1.2893	0.5417
	Naive Bayes		4.4171	2.1017	1.1224
	Random Forest		1.0454	1.0224	0.3653
	Gradient Boosting		2.1408	1.4631	0.6691
	AdaBoost		2.9627	1.7212	0.8901
	XGBoost		1.3558	1.1644	0.4368
	Decision Tree		1.6636	1.2897	0.5486
	Logistic Regression		2.3221	1.5238	0.6857
	Di-Gram		0.4539	0.4048	0.1639
	Tri-Gram		0.8839	0.4048	0.1649
80-20	Support Vector Machine		2.3244	1.5246	0.7126
	K-Nearest Neighbor		1.5886	1.2604	0.5212
	Naive Bayes		4.3786	2.0925	1.1154
	Random Forest		1.0126	1.00632	0.3550
	Gradient Boosting		2.1946	1.4814	0.6849
	AdaBoost		3.0458	1.7452	0.9086
	XGBoost		1.3660	1.1687	0.4363
	Decision Tree		1.5785	1.2564	0.5262
	Logistic Regression		2.2998	1.5165	0.6823
	Di-Gram		0.2639	0.3841	0.1639
	Tri-Gram		0.3475	0.4048	0.1475
90-10	Support Vector Machine		1.3244	1.2246	0.5126
	K-Nearest Neighbor		1.5716	1.2536	0.5209
	Naive Bayes		4.0207	2.0051	1.0556
	Random Forest		0.9977	0.9988	0.3528
	Gradient Boosting		2.1998	1.4831	0.6799
	AdaBoost		3.0513	1.7468	0.9050
	XGBoost		1.3164	1.1473	0.4253
	Decision Tree		1.5896	1.2607	0.5328
	Logistic Regression		2.3067	1.5187	0.6832
	Di-Gram		0.9475	0.3841	0.1475
	Tri-Gram		0.7311	0.3621	0.1311

TABLE VIII
THE MSE, RMSE, AND MAE VALUES FOR THE TESTING, STAGES AFTER UP-SAMPLING

Train-Test	Algorithm Epochs	MSE	RMSE	MAE
80-20	Support Vector Machine	2.4400	1.5620	0.7350
	K-Nearest Neighbor	0.2287	0.4548	0.2347
	Naive Bayes	3.5576	1.8861	1.0842
	Random Forests	0.3485	0.5903	0.1062
	Gradient Boosting	1.5363	1.2394	0.4836
	AdaBoost	3.1334	1.7701	0.9366
	XGBoost	1.5347	1.2388	0.4805
	Decision Tree	0.4287	0.6548	0.1347
	Logistic Regression	2.2998	1.5165	0.73504
	Di-Gram	1.3164	1.1473	0.4253
	Tri-Gram	1.4363	1.2374	0.4839

VII. CONCLUSION AND FUTURE WORK

We have demonstrated in this study that it is possible to identify languages and dialects in cuneiform texts that have been transcribed in Unicode letters. To assess the effectiveness of the Var-Dial evaluation campaign, we used a dataset with different algorithms. This paper has shown excellent performance in cuneiform language classification. The accuracy of studies has also been discussed in this research paper. 0.96%, 0.96%, 0.96% accuracy, sensitivity, and precision have been recorded, respectively.

For future work, make a more accurate dataset and balance it to reduce the error rate that was extracted. Also, use other algorithms to give more accurate results. As for the dataset that was used, it can be improved by adding more text in order to be balanced. One of the ideas that could be done in the future is to number the cuneiform signs so that each sign has its number, and it is possible to write on the computer using the numbers specified for each symbol. This is done by collecting many data sets, each of which relates to a specific language or dialect, and placing a unique number for each symbol. Algorithms can be trained on these data. All work is done to identify words or symbols without pronouncing them. Therefore, one of the things that will be most requested in the future is to create a program to read the tablets.

Funding

None

ACKNOWLEDGEMENT

The author would like to thank the reviewers for their valuable contribution in the publication of this paper.

CONFLICTS OF INTEREST

The author declares no conflict of interest

REFERENCES

[1] T. Jauhiainen, M. Lui, M. Zampieri, T. Baldwin, and K. LindÃ©n, "Automatic language identification in texts: A survey," Journal of Artificial Intelligence Research, vol. 65. AI Access Foundation, pp. 675-782, 2019. doi: 10.1613/JAIR.1.11675.
[2] M. Zampieri et al., "A Report on the Third VarDial Evaluation Campaign," 2019. [Online]. Available: https://sites.google.com/view/
[3] S. Gordin, G. Gutherz, A. Elazary, A. Romach, E. Jimenez, and J. Berant, "Reading Akkadian cuneiform using natural language processing," PLoS One, vol. 15, no. 10, p. e0240511, 2020.

[4] S. Malmasi and M. Zampieri, "Arabic Dialect Identification in Speech Transcripts," 2016. [Online]. Available: http://ttg.uni-saarland.de/vardial2016/dsl2016.html
[5] A. Ali et al., "Automatic Dialect Detection in Arabic Broadcast Speech,"Sep. 2015, [Online]. Available: http://arxiv.org/abs/1509.06928
[6] C. Goutte, S. LÃ©ger, S. Malmasi, and M. Zampieri, "Discriminating Similar Languages: Evaluations and Explorations," Sep. 2016, [Online]. Available: http://arxiv.org/abs/1610.00031
[7] Y. Scherrer, T. Jauhiainen, N. Ljubesic, P. Nakov, J. Tiedemann, and M. Zampieri, "Tenth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2023),l in Tenth Workshop on NLP for Similar Languages, Varieties and Dialects (VarDial 2023), 2023.
[8] F. Rangel, M. Franco-Salvador, and P. Rosso, "A Low Dimensionality Representation for Language Variety Identification,"May 2017, [Online]. Available: http://arxiv.org/abs/1705.10754
[9] Proceedings of the Workshop. Association for Computational Linguistics, 2017.
[10] M. Zampieri, P. Nakov, and Y. Scherrer, "Natural language processing for similar languages, varieties, and dialects: A survey," Nat Lang Eng, vol. 26, no. 6, pp. 595-612, 2020.
[11] A. Y. Uluslu and G. Schneider, "Turkish Native Language Identification," Jul. 2023, [Online]. Available: http://arxiv.org/abs/2307.14850
[12] S. Bykh and D. Meurers, "Native Language Identification Using Recurring N-grams-Investigating Abstraction and Domain Dependence," 2012. [Online]. Available: http://lang-8.com.
[13] A. Kulmizev et al., "The Power of Character N-grams in Native Language Identification," 2017. [Online]. Available: http://scikit-learn.org/stable/
[14] T. Jauhiainen, H. Jauhiainen, and K. LindÃ©n, "The Association for Computational Linguistics," 2022. [Online]. Available: http://hdl.handle.net/10138/352784
[15] K. Yamauchi, H. Yamamoto, and W. Mori, "Building A Handwritten Cuneiform Character Imageset." [Online]. Available: http://cdli.ucla.edu/search/
[16] C. Chiarcos et al., "Annotating a low-resource language with LLOD technology: Sumerian morphology and syntax," Information (Switzerland), vol. 9, no. 11, Nov. 2018, doi: 10.3390/info9110290.
[17] E. Doostmohammadi and M. Nassajian, "Investigating Machine Learning Methods for Language and Dialect Identification of Cuneiform Texts," Sep. 2020, doi: 10.18653/v1/W19-1420.
[18] S. Le Cessiet and J. C. Van Houwelingen, "Ridge Estimators in Logistic Regression," 1992.
[19] Daniel T. Larose and Chantal D. Larose, "k-NEAREST NEIGHBOR ALGORITHM," in Discovering Knowledge in Data: An Introduction to Data Mining, Daniel T. Larose, Ed., Wiley Data and Cybersecurity, 2014, pp. 149â164. doi: 10.1002/9781118874059.ch7.
[20] G. H. John and P. Langley, "Estimating Continuous Distributions in Bayesian Classifiers." [Online]. Available: http://robotics.
[21] I. H. Sarker, "A Machine Learning based Robust Prediction Model for Real-life Mobile Phone Data," Feb. 2019, [Online]. Available: http://arxiv.org/abs/1902.07588
[22] L. Breiman, "Random Forests," Springer, vol. 45, pp. 5â32, 2001.
[23] S. S. Keerthi, S. K. Shevade, C. Bhattacharyya, and K. R. K. Murthy, "Improvements to Platt's SMO algorithm for SVM classifier design,"Neural Comput, vol. 13, no. 3, pp. 637-649, 2001.
[24] Y. Goldberg, "A Primer on Neural Network Models for Natural Language Processing,"2016.
[25] Y. Freund and R. E. Schapire, "Experiments with a New Boosting Algorithm," 1996. [Online]. Available: http://www.research.att.com/
[26] C. Bentejac and A. Csorg O B Gonzalo MartÃnez-Munoz, "A Comparative Analysis of XGBoost." [Online]. Available: https://www.researchgate.net/publication/337048557
[27] J. Han, M. Kamber, and J. Pei, "Data Mining. Concepts and Techniques, 3rd Edition (The Morgan Kaufmann Series in Data Management Systems)," 2011.
[28] P. Majumder, M. Mitra, and B. B. Chaudhuri, "N-gram: a language independent approach to IR and NLP," in In International conference on universal knowledge and language, 2002.
[29] P. M. Nadkarni, L. Ohno-Machado, and W. W. Chapman, "Natural language processing: An introduction," Journal of the American Medical Informatics Association, vol. 18, no. 5. pp. 544-551, Sep. 2011. doi: 10.1136/amiajnl-2011-000464.
[30] Z. Z. Edie and A. D. Jasim, "MALWARE DETECTION SYSTEM BASED ON DEEP LEARNING TECHNIQUE."[Online]. Available: https://ijict.edu.iq
[31] A. M. Hamad Alhussainy and A. D. Jasim, "ECG SIGNAL CLASSIFICATION BASED ON DEEP LEARNING BY USING CONVOLUTIONAL NEURAL NETWORK (CNN)," 2020. [Online]. Available: https://ijict.edu.iq

