MJSAT 2 (4) 163-173

MJSAT

Malaysian Journal of Science and Advanced Technology

journal homepage: https://mjsat.com.my/

Machine Learning for Arabic Text Classification: A Comparative Study

Djelloul BOUCHIHA*[1], Abdelghani BOUZIANE [2], and Noureddine DOUMI [3]

[1]*Ctr Univ Naama, Inst. Sciences and Technologies, Dept. Mathematics and Computer Science, EEDIS Lab., UDL-SBA, Algeria.*
[2]*Ctr Univ Naama, Inst. Sciences and Technologies, Dept. Mathematics and Computer Science, EEDIS Lab., UDL-SBA, Algeria.*
[3]*University of Saida, Faculty of Technologies, Department of Computer Science, Algeria.*

KEYWORDS

Machine learning
Arabic text classification
Natural language processing
Feature extraction

ARTICLE HISTORY

Received 21 September 2022
Received in revised form 30 September 2022
Accepted 1 October 2022
Available online 2 October 2022

ABSTRACT

The ultimate aim of Machine Learning (ML) is to make machine acts like a human. In particular, ML algorithms are widely used to classify texts. Text classification is the process of classifying texts into a predefined set of categories based on the texts' content. It contributes to improving information retrieval on the Web. In this paper, we focus on the "Arabic" text classification since there is a large community in the world that uses this language. The Arabic text classification process consists of three main steps: preprocessing, feature extraction and ML algorithm. This paper presents a comparative empirical study to see which combination (feature extraction - ML algorithm) acts well when dealing with Arabic documents. So, we implemented one hundred sixty classifiers by combining 5 feature extraction techniques and 32 machine learning algorithms. Then, we made these classifiers open access for the benefit of the AI and NLP communities. Experiments were carried out using a huge open dataset. The comparison study reveals that TFIDF-Perceptron is the best performing combination of a classifier.

1. INTRODUCTION

Artificial intelligence (AI) simulates human intelligence in machines that are programmed to think like humans and mimic their behavior. A subset of AI is Machine Learning (ML). According to Arthur Samuel, machine learning is the research field that gives machines the ability to learn without being explicitly programmed [1].

Machine learning algorithms are supervised when they deal with labeled data. One of the problems addressed by supervised learning algorithms is Text Classification. Text classification is the process of assigning a text (document) to its corresponding class (category). Best classification leads to best Information Retrieval (IR) results, especially on the Web. Machine learning algorithms can be also unsupervised, dealing with huge data as input, to be clustered. In this paper, we focus only on supervised learning algorithms used to classify Arabic texts, starting from a labeled corpus (labeled dataset). Labeled corpus means that all of its documents are already assigned to their corresponding classes. The ML algorithm alone isn't sufficient to classify Arabic texts. The classification process needs first two steps: (1) preprocessing, which aims at cleaning the texts, and (2) feature extraction, which aims at numerically representing the texts through a feature matrix. Generally, preprocessing is a common step for all the classifiers. However, several feature extraction techniques (BoW, TFIDF…) and many ML algorithms (SVM, Logistic regression…) can constitute a classifier. Therefore, we conducted a comparative study to determine which combination (feature extraction – ML algorithm) is suitable to classify Arabic texts.

This paper focuses on the Arabic language because Arabic is a pillar of the cultural diversity of humanity. It is one of the most widely used languages in the world; it is used by more than 400 million people [2]. However, fewer efforts are

*Corresponding author:
E-mail address: Djelloul BOUCHIHA <djelloul.bouchiha@univ-sba.dz>.
2785-8901/

made for the Arabic Natural Language Processing (NLP), compared to English and Latin.

To build an Arabic text classification system, we need a labeled corpus since we opt, in this paper, for supervised learning. The more the volume of this corpus is important, the more efficient the system is. After choosing the corpus, two main phases have to be carried out, namely Training and Test. The training phase consists in taking a part of the corpus (70%), called training set, and applying the classification process that consists in turn of three sub-stages: (1) preprocessing that allows removing stop-words, normalization, etc., (2) feature extraction that codifies the Arabic text, and (3) machine learning algorithm that is run on the training set to generate as output a machine learning model; the model represents what was learned by the machine learning algorithm. After training, the testing phase takes the remaining part of the corpus (30%), called test set, and applies the classification process described above to check the system's performance.

In this paper, we first choose the DataSet for Arabic Classification (DSAC) [3], a big corpus for Arabic text classification. Then, we implement several classification prototypes. Each one starts with a common preprocessing step. Then, we implemented five feature extraction techniques and thirty-two ML algorithms, giving 160 classification tools that we compared. The comparison results show that combining TFIDF as a feature extraction technique and Perceptron as a learning model gives the best classification accuracy. This best classifier can now be used to predict the class of a newly introduced Arabic text. One other contribution of this paper is the presentation of a set of free large corpora that can be used for any Arabic classification process.

The rest of this paper is organized as follows: Section 2 reviews some important Arabic text classification comparative studies; Section 3 presents a set of open access big Arabic datasets; Section 4 describes, through a flowchart, the Arabic text classification process; Section 5 provides a detailed explanation about the 160 implemented classifiers; Section 6 gives and discusses the classification results; and finally, Section 7 provides conclusions and future works.

2. RELATED WORKS

In the literature, several articles compare Arabic text classification methods. Next, we present some comparative studies on Arabic text classification done in the last ten years:

The authors, in [4], compare six classification models: Support Vector Machine (SVM), Naïve Bayes (NB), Random Forest (RF), Decision Tree (DT), Logistic Regression (LR) and Stochastic Gradient Descent (SGD). Each classification model has been preceded by a very simple data cleaning step. The authors used only one technique to extract features, namely Bag-of-Words (BoW). For their experiments, the authors used the DSAC corpus [3]. The experimental results show that the Logistic Regression scores the best f-score.

In [5], the author compares NB, SVM and K-Nearest Neighbors (KNN) algorithms. He used his own corpus which was collected from online newspapers and magazines. He didn't say any think about the feature extraction step. The author found that the SVM classifier outperforms the other ones.

In [6], the author evaluates the performance of five supervised ML algorithms: Delta, KNN, SVM, NB and Nearest Shrunken Centroids (NSC). He used the text most frequent words (MFWs) as features. The author concludes that SVM is the most effective of the five machine learning algorithms tested. He used the King Saud University Corpus of Classical Arabic (KSUCCA) as a dataset [7].

In [8], the authors conducted 108 experiments for root N-grams and stem N-grams. The authors used Term Frequency-Inverse Document Frequency (TFIDF) to extract features. Three classifiers were implemented: NB, KNN and SVM. The results show that SVM outperforms NB and KNN with 1-grams. The authors used the Arabic Newspapers (AN) dataset [9].

The authors, in [10], conducted a comparison between six classifiers: NB, KNN, RF, SVM, DT and Neural Network (NN). They used a corpus collected from the Alqabas newspaper in Kuwait. As a feature extraction technique, they used TFIDF. This study reveals that KNN and SVM are the best performing classifiers.

In [11], the authors present a review of the work done in the field of Arabic text classification. They also present large and diverse datasets that can be used for the Arabic text classification process. The authors implemented five classification algorithms: DT, SVM, NB, KNN and MultiLayer Perceptron (MLP). They used two simple techniques for term selection: Term Frequency (TF) and Document Frequency (DF). The results of the comparison show, on average, the superiority of SVM. As a corpus, they used the Saudi Press Agency (SPA) [12].

In [13], the authors compare five ML algorithms: NB, SVM, KNN, DT and Decision Table. They also study the effects of using different Arabic stemmers on the effectiveness of these classifiers. The results show that strong accuracy is provided by SVM. Their experiments use publicly available "Arabic articles" [14].

The authors, in [15], made a comparative study between three well-known classification algorithms: KNN, DT and Rocchio algorithm. These algorithms are applied to the in-house collected Arabic dataset. The results show that Rocchio and KNN are better than DT.

In [16], the authors compare three well-known algorithms used to classify data: SVM, NB and NN. The result shows that SVM gives the best results. As a corpus, the authors used an in-house collected Arabic dataset.

In [17], an automatic SVM and KNN classifiers were developed and compared to classify 800 Arabic documents into four classes (religion, sport, politics and economy). The used weighting scheme was Term Frequency (TF). The results show that SVM outperforms KNN.

The authors, in [18], compare two feature extraction techniques in Arabic text classification: BoW and mixed-words. The results show that the mixed-words technique achieves the highest accuracy when normalization is used. For the experiments, three different datasets were used from the website www.aljazeera.net.

In [19], the authors compare different feature extraction techniques for Arabic text classification. So, they implement a conceptual representation of a text using Arabic WordNet

(AWN) as a lexical and semantic resource to compare it with other usual representation modes (N-gram and BoW). As a classifier, they implement only KNN. The results show the benefits and advantages of the conceptual representation compared to the other conventional techniques. The authors used a corpus of Arabic texts [20] built from online newspapers, such as Al-Nahar, Al-Hayat, Al-Jazeera, Al-Dostor and some other websites.

In [21], the authors developed an Arabic summarizer tool. Two classification methods, KNN and NB, were used to classify Arabic texts before and after summarization. A comparison shows that classification accuracy is almost the same whether applied on full texts or summarized ones. The authors used a corpus collected from newspapers and other websites. In general, NB outperforms KNN in all cases (full texts and summarized texts).

In [22], the author compared six machine learning algorithms: NB, DT, SVM, NN, KNN and Maximum Entropy (ME), using the same dataset and the same empirical settings. The results show that NB is the best one. Experiments were done using a corpus mainly collected from Aljazeera Arabic news channel (www.aljazeera.net).

In [23], the author investigates SVM and NB methods on different Arabic datasets. The results reveal that SVM outperforms NB with regard to all evaluation metrics. In his experiments, the author used the Saudi Newspapers (SNP) corpus [24]. He didn't say any think about the feature extraction step.

In [25], the author compares the performances of three classification algorithms, namely KNN, NB and DT. He used the Saudi Press Agency (SPA) corpus. The results showed that the NB algorithm outperforms KNN and DT.

Table 1 summarizes some comparative studies on Arabic text classification: the column "Comparative study" contains the study's citation; "Corpus" corresponds to the dataset on which the study has been done; "Feature extraction techniques" includes the studied feature extraction techniques; "Machine learning algorithms" cites the compared ML algorithms; in the column "Best Model", we find the best learned model.

Table 1. Comparative studies on Arabic text classification

Comparative study	Corpus	Feature extraction techniques	Machine learning algorithms	Best model
[4]	DSAC	BoW	SVM, NB, RF, DT, LR and SGD.	Logistic Regression
[11]	SPA	TF and DF	SVM, DT, NB, MLP and KNN.	Support Vector Machine
[22]	from www.aljazeera.net	Not specified	NB, DT, ME, SVM, ANN and KNN	Naïve Bayes
[5]	from online newspapers and magazines	Not specified	NB, KNN and SVM	Support Vector Machine
[6]	KSUCCA	MFWs	Delta, KNN, SVM, NB, NSC.	Support Vector Machine
[8]	AN	N-gram and TFIDF	NB, KNN, SVM	Support Vector Machine
[10]	from Alqabas newspaper in Kuwait	TFIDF	NB, KNN, RF, NN, SVM and DT	K-Nearest Neighbors and Support Vector Machine
[13]	Arabic articles	Not specified	NB, SVM, KNN, DT, and Decision Table	Support Vector Machine
[15]	in-house collected Arabic data	Not specified	KNN, C4.5 and Rocchio algorithm	K-Nearest Neighbour and Rocchio
[16]	in-house collected Arabic data	Not specified	SVM, NB and NN	Support Vector Machine
[17]	from newspapers and websites	TF	SVM and KNN	Support Vector Machine
[23]	SNP	Not specified	SVM and NB	Support Vector Machine
[25]	SPA	Not specified	KNN, NB and DT	Naïve Bayes
[21]	from newspapers and websites	TF	NB and KNN	Naïve Bayesian
[18]	from www.aljazeera.net	BoW and mixed words	FRAM classifier	FRAM classifier with mixed words
[19]	from online newspapers and websites	BoW, N-gram and conceptual representation	KNN	KNN classifier with conceptual representation

From Table 1, we note that several studies don't care about the impact of the feature extraction step on the classification result. The comparison study presented in this paper is the most exhaustive one, allowing for comparing 160 classifiers by combining 5 feature extraction techniques and 32 machine learning models using the open access DSAC [3] dataset.

Some other literature reviews, such as [26], [27], [28], [29], [30], [31] and [32], can also be interesting. Usually, these reviews present detailed descriptions of the Arabic text classification process, notably the preprocessing step, feature extraction techniques, feature selection techniques and ML algorithms. Sometimes, they also present some Arabic corpuses. However, these reviews didn't make empirical studies; they only took the information as they found it in the literature.

Other comparative studies have been carried out, however, on English datasets, like in [33], [34], [35] and [36].

3. ARABIC CORPUS

Before launching the Arabic text classification process, the first thing to do is to choose a corpus. So, this section answers the question: which corpus is best for the Arabic classification process?

From our point of view, the best corpus is the one easy to access (preferably free), big in terms of number of documents and categories and used by most researchers working on the

Arabic classification issue. To this end, we present in this section a set of open access big Arabic corpuses: TALAA (French acronym of "Traitement Automatique de la Langue Arabe et Applications") [37]; DataSet for Arabic Classification (DSAC) [3]; Single-label Arabic News Articles Dataset (SANAD) [38]; RTAnews [39]; and News Articles Dataset in Arabic (NADiA) [40]. Table 2 summarizes each corpus's properties: number of documents, number of words, number of categories, and the source from which the corpus has been built.

Table 2. Arabic Text Classification datasets

Corpus	Number of documents	Number of words	Number of categories	Source
DSAC	111728	319254124	5	3 Arabic online newspapers
TALAA	57827	more than 14 million	8	9 Arabic online newspapers
SANAD	190k+	-	7	Three news websites: AlKhaleej, AlArabiya and Akhbarona
RTAnews	23837	-	40	Russia Today in Arabic news website
NADiA1	35416	-	24	SkyNewsArabia news website
NADiA2	451230	-	28	Masrawy news website

4. ARABIC TEXT CLASSIFICATION PROCESS

The text classification system aims at predicting the class (category) of a newly introduced text. To be performed, the text classification relays on three main phases, namely Training, Test and Prediction.

As shown in figure 1, the Arabic text classification process starts by analyzing a labeled corpus (dataset). This dataset will be split into two subsets: training and test set. Typically, the training set consists of 70% of the dataset, and the test set is the remaining 30% of the dataset. Whereas the training set is used to learn the models, the test set is used to evaluate the classifiers by computing their accuracies.

During the training phase, the texts undergo a preprocessing step, which consists in removing numbers, stop-words, punctuations and non-Arabic words. It also consists of tokenization and normalization. Once processed, the texts (training set) are passed to the feature extraction step, which generates a numerical representation of the Arabic texts according to the chosen technique. As a result of the feature extraction, we get m * n feature matrix, with m being the number of samples (texts) and n the number of features. Generally, features are terms (words). The feature matrix represents the input of an ML algorithm that, after running, becomes a machine learning model. The three components, preprocessing, feature extraction and ML model, constitute a classifier.

Since there are many feature extraction techniques and several ML algorithms, various classifiers can be implemented; which will be suitable for the prediction task? The test phase answers this question.

The test set will be given to the classifiers coming from the previous phase. The classification results will be compared to determine the best classifier, which will be finally used to predict the class of a newly introduced text at the prediction phase.

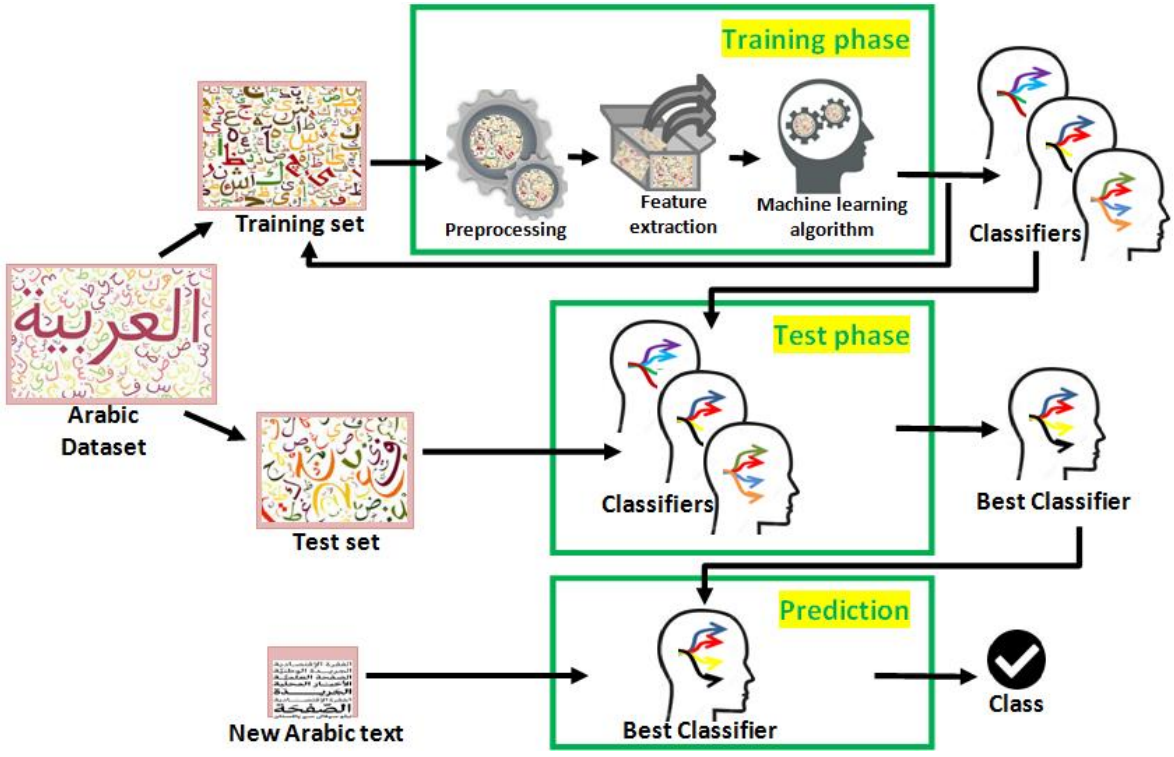

Fig. 1. Flowchart of the Arabic text classification process

We note that feature extraction is also referred to as feature weighting, Indexing, or document representation. We also note that the classification process can be enhanced by more additional step, namely feature selection, which aims at selecting a subset of the features available for describing the texts to reduce the dimensions of the feature matrix; if this step is ignored, all features will be considered to build the feature matrix.

5. IMPLEMENTATION

To perform our experiments, we used DataSet for Arabic Classification (DSAC) [3]. We randomly split the considered part into a training set (70%) and a test set (30%). Then, we opt for Python language to implement 160 classifiers, among which we can select the best one. These classifiers are now Open Access* for the benefit of the AI and NLP communities.

We used the Python packages which contain built-in modules that allow the implementation of our classifiers [41]. A module is a file containing Python definitions and statements. A collection of modules is called package. Python allows us to install other external packages, among which we cite:

- For the preprocessing step, the nltk [42], textblob [43] and tashaphyne [44] packages have to be installed.
- Some features extraction techniques need the installation of gensim [45]. The numpy [46] package is also necessary at this stage.
- To execute ML algorithm, we used scikit-learn package [47] [48].

Next, we give more details about each implemented part of the classifiers:

5.1 Preprocessing

Preprocessing is the first common part of all classifiers. It receives as input an Arabic text and generates a cleaned one to be passed to the following stage (feature extraction).

Next are the main tasks involved in the Preprocessing step:

a) *Removing stop-words, numbers, punctuations, links, white-spaces and non-Arabic words:* In particular, stop-words are high-frequency words that we filter out of a document because their presence in the text fails to distinguish it from the other texts [42]. Thus, stop-words do not contribute to the content of the text [49]. In our implementation, we consider NLTK stop-words for the Arabic language [42], to which we add two more stop-words Github lists†&‡. Thus, to the best of our knowledge, we have got the largest list of Arabic stop-words (907 stop-words) never used before in an implementation.

b) *Tokenization:* means dividing a text into subunits called tokens [50]. A token is a word of the text. Implementing tokenization with Python allows splitting a full text into tokens according to a separator (generally white-space) [51].

c) *Normalization:* converts a word into its standard normal canonical form [6]. Some specific Arabic letters are also converted into their normalized forms [4]. Normalization also consists of striping Tatwil/Kashida (elongation) and striping Tashkeel (diacritics).

d) *Stemming:* allows mapping Arabic words into their roots [15]. In our implementation, we use Tashaphyne stemmer [44] since it has the best performance compared to the other stemmers [15].

5.2 Feature extraction

After preprocessing, the feature extraction step aims at numerically representing the training set or the test set. This new representation, usually a matrix, will be given to the ML algorithm. Next, are some techniques used to extract features:

1) *Bag-of-words (BoW):* are widely applied in the text classification [49]; each word's frequency (occurrence) is used as a feature to train a classifier. So, in this model, a text is represented as a bag (a set) of words, disregarding the word's grammar and order, but keeping only their multiplicities. To implement the bag-of-words model, we used the gensim library [45].

2) *Term Frequency - Inverse Document Frequency (TFIDF):* computes the importance of a word within a text [52]. TFIDF is defined as the product of Term Frequency (TF) and Inverse Document Frequency (IDF). To implement TFIDF, we used two packages: gensim [45] and numpy [46].

3) *Doc2Vec:* is a natural language processing technique that uses a neural network model to learn associations between words from a very large dataset [53]. As a result, word2vec represents each word with a list of numbers, called a vector. An extension of wor2vec, called doc2vec, aims to represent not only a word as a vector but the entire document as a vector [54].

4) *N-gram:* are used in natural language processing [55]; N-gram is usually defined as a sequence of N words. To implement the N-gram method, we used the NLTK [42] package.

5) *HashingVectorizer:* converts a collection of texts into a matrix of token occurrences [47]. The text vectorizer implementation uses the hashing trick [56] to convert a token (string) into a feature (integer).

5.3 Machine learning algorithm

Artificial intelligence (AI) simulates human intelligence processes by machines [57]. A part of artificial intelligence is Machine Learning (ML), which gives computers the ability to learn without being explicitly programmed [1]. Machine learning algorithms receive as input feature matrix of training data, and build machine learning models in order to make predictions and decisions. When the training data is labeled, it is said to be supervised learning algorithms that can be used to address classification issues, notably "Text classification".

Next, we present supervised learning algorithms that we implemented for Arabic text classification:

Support Vector machines (SVM) is introduced as a machine learning algorithm to resolve classification problems [58]. It maps the input vectors into high dimensional space, and constructs separating hyper-plane(s). Scikit-learn API [47]

* https://github.com/khouloud-1/Arabic_Classifiers
† https://github.com/mohataher/arabic-stop-words
‡ https://github.com/saobou/arabic-text-preprocessing

implements three variants of SVM, namely C-Support Vector Classification (SVC), Nu-Support Vector Classification (NuSVC) and Linear Support Vector Classification (LinearSVC). SVC and NuSVC are similar functions, with slightly different sets of parameters. LinearSVC is another faster implementation of SVM with a linear kernel.

Logistic Regression (LR) predicts an output value based on an input value like linear regression [59]. However, unlike linear regression, the output value in Logistic Regression is binary or dichotomous. Scikit-learn API [47] implements two variants of Logistic Regression, namely regularized Logistic Regression (LogisticRegression) and Logistic Regression with built-in cross-validation support (LogisticRegressionCV).

Ridge Classifier is introduced in [60]. Scikit-learn API [47] implements two ridge based classifiers: classifier using Ridge regression (RidgeClassifier) and Ridge classifier with built-in cross-validation (RidgeClassifierCV). RidgeClassifier treats the classification problem as a regression task, but by converting binary outputs into [-1, 1] values. RidgeClassifierCV automatically selects the best hyper-parameters.

Stochastic Gradient Descent - SGD can be considered a stochastic approximation of gradient descent optimization [61]. Through an iterative algorithm, SGD substitutes the current gradient with an estimate thereof for optimizing the loss function with suitable smoothness properties. Scikit-learn API [47] implements SGDClassifier to fit linear classifiers under convex loss functions.

Perceptron is a binary classifier based on a linear discriminant function [62]. Scikit-learn API [47] implements the linear Perceptron classifier (Perceptron).

Passive Aggressive Algorithm updates the classification function when a new input is misclassified or its classification score does not go over some predefined value [63]. PassiveAggressiveClassifier was easily implemented using scikit-learn API [47].

Linear Discriminant Analysis came from Fisher's discriminant analysis [64]; it searches for those vectors that best discriminate among classes in the underlying space. Scikit-learn API [47] implements not only a Linear Discriminant Analysis classifier (LinearDiscriminantAnalysis) that can learn linear boundaries, but also a Quadratic Discriminant Analysis classifier (QuadraticDiscriminantAnalysis) that can learn quadratic boundaries, and is consequently more flexible.

Nearest Neighbors consists in finding a predefined number of training samples closest to the new sample, and predicting its class [65]. Scikit-learn API [47] implements three different Nearest Neighbor classifiers: a classifier that implements the K-Nearest Neighbors vote (KNeighborsClassifier), a classifier that implements a vote among neighbors within a specified radius (RadiusNeighborsClassifier) and Nearest Centroid classifier (NearestCentroid). KNeighborsClassifier is based on the K-Nearest Neighbors of each query point, where K is an integer given by the user. RadiusNeighborsClassifier is based on the number of neighbors within a radius of each training sample, where Radius is a float given by the user. NearestCentroid represents each class by the centroid of its elements.

Gaussian Process (GP) is a generalization of the Gaussian probability distribution [66]. In machine learning, GP is based on an effective method to place a prior distribution over the space of functions. Scikit-learn API [47] implements Gaussian Process Classification based on Laplace approximation (GaussianProcessClassifier).

Naïve Bayes can be used to calculate the probability of a classification problem hypothesis given our prior knowledge [67]. Scikit-learn API [47] implements five Naïve Bayes classifiers: Gaussian Naïve Bayes (GaussianNB), Naïve Bayes classifier for multinomial models (MultinomialNB), the Complement Naïve Bayes classifier (ComplementNB), Naïve Bayes classifier for multivariate Bernoulli models (BernoulliNB) and Naïve Bayes classifier for categorical features (CategoricalNB). GaussianNB uses the Gaussian Naïve Bayes algorithm for classification. MultinomialNB uses the Naïve Bayes algorithm for multinomially distributed features. ComplementNB was designed to correct the severe assumptions that the MultinomialNB classifier can make. BernoulliNB is dedicated to binary features. CategoricalNB is suitable for the categorically distributed discrete features.

Decision Tree predicts the class of an instance by traveling from a root node of a tree to a leaf [68]. Scikit-learn API [47] implements a Decision Tree classifier (DecisionTreeClassifier) and an extremely randomized tree classifier (ExtraTreeClassifier). ExtraTreeClassifier differs from DecisionTreeClassifier in the way trees are built.

Bagging meta-estimator generates several predictors (estimators) and uses these predictors to have one aggregated predictor [69]. Scikit-learn API [47] implements a Bagging classifier (BaggingClassifier) that fits base classifiers, each on random subsets of the initial dataset, and then aggregate their separated predictions to form one ultimate prediction.

Random Forests are a combination of tree-structured classifiers [70]. Each tree depends on a random vector sampled separately and with the same distribution for all trees in the forest. Scikit-learn API [47] implements two averaging algorithms based on randomized forests, namely Random Forest classifier (RandomForestClassifier) and Extra-Trees classifier (ExtraTreesClassifier). For the last one, trees are extremely randomized.

Adaptive Boosting is an iterative procedure that combines the output of many weak classifiers into a weighted sum representing the output of the boosted classifier [71]. Scikit-learn API [47] implements the AdaBoost classifier (AdaBoostClassifier).

Gradient Tree Boosting can produce a classifier by combining weak classifiers based on decision trees [72]. Scikit-learn API [47] implements two classifiers based on Gradient Tree Boosting, namely Gradient Boosting for classification (GradientBoostingClassifier) and Histogram-based Gradient Boosting Classification Tree (HistGradientBoostingClassifier). For big datasets, HistGradientBoostingClassifier is much faster than GradientBoostingClassifier.

Voting Classifier is a strategy defined by Scikit-learn [48]. It combines different machine learning classifiers, and predicts the class labels based on the majority vote or the average predicted probabilities. Scikit-learn API [47]

implements a Soft Voting/Majority Rule classifier for unfitted estimators (VotingClassifier).

Stacked Generalization consists in stacking the predictions of a set of classifiers, and passing these predictions as inputs to the final classifier to compute the prediction [73]. Stack of estimators with a final classifier (StackingClassifier) was easily implemented using scikit-learn API [47].

Multilayer Perceptron (MLP) is a feedforward Neural Network [74]. It consists of at least three layers: input, output and hidden layers. The input layer receives the input data to be treated. Prediction is performed by the output layer. The hidden layer constitutes the true computational engine of MLP. Scikit-learn API [47] implements a Multi-layer Perceptron classifier (MLPClassifier).

6. EXPERIMENTS

As mentioned above, we implemented 160 classification tools[§], by combining 5 feature extraction techniques and 32 machine learning algorithms. All prototypes have the same preprocessing step.

To check the classification tools, we used the publicly available DSAC corpus [3] where documents are categorized into five classes, namely culture (ثقافة\θaqAfaħ\)[**], sport (رياضة\riyADaħ\), politics (سياسة\siyAsaħ\), economy (اقتصاد\Aiq.tiSaAd\) and diverse (متفرقات\mutafar~iqAt\).

As an evaluation metric, we opted for Accuracy, which is the fraction of the study population that is decided correctly [75]. For a classification problem, Accuracy = (TP + TN)/N, where TP (true positives) is the number of documents correctly classified, TN (true negatives) is the number of documents correctly not classified, and N is the total number of documents.

Preliminary experiments showed that classifiers with N-gram feature extraction technique are expensive in terms of memory space and processing time, and give the lowest accuracy. Thus, we early excluded 32 classifiers with N-gram from the classification competition.

Figure 2 shows the classification accuracy histograms of the four remaining feature extraction techniques, namely BoW, TFIDF, Doc2Vec and HashingVectorizer, combined with 32 implemented machine learning algorithms, which gives 128 classifiers.

Figure 3 represents the time histograms of the implemented classifiers. Note that GaussianProcessClassifier, GradientBoostingClassifier, HistGradientBoostingClassifier and StackingClassifier were extremely time-consuming. Thus, they were excluded from Figure 3 to not disturb the time histograms presentation.

From Figure 2 and Figure 3, we note that: BoW acts well (accuracy: 0.96) with LogReg, with reasonable running time; Doc2Vec gives its best accuracy (0.93) with SVC; HashingVectorizer attempts an accuracy of 0.94 with LinearSVC, LogReg, RidgeClassifierCV and SGDClassifier.

We achieved the best accuracy (0.97) with the combinations: TFIDF-Perceptron, TFIDF-GaussianProcessClassifier and TFIDF-VotingClassifier. Among these classifiers, Perceptron was the best time-saving one.

To be more explicit, let's give some numerical results through Table 3, Table 4 and Table 5. Table 3 summarizes the accuracy average and the time average of the feature extraction techniques BoW, TFIDF, Doc2Vec and HashingVectorizer. Remember that we excluded N-gram because of its lowest accuracy and highest time. Each average is calculated by considering the 32 ML algorithms. Generally, classifiers with TFIDF give the best accuracy. However, in terms of time, classifiers with HashingVectorizer are the best time-saving ones.

Table 3. Feature extraction accuracy and time averages

Feature extraction	Accuracy average	Time average (sec)
BoW	0.8203	1303.4172
TFIDF	0.8215	1065.7357
Doc2Vec	0.8203	253.4665
HashingVectorizer	0.8028	150.2168

Table 4 summarizes the accuracy average and the time average of the 32 ML algorithms. Each average is calculated by considering the four feature extraction techniques, as explained above. While VotingClassifier, LogReg, StackingClassifier and LogRegCV give the best accuracy averages, ComplementNB, MultinomialNB, KNeighborsClassifier and RadiusNeighborsClassifier are the best ML algorithms in terms of time averages.

Table 4. ML algorithm accuracy and time averages

ML algorithm	Accuracy average	Time average (sec)
SVC	0.8750	335.8766
NuSVC	0.8725	355.5088
LinearSVC	0.9350	127.6819
LogReg	0.9425	140.8451
LogRegCV	0.9375	209.2508
RidgeClassifier	0.9150	172.8451
RidgeClassifierCV	0.9125	140.9194
SGDClassifier	0.9350	149.2828
Perceptron	0.9275	146.4476
PassiveAggressiveClassifier	0.9225	149.8557
LDA	0.8325	332.3067
QDA	0.3975	152.4336
KNeighborsClassifier	0.4475	119.0609
RadiusNeighborsClassifier	0.1850	119.1284
NearestCentroid	0.8775	119.8522
GaussianProcessClassifier	0.8950	12226.7398
GaussianNB	0.8425	119.4378
MultinomialNB	0.9100	118.6682
ComplementNB	0.9225	117.0245
BernoulliNB	0.8650	120.0071
CategoricalNB	0.2850	126.3035
DecisionTreeClassifier	0.7525	132.6529
ExtraTreeClassifier	0.5925	129.0423
BaggingClassifier	0.8375	159.8989
RandomForestClassifier	0.9175	140.8300
ExtraTreesClassifier	0.9325	148.1819
AdaBoostClassifier	0.7875	210.4441
GradientBoostingClassifier	0.9150	1513.2007
HistGradientBoostingClassifier	0.9325	1326.2069

[§] https://github.com/khouloud-1/Arabic_Classifiers

[**] For readability purposes, an Arabic text is followed by its HSB Buckwalter transliteration [76] and, eventually, English translation.

VotingClassifier	0.9450	314.3939
StackingClassifier	0.9400	2226.6846
MLPClassifier	0.9325	281.6788

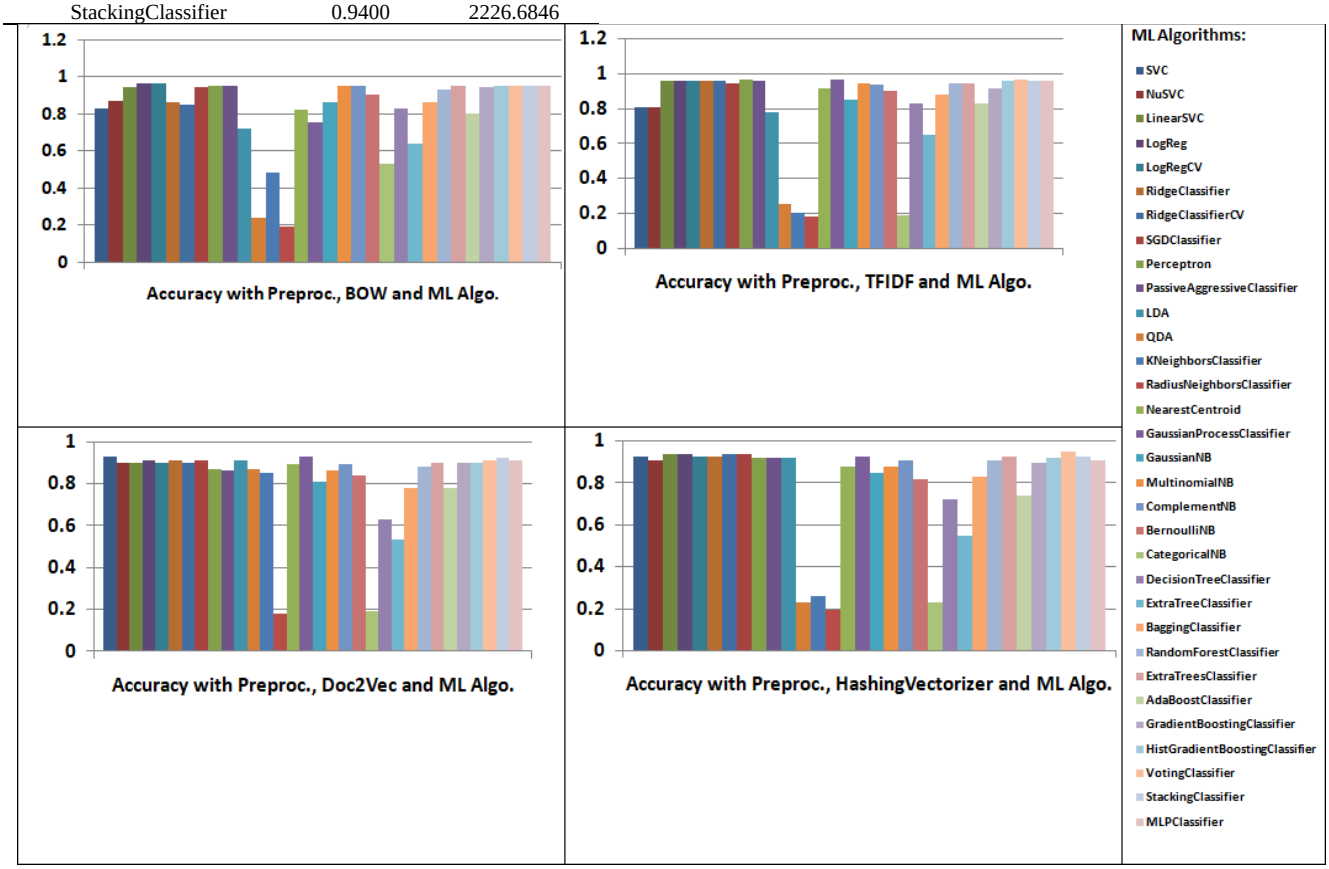

Fig. 2. Accuracy histograms of the implemented classifiers

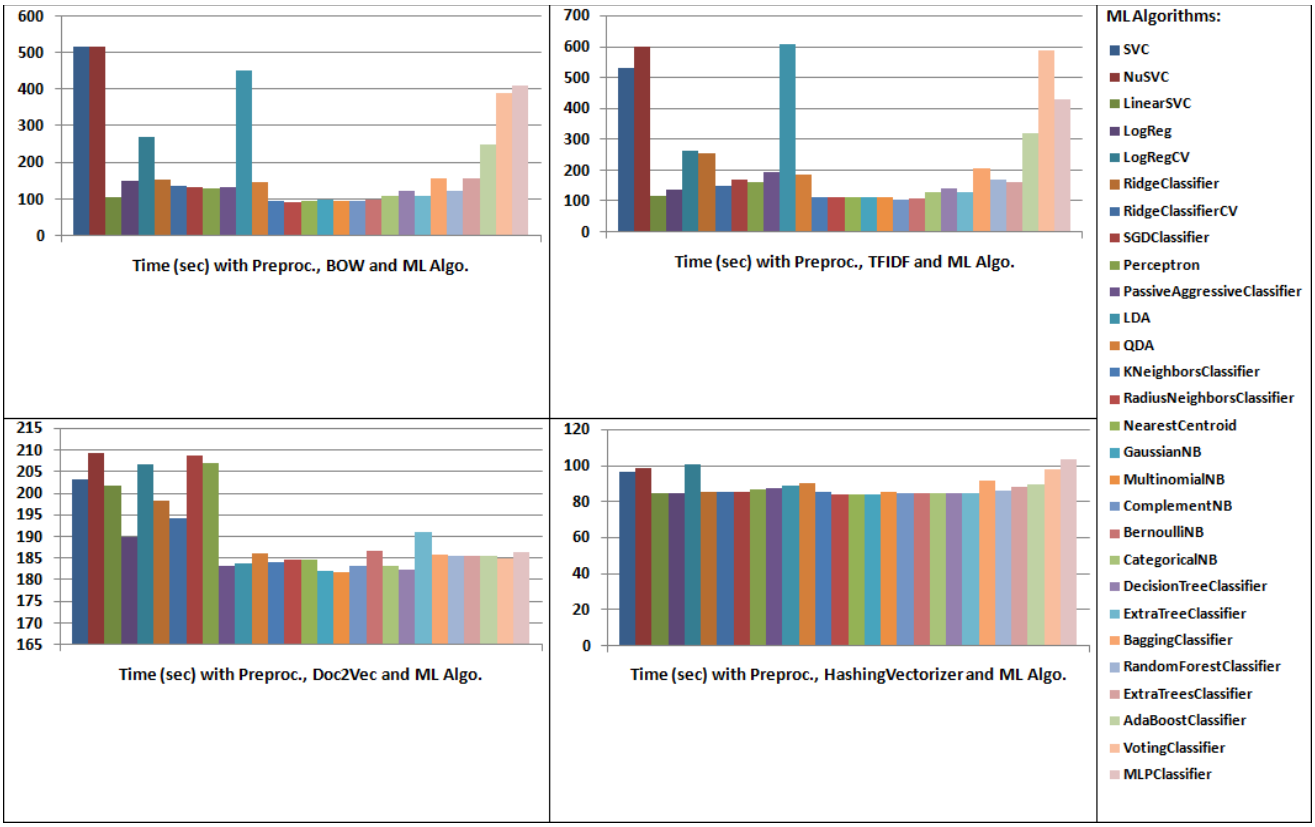

Fig. 3. Time histograms of the implemented classifiers

Table 5 illustrates 10 strong combinations: feature extraction technique and ML algorithm. It is clear that classifiers with TFIDF give the best accuracies. However, classifiers with HashingVectorizer are the best ones in terms of time. To have a compromise accuracy-time, TFIDF-Perceptron is the best classifier.

Table 5. Ten strong combinations (Feature extraction-ML algorithm)

Combination	Accuracy	Time (sec)
BOW-LogReg	0.96	150.4150
BOW-LogRegCV	0.96	267.7653
TFIDF-Perceptron	0.97	161.9522
TFIDF-GaussianProcessClassifier	0.97	18236.6016
TFIDF-VotingClassifier	0.97	585.8644
Doc2Vec-SVC	0.93	203.1290
HashingVectorizer-LinearSVC	0.94	84.8975
HashingVectorizer-LogReg	0.94	84.6917
HashingVectorizer-RidgeClassifierCV	0.94	85.0438
HashingVectorizer-SGDClassifier	0.94	85.6426

Finally, we summarize the main findings of our study as follows:

- Whatever the ML algorithm, a classifier with N-gram is memory and time consuming, and gives the lowest accuracy.
- Classifiers with TFIDF have generally the best accuracy.
- Classifiers with HashingVectorizer are time-saving.
- GaussianProcessClassifier, GradientBoostingClassifier, HistGradientBoostingClassifier and StackingClassifier are time-consuming ML algorithms.
- ComplementNB, MultinomialNB, KNeighborsClassifier and RadiusNeighborsClassifier are time-saving ML algorithms.
- The combinations: TFIDF-Perceptron, TFIDF-GaussianProcessClassifier and TFIDF-VotingClassifier give the best accuracies.
- HashingVectorizer-LinearSVC and HashingVectorizer-LogReg are the best classifiers in terms of time.
- The best combination, by considering accuracy and time simultaneously, is TFIDF-Perceptron.

7. CONCLUSION AND PERSPECTIVES

Machine Learning, a branch of Artificial Intelligence, aims at giving machines the ability to learn without being explicitly programmed. After preprocessing and feature extraction, a machine learning algorithm constitutes the third task of the Arabic text classification process adopted in this paper.

Classifying a text is assigning it to its corresponding category. This can be used, inter alia, to improve the answers quality of the information retrieval systems. In this paper, we focus on the Arabic language since a wide community in the world uses Arabic. However, fewer efforts are made for the Natural Language Processing (NLP) of the Arabic language compared to the other languages.

In this paper, we implemented and compared 160 classifiers with different feature extraction techniques and many machine learning algorithms, to see which classifier gives the best results when dealing with Arabic documents. The results show that the combination TFIDF-Perceptron outperforms the other classifiers.

Most of the feature extraction techniques, found in the literature, are syntactic-based. In future work, we plan to improve the feature extraction step by considering the semantics of Arabic texts instead of their syntax. To this end, we have to use domain ontologies to numerically represent the semantics of Arabic texts. Then, this semantic representation will be given to the machine learning algorithm to learn the Arabic text classification model. We think that this will significantly improve the classification accuracy.

REFERENCES

[1] A. L. Samuel, "Some studies in machine learning using the game of checkers," IBM Journal of Research and Development, vol. 3, pp. 210-229, 1959.

[2] UNESCO. (2020). World Arabic Language Day, December 18, 2020. Available: https://en.unesco.org/commemorations/worldarabiclanguageday. Last visited: June 2022.

[3] M. Biniz, "DataSet for Arabic Classification," Mendeley Data, V2, doi: 10.17632/v524p5dhpj.2, 2018.

[4] M. A. H. Madhfar and M. A. H. Al-Hagery, "Arabic text classification: A comparative approach using a big dataset," in 2019 International Conference on Computer and Information Sciences (ICCIS), 2019, pp. 1-5.

[5] E. Hanandeh, "Arabic text categorization using three classifiers methods: A comparative study," International Journal of Computer Science Issues (IJCSI), vol. 15, pp. 49-52, 2018.

[6] M. Al-Yahya, "A comparative study of machine learning methods for genre identification of classical arabic text," Comput. Mater. Contin, vol. 60, pp. 421--433, 2019.

[7] M. Alrabiah, A. Al-Salman, E. S. Atwell, and N. Alhelewh, "KSUCCA: A key to exploring Arabic historical linguistics," International Journal of Computational Linguistics (IJCL), vol. 5, pp. 27--36, 2014.

[8] R. Ayed, M. Labidi, and M. Maraoui, "Arabic text classification: New study," in 2017 International Conference on Engineering & MIS (ICEMIS), 2017, pp. 1-7.

[9] A. Al-Thubaity, M. Khan, M. Al-Mazrua, and M. Al-Mousa, "New language resources for arabic: corpus containing more than two million words and a corpus processing tool," in International Conference on Asian Language Processing, 2013, pp. 67--70.

[10] F. S. Al-Anzi and D. AbuZeina, "Toward an enhanced Arabic text classification using cosine similarity and Latent Semantic Indexing," Journal of King Saud University-Computer and Information Sciences, vol. 29, pp. 189--195, 2017.

[11] M. S. Khorsheed and A. O. Al-Thubaity, "Comparative evaluation of text classification techniques using a large diverse Arabic dataset," Language Resources and Evaluation, vol. 47, pp. 513-538, 2013.

[12] F. Thabtah, M. Eljinini, M. Zamzeer, and W. Hadi, "Naïve Bayesian based on Chi Square to categorize Arabic data," in Proceedings of the 11th international business information management association conference (IBIMA) conference on innovation and knowledge management in twin track economies, Cairo, Egypt, 2009, pp. 4-6.

[13] I. Hmeidi, M. Al-Ayyoub, N. A. Abdulla, A. A. Almodawar, R. Abooraig, and N. A. Mahyoub, "Automatic Arabic text categorization: A comprehensive comparative study," Journal of Information Science, vol. 41, pp. 114-124, 2015.

[14] D. Abuaiadah, J. El Sana, and W. Abusalah, "On the impact of dataset characteristics on arabic document classification," International Journal of Computer Applications, vol. 101, pp. 31-38, 2014.

[15] A. H. Mohammad, O. Al-Momani, and T. Alwada'n, "Arabic text categorization using k-nearest neighbour, Decision Trees (C4. 5) and Rocchio classifier: a comparative study," International Journal of Current Engineering and Technology, vol. 6, pp. 477-482, 2016.

[16] A. H. Mohammad, T. Alwada'n, and O. Al-Momani, "Arabic text categorization using support vector machine, Naïve Bayes and neural network," GSTF Journal on Computing (JoC), vol. 5, pp. 1-8, 2016.

[17] E. Al-Thwaib, W. Al-Romimah, and et al., "Support vector machine versus k-nearest neighbor for Arabic text classification," International Journal of Sciences, vol. 3, pp. 1--5, 2014.

[18] R. M. Sallam, H. Mousa, and M. Hussien, "A Comparative Study for Arabic Text Classification Based on BOW and Mixed Words Representations," IJCI. International Journal of Computers and Information, vol. 5, pp. 24-34, 2016.

[19] K. Abidi, Z. Elberichi, and Y. Tlili Guissa, "Arabic text categorization: a comparative study of different representation modes," Journal of Theoretical and Applied Information Technology, vol. 38, pp. 1-5, 2012.

[20] A. Moh'd A Mesleh, "Chi square feature extraction based svms arabic language text categorization system," Journal of Computer Science, vol. 3, pp. 430-435, 2007.

[21] K. Al-Hindi and E. Al-Thwaib, "A Comparative Study of Machine Learning Techniques in Classifying Full-Text Arabic Documents versus Summarized Documents.," World of Computer Science & Information Technology Journal, vol. 3, 2013.

[22] A. M. El-Halees, "A comparative study on Arabic text classification," Egyptian Computer Science Journal, vol. 30, 2008.

[23] S. Alsaleem and et al., "Automated Arabic Text Categorization Using SVM and NB.," Int. Arab. J. e Technol., vol. 2, pp. 124--128, 2011.

[24] S. Al-Harbi, A. Almuhareb, A. Al-Thubaity, M. S. Khorsheed, and A. Al-Rajeh, "Automatic Arabic text classification," in JADT 2008: 9es Journées Internationales d'Analyse Statistique des Données Textuelles, 2008, pp. 77-83.

[25] J. Ababneh, "Application of Naïve Bayes, Decision Tree, and K-Nearest Neighbors for Automated Text Classification," Modern Applied Science, vol. 13, p. 31, 2019.

[26] M. A. R. Abdeen, S. AlBouq, A. Elmahalawy, and S. Shehata, "A closer look at arabic text classification," Int. J. Adv. Comput. Sci. Appl, vol. 10, pp. 677--688, 2019.

[27] M. Ahmed and R. Elhassan, "Arabic text classification review," International Journal of Computer Science and Software Engineering, vol. 4, pp. 1--5, 2015.

[28] A. M. F. Al-Sbou, "A survey of arabic text classification models," International Journal of Electrical and Computer Engineering (IJECE), vol. 8, pp. 4352--4355, 2018.

[29] R. Ayadi, M. Maraoui, and M. Zrigui, "A Survey of Arabic Text Representation and Classification Methods.," Res. Comput. Sci., vol. 117, pp. 51--62, 2016.

[30] A. H. Mohammad, "Arabic text classification: A review," Modern Applied Science, vol. 13, 2019.

[31] M. Sayed, R. K. Salem, and A. E. Khder, "A survey of Arabic text classification approaches," International Journal of Computer Applications in Technology, vol. 59, pp. 236-251, 2019.

[32] K. A. Wahdan, S. Hantoobi, S. A. Salloum, and K. Shaalan, "A systematic review of text classification research based ondeep learning models in Arabic language," Int. J. Electr. Comput. Eng, vol. 10, pp. 6629-6643, 2020.

[33] A. Khatun, M. Mafiul Hasan, A. Miah, and R. Miah, "Comparative Study on Text Classification," Int. J. Eng. Sci. Invent, vol. 9, pp. 21-33, 2020.

[34] Q. Li, H. Peng, J. Li, C. Xia, R. Yang, L. Sun, P. S. Yu, and L. He, "A survey on text classification: From shallow to deep learning," arXiv preprint arXiv:2008.00364, 2020.

[35] P. Y. Pawar and S. Gawande, "A comparative study on different types of approaches to text categorization," International Journal of Machine Learning and Computing, vol. 2, p. 423, 2012.

[36] S. Ramasundaram and S. Victor, "Algorithms for text categorization: A comparative study," World Applied Sciences Journal, vol. 22, pp. 1232-1240, 2013.

[37] E. Selab and A. Guessoum, "Building TALAA, a Free General and Categorized Arabic Corpus," in ICAART (1), 2015, pp. 284-291.

[38] O. Einea, Elnagar, A., & Al Debsi, R., "SANAD: Single-Label Arabic News Articles Dataset for Automatic Text Categorization.," Mendeley Data, V2, doi: 10.17632/57zpx667y9.2., 2019.

[39] B. Al-Salemi, M. Ayob, G. Kendall, and S. A. Mohd Noah, "RTAnews: A Benchmark for Multi-label Arabic Text Categorization," Mendeley Data, V1, doi: 10.17632/322pzsdxwy.1, 2018.

[40] R. E. Al-Debsi, Ashraf; Einea, Omar, "NADiA: News Articles Dataset in Arabic for Multi-Label Text Categorization," Mendeley Data, vol. 2, doi: 10.17632/hhrb7phdyx.2, 2019.

[41] P. C. Team. (2021). Python 3.9.7 documentation. Python Software Foundation. Available: https://docs.python.org/3/. Last visited: June 2022.

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

[43] S. Loria. (2018). Textblob documentation. Release 0.15, 2, 269. Available: https://buildmedia.readthedocs.org/media/pdf/textblob/latest/textblob.pdf. Last visited: June 2022.

[44] T. Zerrouki, "Tashaphyne, Arabic light stemmer.", 2019.

[45] R. Rehurek and P. Sojka, "Software framework for topic modelling with large corpora," in Proceedings of the LREC 2010 workshop on new challenges for NLP frameworks, Universoty of Malta, 2010.

[46] C. R. Harris, K. J. Millman, S. J. Van Der Walt, R. Gommers, P. Virtanen, D. Cournapeau, E. Wieser, J. Taylor, S. Berg, and N. J. Smith, "Array programming with NumPy," Nature, vol. 585, pp. 357-362, 2020.

[47] L. Buitinck, G. Louppe, M. Blondel, F. Pedregosa, A. Mueller, O. Grisel, V. Niculae, P. Prettenhofer, A. Gramfort, J. Grobler, and et al., "API design for machine learning software: experiences from the scikit-learn project," presented at the European Conference on Machine Learning and Principles and Practices of Knowledge Discovery in Databases, Prague, Czech Republic., 2013.

[48] F. Pedregosa, G. Varoquaux, A. Gramfort, V. Michel, B. Thirion, O. Grisel, M. Blondel, P. Prettenhofer, R. Weiss, and V. Dubourg, "Scikit-learn: Machine learning in Python," the Journal of machine Learning research, vol. 12, pp. 2825-2830, 2011.

[49] M. F. McTear, Z. Callejas, and D. Griol, "The conversational interface," vol. 6, p. 94, 2016.

[50] G. Grefenstette, "Tokenization," in Syntactic Wordclass Tagging, ed: Springer, 1999, pp. 117-133.

[51] C. R. Severance, Python for Everybody: Exploring Data in Python 3: CreateSpace Independent Publishing Platform, 2016.

[52] A. Rajaraman and J. D. Ullman, Mining of massive datasets: Cambridge University Press, 2011.

[53] T. Mikolov, I. Sutskever, K. Chen, G. S. Corrado, and J. Dean, "Distributed representations of words and phrases and their compositionality," Advances in neural information processing systems, vol. 26, pp. 3111-3119, 2013.

[54] Q. Le and T. Mikolov, "Distributed representations of sentences and documents," in International conference on machine learning, 2014, pp. 1188-1196.

[55] D. M. Jurafsky, J. H. (Draft of September 21, 2021). Speech and Language Processing. Chapter 3: N-gram Language Models. Retrieved from: https://web.stanford.edu/~jurafsky/slp3/3.pdf. Last visited: June 2022.

[56] K. Weinberger, A. Dasgupta, J. Langford, A. Smola, and J. Attenberg, "Feature hashing for large scale multitask learning," in Proceedings of the 26th annual international conference on machine learning, 2009, pp. 1113-1120.

[57] G. Kukreja, D. Bahl, and R. Gupta, "The Impact of FinTech on Financial Services in India: Past, Present, and Future Trends," in Innovative Strategies for Implementing FinTech in Banking, ed: IGI Global, 2021, pp. 191-200.

[58] C. Cortes and V. Vapnik, "Support-vector networks," Machine learning, vol. 20, pp. 273-297, 1995.

[59] D. W. Hosmer Jr, S. Lemeshow, and R. X. Sturdivant, Applied logistic regression vol. 398: John Wiley & Sons, 2013.

[60] M. Arashi, A. M. E. Saleh, and B. G. Kibria, Theory of ridge regression estimation with applications: John Wiley & Sons, 2019.

[61] L. Bottou and O. Bousquet, "The tradeoffs of large scale learning. Optimization for Machine Learning," ed, 2011.

[62] M. N. Murty and R. Raghava, "Support Vector Machines and Perceptrons: Learning, Optimization, Classification, and Application to Social Networks," 2016.

[63] J. Lu, P. Zhao, and S. C. Hoi, "Online passive-aggressive active learning," Machine learning, vol. 103, pp. 141-183, 2016.

[64] A. M. Martinez and A. C. Kak, "Pca versus lda," IEEE transactions on pattern analysis and machine intelligence, vol. 23, pp. 228-233, 2001.

[65] T. Cover and P. Hart, "Nearest neighbor pattern classification," IEEE transactions on information theory, vol. 13, pp. 21-27, 1967.

[66] C. K. Williams and C. E. Rasmussen, Gaussian processes for machine learning vol. 2: MIT press Cambridge, MA, 2006.

[67] J. Brownlee, "Master Machine Learning Algorithms: Discover How They Work and Implement Them From Scratch, 2016," URL https://books.google.com/, 2016. Last visited: June 2022.

[68] S. Shalev-Shwartz and S. Ben-David, Understanding machine learning: From theory to algorithms: Cambridge university press, 2014.

[69] L. Breiman, "Bagging predictors," Machine learning, vol. 24, pp. 123--140, 1996.

[70] L. Breiman, "Random forests," Machine learning, vol. 45, pp. 5--32, 2001.

[71] T. Hastie, S. Rosset, J. Zhu, and H. Zou, "Multi-class adaboost," Statistics and its Interface, vol. 2, pp. 349-360, 2009.

[72] S. M. Piryonesi and T. E. El-Diraby, "Data analytics in asset management: Cost-effective prediction of the pavement condition index," Journal of Infrastructure Systems, vol. 26, p. 04019036, 2020.

[73] A. I. Naimi and L. B. Balzer, "Stacked generalization: an introduction to super learning," European journal of epidemiology, vol. 33, pp. 459-464, 2018.

[74] S. Abirami and P. Chitra, "Energy-efficient edge based real-time healthcare support system," in Advances in Computers. vol. 117, ed, 2020, pp. 339--368.

[75] C. E. Metz, "Basic principles of ROC analysis," in Seminars in nuclear medicine, 1978, pp. 283-298.

[76] T. Buckwalter, "Issues in Arabic orthography and morphology analysis," in proceedings of the workshop on computational approaches to Arabic script-based languages, 2004, pp. 31-34.