International Journal of Cloud Applications and Computing
Volume 14 • Issue 1

Detecting Ambiguities in Requirement Documents Written in Arabic Using Machine Learning Algorithms

Ahmad Althunibat, Al-Zaytoonah University of Jordan, Jordan
Bayan Alsawareah, Al-Zaytoonah University of Jordan, Jordan
Siti Sarah Maidin, INTI International University, Malaysia
https://orcid.org/0000-0003-0714-2186
Belal Hawashin, Al-Zaytoonah University of Jordan, Jordan
Iqbal Jebril, Al-Zaytoonah University of Jordan, Jordan
Belal Zaqaibeh, Jadara University, Jordan
Haneen A. Al-khawaja, Applied Science Research Center, Applied Science Private University, Jordan*
https://orcid.org/0000-0003-4607-9394

ABSTRACT

The identification of ambiguities in Arabic requirement documents plays a crucial role in requirements engineering. This is because the quality of requirements directly impacts the overall success of software development projects. Traditionally, engineers have used manual methods to evaluate requirement quality, leading to a time-consuming and subjective process that is prone to errors. This study explores the use of machine learning algorithms to automate the assessment of requirements expressed in natural language. The study aims to compare various machine learning algorithms according to their abilities in classifying requirements written in Arabic as decision tree. The findings reveal that random forest outperformed all stemmers, achieving an accuracy of 0.95 without employing a stemmer, 0.99 with the ISRI stemmer, and 0.97 with the Arabic light stemmer. These results highlight the robustness and practicality of the random forest algorithm.

KEYWORDS

Arabic Requirements Ambiguous Requirement Classification, Machine Learning AraBERT, Software Engineering, Unambiguous Requirements

Software engineering is the application of engineering principles to ensure the quality, reliability, and efficiency of software systems. The Institute of Electrical and Electronics Engineers (IEEE) defines software engineering as "the application of a systematic, disciplined, quantifiable approach to the development, operation, and maintenance of software" (IEEE, 1990). As technology advances, software engineering faces challenges such as the classification and ambiguity of software system requirements. Requirements are fundamental to software development, and their ambiguity can lead to misunderstandings, miscommunications, and project failure. Understanding and addressing these

DOI: 10.4018/IJCAC.339563
*Corresponding Author

challenges is crucial, and model-driven engineering approaches can enhance requirement clarity and consistency.

A significant obstacle for academics and developers working on Arabic natural language processing (NLP) applications is the scarcity of Arabic datasets, particularly in recognizing ambiguous requirements. Failure to detect and resolve ambiguity early in the software requirements specification (SRS) can result in various defects with serious consequences for the software development process. Ambiguities in software requirements, defined as sentences with multiple meanings, can lead to faults and compromise software reliability, especially in safety-critical systems.

This research addresses the challenge of detecting ambiguous requirements in Arabic language software development by proposing a novel approach that employs supervised learning algorithms for automated classification. This approach contributes significantly to Arabic NLP, offering a potential solution to improve software development accuracy in Arabic-speaking regions. By providing a robust means of detecting ambiguous requirements, the proposed method has the potential to reduce risks associated with Arabic software development, ultimately leading to higher-quality software products.

The importance of an efficient method for classifying Arabic requirements as ambiguous or non-ambiguous is twofold. First, in the research domain, as there are limited prior studies in this area, this method serves as a pioneering effort that could stimulate further research. Second, from a practical standpoint, various stakeholders stand to benefit from improved requirements engineering processes.

The key contributions of this work include proposing the use of machine learning (ML) to classify ambiguous requirements in Arabic, comparing several classifiers to identify the most effective method for this task, and creating a dataset of Arabic requirements for future research. The subsequent sections encompass a literature review, methodology, experimental work, and conclusions, providing a comprehensive exploration of the proposed approach.

RELATED WORK

In the realm of software development, understanding stakeholder needs is crucial for designing complex software systems (Althunibat et al., 2022). Stakeholders, often users, contribute NLP-written requirements for large-scale projects. Ko et al. (2007) proposed an approach wherein initial data needs are automatically categorized into topics, reflecting political analyst perspectives. Experiments, utilizing datasets in both Korean and English, validate the efficacy of this strategy. This highlights the potential for an internet-based requirements analysis-supporting system to efficiently gather and evaluate dispersed end-user requirements via the network.

Moving forward, support vector machine (SVM) algorithms have garnered attention for their ideal academic characteristics and high performance (Al Qaisi et al., 2021). Yang et al. (2010) delved into the analysis of support vector characteristics, presenting a novel learning process that incorporates SVM classification algorithms. The algorithm, rooted in the equivalence of classification between support vector sets, employs incremental learning to accumulate data. Experimental results indicate its potential to expedite training processes, reduce storage costs, and maintain organizational accuracy (Quba et al., 2021).

Artificial intelligence (AI) and deep learning (DL) come to the forefront in the work of Navarro-Almanza et al. (2017). They recommend using a convolutional neural network (CNN) model to categorize software requirements, showcasing promising results on the PROMISE corpus dataset. This dataset, with pre-grouped and labeled criteria for both functional requirements (FR) and non-functional requirements (NFR), serves as a valuable resource for evaluating the suggested model. (Gill et al., 2014)

Lu and Liang (2017) further contributed to understanding user requirements by breaking them down into FRs and NFRs, including usability, portability, performance, and reliability. Their research involved diverse methods such as bag of words (BoW), CHI2, TF-IDF, and AUR-BoW, as well as ML algorithms like J48, naive Bayes, and bagging. Comparative analysis reveals that the bagging

ML algorithm provides the best categorization outcome for NFRs, as validated by feedback from actual customers.

In the domain of ML techniques for classifying FR phrases, AlZu'bi and Jararweh (2020) introduced a novel approach that integrates information from various ML models. This method, implemented and trained using a single dataset, aims to enhance the accuracy and quality of FR classification.

To address imbalanced classes and improve classifier performance, Kurtanović and Maalej (2017) propose a strategy applying cross-validation to classifiers. Their focus is on the automatic identification of NFRs, particularly in the categories of security, usability, operations, and performance. This involves preprocessing steps such as stopword and punctuation removal, coupled with feature selection using BoW, bigrams, and trigrams. Notably, the inclusion of part-of-speech tags emerges as a highly informative feature in their experiments using the SVM classifier algorithm.

The landscape of software requirement classification is further enriched by exploring various methodologies (Alsawareah et al., 2023; Al-Kasabera et al., 2020). These studies aimed to establish correlations between software architecture and NFRs, emphasizing the significance of considering software architecture in addressing NFRs within the software development life cycle.

In order to identify uncertainties in Arabic requirement documents through the utilization of ML algorithms, it is crucial to comprehend the distinct obstacles associated with the Arabic language and the implementation of ML in the field of NLP. Elazhary (2016) emphasized the origins of uncertainties in the Arabic language and effective strategies for composing Arabic user requirements to prevent the introduction of uncertainties. This offers valuable insights into the distinctive linguistic attributes of Arabic that must be taken into account when designing ML algorithms for detecting ambiguity. In addition, Ezzini et al. (2021) examined the management of ambiguity in requirements engineering and the creation of automated systems to detect ambiguity. These sources serve as a basis for comprehending the current research on the topic and the possibility of creating ML-driven solutions for identifying uncertainties in Arabic requirement papers. Furthermore, they provide an extensive examination of Arabic morphological analysis methodologies, which is essential for comprehending the linguistic characteristics of Arabic that may give rise to misunderstandings in requirement papers. Acquiring this comprehension is crucial for constructing efficient ML algorithms customized to the distinctive linguistic attributes of Arabic. Additionally, Das et al. (2020) and Yang et al. (2010) examined the use of ML algorithms in many fields, showcasing the capacity to utilize ML for identifying ambiguity in natural language needs. When these sources are combined, it becomes clear that the literature offers useful knowledge on the difficulties of identifying ambiguity in Arabic requirement papers and the possibility of using ML methods to tackle these difficulties. Comprehending the linguistic subtleties of Arabic, current methodologies for detecting ambiguity, and the use of ML in related fields is essential for the development of efficient ML algorithms to detect ambiguities in Arabic requirement documents. (Nigam et al., 2012).

This synthesis of recent literature demonstrates the evolving landscape of software requirements analysis, encompassing diverse techniques and methodologies that contribute to the enhancement of software development processes.

METHODOLOGY

This section introduces a novel methodology that leverages ML for the identification and resolution of ambiguity in Arabic software requirements. Illustrated in Figure 1, a multistep approach was employed to attain the outlined objective. The following overview summarized the methodology, encompassing the application of decision trees, logistic regression, K-nearest neighbors (KNN), SVMs, random forest, XGBoost, and DL models, particularly the AraBERT model. Additionally, it outlines the experimental design, data preprocessing, model optimization, feature selection, and evaluation methodologies implemented to ensure the robustness and efficacy of the proposed approach.

Figure 1. Training classifier for Arabic ambiguous requirements

Algorithm 1: Training Classifier for Arabic Ambiguous Requirements
- Input: Dataset of Arabic Requirements, Classifier
- Output: Trained Model

1. Preprocess the Arabic Requirements Dataset
2. Remove non-Arabic symbols, characters, and symbols from the text.
3. Tokenize the text into individual words or subworld's using the nltk tokenize.
4. Remove stop words from the tokenized text.
5. Feature Extraction.
6. Feature Selection by applying scaling.
7. Initialize the classifier.
8. Train the classifier using the preprocessed dataset and selected features.
9. The trained model is the output of the algorithm.
10. Return the trained model.

Research Design

The research design in this section unfolds through a series of sequential steps, as depicted in the diagram below, aimed at the development and evaluation of a model. The initial step involved the collection of data from various sources to compile a dataset of Arabic data. Subsequently, the collected data underwent preprocessing, including activities such as cleaning and transformation to ensure their quality. Following the preprocessing phase, features were extracted from the dataset to pinpoint the most informative attributes. Further refinement of these properties was achieved through feature selection. Once the feature selection process was completed, a suitable ML method was selected, and the model was trained using the preprocessed and feature-selected dataset. Finally, the performance of the model was rigorously evaluated using metrics such as accuracy, precision, recall, or F1 score. This comprehensive research strategy is designed to ensure the construction of an effective and reliable model, yielding accurate and valuable insights for the research study.

Figure 2 displays the aredication for arabic software requirements, as shown in the following.

Data Collection

Using various resources such as Google and other search engines, comprehensive data were collected to obtain a reliable dataset. A number of experts were consulted to validate the relevance and quality of the collected requirements.

Data Preprocessing

The second phase in our methodology was to preprocess the raw data so that it could be analyzed further. For data preprocessing, we used the following steps:

Upload File: The software requirements data file was in Txt format.

Text Preprocessing: The extracted requirements contained several steps. These steps included the following:

Figure 2. Predication for Arabic software requirements

Algorithm 2: Predication for Arabic Software Requirements
- Input: Trained model, Testing Arabic Requirements
- Output: Classification of Arabic Requirements

1. Load the trained model.
2. Preprocess the testing Arabic requirements:
3. Remove non-Arabic characters and symbols from the text using regular expressions.
4. Tokenize the text into individual words or sub worlds using the nltk tokenize.
5. Remove stop words from the tokenized text.
6. Apply the same feature extraction and feature selection techniques used during training on the preprocessed testing requirements.
7. Use the trained model to classify the preprocessed testing requirements.
8. The output is the predicted classification of the Arabic requirements (Labeled the testing data)
9 Return Classification
.

Tokenization: Tokenization was one of the most critical data preprocessing procedures, and it also had significant effects on the overall performance of the model when it came to the text classification ta sk. Tokenizing the requirements involved breaking them down into separate words or *tokens*, which in turn made it easier to process and analyze them. This essential step was highlighted (Petrović & Stanković, 2019). In this study we used the **nltk.word_tokenize** () function from the NLTK library.

Removal of Non-Arabic Text: During this step, all non-Arabic text was eliminated to ensure that solely Arabic text remained for analysis.

Stopword Removal: Stopwords, which are commonly used words with little semantic value (e.g., articles, prepositions), were removed from the requirements. This step helped eliminate noise and focused on more meaningful words. In this study we utilized the list of stopwords from the NLTK library for Arabic.

Table 1. Removing non-Arabic text

After Removing Non-Arabic Text	Arabic Requirements
يجب أن يكون البرنامج متوافقًا مع أحدث إصدارات متصفحات الويب الشائعة مثل	يجب أن يكون البرنامج متوافقًا مع أحدث إصدارات متصفحات الويب الشائعة ، مثل Safari و Firefox و Chrome

Table 2. Stopword removal

After Applying Stopword Removal	Arabic Requirements
برنامج متوافقًا أحدث إصدارات متصفحات الويب الشائعة	يجب أن يكون البرنامج متوافقًا مع أحدث إصدارات متصفحات الويب الشائعة مثل

Normalization: In addition to the processes that have been discussed so far, the process of normalization is another method that may be used to continue the standardization of text data. Text normalization consists of a set of procedures that are designed to clean and standardize textual data into a format that can be used as input by other NLP and analytics systems and apps (Lima et al., 2019). The end goal of these stages is to make the data more usable by other systems and applications.

Stemming: We use stemming to reduce words to their base or root form. We utilized the Arabic light stemmer (Larkey et al., 2002) / ISRI stemmer (Taghva et al., 2005) algorithm from the NLTK package for Arabic text stemming, which is particularly intended for this purpose.

Feature Weighting

The TF-IDF (Term Frequency-Inverse Document Frequency) method measures the importance of a word in a document or collection or corpus (Rajaraman & Ullman, 2011) Generally, it is used in searching for information retrieval, text mining, and user modeling which in the formula that's used to compute is:

$$tf - idf \text{ is } tf - idf\left(t,d\right) = tf\left(t,d\right) * idf\left(t\right)$$

Using this method we extracted features by determining the importance of terms in each requirement, which determined the most pertinent terms for a requirement on the basis of the frequency and significance of terms within the requirement. Using the BoW technique, we classified each requirement as a collection of individual words or phrases without addressing their order. This method generates a unique dictionary of terms by constructing a vocabulary from all of the dataset's requirements. Consequently, each requirement is represented by a feature vector in which each dimension corresponds to a vocabulary term (Sarkar, 2016).

Feature Selection

Feature selection is essential for making large problems computationally efficient, that is, for conserving computation, storage, and network resources during the training phase and subsequent classifier application (Forman, 2003).

We used feature selection techniques to reduce the dimensionality of the feature space and to select the most important characteristics for classification. We used the chi-square feature selection method, which is available in SelectKBest in Python method from scikit-learn; it was chosen because it uses the chi-square test to calculate scores. This method selects the K characteristics with the highest scores on the basis of their relevance to the variable of interest. K, representing the number of selected features, is set to 10% of the total number of features. The exact number of features depends on the initial number of features in the TF-IDF representation of the data.

Data Transformation

Following preprocessing and suitable data selection, we trained and improved ML models that can identify ambiguous software requirements. We took the following actions to achieve this:

Scaling: To scale feature vectors to a specific range, we utilized the MinMaxScaler function from scikit-learn. This process ensures that each feature is given the same weight throughout model training.

Hyperparameter Tuning: To find the best set of hyperparameters for the classifiers model, we did a grid search using GridSearchCV from scikit-learn.

Learning Phase

We used six classifiers and the AraBERT model to learn from this data.

ML Algorithms

We employed several types of ML approaches, such as decision trees, logistic regression, KNN, SVM, random forest, and XGBoost, to automate the process of detecting ambiguous Arabic software requirements. We employed these particular models because of their documented track record of exceptional performance in prior research, including decision tree, random forest, and SVM. Moreover, our selection encompassed contemporary models like AraBERT and XGBoost, which represent recent advancements in the field. This allowed us to streamline the process significantly. Decision trees provide a straightforward paradigm for understanding hierarchical organization of if-then criteria. Logistic regression is a common method used in classification problems that involve evaluating the chance of correctly identifying ambiguous software requirements. KNN organizes demands into categories according to the degree to which they resemble their immediate surroundings. The purpose of SVMs is to locate the hyperplane that provides the most accurate differentiation between ambiguous and unambiguous criteria (Amro et al., 2023). In contrast to XGBoost, which uses boosting methods to produce a set of weak learners, random forest mixes a large number of decision trees in an attempt to enhance accuracy. (Dhaliwal et al., 2018)

Decision Trees

Decision trees are algorithms developed by Hunt et al. (1966) to describe a recursive partition of the instance space (Rokach & Maimon, 2005). The automated analysis of unclear Arabic software requirements may be modeled using decision trees, which provide an interpretable model of the process. These models construct a hierarchical framework of if-else conditions by retrieving attributes from the criteria and basing their decisions on those features. By recursively splitting the data on the basis of the qualities that provide the most insightful information, decision trees can effectively classify requirements as ambiguous or unambiguous. The transparency of decision trees is one of its many advantages. This enables domain experts to quickly analyze and comprehend the rules that are generated by decision trees, which in turn provides insights into the factors that contribute to ambiguity in software requirements. In this study, we used DecisionTreeClassifier class from the sklearn.tree module.

Logistic Regression

The use of logistic regression is a common approach for assessing the likelihood of an observation's being associated with a particular category, being associated with a particular category as outlined by Berkson (1944). Logistic regression, a form of regression analysis, relies on an independent variable to predict the dependent variable, as indicated Elsaid et al. (2015). Specifically, binary logistic regression is well suited for situations where the dependent variable exhibits two distinct categories. Conversely, multinomial logistic regression proves valuable when the dependent variable comprises more than two categories. For the purposes of this study, binary classification was employed.

KNN

The foundational principle of this technique is based on the fact that data points within a collection exhibit similar attributes to those in proximity (Fix & Hodges, 1951). In order to categorize novel data, the approach involves assessing its proximity to preexisting data points stored in the database, identifying the k-nearest points, and subsequently calculating their average or mode for regression tasks (Kotsiantis, 2011; Althunibat, 2023).

SVMs

As a supervised approach for classification and regression, SVMs are a very effective and complex ML model (Boser et al., 1992). They have the ability to do linear and nonlinear regression and classification, as well as spot anomalies. Classification is achieved by constructing a linear hyperplane with the greatest separation between categories. The likelihood of incorrectly categorizing new cases is decreased because of this buffer, which also prevents data from being separated from the sample (Ford & VanderPlas, 2016).

Random Forest

Forest (Ho, 1995), which is an ensemble learning approach, uses many decision trees to increase accuracy while avoiding overfitting. Each decision tree in the random forest is trained on a random subset of the data, and the final forecast is formed by voting or averaging the forecasts of all the trees. Random forests excel in dealing with noise, detecting complex feature interactions, and estimating feature values accurately. By pooling predictions from several trees, random forest improves overall effectiveness in identifying ambiguous Arabic software requirements.

XGBoost

XGBoost is a powerful ensemble learning approach that assembles a collection of weak learners using gradient-boosting methods. It incrementally adds decision trees to the model, with each successive tree trained to correct the flaws introduced by the previous trees. XGBoost uses regularization methodologies and efficient algorithms to increase the model's performance. By combining the predictions of several weak learners, XGBoost achieves high accuracy and resilience, making it a viable option for the automated identification of ambiguous software requirements (Alsoub et al, 2018).

DL

AraBERT is a language model that has been pre-trained for Arabic language processing tasks. It has been trained on a massive dataset of Arabic text, giving it a deep comprehension of the language and the capacity to do tasks like language translation, text categorization, and sentiment analysis with extreme precision. Because of its pre-training on a huge dataset, AraBERT requires relatively little further training data to be fine-tuned for unique applications. Such a model would be highly beneficial in accelerating the process and lowering costs when developing and deploying NLP models (Aftan & Shah, 2023). In this study, we used AraBERT v2.

EXPERIMENTS AND RESULTS

Our work on the automatic detection of ambiguous Arabic software requirements using ML approaches is presented in this section, along with the experimental setup and findings. We describe the datasets used, the experimental setup, the evaluation metrics and results of the experiments, the ML algorithms that were used, the hyperparameter tuning and optimization strategies employed, and the analysis of the results.

Dataset Description

This study used 400 Arabic software requirements gathered from different resources such as Google; the average of the ambiguous requirements words was 9.625, and the unambiguous words were 14.94. After that, we divided them into ambiguous and unambiguous categories. The requirements in our dataset lacked a specific structure and were presented as free text, which is a common characteristic of requirements. To assess the presence of ambiguity, two experts independently categorized each requirement as either ambiguous or unambiguous on the basis of their domain knowledge. Out of the 400 requirements, 200 were identified as ambiguous, signifying statements that could be interpreted

in multiple ways or that lacked clarity, while the remaining 200 were deemed unambiguous. The experts engaged in detailed discussions and reached a consensus on the categorization, demonstrating high inter-rater reliability (Al Thunibat et al, 2011; Jebril et al., 2023).

This dataset effectively illustrates the challenges posed by ambiguous requirements and underscores the advantages of clear and unambiguous requirements in software development projects, rendering it a valuable resource for addressing requirement ambiguity.

To evaluate the effectiveness of the proposed methods using this dataset, we employed a cross-validation approach with five folds (Jaber et al, 2020; Adaileh, 2020). This widely adopted method for assessing classification performance involves dividing the data into five parts, where four segments are utilized for training and the fifth segment for testing. This process is iterated five times, ensuring that all data are used for testing once. Cross-validation offers increased accuracy in comparison to a simple train-test split, as it comprehensively considers the entire dataset during evaluation, providing a more robust assessment of performance.

Evaluation Measurements

The performance of a classifier is assessed mainly using evaluation metrics. The accuracy of the model's predictions is determined by comparing the model's predictions to the database's actual values using mathematical methods. Precision is defined as the proportion of correctly identified samples to the total number of samples. Precision is calculated by dividing the total number of correct classifications by the total number of classifications completed.

The confusion matrix compares the model's errors and successes to the expected result. True positive, true negative, false positive, and false negative concepts are defined on the basis of the classification of the response given and the actual response in the database. (Alshehadeh & Al-Khawaja, 2022; Abuhamdah, et al. 2021)

Accuracy

Accuracy is a broader metric because it estimates the total number of correct classifications. When the target classes in the data are (roughly) balanced, this is a good measure:

Table 3. Sample of Arabic dataset

Ambiguous	يجب أن يكون النظام سريعاً
Ambiguous	يجب أن تكون واجهة المستخدم سهلة الاستخدام
Ambiguous	يجب أن يكون التطبيق قابلاً للتطوير
Unambiguous	يجب أن يسمح التطبيق للمستخدمين بمعاينة مدخلات النموذج الخاصة بهم وأ التحقق منها قبل إرسال النموذج
Unambiguous	يجب أن يوفر البرنامج خياراً للمستخدمين للاشتراك وأ إلغاء الاشتراك في النشرات الإخبارية أو التحديثات عبر البريد الإلكتروني
Unambiguous	يجب أن يسمح النظام للمستخدمين بتصفية نتائج البحث حسب نطاق التاريخ وأ فترات زمنية محددة

Table 4. Confusion matrix

		Yes	No
Real	Yes	True Positive (TP)	False Negative (FN)
	No	False Positive (FP)	True Negative(TN)

$$Accuracy = \frac{TP + TN}{TP + TN + FP + FN} \tag{1}$$

Precision

Precision is determined by calculating the ratio of the total correct positive predictions to the total positive predictions made for a specific class.

$$Precision = \frac{TP}{TP + FP} \tag{2}$$

Recall

Recall, also known as true positive rate or sensitivity, represents the percentage of actual positive cases that a classifier correctly identifies. Equation 3 likely refers to the specific mathematical formula used to calculate recall in your context. Typically, the formula for recall is given as

$$Recall = \frac{TP}{TP + FN} \tag{3}$$

This formula measures the proportion of positive instances that were correctly classified by the model out of all actual positive instances.

F1-Score

The F1-Score can be calculated by considering both precision and recall simultaneously.

$$F1Score = \frac{2 * \left(Precision * Recall\right)}{Precision + Recall} \tag{4}$$

Experimental Setting

The research made use of the Colab environment, a cloud-based platform offering access to high-performance computing resources. Leveraging the Colab environment significantly facilitated collaborative endeavors by providing a platform for seamless sharing and execution of code and data analysis tasks. The study operated with the default Colab hardware configuration, comprising an Intel Xeon CPU equipped with 2 vCPUs (virtual CPUs) and 13GB of RAM.

Experimental Result

ML Results

Table 5 summarizes the performance metrics for several ML algorithms using various stemmers, such as No Stemmer, ISRI Stemmer, and Arabic Light Stemmer. Accuracy, precision, recall, and F1 score are among the criteria considered.

Random forest and KNN had the top two performances among the algorithms. KNN outperformed all stemmers in terms of accuracy, scoring 0.96 without a stemmer, 0.97 with ISRI Stemmer, and 0.97 with Arabic Light Stemmer. This demonstrates the KNN algorithm's resilience and generalizability. Random forest also performed well, with an accuracy of 0.95 without a stemmer, 0.99 with the ISRI Stemmer, and 0.97 with the Arabic Light Stemmer. KNN is well suited for the job at hand because

of its capacity to identify data points on the basis of their closeness to nearby samples (Wasmi et al, 2021; Alrwele, 2023).

Random forest and KNN both exhibited constant performance across multiple stemmers, showing their adaptability to various text preparation approaches. As a result, these two methods may be deemed the best in this examination. However, the ultimate decision among them may be influenced by other considerations such as processing efficiency, interpretability, and the application's unique needs (Alrawashdeh et al, 2021).

Figure 3 displays the results of the ML algorithm without the application of a stemmer, as shown in the bar chart.

Figure 4 presents the results of the ML algorithm with the implementation of the ISRI stemmer, as depicted in the bar chart.

In Figure 5, the bar chart depicts the performance of the ML algorithm with the application of the ISRI stemmer.

AraBERT Results

Table 6 provides a comprehensive overview of the AraBERT model's performance across various random states, each evaluated at 5 epochs. The results highlight the model's consistency and effectiveness in Arabic text classification.

Table 5. The top results

Stemmer	No Stemmer				ISRI Stemmer				Arabic Light Stemmer			
Evaluation / ML	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1	Accuracy	Precision	Recall	F1
DT	0.93	0.94	0.93	0.93	0.96	0.96	0.96	0.96	0.95	0.96	0.95	0.95
KNN	0.96	0.96	0.96	0.96	0.97	0.98	0.97	0.97	0.97	0.97	0.97	0.97
SVM	0.95	0.95	0.95	0.95	0.97	0.97	0.97	0.97	0.96	0.96	0.96	0.95
RF	0.95	0.95	0.95	0.95	0.99	0.99	0.99	0.99	0.97	0.97	0.97	0.97
LR	0.96	0.96	0.96	0.95	0.95	0.95	0.96	0.95	0.95	0.95	0.95	0.95
XG	0.95	0.96	0.95	0.95	0.98	0.98	0.98	0.98	0.96	0.96	0.96	0.95

Figure 3. Bar chart for ML algorithm when no stemmer is applied

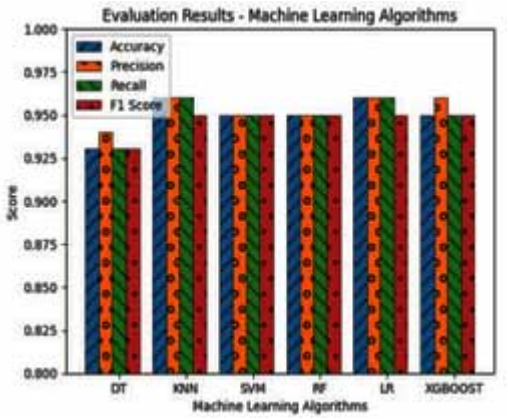


Figure 4. Bar chart for ML algorithm when ISRI stemmer is applied

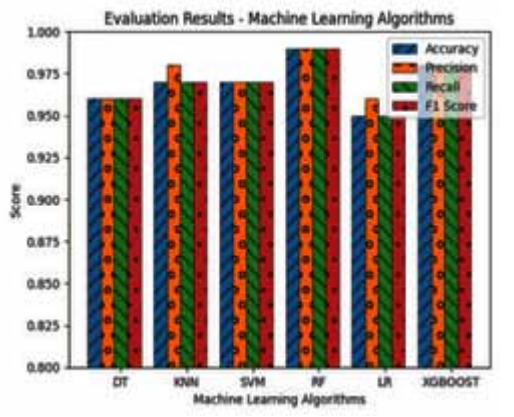


Figure 5. Bar chart for ML algorithm with ISRI stemmer

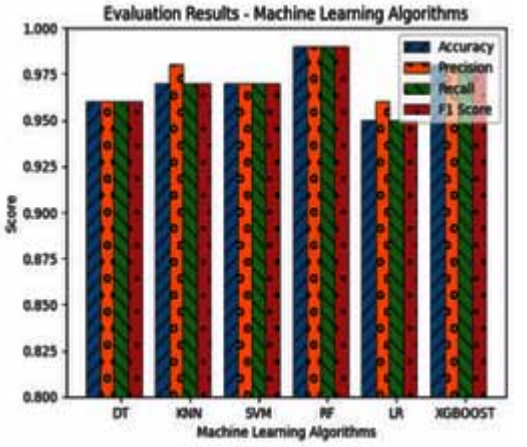


Table 6. AraBERT model results with 5 epochs

Random state Values	Precision	Recall	F1 Score
Random state =5	0.96	0.96	0.96
Random state =10	0.97	0.98	0.97
Random state =15	0.97	0.97	0.97
Random state =20	100	100	100
Random state =42	0.99	0.99	0.99
Average	**0.978**	**0.98**	**0.978**

For the random state of 5, the AraBERT model exhibited commendable accuracy, recall, and F1 score metrics, all registering at 0.96. This indicates a high level of correctness and reliability in the model's predictions.

With the random state set to 10, the model showcased slight improvement, achieving an accuracy of 0.97, recall of 0.98, and an F1 score of 0.97. This signifies the model's capability to capture more true positives while maintaining overall accuracy.

Similarly, at a random state of 15, the AraBERT model maintained consistent and balanced performance, securing accuracy, recall, and F1 score metrics all at 0.97.

A notable highlight occurred at a random state of 20, where the AraBERT model achieved flawless accuracy, recall, and F1 score metrics of 1.00, indicating impeccable and precise predictions.

Adjusting the random state to 42 resulted in high accuracy, recall, and F1 scores of 0.99, further emphasizing the model's accuracy and reliability in its predictions.

In summary, the AraBERT model consistently outperformed across all random states, with accuracy, recall, and F1 scores ranging from 0.96 to 1.00. These results underscore the model's robustness and efficacy in identifying patterns and making accurate predictions. The minimal variance in outcomes suggests that changes in the random state, affecting data selection for training and testing, have a negligible impact on AraBERT's performance. This consistency aligns with expectations, given the proven high performance of BERT and its variations in the existing literature.

Across all random states, the average accuracy stands at 0.978, the average recall at 0.98, and the average F1 score at 0.978, reaffirming the overall strong and reliable performance of the AraBERT model in Arabic text classification tasks.

Results Comparisons

In this study, a comprehensive comparison was conducted between AraBERT and several ML algorithms, namely KNN optimization with no stemmer, random forest with ISRI stemmer, and KNN with Arabic light stemmer. The evaluation metrics, including precision, recall, and F1 score, were utilized to assess their performance, and the results, as illustrated in Table 7, demonstrated that random forest with ISRI stemmer achieved the highest scores across all metrics, notably achieving a remarkable F1 score of 0.99.

Further analysis involved comparing the best-performing ML methods with AraBERT. Notably, random forest with ISRI stemmer outperformed AraBERT, aligning with expectations given their documented high performance in the literature. Additionally, it was observed that the integration of the ISRI stemmer further enhanced the efficiency of random forest. It's crucial to note that AraBERT, not utilizing stemming, and still evolving for the Arabic language, exhibited competitive performance. Moreover, KNN's performance improved with the application of the Arabic light stemmer, emphasizing the effectiveness of ML and DL models in classifying ambiguous requirements.

A separate set of experiments explored the impact of requirement skewness on the model's performance, utilizing the best-performing model, random forest with ISRI stemmer. In cases of long requirements with ambiguous and non-ambiguous parts, the model's performance was sentence-dependent, indicating that the dominance of either the ambiguous or non-ambiguous part is influenced by the frequency and weights of their respective words. Regarding requirements with spelling errors, the model showcased its ability to ignore errors and focus on correctly spelled words, aligning with its objective of determining ambiguity.

Table 8 presents the results of a statistical analysis using paired t-test, focusing on F1 scores. Interestingly, despite different F1 values, AraBERT and RF with ISRI showed no significant difference, highlighting the superiority of these two models in classifying ambiguous requirements.

In summary, the study provides a comprehensive comparison between AraBERT and ML algorithms, showcasing the effectiveness of Random Forest with ISRI stemmer. The findings

Table 7. Comparison between AraBERT and the best result For ML algorithms (no stemmer, ISRI stemmer, Arabic light stemmer)

Evaluation Metrics	AraBERT Model	KNN Optimization With No Stemmer Applied	Random Forest With ISRI Stemmer	KNN With Arabic Light Stemmer
Precision	0.978	0.96	0.99	0.97
Recall	0.978	0.96	0.99	0.97
F1 Score	0.978	0.96	0.99	0.97

Table 8. Comparison between various classifiers using paired t-test

Evaluation Metrics	AraBERT Model	KNN Optimization With No Stemmer Applied	Random Forest With ISRI Stemmer	KNN With Arabic Light Stemmer
AraBERT	-	>	=	=
Knn (no stemmer)	<	-	<	<
Random forest with ISRI stemmer	=	>	-	>
KNN with Arabic light stemmer	<	>	<	-

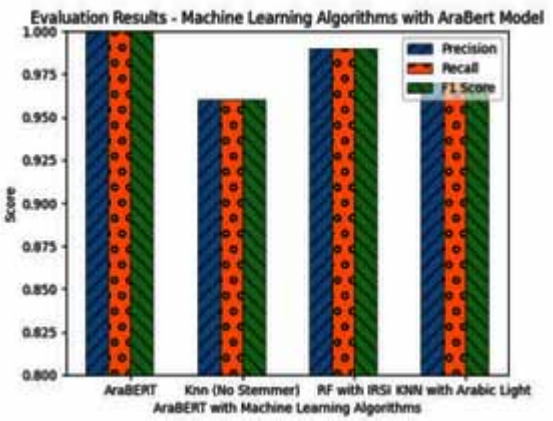
underscore the potential of both DL and traditional ML approaches in handling Arabic text classification tasks, offering insights into their respective strengths and highlighting areas for further exploration and refinement.

Figure 6 provides a comprehensive comparison between AraBERT and the best results achieved by ML algorithms under different stemming conditions, including no stemmer, ISRI stemmer, and Arabic light stemmer.

Limitations

This work, while making notable contributions, is constrained by several limitations. The foremost challenge lies in the scarcity of datasets, prompting the creation of a dedicated dataset for Arabic

Figure 6. Comparison between AraBERT and the best result on ML algorithms (no stemmer, ISRI stemmer, Arabic light stemmer)



requirements. However, further enrichment could be achieved by incorporating datasets from diverse domains and expanding the dataset size. Additionally, the limited landscape of existing studies in this domain hinders meaningful comparisons and optimizations. Future research efforts in the same domain would contribute to a more comprehensive understanding. Moreover, despite leveraging AraBERT, the deep-based method, it is acknowledged that the classifier is not fully developed for the Arabic language. Continuous improvements in AraBERT or the exploration of alternative DL approaches could lead to enhanced results, addressing this limitation and advancing the capabilities of Arabic language classifiers. Overall, these limitations highlight the imperative for ongoing efforts in dataset diversity, comparative studies, and the continuous development of DL models tailored for Arabic language processing.

CONCLUSION AND FUTURE WORK

Conclusion

In this study, both the AraBERT model and various traditional ML algorithms were evaluated for their performance in Arabic text classification. The results showcase the efficacy of ML techniques, including decision trees, KNN, SVMs, random forest, logistic regression, and XGBoost, in achieving commendable accuracy, precision, recall, and F1 scores. These findings affirm the suitability of these ML algorithms for pattern recognition and accurate predictions in the domain of Arabic language processing applications.

Notably, the AraBERT model, a customized pre-trained language model tailored for Arabic text, emerged as a standout performer. It consistently outperformed competitors across key metrics, including accuracy, precision, recall, and F1 scores. This finding underscores the significant enhancement that DL and pre-trained models, such as AraBERT, can bring to the capabilities of Arabic text classification systems.

Among the ML algorithms, random forest and KNN demonstrated top-notch performances. KNN, in particular, showcased resilience and generalizability by outperforming all stemmers, achieving an accuracy of 0.96 without a stemmer, 0.97 with ISRI stemmer, and 0.97 with Arabic light stemmer. This result highlights the robust nature of the KNN algorithm, especially in identifying data points on the basis of their proximity to nearby samples. Similarly, random forest exhibited strong performance, with accuracies of 0.95 without a stemmer, 0.99 with ISRI stemmer, and 0.97 with Arabic light stemmer.

The AraBERT model consistently outperformed various random states, illustrating its effectiveness in accurately identifying data patterns and making valid predictions. Across all random states, the AraBERT model exhibited high levels of accuracy, recall, and F1 scores, ranging from 0.973 to 1.00. This robust and consistent performance further reinforces the reliability and effectiveness of the AraBERT model in Arabic text classification tasks.

In summary, the study demonstrates the success of traditional ML algorithms and highlights the remarkable performance of the AraBERT model in Arabic text classification. The findings underscore the potential for both conventional and DL approaches to contribute significantly to the advancement of Arabic language processing applications.

Future Work

The current study offers valuable insights into ML algorithms and the AraBERT model in the context of Arabic text classification. However, to further enhance and expand upon these findings, several promising avenues for future research can be explored.

First, because of the existing scarcity of Arabic datasets, future studies should prioritize the collection of diverse and representative datasets spanning different domains and dialects. This approach will enable researchers to assess the generalizability and robustness of ML algorithms and the AraBERT model across various linguistic variations and real-world settings. Additionally, there

is a need for concerted efforts to construct benchmark datasets specifically tailored for Arabic text categorization. These specialized datasets would facilitate fair comparisons, fostering advancements in the field.

Second, while the current research relies predominantly on standard evaluation criteria such as accuracy, precision, recall, and F1 scores, future investigations should consider incorporating additional measures. Metrics like information gain or receiver operating characteristic (ROC) curves could provide a more comprehensive assessment of the performance of ML algorithms and the AraBERT model. Exploring the interpretability of these models is another avenue for future research, shedding light on the decision making processes and revealing insights into the characteristics and patterns that these models rely on for categorization.

Third, an intriguing area for future exploration involves evaluating the transfer learning capabilities of the AraBERT model. Researchers could leverage the pre-trained information embedded in the model and adapt it to specific domains by fine-tuning the AraBERT model for particular Arabic text categorization tasks or employing transfer learning approaches. This line of inquiry has the potential to uncover the adaptability and versatility of the AraBERT model in diverse applications.

In summary, future research endeavors in Arabic text classification should focus on expanding and diversifying datasets, incorporating additional evaluation metrics, exploring model interpretability, and investigating the transfer learning capabilities of the AraBERT model. These efforts will contribute to a more nuanced understanding of the model's performance and foster advancements in the field of Arabic NLP.

FUNDING

This research received no specific grant from any funding agency in the public, commercial, or not-for-profit sectors. Funding for this research was covered by the author(s) of the article.

REFERENCES

Abuhamdah, A., Alzaqebah, M., Jawarneh, S., Althunibat, A., & Banikhalaf, M. (2021). Moth optimization algorithm with local search for the permutation flow shop scheduling problem. *International Journal of Computer Applications in Technology*, *65*(3), 189–208. doi:10.1504/IJCAT.2021.116008

Adaileh, A. (2020). Virtual digital currency as a method for funding terrorism. *Al-Zaytoonah University of Jordan Journal for Legal Studies*, *1*(1), 32–56.

Aftan, S., & Shah, H. (2023). Using the AraBERT model for customer satisfaction classification of telecom sectors in Saudi Arabia. *Brain Sciences*, *13*(1), 147. doi:10.3390/brainsci13010147 PMID:36672129

Alrawashdeh, T., ElQirem, F., Althunibat, A., & Alsoub, R. (2021). A prioritization approach for regression test cases based on a revised genetic algorithm. *Information Technology and Control*, *50*(3), 443–457. doi:10.5755/j01.itc.50.3.27662

Alrwele, N. (2023). Efficacy of artificial intelligence in improving EFL phonemic awareness among sixth grade female students. *Journal of Education/Al Mejlh Altrbwyh, 148*(37), 117–145. 10.34120/0085-037-148-005

Alsawareah, B., Althunibat, A., & Hawashin, B. (2023, August 9–10). *Classification of Arabic software requirements using machine learning techniques*. *2023 International Conference on Information Technology (ICIT)*, Amman, Jordan. doi:10.1109/ICIT58056.2023.10225789

Alshehadeh, A. R., & Al-Khawaja, H. A. (2022). Financial Technology as a Basis for Financial Inclusion and its Impact on Profitability: Evidence from Commercial Banks. *International Journal of Advances in Soft Computing and its Applications*, *14*(2).

Alsoub, R. K., Alrawashdeh, T. A., & Althunibat, A. (2018). User acceptance criteria for enterprise resource planning software systems. *International Journal of Innovative Computing, Information, & Control*, *14*(1), 297–307.

Althunibat, A., Abdallah, M., Almaiah, M. A., Alabwaini, N., & Alrawashdeh, T. A. (2022). An acceptance model of using mobile-government services (AMGS). *CMES-Computer Modeling in Engineering & Sciences, 131*(2).

Althunibat, F. (2023). Relativity of the vantage point in the Quranic discourse: The sphericity of the earth and its rotation as a case study. *Arab Journal for the Humanities*, *161*(41), 45–70. doi:10.34120/0117-041-161-002

AlZu'bi, S., & Jararweh, Y. (2020, April 20–23). *Data fusion in autonomous vehicles research, literature tracing from imaginary idea to smart surrounding community* [Conference presentation]. 2020 Fifth International Conference on Fog and Mobile Edge Computing (FMEC), Paris, France. doi:10.1109/FMEC49853.2020.9144916

Amro, R., Althunibat, A., & Hawashin, B. (2023, August 9–10). *Arabic non-functional requirements extraction using machine learning* [Conference presentation]. 2023 International Conference on Information Technology (ICIT), Amman, Jordan. doi:10.1109/ICIT58056.2023.10225951

Berkson, J. (1944). Application of the logistic function to bio-assay. *Journal of the American Statistical Association*, *39*(227), 357–365. doi:10.1080/01621459.1944.10500699

Boser, B. E., Guyon, I. M., & Vapnik, V. N. (1992). A training algorithm for optimal margin classifiers. In *Proceedings of the fifth annual workshop on Computational learning theory – COLT '92* (pp. 144–152). Association for Computing Machinery. doi:10.1145/130385.130401

Das, A., Mishra, S., & Gopalan, S. (2020). Predicting CoVID-19 community mortality risk using machine learning and development of an online prognostic tool. *Peerj, 8*, e10083. 10.7717/peerj.10083

Dhaliwal, S. S., Nahid, A.-A., & Abbas, R. (2018). Effective intrusion detection system using XGBoost. *Information (Basel)*, *9*(7), 149. doi:10.3390/info9070149

Elazhary, H. (2016). Avoiding ambiguities in Arabic software user requirements. *International Journal of Software Engineering and Its Applications*, *10*(6), 141–160. doi:10.14257/ijseia.2016.10.6.12

Elsaid, A. H., Salem, R. K., & Abdul-kader, H. M. (2015). *Automatic framework for the requirement analysis phase* [Conference presentation]. 2015 Tenth International Conference on Computer Engineering & Systems (ICCES), Cairo, Egypt. doi:10.1109/ICCES.2015.7393045

Ezzini, S., Abualhaija, S., Arora, C., Sabetzadeh, M., & Briand, L. (2021). Using domain-specific corpora for improved handling of ambiguity in requirements. In *ACM 43rd International Conference on Software Engineering (ICSE)* (pp. 1485–1497). IEEE. doi:10.1109/ICSE43902.2021.00133

Fix, E., & Hodges, J. L. (1951). *Discriminatory analysis. nonparametric discrimination: Consistency properties*. USAF School of Aviation Medicine. https://www.jstor.org/stable/1403797

Ford, J., & VanderPlas, J. (2016). Cluster-lensing: A python package for galaxy clusters and miscentering. *The Astronomical Journal*, *152*(6), 228. doi:10.3847/1538-3881/152/6/228

Forman, G. (2003). An extensive empirical study of feature selection metrics for text classification. *Journal of Machine Learning Research*, *3*, 1289–1305. https://www.jmlr.org/papers/volume3/forman03a/forman03a_full.pdf

Gill, K. D., Raza, A., Zaidi, A. M., & Kiani, M. M. (2014, May 4–7). *Semi-automation for ambiguity resolution in Open Source Software requirements* [Conference presentation]. 2014 IEEE 27th Canadian Conference on Electrical and Computer Engineering (CCECE). Toronto, Canada. doi:10.1109/CCECE.2014.6900955

Ho, T. K. (1995). Random decision forests. In *Proceedings of the 3rd International Conference on Document Analysis and Recognition* (pp. 278–282). IEEE. doi:10.1109/ICDAR.1995.598994

Hunt, E. B., Marin, J., & Stone, P. J. (1966). *Experiments in induction*. Academic Press.

IEEE Standards Coordinating Committee. (1990). IEEE standard glossary of software engineering terminology (IEEE Std 610.12-1990). Los Alamitos. CA: IEEE Computer Society.

Jaber, T., Abdallah, M., & Al-thunibat, A. (2020). A proposed code inspection model using program slicing technique. In *2020 IEEE 5th International Conference on Computing Communication and Automation (ICCCA)* (pp. 275-279). IEEE. doi:10.1109/ICCCA49541.2020.9250784

Jebril, I., Almaslmani, R., Jarah, B., Mugableh, M., & Zaqeeba, N. (2023). The impact of strategic intelligence and asset management on enhancing competitive advantage: The mediating role of cybersecurity. *Uncertain Supply Chain Management*, *11*(3), 1041–1046. doi:10.5267/j.uscm.2023.4.018

Kasabera, Y., Al-Alzyadat, W., Alhroob, A., Al Showarah, S., & Thunibat, A. (2020). An automated approach to validate requirements specification. *Compusoft*, *9*(2), 3578–3585.

Ko, Y., Park, S., Seo, J., & Choi, S. (2007). Using classification techniques for informal requirements in the requirements analysis-supporting system. *Information and Software Technology*, *49*(11-12), 1128–1140. doi:10.1016/j.infsof.2006.11.007

Kotsiantis, S. (2011). Feature selection for machine learning classification problems: A recent overview. *Artificial Intelligence Review*, *42*(1), 157–176. doi:10.1007/s10462-011-9230-1

Kurtanović, Z., & Maalej, W. (2017, September 4–8). *Automatically classifying functional and non-functional requirements using supervised machine learning* [Conference presentation]. 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal. https://doi.org/ doi:10.1109/RE.2017.82

Larkey, L. S., Ballesteros, L., & Connell, M. E. (2002). Improving stemming for Arabic information retrieval: light stemming and co-occurrence analysis. In *Proceedings of the 25th annual international ACM SIGIR conference on Research and development in information retrieval* (pp. 275–282). Association for Computing Machinery. doi:10.1145/564376.564425

Leskovic, J., Rajaraman, A., & Ullman, J. D. (2011). *Mining of Massive Datasets*. Cambridge University Press., doi:10.1017/CBO9781139058452.002

Lima, M., Valle, V., Costa, E., Lira, F., & Gadelha, B. (2019, September). Software engineering repositories: expanding the promise database. In *Proceedings of the XXXIII Brazilian Symposium on Software Engineering,* Salvador, Brazil*,* (pp. 427-436). doi:10.1145/3350768.3350776

Lu, M., & Liang, P. (2017, June 15–16). *Automatic classification of non-functional requirements from augmented app user reviews* [Conference presentation]. 21st International Conference on Evaluation and Assessment in Software Engineering, Karlskrona, Sweden. https://doi.org/ doi:10.1145/3084226.3084241

Navarro-Almanza, R., Juarez-Ramirez, R., & Licea, G. (2017, October 25–27). *Towards supporting software engineering using deep learning: A case of software requirements classification* [Conference presentation]. 2017 5th International Conference in Software Engineering Research and Innovation (CONISOFT), Merida, Mexico.

Nigam, A., Arya, N., Nigam, B., & Jain, D. (2012). Tool for automatic discovery of ambiguity in requirements. *International Journal of Computer Science Issues*, *9*(5), 350.

Petrović, Đ., & Stanković, M. (2019). The influence of text preprocessing methods and tools on calculating text similarity. *Facta Universitatis, Series. Mathematics and Informatics*, *34*(5), 973–994. doi:10.22190/FUMI1905973D

Qaisi, H. Al, Quba, G. Y., Althunibat, A., Abdallah, A., & Alzu'bi, S. (2021, July 14–15). *An intelligent prototype for requirements validation process using machine learning algorithms* [Conference presentation]. 2021 International Conference on Information Technology (ICIT), Amman, Jordan. doi:10.1109/ICIT52682.2021.9491758

Quba, G. Y., Al Qaisi, H., Althunibat, A., & AlZu'bi, S. (2021, July 14–15). *Software requirements classification using machine learning algorithms* [Conference presentation]. 2021 International Conference on Information Technology (ICIT) Amman, Jordan. doi:10.1109/ICIT52682.2021.9491688

Rokach, L., & Maimon, O. (2005). Decision trees. In *Data mining and knowledge discovery handbook* (pp. 165-192). https://link.springer.com/chapter/10.1007/0-387-25465-X_9

Sarkar, D. (2016). *Text analytics with python* (Vol. 2). Springer. doi:10.1007/978-1-4842-2388-8

Taghva, K., Elkhoury, R., & Coombs, J. (2005). Arabic stemming without a root dictionary. In *International Conference on Information Technology: Coding and Computing (ITCC'05)*-Volume II (pp. 152-157). IEEE. doi:10.1109/ITCC.2005.90

Thunibat, A. (2011). Mobile government user requirements model. *Journal of E-Governance*, *34*(2), 104–111. doi:10.3233/GOV-2011-0260

Wasmi, H., Al-Rifaee, M., Thunibat, A., & Al-Mahadeen, B. (2021). Comparison between proposed convolutional neural network and KNN for finger vein and palm print. In *2021 International Conference on Information Technology (ICIT)* (pp. 946-951). IEEE. doi:10.1109/ICIT52682.2021.9491737

Yang, H., Willis, A., Roeck, A., & Nuseibeh, B. (2010). Automatic detection of nocuous coordination ambiguities in natural language requirements. In *Proceedings of the 25th IEEE/ACM International Conference on Automated Software Engineering* (pp. 53–62). doi:10.1145/1858996.1859007