Evaluating the Impact of Dataset Size on the Accuracy of Arabic Software Requirements Classification Using AraBERT

2025 12th International Conference on Information Technology (ICIT) | 979-8-3315-0894-4/25/$31.00 ©2025 IEEE | DOI: 10.1109/ICIT64950.2025.11049139

Reem Amro
Al-Zaytoonah University of Jordan Amman, Jordan
reemamro844@gmail.com

Ahmad Althunibat
Al-Zaytoonah University of Jordan Amman, Jordan
a.thunibat@zuj.edu.jo

Bilal Hawashin
Al-Zaytoonah University of Jordan Amman, Jordan
b.hawashin@zuj.edu.jo

***Abstract*— Requirement's engineering is crucial in software development, underpinning all the later stages, and heavily impacting whether a project succeeds or fails. A standard software requirements document includes functional and non-functional requirements — both are necessary. But, to classify and represent these requirements in natural language takes much human effort. It's laborious and takes a lot of time to manually classify; expensive and error-prone. Misclassification may cause misunderstandings or ambiguities and consequent incompleteness of the product not meeting customer expectations. Though the classification of software requirements has been studied extensively in English, Arabic still lacks such research. Moreover, Arabic datasets specific to this field are not publicly available: in our prior work, we used some data to classify Arabic software requirements and get results. This paper presents an extension of our work that investigates how dataset size affects classification performance. We propose a method to gather Arabic software requirements datasets and use deep learning techniques for automatic classification into functional and non-functional requirements. The study is intended to assist software developers in saving time, cost, and manual classification effort — thus making requirements engineering more efficient. It also creates a new path for researchers in this field.**

Keywords— Requirements Classification, Functional Specifications, Non-Functional Specifications, AraBERT model.

I. Introduction

Software engineering is a process that consists of phases like requirement analysis, design and development, testing, and maintenance. Following this methodology helps to create software that is both reliable and user-friendly, guaranteeing quality as well as dependable maintenance. Among these phases, the requirements stage is most likely to make or break the project. The objective of this phase is to ensure that the software system meets the expectations set by the stakeholders while respecting the boundaries placed on it. Clearly stated functional and non-functional requirements must be articulated along with those that restrict design and development activities[1].

As the first phrase suggests, in the requirements engineering process, the classification of requirements aims to put them in order as far as their properties are concerned[2]. In general, requirements are grouped into two primary classes: Functional Requirements (FRs) and Non-Functional Requirements (NFRs). FRs outline the system's services, its expected behavior during certain actions or events, and how the system's input responses should be processed. Sometimes, those requirements may specify some of the functions that the system should not perform. On the other hand, NFRs describe other but still important, system features like performance, security, and usability[3]. They also specify the bounds within which the system's services or functionalities may be performed, such as time limits, standard requirements, and other procedural constraints. Unlike FRs, which are directed to particular functions of the system, as the title suggests, NFRs are related to the whole system. [4].

Software requirement classification serves many purposes in software engineering. The primary benefits of this classification are improved traceability, increased efficiency in project management, better risk assessment, higher reputation validation processes, and strengthened quality assurance. These requirements can be easily tracked throughout the software development lifecycle, so they are orderly, which leads to improved software quality alongside the satisfaction of stakeholder expectations[5].

The breakdown of functional and non-functional requirements manually consumes a lot of time and necessitates intensive effort from software engineers. It is cost and time-intensive, and prone to errors. In the user requirements analysis stage, more problems arise because misclassifications can result in a lack of clarity or confusion which results in software that does not meet customer requirements [6].

While there have been several studies on English text software requirement classification, research in Arabic text software requirement classification is still very limited. As far as our knowledge goes, the articles that address Arabic requirement classification are [8] [9]. They specifically proposed a semi-automated classification approach. Arabic is more challenging to classify—thus increasing the ambiguity—because of its complexity in terms of syntactic, and semantic structuring compared to other languages. It is also marked by the very

limited resource and public dataset availability, which slows down the development of this research of course.

This survey is based on the computing performance levels of AraBERT under different sizes of the same dataset. To be more specific, this research harnesses AraBERT capacities under different sizes of the same dataset to identify the real capability variations of data volumes to classify Arabic software requirements into functional requirements (FRs) and non-functional requirements (NFRs). This paper addresses the problems associated with manual classification that are labor-intensive and usually lead to varying results, which could sometimes be misunderstood or lead to ambiguities during the software development process.

This work focuses on Arabic language-specific Software Requirements, thus addressing challenges due to its syntactic and semantic complexity in structure and offering unique challenges. In consideration of the constraints toward this end, AraBERT is utilized to facilitate the classification process- an alternative that saves time and reduces the outcome's margin of error. Moreover, due to the unavailability of public Arabic datasets in the field, we formed an Arabic dataset, the specificity of our research.

We use AraBERT with datasets of varying sizes to investigate how training data volume affects classification accuracy. The survey provides essential information on current field trends and shows why dataset size matters for classification improvements while laying the groundwork for subsequent research in this area.

The primary contributions of this paper encompass:

- This study examines how changing the amount of data affects the classification performance of Arabic software requirements through AraBERT.

The remainder of this paper is organized in the following manner: Section 2 summarizes relevant scholarly works while Section 3 describes our approach before Section 4 evaluates experimental outcomes and Section 5 offers future research recommendations.

II. Literature Review

The identification of non-functional requirements (NFRs) plays a vital role in the analysis of textual requirements within software engineering. Multiple investigations have explored diverse techniques to classify requirements. This section evaluates the different methods while detailing both their advantages and disadvantages.

Hmeidi et al. Hmeidi et al. [10] analyzed Arabic text categorization by using a dataset that consists of 2,700 Arabic articles from a variety of categories. The study implemented five established text classification approaches and applied preprocessing techniques like stemming and text cleaning. The Support Vector Machine (SVM) classifier stood out as the top-performing classification method demonstrating its effectiveness in processing Arabic text.

Amro et al. The study by Amro et al. [6] developed a method to categorize Arabic software requirements into two groups which are functional and non-functional requirements. A dataset in Arabic was created by them to fulfill the particular requirements of their study. The researchers conducted a study that investigates how machine learning algorithms are used to automate the classification of user requirements written in Arabic. The researchers evaluated model performance by measuring precision, recall and F-measure metrics.

Dave et al. [11] explored different data models using machine learning methods like Support Vector Machine (SVM), Stochastic Gradient Descent (SGD), and Random Forest (RF) along with the TF-IDF natural language processing technique. Their study found that SVM combined with TF-IDF produced the best results in identifying Functional Requirements (FRs), achieving an F1 score of 0.88. These models were used to classify FRs, Non-Functional Requirements (NFRs), and sub-categories of NFRs within Software Requirements Specification (SRS) documents. The SVM with TF-IDF provided the most accurate classification results for NFR categories such as Availability, Look & Feel, Maintainability, Operational, and Scalability, while SGD with TF-IDF proved to be the most effective for Security, Legal, and Usability requirements. Additionally, SGD achieved the highest accuracy in distinguishing NFRs, recording an F1 score of 0.92.

Shah et al. [12] introduced an automated classification technique aimed at distinguishing between Functional Requirements (FRs) and Non-Functional Requirements (NFRs) derived from raw software requirements. This methodology utilized machine learning algorithms such as Naïve Bayes, Support Vector Machine (SVM), and Recurrent Neural Network (RNN), in conjunction with TF-IDF for natural language processing. The models demonstrated accuracy rates of 91% for Naïve Bayes, 90% for SVM, and 92% for RNN. This approach effectively tackles the issue of misclassification in software requirements, which can lead to ambiguity, errors, and failures, thereby requiring significant time and resources for rectification.

Casamayor et al. [13] proposed a semi-supervised method for text categorization designed to automatically extract Non-Functional Requirements (NFRs) from textual data. Unlike traditional supervised techniques that rely on large pre-labeled datasets for accurate classification, this approach aimed to reduce manual intervention by using a learning process that required fewer labeled examples. The research successfully applied this method to automate the identification and classification of requirements, resulting in a significant reduction in manual effort. When evaluated using the same document collection criteria as supervised learning techniques, the semi-supervised method showed better performance, achieving accuracy rates above 70%.

Alnemer et al. [14] introduced a sentiment classification method tailored for the Arabic language, which incorporates emoticon extraction, morphological analysis, and the management of negation. They evaluated the effectiveness of a Convolutional Neural Network (CNN) against traditional classifiers such as Naïve Bayes, Decision Trees, Support Vector Machines (SVM), and Neural Networks in categorizing sentiments into positive/negative and positive/negative/neutral groups. The findings revealed that SVM attained the highest accuracy among the supervised classifiers, reaching 97%, while CNN marginally surpassed SVM with an accuracy of 98%.

Saeed et al. [15] proposed a supervised learning method for classifying sentiment in Arabic reviews. This method used two different linear transformation techniques: supervised Latent Dirichlet Allocation (LDA) and unsupervised Principal Component Analysis (PCA). To improve the reliability of the classifier before training, a spam detection system was added. The effectiveness of this approach was tested using five Arabic opinion text datasets, where the lengths of the reviews varied from 1.6K to 94K. The evaluation involved classification tasks with two categories (positive/negative sentiments) as well as three categories (positive/negative/neutral sentiments). In the two-class classification task, this method achieved impressive accuracy rates ranging from 95.5% to 99.8%. On the other hand, the three-class classification task produced accuracy scores between 92% and 97.3%.

Al-Tamimi et al. [16] demonstrated the effectiveness of enhancing AI agents' text classification capabilities for Arabic news articles using an active learning approach. Their results showed that active learning could achieve the desired classification accuracy while using significantly less data than traditional passive learning methods. The NADA dataset was used in three different versions for the experiments conducted. The study emphasized the importance of using augmentation strategies to increase the sample size for underrepresented categories within imbalanced datasets, thus improving the overall accuracy of the system. By reducing the resources and time needed to train AI agents, the active learning method encourages the broader use of machine learning to tackle more complex problems in the Arabic language.

Dahou et al. A novel technique for classifying Arabic sentences based on the integration of the Differential Evolution (DE) algorithm with Convolutional Neural Networks (CNN) is proposed in [17]. The DE algorithm is used to optimize the CNN architecture and its network parameters to make it possible to automatically find the most appropriate configuration. The authors highlighted five important CNN parameters, presenting the DE-CNN framework. DE-CNN was evaluated on five Arabic sentiment datasets and compared with state-of-the-art algorithms in terms of both accuracy and processing speed.

The work done by Alzanin and colleagues [18] developed a system to classify Arabic tweets into five different categories using linguistic and content analysis. Their study analyzed textual data in two different forms: stemmed text, which was transformed through TF-IDF, and text converted to Word2Vec embeddings. The team conducted experimentation with three classifiers, SVM, GNB, and RF, which were all tuned to optimal hyperparameter values. To test their approach, they provided labels for a set of about 35,600 Arabic tweets. The results showed that RF and SVM with RBF kernel, which employed stemming and TF-IDF, gave macro scores between 98.09% and 98.14%. In contrast, word embedding techniques using GNB were less effective. His methodology also surpassed the previously attained accuracy of 92.95% with a deep learning framework of RNN-GRU).

Rahimi et al. [19] developed a new data mining technique aimed specifically at non-functional requirements (NFRs) extraction. This approach seeks to extract specific quality characteristics such as usability, performance, and security that are pertinent to the software system included in the document. In order to aid the modeling of the extracted NFRs in relation to different quality concerns, a framework is created at level of a hierarchy. The proposed approach combines machine learning and data mining methodologies for the automated detection of numerous quality problems within the document. To improve the model, some of the relevant qualities are masked (unexposed) at some levels in the hierarchy so that these quality problems can be meaningfully grouped and categorized.

Al-Smadi et al. [20] developed a sentiment analysis approach aimed at Arabic hotel reviews using an aspect based methodology with two state of the art long short term memory (LSTM) neural networks. The former, Bi-LSTM-CRF, implemented sentiment analysis on aspect opinion target expressions (OTEs) using a Bidirectional LSTM and CRF classifier. The latter model, an aspect oriented LSTM, was designed to determine the sentiment polarity of the expressed aspects. To ensure higher recognition of sentiment polarity, attention expressions were considered on aspect OTEs. Both models were evaluated against a benchmark of Arabic hotel reviews and achieved a F1 score of 69.98% and 82.6% accuracy in the first and second models, respectively.

Ibrahim et al. [21] did assessment of classification of Arabic short texts with respect to effectiveness of three types of Naïve Bayes classifiers: multinomial Naive Bayes, complement Naive Bayes, and Gaussian Naive Bayes. The researchers built a corpus using standard web scraping procedures from various sources, and catalogued the files by their headings. In addition to removing punctuation, they also deleted stop words and converted spaces into vectors. Of the classifiers tried, Complement Naive Bayes achieved the best results with an accuracy rate of 0.84 at feature extraction in the testing phase. The findings of this research suggest that this method has a very high promise for numerous applications pertaining to classification of short texts.

These studies highlight the effectiveness of the machine learning, deep learning, and natural language processing (NLP) techniques automating the classification of software requirements into functional and non-functional. Although the described methods are promising, additional research is needed to improve the precision and effectiveness of the analysis.

From what we know, there seems to be a substantial gap in literature regarding the classification of Arabic software requirements. The existing research on this matter suffers from numerous shortcomings which have made it difficult to formulate all-encompassing approaches to the problem. To begin with, there is a handful of studies that analyze only Arabic requirements which fail to capture the peculiarities and intricacies of language's semantics and syntax. Such an absence, in turn, makes it difficult to develop classification systems that are specialized and effective, thus failing to accurately classify Arabic requirements.

III. Methodology

The prior section's literature review highlighted the need for further examination regarding the classification of Arabic

requirements. It also noted the absence of a system that can automatically divide software requirements stated in Arabic into functional and non-functional elements.

In this work, we extend our analysis by considering the impact of dataset size on classification accuracy. A method for collecting Arabic datasets relevant to software requirements is proposed to automatically classify them into functional and non-functional groups with the aid of the AraBERT model.

The methodology employed to achieve this goal is outlined below, Figure 1 providing a visual representation of the steps involved.

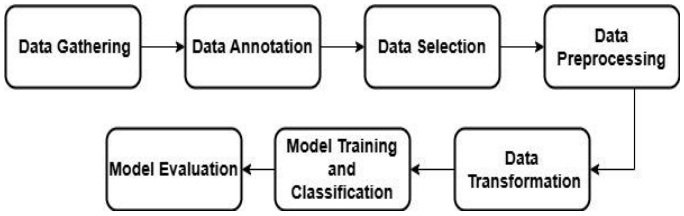

Figure 1: The proposed Methodology

A. Step 1: Data Gathering

Regarding the limited availability of Arabic datasets for software requirements classification in the public domain, we have put together an Arabic software requirements dataset gathered from a diversity of sources. This includes industry standards, academic papers, and repositories on the web where this dataset was used to train and validate our model.

B. Step 2: Data Annotation

Domain experts manually sorted the collected requirements into functional requirements and non-functional requirements with predefined basis. Their expertise was essential for distinguishing functional from non-functional parts of the software requirements.

C. Step 3: Data Selection

After annotation, some labeled data was selected further processing; data quality, diversity and balance of FRs to NFRs were considered in selection so as ensure effective training of the model.

D. Step 4: Data Preprocessing

We preprocessed the text data for AraBERT using multiple techniques:

- Tokenization: AraBERT's native tokenizer was employed to effectively deal with Arabic morphology.
- Stopword Removal: Common Arabic stopwords were removed so that significant words could be focused on.
- Text Normalization: It involved removing diacritics, punctuation, tanween, Hamza, and converting text to its root form for consistency.

E. Step 5: Data Transformation

As an alternative to conventional numerical feature scaling (e. g. G. min-max scaling) we transformed textual requirements into dense numerical vectors that capture semantic meaning using AraBERT embeddings.

F. Step 6: Model Training and Classification

During this stage, we used the pre-trained AraBERT model to classify Arabic software requirements into functional and non-functional categories. First, AraBERT was applied just like that (without fine tuning), and the results while generated did not meet our expectations. Thus, we fine-tuned AraBERT with our dataset in order to improve its performance — breaking it up into training and test sets allowed adaptation of the model to Arabic software requirements. We explored different random state values (32) to see how they affect model performance. The model was trained for five epochs so it could learn from this domain-specific dataset.

G. Step 7: model evaluation

We assessed our method's performance by calculating precision, recall and F1-score. The fine-tuned model significantly improved classification accuracy; indicating that training on a domain specific dataset allows the model to better identify functional and non-functional requirements.

Fine-tuning AraBERT helped us utilize the contextual awareness of Arabic and adapt it for software engineering. This phase proved that fine-tuning deep learning models on specific datasets is essential for accurate and dependable Arabic software requirements classification.

This work covers user requirements categorization in the software engineering domain, concentrated especially on the issues associated with manual classification of Functional Requirements (FRs) and Non-Functional Requirements (NFRs). The manual classification process typically is very time-consuming and riddled with variations, leading to misinterpretations or ambiguities about requirements which in turn adversely affect the development of software products meeting user needs. Another promising area for research is that of requirements classification in Arabic-its structural, syntactic, and semantic complexities make the task even more difficult. We propose a holistic approach, based on a deep learning algorithm, to account for these sources of inefficiency.

The previous work was based on the results obtained from an identified volume of data in classifying Arabic software requirements. We, therefore, present a study that further investigates the impact of dataset size on classification accuracy. This study is thus very important since it can satisfy the customers' needs to reduce the time, cost, and resources involved in manual classification and ensure that the requirements engineering phase can proceed smoothly. It can also be useful as a reliable reference in facilitating further works to the academicians for the study of different alternative approaches. Also, it enriches research on the field of study with new areas to explore and provides an Arabic dataset to be used freely by researchers in that area of study.

IV. Experiments

A. Data set

The research stated in this report performed experiments using software requirements datasets in Arabic language that were collected from different sources and evaluated by domain experts to ensure their quality. Three different sizes of datasets were used: 200, 150 and 100 requirements. Each dataset fulfilled an equal proportion of both functional and non-functional requirements. Each dataset was divided in two parts: training set and test set. Test set was used to evaluate the performance of AraBERT models so the impact of dataset size on classification accuracy could be analyzed.

B. Evaluation measurements

Recall, precision, and F1 score: these are three commonly used evaluation metrics which we used to assess the effectiveness of the proposed method [7].

Recall measures how well the system can identify all positive instances.

$$\boldsymbol{Recall} = \frac{\mathrm{TP}}{\mathrm{TP+FN}}$$

Precision measures how well a system can identify positive instances correctly.

$$\boldsymbol{Precision} = \frac{\mathrm{TP}}{\mathrm{TP+FP}}$$

The F1 score provides an overall performance evaluation of the system and is the harmonic mean of precision and recall.

$$\boldsymbol{F1\text{-}Score} = \frac{2 * \mathrm{Precision} * \mathrm{Recall}}{\mathrm{Precision} + \mathrm{Recall}}$$

Improved values of recall, precision, and F1 score led to enhanced performance of the algorithm.

C. Experimental settings

Hardware and software configurations were combined to create the experimental framework for this study. Google Colab which offers free GPU and TPU resources appropriate for deep learning tasks was used to conduct the experiments. A number of related libraries and Python were used in the implementation. A Dell laptop with an Intel Core i7 processor and 16 GB of RAM served as the hardware used to carry out the experiments.

D. Experimental results

The AraBERT Model showcased remarkable outcomes in deep learning related classification, demonstrating an F1 score of 0.97 while being trained using the largest available dataset of 200 records. This accomplishment reiterates the need of large datasets for deep learning models as they serve as backbone for the model to learn and internalize the information's intricacy which leads to precise prediction. Nonetheless, there was a striking pattern that emerged as the dataset size was scaled down. The model's accuracy dropped noticeably as the dataset was reduced, especially with the smallest dataset of 100 records. This reveals an evident problem in deep learning models, which is the necessity of big data to perform well; without enough data, a model's ability to generalize and accurately predict is severely limited.

This problem becomes further exaggerated in the area of Software Requirements in Arabic due to the contours of the Arabic language in its syntax, morphology, and semantic pose several difficulties. For AraBERT and other deep learning models, the precision brought forth by the Arabic text requires a great quantity of Arabic High quality structured text. In regard to the lack of well refined huge scale Arabic datasets in this field, there is a need for more work to enhance dataset construction and model performance. To improve AraBERTs classification accuracy and tackle these issues it is imperative to train it on a larger corpus of Arabic words and a variety of software requirements. The detailed AraBERT results for different dataset sizes are shown in Table 1 which also highlights how dataset size affects classification performance.

Performance measures	Precision	Recall	F1-Score
200 records	0.97	0.97	0.97
150 records	0.87	0.85	0.86
100 records	0.81	0.79	0.78

Table 1. Results of AraBERT across different dataset sizes

V. Conclusion and Futurework

The study outlines a method for categorizing software requirements expressed in Arabic using the AraBERT model into functional and non-functional categories, as well as non-Arabic requirements. Also, we deepened the science's knowledge base by creating an Arabic requirements dataset. An analysis of the state of automated classification of user requirements written in Arabic and the application of machine and deep learning techniques was also conducted. As a part of this work, a report of the literature on the reviewed studies is presented to highlight other works done in this area.

Previously, we accomplished results with a defined volume of data when classifying Arabic software requirements. In this current work, we broaden our scope by analyzing how the size of the dataset affects the accuracy of classification. We try different sizes of the dataset in order to understand how changes in the amount of data impact the AraBERT model's performance in accurately classifying requirements.

This study is a remarkable first stride towards the achievement of automated systems that can intelligently categorize user needs described in the Arabic language. It highlights the use of machine learning techniques in this area and opens the door for more study from other scientists. However, there is much scope for further work, including the application of advanced natural language processing techniques such as semantic analysis and named entity recognition to improve the accuracy of classification. In addition, expanding the coverage of the dataset to include different types of

requirements and different domains would enhance the relevance of the solution developed.

REFERENCES

[1] A. Mahmoud and G. R. Williams, "Detecting, classifying, and tracing non-functional software requirements," *Requirements Engineering*, vol. 21, no. 3, pp. 357–381, May 2016.

[2] Z. Kurtanovic and W. Maalej, *Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning*. 2017.

[3] F. Khan, S. U. Jan, M. Tahir, S. Khan, and F. Ullah, "Survey: Dealing Non-Functional Requirements at Architecture Level," *VFAST Transactions on Software Engineering, vol*. 9, no. 2, p. 7, Apr. 2016.

[4] T. A. Alrawashdeh, F. A. El-Qirem, A. Althunibat, and R. Alsoub, "A Prioritization Approach for Regression Test Cases Based on a Revised Genetic Algorithm," *Information Technology and Control*, vol. 50, no. 3, pp. 443–457, Sep. 2021.

[5] Y. Al-Kasabera, W. Alzyadat, A. Alhroob, S. Al Showarah and A. Thunibat, "An automated approach to validate requirements specification", *Compusoft*, vol. 9, no. 2, pp. 3578-3585, 2020.

[6] Amro, Reem, Ahmad Althunibat, and Bilal Hawashin. "Arabic User Requirements Classification Using Machine Learning." *2023 International Conference on Information Technology (ICIT)*. IEEE, 2023..

[7] P. Talele and R. Phalnikar, *Classification and Prioritisation of Software Requirements using Machine Learning – A Systematic Review*. 2021.

[8] Althunibat, Ahmad, et al. "Automated Classification of User Requirements written in Arabic using machine learning algorithms." *Appl. Math* 17.6 (2023): 1155-1170.

[9] Amro, Reem, Ahmad Althunibat, and Bilal Hawashin. "Arabic Non-Functional Requirements Extraction Using Machine Learning." *2023 International Conference on Information Technology (ICIT)*. IEEE, 2023.

[10] 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, no. 1, pp. 114–124, Nov. 2014.

[11] D. Dave, V. Anu, and A. S. Varde, *Automating the Classification of Requirements Data*. 2021.

[12] P. S. Shah, I. Ullah, and M. Shoaib, "Automatic Classification of Raw Software Requirements using Machine Learning," Aug. 2021.

[13] A. Casamayor, D. Godoy, and M. Campo, "Identification of non-functional requirements in textual specifications: A semi-supervised learning approach," *Information & Software Technology*, vol. 52, no. 4, pp. 436–445, Apr. 2010.

[14] L. M. Alnemer, B. Alammouri, J. Alsakran, and O. E. Ariss, "Enhanced Classification of Sentiment Analysis of Arabic Reviews," in *Lecture notes on data engineering and communications technologies*, Springer International Publishing, 2019, pp. 210–220.

[15] R. M. K. Saeed, S. Rady, and T. F. Gharib, "Optimizing Sentiment Classification for Arabic Opinion Texts," *Cognitive Computation*, vol. 13, no. 1, pp. 164–178, Jan. 2021.

[16] A.-K. Al-Tamimi, E. Bani-Isaa, and A. Al-Alami, *Active Learning for Arabic Text Classification*. 2021.

[17] A. Dahou, T. Seppänen, J. Zhou, and S. Xiong, "Arabic Sentiment Classification Using Convolutional Neural Network and Differential Evolution Algorithm," *Computational Intelligence and Neuroscience*, vol. 2019, pp. 1–16, Feb. 2019.

[18] S. M. Alzanin, A. M. Azmi, and H. Aboalsamh, "Short text classification for Arabic social media tweets," *Journal of King Saud University - Computer and Information Sciences*, vol. 34, no. 9, pp. 6595–6604, Oct. 2022.

[19] M. Rahimi, M. Mirakhorli, and J. Cleland-Huang, *Automated extraction and visualization of quality concerns from requirements specifications*. 2014.

[20] M. Al-Smadi, B. Talafha, M. Al-Ayyoub, and Y. Jararweh, "Using long short-term memory deep neural networks for aspect-based sentiment analysis of Arabic reviews," *International Journal of Machine Learning and Cybernetics*, vol. 10, no. 8, pp. 2163–2175, Mar. 2018.

[21] M. A. Ibrahim, A. Alhakeem, and N. A. Fadhil, "Evaluation of Naïve Bayes Classification in Arabic Short Text Classification," *Al-Mustansiriyah Journal of Science*, vol. 32, no. 4, pp. 42–50, Nov. 2021.