*DOI: 10.15849/IJASCA.250330.15*

*Int. J. Advance Soft Compu. Appl, Vol. 17, No. 1, March 2025*
*Print ISSN: 2710-1274, Online ISSN: 2074-8523*
*Copyright © Al-Zaytoonah University of Jordan (ZUJ)*

# A Semi-Automated Approach for Classifying Non-Functional Arabic User Requirements using NLP Tools

**Nabil Arman[1] and Faisal Khamayseh[2] and Eman Awad[3]**
[1]Department of Computer Science and Information Technology
Palestine
E-mail: narman@ppu.edu
[2]Department of Computer Science and Information Technology
Palestine
E-mail: faisal@ppu.edu
[3]College of Graduate Studies
Palestine Polytechnic University
Palestine
E-mail: 196003@ppu.edu

**Abstract**

***Requirements engineering is a critical phase in the software development life cycle, encompassing both Functional Requirements (FR) and Non-Functional Requirements (NFR). NFR defines the quality attributes of the system, including performance, security, availability, look and feel, fault tolerance, legal and operational, essential for meeting user needs and imposing additional constraints on software quality. Prioritizing NFR from user requirements is challenging, requiring specialized skills and domain knowledge. Manual categorization is time-consuming and mentally taxing for developers, making automated or semi-automated classification of NFR from requirements documents valuable. This approach reduces manual effort and time in identifying specific NFR among numerous requirements. This paper introduces a novel semi-automated categorization approach for Arabic Non-functional user requirements using CAMeL Tools which is a natural language processing (NLP) tool. We propose a set of heuristics based on fundamental Arabic sentence constructions to extract information and categorize requirements into seven NFR classes. Tokens, PoS tags, and lemmas of parsed user requirements are generated using CAMeL tools. The closest class for each statement is determined by applying heuristic criteria to CAMeL outputs. The implementation of our approach using CAMeL Tools 1.3.1 and Python code in a Windows 10 environment demonstrates its practical applicability and efficiency in classifying Arabic Non-functional user requirements.***

**Keywords**: *Requirements Engineering, Functional Requirements, Non-Functional Requirements, natural language processing tool.*

***Received 17January 2025; Accepted 18 March 2025***

## 1 Introduction

In software development and automated software engineering, requirements play a crucial role in determining a project's success or failure [1],[2],[3],[16]. Requirement Engineering (RE) is an essential phase in the software development lifecycle, where Requirements Analysis is responsible for identifying user expectations for newly developed or modified software systems. This stage evaluates whether the specified requirements are complete and well-structured [4].

Requirements are generally divided into two main types: Functional Requirements (FR) and Non-Functional Requirements (NFR) [1]. FRs describe the expected input-output behavior of a system, outlining the essential functionalities it must perform. Conversely, NFRs—also referred to as quality attributes—define various qualitative aspects of the system, including performance, security, availability, fault tolerance, legal compliance, and operational characteristics [6], [7].

Despite their significance in software engineering, extracting NFRs remains a difficult and intricate task. Research suggests that most NFR extraction techniques are primarily supervised, requiring significant manual effort for model training. These methods are resource-intensive, challenging to scale, and depend heavily on pre-labeled datasets, which are often unavailable for low-resource languages like Arabic. The complexity of Arabic, with its rich syntax, multiple dialects, and cultural variations, further complicates manual classification. Existing NFR extraction techniques are not well-optimized for Arabic-language requirements, and the potential of machine learning in this domain remains largely unexplored. This research seeks to address these challenges by developing an advanced approach specifically tailored for Arabic NFR classification.

This research makes several key contributions to the areas of NFR extraction and Arabic language processing:

1. **Semi-Automated NFR Classification** – A novel semi-automated method is introduced for classifying NFRs in Arabic texts, utilizing Natural Language Processing (NLP) tools specifically designed for Arabic. This approach aims to reduce manual effort, enhance accuracy, and improve the scalability of extracting NFRs from Arabic documents [1].
2. **Cultural Sensitivity in NFR Identification** – The study emphasizes the importance of cultural awareness when classifying NFRs in Arabic, ensuring that the classification framework is contextually appropriate for Arabic-speaking users [2].
3. **Enhanced Accuracy and Efficiency** – By integrating CAMeL Tools and machine learning techniques, this research improves the efficiency and accuracy of NFR extraction, allowing for more effective processing of large datasets [3].
4. **Advancing Arabic NLP in Software Engineering** – This study contributes to the underexplored intersection of Arabic natural language processing and software engineering, specifically addressing challenges related to NFR classification in Arabic-language requirements [4].

To address these challenges, a semi-automated NFR classification framework optimized for Arabic is proposed, incorporating NLP techniques to reduce manual workload and enhance precision. The method leverages the CAMeL Tools suite to improve classification performance while ensuring linguistic and cultural appropriateness for Arabic-speaking users.

The research employs a structured approach that integrates machine learning and NLP techniques tailored for Arabic. The methodology consists of:

- **Data Collection** – Gathering a large annotated corpus of Arabic requirements [5].
- **Feature Extraction** – Applying methods like TF-IDF to extract relevant linguistic features [6].
- **Machine Learning Implementation** – Utilizing machine learning models to classify NFRs with higher efficiency and accuracy [7].

The remainder of the paper is structured as follows: Section 2 discusses related work on NFR extraction and machine learning methodologies. Section 3 provides the necessary background on the challenges of NFR classification. Section 4 outlines the proposed classification framework and its implementation using Arabic NLP tools. Section 5 presents the evaluation process, experimental findings, and their implications. Section 6 summarizes key contributions, and Section 7 explores future research directions, highlighting the broader impact on software engineering practices.

## 2 Related Work

This section examines research efforts aimed at automating the identification, classification, and analysis of Non-Functional Requirements (NFRs). Many of these studies utilize machine learning techniques, natural language processing (NLP), or hybrid methods to enhance the automation of NFR extraction from textual requirement documents.

In [3], an experiment was conducted to classify NFRs into multiple categories, including availability, security, usability, legal and licensing, maintainability, performance, scalability, and fault tolerance, based on the requirements of an IoT-focused healthcare system. The study applied machine learning approaches, using a combination of K-Nearest Neighbors (KNN) and rule-based classification, while leveraging Bag of Words (BoW) and Term Frequency-Inverse Document Frequency (TF-IDF) for feature extraction. However, its reliance on a small dataset of 104 requirements raises concerns regarding the generalizability of its findings, as such a limited sample size may hinder applicability to larger datasets. The issue of scalability also suggests the need for larger, more representative datasets to enhance classification accuracy.

Similarly, [4] introduced a structured method combining semantic and syntactic analysis with machine learning to classify NFRs in unrestricted text. This approach employed four NLP techniques, including Google word2vec and BERT-based models, to extract relevant features from requirement statements. While the study demonstrated advancements in classification methodology, the dataset used—comprising only 79 unrestricted reports—was relatively small. Additionally, the study did not address challenges related to domain-specific terminology or languages with complex syntactic structures, such as Arabic, limiting its adaptability across diverse linguistic and technical contexts.

A semi-supervised machine learning method was proposed in [5] to reduce dependence on extensive labeled datasets. This study utilized a Wikipedia data dump for model training while incorporating supervised elements. Preprocessing techniques, such as Part of Speech (POS) tagging and word augmentation, were used to enhance extraction efficiency. While the results were promising, the approach relied heavily on pre-existing general-purpose datasets, making it less effective for specialized requirement domains. Furthermore, the system's performance, measured by precision, recall, and F-measure, indicated room for optimization, particularly in cases where labeled data is scarce.

In [6], researchers evaluated various machine learning algorithms—including Naive Bayes, K-Nearest Neighbors, and Support Vector Machines—alongside feature selection techniques like BoW and TF-IDF for NFR classification into eleven categories. While the study provided a comprehensive performance analysis, it did not fully explore how domain-specific language variations affect classification accuracy, especially in languages with complex morphology or cultural influences. Despite the study's statistical rigor, the applicability of its methods to languages like Arabic remains uncertain.

An unsupervised approach to NFR classification was introduced in [8], eliminating the need for large training datasets. This method employed word semantic similarity algorithms to extract relevant content from both source code and requirements. However, unsupervised techniques often face accuracy challenges when applied to specialized or domain-specific texts, particularly in languages with intricate syntactic rules. The study did not provide an extensive evaluation of its scalability or its adaptability to under-resourced languages such as Arabic, which presents additional challenges due to dialectal variation and grammatical complexity.

In [9], a fuzzy similarity-based K-Nearest Neighbor (FSKNN) algorithm was used for NFR classification, achieving an accuracy of 41.4%, which improved to 43.7% when incorporating semantic parameters. While this increase in accuracy is noteworthy, it suggests that the fuzzy similarity-based approach may not fully capture the nuanced semantic variations present in requirements documents, particularly in languages with unique syntactic structures. The study did not consider the cultural and linguistic complexities that could affect NFR classification in underrepresented languages.

A classification system tailored for NFRs in Information Systems (IS) was proposed in [10], designed to address web-based and real-time system requirements. While this system structured NFRs into a hierarchical format, its dependence on similarity-based methods raised concerns regarding its effectiveness in identifying emerging or evolving NFR categories. The approach performed well in IS-related applications but lacked adaptability for broader domains with more diverse and complex requirements.

Despite advancements in NFR automation, several limitations persist across existing studies. First, many approaches rely on supervised or semi-supervised techniques that require extensive pre-labeled datasets, which are not readily available, particularly for languages with limited resources like Arabic. Second, most methods are designed for languages with relatively simple syntactic structures and do not address the complexities of Arabic, including its morphological richness and dialectal variations. Third, scalability remains a challenge, as many models perform well on small datasets but struggle with larger and more diverse data.

This paper aims to address these gaps by focusing on Arabic-language requirement documents, a domain largely overlooked in previous research. Unlike prior studies, our approach incorporates NLP tools specifically optimized for Arabic, offering a semi-automated classification method that reduces dependency on pre-labeled datasets. This enhances scalability and flexibility, making it more applicable to large and diverse datasets. Our system demonstrates high accuracy in classifying NFRs, contributing to a more inclusive approach to software engineering that better serves Arabic-speaking communities. By addressing both linguistic and scalability challenges, this research seeks to advance the field of NFR extraction and classification.

## 3 Background

Requirements Engineering (RE) is a basic subject in software engineering which is the driving force of the requirement engineering habits in software engineering on which the excellent structure of software systems will imbue [20]. RE abides by a simple definition. It is a method of developing software repeatedly by tracing user's anticipations and requirements totally and carefully in order to prove that the ultimate product is what the users unwish. It is also the most essential stage inside the whole software development life cycle and is responsible for linking the Conceptual design of software system and its physical formation [21]. RE starts with the identification of stakeholders. Requirements are then further explored, constrained, and expressed as unambiguously as possible in order to be accurately represented as a software requirements specification (SRS), which is the primary RE documentation essential for the software system and includes functional and Non-functional requirements [22].

### A. User Requirements Written in Arabic

Expressing software requirements in Arabic involves distinct challenges due to the language's complex morphology and syntax, cultural nuances, and regional dialects. The rich morphological structure of the Arabic language that introduces several word variations and, in many cases, meaning changes based on vocalization, introduces complexity in writing correct, clear, and unambiguous requirements [23]. The lack of vocalization in written text means that the same sentence can lead to several interpretations by the software developer, causing confusion regarding the functionality intended software. Additionally, the cultural aspects of Arabic; the fact that there are no direct equivalents to a number of technical terms in other languages plays a part in the challenge of translation and getting the terminologies right in software engineering. The presence of different dialects in the Arabic language also adds to the challenge as in one dialect you may be very clear in what you mean in terms of the requirement, but that requirement may be misunderstood in a different dialect, given rise to misunderstandings in the software of what should actually be done [23].

### B. Natural Language Processing Tools

At the core of our research lie Natural Language Processing (NLP) tools, which are the technological enablers to analyze and understand human language with the help of any computational agents [24]. NLP tools are a specialized field of computer science, artificial intelligence, and linguistics that enables computers to process and interpret human language as it is spoken and deciphered, whether that language is in the form of Arabic speech or any type of written text. NLP is a critical component of our research work, as it will be the key player in successfully characterizing the Non-functional needs of Arabic users. NLP Technologies can play a large role in bridging the complexity of human languages, in this case Arabic, and the complexity of the revised world of software development. They can help us overcome the problems that Arabic language texts consist of, which include dialect varieties, tangled morphological functions, and cultural diversions. These techniques have made it possible to understand the text, harvest the relative information, and finally classify the Non-functional requirements. The goal of this research is to increase both the effectiveness and accuracy of our classification system using NLP technology, where the underlying system should be language and culture independent and, most importantly, international, which can eventually help both NLP and software development disciplines [25].

B. CAMeL Tools

The technology developed by CAMeL Tools goes above and beyond basic corpus lexicography. Arguably, groundbreaking even for corpus linguistics, CAMeL Tools sets new standards in Arabic Natural Language Processing (NLP) stemming from a computational perspective. It is about Arabic's specificities in relation, among others, to its morphological system, its dialectal diversity, as well as its orthographic idiosyncrasy. CAMeL Tools doesn't only perform basic analysis on text sets but also advanced operations such as morphological analysis, part-of-speech tagging, named entity recognition, or sentiment analysis [26]. Each tool in the suite has been developed in response to a need in Arabic language processing, harnessing the latest machine learning and deep learning techniques. For example, the morphological analyzer is crucial for a derivational language like Arabic to identify word roots and patterns. The tools have been trained on extensive corpora for Modern Standard Arabic as well as different regional dialects, so as to be effective in different contexts. Unique to CAMEL Tools is how they are developed. They are built by leveraging the open-source software development paradigm to actively invite researchers and developers across the globe to involve, adapt, refine, and optimize them with respect to real-life applications and user feedback, which creates an ecosystem that evolves and creates the tools continuously based on feedback and user satisfaction with proven workable performance metrics. CAMeL Tools value exists in various platforms that were utilized to engage from educational software helping people to learn Arabic and/or reconcile Arabic grammar exceptional cases to big data analytics platforms performing analytics on Arabic social media sentiment. Its development is a milestone in the Arabic NLP, bridging the human languages and computational understanding. The tool robustness and adaptability make inclusiveness and representation in the digital landscape specifically for the Arabic NLP, which is demanding and nuanced but not included in Arabic [26].

## 4 Proposed Classification Approach

In our study we introduce a novel semi-supervised approach that is dedicated to classify NFRs in Arabic software documentation into seven main classes: Performance (PE), Security (SE), Availability (A), Look and Feel (LF), Fault Tolerance (FT), Legal (L), and Operational (O). Scalability class will be excluded in this study as it is overlapped with other classes. This section elucidates the approach for classifying user requirements into the different categories, leveraging the grammatical structure and keywords of Arabic sentences. Our methodology involves a thorough examination of various software graduation projects and Software Requirements Specifications (SRS) documents. From this analysis, we discerned distinctive attributes that enable the classification of different classes. These attributes form the basis for a set of heuristics in our approach. The process of analyzing Arabic sentences entails the utilization of CAMeL NLP tools, which facilitate parsing, tokenization, part-of-speech tagging, and sentence segmentation. We utilize an empirical methodology detailed in Fig 1 to classify Non-functional Arabic user requirements. We devised a set of heuristics specifically tailored to categorize user requirements into seven NFR categories by analyzing features extracted from user requirements, leveraging Arabic grammar, analyzing Parts of Speech (PoS) tags, and compiling relevant NFR keywords. The process begins with inputting a collection of unclassified user requirements in Arabic.

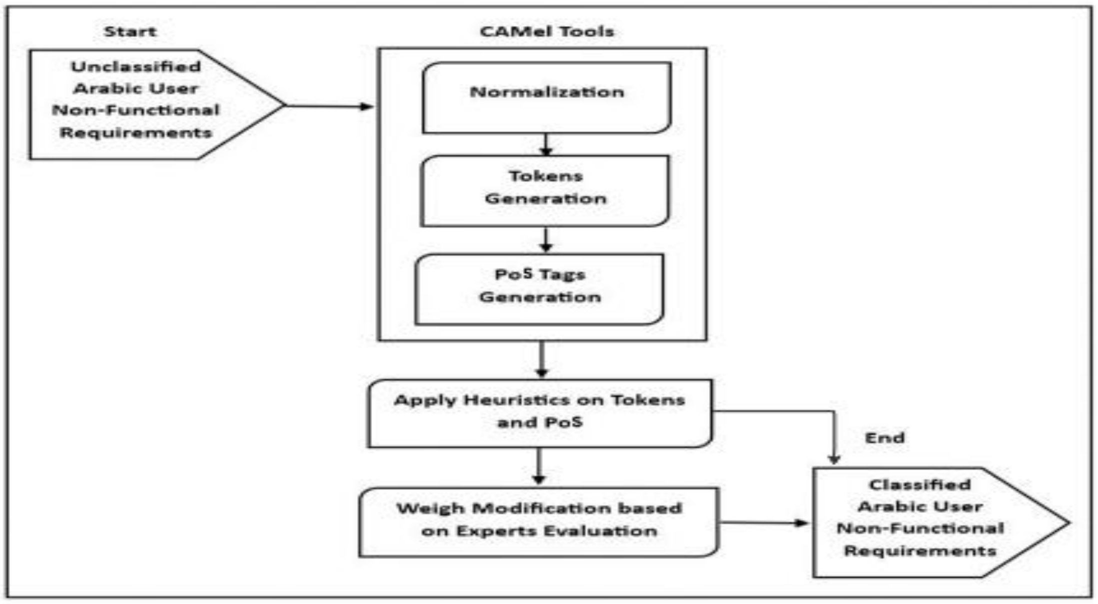
Start
Unclassified Arabic User Non-Functional Requirements
CAMel Tools
Normalization
Tokens Generation
PoS Tags Generation
Apply Heuristics on Tokens and PoS
Weigh Modification based on Experts Evaluation
End
Classified Arabic User Non-Functional Requirements

Fig1. Empirical methodology

Initially, all requirements are normalized using CAMeL tools before being processed further. Tokens for all statements are then generated using the CAMeL tokens generator, followed by the generation of PoS tags for all words in the given sentence. Subsequently, the proposed heuristics are applied utilizing the generated PoS and tokens. Each sentence's classification involves comparing the NFR score with other pertinent metrics such as confidence factors. Ultimately, the output of the approach is a collection of Non-functional Arabic user requirements.

A. The Proposed Heuristics

We developed a set of proposed heuristics organized in set of classes to handle Non-Functional User Requirements Linguistic Features as follows:

- Class 1: Performance (PE)

**H#1**: This Heuristic suggests the possibility of encountering common structural elements containing expressions such as "at an acceptable time" and "in the right time", namely "في وقت مقبول " and " في الوقت المناسب" This heuristic is identified through an examination of diverse instances:
Example: "يجب أن يكون الموظفون قادرين على إكمال مجموعة من المهام في الوقت المناسب"
Translation: "Employees must be able to complete a set of tasks in a timely manner".
CAMeL Tokens: ['يجب', 'أن', 'يكون', "الموظفون", 'قادرين', 'على', "إكمال", 'مجموعة', 'من', "المهام", 'في', "الوقت", "المناسب"]
CAMeL PoS :['verb', 'conj_sub', 'verb', 'noun', 'adj', 'prep', 'noun', 'noun', 'prep', 'noun', 'prep', 'noun', 'adj']

**H#2**: If [‘digit’] or [‘noun num’] tag exists at sentence PoS, then it is more likely to be performance requirements.

The presence of numbers often signifies a high probability of a performance requirement. If numbers appear within sentences, regardless of their representation in numerical or alphabetical form, it tends to indicate that the sentence is likely a performance requirement. This distinction can be made based on their Part of Speech (PoS) tags. To determine the presence of numbers in a sentence, we need to identify all instances tagged as ['digit'] or ['noun num'].

A possible structure for requirements that show the locations of numbers in Arabic sentences:

(Verb + Subject + Object (1) | Object (2) | Object (3) -> Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | + name number + Noun | Digit + Noun) + (Noun | Preposition + Noun | name number + Noun | Digit + Noun) + (Noun | Preposition + Noun | name number + Noun | Digit + Noun)

CAMeL PoS Tags :

(Verb + (noun — pron) + (noun | prep + noun | noun num + noun | digit + noun) + (noun | prep + noun | noun num + noun | digit + noun) + (noun | prep + noun | noun num + noun | digit + noun).

Example:"يقوم النظام بتحديث العرض كل 60 ثانية"

Translation: "The system updates the display every 60 seconds".

CAMeL Tokens: ['يقوم', 'النظام', 'ب', 'تحديث', 'العرض', 'كل', '60', 'ثانية']

CAMeL PoS: ['verb', 'noun', 'prep', 'noun', 'noun', 'noun', 'num', 'noun']

**H#3:** Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the performance requirements.

- **Class2: Security Requirements (SE)**

**H#4:** Security requirements establish the limitations and restrictions on system access to safeguard it from unauthorized entry .

Negative sentences could be categorized more closely with security requirements. This determination can be made by examining whether any negative prefixes are present in the sentence. The presence of negative prefixes in sentences, regardless of whether they are expressed numerically or alphabetically, increases the likelihood that the sentence falls under security requirements. To check for the presence of negative prefixes in a sentence, it is necessary to inspect all ['part neg'] tags.

Negative Arabic sentences is constructed by adding one of the following negation tools:

" لا، ليس، غير، لم، لمّا، لن، لام الجحود، ما "

The following sentences are examples of security requirements with negation tools:

a) Example: " لن يتمكن المستخدمون من الوصول المباشر إلى ملفات البيانات أو قواعد البيانات "

Translation: "Users will not be able to access data files or databases directly".

CAMeL Tokens['لن', 'يتمكن', 'المستخدمون', 'من', 'الوصول', 'المباشر', 'إلى', 'ملفات', 'البيانات', 'أو', : 'قواعد', 'البيانات']

CAMeL PoS: ['part_neg', 'verb', 'noun', 'prep', 'noun', 'adj', 'prep', 'noun', 'noun', 'conj', 'noun', 'noun'].

b) Example" لا يمكن إنشاء حساب مستخدم إلا بواسطة مسؤول النظام" :

Translation: "User accounts can only be created by the system administrator".

CAMeL Tokens: ['لا', 'يمكن', 'إنشاء', 'حساب', 'مستخدم', 'إلا', 'بواسطة', 'مسؤول', 'النظام']

CAMeL PoS: ['part_neg', 'verb', 'noun', 'noun', 'noun', 'conj', 'prep', 'noun', 'noun']

**H#5**: Conditional sentence is a linguistic structure that needs a tool to link two sentences, the first is a condition for the answer to the second, and it states that something happens because of something else associated with it and causes it.

Conditional sentences in Arabic are categorized into two types: Proof sentences and Negation sentences. The structure of these sentences comprises the Conditional Particle, the Conditional sentence, the Answer Particle, and the Conditional Answer:

1.Conditional Particle: Arabic Language utilizes two common conditional particles, namely "إذا" (idha) and "لو" (law) . These particles are represented by (subordinating conjunction) in CAMeL tools [‘conj’].

2. Conditional sentence: A conditional sentence is a verbal statement that falls into two categories: proof and negation sentence. A proof sentence consists of a conditional particle followed directly by the conditional sentence, without the presence of a negation particle .(لم)On the other hand, a negation sentence includes the negation particle (لم) after the conditional particle .
3. Answer Particle: The answer particle functions as an adverb for the conditional answer. In Arabic, the answer particles include(فان, سوف, فسوف) , with the corresponding tags being:
(" ف" connective particle +" إن" Pseudo verb): فإن
(" سوف" Future particle), (" ف" Response conditional): فسوف
(" سوف" Future particle): سوف
4. Conditional Answer: The conditional answer, a verbal sentence.
So, if the sentence structure as follow its more likely to be security requirement :
Subordinating Conjunction + (Verb | Negative Particle +Verb) + (Connective Particle + Pseudo Verb) | Future Particle | (Response Conditional+ Future Particle) + verbal sentence.
CAMeL PoS Tags:
(conj + (verb | part_ neg + verb) + (part_ rc + part_ emphac | part_ fut | part_ rc+ part_ fut) + (Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun))
Example: " إذا تم إبطال حساب مستخدم فلا يمكن إعادة إنشاء مثيل له إلا بواسطة مسؤول النظام "
Translation: "If a user account is deactivated, it cannot be re-created except by the system administrator".
CAMeL Tokens : ['إذا', 'تم', 'إبطال', 'حساب', 'مستخدم', 'فلا', 'يمكن', 'إعادة', 'إنشاء', 'مثيل', 'له', 'إلا', 'بواسطة', 'مسؤول', 'النظام']
CAMeL PoS: ['conj', 'verb', 'noun', 'noun', 'adj', 'part_neg', 'verb', 'noun', 'noun', 'noun', 'prep', 'part', 'noun', 'noun', 'noun']
Example: " سوف يتم الدخول للموقع الاكلينيكي اذا كان الشخص من طاقم التمريض فقط"
Translation: "Entry to the clinical site will only be allowed if the person is a member of the nursing staff".
CAMeL Tokens: ['سوف', 'يتم', 'الدخول', 'للموقع', 'الاكلينيكي', 'اذا', 'كان', 'الشخص', 'من', 'طاقم', 'التمريض', 'فقط']
CAMeL PoS: [' part_ fut ', 'verb', 'noun', 'noun', adj', 'conj', 'verb', 'noun', 'prep', 'noun', 'noun', 'adverb'].

**H#6**: After the study of different projects and SRS documents we noticed that there is a common structure repeated in the security requirements all have the word " access" as followed:
" أن" + Verb + Subject + Object (1) | Object (2) | Object (3) ->" أن" +" Verb" + (Noun | Pronoun) +" قادرا" + (Noun | Preposition + Noun) +" الوصول"
Token [0] = أن+ verb + (Noun | Pronoun) + Token [3] =" قادرا" + (Noun | Preposition + Noun) + Token [5] =" الوصول" + (Preposition + Noun)
Example: "أن يكون الطبيب قادرا على الوصول لكافة سجلات المرضى"
Translation: "The doctor should be able to access all patient records".
CAMeL Tokens['أن', 'يكون', 'الطبيب', 'قادرا', 'على', 'الوصول', 'لكافة', 'سجلات', 'المرضى'] :
CAMeL PoS: ['conj', 'verb', 'noun', 'adj', 'prep', 'verb', 'prep', 'noun', 'noun']
Verb + Subject + Object (1) | Object (2) | Object (3) -> (Verb) + (Noun | Pronoun) + (Noun | Preposition + Noun) + (Noun | Preposition + Noun) + (Noun | Preposition + Noun)
Token [0] = verb + (Noun | Pronoun) + (Noun | Preposition + Noun) +(Adjective | Adverb) + Token [5] =" الوصول" + (Preposition + Noun).
Example : " يستطيع أصحاب العقارات المسجلين فقط من الوصول الى النظام "
Translation: "Only registered property owners can access the system".
CAMeL Tokens: ['يستطيع', 'أصحاب', 'العقارات', 'المسجلين', 'فقط', 'من', 'الوصول', 'إلى', 'النظام']
CAMeL PoS: ['verb', 'noun', 'noun', 'adj', 'adv', 'prep', 'noun', 'prep', 'noun']
Subject + Verb + Object (1) | Object (2) | Object (3)-> (Noun | Pronoun) + Verb + (Noun | Preposition + Noun | Adverb + Noun | + Adjective) "الوصول" + (Noun | Preposition + Noun | Adverb + Noun | Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective).
Example:" الطلاب لا يستطيعون الوصول لشاشة تعديل العلامات"
Translation: "Students cannot access the grade editing screen".
CAMeL Tokens['الطلاب', 'لا', 'يستطيعون', 'الوصول', 'لشاشة', 'تعديل', 'العلامات'] :
CAMeL PoS: ['noun', 'neg', 'verb', 'noun', 'prep', 'noun', 'noun']

**H#7:** Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the security requirements.

- **Class 3: Availability (A)**

**H#8:** When a sentence contains a percentage punctuation, it is more likely to indicate an availability requirement. The percent sign is easily distinguished from other punctuation marks by being preceded by a number. The percentage format is represented as [num, punc]. Therefore, if the part of speech is identified as 'num' and the '%' symbol is present in the token, the specified condition is met.
if pos == 'num' and '%' in token.
Example: " سيكون النظام متاحًا بنسبة 99٪ من الوقت خلال الأشهر الستة الأولى من التشغيل "
Translation: "The system will be available 99% of the time during the first six months of operation".
CAMeL Tokens: ['سيكون', 'النظام', 'متاحًا', 'بنسبة', '٩٩٪', 'من', 'الوقت', 'خلال', 'الأشهر', 'الستة', 'الأولى', 'من', 'التشغيل']
CAMeL PoS: ['verb', 'noun', 'adj', 'prep', 'num', 'punc', 'noun', 'prep', 'noun', 'adj', 'adj', 'prep', 'noun']
Example: " يجب أن لا يفشل المنتج أكثر من 2٪من الوقت المتاح على الانترنت "
Translation: "The product should not fail more than 2% of the available time online".
CAMeL Tokens: ['يجب', 'أن', 'لا', 'يفشل', 'المنتج', 'أكثر', 'من', ' 2 ٪', 'من', 'الوقت', 'المتاح', 'على', 'الانترنت']
CAMeL PoS: ['verb', 'conj_sub', 'neg', 'verb', 'noun', 'adj', 'prep', 'num', 'punc', 'prep', 'noun', 'adj', 'prep', 'noun']

**H#9:** A sentence indicating time duration typically adheres to the following structured formats :
Digit + punctuation + digit + token [] = صباحًا أو مساءً
The presence of such a structured sentence format is a strong indicator of an availability requirement. If the sentence follows the pattern of Digit + punctuation + digit + token [], it is more likely to convey a specific time duration, either in the morning (صباحًا) or evening (مساءً).
Example: " يجب أن يكون النظام متاحًا للاستخدام بين الساعة 8:00 صباحًا و 6:00 مساءً "
Translation: "The system must be available for use between 8:00 AM and 6:00 PM".
CAMeL Tokens: ['يجب', 'أن', 'يكون', 'النظام', 'متاحًا', 'للاستخدام', 'بين', 'الساعة', '8 ', ' : ', '00', 'صباحًا', 'و', '6 ', ' : ', '00', 'مساءً']
CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'adj', 'noun_prop', 'noun', 'noun', 'digit', 'punc', 'digit', 'noun', 'conj', 'digit', 'punc', 'digit', 'noun',]
Digit + token = صباحًا أو مساءً
Example: " يجب أن يكون المنتج متوفرا على الموقع يوميا من الساعة 9 صباحًا و لغاية الساعة 9 مساءً "
Translation: "The product must be available on the website daily from 9:00 AM to 9:00 PM".
CAMeL Tokens: ['يجب', 'أن', 'يكون', 'المنتج', 'متوفرًا', 'على', 'الموقع', 'يوميًا', 'من', 'الساعة', '9', 'صباحًا', 'و', 'لغاية', 'الساعة', '9', 'مساءً']
CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'adj', 'prep', 'noun', 'adv', 'prep', 'noun', 'digit', 'noun', 'conj', 'noun', 'noun', 'digit', 'noun']

**H#10:** The presence of these linguistic elements indicates a heightened probability of conveying availability requirements.
" When a sentence is tagged with either 'adj' or 'adv' for its part of speech, it significantly raises the likelihood of expressing availability requirements." The sentence structure may take the form of a verbal or nominal sentence, as outlined below:
Verbal sentence format :
Verb + Subject + Object (1) | Object (2) | Object (3) -> Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun | + Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective).
CAMeL PoS Tags: Verb + (noun | pron) + (noun | prep + noun | noun + adv | adj) + (noun | prep + noun | noun + adv | adj) + (noun | prep + noun | noun + adv | adj)
Example: " يكون المنتج متاحًا خلال ساعات العمل العادية "
Translation: "The product is available during regular business hours".
CAMeL Tokens: ['يكون', 'المنتج', 'متاحًا', 'خلال', 'ساعات', 'العمل', 'العادية']
CAMeL PoS: ['verb', 'noun', 'adj', 'prep', 'noun', 'noun', 'adj']
Nominal sentence format :

Subject + Verb + Object (1) | Object (2) | Object (3) -> (Noun | Pronoun) + Verb + (Noun | Preposition + Noun | Adverb + Noun | + Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective) + (Noun | Preposition + Noun | Adverb + Noun | Adjective).
CAMeL PoS Tags: (noun | pron | foriegn) + Verb + (noun | prep + noun | noun + adv | adj) + (noun | prep + noun | noun + adv | adj) + (noun | prep + noun | noun + adv | adj)
Example: "فترة تعطل النظام قصيرة بحيث لا تزيد عن 10 دقائق في السنة"
Translation: "The system downtime period is short, not exceeding 10 minutes per year".
CAMeL Tokens: ['فترة', 'تعطل', 'النظام', 'قصيرة', 'بحيث', 'لا', 'تزيد', 'عن', '10', 'دقائق', 'في', 'السنة']
CAMeL PoS (Part of Speech): ['noun', 'verb', 'noun', 'adj', 'conj', 'neg', 'verb', 'prep', 'num', 'noun', 'prep', 'noun']

**H#11:** Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the availability requirements.

- **Class 4: Look and feel (LF):**

**H#12:** Look and feel requirements, usually consider the unique needs associated with various nationalities and locations. These considerations involve recognizing the diverse cultural elements and geographical factors that shape user preferences. The words specific for names of countries, cities, or specific cultural terms are called proper nouns, and they are serving as linguistic tools that specifically denote to look and feel requirement. Therefore, the presence of the tag [noun_prop] enhances the probability of look and feel requirement.
Some examples show look and feel requirements that have proper nouns:
Example: " يجب أن يكون للموقع طابع أفريقي "
Translation: "The website should have an African character".
CAMeL Tokens: ['يجب', 'أن', 'يكون', 'للموقع', 'طابع', 'أفريقي']
CAMeL PoS: ['verb', 'conj_sub', 'verb', 'prep', 'noun', 'noun_prop']
Example: " يجب أن يتوافق المنتج مع إطار دليل تطوير التطبيقات في مدينة شيكاغو "
Translation: "The product must comply with the application development guidelines framework in the city of Chicago ".
CAMeL Tokens:['يجب', 'أن', 'يتوافق', 'المنتج', 'مع', 'إطار', 'دليل', 'تطوير', 'التطبيقات', 'في', 'مدينة', 'شيكاغو']
CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'prep', 'noun', 'noun', 'noun', 'noun', 'prep', 'noun', 'noun_prop']

**H#13:** Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the look and feel requirements.

**Class 5: Fault Tolerance (FT)**

**H#14:** Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the look and feel requirements.

- **Class 6: Legal (L)**

**H#15:** Through the study of different SRS documents, we notice that there a certain sentence structure is repeated in the legal requirement as illustrated bellow:
('verb', 'subordinating conjunction',): "يجب أن " + Subject + Object -> Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun).
CAMeL PoS Tags: Verb + (noun | pron) + (noun | prep + noun | noun + adv) + (noun | prep + noun | noun + adv) + (noun | prep + noun | noun + adv).
Example:" يجب أن يتوافق تطبيق المنازعات مع المتطلبات القانونية على النحو المحدد في لوائح التشغيل "
Translation: "The dispute resolution application must comply with the legal requirements as specified in the operating regulations".

CAMeL Tokens: ['يجب', 'أن', 'يتوافق', 'تطبيق', 'المنازعات', 'مع', 'المتطلبات', 'القانونية', 'على', 'النحو', 'المحدد', 'في', 'لوائح', 'التشغيل']
CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'noun', 'prep', 'noun', 'adj', 'prep', 'noun', 'adj', 'prep', 'noun', 'noun']
Example: " يجب أن يتوافق المنتج مع لوائح التأمين المتعلقة بمعالجة المطالبات "
Translation: "The product must comply with the insurance regulations related to claims processing".
CAMeL Tokens: ['يجب', 'أن', 'يتوافق', 'المنتج', 'مع', 'لوائح', 'التأمين', 'المتعلقة', 'بمعالجة', 'المطالبات']
CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'prep', 'noun', 'noun', 'adj', 'prep', 'noun', 'noun']
**H#16**: Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the legal requirements.

- **Class 7: Operational (O)**

**H#17:** It is more likely to be operational requirement if the ['foreign'] tag is present at sentence PoS. Non-Arabic words frequently indicate programming languages or techniques (such as HTML, SQL, etc.). It is more likely that a sentence is a non-functional requirement if there are foreign words present in it, regardless of the sentence syntax. Therefore, based on their PoS tags, foreign words are able to distinguish the operational class.
There are several potential structures that indicate when foreign terms are used in Arabic sentences:
Verbal sentence :
Verb + Subject + Object (1) | Object (2) | Object (3) -> Verb + (Noun | Pronoun | Foriegn Word) + (Noun | Preposition + Noun |Preposition + Foriegn Word | Foriegn Word) + (Noun | Preposition + Noun | Preposition + Foriegn Word | Foriegn Word) + (Noun | Preposition + Noun | Preposition + Foriegn Word | Foriegn Word).
CAMeL PoSTags :Verb + (noun | pron | foriegn) + (noun | prep + noun | prep + foriegn | foriegn) + (noun | prep + noun | prep + foriegn | foriegn) + (noun | prep + noun | prep + foriegn | foriegn).
Example: " HTML يتفاعل النظام مع أي متصفح "
Translation: "The system interacts with any browser HTML".
CAMeL Tokens: ['يتفاعل', 'النظام', 'مع', 'أي', 'متصفح', 'HTML']
CAMeL PoS: ['verb', 'noun', 'prep', 'adj', 'noun', 'foriegn']
Nominal sentence :
Subject + Verb + Object (1) | Object (2) | Object (3) -> (Noun | Pronoun | Foriegn Word) + Verb + (Noun | Preposition + Noun | Foriegn Word | Preposition + Preposition) + (Noun | Preposition + Noun | Foriegn Word | Preposition + Preposition) + (Noun | Preposition + Noun | Preposition + Foriegn Word | Foriegn Word)
CAMeL PoS Tags :(noun | pron | foriegn) + verb + (noun | prep + noun | prep + foriegn | foriegn) + (noun | prep + noun | prep + foriegn | foriegn) + (noun | prep + noun | prep + foriegn | foriegn)
Example: " Oracle SQL Server المنتج يجب أن يستخدم برنامج قواعد البيانات "
Translation: "The product should use a database management program like Oracle SQL Server ".
CAMeL Tokens: ['المنتج', 'يجب', 'أن', 'يستخدم', 'برنامج', 'قواعد', 'البيانات', 'Oracle', 'SQL', 'Server']
CAMeL PoS: ['noun', 'verb', 'prep', 'verb', 'noun', 'noun', 'noun', 'foriegn', 'foriegn', 'foriegn']

**H#18:** Operational requirement is more likely if the primary actor is the "system "النظام" ," or one of its Arabic synonyms like(الموقع, التطبيق, المنتج, البرنامج) :

The Arabic sentence could be verbal or nominal sentence, if the sentence is verbal then the main subject in the sentence will be: the system "النظام" ,that follow the main verb in the sentence. If the sentence is nominal, then the main subject in the sentence will be the system ,"النظام" ,as illustrated bellow:
Nominal sentence :
Subject + Verb + Object -> Subject +Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun)

CAMeL PoS: Subject +Verb + (noun | pron) + (noun | prep + noun | noun + adv) + (noun | prep + noun | noun + adv) + (noun | prep + noun | noun + adv)
Where token "النظام" = [0] or "المنتج" or "البرنامج" or "التطبيق" or"الموقع"
Example: " النظام يعمل على نسخ بيانات الأعمال احتياطيًا تلقائيًا واستعادتها عند الطلب "
Translation: "The system automatically backs up business data and restores it upon request".
CAMeL Tokens: ['النظام', 'يعمل', 'على', 'نسخ', 'بيانات', 'الأعمال', 'احتياطيًا', 'تلقائيًا', 'واستعادتها', 'عند', 'الطلب']
CAMeL PoS: ['noun', 'verb', 'prep', 'verb', 'noun', 'noun', 'adv', 'adv', 'conj_sub', 'prep', 'noun']
Verbal Sentence:
Verb + Subject + Object -> Verb + (Noun | Pronoun) + (Noun | Preposition + Noun | Adverb + Noun)
CAMeL PoS: Verb + (noun | pron) + (noun | prep + noun | noun + adv) + (noun | prep + noun | noun + adv) + (noun | prep + noun | noun + adv).
Where token "النظام" = [1] or "المنتج" or "البرنامج" or "التطبيق" or"الموقع"
Example: (يعمل المنتج على الأجهزة الموجودة لجميع البيئات)
Translation: "The product works on the available devices for all environments".
CAMeL Tokens ['يعمل', 'المنتج', 'على', 'الأجهزة', 'الموجودة', 'لجميع', 'البيئات'] :
CAMeL PoS: ['verb', 'noun', 'prep', 'noun', 'adj', 'prep', 'noun']
**H#19**: Through the study of the different SRS for different projects we notice that there are many terms and words are repeated in the operational requirements such as:
Performance: استجابة Response, التحقق Verification, سرعة Speed, etc.
Security: استعلامات Inquiry, آمن Security, بيانات Data, etc.
Availability: اسبوع Week, الإنترنت Internet, تعطل Failure, etc.
Look and Feel: احترافي Professional, احترام Respect, جذاب Attractive, etc.
Fault Tolerance: استثناء Exception, استعادة Recovery, انقطاع Disconnection, etc.
Legal Requirements: الترخيص License, التوجيه Guidance, تشريع Legislation, etc.
Operational: الإلكتروني Electronic, البريد Mail, الخادم Server, etc.

## 5 Evaluation

A. Heuristics Evaluation

To enhance our method's accuracy, we enlisted the help of three seasoned software engineering specialists to evaluate it. Two of them have doctorates in software engineering, while the third is a distinguished engineer working for a top software engineering company. The experts provided a numerical percentage rating out of 100 for each heuristic as shown in table 1.

Table 1: Evaluation of heuristics.

| Class | Heuristic # | Average Percentage |
|---|---|---|
| Performance (PE) | 1 | 85 |
| | 2 | 86.7 |
| | 3 | 81.7 |
| Security (SE) | 4 | 90 |
| | 5 | 83.3 |
| | 6 | 90 |
| | 7 | 90 |
| Availability (A) | 8 | 87.3 |
| | 9 | 87 |
| | 10 | 85.6 |
| | 11 | 87.7 |
| Look and Feel (LF) | 12 | 82.3 |
| | 13 | 85 |
| Fault Tolerance (FT) | 14 | 85 |
| Legal (L) | 15 | 83.3 |
| | 16 | 86.7 |

| | | |
|---|---|---|
| Operational (O) | 17 | 81.7 |
| | 18 | 83.3 |
| | 19 | 81.6 |

Based on the above percentages we derive the confidence factor in the heuristic-based classification. It varies across several consistent categories. The average percentage of Performance (PE) shows a moderate to high confidence level. Both, Security (SE) and Availability (A) clearly show consistently high percentages, suggesting a high confidence level in the classification. Classes including Look and Feel (LF), Fault Tolerance (FT), Legal (L), and Operational (O) exhibit moderate percentages, indicating a moderate confidence in the classification within these areas.

B. Outcomes Evaluation

We used 105 requirement sentences from PROMISE Software Engineering Repository [27], which was divided into seven different classes: legal, operational, performance, security, availability, look and feel, and fault tolerance, which tested on our software.

a. Data Preparation

Prior to conducting the experiments, we divided the requirement sentences into individual classes. For each class, we created a separate CSV file containing the respective sentences. Additionally, we prepared another CSV file containing all the requirement sentences combined.

b. Experimental Procedure

1. Single- Class Testing: We started by giving individual tests to each class. In order to accomplish this, we fed the sentences from each class into a Python code that used the CAMeL tools to process the data and provide the appropriate PoS tags and tokens. We were able to evaluate the effectiveness of our method for each non-functional category separately since this process was performed for every class.

2. Multi- Class Testing: We then tested all the classes collectively as part of a thorough assessment. We delivered the CSV file comprising sentences from every class along with the Python code. After processing the combined data, the code classified the required statements into the appropriate non-functional classes using our suggested heuristics.

c. Result Analysis

Through this section we show and analyze the resulting classification report for single – class testing and multi- class testing.

1. Single-Class Testing

The Single- Class Testing Results give a thorough summary of how well each class performed in the examined system's categorization. The number of input sentences per class, the number of sentences that were successfully classified, and specific classification metrics: precision, recall, F1-score, and accuracy are summarized in table 2.

Table 2: Single – class testing results

| **#** | **Class** | **# of Input Sentences** | **# of Correctly Classified Sentences** | **Classification Report** | | | |
|---|---|---|---|---|---|---|---|
| | | | | Precision | Recall | F1- Score | Accuracy |
| 1 | Performance | 19 | 16 | 1.00 | 0.84 | 0.91 | 0.84 |
| 2 | Security | 17 | 13 | 1.00 | 0.82 | 0.90 | 0.82 |
| 3 | Availability | 10 | 8 | 1.00 | 0.88 | 0.93 | 0.88 |
| 4 | Look and feel | 20 | 19 | 1.00 | 0.90 | 0.95 | 0.90 |
| 5 | Fault tolerance | 10 | 9 | 1.00 | 0.90 | 0.95 | 0.90 |
| 6 | Legal | 9 | 7 | 1.00 | 0.86 | 0.92 | 0.86 |
| 7 | Operational | 20 | 17 | 1.00 | 0.85 | 0.92 | 0.85 |

2. Multi- Class Testing

The Multi-Class Testing Results are shown in table 3, providing a detailed summary of how well each class performed overall in classifying the input sentences of the case study. It lists all of the input sentences in all classes, counts the number of sentences that were successfully classified, and provides comprehensive classification metrics including recall, precision, F1-score, and overall accuracy.

Table 3: Multi – class testing results

| Class | # of Input Sentences | # of Correctly Classified Sentences | Classification Report | | | |
|---|---|---|---|---|---|---|
| | | | Average Precision | Average Recall | Average F1- Score | Overall Accuracy |
| All Classes | 105 | 94 | 0.88 | 0.89 | 0.88 | 0.88 |

## 6 Conclusion

This paper introduces a novel method for automating the classification of Arabic user non-functional requirements (NFRs) using CAMeL Tools, a set of Natural Language Processing (NLP) tools specifically developed for Arabic. We designed strategies to classify NFRs by analyzing critical linguistic elements such as tokens, Part of Speech (PoS) tags, and lemmas. The approach, implemented in Python, aims to support software engineers in managing Arabic NFRs more efficiently, cutting down on the time and resources that are typically needed for manual classification.

The proposed methodology significantly improves the accuracy and efficiency of NFR analysis, enabling software engineers to make more informed decisions and deliver higher-quality software products. By automating the extraction and categorization of NFRs, this research optimizes workflows and provides a scalable solution for large-scale software development projects. Beyond its impact on software engineering, the study makes a noteworthy contribution to the advancement of Arabic Natural Language Processing (NLP). It addresses the distinct linguistic challenges posed by Arabic, such as its complex syntax, dialectal diversity, and cultural context, opening new avenues for further developments in NLP applications for Arabic texts.

This work establishes a foundation for enhancing machine-learning models specifically tailored to Arabic, with the potential for extension to other languages in the future.

While this research presents an effective solution for automating NFR classification in Arabic, there are numerous opportunities for further exploration. Future studies could focus on expanding the dataset and exploring cross-linguistic applications.

## References

[1] Khurshid, I., Imtiaz, S., Boulila, W., Khan, Z., Abbasi, A., Javed, A., & Jalil, Z. (2022). Classification of Non-Functional Requirements From IoT Oriented Healthcare Requirement Document. *Frontiers in Public Health, 10,* 860536. doi: 10.3389/fpubh.2022.860536.

[2] Younas, M., Jawawi, D. N. A., Ghani, I., et al. (2020). Extraction of Non-functional Requirement Using Semantic Similarity Distance. *Neural Computing & Applications, 32*, 7383–7397.

[3] Mahmoud, A., & Williams, G. (2016). Detecting, Classifying, and Tracing Non-functional Software Requirements. *Requirements Engineering, 21,* 357–381.

[4] Ramadhani, D. A., Rochimah, S., & Yuhana, U. L. (2015). Classification of Non-functional Requirements Using Semantic-FSKNN Based on ISO/IEC 9126. *TELKOMNIKA (Telecommunication Computing Electronics and Control), 13(4),* 1456-1465.

[5] Gazi, Y., Umar, M. S., & Sadiq, M. (2015). Classification of NFRs for Information Systems. *International Journal of Computer Applications, 115(22)*, 19-22.

[6] Arman, N., & Jabbarin, S. (2015). Generating Use Case Models from Arabic User Requirements in a Semi-Automated Approach Using a Natural Language Processing Tool. *Journal of Intelligent Systems, 24(2),* 277-286.

[7] Arman, N. (2015). Normalizer: A Case Tool to Normalize Relational Database Schemas. *Information Technology Journal, 5*, 329–331. ISSN: 1812-5638.

[8] Alami, N., Arman, N., & Khamayseh, F. (2020). Generating Sequence Diagrams from Arabic User Requirements Using Mada+ Tokan Tool. *International Arab Journal of Information Technology, 17(1),* 65-72.

[9] Khurana, D., Koli, A., Khatter, K., & Singh, S. (2023). Natural Language Processing: State of the Art, Current Trends, and Challenges. *Multimedia Tools and Applications, 82(3),* 3713-3744.

[10] Wahdan, A., Al-Emran, M., & Shaalan, K. (2023). A Systematic Review of Arabic Text Classification: Areas, Applications, and Future Directions. *Soft Computing,* 1-22.

[11] Singh, P., Singh, D., & Sharma, A. (2016, September). Rule-based System for Automated Classification of Non-Functional Requirements from Requirement Specifications. *In International Conference on Advances in Computing,*

*Communications and Informatics (ICACCI)* (pp. 620-626). IEEE. doi: 10.1109/ICACCI.2016.7732115.

[12] Shreda, Q. A., & Hanani, A. A. (2021). Identifying Non-functional Requirements from Unconstrained Documents using Natural Language Processing and Machine Learning Approaches. *In IEEE Access*. doi: 10.1109/ACCESS.2021.3052921.

[13] M. A. Haque, M. Abdur Rahman, & M. S. Siddik. (2019). Non-Functional Requirements Classification with Feature Extraction and Machine Learning: An Empirical Study. *In 1st International Conference on Advances in Science, Engineering and Robotics Technology (ICASERT)* (pp. 1-5). IEEE. doi: 10.1109/ICASERT.2019.8934499.

[14] Mengmeng, L., & Peng, L. (2017). Automatic Classification of Non-Functional Requirements from Augmented App User Reviews. *In EASE'17: Proceedings of the 21st International Conference on Evaluation and Assessment in Software Engineering*.

[15] Awad, E., Khamayseh, F., & Arman, N. (2023, June). Semi-Automated Classification of Non-Functional Arabic User Requirements using NLP Tools. *In Proceedings of the 41st IBIMA Conference on Artificial Intelligence and Machine Learning*, Granada, Spain. ISSN: 2767-9640.

[16] Shehadeh, K., Arman, N., & Khamayseh, F. (2024, November). Classification of Arabic User Requirements: A Semi-Automated Approach using NLP Tools. *International Journal of Advances in Soft Computing and its Application*, *16(3)*, 1-14.

[17] Shehadeh, K., Arman, N., & Khamayseh, F. (2021). Semi-Automated Classification of Arabic User Requirements into Functional and Non-Functional Requirements using NLP Tools. *In 2021 International Conference on Information Technology (ICIT)* (pp. 527-532). IEEE. doi: 10.1109/ICIT52682.2021.9491698.

[18] Jabbarin, S., & Arman, N. (2014, January). Constructing Use Case Models from Arabic User Requirements in a Semi-Automated Approach. *In 2014 World Congress on Computer Applications and Information Systems (WCCAIS)* (pp. 1-4). IEEE.

[19] Nassar, I. N., & Khamayseh, F. T. (2015, April). Constructing Activity Diagrams from Arabic User Requirements Using Natural Language Processing Tool. *In 2015 6th International Conference on Information and Communication Systems (ICICS)* (pp. 50-54). IEEE.

[20] Alami, N., Arman, N., & Khamayseh, F. (2017, May). A Semi-Automated Approach for Generating Sequence Diagrams from Arabic User Requirements Using a Natural Language Processing Tool. *In 2017 8th International Conference on Information Technology (ICIT)* (pp. 309-314). IEEE.

[21] Karim, S., Warnars, H. L. H. S., Gaol, F. L., Abdurachman, E., & Soewito, B. (2017, November). Software Metrics for Fault Prediction Using Machine Learning Approaches: A Literature Review with PROMISE Repository Dataset. *In 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)* (pp. 19-23). IEEE.

[22] Sommerville, I. (2011). *Software Engineering* (9th ed.). ISBN: 137035152, p.18.

[23] Gottesdiener, E. (2003). Requirements by Collaboration: Getting it Right the First Time. *IEEE Software, 20(2),* 52-55. doi: 10.1109/MS.2003.1184167.

[24] Batool, I., Kosar, L., & Mehmood, M. (2018). Non-Functional Requirements as Constraints and Their Values in Software Development: A Review.

[25] Umar, M., & Khan, N. A. (2011). Analyzing Non-Functional Requirements (NFRs) for Software Development. *In IEEE 2nd International Conference on Software Engineering and Service Science (ICSESS).* IEEE. doi: 10.1109/ICSESS.2011.5982328.

[26] Obeid, O., Zalmout, N., Khalifa, S., Taji, D., Oudah, M., Alhafni, B., ... & Habash, N. (2020, May). CAMeL Tools: An Open Source Python Toolkit for Arabic Natural Language Processing. *In Proceedings of the Twelfth Language Resources and Evaluation Conference* (pp. 7022-7032).

[27] Karim, S., Warnars, H. L. H. S., Gaol, F. L., Abdurachman, E., & Soewito, B. (2017, November). Software Metrics for Fault Prediction Using Machine Learning Approaches: A Literature Review with PROMISE Repository Dataset. *In 2017 IEEE International Conference on Cybernetics and Computational Intelligence (CyberneticsCom)* (pp. 19-23). IEEE.