Palestine Polytechnic University

Deanship of Graduate Studies and Scientific Research

Master of Informatics

Semi-Automated Classification of Non-Functional Arabic User Requirements using NLP Tools

By:

Eman Faiz Awad

Supervised By:

Dr. Faisal Khamayseh

Prof. Nabil Arman

Thesis submitted in partial fulfillment of requirements of the degree

Master of Informatics

February, 2024

The undersigned hereby certify that they have read, examined and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University:

The undersigned hereby certify that they have read, examined, and recommended to the Deanship of Graduate Studies and Scientific Research at Palestine Polytechnic University the approval of a thesis entitled:

Semi-Automated Classification of Non-Functional Arabic User Requirements using NLP Tools, submitted by **Eman Awad** in partial fulfilment of the requirements for the degree of Master in Informatics.

Graduate Advisory Committee:

Dr. Faisal Khamayseh (Supervisor), Palestine Polytechnic University.

Signature:———————————— Date:————————————

Prof. Nabil Arman (Co-Supervisor), Palestine Polytechnic University.

Signature:———————————— Date:————————————

Dr. Nancy Alriji

(Internal committee member), (Palestine Polytechnic University.).

Signature:———————————— Date:————————————

Dr. Husam Suwad

(External committee member), (Palestine Technical University - Kadoori).

Signature:———————————— Date:————————————

Thesis Approved by:

Dr. Nafeth NaserAldeen

Dean of Graduate Studies & Scientific Research

Palestine Polytechnic University

Signature:……...……

Date:…..……...……

DECLARATION

I declare that the Master Thesis entitled" Semi-Automated Classification of Non-Functional Arabic User Requirements using NLP Tools" is my own original work, and herby certify that unless stated, all work contained within this thesis is my own independent research and has not been submitted for the award of any other degree at any institution, except where due acknowledgement is made in the text.

Eman Faiz Awad

Signature:———————————— Date:————————————

STATEMENT OF PERMISSION TO USE

In presenting this thesis in partial fulfilment of the requirements for the master's degree in Informatics at Palestine Polytechnic University, I agree that the library shall make it available to borrowers under rules of the library. Brief quotations from this thesis are allowable without special permission, provided that accurate acknowledgment of the source is made. Permission for extensive quotation from, reproduction, or publication of this thesis may be granted by my main supervisor, or in his absence, by the Dean of Graduate Studies and Scientific Research when, in the opinion of either, the proposed use of the material is for scholarly purposes. Any copying or use of the material in this thesis for financial gain shall not be allowed without my written permission.

Eman Faiz Awad

Signature:———————————— Date:————————————

الملخص

تعتبر هندسة المتطلبات مرحلة مهمة في دورة حياة تطوير البرمجيات، حيث تشمل كل من المتطلبات الوظيفية والمتطلبات غير الوظيفية. تحدد المتطلبات غير الوظيفية خصائص الجودة لنظام، بما في ذلك الأداء والأمان والتوافر والشكل والمظهر والتسامح مع الأخطاء والقانونية والتشغيلية، وهي أمور أساسية لتلبية احتياجات المستخدمين وفرض قيود إضافية على جودة البرمجيات. يعتبر تصنيف المتطلبات غير الوظيفية من مستندات متطلبات المستخدم أمرًا صعبًا، يتطلب مهارات متخصصة ومعرفة في المجال. يتطلب التصنيف اليدوي لهذه المتطلبات وقتًا وجهدًا عقليًا كبيرًا من المطورين، مما يجعل التصنيف التلقائي أو شبه التلقائي لهذه المتطلبات من مستندات المتطلبات ذو قيمة عظيمة بحيث تقلل هذه الطريقة من الجهد اليدوي والوقت في تصنيف هذه المتطلبات. تقدم هذه الأطروحة نهجًا جديدًا لتصنيف متطلبات المستخدم غير الوظيفية باللغة العربية باستخدام أدوات **CAMeL** وهي أداة لمعالجة اللغات الطبيعية بحيث أننا نقترح مجموعة من القواعد الشبه التلقائية بالاستناد إلى بناء وتركيب الجملة العربية الأساسية لاستخراج صفات مميزة لكل منها ومن ثم تصنيف هذه المتطلبات إلى سبع فئات. يتم استخراج الجذور اللغوية وأقسام الكلمة لمتطلبات المستخدم المحللة باستخدام أدوات **CAMeL** ومن ثم يتم تحديد الفئة الأقرب لكل جملة عن طريق تطبيق القواعد على مخرجات أداة **CAMeL**. تم تنفيذ الطريقة المقترحة باستخدام أدوات **CAMeL 1.3.1** وبرنامج كتب بلغة بايثون في بيئة **Windows 10** وأظهرت النتائج كفاءة وفاعلية الطريقة المقترحة في تصنيف متطلبات المستخدم غير الوظيفية باللغة العربية.

Semi-Automated Classification of Non-Functional Arabic User Requirements using NLP Tools

Eman Faiz Awad

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 thesis introduces a novel semi-automated categorization approach for Arabic non-functional user requirements using CAMeL Tools, a natural language processing 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.

DEDICATION

To My Beloved Family,

To my parents, whose unwavering support and encouragement have been my guiding light throughout this journey.

To my dear husband, for being my rock, my confidant, and my biggest cheerleader. Your constant encouragement and understanding have given me the strength to pursue my dreams.

To my precious children, who inspire me every day to be the best version of myself. Your love and joy have filled my life with purpose and motivation.

To my supportive sisters and caring brothers, thank you for always standing by my side, offering words of wisdom, and being my pillars of strength.

This thesis is dedicated to all of you…

ACKNOWLEDGEMENT

I would like to express my deepest gratitude and appreciation to my thesis supervisors, Dr. Nabil Arman and Dr. Faisal Khamayseh, for their invaluable guidance, unwavering support, and insightful feedback throughout the process of conducting this research. Your expertise, encouragement, and dedication have been instrumental in shaping this thesis and enriching my academic journey.

I am also immensely grateful to my colleague in the master's program, Karmel Shehada, for her endless assistance and support. Her contributions and willingness to lend a helping hand have been truly invaluable, and I am sincerely thankful for her generosity.

I would like to express my sincere gratitude to the faculty members at the College of Graduate Studies for their invaluable guidance, support, and encouragement throughout the course of my thesis.

Special appreciation goes to Eng Maeen and Khalil for their invaluable assistance and for generously providing the necessary data for my research. Your support and collaboration have been integral to the success of this project.

Finally, I extend my sincere thanks to all those who have contributed in any way to the completion of this thesis. Your support and encouragement have been instrumental in this achievement.

Thank you all

Table of Content

LIST OF FIGURES

LIST OF TABLES

LIST OF ABBREVIATIONS

ANN	Artificial Neural Networks
BERT	Bi-directional Encoder Representations from Transformers
CR	Classification Rules
FN	False Negative
FP	False Positive
FRs	Functional Requirements
GATE	General Architecture for Text Engineering
LSTM	Long Short-Term Memory
ML	Machine Learning
NLP	Natural Language Processing
NFRs	Non-Functional Requirements
PoS	Part of Speech
RNN	Recurrent Neural Network
RE	Requirements Engineering
SDLC	Software Development Lifecycle
SRS	Software Requirements Specification
TN	True Negative
TP	True Positive
UML	Unified Modeling Language

CHAPTER ONE
INTRODUCTION

1.1. Motivation

The realm of software development is a dynamic and ever-evolving field where the ability to meet and exceed user expectations is the key to the success of a project. Central to this endeavor is the accurate identification and classification of user requirements, a process that can be both complicated and challenging, particularly when dealing with non-functional requirements (NFRs). Considering this backdrop, our thesis is propelled by a deep-seated motivation to address critical challenges in the software engineering landscape [1].

Non-functional requirements described as the quality attributes of the system. Determining the NFR is one of the most difficult steps during software development. These NFRs encompass attributes such as performance, security, usability, reliability, legal, look and feel, fault tolerance and scalability, which are crucial in determining the overall quality of a software product. They dictate not just what a software system should do but how it should do it, encompassing a wide spectrum of qualitative aspects that have a profound impact on user satisfaction [2]. Accurately identifying, extracting, and classifying NFRs is a complex task that demands meticulous attention to detail, cultural sensitivity, and linguistic expertise [3]. Within this complicated ecosystem, there is an obvious need for a solution that not only solves these difficulties but also streamlines the process for Arabic-speaking consumers [4].

Non-functional requirements (NFRs) are a critical aspect of software design that deeply influences its overall quality and user experience. They extend beyond the core functionality to define how a system operates, and they are often critical to the success or failure of software applications [5].

This research focuses on leveraging Natural Language Processing (NLP), which lies at the intersection of computational linguistics and artificial intelligence [6]. NLP has revolutionized human-machine interactions by enabling machines to understand and process human language in ways previously unimaginable. This advancement has broader implications, including its application to the Arabic language.

Applying NLP to Arabic language processing opens up opportunities for redefining how non-functional requirements (NFRs) are identified and categorized. Customized NLP tools for Arabic aim to address the unique linguistic and cultural nuances of the Arabic-speaking world.

Arabic-speaking users represent a significant and growing global demographic, and their distinct characteristics pose both challenges and opportunities in software development. Neglecting these nuances can lead to misalignment between software products and users, compromising user experiences [7]. This research aims to create culturally and linguistically accurate software tailored to Arabic-speaking audiences, going beyond basic functionality.

The thesis tackles the complex intersection of NFRs, advanced NLP capabilities, and the intricate Arabic cultural context. It aims to develop a semi-automated classification system that combines human expertise with NLP, improving the precision and speed of NFR classification while respecting Arabic's linguistic subtleties and cultural nuances [8].

This proposed system will be intricately designed to handle Arabic's complexity, including its extensive lexicon, diverse dialects, and cultural expressions. The research aims to set a new standard in NFR classification for Arabic-speaking users, advancing culturally sensitive and linguistically inclusive software engineering tools. The ultimate goal is not only to enhance technology but also to contribute to a more inclusive and empathetic global community, harnessing NLP's transformative power.

1.2. Problem Statement:

In the field of software engineering, the correct classification of user requirements, mainly non-functional requirements (NFRs), has a major role to play in achieving the functionality and quality of software systems. Despite their importance in clearly influencing the overall quality and performance of software, NFRs often take the back seat behind functional requirements. Because it is often considered that the requirements are only meant to specify a system’s functionality and capabilities and they won’t involve operational aspects. The Classification of these requirements into certain categories is a difficult task especially if the requirements are in Arabic language because of the Arabic language unique linguistic and cultural features. NFRs consist of a wide range of qualities such as Performance, Security, Availability, Look and Feel, Fault Tolerance, Legal, Operational; these are prerequisites, which are essential to use positive experience and, moreover, they are necessary to provide good operational systems. Proper NFR classification is important to enhance the integration and management of NFR within the Software Development Lifecycle (SDLC) and reflects the variety of users' intricate needs.

Manually classifying NFRs is an onerous task, requiring domain-specific knowledge. It could be error prone and inefficient in large-scale projects. Complexity increases further with Arabic content because of its varied dialects, complex syntax, and deep culture. Traditional methods are manual and have mainly addressed the content classification problem. There is a limitation to manual and traditional contents classification, especially with little or no Arabic language datasets available for applying machine learning algorithms. Machine learning algorithms have been used in the requirement classification for some languages like English. To tackle these challenges, our research proposes a semi-automated approach using NLP tools specifically designed for Arabic. Leveraging the capabilities of CAMeL Tools, our approach aims to reduce the manual effort, time and labour involved in classification. We are developing a system that can automatically identify and classify NFRs in Arabic text with a high degree of accuracy as well as cultural sensitivity. This ground-breaking approach will transform the way NFRs are categorised, resulting in software development practices that are more inclusive, efficient and responsive to the needs of their global users, particularly Arabic speakers.

1.3. Proposed Solution

We introduce in our thesis a novel semi-supervised approach that is dedicated to classify NFRs in Arabic software documentation. We introduce a novel semi-supervised approach in our thesis, dedicated to classifying NFRs in Arabic software documentation. Our proposed definition and systematic categorization of NFRs draw from methodologies utilized by prominent researchers in the field [9][10][8]. This existing literature may not specifically categorize NFRs into these seven categories. However, our categorization comprises seven categories: Performance, Security, Availability, Look and Feel, Fault Tolerance, Legal, and Operational.

This classification categorized the non-functional requirements (NFRs) in seven categories which includes Performance, Security, Availability, Look and Feel, Fault Tolerance, Legal, and Operational. This categorization is in line with published frameworks used in other scholarly works which make our analysis comprehensive and clear. Our approach is designed to work specifically with the linguistic and cultural features that are unique to the Arab world. It relies on a heuristic approach that exploits specific features of CAMeL Tools, an NLP toolkit developed to handle the specific challenges of the Arabic language. We utilize NLP methods, including tokenization, part-of-speech tagging, and sentence segmentation, tailored to efficiently analyze and understand Arabic texts. Our approach specifically targets the unique patterns and idioms of the Arabic language. The salt of our technique is a set of heuristics rooted

in Arabic sentence structure basics extracted based on domain wisdom and linguistic breakdown. These heuristics are well designed to resolve classify NFRs into the seven predefined category types, seeking accurate classification without demanding extensive training datasets which are painfully insufficient for Arabic NFRs. We have illustrated our approach for classifying Arabic NFRs in Figure 1.1 in our thesis. Our method of classification is a semi-supervised method rather than using typical data-driven machine learning techniques. This is a shift to a more contextually aware and nuanced analysis in software requirements and not just a data analysis approach but considering and complying with the user's needs for software requirements. This is noticeably different from the current software engineering practices hence it is a great contribution to the practice of software engineering. We fill in the gap of software engineering in recognizing NFRs in the To-be operational system. This is a break of the current machine learning culture on software requirement analysis. This will not only add value to the software engineering industry but also take into context the user's requirements for software, especially in the Arabic speaking market.

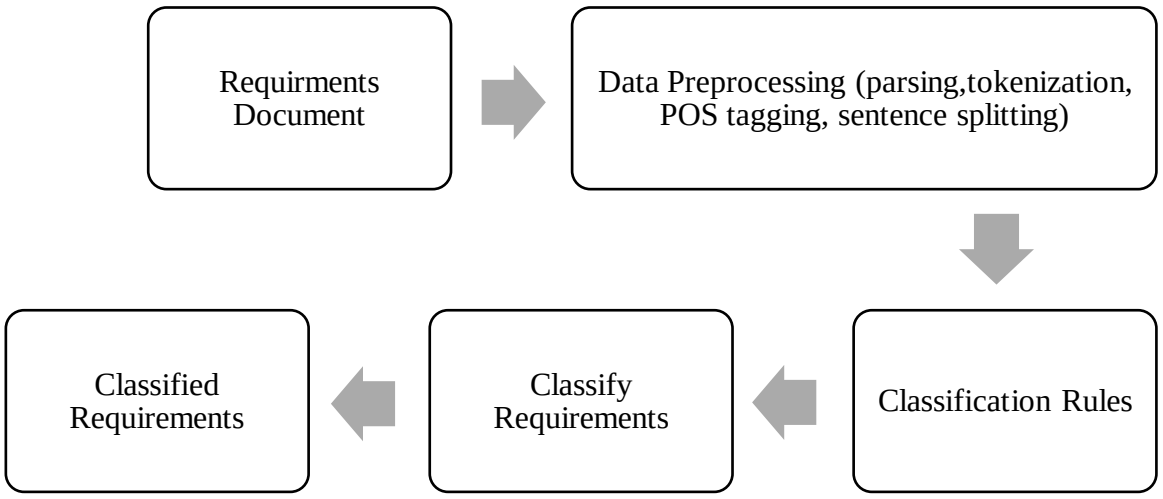

Figure 1.1: Proposed Solution

1.4. Research Steps

As part of our research, we have followed a systematic and holistic approach while categorizing NFRs in Arabic documentation with the help of NLP (Figure 1.2). Attempts have been made to cover the entire process in key phases, focusing on the unique complexities of the Arabic language and its cultural diversity.

- **Data Collection:** We collected data which contained a huge collection of requirement documents.
- **Data Preprocessing**: Utilizing CAMeL Tools, we perform preprocessing tasks on the collected data. This step involves parsing the Arabic text for morphological analysis, tokenization is done to divide the text into individual words, also called tokens. PoS tagging is responsible for assigning the grammatical structure of the word. Sentence splitting: tokenize the sentence into several sentences.
- **Classification Rules Development**: Instead of using machine learning algorithms, we use a heuristics approach. We propose a set of classification rules (CR) derived from a deep linguistic analysis of Arabic language considering Arabic sentence constructs to categorize the NFR into seven classes: Performance, Security, Availability, look and feel, Fault Tolerance, Legal, Operational. We did not consider scalability class SC as it overlap with the other classes.
- **Classify Requirements**: Once the rules have been put in place, we are able to categorize the preprocessed NFR's. This step is semi-automated where we are guided by the rules which we setup that are effective at distinguishing the seven types of NFRs.
- **Post-Classification Processing (Evaluation)**: The processed classified necessities then go for post-processing, which is utilized to refine and approve the consequences of posting. The reason here is to expand the level of precision in ordering non-useful necessities. The framework considers the setting and the Arab semantics shown in the created content.

In this research, we have taken a broad and detailed approach to developing a partially automated system that can successfully and efficiently classify non-functional requirements in Arabic documents, with special attention to the complexity of the language and the cultural characteristics it carries.

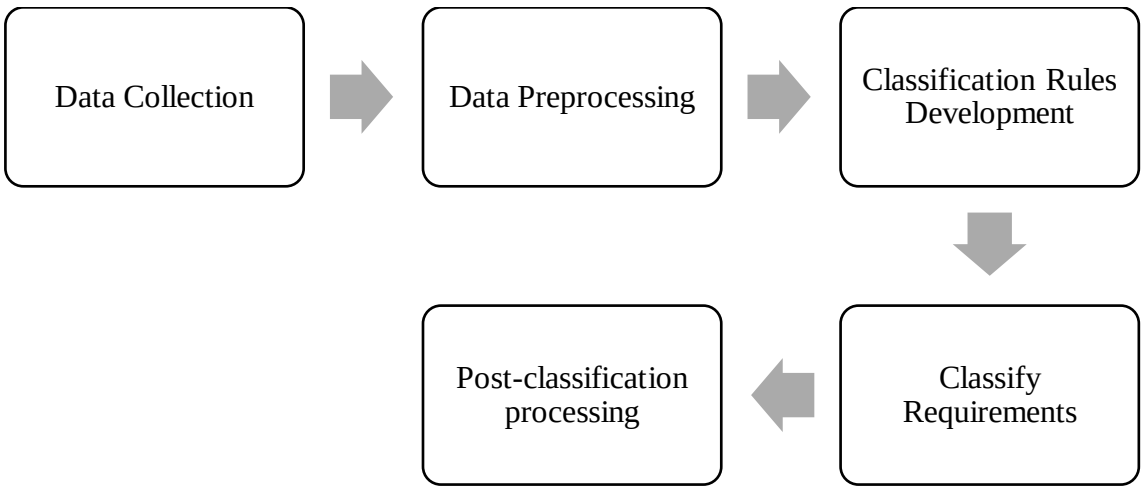

Figure 1.2: Flowchart of the Steps of Our Approach

1.5. Research Objectives

This research aims to achieve the following objectives:

1. **Develop a Heuristic-Based NFR Classification Framework:**
   - Develop and implement a more complex semi-automated system that heuristically classifies non-functional requirements (NFRs) in Arabic language documents. This system will use Arabic-optimized NLP tools and will be able to process and comprehend complex language structures and cultural idioms.
   - We are devising a set of heuristics that are developed by the analysis of the language to do the task of classifying NFR's into predefined classes like performance, security, availability, look and feel, fault tolerance, legal, and operational without having to build machine learning models.

2. **Optimize System for Arabic Linguistic and Cultural Nuances:**
   - The classification process must be tailored to the specific challenges of the Arabic language, such as morphological richness and dialectal variation, while ensuring high linguistic fidelity and cultural relevance.
   - Integrate contextual understanding and semantic analysis into our heuristics to enhance the system's capability to understand the contextual context of Arabic NFRs and also the semantics of the Arabic NFRs.

3. **Validate the Classification Heuristics and System Performance:**
   - Application of vigorous tests with an extensive and varied set of data in order to validate the efficiency of the heuristic rules and the overall system of classification.

- The system performance should be evaluated by both qualitative and quantitative measures such as classification accuracy, rule precision, and adaptability of the system with different Arabic dialects and domains; hence, the approach should be robust and scalable to be fit for real-world applications.

1.6. Contribution

In the area of natural language processing (NLP) applied to software development, our work presents a new approach compared to what has already been published. With a special interest in Arabic, we are building a semi-automated, heuristic-based system that helps to classify non-functional requirements (NFRs) from Arabic language documentation. What we contributed in the natural language processing (NLP) field for software development is considered new compared to the published papers before. Our main focus is on the Arabic language. We are producing a system that depends on heuristics and is semi-automated to classify the non-functional requirement (NFR) from an Arabic-language document. Our approach provides an in-depth engagement with the complex grammar, rich morphology, and diverse dialects of Arabic, which, combined with an accurate interpretation of the subtleties of the language, allow us to achieve a level of understanding higher than any other available commercially, resulting in unparalleled technological advancement and cultural sensitivity to the issues at hand. Since we did not have large datasets available in Arabic, machine learning was not considered a worthwhile solution for NLP. Instead, we adopted heuristic rules based on the expert linguistic analysis system that helped us categorize the NFRs into specific classes such as performance, security, availability, look and feel, fault tolerance, legal, and operational. The heuristic method is very efficient in processing millions of Arabic documents in a scalable and very efficient way. This methodology is geared towards setting a new standard of NFR classification that covers a large proportion of the world-wide non-English-speaking linguistic community and consequently opens a new era of NLP and software development software and opens new areas and NFR standards to development, thus contributing to world-wide software development. This research in a totally connected world is very important to provide technology with the Arabic and Oriental languages as the only way to become closer to or friends with other nations via technology. This way, we can provide sustainable technological development globally.

1.7. Research Importance

This study is a big step forward in achieving the integration of software engineering and Arabic language processing. We are concerned with identifying and classifying non-functional requirements (NFRs) in software development, an issue that has not been given enough

attention. Non-functional requirements are required for software, like required functionality; they determine the overall behavior of the system in regards to the user experience, such as usability, reliability, security, and certain software features. We have developed an algorithm applied as a semi-automated system where we process information based on racial, lexicon, and heuristic approaches, enriched by Arabic-optimized NLP tools. Unlike the typical machine learning algorithms, our system is based on a number of heuristics that have been developed through an extensive linguistic analysis of Arabic. Our novel approach significantly extends the state of the art of software engineering methodologies and will result in higher software quality from non-functional aspects, which meets user expectations. Moreover, our research has a direct, meritorious impact on the Arabic-speaking society worldwide, where it is specifically concerned with the idiosyncratic challenges imposed by the linguistic nature of Arabic on software development. Building such a system will allow us to make the development of software that is consistent with Arabic, which will increase the satisfaction and user engagement of the Arab speakers, and automating the NFR classification process will make the NFR classification process less error-prone and more efficient, which is critically important for languages like Arabic in order to develop a better quality of software that is sensitive to the Arabic culture. The research that we have undertaken illustrates a somewhat modern software development in a cross-cultural, global setting. Integrating linguistic and cultural intelligence into software development is crucial in today's interconnected world. Our research serves not only the Arabic-speaking community but also sets a standard in software development, combining NLP and heuristic analysis. This approach is transformative for the broader field of global software engineering, showcasing the importance of linguistic and cultural considerations in creating effective software solutions.

1.8. Thesis Organization

This thesis is organized as follows:

- **Chapter 1: Introduction**

In the first chapter, the context for the thesis is established. It begins with the motivation, highlighting the need to address requirements engineering challenges, especially for Arabic-speaking customers. The problem statement outlines specific challenges, and the proposed solution introduces the conceptual framework. Research steps detail the methodology, and research aims clarify the goals. Contributions outline expected scientific and practical impacts, while research importance emphasizes relevance within the field. Thesis organization provides a structure overview.

- **Chapter 2: Background**

Chapter 2 offers a comprehensive background to the thesis. It explores requirements engineering (RE), requirements specification (SRS), types of requirements, and the importance of clear user requirements. The chapter then delves into user requirements written in Arabic, setting the stage for the relevance of Natural Language Processing (NLP) tools, with a focus on CAMEL Tools for Arabic language analysis.

- **Chapter 3: Literature Review**

This chapter conducts a systematic review of literature related to the thesis. It compares software requirements classification using rule-based and machine learning approaches for English and German. The review also covers automated production of UML diagrams and its relevance to Arabic and English user requirements.

- **Chapter 4: Research Approach**

Chapter 4 details the research approach for categorizing non-functional user needs. It describes the step-by-step methodology, reasoning, and unique methodologies developed to address Arabic language complexities in software development.

- **Chapter 5: Evaluation**

The evaluation chapter explains the process for assessing the solutions offered. It discusses the evaluation methodology, experiments, and results, providing empirical evidence of the research approach's effectiveness, including both achievements and limitations observed during the assessment.

- **Chapter 6: Conclusion and Future Work**

The final chapter includes a conclusion and future work section. It summarizes significant results and contributions, reflecting on the research journey and offering a critical appraisal of the work performed. Additionally, it suggests directions for future research and developments in the subject, providing a comprehensive conclusion to the entire research endeavor

CHAPTER TWO
BACKGROUND

2.1. Requirements Engineering (RE)

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 [11]. RE abides by a simple definition. It is a method of developing software repeatedly by tracing users anticipations and requirements totally and carefully in order to prove that the ultimate product is what the users unwish. [10]. 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 [12]. 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 [13]. The purpose of these specifications is not only to provide what the software system should do but also how it must respond under different conditions, the limitations of the software, and how it should be expected to react not only to problems but to normal jump situations. So it is extremely beneficial to have a well-structured SRS in order to have all of the project's stakeholders engaged in the software development process, such as developers, testers, managers of the project, and clients [7]. RE separates its requirements into two categories, namely functional and non-functional. Functional requirements (FRs) describe in detail the functions, processes, and features that the software system must deliver, whereas non-functional requirements (NFRs) cover the quality attributes that govern how the system should behave. These kinds of non-functional requirements are also critical in designing the whole user experience as well as making sure that not only the software system works properly but also constantly exceeds user expectations.

RE is a discipline that concentrates on good communication, comprehensible documentation and determination of user objectives to result in software that is before the allocated time. If it is not implemented then there is a huge chance that the development of the software will be affected. It is a proactive risk reduction activity that is used to eliminate the probability of project failure. RE also reduces the development cost by doing the software development right at the first attempt and revealing what it is that the software has to do. In the world that we live

in today, RE is the best practice and gives the best results, this means that software products are enforced to under the development according to what the user is expecting [14].

The process of requirements engineering is an intricate journey that can be divided into four core sections, each of which is integral to the success of a software development project. The first component is known as a feasibility study, and it is used to assess the feasibility and viability of the project. This is followed by a phase where the project is broken apart in depth with respect to the goals in terms of the difficulties as well as the potential resources, helping the stakeholders realize their decisions on the project. The requirement elicitation and analysis phase is the next phase, which comes after the feasibility study. During this phase, end-users, customers, and domain experts will work together to understand and establish what their requirements and expectations are. The criteria, which have been obtained, are then assessed by the team to ensure they are unambiguous, have no missing requirements, and that all parties understand the requirements [15].

The next important stage is requirement specification, which means the determination of the needs of the clients. More importantly, it contains a comprehensive and cohesive list of these items. It's usually in the form of a SRS document. In this stage, the SRS will act like a system blueprint. SRS describes what the software is going to do. The software requirement specification states both the functional and non-functional requirements and the way the software should behave and perform [16]. Finally, after the requirements have been stated, they undergo a process of requirement validation. Requirement validation is a process in which the stated requirements are checked and validated to make sure that the requirements that the customer says he is looking for are really what the project objectives are and are also realized and checked by the various stakeholders. From the software engineering perspective, a major goal of the requirements engineering processes is the quality control of requirements, which, through systematic requirements validation, prevents mistakes in the early development of the software and reduces the need for costly changes later in the development. In other words, the four processes of requirements engineering aid effective software development mainly through communication support, risk management, and the delivery of the software product that users want and industry standards [17].

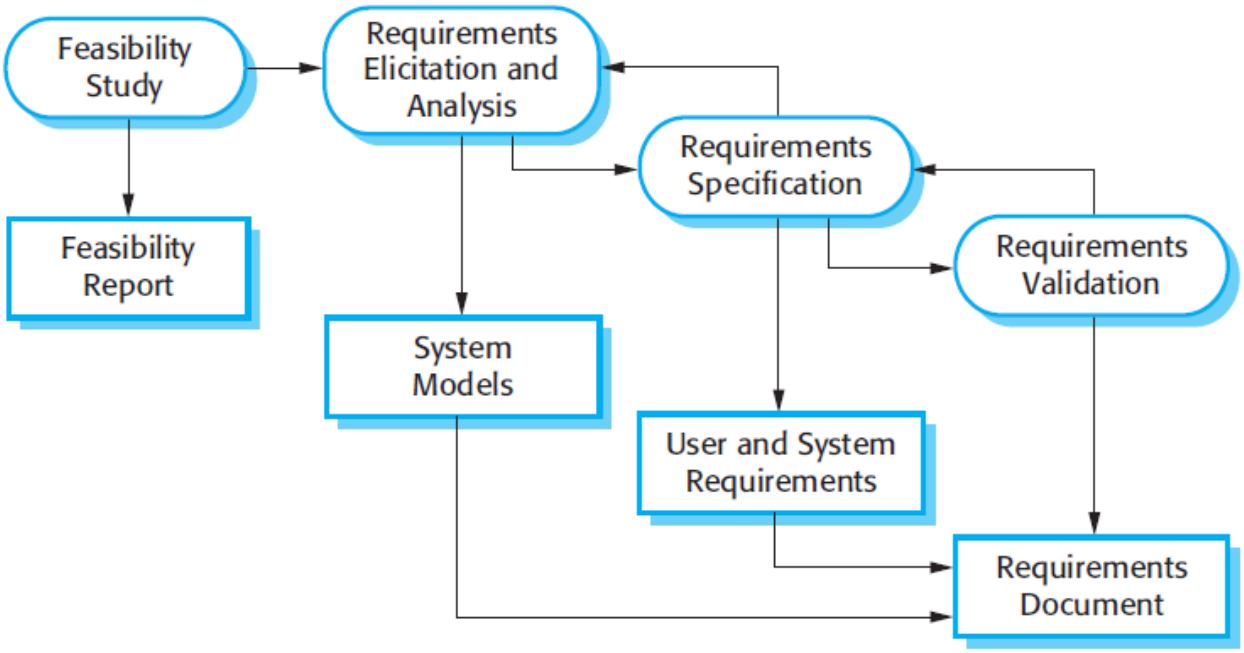

Figure 2.1:Process of Requirements Engineering

2.1.1. Software Requirements Specification (SRS)

The Software Requirements Specification (SRS) document is crucial in software engineering, acting as a detailed guide outlining the software's requirements and specifications. It forms a key agreement among stakeholders like clients, developers, and project managers, defining the software's intended functionality and operational constraints. The SRS specifies both functional requirements (actions the software must perform) and non-functional requirements (quality attributes like performance, security, and usability). This document guides all stages of the software development lifecycle, from design to updates, and helps avoid misunderstandings that can cause project delays and failures to meet user needs.

The SRS also serves as a validation tool, ensuring the final product matches the agreed specifications. It assists in estimating costs and timeframes, enhancing project planning and management. Furthermore, it improves communication among team members, reducing ambiguities and providing a common language for different expertise. Essentially, the SRS is a comprehensive document that forms the foundation for a software project, offering a detailed roadmap from inception to deployment.

A well-crafted SRS includes various essential elements. It starts with an introduction that outlines the document's purpose, scope, definitions, acronyms, abbreviations, and references, providing the reader with context. It then describes the intended audience and the software's practical applications, ensuring clarity and relevance of the information provided [18-21].

The Software Requirements Specification (SRS) is a crucial document in the software development lifecycle (SDLC), serving as a comprehensive blueprint that guides the entire development process. The main body of the SRS details functional requirements—actions the system must perform, including user interactions, data processing, and behavior under specific conditions. It also outlines non-functional requirements like performance metrics, security features, and reliability standards. These combined elements form the SRS's core, defining what the software will do and how it will perform [22].

The SRS also specifies system constraints and assumptions, user scenarios, use cases, interface requirements, and data models, providing a holistic view of the software's intended functionality, interactions, and data flow. It includes performance requirements and design constraints related to external standards or regulations. Appendices offer additional details, while an index facilitates document navigation. Altogether, the SRS serves as a detailed specification for subsequent software design, development, and deployment phases, aiming to create a product that meets user needs and expectations [23].

Throughout the SDLC, the SRS plays a multifaceted role. Initially, it acts as a repository of agreed-upon requirements, establishing a clear vision of the final product. It then guides designers in making architectural decisions and formulating system design. During the testing phase, the SRS becomes a basis for developing test cases and validation protocols, serving as the standard against which software is verified. It helps manage the project scope, preventing scope creep and facilitating a formal change control process. In the maintenance phase, the SRS aids in troubleshooting and guiding future enhancements [24]. Furthermore, it is a vital communication tool across technical and non-technical stakeholders, aiding in project planning, time and cost estimation, and risk management. Overall, the SRS is indispensable in ensuring that each development phase aligns with the initial vision and that the final software product fulfills stakeholder expectations.

2.1.2. Types of Requirements

In the realm of software engineering, the distinction between functional and non-functional requirements is pivotal, delineating the broad spectrum of expectations that software is anticipated to meet. Functional requirements detail the various tasks the software must perform, specifying actions the system must be able to execute in direct response to user inputs, events, or interactions. They define the core operations of the software: what it will do in terms of processes, data manipulation, and workflows. For instance, a functional requirement for an email application might stipulate that the system should be able to send an email within a certain

time frame after the user hits the send button. These requirements are often characterized by use cases that provide a narrative of typical user interactions, offering concrete examples of how the software will be used [25]. Conversely, non-functional requirements (NFRs) specify the quality attributes of the software system, describing not what the software will do, but how it will perform under various conditions and constraints. They encompass criteria such as performance, which dictates the response times and throughput rates; reliability, which ensures the software's stability and consistency over time; and usability, which focuses on the user experience and ease of use. Non-functional requirements also address security, dictating the levels of data protection and user authentication necessary; scalability, outlining how well the software can adapt to increased workloads; and maintainability, defining the ease with which the software can be updated and modified. For example, a non-functional requirement for the same email application might require that it maintains a 99.9% uptime, indicating the system's reliability and availability.

The dichotomy between functional and non-functional requirements is fundamental because it collectively encapsulates the end-to-end spectrum of software capabilities, from its utility to its endurance and efficiency. Functional requirements drive the development of features, while non-functional requirements guide the system's architecture and long-term viability. Together, they form a comprehensive suite of specifications that software must fulfill to be considered complete and operational. In drafting an SRS, a meticulous balance between these two types of requirements is crucial, as overlooking either can lead to a system that fails to satisfy end-user expectations or one that is not sustainable in the long term [26]. Thus, in defining the success of a software product, both functional and non-functional requirements hold significant weight, with each playing an integral role in delivering a well-rounded, robust software solution. In the tapestry of software engineering documentation, the distinction between user requirements and system requirements is a critical one, each serving a unique function in capturing the expectations and specifications of a software system. User requirements are expressed from the perspective of the end-user and encapsulate the goals, needs, and desired outcomes of the software utilization. These requirements are typically articulated in natural language and are often less technical, focusing on what the user wants to achieve without delving into the intricacies of how the system will deliver those functionalities. For example, a user requirement might state that a user needs to retrieve a record quickly and easily, without specifying the underlying technologies or algorithms that will enable this functionality.

System requirements, on the other hand, are derived from user requirements but are characterized by their detailed and technical nature. They provide a comprehensive and precise description of the functionality and conditions necessary for the software system to fulfill user requirements. These are often detailed enough for a system designer to use as a blueprint for building the system. System requirements include detailed specifications of data structures, algorithms, system interfaces, and other technical parameters that dictate the system's development and operation [27]. For instance, a system requirement might specify the database management system to be used, the query response time, or the exact method by which records are to be retrieved and presented to the user. The clear delineation between user and system requirements is vital as it addresses different stages of the requirements engineering process and caters to various audiences. User requirements are primarily concerned with ensuring that the stakeholders' and users' needs are comprehensively gathered and understood, serving as an initial guide for the development process. In contrast, system requirements translate these needs into technical specifications for the development team, ensuring that the software built aligns with the functional and non-functional demands of the user base. This distinction also aids in effective communication within the project team and among stakeholders. While user requirements are accessible to a non-technical audience, providing a basis for initial agreements and approvals, system requirements are primarily utilized by the project's technical team members as a definitive reference for designing and building the system. Together, user and system requirements form the continuum of specification that guides the transition from conceptual understanding of user needs to the technical realization of those needs in the form of a software product [28].

In the intricate process of system design, the interplay between different types of requirements is both subtle and substantial, with each category bearing its unique impact on the final design. Functional requirements, for example, might include specifics such as the ability of an e-commerce platform to process transactions securely and efficiently, or the capability of a database system to execute queries and return results within a specified timeframe. These requirements directly inform the system's core functionalities, leading to design decisions about the necessary algorithms, data processing mechanisms, and transaction handling protocols. The implementation of such functional requirements will dictate the development of particular features and user interfaces, compelling system designers to create architectures that enable these functionalities while ensuring user-friendly interactions [29]. Non-functional requirements, which address the quality attributes of the system, have a more pervasive and

often more complex impact on system design. Consider the requirement for high availability in a cloud storage service, which necessitates the design of redundant systems and failover mechanisms to ensure uninterrupted service. Or take the demand for scalability in a social media application, which requires a design that accommodates a growing number of users and data without degradation in performance. Non-functional requirements like these influence decisions regarding the underlying infrastructure, the choice of technologies, and the system's overall resilience and adaptability. They often lead to the incorporation of advanced design patterns, the selection of robust frameworks, and the consideration of future growth during the design phase.

User requirements, expressing the end-user's perspective, might manifest as the need for an intuitive workflow in a software application or the requirement for real-time notifications in a project management tool. These requirements shape the system design by focusing on user experience (UX) design elements, necessitating a user-centered design approach that prioritizes ease of use, accessibility, and user engagement. They influence the layout of the user interface, the interaction models, and the visual design, ensuring that the system is not only functional but also pleasurable and efficient to use. System requirements, translating user needs into technical specifications, have a definitive and detailed influence on system design. For instance, a system requirement might stipulate the use of a particular SQL database with specific performance benchmarks or the integration with a third-party authentication service using OAuth protocols. These precise technical details compel designers to incorporate specific technologies and architectures into the system to meet these stipulated requirements, often dictating the system's structure, its modules, and the interactions between its components [30].

2.1.3. Benefits of Good User Requirements

The requirements of a software project are very important to the successful completion of the latest version. The requirements give the development team a firm direction from the customer and a common goal alignment among the stakeholders. The project requirements help the project or development leader guide the team to the outcome desired [5]. For clients or end-users, clear requirements guarantee that their needs are accurately captured, leading to a product that meets their demands and enhances satisfaction. For developers, clear requirements define the scope of work, enabling focused efforts and reducing the likelihood of rework. This clarity also reduces cognitive load, allowing developers to focus on problem-solving and innovation. For project managers and business analysts, clear requirements are benchmarks for measuring project progress, aiding in accurate planning, resource allocation, and timeline estimation. They

simplify change management, allowing effective decision-making and change control. For quality assurance teams, clear requirements are the basis for comprehensive testing strategies, ensuring each requirement is verified and improving communication with developers. In regulatory compliance and auditing, clear requirements ensure the software meets industry standards and legal requirements, protecting the organization from liabilities [31]. Clear requirements build trust among stakeholders, manage expectations throughout the project lifecycle, and ensure the delivery of a product that aligns with stakeholders' visions and users' needs. Well-defined requirements are essential for successful project execution and delivery, providing a detailed roadmap for all project activities [32]. They enable effective risk management, detailed system design, and high-quality software development. Well-defined requirements enhance team communication and stakeholder engagement and are vital for agile project adaptation. Accurate initial requirements result in cost savings throughout the project lifecycle, preventing revisions and rework caused by evolving or misunderstood requirements. They ensure optimal resource allocation and reduce the costs associated with ambiguous requirements, feature creep, and post-deployment fixes. Accurately capturing requirements from the beginning aligns with cost avoidance in project management, reducing developmental expenditures and maintenance costs, thus contributing to a favorable total cost of ownership and a robust bottom line for the organization [33].

2.1.4. 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 [33]. 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 [33]. Further, incorrect formatting in the Arabic language may arise from inadequately equipped requirements

management tools for Arabic in terms of text direction, wrong character encoding, and integration with a language system on the level from left to right [34]. In addition, the lack of standard software engineering terminology for support being communicated in Arabic, as mentioned above, is another major challenge [35]. To overcome these challenges, suitable methodologies and tools should be developed for Arabic, including Natural Language Processing (NLP) tools for analyzing the text, tools for building a database for standardized technical terminology, and comprehensive stakeholder training in, but not limited to, how to document requirements in Arabic. Ultimately, this will ensure suitable software is developed for the Arabic market, meeting user needs and requirements [21].

Cultural and linguistic aspects are very important in software development, especially when dealing with global audiences and languages such as Arabic, which encompasses deep history and complex structures. For example, its right-to-left script dominantly impacts software interface design and textual content presentation. The unique linguistic features of Arabic, such as the root and pattern system, lots of diacritics, and vast dialects, all in all require a special localization treatment in order to be accurately conveyed [36]. It's very critical to recognize cultural values, behaviors, and communication styles in the Arab world because software user expectations and user interaction with technology are significantly influenced by their cultures. For instance, cultural sensitivity in software design calls for such issues as choosing the appropriate symbols to represent functions, the exact greeting formulations to use, and the actual user interface structure. Besides, language translation of technical documentation and interfaces is far beyond literal translation; it involves cultural adaptation to the local market, including different socio-cultural contexts and idiomatic expressions used in the Arabic language. Bilingual subject matter experts with a deep cultural understanding are essential when developing software applications for Arabic speakers or capturing requirements in Arabic to ensure that software products are culturally appropriate and linguistically accurate, providing an intuitive and respectful user experience [36]. Case studies of Software Requirements Specification (SRS) documents for Arabic software applications will illustrate these challenges and solutions. For instance, a system for learning Arabic language dictated that the SRS should specify how the system interacted with the user through the dual bilingual interface. This SRS also addressed issues including script rendering, right-to-left text support, and culturally appropriate content. Another example is an e-commerce application designed for the Middle Eastern market, whose SRS included respect for specific linguistic requirements, such as handling Arabic search queries, which are complicated by the morphological and agglutinative

nature of the language. As we can see, it is very important to deal with the linguistic and cultural aspects in software production in Arabic market.

When developing the Software Requirements Specification (SRS) for the Government Public Service Portal in a particular Arabic-speaking country, special attention had been given to comprehensively localize it, beyond language to include the administrative and legal issues, cultural nuances as well as details of the region. The mentioned SRS is having detail that it ensues the date and currency formats as per the region along with the incorporation of cultural elements within the user interface of the developed application to enhance the local citizens' familiarity and comfort level [35]. These case studies are sufficient to demonstrate the complexity of structuring SRS documents for Arabic software systems and reveal the linguistic aspects, cultural considerations, and engineering aspects a development team needs to know in order to achieve a well-designed SRS document. If all these have been put into mind during the development of these case studies, the outcome will be considered good practice and guidelines for software engineers in the future, helping to bridge the cultural contexts of Arabic with the technical aspects of software development.

2.2. 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 [37]. 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 [38].

2.2.1. 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 [39]. 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 [39].

2.2.2. Importance of CAMeL Tool

In Arabic NLP, the CAMeL Tool is of supreme importance, as it is one of the few tools that gives the aforementioned Windows interface while also covering many complexities and hidden intricacies of the Arabic language [39]. As already mentioned, Arabic is a prominent language with very rich morphology, different dialects, and a different structure in terms of its script, which makes it challenging for most of the conventional NLP tools. CAMeL Tool is one of the few tools that has given particular attention to all those challenges and is offering specialized algorithms and functionalities for those mentioned Arabic linguistic phenomena.

2.2.3. Functionality and Features

The CAMeL Tool is a package with lots of different tools that are very vital to the Arabic process. For instance, it has advanced tokenization for Arabic, script-based tokenization, morphological analysis for Arabic based on root words and derivatives, and PoS tagging for correct syntactic analysis. The tool also has a sentence splitter and a text normalization, which are very important due to the variations in Arabic script and dialectal differences. And the tools have named entity recognition for Arabic, which is very difficult because of the nature of the Arabic language.

2.2.4. Superiority Over Other Tools

What differentiates the Arabic NLP tool, CAMeL Tool, from any other available NLP tool is the fact that Arabic is thoroughly studied. Even though the Arab World is very cohesive and highly homogenous, unlike what is widely believed, Arab people speak different dialects of Arabic and not just standard Arabic. CAMeL Tool is known for being able to process Standard Arabic, which is mostly written in newspapers, books, or presented on news channels. The dialects that are spoken in the Arab World are therefore processed by CAMeL Tool. While most of the NLP tools have an Arabic module, the Arabic NLP Tool, CAMeL Tool, is updated continuously to include the latest research findings in Arabic NLP [39].

2.2.5. Symbol Representation and Their Significance

Within the tool, there is a wide range of symbols and annotations that play an essential part to the understanding/interpretation of Arabic text. The symbols used in the CAMeL Tool are annotations that have a specific meaning or purpose. There are a few different types of annotations that can be marked; for example, a word is a root word, the plural of the word, the gender of the word, and other stuff. Song text could be used for NLP tasks. To be able to use the NLP task, we need to know these symbols to use them correctly. Because these symbols give deep insights into the Arabic language's grammar and syntax.

The CAMeL tool is without a doubt the best resource for Arabic NLP due to its specialization, extensive features, and ability to adapt to the complexity of the Arabic language. Using the CAMeL tool is ideal for our research because it simply brings such accuracy and no wasted effort to processing and analyzing Arabic non-functional requirements.

CHAPTER THREE
LITERATURE REVIEW

3.1. Automated Classification of Non-Functional User Requirements Using Machine Learning Algorithms

In this section, we consider software requirement classification by means of machine learning (ML) approaches. We introduce innovative approaches to identify and classify non-functional requirements (NFRs) and comprehend informal text descriptions in open-source projects. These articles present fascinating ideas that may automate the process of requirements analysis, potentially saving time and enhancing the accuracy of software development.

The main goal of this paper [40] is to solve the issue of the classification of non-functional requirements in an IoT-oriented healthcare system, which is error-prone and time-consuming when done manually. The paper conducts experiment on various machine learning techniques such as logistic regression, support vector machine, multinomial naïve bayes, k-nearest neighbors, ensemble, and random forest, and they also introduce a novel hybrid KNN rule-based approach. The present study points out that the hybrid KNN rule-based machine learning algorithm presents better performance with a classification accuracy of 75.9% on average. The research is further important for proposing a machine learning technique to classify non-functional requirements of IoT healthcare systems and provides a new dataset, even though it is limited (104 requirements). This dataset can be used for further analysis in the area since it has been built specifically for this purpose. It should be noted that the results should be compared with other situations with a large data set to gather a more general conclusion or use other classifiers.

This article [41] discusses how non-functional requirements can be automatically identified and classified in software development. This research conducts a study to analyse a model that has been produced automatically, through syntactic as well as semantic analysis aided by machine learning. In order to be able to study this new model, the authors employ 79 public non-functional requirements documents without any constraints and use several machine learning algorithms like Naïve Bayes. In addition, this research uses a number of statistical tools to analyse the same dataset, while employing both traditional and advanced semantic analysis tools, such as word2vec developed publicly by Google and BERT, which represent bi-directional encoder representations from Transformers (BERT) models. This approach shows classification accuracy from 84% to 87% using statistical-based vectorization, 88% to 92% with

word embedding semantic methods, and further improves 2.4% when combined with different models trained on different features than the best individual classifier. This paper advocates that the proposed method is an efficient and accurate approach to classifying non-functional requirements in software documents compared to existing methods.

The purpose of the paper is [42] about the second RE17 data challenge on the detection of functional requirements (FR) vs. non-functional requirements (NFR), focused on the data with “quality attributes (NFR)" by the implementation of supervised machine learning. Further, the paper looks at the fine-grained detection of certain types of NFR, such as usability, security, operational, and performance requirements. The authors applied a machine learning approach, while features are implemented by meta-data, lexical, and syntactical. The class imbalance in the dataset has further been handled by under- and over-sampling approaches. In regard to validating the coding, the used classifier was the Support Vector Machine classifier (SVM), which achieved approx. 92% precision and recall for the functional vs. non-functional task (FR vs. NFR), but further on, for specific NFRs, such as security or performance NFRs, the precision and recall were very high. Further, in the paper, the authors tried to understand the discriminative power of FRs and NFRs, the impact of the sampling strategies, and the impact of the additional dataset on the accuracy of the classification.

The paper [43] has proposed a new hybrid deep learning model to detect and classify non-functional requirements (NFRs) of mobile applications, such as performance, supportability, usability, and reliability, from user reviews written in Arabic. The proposed model in this paper combines three deep learning architectures, i.e., a recurrent neural network (RNN) and two long short-term memory (LSTM) models. In the beginning, Arabic textual user reviews were collected to establish the grounds for the study since the user reviews provide crucial information about the apps. For evaluating the performance of the proposed hybrid model, it was compared with different machine learning classifiers and deep learning architectures such as artificial neural networks (ANN), LSTM, and bidirectional LSTM. Results achieved through the study showed that the proposed hybrid model outperforms by achieving a 96% F1 score compared to others. This will be useful for mobile app developers to improve the quality of their apps, as it can effectively detect and address NFR issues based on natural user feedback.

The paper [44] explores the critical role of Non-Functional Requirements (NFR) in software development, specifically in influencing system architecture. The main aim is to extract relevant keywords from NFR descriptions using text mining techniques and then classify these descriptions into one of nine NFR types. The methodology involves using Information-Gain

measure to extract keywords from pre-categorized specifications and developing classification models using eight different Machine Learning (ML) techniques. The study utilized 15 projects developed by MS students at DePaul University, which included 326 NFR descriptions, to evaluate these models. The results focus on the performance of these ML models in terms of classification and misclassification rates to identify the most effective model for predicting each NFR type. Notably, the Naïve Bayes model was found to be the most effective for predicting NFRs related to "maintainability" and "availability".

The paper [45] highlights the importance of non-functional requirements in the early stages of software development, emphasizing their role in determining model alternatives and implementation criteria. It discusses how recent advancements in artificial intelligence, specifically machine learning and text mining, have enabled the automated extraction and classification of quality attributes from textual data. The study proposes a supervised categorization approach for the automated extraction and classification of non-functional specifications. A well-known dataset was utilized to validate this approach, yielding significant results, particularly in terms of security and performance, with a specific range of 85% to 98% effectiveness. The best results were achieved in combining security, performance, and usability considerations.

Table 3.1:Summary of related works of automated classification of NFR using Machine Learning Algorithms

Ref.	Main Aims	NFR Extraction Techniques	Dataset Used	Output NFR Classes
[40]	Automated Classification of NFR From IoT Oriented Healthcare Requirement Document.	For classification ML algorithm: LR, SVM, MNB, KNN, RF, KNN rule-based. For Feature Extraction: BoW and TF-IDF.	New dataset is created which includes requirements for IoT-oriented healthcare systems.	availability, security, usability, look and feel, legal and licensing, maintainability, operability, performance, scalability, fault tolerance, portability.
[41]	Automated Classification of NFR utilising semantic and syntactic investigation.	ML approaches: NB, SVM, LR, CNN. NLP techniques: Random and Word embedding vectorization methods.	The author utilised a dataset of open necessities archives (unadulterated) that comprises 79 unconstrained requirements reports.	reliability, performance, security, availability, and usability.
[42]	Automatic classification of FRs and NFRs using supervised machine learning.	ML approach: SVM classifier. Incorporates meta-data, lexical, and syntactical features. Under- and over-sampling strategies for imbalanced classes.	"Quality attributes (NFR)" dataset.	Functional Requirements (FRs) and Non-functional Requirements (NFRs), with specific focus on usability, security, operational, and performance NFRs.
[43]	Detecting and Classifying NFRs of Mobile Apps.	Deep Learning Models: RNN, LSTM, Bidirectional LSTM. Uses NLP methods for dataset extraction from user reviews.	Dataset constructed from Arabic textual user reviews of mobile apps.	Usability, reliability, performance, and supportability NFRs of mobile apps.
[44]	Mining NFRs using Machine Learning Techniques.	ML techniques: 8 different ML algorithms. Uses Information-Gain measure for keyword extraction from NFR descriptions.	15 projects developed by MS students at DePaul University, containing 326 NFR descriptions.	Nine types of NFRs
[45]	Extraction and classification of non-functional requirements from text files.	Supervised learning approach. Uses AI techniques like machine learning and text mining for extraction and classification.	Well-known dataset (specifics not detailed in the summary).	Specific range for security, performance, and best results together for security, performance, and usability NFRs (specific NFR types not detailed in the summary).

3.2. Non-Functional User Requirements Classification Using Feature extraction

This paper [19] proposes a dual approach to classifying non-functional requirements (NFRs) to enhance software quality and reduce the manual effort involved in identifying requirement sentences from Software Requirement Specification (SRS) documents. The classification is performed using a rule-based technique based on linguistic relations. The classification accuracy is also tested on the PROMISE corpus, which achieves high accuracy with 97% precision and 96% recall. The paper also investigates the inspection of NFRs, thematic roles, and the General Architecture for Text Engineering (GATE) framework for improving software requirement analysis.

The paper [46] deals with classifying NFRs in software requirements specification (SRS) documents that usually contain a mix of NFR and functional requirements (FR) statements. The NFRs are subjective and important for software system constraints and behavior, and thus, they require special attention in software modeling and development. The manuscript provides an automated way of detecting NFR sentences using a text classifier that is also supplied with a PoS tagger. The approach outperforms existing works with an achieved accuracy of 98.56%, employing 10-fold cross-validation. This work is part of a bigger project aiming to apply natural language processing techniques in software requirements engineering.

This paper [47] discusses how to automate the detection and classification of NFRs based on information retrieval (IR) techniques. The key motivation behind this paper's theme is the importance of early detection of NFRs, which allows us to provide for system-level constraints in the early stages of architectural design rather than inserting them later. The technique described has identified candidate NFRs in requirements specifications, meeting minutes, interview notes, memos with stakeholder comments, etc. The effectiveness of our classification algorithm is validated with an experiment involving fifteen requirements specifications that were developed as part of MS class project work at DePaul University, and a case study on classifying NFRs in a large (350–400 page) free-form requirements document originating from Siemens Logistics and Automotive Organization is also described.

Table 3.2:Summary of related works of NFR Classification using Feature extraction

Ref.	Main Aims	Technique for NFR Extraction	Dataset Used	Output NFR Classes
[19]	Classifying NFRs to enhance software quality and reduce	Rule-based technique using linguistic relations.	PROMISE corpus.	Accuracy, Suitability, Security, Operability, Understandability, Attractiveness, Time

	manual effort in SRS documents.			Behavior, Resource Utilization
[46]	Automating detection of NFR sentences in SRS documents, focusing on subjective and crucial NFRs.	Text classifier with part-of-speech (POS) tagger.	An integrated engineering toolset (IET)	-
[47]	Automating detection and classification of NFRs for early integration into architectural designs.	Information retrieval-based method, identifying NFRs in various document types.	Fifteen requirements specifications from MS students at DePaul University, and a document from Siemens Logistics and Automotive Organization.	Availability, Legal, Look & Feel, Maintainability, Operability, Performance, Scalability, Security, Usability

3.3. Previous studies Related to Arabic User Requirements Conducted at PPU

The paper [48] introduces a semi-automated approach for classifying functional and non-functional requirements in software engineering, specifically for documents written in Arabic. It utilizes natural language processing (NLP) tools combined with a set of heuristics based on the basic constructs of Arabic sentences. The goal is to efficiently extract and categorize requirements from Arabic software requirement documents into functional and non-functional requirements. This research aims to reduce the cost and time associated with manual classification, thereby aiding software engineers in delivering quality software that fully meets user expectations.

The paper [49] focuses on automating the construction of Unified Modeling Language (UML) models, particularly use case models, from textual user requirements in the field of automated software engineering. It emphasizes the importance of UML use case models in object-oriented software system development and describes the main principles for obtaining these models. The approach utilizes a natural language processing tool to analyze user requirements written in Arabic, extracting nouns, noun phrases, verbs, and verb phrases to identify potential actors and use cases. The paper outlines the steps of this approach and validates it through an experiment with graduate students experienced in use case modeling.

The paper [50] is about the significance of automated software engineering, especially in the domain of requirements analysis and modeling. In this paper, the major drawbacks of manual development of systems and software requirements are removed quite a lot. The main advantage of using automated systems and software engineering is improved system and software quality. The paper has also introduced a semi-automated method for drawing activity diagrams from user requirements written in Arabic by using the MADA+TOKAN parser. The proposed

approach has removed the cost and time involved in manual activity diagram drawing, and it increases the cost and time effectiveness of the whole software development life cycle by reducing the time and cost of software development when using automated software engineering.

The article [51] deals with a subject slightly touched on by very little academia so far: automated software engineering, and more specifically, constructing UML models semi-automatically from Arabic textual user requirements. The article initially argues for the importance of UML use case models, which are very essential and significant artifacts in object-oriented (OO) development methodologies. The article introduces using MADA+TOKAN to parse the Arabic user requirement statements to decompose the components like nouns, noun phrases, verbs, and verb phrases. These components are crucial to identify the potential use cases and the actors. The use case modeling steps are explained, and the future will be to validate the steps and implement them in the research project.

The research paper [52] reveals a recent approach that helps software engineers analyze phases of the software system's life cycle by generating sequence diagrams from a set of users' requirements written in Arabic. The proposed approach is a semi-automated approach that yields a semi-automated result, with the software developer still required to provide subjectivity input during the analysis phase. The proposed approach uses an NLP tool to generate PoS tags for the Arabic user's requirements. Then, from the proposed sets of heuristics added and based on these PoS tags, the proposed approach identified the sequence diagram components, which are objects, messages, and workflow transitions. The generated sequence diagrams are represented using XML format, enabling them to be drawn with sequence diagram drawing tools. The approach's effectiveness is evaluated using three case studies from Isra Computer and Programming Company, focusing on the correctness and completeness of the participants and the messages exchanged between them.

Table 3. 3: Previous Studies Related to Arabic User Requirements Conducted at PPU

Ref.	Main Aims	Technique for NFR Extraction	Dataset Used	Output NFR Classes
[48]	Semi-Automated classification of Arabic functional and non-functional requirements using NLP tools.	NLP tools combined with a set of heuristics based on Arabic sentence constructs.	SRS document	Functional and Non-functional Requirements.
[49]	Generating UML use case models from Arabic user requirements in a semi-	NLP tool to analyze Arabic user requirements, extracting nouns, noun phrases, verbs, verb phrases.	-	UML Use Case Models.

	automated approach using NLP tools.			
[50]	Constructing activity diagrams from Arabic user requirements using NLP tools.	MADA+TOKAN parser for Arabic user requirements.	-	Activity Diagrams.
[51]	Generating UML use case models from Arabic user requirements in a semi-automated approach.	MADA+TOKAN parser to extract components from Arabic user requirements.	-	UML Use Case Models.
[52]	Generating sequence diagrams from Arabic user requirements in a semi-automated approach using NLP tools.	NLP tool for parsing Arabic user requirements to produce PoS tags, with heuristics.	-	Sequence Diagrams with focus on correctness and completeness of components.

CHAPTER FOUR
RESEARCH APPROACH

This chapter is dedicated to the classification methodology for software Arabic user non-functional requirements, distinguishing between the different categories. The novel approach outlines the process of analyzing Arabic user requirements using the NLP CAMeL Tools, that used for tokenizing and generating the part of speech tags of the requirement sentences that implemented using Python programming language. We introduce a set of heuristics based on fundamental structures of Arabic sentences to facilitate the classification.

4.1 Arabic User Non-Functional Requirements Classification Approach

In our study we classify NFR into seven main types: 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 undertaken by PPU students and Software Requirements Specifications (SRS) documents aimed at developers. 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.

In our project, we utilize an empirical methodology detailed in Figure 4.1 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. 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 categorized collection of non-functional Arabic user requirements.

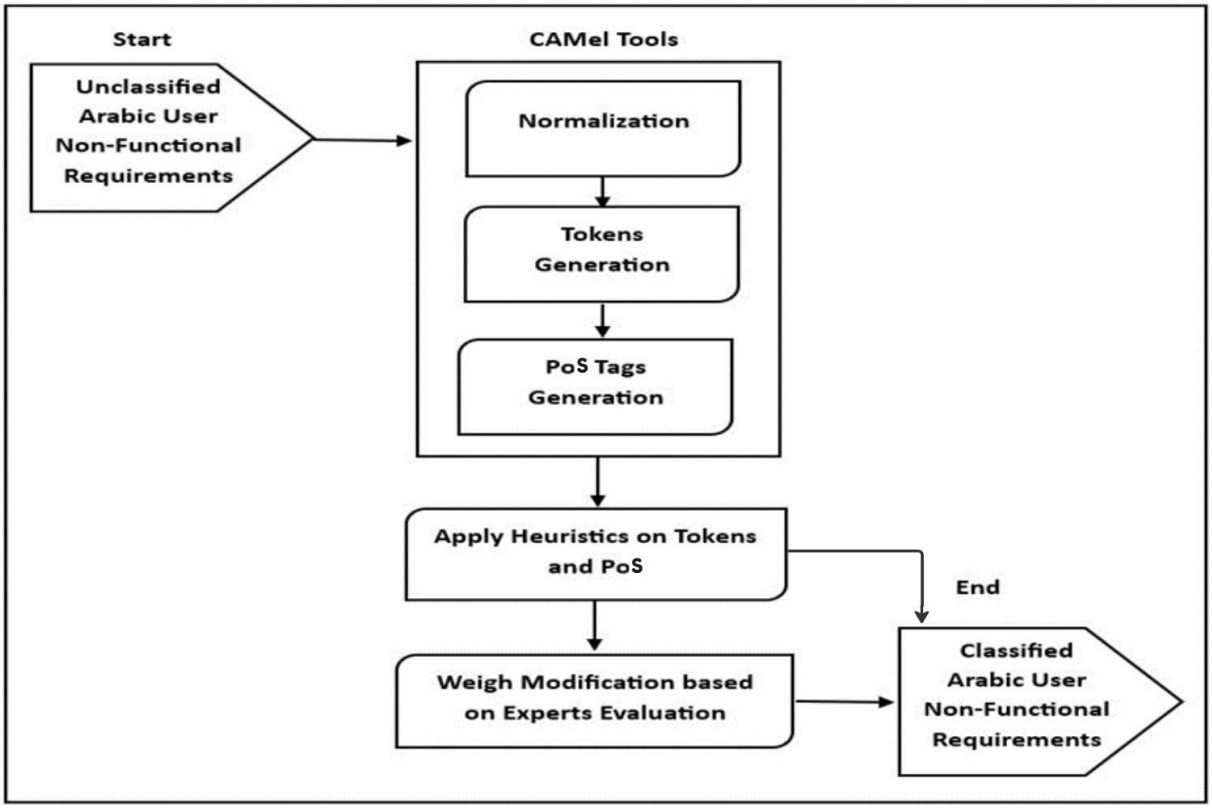

Figure 4.1:Empirical Methodology

4.1.1 Non-Functional User Requirements Linguistic Features

- **Class 1: Performance (PE):**

1. May have Common structural elements may encompass expressions like: "at an acceptable time" and "in the right time "في الوقت المناسب", "في وقت مقبول". This is represented by H#1.

2. Frequent use of numbers and digits, these numbers typically refer to: number of users, speed, or time. This is represented by H#2.

3. Frequent use of several keywords and terms, these keywords are summarized in Table 4.1 This is represented by H#3.

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

1. May have negation tool in the sentence. This is represented by H#4.

2. May have Conditions, permissions written in conditional sentence format. This is represented by H#5.

3. May have common structures that have word "Access ."الوصول" ,"This is represented by H#6.

4. May contain some keyword regarding (sign in, viruses protection) as shown in the keyword Table 4.2. This is represented by H#7.

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

1. The presence of a percentage (%) punctuation followed by a number is denoted as H#8.

2.May have Time duration like (8:00 am till 8:00 pm). This is represented by H#9.

3. May have Adjectives or adverbs like (Available, break). This is represented by H#10.

4. May have keywords as mentioned in Table 4.3. This is represented by H#11

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

1. May have a proper noun in the sentence. This is represented by H#12.

2. May have keywords of Colors, shape, GUI, sounds, and list format as shown in Table 4.4. This is represented by H#13.

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

1. May have verbs and keywords as: break down, fail, error occur, problems, emergency, and power outage which are compiled in Table 4.5. This is represented by H#14.

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

1. The sentences usually have a common structure. This is represented by H#15.

2. May have keywords: (law, legislation, legal requirements, rules, licensing, standards, regulations, legal issues). Shown in the keyword Table 4.6. This is represented by H#16.

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

1. May contain non-Arabic words like: (HTML, Windows, firewall). This is represented by H#17.

2. May Use the word system as the main subject (system work, system interaction). This is represented by H#18.

3. May have keywords, as shown in the keywords Table 4.7. This is represented by H#19.

4.1.2 The Proposed Heuristics

- **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, and an illustrative example is provided below:

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, theses terms are summarized in Table 4.1.

- **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 ,'قواعد' : ,'أو' ,'البيانات' ,'ملفات' ,'إلى' ,'المباشر' ,'الوصول' ,'من' ,'المستخدمون' ,'يتمكن' ,['لن'
['البيانات'

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

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

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

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

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 [29].

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 [29]:

1.Conditional Particle: Arabic Language utilizes two common conditional particles, namely "إذا"(idha) and "لو" (law) [29]. 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 [29]. 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) [29] ,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, theses terms are summarized in Table 4.2.

- **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, theses terms are summarized in Table 4.3.

- **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, theses terms are summarized in Table 4.4.

- **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, theses terms are summarized in Table 4.5.

- **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, theses terms are summarized in Table 4.6.

- **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: ['متصفح', 'أي', 'مع', 'النظام', 'يتفاعل', '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, theses terms are summarized in Table 4.7.

Table 4.1:Performance Requirements Keywords

#	English keywords	Arabic keywords
1	Response	استجابة
2	Verification	التحقق
3	Register	التسجيل
4	Waiting	انتظار
5	Answer	إجابة
6	Administration	إدارة
7	Repetition	إعادة
8	Completion	إكمال
9	Cancellation	إلغاء
10	Construction	إنشاء
11	Performance	أداء
12	Maximum	أقصى
13	Research	بحث
14	Update	تحديث
15	Flow	تدفق

16	Registration	تسجيل
17	Activation	تفعيل
18	Report	تقرير
19	Second	ثانية
20	Save	حفظ
21	Minute	دقيقة
22	Speed	سرعة
23	Long	طويل
24	Practical	عملية
25	Attached	مرفق
26	Synchronization	مزامنة
27	User	مستخدم
28	Complete	مكتمل
29	Time	وقت
30	Takes	يأخذ
31	Leave	يترك

Table 4.2: Security Requirements Keywords

#	English keywords	Arabic keywords
1	Inquiry	استعلامات
2	Revocation	إبطال
3	Clinical	إكلينيكي
4	Security	آمن
5	Data	بيانات
6	Audit	تدقيق
7	Registration	تسجيل
8	Access	الدخول
9	Collision	تصادم
10	Categories	تصنيفات
11	Report	تقرير
12	Integration	تكامل
13	Protection	حماية

14	Harmful	ضارة
15	Display	عرض
16	Correct	صحيح
17	Viruses	فيروسات
18	Authorized	مخول
19	User	مستخدم
20	Authentication	مصادقة
21	Processing	معالجة
22	Product	منتج
23	Forbid	منع
24	Trusted	موثوق
25	Employees	موظفين
26	System	نظام
27	Access	وصول
28	Protect	يحمي
29	Tell	يخبر
30	Guaranteed	يضمن
31	Prevent	يمنع
32	Access	الوصول
33	Only	الوحيد
34	Members	أعضاء
35	Safe	آمنًا

Table 4.3: Availability Requirements Keywords

#	English keywords	Arabic keywords
1	Week	اسبوع
2	Response	استجابة
3	Internet	الإنترنت
4	Operation	التشغيل
5	Service	الخدمة
6	Work	العمل
7	Customers	العملاء
8	Web	الويب

9	Day	اليوم
10	Disruption	انقطاع
11	Broadcast	بث
12	Percentage	نسبة
13	Failure	تعطل
14	Technical	تقني
15	Presence	تواجد
16	Availability	توفر
17	Schedule	جدول
18	Fault	خلل
19	Support	دعم
20	Hour	ساعة
21	Year	سنة
22	Long	طويل
23	Period	فترة
24	Duration	مدة
25	Rate	معدل
26	Product	منتج
27	Percentage	نسبة
28	Time	وقت
29	Achieve	يحقق
30	Daily	يوميا

Table 4.4:: Look and Feel Requirements Keywords

#	English keywords	Arabic keywords
1	Professional	احترافي
2	Respect	احترام
3	Framework	إطار
4	Buttons	أزرار
5	Formatting	تنسيق
6	Appeal	جاذبية
7	Attractive	جذاب

8	Logo	شعار
9	Shape	شكل
10	Sound	صوت
11	Panel	لوحة
12	Color	لون
13	Brand	ماركة
14	Simulation	محاكاة
15	Appearance	مظهر
16	Standards	معايير
17	Animated	متحركة
18	Profession	مهنة
19	Enjoyable	ممتع
20	View	نظرة
21	Interface	واجهة
22	Feels	يشعر
23	Attracts	يجذب
24	Graphics	الرسومية
25	Bright	مشرقة
26	Dark	داكنة
27	Groups	الجماعات
28	Religious	الدينية

Table 4.5: Fault Tolerance Requirements Keywords

#	English keywords	Arabic keywords
1	Exception	استثناء
2	Recovery	استعادة
3	Connection	الاتصال
4	Broadcast	البث
5	Power	الطاقة
6	Disconnection	انقطاع
7	Failure	تعطل
8	Compensatory	تعويضية
9	Preferences	تفضيلات

10	Save	حفظ
11	Server	خادم
12	Error	خطأ
13	Log	سجل
14	Emergency	طارئ
15	Malfunction	عطل
16	Connected	متصل
17	Strength	قوة
18	Robust	متين
19	Problem	مشكلة
20	Displays	يعرض
21	Fails	يفشل
22	Exception	الاستثناء
23	Failure	الفشل

Table 4.6: Legal Requirements Keywords

#	English keywords	Arabic keywords
1	Consideration	اعتبار
2	Parts	الأجزاء
3	Merchant	التاجر
4	License	الترخيص
5	Estimation	التقدير
6	Guidance	التوجيه
7	Returned	المعاد
8	Compliance	امتثال
9	Insurance	تأمين
10	Recycling	تدويرها
11	Legislation	تشريع
12	Estimative	تقديري
13	Record	سجل
14	Rule	قاعدة
15	Law	قانون
16	Regulation	لائحة

17	Principles	مبادئ
18	Requirements	متطلبات
19	Standards	معايير
20	Disputes	منازعات
21	Requires	يتطلب
22	Compliant	متوافقة
23	Organizations	المنظمات
24	Case	قضية

Table 4.7: Operational Requirements Keywords

#	English keywords	Arabic keywords
1	Electronic	الإلكتروني
2	Parts	الأجزاء
3	Devices	الأجهزة
4	Programming	البرمجة
5	Mail	البريد
6	Card	البطاقة
7	Data	البيانات
8	Environment	البيئة
9	Application	التطبيق
10	Server	الخادم
11	Company	الشركة
12	Maintenance	الصيانة
13	Work	العمل
14	Statistics	إحصاء
15	Repair	إصلاح
16	Interaction	تفاعل
17	Technology	تكنولوجيا
18	Distribution	توزيع
19	Timing	توقيت
20	Firewall	جدار الحماية
21	Server	خادم
22	Support	دعم

23	Disk	قرص
24	Sheets	كشوف
25	Accounts	الحسابات
26	Language	لغة
27	Browser	متصفح
28	Compressed	مضغوط
29	Windows	نوافذ
30	Engineering	هندسة
31	Windows	وندوز

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, their evaluations depicted in the Expert Evaluation Table 4.8 below:

Table 4.8: Expert Evaluation

Class	**Heuristic #**	**Evaluation Percentage (Expert #1)**	**Evaluation Percentage (Expert #2)**	**Evaluation Percentage (Expert #3)**	**Average Percentage**
Performance (PE)	1	90	85	80	85
	2	80	90	90	86.7
	3	85	80	80	81.7
Security (SE)	4	95	90	85	90
	5	85	85	80	83.3
	6	95	90	85	90
	7	95	90	85	90
Availability (A)	8	92	85	85	87.3
	9	96	80	85	87
	10	92	80	85	85.6
	11	88	85	90	87.7
Look and Feel (LF)	12	85	80	82	82.3
	13	85	80	90	85
Fault Tolerance (FT)	14	90	85	80	85
Legal (L)	15	90	80	80	83.3
	16	90	80	90	86.7

Operational (O)	17	90	75	80	81.7
	18	80	80	90	83.3
	19	80	80	85	81.6

Leveraging their collective expertise and insights, we meticulously refined our heuristics based on their evaluations, invaluable comments, and constructive feedback. Overall, the evaluation shows our heuristics' advantages as well as their shortcomings. We can prioritize improvements by identifying the particular areas that need attention based on our analysis of these outcomes. For the purpose of enhancing our heuristics and ultimately raising the caliber of our program, this iterative review and refining process is essential.

4.2 Algorithm of the Novel Approach for Classification of Non-Functional Arabic User Requirements

The bellow algorithm presents our heuristic-based requirements classification, which utilizes a set of predefined heuristic rules to predict labels for input sentences. These heuristic rules are designed to capture specific linguistic patterns or conditions indicative of different categories, such as performance, security, availability, look and feel, fault tolerance, legal, and operational aspects.

Algorithm #1: Heuristic-based Non- Functional Arabic User Requirements Classification:

Input:

- Input CSV file containing text data and associated labels (testing.csv)

- Text files containing key phrases related to different categories (e.g., 1.csv, 2.csv, ..., 7.csv, 5ft.csv)

- Pretrained models and modules for tokenization, disambiguation, tagging, and normalization

Output:

- CSV file containing predicted labels ('pe', 'se', 'a', 'lf', 'ft', 'l', 'o') for each input sentence (outFile.csv)

- Classification report showing accuracy, precision, recall, and F1-score for each class based on true and predicted labels

Steps:

1. Import necessary libraries and modules:

2. Initialize MLEDisambiguator and DefaultTagger:

3. Read input data:

4. Initialize output file:

5. Data Preparation:

- Normalize each sentence in the input data.

- Tokenize normalized sentences into words.

- Tag tokens with part-of-speech.

6. Apply Heuristic Rules:

- Define and apply a set of heuristic rules to classify sentences into different categories ('pe', 'se', 'a', 'lf', 'ft', 'l', 'o').

- Each heuristic rule assigns a score to each sentence based on specific conditions and patterns found in tokens or part-of-speech tags.

7. Prediction and Output Writing:

- Use heuristic scores to predict the label (category) for each sentence.

- Write predicted labels and original sentences to the output CSV file.

8. Evaluation:

- Print the heuristic scores for each category and a classification report showing accuracy, precision, recall, and F1-score for each class based on true and predicted labels.

End of Algorithm

4.3 Case Study

In this section, we illustrate the proposed approach and show how we classify software requirements. We used the PROMISE Software Engineering Repository [53] to test our methodology. It is a collection of freely accessible datasets and resources designed to help the software engineering community as a whole and researchers in the process of developing predictive software models (PSMs). The repository is designed to support software engineering prediction models that are repeatable, verifiable, refutable, and/or improvable.

To assess the efficacy of our suggested method, a subset of the PROMISE Software Engineering Repository dataset was used in this investigation. Our methodology was tested and validated using the chosen portion of dataset, which included 105 labeled non- functional user requirement sentences. A qualified translator translated the dataset from English to Arabic to

guarantee accessibility and inclusivity. The portion of the dataset we chose and its translation is shown in the Case Study Table 4.9 bellow:

Table 4.9: Case Study

Class	Requirements in English Language	Translation to Arabic Language
Performance	The product must respond quickly to maintain updated data on the screen.	يجب أن يستجيب المنتج بسرعة للحفاظ على البيانات المحدثة في الشاشة
	The product should produce search results in an acceptable time.	يجب أن ينتج عن المنتج نتائج بحث في وقت مقبول
	Search results should be returned within 30 seconds after the user enters search criteria.	يجب إرجاع نتائج البحث في موعد لا يتجاوز 30 ثانية بعد إدخال المستخدم لمعايير البحث
	The product should generate a CMA report in an acceptable time.	يجب أن يُنشئ المنتج تقرير CMA في وقت مقبول.
	CMD report should be regenerated within 60 seconds after the user enters report criteria.	يجب إعادة تقرير CMD في موعد لا يتجاوز 60 ثانية بعد إدخال المستخدم لمعايير تقرير
	The product should synchronize contacts and appointments in an acceptable time.	يجب أن يقوم المنتج بمزامنة جهات الاتصال والمواعيد في وقت مقبول
	The product should synchronize with the desktop system every hour.	يجب أن يتزامن المنتج مع نظام المكتب كل ساعة
	Response time for general student management tasks should not exceed 5 seconds, and the response time to create the timetable should not exceed 30 seconds.	يجب ألا يستغرق وقت استجابة مهام إدارة الطلاب العامة أكثر من 5 ثوانٍ، ويجب ألا يستغرق وقت الاستجابة لإنشاء الجدول الزمني أكثر من 30 ثانية
	The maximum wait time for a user navigating between screens within the dispute's application should not exceed 5 seconds.	يجب ألا يزيد الحد الأقصى لوقت انتظار المستخدم الذي يتنقل من شاشة إلى أخرى داخل تطبيق المنازعات عن 5 ثوان
	The disputes application should support 350 concurrent users without any performance degradation in the application.	يجب أن يدعم تطبيق المنازعات 350 مستخدمًا متزامنًا دون أي تدهور في الأداء في التطبيق
	Searching for recyclable parts should not take more than 15 seconds; search results should be displayed in less than 15 seconds.	يجب ألا يستغرق البحث عن الأجزاء المعاد تدويرها أكثر من 15 ثانية سيتم عرض نتائج البحث في أقل من 15 ثانية
	Searching for the preferred repair attachment should not take more than 8 seconds; the preferred repair method should be returned within 8 seconds.	لن يستغرق البحث عن مرفق الإصلاح المفضل أكثر من 8 ثوانٍ يتم إرجاع وسيلة الإصلاح المفضلة في غضون 8 ثوان
	The audit report for recyclable parts should be returned to the user within 10 seconds; the audit report should be returned within 10 seconds.	يجب إعادة تقرير تدقيق الأجزاء المعاد تدويرها إلى المستخدم في غضون 10 ثوان يجب إرجاع تقرير التدقيق في غضون 10 ثوان
	Saving preferred repair attachment classifications should occur within 5 seconds; saving should occur within 5 seconds.	يجب حفظ تصنيفات مرفق الإصلاح المفضل في غضون 5 ثوان يجب أن يحدث الحفظ في غضون 5 ثوان

	Modifying inventory quantities for the previous thirty days should not take more than 30 minutes.	يجب ألا يستغرق إعادة تعديل كمية المخزون لفترة الثلاثين يومًا السابقة أكثر من 30 دقيقة.
	The system should allow at least 6 users to work simultaneously.	يجب أن يسمح النظام لـ 6 مستخدمين على الأقل بالعمل في نفس الوقت
	The administrator should be able to activate a prepaid card through the management section in less than 5 seconds.	يجب أن يكون المسؤول قادرًا على تنشيط بطاقة مسبقة الدفع عبر قسم الإدارة في أقل من 5 ثوان
	The customer should be able to verify the status of their prepaid card by entering a PIN number in less than 5 seconds.	يجب أن يكون العميل قادرًا على التحقق من حالة بطاقته المدفوعة مسبقًا عن طريق إدخال رقم PINفي أقل من 5 ثوان
	The system should allow customers to register on the website as 'pay while roaming' users in less than 5 minutes.	يجب أن يسمح النظام للعملاء بالتسجيل على موقع الويب كمستخدم 'الدفع أثناء التنقل' في أقل من 5 دقائق
Security	The product will be able to distinguish between authorized and unauthorized users in all access attempts.	سيكون المنتج قادرًا على التمييز بين المستخدمين المصرح لهم وغير المصرح لهم في جميع محاولات الوصول
	Authentication and licensing of each system user must be performed.	يجب مصادقة وترخيص كل مستخدم للنظام
	The product must prevent the entry of incorrect data.	يجب أن يمنع المنتج من إدخال بيانات غير صحيحة
	The system should include a fundamental data integrity check to reduce the likelihood of submitting incorrect or invalid data.	أن يشتمل النظام على فحص أساسي لتكامل البيانات لتقليل احتمالية تقديم بيانات غير صحيحة أو غير صالحة
	The system must protect private information according to the organization's information policy.	يجب أن يحمي النظام المعلومات الخاصة وفقًا لسياسة معلومات المنظمة
	The system should be built to be as secure as possible against malicious interference.	يجب بناء النظام بحيث يكون آمنًا قدر الإمكان من التدخل الضار
	All additions of new users and modifications to user access must be logged in the user report.	يجب تسجيل جميع الإضافات الخاصة بالمستخدمين الجدد والتعديلات على وصول المستخدم في تقرير المستخدم
	Only the system administrator should be able to reset a canceled user login account.	يجب أن يكون مسؤول النظام فقط قادرًا على إعادة تعيين حساب تسجيل دخول مستخدم تم إلغاؤه
	All updates to data files or databases must start from the dispute system.	يجب بدء جميع التحديثات لملفات البيانات أو قاعدة البيانات من نظام المنازعات
	Only officers can request audits of recyclable parts. Any user without an officer role cannot request audits of recyclable parts.	الضباط فقط يمكنهم طلب تقارير تدقيق الأجزاء المعاد تدويرها. لا يجوز لأي مستخدم ليس لديه دور الضبط أن يطلب عمليات تدقيق للأجزاء المعاد تدويرها
	Only officers with a supervisor role can update preferred repair facility ratings. Users without a supervisor role cannot access preferred repair facility ratings.	الضباط الذين لهم دور مشرف فقط هم من يمكنهم تحديث تصنيفات منشأة الإصلاح المفضلة. لا يمكن للمستخدمين الذين ليس لديهم دور المشرف الوصول إلى تصنيفات منشأة الإصلاح

	Only collision estimators should search for recyclable parts. Users without a collision estimator role cannot search for recyclable parts.	مقدر التصادم فقط يجب أن يبحث عن الأجزاء المعاد تدويرها. لا يجوز للمستخدمين الذين ليس لديهم دور مقدر التصادم الوصول إلى البحث عن الأجزاء المعاد تدويرها
	Only valid data should be entered into the system. No invalid data will be entered into the system.	يجب إدخال البيانات الصالحة فقط في النظام. لن يتم إدخال أي بيانات غير صالحة في النظام
	One insurance company will not be able to view claim data from other insurance companies.	لن تتمكن شركة تأمين واحدة من عرض بيانات مطالبة شركات التأمين الأخرى
	The product must be free from computer viruses.	يجب أن يكون المنتج خاليًا من فيروسات الكمبيوتر
	The system must prevent malicious attacks, including denial of service.	يجب أن يمنع النظام الهجمات الخبيثة بما في ذلك الحرمان من الخدمة
	The product must ensure that only company employees or external users with company-approved user IDs can access the product.	يجب أن يضمن المنتج أن موظفي الشركة أو المستخدمين الخارجيين فقط الذين لديهم معرفات مستخدم معتمدة من الشركة يمكنهم الوصول إلى المنتج
Availability	The product must be available for use 24 hours a day, 365 days a year.	يجب أن يكون المنتج متاحًا للاستخدام 24 ساعة يوميًا 365 يومًا في السنة.
	The system will be available 999 out of 1000 times when accessing the system.	يكون النظام متاحا 999 مرة من بين 1000 وصول إلى النظام.
	The system must achieve a 95% uptime.	يجب أن يحقق النظام وقت تشغيل بنسبة 95%.
	The product must comply with the company's online availability schedule. The application is only brought up during 98% of scheduled downtime periods according to the availability schedule.	يجب أن يلتزم المنتج بجدول التوفر عبر الإنترنت للشركات. يتم إحضار التطبيق فقط في غضون 98٪ من فترات الانقطاع المجدولة وفقًا لجدول التوفر.
	The website must be available for use 24 hours a day, 365 days a year.	يجب أن يكون موقع الويب متاحًا للاستخدام 24 ساعة يوميًا 365 يومًا في السنة.
	The site must achieve 99.5% uptime.	يجب أن يحقق الموقع وقت تشغيل بنسبة 99.5٪.
	All on-demand movies must be available at any time of day.	يجب بث جميع الأفلام عند الطلب في أي وقت من اليوم.
	The system will provide 800 toll-free numbers 24 hours a day for customer support.	سيوفر النظام 800 رقم مجاني على مدار 24 ساعة لدعم عملائه.
Look and Feel	The application must match the color scheme specified by the Ministry of Internal Security.	يجب أن يتطابق التطبيق مع لون المخطط المحدد من قبل وزارة الأمن الداخلي
	The system's form and appearance must comply with the smart device's user interface standards.	يجب أن يتوافق شكل ومظهر النظام مع معايير واجهة المستخدم الخاصة بالجهاز الذكي
	The user interface must have standard menu buttons for navigation.	يجب أن تحتوي واجهة المستخدم على أزرار قوائم قياسية للتنقل
	The system must have a professional appearance.	يجب أن يكون للنظام مظهر احترافي
	The product must have consistent color schemes and fonts.	يجب أن يكون للمنتج مخطط ألوان وخطوط متناسقة

	The dispute application must comply with the company's standards for creating an internal and external application user interface.	يجب أن يتوافق تطبيق المنازعات مع معايير الشركة لإنشاء واجهة مستخدم للتطبيقات المستخدمة داخليًا وخارجيًا
	All screens created as part of the dispute application must comply with the company's interface creation standards.	يجب أن تتوافق جميع الشاشات التي تم إنشاؤها كجزء من تطبيق النزاعات مع معايير الشركة لإنشاء الواجهة
	The product must comply with company user interface guidelines.	يجب أن يتوافق المنتج مع إرشادات واجهة المستخدم الخاصة بالشركات
	The product must comply with the company's color scheme.	يجب أن يتوافق المنتج مع نظام ألوان الشركة
	The product's appearance must be professional.	يجب أن يظهر مظهر المنتج احترافيًا
	The product's form and appearance should be able to incorporate aspects of the client's enterprise, such as branding, logo, and identity.	يجب أن يكون شكل المنتج وشكله قادرين على دمج جوانب مؤسسة العميل مثل العلامة التجارية والشعار والهوية
	The product must have a conservative and professional appearance.	يجب أن يكون للمنتج مظهر محافظ ومهني
	The website must be appealing to all audiences. It should appear enjoyable with bright and lively colors.	يجب أن يكون الموقع جذابًا لجميع الجماهير. يجب أن يبدو الموقع ممتعًا ويجب أن تكون الألوان مشرقة ونابضة بالحياة
	The design of the website should be modern, clean, and concise.	يجب أن يكون تصميم الموقع حديثًا ونظيفًا وموجزًا
	The website must not discriminate against religious or ethnic groups.	يجب ألا يسيء الموقع إلى الجماعات الدينية أو العرقية
	The website should attract all Africans, not just Nigerians.	يجب أن يجذب الموقع جميع الأفارقة وليس النيجيريين فقط
	The product should emulate the appearance of ships at sea.	يجب أن يحاكي المنتج مظهر السفن في البحر
	The product should display networks within a circle as if viewed through a periscope.	يجب أن يعرض المنتج الشبكات داخل دائرة كعرض من المنظار
	The product should showcase each type of ship in a network using an image of a specific type of ship.	يجب أن يعرض المنتج كل نوع من السفن في شبكة باستخدام صورة لنوع معين من السفن
	When the attacking player takes a shot, the product should mimic the sound of a ship at sea.	عندما يأخذ اللاعب المهاجم لقطة، يجب أن يحاكي المنتج صوت سفينة في البحر
Fault Tolerance	The product must operate in offline mode when internet connection is unavailable.	يجب أن يعمل المنتج في وضع عدم الاتصال عندما يكون الاتصال بالإنترنت غير متوفر
	he product should allow the user to view reports and schedules of previously downloaded search results.	يجب أن يسمح المنتج للمستخدم بعرض تقارير ومواعيد نتائج البحث التي تم تنزيلها مسبقًا
	The product must retain user preferences in case of a malfunction.	يجب أن يحتفظ المنتج بتفضيلات المستخدم في حالة حدوث عطل

	100% of the user preferences saved must be restored upon system return to an internet-connected state.	يجب استعادة 100٪ من تفضيلات المستخدم المحفوظة عند عودة النظام متصل بالإنترنت
	The product must create an exception log for issues encountered within the product to be sent to our company for analysis and resolution.	يجب على المنتج إنشاء سجل استثناء للمشكلات التي تمت مواجهتها داخل المنتج لإرساله إلى شركتنا لتحليلها وحلها
	The website must continue to function if the streaming server fails.	يجب أن يستمر موقع الويب في العمل إذا تعطل خادم البث
	The website must continue to function if the payment gateway fails.	يجب أن يستمر الموقع في العمل إذا تعطلت بوابة الدفع
	The product must create an exception log for issues encountered within the product to be sent to our company for analysis and resolution.	يجب على المنتج إنشاء سجل استثناء للمشكلات التي تمت مواجهتها داخل المنتج لإرساله إلى شركتنا لتحليلها وحلها
	The product must be robust, and error avoidance should be based on standards compliance.	يجب أن يكون المنتج متينًا، ويجب أن يكون المنتج متجنبًا للخطأ بناءً على اعتماد المعايير
	The product should be strong with error tolerance; it should be fault-tolerant using compensatory transaction handling for recovery and routing around failure scenarios.	يجب أن يكون المنتج قويًا مع التسامح مع الخطأ، يجب أن يكون المنتج متسامحا مع الخطأ باستخدام المعاملة التعويضية لتقنية الاسترداد والتوجيه حول حالات الفشل
Legal	The dispute resolution application must comply with the legal requirements as specified in the Merchant Operating Regulations.	يجب أن يتوافق تطبيق المنازعات مع المتطلبات القانونية على النحو المحدد في لوائح تشغيل التاجر
	All operational rules specified in the dispute resolution system must be compliant with the Merchant Operating Regulations.	يجب أن تكون جميع قواعد العمل المحددة في نظام المنازعات متوافقة مع لوائح تشغيل التاجر
	The dispute resolution application must adhere to the legal requirements as specified in Regulations E and Regulation Z governing credit card dispute processing.	يجب أن يتوافق تطبيق المنازعات مع المتطلبات القانونية على النحو المحدد في اللوائح E واللائحة Zالتي تحكم معالجة منازعات بطاقات الائتمان
	All operational rules specified in the dispute resolution system must be compliant with the guidance principles of Regulations E and Regulation Z.	يجب أن تكون جميع قواعد العمل المحددة في نظام المنازعات متوافقة مع المبادئ التوجيهية للائحة E واللائحةZ
	The dispute application must maintain a detailed record of every action taken by the user in a dispute case.	يجب أن يحتفظ تطبيق النزاعات بسجل مفصل لكل إجراء يتخذه المستخدم في قضية النزاع.
	All actions modifying an existing dispute case must be logged in the case's history.	يجب تسجيل جميع الإجراءات التي تعدل قضية نزاع قائمة في تاريخ القضية
	The product must comply with laws regarding the use of recycled parts.	يجب أن يتوافق المنتج مع قوانين التقدير المتعلقة باستخدام الأجزاء المعاد تدويرها

Operational	The system must be able to operate within a typical office environment for the nursing department at DePaul University.	يجب أن يكون النظام قادرًا على العمل داخل بيئة مكتب أعمال نموذجية لقسم التمريض في جامعة ديبول
	The system must adhere to the specifications set by the computers used by program managers/nursing staff members.	يجب استخدام النظام ضمن المواصفات المحددة بواسطة أجهزة الكمبيوتر المستخدمة من قبل مديري البرنامج / أعضاء طاقم التمريض
	The system must operate within the Windows XP Professional operating system.	يجب أن يعمل النظام ضمن نظام التشغيل Windows XP Professional
	The system must interact with the central server for CampusConnect.	يجب أن يتفاعل النظام مع الخادم المركزي لـ CampusConnect
	The system must interact with the college's central server.	يجب أن يتفاعل النظام مع الخادم المركزي للكلية
	The system must interact with the main student server.	يجب أن يتفاعل النظام مع الخادم الرئيسي للطالب
	The dispute application must be available 24/7, except during scheduled maintenance windows: Monday - Saturday 3:00 AM to 4:00 AM Eastern Time, Sunday 1:00 AM to 5:00 AM Eastern Standard Time.	يجب أن يكون تطبيق النزاعات متاحًا على مدار 24 ساعة طوال أيام الأسبوع باستثناء نوافذ الصيانة المجدولة التالية: الاثنين - السبت 3:00 صباحًا حتى 4:00 صباحًا بالتوقيت الشرقي، الأحد 1:00 صباحًا إلى 5:00 صباحًا بالتوقيت الرسمي الشرقي
	The dispute application must interact with a data account database. The account database provides transaction details for the dispute system. All transaction details must be obtained from the account database.	يجب أن يتفاعل تطبيق المنازعات مع قاعدة بيانات البيانات. توفر قاعدة بيانات كشوف الحسابات تفاصيل المعاملة لنظام المنازعات. يجب الحصول على جميع تفاصيل المعاملة من قاعدة بيانات كشوف الحسابات
	The dispute application must interact with a card member information database. The card member information database provides detailed information regarding the card member. All detailed card member information must be obtained from the card member information database.	يجب أن يتفاعل تطبيق المنازعات مع قاعدة بيانات معلومات عضو البطاقة. توفر قاعدة بيانات معلومات Cardmember معلومات مفصلة فيما يتعلق بعضو البطاقة. يجب الحصول على جميع المعلومات التفصيلية الخاصة بعضو البطاقة من قاعدة بيانات معلومات عضو البطاقة
	The dispute application must interact with a merchant information database. The merchant information database provides detailed information regarding the merchant. All merchant detail information must be obtained from the merchant information database.	يجب أن تتفاعل تطبيقات النزاعات مع قاعدة بيانات معلومات التاجر. توفر قاعدة بيانات التاجر معلومات مفصلة فيما يتعلق بالتاجر. يجب الحصول على جميع معلومات تفاصيل التاجر من قاعدة بيانات التاجر
	The dispute application must interact with the Letters application. This allows the dispute application to request letters as part of the dispute initiation and follow-up process. All letter requests must be sent to the Print Letter Utility application.	يجب أن يتفاعل تطبيق المنازعات مع تطبيق Lettersسيسمح هذا لتطبيق المنازعات بطلب خطابات كجزء من عملية بدء النزاع ومتابعة النزاع. يجب إرسال جميع طلبات الخطابات إلى تطبيق Print Letter Utility

	The dispute application must interact with the card member migration and billing system. This allows the dispute application to request modifications to card member and merchant accounts. All modification requests must be sent to the card member migration and billing system.	يجب أن يتفاعل تطبيق المنازعات مع نظام الترحيل والفوترة لعضو البطاقة. سيسمح هذا لتطبيق المنازعات بطلب تعديلات على حسابات صاحب البطاقة والتاجر. يجب إرسال جميع طلبات التعديل إلى نظام الترحيل والفوترة لعضو البطاقة
	The product must interact with the parts selection system. This provides recycled parts data feed.	يجب أن يتفاعل المنتج مع نظام اختيار الأجزاء يوفر هذا تغذية بيانات الأجزاء المعاد تدويرها
	The product must adhere to company engineering guidelines.	يجب أن يلتزم المنتج بإرشادات هندسة الشركة
	For estimators, the product must be able to operate in a repair facility during dirty and noisy conditions.	بالنسبة إلى المقدرين، يجب أن يكون المنتج قادرًا على العمل في منشأة إصلاح أثناء الظروف المتسخة والصاخبة
	The product must interact with the once daily CoiceParts system around 1:00 AM.	يجب أن يتفاعل المنتج مع نظام مرة واحدة يوميًا في حوالي الساعة 1:00 صباحًاCoiceParts
	The product will be available for licensing as a single server and five servers or five servers or more.	لمنتج سيكون متاحًا للترخيص كخادم واحد وخمسة خوادم أو خمسة خوادم أو أكثر
	The product must be installable in any operating environment within two days.	جب أن يكون المنتج قادرًا على التثبيت في أي بيئة تشغيل في غضون يومين
	The product must be developed using J2SE/J2EE programming language libraries.	يجب تطوير المنتج باستخدام مكتبات لغة البرمجةJ2SE / J2EE
	The system must utilize currently owned computers.	يجب أن يستخدم النظام أجهزة الكمبيوتر المملوكة حاليًا

Upon inputting these requirement sentences into CAMeL Tools, the resulting output will encompass both PoS tags and tokens. An illustrative example of each class will be provided below

Arabic sentence:

"يجب ارجاع نتائج البحث في موعد لا يتجاوز 30 ثانية بعد إدخال المستخدم لمعايير البحث"

CAMeL Tokens: ['يجب', 'إرجاع', 'نتائج', 'البحث', 'في', 'موعد', 'لا', 'يتجاوز', '30', 'ثانية', 'بعد', 'إدخال', 'المستخدم', 'لمعايير', 'البحث"]

CAMeL Tokens: ['verb', 'noun', 'noun', 'noun', 'prep', 'noun', 'part_neg', 'verb', 'digit', 'adj', 'noun', 'noun', 'noun', 'noun', 'noun']

Arabic sentence:

"سيكون المنتج قادرًا على التمييز بين المستخدمين المصرح لهم وغير المصرح لهم في جميع محاولات الوصول"

CAMeL Tokens: ['سيكون', 'المنتج', 'قادرًا', 'على', 'التمييز', 'بين', 'المستخدمين', 'المصرح', 'لهم', 'وغير', 'المصرح', 'لهم', 'في', 'جميع', 'محاولات', 'الوصول']

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

Arabic sentence:

"سيوفر النظام 800 رقم مجاني على مدار 24 ساعة لدعم عملائه"

CAMeL Tokens: ['سيوفر', 'النظام', '800', 'رقم', 'مجاني', 'على', 'مدار', '24', 'ساعة', 'لدعم', 'عملائه']

CAMeL PoS: ['verb', 'noun', 'digit', 'noun', 'adj', 'prep', 'noun', 'digit', 'noun', 'noun', 'noun']

Arabic sentence:

"يجب أن يكون للمنتج مخطط ألوان وخطوط متناسقة"

CAMeL Tokens: ['يجب', 'أن', 'يكون', 'للمنتج', 'مخطط', 'ألوان', 'وخطوط', 'متناسقة']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'noun', 'noun', 'noun', 'adj']

5. Arabic sentence:

"يجب على المنتج إنشاء سجل استثناء للمشكلات التي تمت مواجهتها داخل المنتج لإرساله إلى شركتنا لتحليلها وحلها"

CAMeL Tokens: ['يجب', 'على', 'المنتج', 'إنشاء', 'سجل', 'استثناء', 'للمشكلات', 'التي', 'تمت', 'مواجهتها', 'داخل', 'المنتج', 'لإرساله', 'إلى', 'شركتنا', 'لتحليلها', 'وحلها']

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

6. Arabic sentence:

"يجب أن يحتفظ تطبيق النزاعات بسجل مفصل لكل إجراء يتخذه المستخدم في قضية النزاع"

CAMeL Tokens: ['يجب', 'أن', 'يحتفظ', 'تطبيق', 'النزاعات', 'بسجل', 'مفصل', 'لكل', 'إجراء', 'يتخذه', 'المستخدم', 'في', 'قضية', 'النزاع']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'noun', 'noun', 'adj', 'noun', 'noun', 'verb', 'noun', 'prep', 'noun', 'noun']

7. Arabic sentence:

CampusConnect ”يجب أن يتفاعل النظام مع الخادم المركزي لـ“

CAMeL Tokens: ['CampusConnect', 'لـ', 'المركزي', 'الخادم', 'مع', 'النظام', 'يتفاعل', 'أن', 'يجب']

CAMeL PoS: ['verb', 'conj_sub', 'verb', 'noun', 'prep', 'noun', 'adj', 'prep', 'foreign']

CHAPTER FIVE
EVALUATION

In this chapter, we go into the critical process of evaluating our approach performance. Evaluation serves as the compass guiding the effectiveness and reliability of the approach, we developed. Through rigorous assessment, we gain insights into how well our approach generalize to unseen data, their strengths, and their limitations. By employing various evaluation metrics, we can quantify and interpret the performance of our approach.

5.1 Evaluation Metrics

Evaluation metrics offer numerical measurements for evaluating approach performance. They provide insights into several facets of an approach's functionality, including its precision in predicting outcomes, its ability to locate pertinent data, and its general efficacy in resolving the intended issue. In our study, the evaluation of classification models was conducted utilizing the classification report functionality provided by the Sklearn.metrics library within the Python programming environment. The metrics included in the classification report are: Precision, Recall, Accuracy, and F1-Score.

5.1.1 Precision

Precision, also known as positive predictive value, measures the accuracy of positive predictions made by the approach. It quantifies the proportion of true positive predictions (correctly identified instances of a class) out of all positive predictions made, including both true positives and false positives. Mathematically, precision is calculated as:

$$Precision = \frac{TP}{TP + FP} \ldots\ldots (5.1)$$

A high precision indicates that our approach tends to make accurate positive predictions and has a low rate of false positives, which is crucial in tasks where the cost of false positives is high.

5.1.2 Recall

Recall, also known as sensitivity or true positive rate, measures the approach's ability to correctly identify all positive instances, including both true positives and false negatives. It quantifies the proportion of true positive predictions out of all actual positive instances in the data. Mathematically, recall is calculated as:

$$Recall = \frac{TP}{TP+FN} \ldots\ldots (5.2)$$

A high recall indicates that the approach captures a large proportion of positive instances, minimizing the number of false negatives, which is crucial in tasks where identifying all positive instances is essential, even at the cost of some false positives.

5.1.3 Accuracy

Accuracy measures the overall correctness of the approach's predictions across all classes. It quantifies the proportion of correctly classified instances (both true positives and true negatives) out of the total instances evaluated. Mathematically, accuracy is calculated as:

$$Accuracy = \frac{TP + TN}{TP + FP + FN + TN} \dots\dots (5.3)$$

Accuracy provides a general assessment of the approach's performance.

5.1.4 F1-Score

The harmonic mean of recall and precision is what determines an F1-score. The F1-score, which combines precision and recall into a single metric for evaluation purposes in binary and multi-class classification, is frequently used to improve comprehension of model performance.

$$F1 - Score = \frac{2 * Precision * Recall}{Precision + Recall} \dots\dots (5.4)$$

5.2 Experiments

In this section, we detail the experimental setups and the results obtained from evaluating our proposed approach for classifying non-functional Arabic user requirements using a set of heuristics.

5.2.1 Experimental Setup

We used 105 requirement sentences from the PROMISE dataset, which was divided into seven different classes: legal, operational, performance, security, availability, look and feel, and fault tolerance, which tested on our software.

5.2.2 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.

5.2.3 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.

5.3 Result Analysis

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

5.3.1 Single-Class Testing

The Single- Class Testing Results are shown in Table 5.1, giving 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 total accuracy are all summarized in this table.

Table 5.1: 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

The performance class shows a high precision score of 1.00, indicating that when the model predicts an input sentence as related to performance, it is almost always correct. However, the recall score of 0.84 suggests that the model missed some relevant sentences related to

performance. Overall, the F1-score and accuracy are also quite high but could be improved by increasing recall.

Additionally, the security class shows a minimal false positive rate with a high precision score of 1.00. Nonetheless, it appears from the recall score of 0.82 that the model did not accurately identify all lines linked to security. Although the accuracy and F1-score are often rather excellent, recall might be raised.

With a precision score of 1.00 for the availability class, there are no false positives. The majority of the availability-related statements were accurately detected by the model, according to the recall score of 0.88. While still quite high overall, the F1-score and accuracy could be raised by sharpening their focus. The look and feel class exhibit high accuracy, recall, F1-score, and precision, demonstrating how well the model identified phrases pertaining to look and feel. High precision, recall, F1-score, and accuracy are also displayed by the fault tolerance class, indicating that the model did a good job of detecting sentences that were linked to fault tolerance.

The legal class has a lower F1-score due to their high precision but significantly lower recall. This suggests that although the model accurately recognized statements with a legal theme. The operational class has a lower F1-score due to its high precision but significantly lower recall. This suggests that although the model detected sentences related to operations accurately, it failed to identify certain other relevant sentences. To summarize;

1. There is a low false positive rate in every category, as seen by the consistently high precision ratings for all classes.
2. The recall scores vary across classes, with some classes showing higher recall rates than others.
3. All classes have typically high F1-scores and accuracies, showing that the model performs well overall in categorizing requirements into the appropriate categories.
4. To improve the total F1-score and accuracy, more relevant sentences could be captured by raising recall for classes with lower recall scores. Further improving classification performance may involve fine-tuning the model and possibly expanding and diversifying the training set.

5.3.2 Multi- Class Testing

The Multi-Class Testing Results are shown in Table 5.2, 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 5.2: 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**

In the initial script iteration, our algorithm was designed to increment a category score by one if there was a match between the sentence's tokens and the category's table of keywords. The initial results yielded a satisfactory accuracy of 67%. However, upon consultation with experts, they recommended enhancing the code by not only adding one to the category score in the case of a match but also determining the number of matches and incorporating that count into the score. This modification proved to be a significant improvement, resulting in a new overall accuracy of 88% as shown in the classification report for all seven classes of our case study.

In summary, although the system performs well overall (88% accuracy), there are differences in precision, recall, and f1-score amongst classes. This study shows the model's strong points and potential areas for modification to increase overall classification performance.

CHAPTER SIX
CONCLUSION AND FUTURE WORKS

6.1 Conclusion

In summary, this thesis has centered on the semi-automated categorization of Arabic user non-functional requirements into seven categories using natural language processing tools called CAMeL tools. We have devised a series of strategies to effectively classify these requirements, utilizing the linguistic components generated by CAMeL tools, including tokens, Part of Speech (PoS) tags, and lemmas.

Our approach, implemented through Python code and the CAMeL tools, offers a practical solution tailored for software engineers tasked with handling Arabic user non- functional requirements. By automating aspects of the classification process, our research aims to streamline the analysis phase, thereby reducing the time and resources traditionally required for manual classification.

Through the enhancement of efficiency and accuracy in user non- functional requirements analysis, our methodology equips software engineers with the tools to make well-informed decisions, ultimately resulting in the delivery of higher-quality software products.

Moreover, our research plays a pivotal role in advancing the field of natural language processing (NLP) specifically tailored to the Arabic language. By addressing the unique linguistic nuances and challenges inherent to Arabic, we contribute to the expansion and refinement of NLP capabilities in this domain. This, in turn, lays the foundation for future innovations and developments in Arabic NLP, opening up new avenues for research and application across various industries and sectors.

Overall, this project not only improves the software development process but also drives progress in the broader field of Arabic natural language processing, offering far-reaching benefits for both technological advancement and societal impact.

Ultimately, this project serves as a crucial step towards optimizing software development practices and enhancing the overall quality of software products in Arabic-speaking regions.

6.2 Future Work

Some directions for further research and development are presented in the future work section of our thesis on semi-automated classification of non-functional Arabic user requirements:

1. Heuristic refinement: Future research should focus on improving the heuristics created in this thesis in order to increase classification accuracy. This could entail adding more language features, investigating different classification strategies, or optimizing the heuristics' rules and criteria.
2. Data Collection Expansion: In order to confirm and improve our methodology, we need to collect a larger and more varied dataset of Arabic-language SRS. This can entail contacting more organizations or software providers in order to secure more SRS projects in Arabic.
3. Testing on More Case Studies: Future study should test our method on a wider variety of case studies in order to evaluate its robustness and generalizability. This might comprise SRS documents from different sectors and disciplines to assess how well our classification algorithm works in diverse scenarios.
4. Creating Huge Datasets: Work should be done to create sizable datasets of user requirements that are written in Arabic in cooperation with user requirements researchers. These datasets would be very helpful in creating and educating new completely automated categorization systems. They could also be utilized to implement machine learning techniques to raise the efficiency and accuracy of classification.

Our goal in exploring these areas further is to improve the current state of the art in semi-automated classification of non-functional Arabic user requirements. This will ultimately aid in the creation of more precise and effective software development processes in Arabic-speaking environments.

Bibliography

[1] Mairiza, D., Zowghi, D., & Nurmuliani, N. (2010, March). An investigation into the notion of non-functional requirements. In Proceedings of the 2010 ACM symposium on applied computing (pp. 311-317).u

[2] Habibullah, K. M., Gay, G., & Horkoff, J. (2023). Non-functional requirements for machine learning: Understanding current use and challenges among practitioners. *Requirements Engineering*, *28*(2), 283-316.

[3] Kopczyńska, S., Ochodek, M., & Nawrocki, J. (2020). On importance of non-functional requirements in agile software projects—a survey. Integrating Research and Practice in Software Engineering, 145-158.

[4] R. Amro, A. Althunibat and B. Hawashin, "Arabic Non-Functional Requirements Extraction Using Machine Learning," 2023 International Conference on Information Technology (ICIT), Amman, Jordan, 2023, pp. 489-494, doi: 10.1109/ICIT58056.2023.10225951.

[5] Engström, E., Storey, M. A., Runeson, P., Höst, M., & Baldassarre, M. T. (2020). How software engineering research aligns with design science: a review. *Empirical Software Engineering*, *25*, 2630-2660.

[6] Chowdhary, K., & Chowdhary, K. R. (2020). Natural language processing. *Fundamentals of artificial intelligence*, 603-649.

[7] Ali, A., Chowdhury, S., Afify, M., El-Hajj, W., Hajj, H., Abbas, M., ... & Alqudah, A. (2021). Connecting Arabs: Bridging the gap in dialectal speech recognition. *Communications of the ACM*, *64*(4), 124-129.

[8] Younas, M., Jawawi, D. N., Ghani, I., & Shah, M. A. (2020). Extraction of non-functional requirement using semantic similarity distance. *Neural Computing and Applications*, *32*, 7383-7397.

[9] Behutiye, W., Karhapää, P., Costal, D., Oivo, M., & Franch, X. (2017). Non-functional Requirements Documentation in Agile Software Development: Challenges and Solution Proposal. Product-Focused Software Process Improvement. Springer. doi: 10.1007/978-3-319-69926-4_41

[10] Iftikhar, K., Ali, S., & Ngadi, M. A. (2016). Enhancement of Non Functional Requirements in Agile Software Development. *International Journal of Computer Science and Information Security*, *14*(12), 820.

[11] Adetoba, B., & Ogundele, I. (2018). Requirements engineering techniques in software development life cycle methods: A systematic literature review. *International Journal of Advanced Research in Computer Engineering & Technology*, *7*(10), 733-743.

[12] BATOOL, I., KOSAR, L., & MEHMOOD, M. NON-FUNCTIONAL REQUIREMENTS AS CONSTRAINTS AND THEIR VALUES IN SOFTWARE DEVELOPMENT: A REVIEW. 2018.

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

[14] Wang, X., Zhao, L., Wang, Y., & Sun, J. (2014). The role of requirements engineering practices in agile development: an empirical study. In *Requirements Engineering: First Asia Pacific Requirements Engineering Symposium, APRES 2014, Auckland, New Zealand, April 28-29, 2014. Proceedings* (pp. 195-209). Springer Berlin Heidelberg.

[15] Daun, M., Grubb, A. M., Stenkova, V., & Tenbergen, B. (2023). A systematic literature review of requirements engineering education. *Requirements Engineering*, *28*(2), 145-175.

[16] Pandey, D., & Pandey, V. (2012). Requirement Engineering: An Approach to Quality Software Development. *Journal of Global Research in Computer Science*, *3*(9), 31-33.

[17] Adetoba, B., & Ogundele, I. (2018). Requirements engineering techniques in software development life cycle methods: A systematic literature review. *International Journal of Advanced Research in Computer Engineering & Technology*, *7*(10), 733-743.

[18] AlSanad, A., & Chikh, A. (2014). Reengineering of Software Requirement Specification. New Perspectives in Information Systems and Technologies, Volume 2. Springer. doi: 10.1007/978-3-319-05948-8_10

[19] Singh, P., Singh, D., & Sharma, A. (2016, December). Classification of non-functional requirements from SRS documents using thematic roles. In *2016 IEEE International Symposium on Nanoelectronic and Information Systems (iNIS)* (pp. 206-207). IEEE.

[20] Tayefeh Hashemi, S., Ebadati, O. M., & Kaur, H. (2020). Cost estimation and prediction in construction projects: A systematic review on machine learning techniques. *SN Applied Sciences*, *2*, 1-27.

[21] Choi, S. S., Chae, S. Y., & Lee, G. S. (2005, May). SRS-tool: A security functional requirement specification development tool for application information system of organization. In *International Conference on Computational Science and Its Applications* (pp. 458-467). Berlin, Heidelberg: Springer Berlin Heidelberg.

[22] Abdeen, W., Chen, X., & Unterkalmsteiner, M. (2023). An approach for performance requirements verification and test environments generation. *Requirements Engineering*, *28*(1), 117-144.

[23] Navita, M. (2017). A Study on Software Development Life Cycle & its Model. *International Journal of Engineering Research in Computer Science and Engineering (IJERCSE)*, *4*, 9.

[24] Hussain, S., Asghar, M. Z., Ahmad, B., & Ahmad, S. (2009). A step towards software corrective maintenance using RCM model. *arXiv preprint arXiv:0909.0732*.

[25] Budake, R., Bhoite, S., & Kharade, K. (2023). Identification and classification of functional and non-functional software requirements using machine learning. AIP Conf. Proc., 2946(1). doi: 10.1063/5.0178116

[26] Handa, N., Sharma, A., & Gupta, A. (2022). Framework for prediction and classification of non functional requirements: a novel vision. *Cluster Computing*, *25*(2), 1155-1173.

[27] Maiden, N. (2008). User requirements and system requirements. *IEEE Software*, *25*(2), 90-91..

[28] Coughlan, J., & Macredie, R. D. (2002). Effective communication in requirements elicitation: a comparison of methodologies. *Requirements Engineering*, *7*, 47-60.

[29] Benfell, A. (2021). Modeling functional requirements using tacit knowledge: a design science research methodology informed approach. *Requirements engineering*, *26*(1), 25-42.

[30] Maguire, M., & Bevan, N. (2002, August). User requirements analysis: a review of supporting methods. In *IFIP World Computer Congress, TC 13* (pp. 133-148). Boston, MA: Springer US.

[31] Burek, P. (2008). Creating clear project requirements: differentiating "what" from "how" Paper presented at PMI® Global Congress 2008—North America, Denver, CO. Newtown Square, PA: Project Management Institute.

[32] Serrador, P. (2012). The importance of the planning phase to project success. Paper presented at PMI® Global Congress 2012—North America, Vancouver, British Columbia, Canada. Newtown Square, PA: Project Management Institute.

[33] E. Gottesdiener, "Requirements by collaboration: getting it right the first time," in IEEE Software, vol. 20, no. 2, pp. 52-55, March-April 2003, doi: 10.1109/MS.2003.1184167.

[34] Tukur, M., Umar, S., & Hassine, J. (2021). Requirement engineering challenges: A systematic mapping study on the academic and the industrial perspective. *Arabian Journal for Science and Engineering*, *46*, 3723-3748.

[35] B. Alsawareah, A. Althunibat and B. Hawashin, "Classification of Arabic Software Requirements Using Machine Learning Techniques," 2023 International Conference on Information Technology (ICIT), Amman, Jordan, 2023, pp. 631-636, doi: 10.1109/ICIT58056.2023.10225789.

[36] Mishra, A., & Mishra, D. (2014). Cultural issues in distributed software development: A review. In *On the Move to Meaningful Internet Systems: OTM 2014 Workshops: Confederated International Workshops: OTM Academy, OTM Industry Case Studies Program, C&TC, EI2N, INBAST, ISDE, META4eS, MSC and OnToContent 2014, Amantea, Italy, October 27-31, 2014. Proceedings* (pp. 448-456). Springer Berlin Heidelberg.

[37] 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.

[38] Wahdan, A., Al-Emran, M., & Shaalan, K. (2023). A systematic review of Arabic text classification: areas, applications, and future directions. *Soft Computing*, 1-22.

[39] 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).

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

[41] Shreda, Q. A., & Hanani, A. A. (2021). Identifying non-functional requirements from unconstrained documents using natural language processing and machine learning approaches. *IEEE Access*.

[42] Z. Kurtanović and W. Maalej, "Automatically Classifying Functional and Non-functional Requirements Using Supervised Machine Learning," 2017 IEEE 25th International Requirements Engineering Conference (RE), Lisbon, Portugal, 2017, pp. 490-495, doi: 10.1109/RE.2017.82.

[43] Yahya, A. E., Gharbi, A., Yafooz, W. M. S., & Al-Dhaqm, A. (2023). A Novel Hybrid Deep Learning Model for Detecting and Classifying Non-Functional

Requirements of Mobile Apps Issues. Electronics, 12(5), 1258. doi: 10.3390/electronics12051258

[44] Jindal, R., Malhotra, R., Jain, A., & Bansal, A. (2021). Mining Non-Functional Requirements using Machine Learning Techniques. *e-Informatica Software Engineering Journal*, *15*(1).

[45] Kumar, M. S., & Harika, A. (2020). Extraction and classification of non-functional requirements from text files: a supervised learning approach. *Psychology and Education*, *57*(9), 4120-4123.

[46] Hussain, I., Kosseim, L., & Ormandjieva, O. (2008). Using linguistic knowledge to classify non-functional requirements in SRS documents. In *Natural Language and Information Systems: 13th International Conference on Applications of Natural Language to Information Systems, NLDB 2008 London, UK, June 24-27, 2008 Proceedings 13* (pp. 287-298). Springer Berlin Heidelberg.

[47] Cleland-Huang, J., Settimi, R., Zou, X., & Solc, P. (2006, September). The detection and classification of non-functional requirements with application to early aspects. In *14th IEEE International Requirements Engineering Conference (RE'06)* (pp. 39-48). IEEE.

[48] Shehadeh, K., Arman, N., & Khamayseh, F. (2021, July). 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.

[49] Arman, N., & Jabbarin, S. (2015). Generating use case models from Arabic user requirements in a semiautomated approach using a natural language processing tool. *Journal of Intelligent Systems*, *24*(2), 277-286.

[50] 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.

[51] Arman, N. (2015). Using MADA+ TOKAN to Generate Use Case Models from Arabic User Requirements in a Semi-Automated Approach. ICIT 2015 The 7th International Conference on Information Technology.

[52] Alami, N., Arman, N., & Khamyseh, 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

[53] 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.

Appendix A

Code

```
from camel_tools.tokenizers.word import simple_word_tokenize
from camel_tools.disambig.mle import MLEDisambiguator
from camel_tools.tagger.default import DefaultTagger
from camel_tools.utils.normalize import normalize_alef_maksura_ar
from camel_tools.utils.normalize import normalize_alef_ar
from camel_tools.utils.normalize import normalize_teh_marbuta_ar
import pandas as pd
import csv
from sklearn.metrics import classification_report
mle = MLEDisambiguator.pretrained()
tagger = DefaultTagger(mle, 'pos')
infile = pd.read_csv('case1.csv',encoding='utf-8',quotechar='"')
outfile = open('outFile.csv', 'w', encoding='utf-8' ,newline='')


true = []
pred = []
sentences = []
tokens = []
pos_tags=[]
NFRKey = []
Stype = []
Data = []
lemmas = []
true = infile["T"].values.tolist()
sentences = infile["S"].values.tolist()
F_CF = [0]*(len(sentences))
NF_CF = [0]* (len(sentences))
F_Score = [0]*(len(sentences))
NF_Score = [0]* (len(sentences))
c1 = 0
```

```
c2 = 0
c3 = 0
c4 = 0
c5 = 0
c6 = 0
c8 = 0
for i in range(0, len(sentences)):
    # Normalize alef variants to 'ا'
    sentences[i] = normalize_alef_ar(sentences[i])
    # Normalize alef maksura 'ى' to yeh 'ي'
    sentences[i] = normalize_alef_maksura_ar(sentences[i])

    # Normalize teh marbuta 'ة' to heh 'ه'
    sentences[i] = normalize_teh_marbuta_ar(sentences[i])
    # pre-tokenized text
    tokens.append(simple_word_tokenize(sentences[i]))
    # POS tagger of tokens
    pos_tags.append(tagger.tag(tokens[i]))
#********************H1********************
DigPOS= ['digit', 'noun_num']
for lst in pos_tags:
    if any(item in lst for item in DigPOS):
        NF_CF[c1] += 78.33
        NF_Score[c1] +=1
           c1+=1
#********************H2********************

    star = ['foreign']
for lst in pos_tags:
    if any(item in lst for item in star):
        NF_CF[c2] += 80
        NF_Score[c2] +=1
```

```
    c2 +=1
#********************H3********************
AdjPOS =['adj' ,'adv']
for lst in pos_tags:
    if any(item in lst for item in AdjPOS):
        NF_CF[c3] += 86
        NF_Score[c3] +=1
    c3 += 1
#********************H4********************
with open('NFRKey.txt', 'r' ,encoding='utf-8') as filehandle:
    for line in filehandle:
    # remove linebreak which is the last character of the string
        currentPlace = line[:-1]
    # add item to the list
        NFRKey.append(currentPlace)


for i in range(0, len(NFRKey)):
        NFRKey[i] = normalize_alef_ar(NFRKey[i])
        NFRKey[i] = normalize_alef_maksura_ar(NFRKey[i])
        NFRKey[i] = normalize_teh_marbuta_ar(NFRKey[i])
       for lst in tokens:
        if any(item in lst for item in NFRKey):
        NF_CF[c4] += 90
        NF_Score[c4] +=1
    c4 +=1
#********************H5********************
    for c5 in range(0, len(sentences)):
    if(len(tokens[c5]) >= 3):
        if (tokens[c5][0] == 'ان' and
        pos_tags[c5][1] == 'verb' and
        tokens[c5][1] != 'يكون' and
```

```
        tokens[c5][2] == 'النظام' or
        tokens[c5][2] == 'التطبيق' or
        tokens[c5][2] == 'البرنامج'):
            NF_CF[i] += 53.33
            NF_Score[c5] +=1
    if(len(tokens[c5]) >= 3):
        if (pos_tags[0] == 'verb' and #يجب
        tokens[c5][1] == 'ان' and
        pos_tags[c5][2] == 'verb' and
        tokens[c5][2] != 'يكون' and
        tokens[c5][3] == 'النظام' or
        tokens[c5][3] == 'التطبيق' or
        tokens[c5][3] == 'البرنامج'):
            NF_CF[c5] += 53.33
            NF_Score[c5] +=1
c5 +=1
#********************H6********************
DigPOS= ['part_neg']
 for lst in pos_tags:
    if any(item in lst for item in DigPOS):
        NF_CF[c6] += 82.66
        NF_Score[c6] +=1
    c6 +=1
#********************H7********************
#H7.1
for i in range(0, len(sentences)):

    if(len(tokens[i]) >= 5):
        if (tokens[i][0] == 'ان' and
        pos_tags[i][1] == 'verb' and
        pos_tags[i][2] == 'noun' and
        pos_tags[i][3] == 'adj' and
```

```
        (pos_tags[i][4] == 'prep' and
        pos_tags[i][5] == 'noun')):
            F_CF[i] += 88.33
            F_Score[i] +=1
        #H7.2
    if(len(tokens[i]) >= 6):
        if ( tokens[i][0] == 'أن' and
        pos_tags[i][1] == 'verb' and
        pos_tags[i][2] == 'noun' and
        tokens[i][3] == 'من' and
        pos_tags[i][4] == 'noun' and
        (pos_tags[i][5] == 'noun' or
        pos_tags[i][5] == 'adj' or
        pos_tags[i][5] == 'prep')):
            F_CF[i] += 88.33
            F_Score[i] +=1
    if(len(tokens[i]) >= 6):
        if ( tokens[i][0] == 'يجب' and
        tokens[i][1] == 'ان' and
        pos_tags[i][2] == 'verb' and
        pos_tags[i][3] == 'noun' and
        pos_tags[i][4] == 'adj' and
        pos_tags[i][5] == 'prep' ):
            F_CF[i] += 88.33
            F_Score[i] +=1

#********************H8*********************
x = ['يضيف', 'يسجل' , 'يعدل' , 'يحدث' , 'يعبئ ' ,'يفعل ', 'يعرض',
'يحذف','يحسب','ينشئ','يبحث']

for lst in tokens:
```

```
    if any(item in lst for item in x):
        F_CF[c8] += 88.33
        F_Score[c8] +=1
    c8 +=1
#********************H9********************
#conditinal sentences
for i in range(0, len(sentences)):
    if(len(tokens[i]) > 5):
        if (pos_tags[i][0] == 'conj' and #إذا ،لو
        pos_tags[i][1] == 'verb' and
        pos_tags[i][2] == 'noun' and
        pos_tags[i][3] == 'noun' and
        pos_tags[i][4] == 'verb_pseudo'):
            F_CF[i] += 76.66
            F_Score[i] +=1
#-------------------------------------------------------------------------
writer = csv.writer(outfile , escapechar=' ', quoting=csv.QUOTE_NONE)
for i in range(0, len(sentences)):
     if(F_Score[i] > NF_Score[i]):
        pred.append('FR')
    elif(F_Score[i] < NF_Score[i]):
        pred.append('NFR')
    elif(F_Score[i] == NF_Score[i]):
        if(F_CF[i] > NF_CF[i]):
            pred.append('FR')
        elif(F_CF[i] < NF_CF[i]):
            pred.append('FR')
        elif(F_CF[i] == NF_CF[i]):
            pred.append('Nan')
            Data.append([pred[i], sentences[i] ])
    writer.writerow(Data[i])
outfile.close()
```

```
print(classification_report(true, pred))
```